DBKDA 2013

The Fifth International Conference on Advances in Databases, Knowledge, and Data Applications

ISBN: 978-1-61208-247-9

January 27 - February 1, 2013

Seville, Spain

DBKDA 2013 Editors

Friedrich Laux, Reutlingen University, Germany

Lena Strömbäck, SMHI, Sweden

DBKDA 2013

Foreword

The Fifth International Conference on Advances in Databases, Knowledge, and Data Applications [DBKDA 2013], held between January 27th- February 1st, 2013 in Seville, Spain, continued a series of international events covering a large spectrum of topics related to advances in fundamentals on databases, evolution of relation between databases and other domains, data base technologies and content processing, as well as specifics in applications domains databases.

Advances in different technologies and domains related to databases triggered substantial improvements for content processing, information indexing, and data, process and knowledge mining. The push came from Web services, artificial intelligence, and agent technologies, as well as from the generalization of the XML adoption.

High-speed communications and computations, large storage capacities, and load-balancing for distributed databases access allow new approaches for content processing with incomplete patterns, advanced ranking algorithms and advanced indexing methods.

Evolution on e-business, ehealth and telemedicine, bioinformatics, finance and marketing, geographical positioning systems put pressure on database communities to push the 'de facto' methods to support new requirements in terms of scalability, privacy, performance, indexing, and heterogeneity of both content and technology.

We take here the opportunity to warmly thank all the members of the DBKDA 2013 Technical Program Committee, as well as the numerous reviewers. The creation of such a high quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated much of their time and efforts to contribute to DBKDA 2013. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

Also, this event could not have been a reality without the support of many individuals, organizations, and sponsors. We are grateful to the members of the DBKDA 2013 organizing committee for their help in handling the logistics and for their work to make this professional meeting a success.

We hope that DBKDA 2013 was a successful international forum for the exchange of ideas and results between academia and industry and for the promotion of progress in the fields of databases, knowledge and data applications.

We are convinced that the participants found the event useful and communications very open. We also hope the attendees enjoyed the charm of Seville, Spain.

DBKDA Chairs:

Friedrich Laux, Reutlingen University, Germany
Aris M. Ouksel, The University of Illinois at Chicago, USA
Lena Strömbäck, SMHI, Sweden
Serge Miranda, Université de Nice Sophia Antipolis, France

DBKDA 2013

Committee

DBKDA Advisory Chairs

DBKDA 2013 Technical Program Committee

Table of Contents

A New Representation of WordNet® using Graph Databases 1
Khaled Nagi

FIMIOQR: Frequent Itemsets Mining for Interactive OLAP Query Recommendation 9
Rym Khemiri and Fadila Bentayeb

Geometric Mean based Boosting Algorithm 15
Myoung Jong Kim

Primitive Operation Aggregation Algorithms for Improving Taxonomies for Large-Scale Hierarchical Classifiers 21
Kiyoshi Nitta

Data Management According to the Good Scientific Practice 27
Jan Potthoff, Marius Walk, and Sebastian Rieger

Parallel Processing of Multiple Graph Queries Using MapReduce 33
Song-Hyon Kim, Kyong-Ha Lee, Hyebong Choi, and Yoon-Joon Lee

On Using Object-Relational Technology for Querying LOD Repositories 39
Iztok Savnik

Using Social Network Information to Identify User Contexts for Query Personalization 45
Diogo Alves, Marcello Freitas, Thiago Moura, and Damires Souza

Finding Nearest Neighbors for Multi-Dimensional Data 52
Yong Shi and Marcus Judd

Parallel Genetic Algorithm Model to Extract Association Rules 56
Ahmad Taleb, Anwar Yahya, and Nasser Taleb

Optimization of SuperSQL Execution by Query Decomposition 65
Ria Mae Borromeo and Motomichi Toyama

Comparison of Subspace Projection Method with Traditional Clustering Algorithms for Clustering Electricity Consumption Data 71
Minghao Piao, Hyeon-Ah Park, Kyung-Ah Kim, and keun Ho Ryu

An Approach for Dynamic Materialization of Aggregates in Mixed Application Workloads 77
Stephan Muller, Carsten Meyer, and Hasso Plattner

About Summarization in Large Fuzzy Databases 87
Ines BenAli Sougui, Minyar Sassi Hidri, and Amel Grissa Touzi

Provenance Policies for Subjective Filtering of the Aggregated Linked Data 95
Tomas Knap

Merging Multidimensional Data Models: A Practical Approach for Schema and Data Instances 100
Michael Mireku Kwakye, Iluju Kiringa, and Herna L. Viktor

Considerations about Implementation Variants for Position Lists 108
Andreas Schmidt and Daniel Kimmig

Aspects of Append-Based Database Storage Management on Flash Memories 116
Robert Gottstein, Ilia Petrov, and Alejandro Buchmann

Mapping XML to Key-Value Database 121
Pavel Strnad, Ondrej Macek, and Pavel Jira

Typing XPath Subexpressions With Respect to an XML Schema 128
Yasunori Ishihara, Kenji Hashimoto, Atsushi Ohno, Takuji Morimoto, and Toru Fujiwara

Modeling a Data Storage System (DSS) for Seamless Real-Time Information Support from Information Manufacturing System (IMS) 134
Mohammad Shamsul Islam and Paul Young

Knowledge Enhanced Framework for designing e-Workflow Systems 143
Farhi Marir and John Ndeta

Modeling and Simulation Versions of Business Process using Petri Nets 150
Fatma Ellouze, Mohamed Amine Chaabane, Rafik Bouaziz, and Eric Andonoff

Parameterized Attribute and Service Levels Semantic Matchmaking Framework for Service Composition 159
Salem Chakhar

Sales Prediction with Parametrized Time Series Analysis 166
Michael Schaidnagel, Christian Abele, Fritz Laux, and Ilia Petrov

Enhancing Distributed Data Mining Performance by Multi-agent Systems 174
Maria del Pilar Angeles, Francisco Javier Garcia-Ugalde, and Jonathan Cordoba-Luna

An Implementation Model for Managing Data and Service Semantics in Systems Integration 182
Miklos Kasza, Vilmos Szucs, Vilmos Bilicki, Gabor Antal, and Andras Banhalmi

Explanations of Recommendations Considering Human Factors in Recommender Systems 190
Muhammet Tugberk Isyapar

Data and Provenance Management for Climate Effect Studies. Daption of Climate Data with Distribution Based Scaling for Hydrological Simulations 198
Lena Stromback, Kean Foster, and Jorgen Rosberg

Dr Warehouse - An Intelligent Software System for Epidemiological Monitoring, Prediction, and Research 204
Vladimir Ivancevic, Marko Knezevic, Milos Simic, Ivan Lukovic, and Danica Mandic

Shrinking Data Balls in Metric Indexes 211
Bilegsaikhan Naidan and Magnus Lie Hetland

An Epidemiological Data Mining Application Based on Census Databases 217
Joaquin Perez-Ortega, Crispin Zavala-Diaz, Alicia Martinez, Emmanuel Iturbide-Dominguez, Miguel Hidalgo-Reyes, and Adriana Mexicano-Santoyo

Clustering XML Data Streams by Structure based on SlidingWindows and Exponential Histograms 224
Mingxia Gao and Furong Chen

Feasibility of the Implementation of a UML to XML Evolution Architecture Using Eclipse as Technological Ecosystem 231
Beatriz Perez, Angel Luis Rubio, and Gloria Yanguas

Capturing the History and Change Structure of Evolving Data 235
George Papastefanatos, Yannis Stavrakas, and Theodora Galani

Business Intelligence in an Educational Landscape 242
Marcio Rodrigo Teixeira and Mehran Misaghi

BA: Lean Manufacturing Indicators Applied to HR Applications - An Implementation Study 246
Robson Thanael Poffo and Mehran Misaghi

A New Representation of WordNet® using Graph Databases

Khaled Nagi
Dept. of Computer and Systems Engineering
Faculty of Engineering, Alexandria University
Alexandria, Egypt
khaled.nagi@alexu.edu.eg

***Abstract*— WordNet® is one of the most important resources in computation linguistics. The semantically related database of English terms is widely used in text analysis and retrieval domains, which constitute typical features, employed by social networks and other modern Web 2.0 applications. Under the hood, WordNet® can be seen as a sort of read-only social network relating its language terms. In our work, we implement a new storage technique for WordNet® based on graph databases. Graph databases are a major pillar of the NoSQL movement with lots of emerging products, such as Neo4j. In this paper, we present two Neo4j graph storage representations for the WordNet® dictionary. We analyze their performance and compare them to other traditional storage models. With this contribution, we also validate the applicability of modern graph databases in new areas beside the typical large-scale social networks with several hundreds of millions of nodes.**

Keywords-WordNet®; semantic relationships; graph databases; storage models; performance analysis.

I. Introduction

WordNet® [1] is a large lexical database of English terms and is currently one of the most important resources in computation linguistics. Several computer disciplines, such as information retrieval, text analysis and text mining, are used to enrich modern Web 2.0 applications; typically, social networks, search engines, and global online marketplaces. These disciplines usually rely on the semantic relationships among linguistic terms. This is where WordNet® comes to action.

A parallel development over the last decade is the emergence of NoSQL databases. Certainly, they are no replacement for the relational database paradigm. However, Web 2.0 builds a rich application field for managing billions of objects that do not have the regular and repetitive pattern suitable for the relational model. One major type of NoSQL databases is the *graph database* model. Since social networks can be easily modeled as one large graph of interconnected users, they can be the killer application for graph databases.

However, little to no work has been done to investigate the use of graph database management systems in moderate sized databases. Of course, the database has to be relationship-rich for the implementation to make sense. In our work, we implement a new storage technique for WordNet® based on Neo4j [2]; currently, a leading graph database. WordNet® dictionary has several characteristics that promote our proposition: *it is used in several modern Web 2.0 applications*, such as social networks; *it is has a moderate size of datasets*; and *traversing the semantic relationship graph is a common use case*.

Since the modeling and benchmarking experiences of these new graph databases are not as established as in the relational database model, we implement two variations and conduct several performance experiments to analysis their behavior and compare them to the relational model.

The rest of the paper is organized as follows. Section II provides a background on WordNet® and its applications as well as a brief survey on graph database technology. Our proposed system is presented in Section III. Section IV contains the results of our performance evaluation and Section V concludes the paper and presents a brief insight in our future work.

II. Background

A. *WordNet®*

The WordNet® project began in the Princeton University Department of Psychology, and is currently housed in the Department of Computer Science. WordNet® is a large lexical database of English [1]. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. A synset contains a brief definition (gloss). Synsets are interlinked by means of conceptual-semantic and lexical relations. WordNet® labels the semantic relations. The most frequently encoded relation among synsets is the super-subordinate relation (also called hyperonym, hyponym or IS-A relation). Other semantic relations include meronyms, antonyms, and holonyms. The majority of the WordNet®'s relations connect words from the same part of speech (POS). Currently, WordNet® comprises 117,000 synsets and 147,000 words.

Today, WordNet® is considered to be the most important resource available to researchers in computational linguistics, text analysis, text retrieval and many related areas [3]. Several projects and associations are built around WordNet®.

The Global WordNet Association [4] is a free, public and non-commercial organization that provides a platform for discussing, sharing and connecting wordnets for all languages in the world. The Mimida project [5], developed by Maurice Gittens, is a WordNet-based mechanically-generated multilingual semantic network for more than 20 languages based on dictionaries found on the Web. EuroWordNet [6] is a multilingual database with wordnets for several European languages (Dutch, Italian, Spanish, German, French, Czech and Estonian). It is constructed according to the main principles of Princeton's WordNet®. One of the main results of the European project that started in 1996 and lasted for 3 years is to link these wordnets to English WordNet® and to provide an Inter-Lingual-Index to connect the different wordnets and other ontologies [7]. MultiWordNet [8], developed by Luisa Bentivogli and others at ITC-irst, is a multilingual lexical database. In MuliWordNet, the Italian WordNet is strictly aligned with the Princeton WordNet®. Unfortunately, it comprises a small subset of the Italian language with 44,000 words and 35,400 synsets. Later on several projects; such as ArchiWN [9], attempt to integrate WordNet with domain-specific knowledge.

RitaWN [10], developed by Daniel Howe, is an interesting library built on WordNet®. It provides simple access to the WordNet ontology for language-oriented artists. RitaWN provides semantically related alternatives for a given word and POS (e.g., returning all synonyms, antonyms, hyponyms for the noun "cat"). The library also provides distance metrics between ontology terms, and assigns unique IDs for each word sense/pos.

Several projects aim at providing access to the WordNet® native dictionary. For example, JWNL [11] provides a low-level API to the data provided by the standard WordNet® distribution. In its core, RitaWN uses JWNL to access the native file-based WordNet® dictionary. Other projects, such as WordNetScope [12], WNSQL [13], and wordnet2sql® [14], provide a relational database storage for WordNet®.

B. *Graph Databases*

NoSQL databases are older than relational databases. Nevertheless, their renaissance came first with the emergence of Web 2.0 during the last decade. Their main strengths come from the need to manage extremely large volumes of data that are collected by modern social networks, search engines, global online marketplaces, etc. For this type of applications, ACID (Atomicity, Consistency, Isolation, Durability) transaction properties [15] are simply too restrictive. More relaxed models emerged such as the CAP (Consistency, Availability and Partition Tolerance) theory or eventually consistent [16], which in general means that any large scale distributed DBMS can guarantee for *two* of *three* aspects: *Consistency*, *Availability*, and *Partition tolerance*. In order to solve the conflicts of the CAP theory, the BASE consistency model (Basically, soft state, eventually consistent) was defined for modern applications [16]. In contrast to ACID, BASE concentrates on availability at the cost of consistency. BASE adopts an optimistic approach, in which consistency is seen as a transitional process that will be *eventually* reached. Together with the publication of Google's BigTable and Map/Reduce frameworks [17], dozens of NoSQL databases emerged. A good overview of existing NoSQL database management systems can be found in [18].

Mainly, NoSQL database systems fall into four categories:

- Key-value systems,
- Column-family systems,
- Document stores, and
- Graph databases.

Graph databases have a long academic tradition. Traditionally, research concentrated on providing new algorithms for storing and processing very large and distributed graphs. These research efforts helped a lot in forming object-oriented database management systems and later XML databases.

Since social networks can be easily viewed as one large graph of interconnected users, they offer graph databases the chance for a great comeback. Since then, the whole stack of database science was redefined for graph databases. At the heart of any graph database lies an efficient representation of entities and relationships between them. All graph database models have, as their formal foundation, variations on the basic mathematical definition of a graph, for example, directed or undirected graphs, labeled or unlabeled edges and nodes, hypergraphs, and hypernodes [19]. For querying and manipulating the data in the graph, a substantial work focused on the problem of querying graphs, the visual presentation of results, and graphical query languages. Old languages such as G, G++ in the 80s [20], the object-oriented Pattern Matching Language (PaMaL) in the 90s [21], through Glide [22] in 2002 appeared. G is based on regular expressions that allow simple formulation of recursive queries. PaMaL is a graphical data manipulation language that uses patterns. Glide is a graph query language where queries are expressed using a linear notation formed by labels and wildcards. Glide uses a method called GraphGrep [22] based on sub-graph matching to answer the queries.

However, modern graph databases prefer providing traversal methods instead of declarative languages due to its simplicity and ease use within modern languages such as Java. Taking Neo4j as example, when a `Traverser` is created, it is parameterized with two evaluators and the relationship types to traverse, with the direction to traverse each type. The evaluators are used for determining for each node in the set of candidate nodes if it should be returned or not, and if the traversal should be pruned (stopped) at this point. The nodes that are traversed by a `Traverser` are each visited exactly once, meaning that the returned iterator of nodes will never contain duplicate nodes [2].

Several systems such as Neo4j [2], InfoGrid [23], and many other products are available for research and commercial use today. Typical uses of these new graph database management systems include social networks, GIS, and XML applications. However, they did not find application in moderate sized text analysis applications or relationship mining.

III. Proposed Implementation

Fig. 1 provides an overview of the proposed implementation. RitaWN [10] provides synonyms, antonyms, hypernyms, hyponyms, holonyms, meronyms, coordinates, similars, nominalizations, verb-groups, derived-terms glossaries, descriptions, support for pattern matching, soundex, anagrams, etc. In Fig. 1, RitaWN is represented by an arbitrary client in this domain which sends semantic inquiries and receives the results as a list of related terms. In the actual RitaWN, the library wraps Jawbone/JWNL [11] functionality for Java processing; which, in turn, accesses the native WordNet® dictionary.

In order to separate the storage layer from the logic, we extract a `RiWordNetIF` Java interface. The interface defines methods to return semantically related words. The methods are categorized into 4 groups:

- Attribute inquiries: these methods return single attribute values for a given word, such as `String getBestPos(String w)` and `boolean isNoun(String w)`.
- Semantic relationships inquiries: in this set, methods return all semantically related words for a given word and POS, such as `String[] getHolonyms(String w, String pos)` and `String[] getHypernyms(String w, String pos)`. In our system, we define eight such methods.
- Relationship tree inquiries: in this set of methods, the library returns the whole path from the first synset for a given word and POS to the root word. Typical root words in WordNet® are “Entity” or “Object”. In our implementation, we have `String[] getHyponymTree(String w, String pos)` and `String[] getHypernymTree(String w, String pos)`; which basically trace back `getHyponym(String w, String pos)` and `getHypernym(String w, String pos)` respectively to the root word.
- Common parent inquiries: methods of this group find a common semantic path between two words in a POS subnet by traversing the WordNet® synset graph. For example, the method `String[] getCommonParent(String w1, String pos, String w2)` finds the following path dog : canis familiaris : domestic dog → domestic animal : domesticated animal → animate being : beast : animal for the nouns “dog” and “animal”. Traversal is done based on a Depth First Search algorithm with a slight adaptation to stop traversing whenever one of the synsets of the sink term `w2` is reached.

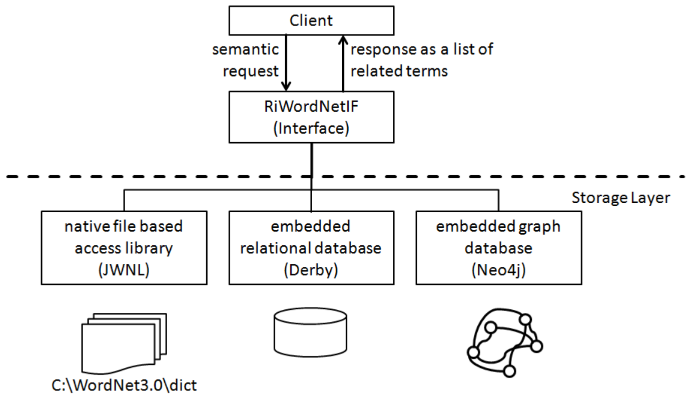

Client
semantic request
response as a list of related terms
RiWordNetIF (Interface)
Storage Layer
native file based access library (JWNL)
embedded relational database (Derby)
embedded graph database (Neo4j)
C:\WordNet3.0\dict

Figure 1. Architecture of the proposed system.

A. *Storage Layer*

In the storage layer, we provide four different representations for the WordNet® dictionary as described in the following subsections.

1) *File-based Storage*

In its original implementation, RiTa.WordNet uses the JWNL [11] library to directly browse the native dictionary provided by a standard WordNet® installation. As will be shown later, this implementation has the worst performance. We use it for validation purposes for the other three implementations.

2) *Relational Database Storage*

We use a database model similar to the one used in [14]. Fig. 2 illustrates a UML class diagram for the relevant classes. The `words` entity has a `wordid` as a primary key, the `lemma` definition and the different `POSs` are coded as string with the best POS as the first character of the string. Similarly, the `synsets` entity holds all WordNet® synsets, their `POS`, and definition. The primary key is `synsetid`. The many-to-many relationship between words and synsets is modeled by the `senses` entity. It contains the foreign keys `wordid` and `synsetid`. `Synsets` are related to each other via the `semlinks` entity. `Synset1id` points to the `from` direction and `Synset2id` to the `to` direction. The types of semantic links are defined by `linkid` which is a foreign key to the `linktype` entity. All types of links are listed in the `linktype` entity. We choose Apache Derby [24] as the database management system to hold this data model. Apache Derby is part of the Apache Group. It gained a good reputation and a high spread for applications requiring embedded relational DBMS. It is distributed as a java jar file to be added to the classpath of the application. It also comes as a stand-alone version.

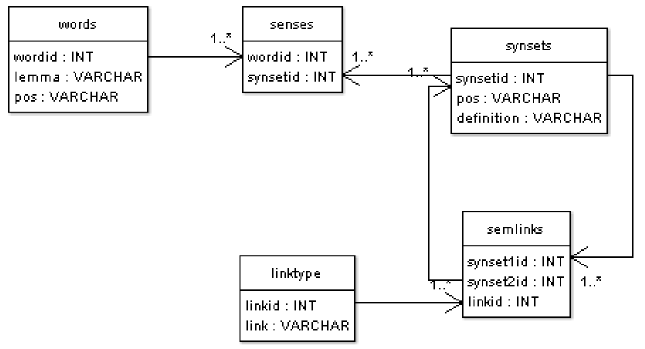


Figure 2. UML class diagram for the relational database.

3) Graph Database Storage

In our proposed work, we model the WordNet® as a graph database. An object diagram is illustrated in Fig. 3. We have two types of nodes: `words` (illustrated as ellipses) and `synsets` (illustrated as hexagons). The attributes of a word are a lemma and the different `POSs`, which are coded as a string with the best POS as the first character of the string. The `synset` has a property `definition`. There exists a bi-directional relation `Rel_sense` between `words` and `synsets`. The attribute `pos` of the relation indicates the POS associated with the sense. `Synsets` are interconnected by directed relations. These relationships `Rel_SemanticLink` carry the `type` of the link in the attribute type. For example, in Fig. 3, word *`w1`* has one sense as a noun with link to `sysnset` *`sa`* and two senses as verbs for `synsets` *`sc`* and *`sd`*. `Synset` *`sa`* has two hyponyms *`sb`* and *`se`* by following the relationships `Rel_SemanticLink` with type "hyponym". *`w4`* has one sense *`sb`* as a noun. *`w2`* and *`w3`* – as nouns - share the same `synset` *`se`*. *`w5`* has only one sense as a verb which is *`sc`*. So, if `getHoponyms("w1", "n")` is called, the result will be *`w2`*, *`w3`*, and *`w4`*.

4) Graph Database Storage with Extra Directly Derived Relationships

In the RiTa.WordNet application scenario, we expect lots of inquiries about semantically related words (e.g., hyponyms, synonyms, meronyms, etc.). Synsets are mainly the means to return the semantically related words. At the same time, the application is typically read-only and represents a good example for a wide range of read-only (or low-update/high-read) applications. The graph database is only updated with the release of a new WordNet® dictionary. This motivates us to augment the design mentioned in the previous section with the derived semantic relationships between words and not only synsets. The idea is similar to materialized views known in relational databases. the result of semantic relationship inquiries (e.g., `getHyponyms()`, `getSynonyms()`, `getMeronyms()`, etc.) is generated by traversing only one relationship for each result word. We intuitively expect a quicker response time at the cost of a high storage volume since the connectivity of the graph is highly increased.

In terms of implementation, these relationships are identified through the relationship type. Fig. 4 illustrates the derived relationships for the example in Fig. 3. Only the relationship of type `Rel_Hyponym` for noun POS of word *`w1`*; namely, *`w2`*, *`w3`*, and *`w4`* is drawn. For more complex inquiries of category "relationship tree" and "common parent", a combination of original and derived relationships are used in the traversal.

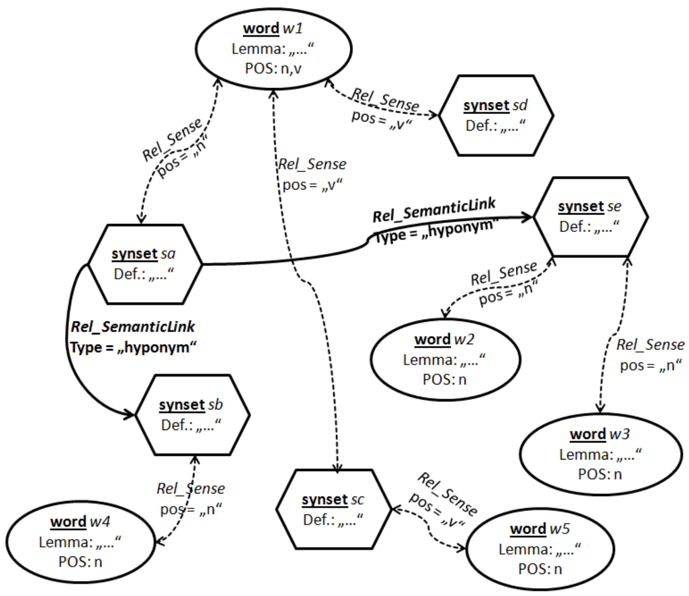


Figure 3. Object diagram for the proposed WordNet® graph database storage.

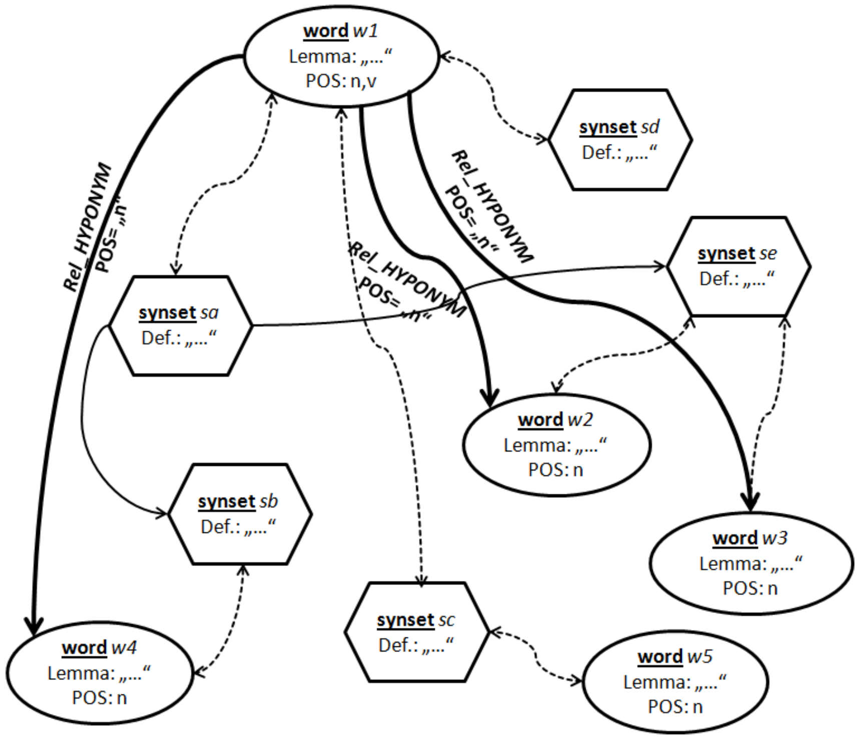


Figure 4. Object diagram with the extra derived relationships.

IV. Performance Evaluation

In order to evaluate the performance of our proposed system, we provide *four* implementations for the Java interface `RiWordNetIF` mentioned in Section III. The implementations are file-based storage, relational DBMS using Apache Derby, the graph database using Neo4j, and a second implementation using the directly derived relationships also using Neo4j.

It is important to notice that the purpose of this evaluation is to give a general impression on the performance impact and not to give concrete benchmarking figures. For sure, the optimization of all DBMS implementations; such as using indices or even exchanging the DBMS itself versus using future versions of Neo4j might lead to different results. *We would be satisfied when our proposed solution provides slightly better results than relational DBMS.* It is also clear that in-memory databases and large caching mechanisms will outperform all implementations. But we rule them out assuming memory size restrictions.

We develop a simple performance evaluation toolkit around these four implementations. A workload generator sends inquiries to all back-ends. The inquiries are grouped into four categories, as mentioned in Section III. The workload generator submits the inquiries in parallel to the application with each inquiry executing in a separate thread.

The input for the inquiry is chosen at random from an input file containing WordNet® words and their associated best POS. In case of `getCommonParent()`, another input file is used, which contains tuples of somehow related words, together with their common POS (e.g., "tiger", "cat", and "noun"). The tuples are chosen carefully to yield paths of different lengths.

The performance of the system is monitored using a performance monitor unit that records the response time of each inquiry and the number of inquiries performed by each thread in a regular time interval.

A. Input Parameters and Performance Metrics

The number of concurrent inquiry threads is increased from 1 to 50. Each experiment executes on each backend for 5 minutes in order to eliminate any transient effects. The experiments are conducted for each type of inquiries separately.

In all our experiments, we monitor the system *response time* in terms of micro-seconds per operation from the moment of submitting the inquiry till receiving the result.

We also monitor the system *throughput* in terms of inquires per hour for each thread.

B. System Configuration

In our experiments, we use an Intel CORE™ i7 vPro 2.7GHz processor, 8 GB RAM and a Solid State Drive (SSD). The operating system is Windows 7 64-bits. We use JDK 1.6.0, Neo4j version 1.6 for the graph database engine, embedded Derby™ version 10.7.1.1 for the SQL backend, JWNL library version 1.4 [11] for file system based storage.

C. *Experiment Results*

The performance evaluation considers all four types of inquiries:

- Attribute,
- Semantic relationships,
- Relationship trees, and
- Common parent

for the four back-end implementations.

We drop plotting the results of the native file system-based implementation from our graphs, although it is the only available implementation previous to this work. The reason behind this is that the results are far worse than the other implementations. The difference in most case is more than *one order of magnitude*.

1) *Attribute Inquiries*

In this set of experiments, the inquiries sent by the workload generator comprise attribute inquiries only. Both response time, illustrated in Fig. 5, and throughput, illustrated in Fig. 6, degrade gracefully with the increase in number of threads while having good absolute values. Remarkably, the simple Neo4j implementation (without the extra directly derived relationships) has a 20% better response time than the other two implementations, while the full blown Neo4j implementation has a 40% decrease in system throughput. The reason for that is the attribute inquiries are mainly affected by the node (or tuple in case of relational databases) retrieval and caching. No relationship traversal is done and hence the Neo4j only suffers from its large database size especially with the augmented directly derived relationships (see Section IV.D). In summary, this set of experiments demonstrates that the caching mechanisms of graph databases are in general as good as the relational databases and that simple operations without graph traversals are not underprivileged in this environment.

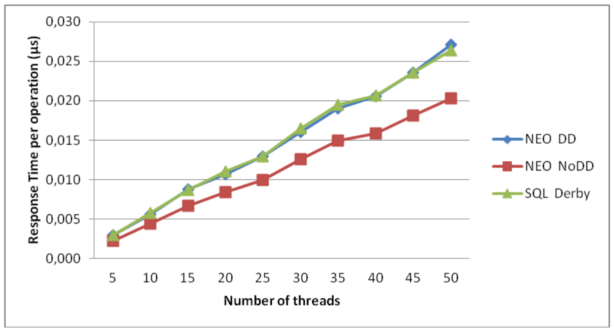


Figure 5. Response time for attribute inquiries.

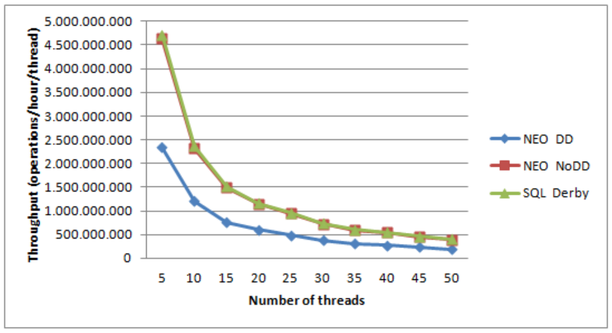


Figure 6. Throughput for attribute inquiries.

2) *Semantic Relationship Inquiries*

In this set of experiments, the explicit storage of semantic relationships shows its benefit. The results are retrieved by traversing one relationship only, in contrast to 3 for the simple implementation and several joins in the relational database implementation. The response time, as illustrated in Fig. 7 is enhanced by approx. 50% for all number of threads when compared to SQL Derby and 30% by adding these directly derived relationships to a simple Neo4j implementation. However, all three back-ends behave identically when it comes to throughput as illustrated in Fig. 8. The absolute values are far below those of the simple attribute inquiries described in the previous section which is expected due to the complexity of these inquiries as compared to attribute inquiries. In case of response time, it is almost 10 times higher than the previous set of experiments. The same applies to the throughput, which is lower by a factor of 10 as well.

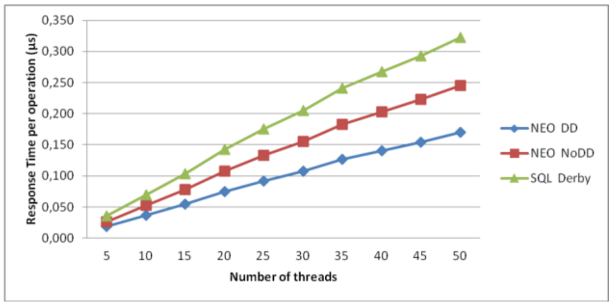


Figure 7. Response time for semantic relationship inquiries.

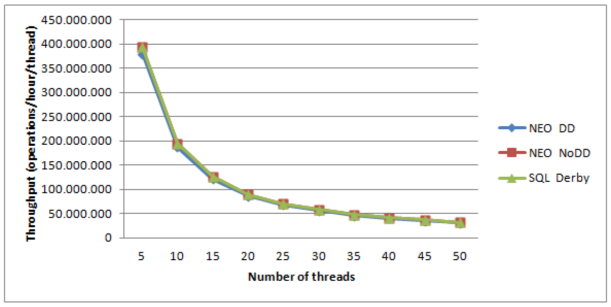


Figure 8. Throughput for semantic relationship inquiries.

3) *Relationship Tree Inquiries*

The operations of this set of experiments are more complex than the previous ones. This explains the drop in absolute values of the response time and throughput, illustrated in Fig. 9 and Fig. 10, respectively when compared to the previous experiment. This time the degradation factor is only 4. Yet, the system behavior remains the same. The response time of Neo4j with the directly derived relationships is half that's of the SQL implementation. Even without the extra relationships, the response time Neo4j is 25-30% better than the relational model. Here, again, the throughput, illustrated in Fig. 10, for all three implementations is the same. The equality of the throughput performance index of Derby and the Neo4j implementation, despite the short response time of the later, is an indication that the internal pipeline capabilities of Neo4j is *not* as good as that of the relational model.

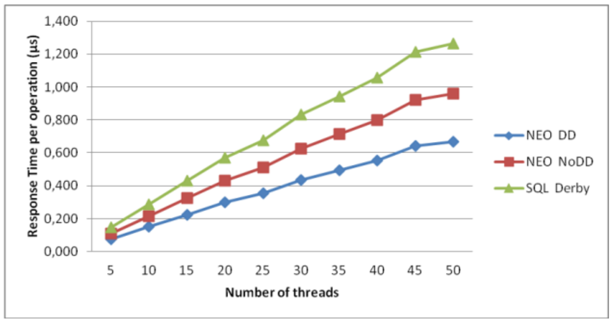


Figure 9. Response time for relationship tree inquiries.

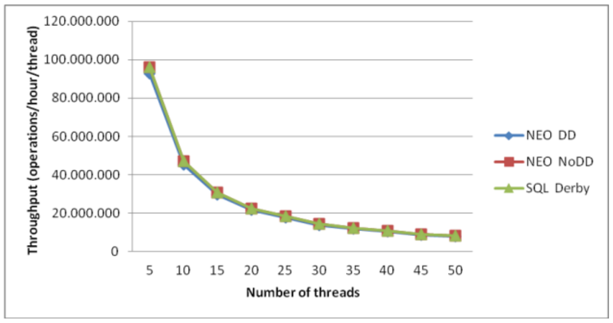


Figure 10. Throughput for relationship tree inquiries.

4) *Common Parent Inquiries*

The inquiries for this set of experiments are the most complicated among all experiments. Yet, this is a very common use case in social networks. For example, in XING [25], the user can always see all paths of relationships leading from the user to any arbitrary user in the network. No wonder here that Neo4j implementations outperform the SQL Derby implementation (and the file system implementation which seems to be not able to handle all the running threads) in requesting depth first searches of the semantic network of WordNet®. Again, Fig. 11 illustrates the extreme superiority of graph database, especially with the addition of the extra relationships. The response time is also enhanced by 45% and 30% with and without directly derived relationships, respectively. The throughput, illustrated in Fig. 12, holds its trend across all experiments of being almost the same for the three implementations (and omitting the file system implementation of course, whose values cannot be plotted with the same scale next to their counterparts).

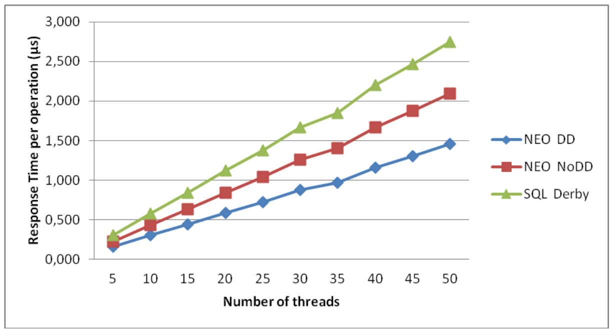


Figure 11. Response time for common parent inquiries.

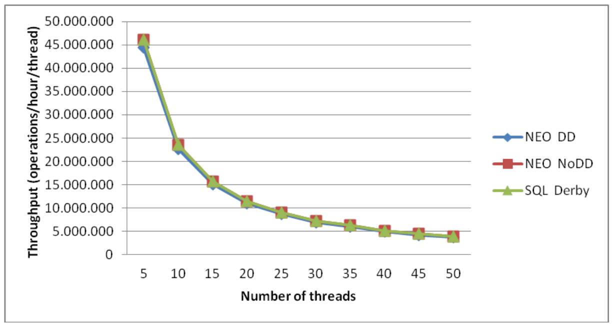


Figure 12. Throughput for common parent inquiries.

D. *Storage Requirements*

Performance in terms of good response time comes with its price. Fig. 13 illustrates the storage requirements for all four implementations. The SQL Derby and the normal Neo4j implementation occupy slightly more than double the original size of the WordNet® file-based dictionary. The redundant relationships account for more than 350 MB, making the size of the graph database 12 times larger than the file-based dictionary taken as a reference point. The good side of this particular application scenario is the absolute size of the back-ends is affordable by any desktop application.

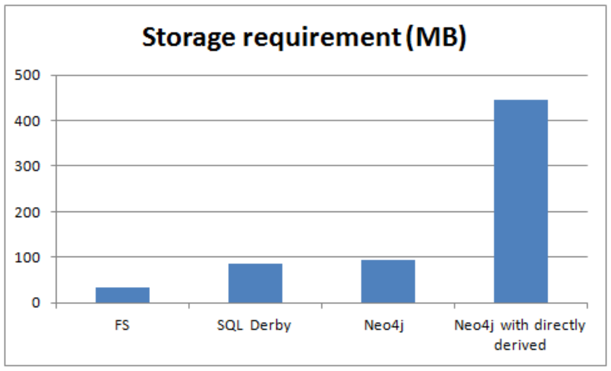


Figure 13. Storage for each backend implementation.

V. Conclusion and Future Work

In this paper, we presented two Neo4j graph storage representations for the WordNet® dictionary. We use Ri.WordNet as a typical client application that submits semantic inquiries discovering the relationships between English terms. We divide the inquiries into 4 categories depending on the complexity of their operations. Our performance analysis demonstrates that graph databases yield much better results than traditional relational databases in terms of response time even under extreme workloads thus speaking for their promised scalability. We also show that storing directly derived relationships can improve the performance by factors of 2. This redundancy has its price in terms of storage requirements, which is acceptable due to the moderate size of the database with 117,000 synsets and 147,000 terms and the read-only nature of this small scale social network.

One important contribution of this work is that it opens the door for new application areas for NoSQL databases (in this case the Neo4j graph database), namely smaller read-intensive database applications, in contrast to typical applications of the NoSQL in large scale Web 2.0 such as social networks.

Yet, this is only the beginning. In the future, we plan on benchmarking other graph database providers, such as InfoGrid [23]. We also plan on migrating several research done on relationship mining to work on graph database back-ends. If the benchmarking experiments show promising results, this will open the door for the application of graph databases in OLAP applications.

References

[1] Fellbaum, C.: WordNet and wordnets. In: Brown, Keith et al. (eds.) Encyclopedia of Language and Linguistics, Second Edition, pp. 665--670. Elsevier, Oxford , 2005.

[2] Neo4j. The World's Leading Graph Database, http://www.neo4j.org [retrieved: November, 2012].

[3] Voorhees, E.: Using WordNet for Text Retrieval. In: Fellbaum, C. (ed.) WordNet An Electronic Lexical Database, 0-262-06197-X. MIT Press, 1998.

[4] The Global WordNet Association, http://www.globalwordnet.org [retrieved: November, 2012].

[5] Mimida: A mechanically generated Multilingual Semantic Network, http://gittens.nl/gittens/topics/SemanticNetworks.html [retrieved: November, 2012].

[6] Vossen, P.: EuroWordNet: a multilingual database for information retrieval. In: Proceedings of the DELOS workshop on Cross-language Information Retrieval. Zürich , 1997.

[7] Vossen, P., Peters, W., and Gonzalo, J.: Towards a Universal Index of Meaning. In: Proceedings of the ACL-99 Siglex workshop, Maryland, 1999.

[8] Pianta, E., Bentivogli, L., and Girardi, C.: MultiWordNet: developing an aligned multilingual database. In: Proceedings of the First International Conference on Global WordNet, Mysore, India, 2002.

[9] Bentivogli, L., Bocco, A., and Pianta, E.: ArchiWordNet: Integrating WordNet with Domain-Specific Knowledge. In: Proceedings of the Second Global WordNet Conference, pp. 39—46, Brno, Czech Republic, 2004.

[10] RiTa.WordNet: a WordNet library for Java/Processing, http://www.rednoise.org/rita/wordnet/documentation [retrieved: November, 2012].

[11] Java WordNet Library, http://sourceforge.net/projects/jwordnet [retrieved: November, 2012].

[12] WordNetScope, http://wnscope.sourceforge.net [retrieved: November, 2012].

[13] WordNetSQL, http://wnsql.sourceforge.net [retrieved: November, 2012].

[14] wordnet2sql, http://www.semantilog.org/wn2sql.html [retrieved: November, 2012].

[15] Gray, J. and Reuter, A.: Transaction Processing: Concepts and Techniques, Morgan Kaufmann, 1983.

[16] Brewer, E.: Towards Robust Distributed Systems. In: ACM Symposium on Principles of Distributed Computing, Keynote speech, 2000.

[17] Chang, F., Dean, J., Ghemawat, S., Hsieh, W.C., Wallach, D.A., Burrows, M., Chandra, T., and Fikes, A., Bigtable: A distributed storage system for structured data. In: the Seventh Symposium on Operating System Design and Implementation. Seattle, 2006.

[18] Edlich, S., Friedland, A., Hampe, J., and Brauer, B: NoSQL: Introduction to the World of non-relational Web 2.0 Databases (In German) NoSQL: Einstieg in die Welt nichrelationaler Web 2.0 Datenbanken. Hanser Verlag, 2010.

[19] Angles, R. and Gutierrez, C.: Survey of Graph Database Models. In: ACM Computing Surveys, Vol. 40. No. 1 Article 1, 2008.

[20] Cruz, I.F., Mendelzon, A.O., and Wood, P.T.: A graphical query language supporting recursion. In: Proceedings of the Association for Computing Machinery Special Interest Group on Management of Data, pp. 323—330. ACM Press, 1987.

[21] Gemis, M. and Paredaens, J.: An object-oriented pattern matching language. In: Proceedings of the First JSSST International Symposium on Object Technologies for Advanced Software, pp. 339–355. Springer-Verlag, 1993.

[22] Giugno, R. and Shasha, D.: GraphGrep: A fast and universal method for querying graphs. In: Proceedings of the IEEE International Conference in Pattern recognition, 2002.

[23] InfoGrid: The Web Graph Database, http://infogrid.org/trac [retrieved: November, 2012].

[24] Apache Derby, http://db.apache.org/derby [retrieved: November, 2012].

[25] XING das professionelle Netzwerk, http://www.xing.com [retrieved: November, 2012].

FIMIOQR: Frequent Itemsets Mining for Interactive OLAP Query Recommendation

Rym Khemiri, Fadila Bentayeb
ERIC Laboratory, University of Lyon, Lumière Lyon2
5 av. P. Mendès-France 69676 Bron Cedex Lyon, France
{Rym.Khemiri, Fadila.Bentayeb}@univ-lyon2.fr

***Abstract*—We propose in this paper an interactive query recommendation system, namely FIMIOQR. It is designed to help OLAP (On-line Analytical Processing) users in their decision query writing task based on both a set of selected measures and decision queries log file. Our FIMIOQR system is designed to discover associations from decision queries log file. For this end, we use association rules method to extract frequent itemsets from dimensions attributes according to user selected set of measures. This allows end users in OLAP systems to write relevant queries guided by an interactive recommending system and helps them to meet their analysis objectives. In addition, we propose a tool for the automatic implementation of FIMIOQR which provides a visual interface to OLAP users which helps them to write their queries step by step in an interactive way. We also carried out some experimental tests to evaluate our system. The experimental evaluation proves our FIMIOQR framework is efficient in term of recommendation quality.**

***Index Terms*—Interactive Recommendation; Data warehouse; Decision Query; Measure; Dimension attribute; Frequent Itemsets Mining; OLAP; Data warehouse**

I. Introduction

End users in OLAP systems tend to achieve the same goal for obtaining valuable and useful information out of data warehouse. However, OLAP is characterized by decision queries that are often very complex and involve a lot of aggregations. Due to the constantly and rapidly growth of data volumes, manipulation and analysis complexity also increases. In such situation, OLAP users would quite benefit from assistance for this task. This assistance may be accomplished through recommendation process.

Recommendation is a means of meeting user's needs more efficiently, making interactions faster and easier and, consequently, increasing user satisfaction. This assumption constitutes the starting point used in this paper to present a new recommendation approach over data warehouses.

In OLAP context, query recommendation approaches mostly focused on recommending alternative queries with close search intent to the original query. In this case, recommended queries are existing queries. However, the recommendation of only alternative existing queries may generate less interactivity with the user and does not create new decision queries.

Furthermore, there is a lack of a framework that assists the users while querying data warehouse. We believe that such a framework would be interactive during query writing and consequently increases user implication in the exploration task. It allows him/her to construct his/her decision query step by step (incrementally).

Query logs usually contain a sequence of SQL queries that show the action flows of users, their preference, their interests, and their behaviours during the action. In this paper, we aim to assist the user in formulating accurate queries to express his/her analysis needs. We propose an interactive real-time assistance process which aims to give users opportunities to refine their queries by suggesting queries completions.

Decision queries on which we are interested in this paper are in the form "SELECT ... FROM ... WHERE ... GROUP BY CUBE (ROLLUP)". Assuming that the relevance of a recommended query is strongly correlated to the usage frequency of the corresponding attributes within a given workload, the search for frequent itemsets [1] appeared well adapted to highlight this correlation and to facilitate query recommendation. Our tool parses the transaction log file (set of queries executed by the DBMS) and groups queries respecting the same measure(s) to build a context for mining frequent itemsets. This context connects queries from the input workload to the query attributes. The output frequent itemsets are sets of attributes forming a configuration of candidate recommendations. Finally, recommendation strategy can be applied to select the relevant attributes to effectively suggest from within this configuration.

Besides attributes (qualitative data) in a decision query, there is at least one numeric measure (quantitative data) that provide the object of analysis. Thereby, after asking user for his/her object of analysis (measure or combination of measures), recommendations during query writing will focus on query attributes. Queries from the transaction log constitute a workload that is treated by an SQL query analyzer. Thus, the SQL query analyzer extracts all the attributes. Then, we build a "query-attributes" matrix, the rows of which represent the workload queries, and the columns represent attributes. The role of this matrix is to link each attribute to the workload queries it appears in. This matrix represents the extraction context for frequent itemsets. To compute these frequent itemsets, we applied the Close algorithm [2].

Our approach goal is to assist OLAP user to accomplish his/her decision query writing by suggesting him/her possible candidate query attributes with respect to both their appearance in different *Clauses* (SELECT, WHERE, GROUP BY, etc.) and the selected measures. Consequently, the user can construct progressively his/her decision query by selecting attributes among different proposed suggestions by our approach. Thus, the new obtained query can be submitted by the user to the data warehouse. In order to validate our recommendation approach, we have implemented the recommendation framework called FIMIOQR (Frequent Itemsets Mining for Interactive OLAP Query Recommendation) with Java using the integrated development environment Netbeans [3]. The experimental evaluation proves our framework is efficient in term of recommendation quality.

The remainder of this paper is structured as follows. Section II presents related works regarding query recommendation. In Section III, we present our query recommendation approach. In Section IV, we present an example to motivate our approach. Implementation features are described in Section V with some preliminary experimental results. Finally, conclusions are given in Section VI, together with a summary of our expected future work.

II. Related Works

Recently, recommendation systems have obtained the attention of research society in both database and data warehouse fields. In fact, items to recommend are queries. We find in literature two kinds of recommendations (1) alternative entire queries proposition and (2) assistance to query construction through query completions.

In the database field, based on results of each query, Stefanidis et al. recommend additional results called “You May Also Like” or YMAL results which might be of user’s potential interests [4]. YMAL appraoch exploits the history of previously submitted queries to the database system, e.g. by using query logs. Recommendations that are generated can be either query-based YMAL results (similar to content-based recommendations), either user-based YMAL results (similar to collaborative recommendations).

Another approach for recommending queries in the database field creates automatically join query recommendations based on input and output specifications [5]. This approach analyses query log and extracts joins from previous submitted queries that are used to generate recommendations for the current user. After presenting the conceptual framework and its instantiation in [6], Chatzopoulou et al. came up with a system named *QueRIE* for personalized query recommendations (full SQL queries). To generate recommendations, *QueRIE* first constructs the summary for each user, then generates a “predicted” summary for the users and finally generates recommendation queries based on “predicted” summary. Finally, based on the analysis of query log, Khoussainova et al. introduced the *Snipsuggest* framework. *Snipsuggest* assists users in composing complex queries by recommending a set of additions to a specific clause in a partially stated SQL query [7].

In OLAP context, the first approach to be considered is based on providing query recommendations to the user by means of User Preference Analysis (RUPA) [8], [9]. In RUPA approach, OLAP analyses are represented using a graph-based model. In fact, both of user profile and current query are expressed by means of trees. Then, a Tree Matching Algorithm is applied to compare the two trees.

More recently, Golfarelli et al. propose an approach where preferences are used to annotate the query. They defined an algebra that allows formulating preferences on attributes, measures and hierarchies [10]. The used technique consists in personalizing a query by dealing with a sub-query of the current query. In this case, the recommended query is the sub-query which returns a non empty preferred result.

In this section, we reviewed the current approaches for query recommendation for databases and data warehouses. It seems that neither solution by now assist decision query construction. We have to come up with better, more meaningful solution to recommend query completions to the user instead of whole queries. We present a comparative table (table I) confronting the panoply of the proposed approaches. We define some criteria that we consider relevant to study exactly the recommendation approach.
Recommendation type: the recommendation system may be content-based or collaborative. Within the content-based context, it consists in considering the user individually with respect to his/her requirements. In the social or collaborative context, it consists in considering the context of other users who may have similar preoccupations.
Recommendation input data: this criterion presents recommendation source which can be a user profile, a query history (log file) or an external source(ontologies, web pages...).
Recommendation Time: this criterion presents time of recommendation: before querying, while querying or after querying.
Recommendation features: this criterion presents the recommendation object which can be a set of entire queries or a set of query fragments.
Research field: this criterion presents the domain of application of the recommendation approach. It can be database field (DB) or data warehouse field (DW).

To the best of our knowledge, only whole queries can be provided to users in OLAP recommendation context. However, we are more interested in the instant-response query recommendation. In fact, our work comes close to works that propose approaches of user query refinement in database field [7]. Even these works present some similarities with our approach (recommendation of query fragments), the challenges that need to be addressed and the techniques are very different to the ones we propose. In fact, we recommend decision queries based on a set of measures fixed by the user. Therefore, we use the Close algorithm based on minimal support which must be fixed also by the user.

III. Interactive query recommendation

It is important for query recommendations to identify the current user’s purpose in order to make an accurate recommendation. However, previous queries may include too many features that could not provide the central themes of the analysis, while the current query has little information. To overcome this drawback, our framework introduces a new

TABLE I
SURVEY OF QUERY RECOMMENDATION APPROACHES

Research Works		Stefanidis et al. 2009	Chatzopoulou et al. 2009, 2011 Yang et al. 2009	Khoussainova et al. 2010	Jerbi et al. 2009	Golfarelli et al. 2011	Khemiri et al. 2012
Recommendation Type	Content-based	×			×		×
	Collaborative	×	×	×		×	
Recommendation Time (% querying)	Before				×		
	While			×			×
	After	×	×		×	×	
Recommendation Input Data	User profile				×	×	
	Log file	×	×	×		×	×
	External source	×					
Recommendation Features	Queries	×	×		×	×	
	Query fragments			×			×
Research Field	DB	×	×	×			
	DW				×	×	×

approach that asks the user for the analysis object (measure or combination of measures) and recommends attributes based on this analysis object. To extract the previous queries, the system can use the query log to summarize the querying behaviour from past users. Our recommendation framework (Figure 1) works on the OLAP system using decision queries. Every user's query is also recorded in the log with the *user ID* to store its queries. Our recommendation approach relies on current inputting query attributes and past user's queries which are stored in query log to generate a set of attributes recommendations to the user.

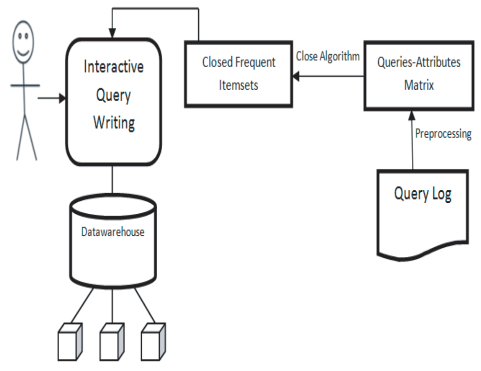


Fig. 1. Interactive query recommendation framework

A. Decision query

In data warehouse context, extracted queries are decision analysis oriented. Based on a star schema model, decision queries can be expressed on relational algebra as below:

$$q = \pi_{A,M}\sigma_P(F \bowtie D_1 \bowtie D_2 \bowtie ... \bowtie D_d)$$

where P is a conjunction of simple predicates on the attributes of dimension tables, A is a set of attributes of dimension tables D_i(attributes of Group by clause) and M is a set of measures, each measure is defined by applying an aggregation operator on the measure of fact table.

A major distinctive feature of data warehouse query is its aggregation of measures by one or more dimensions as one of the key operations; e.g., computing and ranking the total sales by each country (or by each year). Other popular operations include comparing two measures (e.g., sales and budget) aggregated by the same dimensions. Thus, the queries themselves return results computed over the measure attributes. Each of the numeric measures depends on a set of dimensions, which provide the context for the measure. The dimensions together are assumed to uniquely determine the measure [11].

In our previous study [12], we have shown the potential of recommending OLAP queries through a data mining based-approach using Apriori method. Our first approach deals with the recommendation without taking into account the measures. It is why, in this paper, we focus on first the selected measures by the user after what we recommend to him/her a set of correlated attributes.

In this paper, a key assumption in our recommendation approach is that the measure attributes have an extreme importance in analytic queries. It represents the analysis object.

B. Query recommendation process

There are three steps in our approach. First, data preparation and pattern discovery phases. Second, extracting frequent itemsets by Close algorithm and then we focus on the details of our recommendation engine.

1) Preparing and Preprocessing data: The starting and critical point for successful recommendation based on usage data is data preprocessing. It is an offline component of our recommendation approach. It can be divided into three separate stages. The first stage is that of preprocessing and data preparation, it consists in queries extraction and attributes extraction. The second is the measures lattice construction and the final stage is the binary matrix construction. Each of these components is discussed below.

a) Query extraction: Each data warehouse system keeps logs whenever users input queries. The information displayed and generated depends on the attributes that a user adopts. Attribute information can be provided to help users consider business information from different angles. Therefore, we try to discover how users make use of attributes when querying a data warehouse. Such information can be provided to other users for reference to reduce their efforts when using business intelligence systems. Our recommendation mechanism provides such information based on existing query logs. In order to create the attribute-based query, we needed to preprocess the queries included in the query logs and decompose them. This process consists of two steps, namely query generalization and query parsing. In the first step, the queries are generalized based on a set of rules, in order to be analysed and matched more efficiently. Then, they are parsed in preparation for filling the binary matrix. In fact, a workload can be easily obtained from the DBMS transaction logs. The queries workload models decision-oriented queries involving common OLAP operations, such as cube, roll-up and drill down.

b) Measures lattice construction: We group queries using the same measure(s). We represent different possible combinations of these measures in the form of a lattice structure as shown in Figure 3. Each node in the measures lattice structure represents a combination of measures. The number of levels in a lattice is equal to the number of measures. The size of the lattice (i.e., the total number of nodes where each node represents a set of frequent itemsets) grows exponentially with the number of attributes that are considered. However, generally, in a data warehouse we have few measures. Let N be the number of nodes in a lattice, then $N = (2^A - 1)$ where A represents the number of measures.

c) Building the extraction context for the frequent closed itemsets: For every node (mesaure(s) workload), we build a matrix, the rows of which represent the workload queries, and the columns represent the set of all the attributes identified in the previous step. In each node, the "query-attributes" matrix links each query to the attributes within it. Attribute presence in a query is symbolized by 1, and absence by 0.

2) Frequent Closed Itemsets Mining: The Close algorithm scans in breadth first a lattice of closed itemsets in order to extract the frequent closed itemsets and their support. Its input is an extraction context.We selected the Close Algorithm because its output is the set of the frequent closed intemsets (closed regarding the Galois connection [2]), which is a generator for all the frequent itemsets and their support. In most cases, the number of frequent closed itemsets is much lower than the total number of frequent itemsets obtained by classical algorithms such as Apriori in our previous work [12]. In our context, using Close enables us to obtain a smaller (though still significant) configuration of candidate recommendations. For each query within the workload, we extract attributes in all the clauses prefixed by the name of the clause. Intuitivelty, a closed itemset is a maximal set of items (attributes) that are common to a set of transactions (queries).

A closed itemset is a maximal set of items (attributes) that are common to a set of transactions (queries). A maximal set of items is an independent set of items that is not a subset of any other independent set. In fact, an itemset is said frequent when its support is greater or equal to a threshold parameter named minsup (minimal support).

Eventually, the application of *Close* algorithm on the extraction context outputs the set of frequent itemsets (and their support) for a minimal support fixed by the user. We consider this set as our configuration of candidate recommendations.

3) Recommendation phase: The recommendation engine is the on line component of our recommendation system based on frequent itemsets mining. Our system restricts its processing to the measure combination selected by the user. Furthermore, when a user selects a node representing a measure combination, the underlying system can exploit the list of possible frequent itemsets that involve these measures to recommend appropriate attributes to the user. Then, the user may choose to further accept the proposed suggestions by selecting one or more of the frequent items that appear on the list. The role of the data mining process becomes, simply, to find the frequent itemsets that are of interest to the user and that satisfy a minimum support value if the user provides such minimums.

IV. Illustrative example

To illustrate our approach, we use as an example FoodMart data warehouse [13] which contains sales data of a supermarket chain that is specialized in food products. Sales are represented as a fact table namely *Sales* and the contexts of analysis are represented as dimension tables namely *Product*, *Promotion*, *Time*, *Store* and *Customer*.

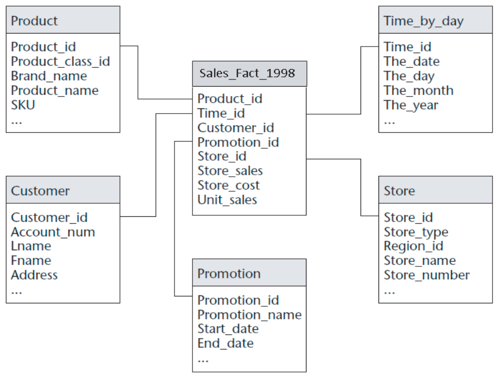


Fig. 2. Excerpt of FoodMart data warehouse

Based on this data warehouse and a related workload containing 100 queries, we show in this section how our FIMIOQR system works.

A. Preprocessing task

It is an offline task containing 3 steps.

a) Step 1: Lattice construction: On Foodmart data warehouse, we have 7 measures namely *Store_sales*, *Store_cost*, *Unit_sales*, *Amount*, *Warehouse_sales*, *Warehouse_cost* and *Store_invoice*. In this example, we are interested on sales cube (Figure 2, therefore, we use only three measures: *Store_sales*, *Store_cost* and *Unit_sales*. We represent different combinations of these measures in the above lattice in Figure 3.

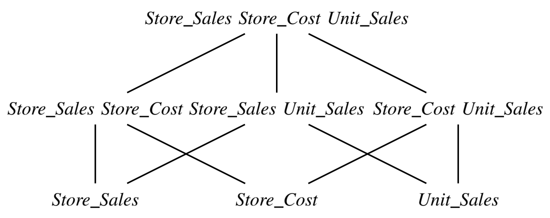


Fig. 3. Measures lattice

b) Step 2: Analysis context building: For every measure or a combination of measures in lattice nodes, we build the respective binary matrix, the rows of which represent the workload queries, and the columns represent the set of all the attributes identified in the previous step.

c) Step 3: Close algorithm application: The input of the Close algorithm consists in the binary matrix generated in step 2 and the output consists in closed frequent itemsets to each measure or a combination of measures. For instance, for the combination of measures *Store_sales*, *Store_cost* and *Unit_sales*, the output is: $\{$*product_class_id, store_name, store_manager, product_name, product_id*$\}$, $\{$*customer_id, member_card, grocery_sqft, city, the_date*$\}$,...

B. Interactive query writing task

In this task, the user is assisted by our system. The basic principle that underlies our recommendation engine is the completions of user current query. In fact, after measures selection by the user, FIMIOQR will suggest to him/her dimension attributes while his/her query writing. Assume a user has selected *Store_sales*, *Store_cost* and *Unit_sales* measures. Therefore, our system will use the frequent itemsets of the corresponding lattice node. Assume a user enters the attribute *brand_name*, our system will search the closed frequent itemsets corresponding to the selected measures and containing this attribute. FIMIOQR will recommend to the user the other correlated attributes as possible completions. Our approach provides non-intrusive recommendations to the user for query composing. Thus, the user has always the choice to accept or not these suggestions. If the user accepts an attribute suggestion, FIMIOQR will search again closed frequent itemsets containing this attribute and so forth.

V. Implementation and experimentation

In order to validate our approach, we implemented the recommendation framework called FIMIOQR (Frequent Itemsets Mining for Interactive OLAP Query Recommandation) using 'JAVA' on Netbeans environment on top of SQL Server 2005 DBMS [3]. We have applied it on a test workload of 100 queries related to FoodMart data warehouse used in our example.

To evaluate the performance of our system, several aspects of FIMIOQR can be used for answering these questions:

- Is FIMIOQR able to effectively recommend relevant attributes? (recommendation quality).
- Is the log size influence on recommendation time?
- Is the Close algorithm effective at maintaining recommendation quality, while changing the minimal support?

However, evaluating the quality of query recommendation is difficult, since there is usually no ground truth of recommendations and different annotators will have different judgements over the recommendation results. However, two metrics, precision and recall, are commonly used to measure the quality of a recommendation [14]. A recommendation system may suggest interesting or uninteresting objects. The recall measure indicates the effectiveness of a method for locating interesting objects, while the precision measure represents the extent to which the instances recommended by a method really are interesting to users. The formulas are as follows:

$$recall = \frac{\textit{number of correctly recommended objects}}{\textit{number of interesting objects}}$$

$$precision = \frac{\textit{number of correctly recommended objects}}{\textit{number of recommended objects}}$$

Whereas, in our study, the number of interesting objects is fixed since it is equal to the number of attributes used in a data warehouse. In our example in section IV, we have 95 attributes. Therefore, the range or scope of recommendations is limited; consequently, the recall metric is not applicable in our study. We can only use the precision metric to evaluate the quality of the proposed recommendations.

In our approach, the number of correctly recommended objects presents the number of accepted recommendations (attributes). The relevance of each attribute to the input query was judged by members of our laboratory (students and professors). They analysed the recommended itemsets and determined the items that are of interest to the input query. In fact, the recommendation precision (RP) is calculated as follows:

$$RP = \frac{\textit{number of accepted recommended attributes}}{\textit{number of recommended attributes}}$$

Our first experiment evaluates the accuracy of the proposed approach to make the recommendations. Figure 4 shows the performance recommendation time according to the size of query log.

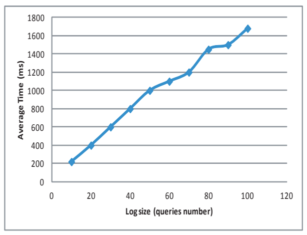


Fig. 4. Accuracy Analysis

As can be seen from Figure 4, it is obvious that the trend of execution time is upwards with the log size. The execution time is acceptable with the size of 50 queries in query log.

The Figure 5 (a) shows that the precision slightly increases with the log size. This figure demonstrates that most current queriess can obtain a successful recommendation by our approach with precision between 0.14 and 0.58 from the query log.

In addition, our recommendation framework has been executed for various values of the Close minsup (minimal support) parameter. In practice, this parameter helps us limiting the number of candidates to generate by selecting only those that are the most frequently used by the workload. Figure 5 (b) shows the precision of recommendation according to the value of minimal support. As can be seen from Figure 5 (a), it is obvious that the trend of recommendation precision is upwards with minsup parameter until 60% where it decreases.

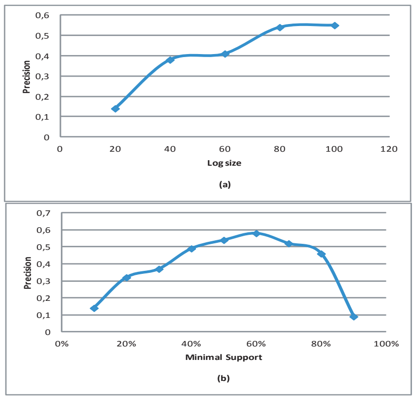


Fig. 5. (a) Recommendation precision for different values of minsup; (b) Average precision

VI. Conclusion and Future Works

In this paper, we have proposed a novel approach for interactive real time decision query recommendation. Our method allows to ease the user's burden of the query formulation based on decision query logs, namely FIMIOQR. The contribution of our paper consists in providing a tool to help users interact with data warehouse while formulating their queries. This tool is based on Close algorithm in order to extract frequent closed itemsets.

Unlike most previous works, the method we propose attempts to recommend dimension attributes rather than whole queries. Throughout the paper we tended to justify that mining log files can give us a concise set of information about the usage of the system and enables us efficient handling of that information. In the context of query recommendations, when a user submits a query, the system should be able to assist him/her in this task.

Our approach can be easily extended to collaborative recommendation by identifying user preferences in order to exploit query logs of all users and to recommend to the current user dimension attributes of similar users. Moreover, a data warehouse often contains vast amounts of information that is difficult for a new user to comprehend. What attributes should be used initially for a new user without any data warehouse experience? It is necessary to employ a recommendation mechanism to assist such users by suggesting him/her for example attributes based on the logs of other experienced users.

In terms of future work, there is much to be done. First and foremost, we intend on looking for most accepted recommendations by the user in order to rank them through skyline queries and to recommend them first to the user in his/her future sessions. Therefore, the recommendation engine matches past queries to an improved ranking, leading to recommendations of higher quality.

In addition, a data warehouse user follows a logical reasoning in a querying process. Data in an analysis session is often correlated. Therefore, we must think about generalizing our approach against user sessions.

Finally, a real data set should be used for the simulation. We assume that the query logs follow a normal distribution. However, the generated data has drawbacks that impact on the effectiveness of recommendations. Future work could be extended to make more solid recommendations by working with some organizations to obtain real data so that the analysis would be closer to real-world situations.

References

[1] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *VLDB*, 1994, pp. 487–499.
[2] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, "Discovering frequent closed itemsets for association rules," in *ICDT*, 1999, pp. 398–416.
[3] http://eric.univ-lyon2.fr/~bentayeb/logiciels.html.
[4] K. Stefanidis, M. Drosou, and E. Pitoura, " "you may also like" results in relational databases," in *PersDB workshop*, 2009.
[5] X. Yang, C. M. Procopiuc, and D. Srivastava, "Recommending join queries via query log analysis," in *ICDE*, 2009, pp. 964–975.
[6] G. Chatzopoulou, M. Eirinaki, and N. Polyzotis, "Query recommendations for interactive database exploration," in *SSDBM*, 2009, pp. 3–18.
[7] N. Khoussainova, Y. Kwon, M. Balazinska, and D. Suciu, "Snipsuggest: Context-aware autocompletion for sql," *PVLDB*, vol. 4, no. 1, pp. 22–33, 2010.
[8] H. Jerbi, F. Ravat, O. Teste, and G. Zurfluh, "Applying recommendation technology in olap systems," in *ICEIS*, 2009, pp. 220–233.
[9] H. Jerbi, F. Ravat, O. Teste, and G. Zurfluh, "Preference-based recommendations for olap analysis," in *DaWaK*, 2009, pp. 467–478.
[10] M. Golfarelli, S. Rizzi, and P. Biondi, "myolap: An approach to express and evaluate olap preferences," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 7, pp. 1050–1064, 2011.
[11] S. Chaudhuri and U. Dayal, "An overview of data warehousing and olap technology," *SIGMOD Record*, vol. 26, no. 1, pp. 65–74, 1997.
[12] R. Khemiri and F. Bentayeb, "Interactive query recommendation assistant," in *DEXA Workshops*, 2012, pp. 93–97.
[13] http://www.e-tservice.com/downloads.html.
[14] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.

Geometric Mean based Boosting Algorithm to Resolve Data Imbalance Problem

Myoung-Jong Kim
Department of Business Administration,
School of Business, Pusan National University
Busan, South Korea
mjongkim@pusan.ac.kr

***Abstract*—Data imbalance problem has received a lot of attention in machine learning community because it is one of the causes that degrade the performance of classifiers or predictors. In this paper, we propose geometric mean based boosting algorithm (GM-Boost) to resolve the data imbalance problem. GM-Boost enables learning with consideration of both majority and minority classes because it uses the geometric mean of both classes in error rate and accuracy calculation. We have applied GM-Boost to bankruptcy prediction task. The results indicate that GM-Boost has the advantages of high prediction power and robust learning capability in imbalanced data as well as balanced data distribution.**

Keywords - data imbalance; GM-Boost; bankruptcy prediction

I. Introduction

Data imbalance problem is frequently observed in various classification and prediction tasks when most of training samples belong to one majority class. Data imbalance problem is reported in a wide range of classification tasks, such as oil spill detection [16], response modeling [23], remote sensing [1], scene classification [27], card fraud detection [9] and credit rating [18].

Data imbalance problem could be one of the main causes that degrade the performance of machine learning algorithms in classification tasks. There are two main reasons why data imbalance causes degradation in performance of machine learning algorithms [13,14,24]. The first reason is associated with the objective function of classification algorithms. One of widely used objective functions for classification algorithms is the arithmetic mean based accuracy (hereafter, arithmetic accuracy) which is a ratio of the number of correctly classified instances over the number of total instances. However, in the presence of data imbalance, arithmetic accuracy can be inappropriate because the accuracy is highly dependent on the classification accuracy of majority class samples. More specifically, in very imbalanced domains, most standard classifiers will tend to learn how to predict the majority class. While these classifiers can obtain higher predictive accuracies than those that also try to consider the minority class more, this seemingly good performance can be argued as being meaningless [24].

The second reason for the degradation in performance is the distortion of decision boundaries resulting from imbalanced distribution of the classes. As the imbalance of data is getting severe, the decision (classification) boundary of majority class tends to invade the decision boundary of the minority class, so that the decision boundary of majority class is gradually expanded while the decision boundary of minority class is gradually reduced. This problem eventually causes the decrease in the accuracy for minority class.

For the alternatives to solve this problem, various methods have been proposed including under-sampling, over-sampling, cost adaptive strategies, and boosting algorithms. Recently, various boosting algorithms have been proposed as alternatives for data imbalance problems including SMOTEBoost [3] and RUSBoost [22]. In particular, SMOTEBoost is an application of boosting techniques to over-sampled data generated by synthetic minority over-sampling technique (SMOTE) [2]. SMOTE effectively creates a new minority samples, while boosting algorithm proceeds training on over-sampled data through repetitive sampling process which focuses on misclassified observations. In this way, SMOTEBoost can reinforce the training over samples from minority class to be likely misclassified. However, the boosting algorithm can be inappropriate as for over-fitting problem because its objective function is still measured in terms of arithmetic accuracy and arithmetic errors. New minority class samples, which are generated from SMOTE, are likely to have the higher similarity than majority data samples. Most standard learning algorithms will tend to generate classifiers focusing on samples with higher similarity because that strategy is helpful to maximize the objective function, i.e. arithmetic accuracy. This drawback might increase generalization errors when classifiers are applied to new validation data set which is not trained.

This paper proposes geometric mean based boosting (GM-Boost) which is a novel boosting algorithm applying the concept of geometric accuracy to AdaBoost algorithm [10]. It has the advantage of enabling balanced learning against both majority and minority classes. The proposed GM-Boost algorithm is applied to bankruptcy prediction task which is one of the typical data imbalance problems in business domains. Two different data samples, imbalanced data and balanced data samples, are constructed to verify the performance of GM-Boost algorithm.

Experimental results show that GM-Boost has the advantages of high prediction power and robust learning capability in imbalanced data distribution as well as in balanced data distribution.

II. Data Imbalance Problem in Binary classification problems

A. Data Imbalance Problem

Kang and Cho [13] constructed six sample groups according to different data balance rates (1:1, 1:3, 1:5, 1:10, 1:30, and 1:50) in order to analyze the effects of data imbalance on classification accuracy of SVM. From their experimental results, it can be seen that, for the two sample groups with little or no data imbalance problem (1:1, and 1:3), the sizes of classification boundary areas of the two classes are similar to each other. However, for the sample groups with serious data imbalance problems (1:5, and 1:10), the area of minority class is reduced because the area of the majority class invades the area of minority class, and thus the classification accuracy for minority class samples is degraded. Especially, for the sample groups with extreme data imbalance (1:30, and 1:50), it is reported that the classification boundary area for minority class is excessively small, which makes the classification for minority class meaningless. Also, they report that, as the data imbalance is getting severe, arithmetic accuracy over total samples steadily increases due to the high accuracy over samples of majority class, while the arithmetic accuracy for minority class is dramatically reduced, and thereby geometric accuracy over total samples gradually decreases. They argue that these results demonstrate that arithmetic accuracy is not a suitable objective function for imbalanced data.

Wu and Chang [24] assert two following results as a cause of skewed boundaries of SVM due to data imbalance. Firstly, data imbalance problem causes a tendency that samples of minority class do not reside in the boundary area of minority class. Secondly, as the data imbalance is getting severe, boundary area of majority class is expanded and boundary area of minority class is reduced due to the imbalance of support vectors. This problem causes the distortion of boundary area. Consequently, the possibility becomes very high that the classifier will classify a sample as a majority class.

B. The Approaches to Resolve Performance Measure Problem

Arithmetic accuracy is a proper performance measure for classifiers in balanced data set. However, under data imbalance, it is not a proper performance measure anymore because it is highly influenced by the classification accuracy of majority class [12, 13, 23]. Geometric accuracy and ROC analysis are proposed to resolve this problem. The geometric accuracy is calculated as a square root of sensitivity multiplied by specificity [14] where sensitivity and specificity are TP/(TP+FN) and TN/(FP+TN) respectively. In ROC analysis, we usually plot and connect each sample to generate a polyline ordered by their classification score in two dimensional Cartesian coordinate system where x axis denotes 1- specificity and y axis denotes sensitivity. The accuracy of the classifier is calculated as an area under the ROC curve (AUROC). In a perfect model, AUROC is 1.0 and in a random guess model, AUROC is 0.5. Most models generally have AUROC which is higher than 0.5 and lower than 1.0. As AUROC becomes closer to 1.0, the model is regarded as more accurate [7].

C. The Approaches to Resolve Data Distribution Problem

The previously proposed methods to resolve data imbalance can be divided into twofold: data sampling and the assignments of weights (penalties) to misclassified instances [13].

There have been two types of data sampling strategies, under-sampling and over-sampling, which is generally used to resolve data imbalance problems. Under-sampling removes a portion of majority class samples randomly or predefined rules in accordance with the number of minority class samples. Obviously this method incurs information loss of majority class samples. However, it has been shown that, if we adopt adequate rules to select and remove samples, it can successfully resolve data imbalance problems [11, 15, 18]. Over-sampling increases the number of minority class samples using data duplication and data generation [3, 11]. This method is advantageous in that there is no information loss in the majority class, however, overall learning time will increase as the number of data samples increases. In particular, since it creates new samples based on the similarity among minority samples, it could trigger the over-fitting problem and generalization error for novel data samples.

In weights assignments methods, cost adaptive learning strategies are generally used to impose different penalties on misclassified patterns. That is, if a sample in minority class is misclassified, the higher penalty is imposed on the misclassification than the penalty when a sample in majority class is misclassified [6, 20]. Although this method does not have problems like information loss of under-sampling or generalization error of over-sampling, it can cause to generate unstable classifiers due to the excessive sensitivity about the samples.

Recently, the combinations of sampling and boosting algorithms such as SMOTEBoost [3], RUSBoost [21], etc. have been applied to data imbalance problem and shown successful results. Boosting algorithms sequentially generate ensemble of classifiers, assigning higher weights to misclassified observations than to correctly classified observations, and thereby it has an advantage that it can strengthen learning on minority class samples with the high probabilities of misclassification.

III. GM-Boost Algorithm

In this section, we will explain SMOTE, AdaBoost, and GM-Boost algorithms which are used in this research.

A. SMOTE Algorithm

SMOTE algorithm is used to generate new samples for minority class data. SMOTE algorithm combines a certain observation with k similar minority class samples to generate a new sample according to the following calculation: $X_{new}=X+rand(0,1)\times(X_n-X)$ where X_{new}, X, and X_n respectively means newly generated sample, the original sample, and the nearest k samples to the original sample. SMOTE algorithm consists of three steps as followings; Firstly, the nearest k samples to the original sample is chosen, secondly the distances of the original sample and k samples is multiplied by a random number between zero and one, and finally, the average of the multiplied distances is added to the original sample in order to generate a new sample. In this way, we repeat SMOTE sampling to increase the samples of minority class until both the numbers of the minority class and majority class become same.

B. AdaBoost

To explain AdaBoost, we assume an ensemble $C=\{C_1,C_2,\ldots,C_K\}$ composed of K base classifiers from n training samples. Then the error rate for kth base classifier (e_k) is calculated as an arithmetic mean, which is as follows.

$$e_k=\sum_{i=1}^{n} w_k(i)L(C_k(x_i),y_i)$$

where, $L(C_k(x_i),y_i)=\begin{cases}1 & C_k(x_i)\neq y_i\\ 0 & C_k(x_i)=y_i\end{cases}$ and $\sum_i w_k(i)=1$

Note that x_i is a vector of predictor variables for ith observation, y_i is a category of ith observation, and $C_k(x_i)$ is a classification result of kth classifier on the predictor variable vector x_i. For the (k+1)th classifier, the weight for ith observation is adjusted as follows, which impose higher weights on misclassified observations.

$$w_{k+1}(i)=\frac{w_k(i)exp(-\alpha_k C_k(x_i)y_i)}{Z_k}$$

where $Z_k=\sum_i w_k(i)exp(-\alpha_k C_k(x_i)y_i)$

Note that α_k is conceptually interpreted as an importance or accuracy of the classifier, and calculated as $\alpha_k=\frac{1}{2}ln\big((1-e_k)/e_k\big)$. When the training samples are constructed for (k+1)th classifier, since higher weights are assigned to misclassified observations, the boosting algorithm can proceed training focused on misclassified observations. The ensemble learning algorithm stops when $e_k > 0.5$, and the classification result of the ensemble for ith observation is a weighted mean of base classifiers' classification expressed as follows:

$$C(x_i)=sign\left(\sum_{k=1}^{K}\alpha_k C_k(x_i)\right)$$

Because of the advantage that AdaBoost algorithm provides learning opportunity to minority class samples, various boosting algorithms based on AdaBoost are frequently applied to data imbalance problem as an alternative solution. As data imbalance is more severe, the error rate for minority class is higher whereas the error rate for majority class is lower. Since higher weights are assigned to minority class samples in the process of constructing training samples for new classifier, the new classifier will strengthen its learning for minority class. In this way, although learning algorithm is concentrated on majority class samples in the beginning stage of ensemble learning, gradually there become more learning opportunities for minority class samples. Because of this characteristic, those boosting algorithms have an advantage that it yields robust learning performance even under data imbalance.

However, the boosting algorithms can exhibit the over-fitting and generalization problems because they try to maximize arithmetic accuracy. The error rate of the classifier e_k and the performance of the classifier, a_k, are measures based on arithmetic mean. As is mentioned before, measures based on arithmetic accuracy might not be valid as a useful objective function because the objective function based on arithmetic measures tends to generate a strongly biased classification function towards majority class or class with high similarity among samples. Especially, when the boosting algorithms are applied after SMOTE algorithm, which generates a new data sample from a group of adjacent data samples weighted with their inter-distances, it will increase the inductive bias due to the increased similarity among the group of data samples and will eventually aggravate the over-fitting effects. To alleviate these problems, we introduce a notion of accuracy based on geometric mean, which can consider predictive performances of both majority class and minority class, to machine learning algorithms.

C. GM-Boost Algorithm

In addition to the aforementioned assumptions for AdaBoost algorithm, we assume that, out of n training samples, n^+ samples are in minority class and n^- samples are in majority class. We let e_k^+ be the error rate for minority class of kth classifier and e_k^+ be the error rate for minority class of kth classifier. Then the geometric mean based error rate e_k, can be defined as follows:

$$e_k=\sqrt{e_k^+\cdot e_k^-},$$

$$\text{where } e_k^+=\frac{\sum_{i=1}^{n^+} w_k(i)L(C_k(x_i),y_i)}{\sum_{i=1}^{n^+} w_k(i)} \text{ and}$$

$$e_k^-=\frac{\sum_{i=1}^{n^-} w_k(i)L(C_k(x_i),y_i)}{\sum_{i=1}^{n^-} w_k(i)}$$

Accordingly, α_k which means classification accuracy of the classifier is calculated as a geometric mean based accuracy of classification accuracies of minority class and majority class.

$$\alpha_k=ln\left(\sqrt{\mu\cdot\alpha_k^+\cdot\alpha_k^-}\right),$$

$$\text{where } \alpha_k^+=\frac{1-e_k^+}{e_k} \text{ and } \alpha_k^-\frac{1-e_k^-}{e_k}$$

Note that μ is a weighting degree that controls the weight value multiplied to each instance. Following AdaBoost, the weight imposed on the samples for (k+1)th classifier is calculated as follows:

$$w_{k+1}(i) = \frac{w_k(i)\exp(-\alpha_k C_k(x_i)y_i)}{Z_k}$$

$$\text{where } Z_k = \sum_i w_k(i)\exp(-\alpha_k C_k(x_i)y_i)$$

And the final classification result for ith observation is calculated as a linear combination of ensemble results and α_k.

$$C(x_i) = sign\left(\sum_{k=1}^{K} \alpha_k C_k(x_i)\right)$$

Having an advantage of providing learning opportunity to minority class samples, various boosting algorithms based on AdaBoost are frequently applied to data imbalance problem as an alternative solution. As data imbalance is more severe, the error rate for minority class is higher whereas the error rate for majority class is lower. Since higher weights are assigned to minority class samples in the process of constructing training samples for new classifier, the new classifier will strengthen its learning for minority class. In this way, although learning algorithm is concentrated on majority class samples in the beginning stage of ensemble learning, gradually there become more learning opportunities for minority class samples. Upon such characteristic, AdaBoost has an advantage of yielding robust learning performance even under data imbalance.

However, the boosting algorithms can exhibit the over-fitting and generalization problems because they try to maximize arithmetic accuracy. The error rate of the classifier ek and the performance of the classifier, a_k, are measures based on arithmetic mean. As mentioned before, measures based on arithmetic accuracy might not be valid as a useful objective function because the objective function based on arithmetic measures tends to generate a strongly biased classification function towards majority class or class with high similarity among samples. Especially, when the boosting algorithms are applied after SMOTE algorithm, which generates a new data sample from a group of adjacent data samples weighted with their inter-distances, it will increase the inductive bias due to the increased similarity among the group of data samples and will eventually aggravate the over-fitting effects. The notion of geometric accuracy, which can consider predictive performances of both majority class and minority class, is introduced to alleviate these problems.

IV. Research Design

We collected the experimental data used for this research from a Korean commercial bank. The bankrupt companies are 500 audited manufacturing companies during year 2002 to year 2005, while the non-bankrupt companies are 2,500 audited manufacturing companies during 2002-2005. For the non-bankrupt companies, we collected 10,000 firm-year financial statements during 2001-2004. In this way, we collected a total of 10,000 financial statements based on firms-year standard, and the average bankrupt rate for the four years is about five percent, which falls in the expected range of bankruptcy rate (three to five percent) estimated by professional credit rating agencies.

As for the financial ratios for bankruptcy prediction, we collected seven thirty financial ratios, which have been usefully applied in the previous corporate bankruptcy prediction researches. The collected ratios are divided into seven financial ratio groups including profitability, debt coverage, leverage, capital structure, liquidity, activity, and size. Consequently, the seven final input variables, each of which has the highest AUROC in each group, are selected.

Variance information factor (VIF) analysis is performed to check for multicollinearity among the seven financial ratios. Table 1 shows the AUROC and VIF of the seven final input variables. We can see that the chosen variables do not exhibit any substantial multicollinearity because all the VIFs are below four.

TABLE 1. THE RESULT OF VARIANCE INFLATION FACTOR ANALYSIS ON THE CHOSEN VARIABLES.

Variables	AUROC	**VIF**
Ordinary income to total assets	51.7	1.36
EBITDA to Interest expenses	51.2	2.11
Total debt to total assets	50.9	1.77
Retained earning to total assets	52.5	2.53
Cash ratio	45.5	1.34
Inventory to sales	30.5	1.59

V. Research Results

Sequential minimal optimization (SMO) is used as a SVM base classifier and the radial basis function (RBF) is used as as a kernel function. There are two parameters in RBF kernels: acceptable error C and kernel parameter δ^2. We made up various configurations of the two parameters: varying C from 1 to 250, and δ^2 from 1 to 200.

We prepared samples through two stages. At the first stage, we chose samples from the total of 10,500 cases, with the ratio of bankrupt companies to normal companies as 1:1(A), 1:3(B), 1:5(C), 1:10(D), and 1:20(E). Then we set 60% of each of them as training samples, and the rest 40% of each of them as test samples. Table 2 shows these configurations of samples. We repeated these steps of the first stage fifty times to generate fifty training sample sets and fifty test sample sets for each of the five configurations (A, B, C, D, and E).

TABLE 2. CONFIGURATIONS OF IMBALANCED DATA SAMPLES

Set		**Training**			**Validation**		
		Bankrupt	***Normal***	***Total***	***Bankrupt***	***Normal***	***Total***
A	1:1	300	300	600	200	200	400
B	1:3	300	900	1,200	200	600	800
C	1:5	300	1,500	1,800	200	1,000	1,200
D	1:10	300	3,000	3,300	200	2,000	2,200
E	1:20	300	6,000	6,300	200	4,000	4.200

At the second stage, we used SMOTE algorithm, where k is set to five, to generate new bankrupt companies, so that we obtained the number of bankrupt companies same with that of normal companies. Table 3 shows these configurations of samples. We repeated the same sampling process fifty times to generate fifty training sample sets and fifty test sample sets for each of four configurations (B, C, D, and E).

TABLE 3. CONFIGURATIONS OF BALANCED DATA SAMPLES.

Set		Training			Validation		
		Bankrupt	*Normal*	*Total*	*Bankrupt*	*Normal*	*Total*
A	1:1	300	300	600	200	200	400
B	1:3	900	900	1,200	200	600	800
C	1:5	1,500	1,500	1,800	200	1,000	1,200
D	1:10	3,000	3,000	3,300	200	2,000	2,200
E	1:20	6,000	6,000	6,300	200	4,000	4.200

A. Experimental Results in Imbalanced Data

Table 4 shows the results of average accuracy of fifty validations. In case of AdaBoost, as the data imbalance is getting severe, arithmetic accuracy over total samples is steadily increased due to the high accuracy over samples of majority class, while the arithmetic accuracy for minority class is dramatically reduced, and thereby geometric accuracy over total samples is gradually decreased. In particular, average accuracy for minority class of sample groups C, D, and E is 7%, 4.5%, and 3.5%, respectively. It indicates that the classification for minority class is meaningless. Those results are caused by arithmetic error and accuracy calculation of AdaBoost.

Comparing to AdaBoost, however, GM-Boost shows stable arithmetic accuracy for minority class and geometric accuracy over total samples. T-test is performed to analyze the difference of geometric accuracy between both boosting algorithms for the five configurations (A, B, C, D, and E). The results of T-test show that the prediction accuracy between two training algorithms for sample group A is significantly different at 5% level and for sample group B, C, D, and E is different at 1% level, respectively. The difference in geometric accuracies becomes higher, as the data imbalance becomes more severe.

B. Experimental Results in Balanced Data

We apply the final sampled sets generated from SMOTE to AdaBoost and GM-Boost experiments. Table 5 shows the results of average accuracy of fifty validations. As noted, the higher is the proportion of new generated samples in minority class, the higher is the similarity among minority class samples. SVM, the base classifier of AdaBoost, will tend to learn focusing on minority samples with high similarity because this strategy is helpful maximizing arithmetic accuracy. Boosting algorithms also try to modify the weight of each instance based on misclassification, but do not try to balance majority class error and minority class error. This problem leads to over-fitting problem and deteriorates the performance of SMOTEBoost in the perspectives of generalization and prediction for novel samples.

In our case, since data set E has the higher proportion of new generated samples and the higher similarity among data samples than any other data sets, it is likely to show the lower prediction performance for novel samples. Hence, while the accuracy of AdaBoost for majority class samples consistently lies on the interval between 0.750 and 0.830, its accuracy for minority class samples becomes lower as the degree of data imbalance is higher. Thus, arithmetic accuracy of AdaBoost stably lies between 0.750 and 0.798, but geometric accuracy continues to deprecate from 0.797 to 0.734. On the contrary, GM-Boost, that employs geometric accuracy, systematically avoids this over-fitting problem because it considers both accuracies of majority class category and minority class category. Consequently, GM-Boost exhibits more robustness and generalization than AdaBoost does for novel test samples. T-test is performed to compare the prediction accuracy between AdaBoost and GM-Boost for the five configurations (A, B, C, D, and E). The results show that significant difference between two algorithms in classification accuracies for all configurations except the configuration A.

TABLE 4. PREDICTION ACCURACY AND THE T-TEST FOR THE FIVE CONFIGURATIONS OF IMBALANCED DATA SAMPLES

Set	**SVM**				**GM-Boost**				**t-value**
	Majority	**Minority**	**Arithmetic**	**Geometric**	**Majority**	**Minority**	**Arithmetic**	**Geometric**	
A	0.820	0.755	0.788	0.787	0.820	0.780	0.800	0.800	1.851*
B	0.960	0.330	0.803	0.563	0.893	0.630	0.828	0.750	2.435**
C	0.990	0.070	0.837	0.263	0.891	0.610	0.844	0.737	2.704**
D	0.999	0.045	0.912	0.212	0.916	0.505	0.879	0.680	3.291**
E	0.998	0.035	0.952	0.187	0.912	0.420	0.889	0.619	3.557**

** and * represent significance levels at 1% and 5%, respectively.

TABLE 5. PREDICTION ACCURACY AND THE T-TEST FOR THE FIVE CONFIGURATIONS OF BALANCED DATA SAMPLES

Set	**AdaBoost**				**GM-Boost**				**t-value**
	Majority	**Minority**	**Arithmetic**	**Geometric**	**Majority**	**Minority**	**Arithmetic**	**Geometric**	
A	0.830	0.765	0.798	0.797	0.820	0.780	0.800	0.800	0.152
B	0.750	0.750	0.750	0.750	0.747	0.770	0.753	0.758	1.852*
C	0.775	0.720	0.766	0.747	0.808	0.745	0.798	0.776**	2.438**
D	0.755	0.720	0.751	0.737	0.775	0.765	0.774	0.770*	3.257***
E	0.775	0.695	0.771	0.734	0.776	0.785	0.776	0.780*	3.997***

***, **, and * represent significance levels at 1%, 5%, and 10%, respectively.

VI. Conclusion and Future Research Directions

Data imbalance problem has received a lot of attention in machine learning community because it is one of the causes that degrade the performance of classifiers or predictors. In our research, we proposed GM-Boost algorithm to resolve data imbalance problem. The proposed GM-Boost algorithm is applied to bankruptcy prediction task to verify the performance of GM-Boost algorithm. At the first stage, five sample groups are constructed according to different data balance rates (1:1, 1:3, 1:5, 1:10, and 1:20) and classification experiments using AdaBoost and GM-Boost are performed against those imbalanced data sets. At the second stage, SMOTE algorithm is used to generate new bankrupt company data sets and the newly sampled sets is applied to AdaBoost and GM-Boost experiments for the performance verification of GM-Boost in balanced data. Experimental results show that GM-Boost has the advantages of high prediction power and robust learning capability in imbalanced data distribution as well as balanced data distribution.

We expect the following future researches to be conducted to cope with the limitations of GM-Boost. Firstly, boosting algorithms have drawbacks that degrade classification accuracy when outliers are included in the learning samples or when there is high correlation between the classifiers in the ensemble. Various methods have been proposed to compensate these shortcomings [4,5,20], and we plan to conduct researches to develop algorithms coupled with those methods. Secondly, the ensemble algorithm we propose in this research is a modification of a boosting algorithm to solve data imbalance problem. However, it can be possible to solve the data imbalance problem by combining our results with SVM kernel management [11,26], so we anticipate future researches in this direction.

References

[1] L. Bruzzone, and S. B. Serpico, “Classification of imbalanced remote-sensing data by neural networks,” Pattern Recognition Letters 18(11-13), 1997, pp. 1323–1328

[2] N. Chawla, K. Bowyer, L. Hall, and W. Kegelmeyer, “SMOTE: synthetic minority oversampling techniques,” Journal of Artificial Intelligence Research, 16, 2002, pp. 321-357.

[3] N. Chawla, A. Lazarevic, L. Hall, and K. Bowyer, “SMOTEBoost: improving prediction of the minority class in boosting,” 7th European conference on principles and practice of knowledge discovery in databases. Cavtat-Dubrovnik, Croatia, 2003, pp. 107-119.

[4] T. M. Cover, and J. A. Thomas, “Element of information theory”, John Wiley & Sons, 1991.

[5] G. A. Darbellay, “An estimator of the mutual information based on a criterion for independence,” Computational Statistics and Data Analysis, 32, 1999, pp. 1-17.

[6] C. Elkan, “The foundation of cost-sensitive learning,” Proceedings of the 17th International Joint Conference on Artificial Intelligence, Seattle, WA, 2001, pp. 973-978.

[7] T. Fawcett, “An introduction to ROC analysis,” Pattern Recognition Letters, 27, 2006, pp. 861-874.

[8] T. Fawcett, and F. Provost, “Adaptive fraud detection. Data Mining and Knowledge discovery,” 1(3) , 1997, pp. 291-316.

[9] Y. Freund, and R. E. Schapire, “A decision theoretic generalization of online learning and an application to boosting,” Journal of Computer and System Science, 55(1) , 1997, pp. 119-139.

[10] X. Hong, “A kernel-based two-class classifier for imbalanced data sets,” IEEE Transactions on neural networks, 18(1) , 2007, pp. 28-40.

[11] N. Japkowicz, and S. Stephen, “The class imbalance problem: a systematic study,” Intelligent Data Analysis, 6(5) , 2002, pp. 429-250.

[12] P. Kang, and S. Cho, “EUS SVMs: Ensemble of Under-Sampled SVMs for Data Imbalance Problems,” ICONIP 2006, Part I, LNCS 4232, pp 837-846.

[13] S. Kotsiantis, D. Tzelepis, E. Kounmanakos, and V. Tampakas, “Selective costing voting for bankruptcy prediction,” International Journal of Knowledge-based and Intelligent Engineering Systems, 11, 2006, pp. 115-127, 2007.

[14] M. Kubat, R. Holte, and S. Matwin, “Learning when Negative example abound,” Proceedings of the 9th European Conference on Machine Learning, ECML’97, 1997.

[15] M. Kubat, and S. Matwin, “Addressing the curse of imbalanced training sets: one-sided selection,” Proceedings of the Fourteenth International Conference on Machine Learning, 1997, pp. 179-186.

[16] M. Kubat, R. Holte, and S. Matwin, “Machine Learning for the detection of oil spills in satellite radar images,” Machine Learning 30(2), 1998, pp. 195-215

[17] Y. S. Kwon, I. G. Han, and K. C. Lee, “Ordinal Pairwise Partitioning (OPP) approach to neural networks training in bond rating,” Intelligent Systems in Accounting, Finance and Management, 6, 1997, .pp. 23-40.

[18] J. Laurikkala, “Instance-based data reduction for improved identification of difficult small classes,” Intelligent Data Analysis, 6(4), 2002, pp. 311-322.

[19] T. T. Maia, A. P. Braga, and A. F. Carvalho, “Hybrid classification algorithms based on boosting and support vector machines,” Kybernetes, 37(9), 2008, pp. 1469-1491.

[20] F. Provost, and T. Fawcett, “Robust classification for imprecise environments,” Machine Learning, 42, 2001, pp. 203-231.

[21] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, “RUSBoost: Improving classification performance when training data is skewed,” 19th International Conference on Pattern Recognition, 2008, pp. 1-4.

[22] H. J. Shin, and S. Z. Cho, “Response Modeling with Support Vector Machine,” Expert Systems with Applications 30(4), 1997, pp. 746–760

[23] B. X. Wang, and N. Japkowicz, “Boosting support vector machines for imbalanced data sets,” Knowledge and Information Systems, Knowledge and Information Systems, 25, 2009, pp. 1-20.

[24] G. Wu, and E. Chang, “Adaptive feature-space conformal transformation for imbalanced data learning,” Proceedings of the 20th International Conference on Machine Learning, 2003

Primitive Operation Aggregation Algorithms for Improving Taxonomies for Large-Scale Hierarchical Classifiers

Kiyoshi Nitta
Yahoo Japan Research
Midtown Tower, 9-7-1 Akasaka, Minato-ku, Tokyo, Japan
Email: knitta@yahoo-corp.jp

***Abstract*—Naive implementations of hierarchical classifiers that classify documents into large-scale taxonomy structures may face the contradiction between relevancy and efficiency performances. To address this problem, we focused on taxonomy modification algorithms for gradually improving the relevance performances of large-scale hierarchical classifiers. We developed four taxonomy modification algorithms that aggregate primitive operations before investigating hierarchical relevance performances. All but one produced taxonomy sequences that generate classifiers exhibiting practical efficiencies. One algorithm, which strictly maintains balanced proportions of taxonomy structures, generated a taxonomy sequence producing classifiers that exhibit stable relevancy performances. Another algorithm, which roughly maintains the proportions of taxonomy structure, but strictly maintains the maximum size of the training corpus for each local classifier, generated a taxonomy producing a classifier that exhibited the best relevance performance in our experiment. The base classification system we developed for this experiment uses an approach that locates local classifiers per parent node of taxonomies. It is able to classify documents into directed-acyclic-graph structured taxonomies. The system reached the level of practical hierarchical classification systems that efficiently and relevantly predict documents into over 10,000 taxonomy classes.**

***Keywords*-hierarchical classification; taxonomy modification**

I. Introduction

A classifier with a larger set of classes has the potential to enable more precise predictions than one with fewer classes. Hierarchical structures are commonly applied to large sets of classes to increase the usability of the classes. A hierarchical structure for a classifier is called a *taxonomy*. Most taxonomies are constructed manually. They provide natural and easy-to-use methods for users to access categorized documents. Automatic classification systems that classify documents into hierarchical taxonomies are called *hierarchical classifiers*. What strategy we should take to implement hierarchical classifiers is one of the most essential aspects when classifying documents into large-scale taxonomies. There are three basic classification strategies: the *top-down (or local) strategy* [1]–[6], the *big-bang (or global) strategy*, and the *flat strategy* [7].

While taxonomies usually provide natural and easy-to-use methods for users, they rarely provide the best efficiency and relevancy hierarchical structures for automatic classifiers. There are two approaches for improving hierarchical classifier performances by maintaining the benefit of manually constructed taxonomies:

1) *clustering approach* — decomposing a taxonomy into a flat set of classes and applying an ordinary clustering algorithm to those classes for building the best performance hierarchical structure [8],
2) *gradually modifying approach* — gradually modifying taxonomy structures by applying promote, demote, and merge primitive operations on hierarchical structures [9]–[11].

The first approach has a high possibility of obtaining the best performing hierarchical structure because there is no restriction in arranging structures. The second approach has an advantage that it can precisely control the balance of modified hierarchical structures. Unbalanced hierarchical structures may result in producing inefficient or irrelevant classifiers, especially when they used the local classification strategy.

We present four versions of taxonomy modification algorithms formulated based on the gradually modifying approach for local strategy hierarchical classifiers. The first algorithm simply and simultaneously accumulates primitive operations for all taxonomy nodes. The second algorithm reduces primitive operations that induce child node concentrations to specific nodes. The third algorithm limits the average depth of generating a taxonomy, while the fourth one also limits each corpus size used for training a parent node of the taxonomy. We developed an experimental classification system for evaluating these algorithms. We conducted an experiment with at most 120 iterations for each taxonomy modification algorithm. The result showed that the third algorithm produced stable relevance metric scores, while the fourth one produced much better relevance scores than the others.

Contributions of the paper can be summarized as follows:

- presentation of a taxonomy modification algorithm with stable relevancy performance,
- presentation of a taxonomy modification algorithm with best relevancy performance,

- implementation of practical hierarchical classifiers that take the "local classifier per parent" approach [7] to directed acyclic graph (DAG)-structured taxonomies, and
- implementation of practical hierarchical classifiers that efficiently and relevantly predict documents into over 10,000 taxonomy classes.

II. Large-Scale Hierarchical Document Classification

A. *Hierarchical Classifier Specifications*

General definitions and categorizations of machine-learning document classifiers have previously been published [12], as have the results of experiments on various types of hierarchical taxonomy classifiers [7], [13]–[15]. According to their applications and requirements, hierarchical classification systems might have different problems, algorithms, and corpus arrangements. This often causes confusion when the performance of various classifiers are compared. To guarantee fair comparisons with the classification system we developed for this experiment, we represent the specifications using the unifying framework for hierarchical classification [7], which provides comprehensive and essential notations of hierarchical classification tasks and solutions.

The specifications of our system are described as follows:

$$\begin{aligned} < \Gamma, \Psi, \Phi > &= < D, MPL, PD > \\ < \Delta, \Xi, \Omega, \Theta > &= < SPP, NMLNP, D, LCPN >, \end{aligned}$$

where $< \Gamma, \Psi, \Phi >$ and $< \Delta, \Xi, \Omega, \Theta >$ specify the types of problems and algorithms of the framework, respectively.

The property of original taxonomy data is mainly affected by the problem specifications. '$\Gamma = D$' indicates the DAG taxonomy graph structure. '$\Psi = MPL$' indicates that data instances can have multiple paths of labels, and '$\Phi = PD$' indicates that data instances may have labels corresponding to interim nodes of taxonomies (partial depth labeling). Further description of the data is explained in Subsection II-C.

Our primary goal was to develop practical web-document classifiers for large-scale web directories. This application category majorly affected the determination of the algorithm specifications. '$\Delta = SPP$' indicates that the algorithm performs single path predictions. Although multiple path predictions might help some types of users, they may complicate algorithms and lose classifier's efficiency. '$\Xi = NMLNP$' indicates that the algorithm performs non-mandatory leaf-node predictions. It is normal for web pages to be categorized not only in leaf nodes but also in interim nodes of web directory structures. '$\Omega = D$' indicates that the algorithm can handle DAG-structured taxonomies, and '$\Theta = LCPN$' indicates that the type of algorithm is 'local classifier per parent node'. The reasons for choosing this type is discussed in Subsection V-C.

B. *System Implementation*

We developed an experimental classification system that consists of eight modules: taxonomy translator, crawler, feature extractor, training and evaluation corpus generator, trainer, predictor, evaluator, and taxonomy modifier. The *taxonomy translator* translates the whole category structure of the original web directory data into a taxonomy that can be processed by the system. The *crawler* accesses the Internet, fetches the contents of all the documents (URLs) categorized in the taxonomy, and saves them as HTML files. The *feature extractor* first divides each crawled HTML file into four parts: the title, body, meta-keywords, and meta-description. It then selects a content string from each part, tokenizes the string into a set of terms, and saves them as bag of words (BOW) features. If the content is written in Japanese, the tokenization is performed by applying a Japanese morphological analyzer. The *training and evaluation corpus generator* reads relational information between the nodes and documents in the latest taxonomy and builds a relationship table that combines each taxonomy node with documents for training and evaluation purposes. The *trainer* trains a linear-kernel support vector machine (SVM) model of a multi-class classifier for each parent node of the latest taxonomy using the saved BOW features of training corpus documents. The *predictor* generates a BOW feature from a URL or loads a saved BOW feature of the URL, performs local classification, and predicts a taxonomy node class. The *evaluator* applies the predictor to the test corpus, saves all the classification results, and calculates several evaluation metrics. The *taxonomy modifier* generates a new taxonomy from the classification results saved by the evaluator. The next generation of the taxonomy can be produced by repeating the process sequence after the *training and evaluation corpus generator*.

The system is executed on a common Linux PC server that has four 2.33-GHz Xeon CPUs and 64-GB memory. Each CPU has 4 independent cores. The PC has totally 16 cores. The codes are written in Shell, Perl, and Erlang with a total of 25,532 lines.

C. *Data Preparation*

The taxonomy used in this experiment was constructed from the Yahoo! Japan category (http://dir.yahoo.co.jp/) snapshot on May 2, 2007. The taxonomy translator generated the original taxonomy, which had a total of 85,791 classes. The crawler fetched 490,018 documents representing 90.6% of all the documents in the taxonomy. The trainer produced 25,747 models for parent nodes of the taxonomy, each of which had more than one child node. The taxonomy modifier generated taxonomies that have more than 200,000 nodes in the experiment.

Our training example assigning policy can be formalized as follows using notations described by Fagni and Sebastiani

[16] and Cai and Hoffmann [7]:

$$Tr^+(c_j) = \bigcup_{c \in \Downarrow(c_j)} Tr^+(c) \setminus \bigcup_{c \in \Downarrow(\downarrow(\uparrow(c_j))\setminus c_j)} Tr^+(c) \ . \quad (1)$$

Our classification, not taxonomy modification, algorithm used in the experimental hierarchical classifiers is classified as "local classifier per parent node" classification ($\Theta = LCPN$). Each parent classifier of this algorithm is a multi-class (not a binary) classifier. Only positive examples are explicitly assigned to class c_j for training its parent node $\uparrow(c_j)$ multi-class classifier. Because the handling taxonomy is a DAG structure ($\Omega = D$), child node class c_j may appear under different parent node classes ($c_{p1}, c_{p_2} \in \uparrow(c_j)$, $c_{p1} \neq c_{p2}$). Sibling class sets $\downarrow(\uparrow(c_j)) \setminus c_j$ may differ under different parent nodes c_{p1} and c_{p2}. Therefore, the example set $Tr^+(c_j)$ under c_{p1} and $Tr^+(c_j)$ under c_{p2} are not always identical in the policy (1). This issue impacts many subsystems of hierarchical classifiers. Our classification system was designed to handle the issue by introducing example class identifiers, each of which includes not only the target class identifier but also the parent class identifier.

III. Improvement Methods for Large-Scale Taxonomy

A. Tang et al.'s Methods

The basic idea behind the methods proposed by Tang et al. [9] is that taxonomy improvements can be achieved by iterating three types of primitive operations (promote, demote, and merge) for taxonomy modifications. The promote operation raises a target node to the same hierarchical level as its parent node. The demote operation lowers a target node to the same hierarchical level as its child nodes as a sibling. The merge operation gathers two sibling target nodes into a newly created sibling node. The most conservative approach to taxonomy improvements might involve the following steps: 1) investigate hierarchical classifier relevance performances of taxonomies that can be produced by all possible single operations from the initial taxonomy, 2) select the operation and target nodes that perform the best, 3) apply the selected operation to the taxonomy, 4) take the modified taxonomy as the next initial taxonomy, and 5) repeat from step 1). Because this approach is far from efficient, Tang et al. proposed two heuristic methods [10] that perform more efficiently: a greedy approach, which selects possible single operations according to the classification result statistics of the initial taxonomy, and a local approach, which prioritizes target nodes according to the hierarchical structure of the initial taxonomy.

B. Accumulate Primitive Operations Algorithm

The two heuristic methods described above require training and evaluation processes for each taxonomy modified by an operation candidate before the operation is actually applied to the taxonomy. The total computational cost for those processes increases drastically according to the scale of the taxonomy. While the cost of training processes might be reduced by limiting the training node classifiers to only ones affected by single taxonomy modification operations, the cost of evaluation processes cannot be easily reduced. Local classification results might be affected by modifications of distantly located nodes, especially if those nodes belong to the upper positions in the taxonomy. The 'upper positions' mean the positions of taxonomy nodes located near the root node. Therefore, we developed four algorithms for modifying large-scale taxonomies by extending Tang et al.'s methods to drastically reduce the training and evaluation costs.

The first modification algorithm extended for large-scale taxonomies, *accumulate primitive operations (APO)*, consists of processes that find promote, demote, and merge operation candidates from all nodes in the initial taxonomy, solve conflicts among them, simultaneously apply those operation candidates to produce next-generation taxonomies, train a hierarchical classifier using the taxonomy, and evaluate the classifier. These processes reduce the number of the evaluation process iterations to one for generating the next taxonomy, while Tang et al.'s methods might iterate the evaluation process $O(n)$ times, where n is the size of the taxonomy.

C. Avoid Child Concentration Algorithm

Although the APO algorithm improved the efficiency of taxonomy improvement processes, it worsened training and predicting process efficiency. The later generations of taxonomies modified by the APO algorithm tend to concentrate child nodes to particular parent nodes. Nodes holding many classes and training documents require enough memory to maintain large number of support vectors in training and predicting processes of parent multi-class SVM classifiers. Such nodes also consume certain computational time. Therefore, we developed the *avoid child concentration (ACC)* algorithm to avoid the child concentration phenomena by extending the APO algorithm. The difference is that the new algorithm has an additional constraint in selecting promote candidates. The constraint limits each parent node to having at most a predefined number of child nodes.

D. Limit Average Depth Algorithm

The ACC algorithm solved the problem of training and predicting process efficiency by forcing the taxonomy structure not to spread widely. This results in deepening of the structure. Local classifiers have a weak point in that relevancy scores might worsen by accumulating errors through the path of parent node classifiers. The deeper structure enhances this weak point. Therefore, we developed the *limit average depth (LAD)* algorithm by restricting primitive operations that produce deeper structures. This algorithm

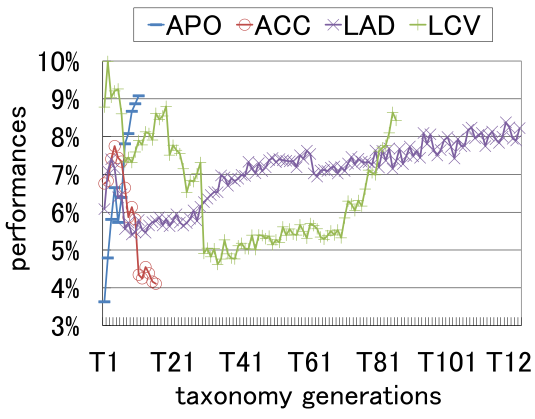


Figure 1. Flat f1 scores of classifiers generated using APO, ACC, LAD, and LCV algorithms

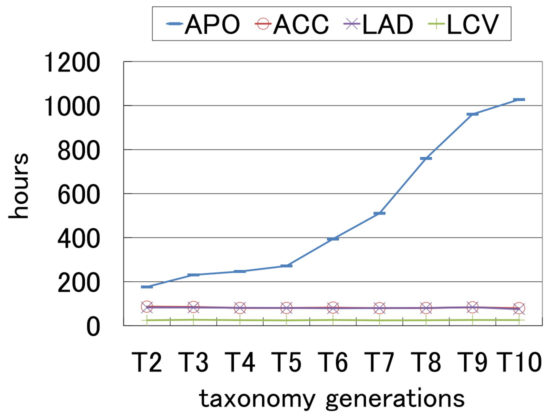


Figure 2. Predicting process efficiency of classifiers generated using APO, ACC, LAD, and LCV algorithms

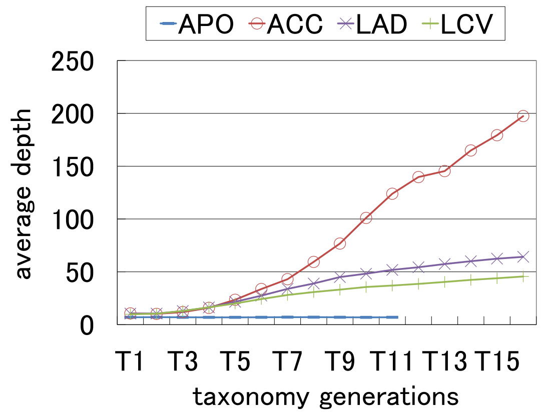


Figure 3. Average depths of taxonomies generated using APO, ACC, LAD, and LCV algorithms

differs from the ACC algorithm in definitions of the demote and merge candidate node set. The LAD algorithm may eliminate nodes, which are located at depths deeper than the predefined depth, from the candidates node set.

E. Limit Corpus Volume Algorithm

The LAD algorithm has two contradicting restrictions for selecting primitive operations. While the restriction for the number of child nodes deepens the taxonomy structure, the depth restriction attempts to increase the number of child nodes for several parent nodes. The contradicting restrictions decrease the number of primitive operation candidate nodes and require many taxonomy generations in order to improve relevance performances. Our detailed observation of taxonomy modification experiments revealed that the size of the training corpus affects training and predicting process efficiency more directly than the number of child nodes. Therefore, we developed the *limit corpus volume (LCV)* algorithm by primarily restricting the size of the corpus used for each parent node to train its classifier. Although it also applies the restrictions used in the LAD algorithm, the parameters are partly loosened for the primitive operation candidate sets to obtain more freedom to improve relevance efficiency.

IV. Empirical Results

Figure 1 shows flat f1 scores of hierarchical classifiers, whose companion taxonomies are generated by the APO, ACC, LAD, and LCV algorithms. On the X-axis, $T1$, $T2$, $T3$, $T4, \ldots$ denote generations for the 1st-, 2nd-, 3rd-, 4th-, $\ldots$ taxonomies, respectively. The 'flat f1' scores are macro averaged f1 scores of taxonomy nodes, each of which has at least one expected document and one predicted document. The APO algorithm iteration was terminated at the 11th generation because the computational cost for training and predicting processes had became too large. The predicting process efficiencies for the first ten generations are shown in Figure 2. The ACC algorithm iteration was terminated at the 16th generation because the depth of the taxonomy structure had become too deep. The average depths of taxonomies generated by the APO, ACC, LAD, and LCV algorithms are shown in Figure 3. There is no significant reason for the termination of the LAD and LCV algorithms.

The metrics of flat precision, recall, and f1 scores might be too strict for relevance performances of hierarchical classifiers. More adequate metrics, hierarchical precision, recall, and f1 scores, have been proposed [7]. Because we had accidentally lost all raw classification results of the APO, ACC, and LAD algorithms, only the LCV algorithm results could be processed to calculate for those metrics. The hierarchical f1 scores of classifiers generated for the LCV algorithm are shown in Figure 4.

V. Discussion

A. Stability of Relevance Performance

The LAD algorithm generated a sequence of taxonomies, which produce hierarchical classifiers exhibiting stable relevance scores, as shown in Figure 1. Although each process

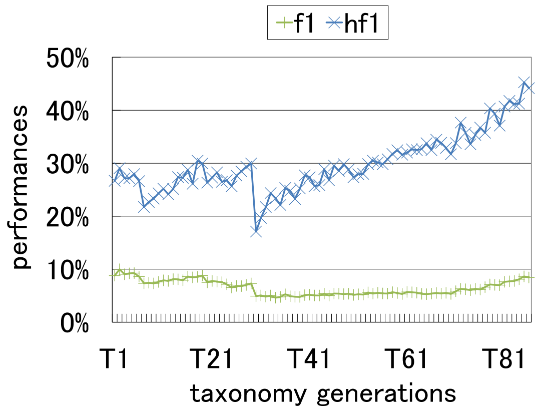


Figure 4. Flat f1 score (f1) and hierarchical f1 score (hf1) of classifiers generated using LCV algorithm

for generating a new taxonomy modifies a small number of nodes compared to other algorithms, the LAD flat f1 scores exhibited small improvements and reached the best record of such scores sometimes even after the hundredth generation. While the LAD algorithm has strict restrictions for solving the child node concentration and deep structure problems, it sometimes found a successful primitive operation candidate set. The LAD algorithm will perform well when that a) the original taxonomy structure is stable and b) the user has leeway to compute enough taxonomy modification iterations.

B. Best Relevance Performance

The LCV algorithm generated the 2nd generation taxonomy, which produced a hierarchical classifier that attained the best flat f1 score record (9.99%) of the experiment, as shown in Figure 1. The figure also indicates that the performance of this algorithm is not stable. There were several rapid depressions at the 7th, 12th, and 13th generations. The determinant reason for the depressions is currently under investigation. The major differences between the LCV from LAD algorithm were

1) avoiding corpus concentration and
2) loosening the parameter for child concentrations.

Differences 1) seems to play the same role as the restriction for avoiding child concentration, and succeeded in maintaining training and predicting process efficiency at the practical level, as shown in Figure 2. The effects of difference 2) seems to be both good, recording the best relevancy score, and bad, unsteadiness of relevancy scores of modified taxonomy sequences. The LCV algorithm should be applied when a) the original taxonomy structure is unstable and b) classification systems must be developed rapidly.

Figure 1 shows that the LCV algorithm performed the best at the 2nd generation measured using flat f1 scores. Figure 4 shows that the algorithm performed the best at the 85th generation measured using the hierarchical f1 scores. This means that the hierarchical classifier for the 85th generation taxonomy miss-classified more times than the 2nd taxonomy classifier, but the degree of miss-classification of the 85th one was less significant than the 2nd one. The flat f1 scores penalize miss-classifications uniformly. Nevertheless, the hierarchical f1 scores penalize them heavily if each one predicts a document to the more distanced node from its expected node. The distance-based loss scores [17] are commonly used to measure the degree of miss-classification. We could not use the loss scores successfully because it was difficult to combine the score with other relevancy scores such as f1. The hierarchical precision, recall, and f1 scores naturally combine relevance performance metrics with hierarchical penalty metrics. The criteria of selecting between the flat and hierarchical relevance metrics depends on the types of classification applications. If an application emphasizes the number of documents that were predicted to the exact matching nodes, the flat metrics should be used for evaluating classifiers. If another application tolerates miss-classification between nearby leaf-level sibling nodes, the hierarchical metrics should be used. We consider the web document classification applications as the latter type.

C. Classifier Design Selection

While we selected the ‘local classifier per parent node’ type classification algorithm ($\Theta = LCPN$), Silla and Freitas [7] presented other types as ‘local classifier per node’ (LCN), ‘local classifier per level’ (LCL), and ‘global classifier’ (GC). Implementations of the LCN algorithm locate local binary classifiers at all nodes that have the possibility of being predicted. The number of LCN local classifiers is always greater than that of the $LCPN$ algorithm. The local strategy hierarchical classifiers designed for large-scale taxonomies must have a large number of local classifiers. This is a fatal disadvantage for training and predicting process efficiency. Considering that we are using SVM-based multi-class local classifiers that consist of optimized combinations of binary classifiers, it might be feasible to apply the LCN algorithm with careful selection of the training corpus. The LCN classifiers tend to be trained using unbalanced corpus that consist of a few positive examples and a huge number of negative examples. It requires additional processing time to improve relevancy performance. The LCL and GC classifiers have to simultaneously treat many prediction classes. When they are applied for large-scale taxonomies, they increase the size of model, training time, and prediction latency.

D. Performance Limits of Large-Scale Hierarchical Classifiers

There are several issues to improving relevance and efficiency performances that have not been applied to our experimental hierarchical classifiers. While the classifiers might not exhibit the best relevance or efficiency, they seem to balance better relevance and efficiency performances considering they deal with large-scale taxonomies and are executed on a commonly available server machine. The classifier design choices of the feature extraction (full BOW) and the classification algorithm ($\Theta = LCPN$) support balanced performance improvements. A large number of the improvements was achieved using the taxonomy modification method. The 11th generation taxonomy of the APO algorithm produced a classifier that resulted in 9.08% flat f1 value, as shown in Figure 1. Nevertheless, the efficiency of the classifier is inferior to all other evaluated hierarchical classifiers, as shown in Figure 2. One of the LCV classifiers exceeded the relevancy performance by maintaining practical efficiencies, as shown in Figure 1. The LCV classifier gives an example of practically executable large-scale hierarchical classifiers, each of whose taxonomy has more than 10,000 classes.

VI. Conclusion

We focused on taxonomy modification algorithms that gradually improve the relevance performances of large-scale hierarchical classifiers of web documents. Considering the research results from Tang et al. [9], [10], who took the same approach, we investigated and implemented four taxonomy modification algorithms that aggregate primitive operations before evaluating hierarchical relevance performances. The LAD algorithm generated a sequence of taxonomies that produce classifiers exhibiting stable relevancy performances. The LCV algorithm generated a taxonomy that produces a classifier that resulted in the best flat f1 score (9.99%) in our experiment. The hierarchical classifiers executed in the experiment used the “local classifier per parent” approach [7] and classified documents into DAG-structured taxonomies. The system we developed for this experiment reached the level of practical hierarchical classification systems that relevantly and efficiently predict documents into taxonomies containing more than 10,000 classes.

References

[1] M. Ceci and D. Malerba, “Classifying web documents in a hierarchy of categories: a comprehensive study,” *J. Intell. Inf. Syst.*, vol. 28, no. 1, pp. 37–78, 2007.

[2] S. C. Gates, W. Teiken, and K.-S. F. Cheng, “Taxonomies by the numbers: building high-performance taxonomies,” in *CIKM ’05: Proceedings of the 14th ACM international conference on Information and knowledge management*. New York, NY, USA: ACM Press, 2005, pp. 568–577.

[3] Y. Labrou and T. Finin, “Yahoo! as an ontology: using yahoo! categories to describe documents,” in *CIKM ’99: Proceedings of the eighth international conference on Information and knowledge management*. New York, NY, USA: ACM, 1999, pp. 180–187.

[4] D. Mladenic, “Turning yahoo into an automatic web-page classifier,” in *Proceedings of the 13th European Conference on Aritficial Intelligence ECAI’98*, 1998, pp. 473–474.

[5] A. Sun and E.-P. Lim, “Hierarchical text classification and evaluation,” in *ICDM ’01: Proceedings of the 2001 IEEE International Conference on Data Mining*. Washington, DC, USA: IEEE Computer Society, 2001, pp. 521–528.

[6] D. Koller and M. Sahami, “Hierarchically classifying documents using very few words,” in *ICML ’97: Proceedings of the Fourteenth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 170–178.

[7] C. N. Silla, Jr. and A. A. Freitas, “A survey of hierarchical classification across different application domains,” *Data Min. Knowl. Discov.*, vol. 22, no. 1-2, pp. 31–72, Jan. 2011.

[8] T. Li, S. Zhu, and M. Ogihara, “Hierarchical document classification using automatically generated hierarchy,” *J. Intell. Inf. Syst.*, vol. 29, no. 2, pp. 211–230, 2007.

[9] L. Tang, J. Zhang, and H. Liu, “Acclimatizing taxonomic semantics for hierarchical content classification from semantics to data-driven taxonomy,” in *KDD ’06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM Press, 2006, pp. 384–393.

[10] L. Tang, H. Liu, J. Zhang, N. Agarwal, and J. J. Salerno, “Topic taxonomy adaptation for group profiling,” *ACM Trans. Knowl. Discov. Data*, vol. 1, no. 4, pp. 1–28, 2008.

[11] K. Nitta, “Improving taxonomies for large-scale hierarchical classifiers of web documents,” in *Proceedings of the 19th ACM international conference on Information and knowledge management*, ser. CIKM ’10. New York, NY, USA: ACM, 2010, pp. 1649–1652.

[12] F. Sebastiani, “Machine learning in automated text categorization,” *ACM Comput. Surv.*, vol. 34, no. 1, pp. 1–47, 2002.

[13] S. Dumais and H. Chen, “Hierarchical classification of web content,” in *SIGIR ’00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*. New York, NY, USA: ACM Press, 2000, pp. 256–263.

[14] T.-Y. Liu, Y. Yang, H. Wan, H.-J. Zeng, Z. Chen, and W.-Y. Ma, “Support vector machines classification with a very large-scale taxonomy,” *SIGKDD Explor. Newsl.*, vol. 7, no. 1, pp. 36–43, 2005.

[15] A. Kosmopoulos, E. Gaussier, G. Paliouras, and S. Aseervatham, “The ecir 2010 large scale hierarchical classification workshop,” *SIGIR Forum*, vol. 44, no. 1, pp. 23–32, Aug. 2010.

[16] T. Fagni and F. Sebastiani, “On the selection of negative examples for hierarchical text categorization,” in *Proceedings of the 3rd Language Technology Conference (LTC 2007)*, 2007, pp. 24–28.

[17] L. Cai and T. Hofmann, “Hierarchical document categorization with support vector machines,” in *CIKM ’04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*. New York, NY, USA: ACM Press, 2004, pp. 78–87.

Data Management According to the Good Scientific Practice

Jan Potthoff, Marius Walk, Sebastian Rieger
Steinbuch Centre for Computing
Karlsruhe Institute of Technology
Karlsruhe, Germany
[name.surname]@kit.edu

***Abstract*—Data Management in scientific and research processes needs to comply with rules defined in the Good Scientific Practice polices, e.g., issued by scientific institutes and research organizations. Typically these regulations include definitions to protect the data privacy and security throughout the entire lifecycle of the scientific data. Beginning from the generation of the data and ensuring its provenance up to the final archival and corresponding long-term data storage management. This paper introduces a simple way to manage the scientific data that ensures long-term protection and provability of the data quality. Furthermore an easy to use implementation which offers a possibility to structure the scientific data and to archive it according to Good Scientific Practice is presented.**

Keywords-ELN; scientific data; data formats; long-term preservation; evidence

I. Introduction

During the research process scientists have to deal with electronic data and its management. Complex applications like electronic laboratory notebooks (ELN) and laboratory information and management systems (LIMS) are developed to support the scientists during day-to-day work. In addition several rules, e.g., the good scientific practice (GSP) defined by scientific institutes and research organizations, have to be respected in the research process and the entire lifecycle of the scientific data. In some research areas, e.g., social science or computer science, the documentation with a laboratory notebook is not common and sometimes inefficient regarding the amount of data that is processed in the research process. In this case the documentation of the research process is typically not based on laboratory notebooks and applications such as ELN or LIMS are less frequently used. Nevertheless a data management according to GSP is needed.

The paper is outlined as follows. Section II describes the requirements for good scientific practice (GSP) and its implications for data management. In Section III different forms and formats of scientific data and related work in this research area are shown. Using the requirements given in Section II and the various forms of scientific data described in Section III, Section IV focusses on appropriate research data management tools and practices. Section V contains a software implementation that facilitates data management according to GSP requirements introduced in this paper. Finally in Section VI a conclusion, major strengths and weaknesses of the solution along with future work are presented.

II. The Good Scientific Practice

A sustainable documentation of the research results requires keeping several rules, e.g., the rules of GSP. The intention is to guarantee a high quality in the research area and the work of scientists to prevent scientific deception or fraud. The spectrum of scientific misconduct ranges from several violations of scientific ethics to criminal intentions [1]. Hence, the transparency in dealing with primary data is a basic claim of GSP. With the term “primary data” all data from an experiment or a scientific survey is covered [1].

By the rules of GSP the following requirements are addressed: traceability, long-term interpretability and sustainable archiving. The traceability is deemed necessary when the scientific process is documented in a way such that the results need to be completely reproducible. Hence, a long-term interpretability to understand and reproduce the results is important as well. For the sustainable archiving, corresponding techniques are required. For example, the German research foundation (DFG) requires archiving for 10 years [2]. The DFG proposes the documentation based on a laboratory notebook, but nowadays especially for large amounts of data digital archiving is needed as well. In addition the rules of the DFG are used by several organizations as well [1][3]. Especially for the work in a laboratory there are further regulations, e.g., the good laboratory practice [4]. There are also legal regulations, for example in the healthcare sector, e.g., rules with regard to the archiving period for documents like the digital patient file.

So, the data management in the research process has not only to deal with data structures, but also with several regulations relating to the respective research areas.

III. Scientific Data

Establishing policies for GSP has an influence on the data being generated and processed in laboratories and scientific processes. The following sections give an overview of typical forms and formats of scientific data and related work that focusses on the proper management of scientific data and metadata.

A. *Data Formats and the Long-term Preservation*

In the research process electronic data is generated and used in varying volumes depending on the research field and approach. Based on this data existing research results are confirmed or new approaches are elaborated. For example, in social science, data, which was generated in the research process, is used, e.g., in surveys, experiments or observations, or in non-scientific works, e.g., official statistics [5]. The scientific data and its data formats vary over different kinds of approaches and the software used in the research area.

In experimental research electronic data is generated in varying file formats in each process phase as well. For example, at the Max Planck Institute for Dynamics and Self-Organization (Göttingen, Germany) experiments are carried out in a wind tunnel using three high speed cameras. Each of these cameras generates 10,000 images per second [6]. While digital cameras usually take pictures in widely-used file formats other measurement instruments generate data whose proprietary format is specified by the manufacturer. It is even possible that the format is dependant on the device. This could be common formats in ASCII code or XML, but also complex formats specified individually by applications.

The scientific data is managed using its metadata in the research process. With the metadata the corresponding data will be described (descriptive metadata) and technical information will be documented (administrative metadata).

Although standards such as Dublin Core, Metadata Object Description Schema (MODS) or Metadata Encoding and Transmission Standard (METS) aim to uniform the format of metadata, the volume and form of metadata strongly vary throughout the research areas [7].

Metadata is very important for the long-term preservation as well. Using the metadata, the corresponding data can be found even if a unique identifier does not exist, is unknown or has been lost. Therefore the long-term preservation data formats such as XML formatted Archival Information Package (XAIP) or the universal object format (UOF) usually contains a combination of data and metadata. For example, XAIP was designed for an archive system, whose construction is based on the technical directive of the Federal Office for Information Security (BSI, Germany) [8]. The archival information package is an XML file that contains the data in the Base64 format and the corresponding metadata. In the UOF the data is stored as a file in a tar archive instead of the Base64 format. To save the metadata, a separate file, based on METS, is also stored in the tar archive [9].

B. Related Work

The data handling in the research process [7] and the requirements of the scientists [10] are current research topics. Because of the amount of information and data [28], respectively, the goal is to find a way to support the scientists in their day-to-day work [30].

Many software applications for the documentation of the research process, e.g., ELN or LIMS, with different emphasis are available today [11]. The requirements of scientific data management are addressed in several research projects [12][13]. Even if the documentation varies with the research area, all research fields need sustainable archiving of the scientific data [14]. The use of designed applications and archiving solutions has to be kept simple [10].

We address these requirements with an intuitive and simple application that will be described in Section V.B. Using this application a simple data collecting and management is achieved as well as a sustainable archiving mechanism. Other projects in the same area can be found, e.g., in [27][29]. Compared to our solution these approaches do not offer explicit protection of the probative force of the scientific data. Somehow [27] focusses on the provenance of scientific data without implementing long-term preservation of the probative force. Our solution offers a small light-weight client that can be simply used like a file explorer by the scientists. In addition, by using the BeLab system as basis for our implementation, as explained in Section V.A., the data will be long-term archivable according to GSP.

IV. Research Data Management

As described in Section III.A, data and corresponding metadata is produced and used in a variety of formats throughout scientific processes. As the processing and interpretation of the data is essential for generating results from the scientific process (e.g., for publications), efficient storage and management of the generated research data is necessary in scientific environments. Several management tools and frameworks for scientific data have been developed in the last decades. Especially web-based information and document management tools have evolved and extended to fulfill the requirements of scientific processes. To allow a profound integration with specific scientific processes a new category of software products has been formed, e.g., ELN, that is described in the next section. These tools enhance the management of research data by facilitating its retrieval and processing beginning from the generation (e.g., by directly importing data from sensors during experiments), modification (e.g., data processing or manual interaction) up to publication, archival and deletion of data forming the scientific data lifecycle.

A. Electronic Documentation

The increased use of computers and the corresponding amount of electronic data led to the need of electronic documentation in the research process. By using a paper-based laboratory notebook and storing the electronic data separately, the danger of losing data is increased. Nevertheless this kind of documentation is used in several institutions [15]. A central electronic data management reduces these risks and also offers further advantages. For example, a collaboration with other colleagues is facilitated, the search for data becomes easier and faster and the research process gets more efficient [16].

To implement a central approach of documentation and data handling a corresponding software solution is needed. Hence, various systems are available on the market today. ELN can be defined as "a secure system assembling scientific content from multiple sources related to each other, allowing for contextual annotation, and packaging it in a legally acceptable document to be searched, mined and collaborated" [17]. In general, ELN can be understood as "an electronic embodiment of what is currently being done in a paper laboratory notebook" [18] whereas LIMS is more integrated in the research process. It offers the possibility to collect data from connected measurement instruments, such that the systems naturally have to deal with more scientific data. Additionally, LIMS is used for administrative processes as well as for documentation. It is also possible to integrate a separate ELN in LIMS. A clear separation between these systems is therefore difficult because they deal with similar use cases and the precise definition depends on the range of use [19].

In addition to the applications named here, other programs that are not designed for this purpose are used in the research process. For example, web-based information and document management systems (e.g., Wiki systems) or even simple office applications are used [20]. In other cases, an individual ELN solution has been developed by the research group. For example, the solution "open enventory" has been developed as an administrative system for chemical substances in the first phase. Later on the system has grown to an ELN that is tailored to address the requirements of research groups in chemistry [21].

B. *Practical Requirements*

As the workflows that generate and process data in institutions and groups differ, it is hard to establish a unified data management, e.g., using ELN and LIMS as described in the previous section. The trade-off between individual optimization of the data processing workflows and using standardized software solutions holds for both scientific and corporate scenarios. Often the users simply want to archive the data during or at the end of the scientific process. Moreover not every scientific institute has the necessary resources to implement ELN or LIMS systems for their research environments.

Establishing GSP for scientific data does not require specific frameworks like ELN or LIMS. The benefit of using ELN and LIMS regarding the GSP depends on the integration of workflows and, i.e., measuring the probative value of the data in an early phase during the data generation in the scientific process (e.g., combing and verifying information from different sensors and user interaction). However this is typically not required by GSP regulations. Additionally it is nearly impossible to get the entire raw data during its generation and verify it in real-time, i.e., because of its increasing volume, e.g., due to high resolution sensors. For example, the regulations by the DFG require the scientific institutions to protect the integrity of the data on the long-term, e.g., while being stored in a digital archive, and not to include the data of every single sensor and verify the entire workflow of data generation which would be rather complex both during archival and verification. This could also be achieved by checking the integrity and sign the data before archiving it using a simple client that is able to store generic files and thus data and metadata containers. To allow a long-term preservation of the data and proving its probative value, standards for the signature and archival format should be used. Also the client application should offer a simple way for verifiers and scientists to search retrieve and verify the data. It should also handle the evaluation of the probative value and the import and export of data and metadata extraction in a highly automated manner. The effort necessary to evaluate the probative value and archive the data should be as minimal as possible to allow for a higher acceptance. The majority of the data is stored in files, so the application should be file-based.

V. Uniform Data Preparation

As described in Section III, specialized applications can be used for the documentation of the research process and to manage the scientific data. For many research areas these applications are over-sized, because the scientist simply has to deal with a few files during a workday. In this case individual programs, e.g., office applications, are used [20]. However, this case requires a solution to address the regulations of the GSP. In the following section a system that can be used for the long-term preservation of the scientific data and its probative value is described. An application that is based on this system is presented in Section V.B. The application enables the scientist to use the system in an easy way to manage the data and to ensure the GSP requirements.

A. *Evidential Long-term Preservation*

In many research areas a long-term preservation of scientific data is required, e.g., by German law or internal regulations of research organizations like the GSP, as described in Section II. The goal of the BeLab project founded by DFG is to develop a concept for the long-term preservation of scientific data and its probative value to secure the quality of complex data. Therefore the requirements of the long-term preservation of scientific data, the probative value of electronic data and the possibilities of its conservation are analyzed.

Even if paper-based laboratory notebooks are still used in the scientific process [15], today's scientific documentations include an increasing amount of electronic data. Also, in research areas in which the documentation is typically not based on the use of an ELN or a paper-based laboratory notebook, the volume of electronic data is constantly growing, as described in Section III.

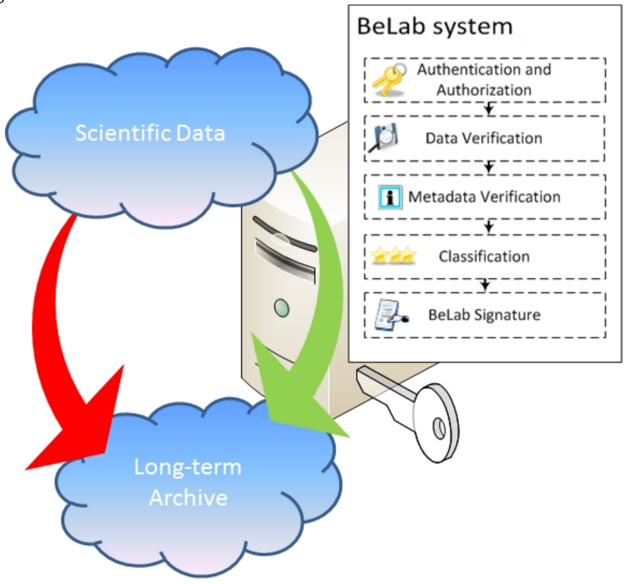


Figure 1. The BeLab system.

Electronic documents are considered to be insecure and to have less probative value in comparison to digital certificates relating to the German law [22]. By using electronic security technologies, the probative value can be increased to the same level as in paper-based documents. Especially to secure the integrity and authenticity, digital signatures can be used, based on the German Digital Signature Act. In the BeLab project a concept for a middleware (BeLab system) was designed and implemented. The BeLab system, as shown in Figure 1, can be used from various specialized applications, e.g., ELN, to prepare the scientific data for the evidential long-term preservation. After that the scientific data will be submitted to a connected archive system [23]. For example, the archive can be used as mentioned in Section III.A.

The technical directive (BSI TR-03125) defines an archives system, which is able to update digital signatures before they are invalid [8]. To enable a uniform processing of the data and evaluation of its probative value within the BeLab system, a data model based on UOF is used. Therefore the files to be submitted will be packed in a TAR or ZIP archive. Additionally, a metadata file (mets.xml) will be separately added to the archive [23]. The structure of the metadata file is based on METS that was designed to ensure a uniform management and exchange of electronic data [24].

The BeLab system executes verification modules, which are based on the file format [23]. These modules analyze the probative value, the suitability for long-term preservation and the characteristics of the data generation. The verification of the probative value is mainly based on the use of digital signatures but also on hash values of files being submitted, that are defined in the metadata file. For the verification of the suitability for the preservation, the current file format is analyzed and a verification of the data generation is performed.

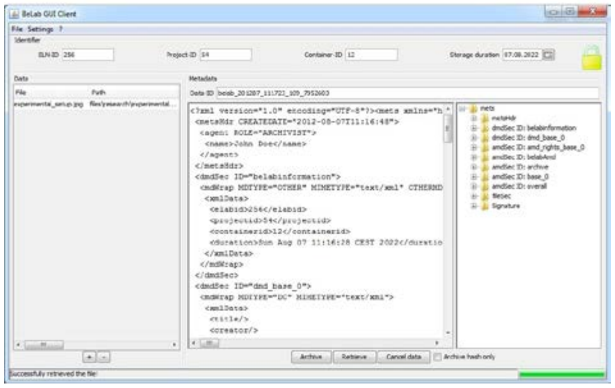
In a subsequent classification process, as depicted in Figure 1, the result of the data and metadata verification is mapped to a category, which describes the degree of the probative value, the suitability for the long-term preservation and the security measures during the data generation [23].

The results of the data and metadata verification are recorded together with the classification in a log file as well as in the updated metadata. To ensure the integrity of the metadata file the results are added in a separate copy (belab.xml) of the file. The copy of the metadata will be signed by the BeLab system and is added to the data model (BeLab object), which contains the scientific data in the UOF. Using this digital signature the integrity and authenticity can be verified while the data is stored in the archive. Finally, the BeLab object will be submitted to the connected archive system. The user gets a unique identifier, which represents the archived data model. Additionally, the BeLab object is managed with an individual project, ELN and container identifier being defined by the user [23].

By using the BeLab system to archive the scientific data, as shown in Figure 1 with the right arrow, the scientist gets useful information about the probative value and the suitability for long-term preservation of the data being submitted, that would not be available upon archiving the data without any data verifications, as shown in Figure 1 with the left arrow.

B. Data and Metadata Detection

Based on the results of the BeLab project a client application was implemented, which offers a possibility for the management of scientific data and detection of corresponding metadata in the research process. Instead of a complex ELN or LIMS the client was designed for a simple use case which allows the scientist to archive the data with respect to the GSP, as described in Section IV.B. The graphical user interface is divided in three sections, as shown in Figure 2. In these sections the user has the possibility to collect files to be archived or to check out archived files and edit them. The section of metadata refers to the structure of data, i.e., project, ELN and general container id. Further metadata can be indicated for each added file in the data section, e.g., document title, author and creation date. This metadata is used by the archive system to manage the data and in the BeLab verification process, as described in Section V.A.. For example, the author will be compared with the given author in the document, e.g., a Word file.



ELN-ID 256 Project-ID 54 Container-ID 12 Storage duration 07.08.2022

File: experimental_setup.jpg Path: files/research/experimental...

Data ID belab_201207_111722_109_7952603

```
<?xml version="1.0" encoding="UTF-8"?><mets xmlns="h
 <metsHdr CREATEDATE="2012-08-07T11:16:48">
  <agent ROLE="ARCHIVIST">
   <name>John Doe</name>
  </agent>
 </metsHdr>
 <dmdSec ID="belabinformation">
  <mdWrap MDTYPE="OTHER" MIMETYPE="text/xml" OTHERMD
   <xmlData>
    <elabid>256</elabid>
    <projectid>54</projectid>
    <containerid>12</containerid>
    <duration>Sun Aug 07 11:16:28 CEST 2022</duratio
   </xmlData>
  </mdWrap>
 </dmdSec>
 <dmdSec ID="dmd_base_0">
  <mdWrap MDTYPE="DC" MIMETYPE="text/xml">
   <xmlData>
    <title/>
    <creator/>
```

```
mets
  metsHdr
  dmdSec ID: belabinformation
  dmdSec ID: dmd_base_0
  amdSec ID: amd_rights_base_0
  amdSec ID: belabAmd
  amdSec ID: archive
  amdSec ID: base_0
  amdSec ID: overall
  fileSec
  Signature
```

Successfully retrieved the file!

Figure 2. Grafical user interface of the BeLab client.

The generation of an archive according to the UOF that was described in the previous section can be performed by adding files to the data section. This is done with the plus symbol below the file table. After all files have been added and all needed or desired metadata has been entered, the data collection will be automatically converted to the UOF object and will be send to the BeLab system by pressing the button "archive". It should be noted that the file format is very important for proper long-term preservation. Here, all file formats will be accepted. But by using the universal object format, the possibility of file migration is given. The migration has to be supported by the archive system.

The BeLab system will convert the UOF object to the BeLab data model. After that the BeLab system will evaluate the data, as described in Section V.A., and submit it to the integrated archive system. After the data model was stored in the archive the client application will automatically retrieve and display the generated BeLab object from the archive. To retrieve the data the corresponding identifier that was returned by the BeLab system is used.

The main focus of the client application lies on the representation of the metadata. This includes the metadata that was specified by the user as well as the metadata that was generated during the BeLab process, as described in Section V.A. In addition, the unique identifier of the corresponding archive will be displayed. The metadata will be displayed in a text field as well as a tree structure. By selecting a tree node, the corresponding content will be displayed as XML in the text field, e.g. the log file which has been written during the BeLab process.

Only the part of the digital signature and the BeLab classification will be presented in a different way. The classification will be shown as a table that contains all filenames and the corresponding probative values that were calculated by the BeLab system, as described in Section V.A. The validation of the digital signature will not only be represented as a detailed list in the text field, but also as a lock symbol at the top of the window. An open lock is associated with an invalid signature whereas a closed lock stands for a valid signature. Also the background color of the text field will be green if the signature is valid.

In addition to the representation in the application, the information can be exported to a PDF report. This report contains the metadata as well as an evaluation of the documented hash values and the result of their verification. Hence the integrity of the files can be verified and demonstrated, respectively.

To retrieve an existing archive file the corresponding unique identifier has to be entered into the data id field. By clicking the button “retrieve” the data will be requested and represented after a successful authentication and authorization of the user. Then it is possible to edit the data and metadata again. For example, files can be added or removed from the file table and archive object, respectively. To update the corresponding object in the archive, the “archive” button can be clicked again. The new data will be stored with an increased version number. To update data that has already been archived previously, the old data identifier has to be entered into the data id field.

Additionally, a data object can be deleted from the archive by using the client application. It should be noted that the corresponding object will only be marked as deleted instead of being physically removed from the archive. Using a further metadata item the storage duration can be indicated. After this time the corresponding object will be automatically canceled by the archive system.

In cases in which the files cannot be submitted to an external system, e.g., to ensure data protection or confidentiality of private data or because the amount of data is too large, the client application offers the possibility to submit only the corresponding unique hash values to the BeLab system. In this case the user needs to take care of the long-term preservation of the data. It should be noted that the solution has not been designed for large scale data management, but rather for research areas with limited amount of data.

For the data transmission, the user can choose between two transfer protocols. The first option is the Simple Object Access Protocol (SOAP) [25]. SOAP supports an authentication using a password or certificate. The second alternative is the REpresentational State Transfer (REST) [26]. Here, only the password-based authentication is possible.

In research areas in which the documentation based on a laboratory notebook is not common, the presented solution can be used to structure the scientific data and finally submit it to an archive according to the rules of GSP.

VI. Conclusion and Future Work

In this paper we introduced a simple client application that allows preserving GSP requirements as defined, i.e., by the German Research Foundation (DFG) [2] or corresponding regulations issued by scientific institutions (e.g., [1]). The client connects to a web service (the BeLab system as described in [23]) that evaluates the data and metadata submitted by the client regarding its probative force and suitability for long-term storage. After the evaluation, the data is digitally signed to protect the integrity and authenticity of the data while being stored in a long-term archive. Results of the evaluation are delivered back to the client. The client can be used later on to prove the probative force, consistency and integrity of the scientific data using the embedded digital signature. Compared to existing information management solutions and specific scientific data management tools like ELN and LIMS, the client offers a simplified way to ensure integrity and authenticity of scientific data. It can be easily integrated into every scientific process or workflow that produces or processes data in form of files. Moreover most of the existing tools that support the data management in scientific workflows do not support the protection of integrity and authenticity of the data or its long-term interpretability. As existing metadata standards are used, the long-term interpretability of the results of the evaluation (and the data) is addressed.

Together with other scientific institutions the project currently identifies an integration of the client and the BeLab web service into existing scientific processes. A future enhancement of the client will therefore focus on the automatic usage of the client. One example could be to automatically submit files to the BeLab system using predefined rules as they are being created in a directory monitored by the client. Furthermore, lab equipment could be connected to the client using custom or industry standard interfaces. This way additional metadata context could be provided to allow further evaluation of the probative force in the BeLab system. To be used as a generic tool in a variety of scientific processes and research areas, the support for specific data formats and interfaces of lab equipment have to be enhanced in future versions. If the client collects metadata throughout the entire research process, the integrity of the workflow could be proven later on by verifying the digital signature attached to the evaluation results stored in the archive. NoSQL databases for the storage of scientific data, as described, e.g., in [28], could also be a promising option. Especially the document-oriented types are well suited to store scientific data and metadata. Extensions to evaluate the probative force, to store the digital signature, as described for our solution, and to retrieve and check the integrity have to be developed to ensure GSP with NoSQL databases.

Another field for future research could be the usage of the stored data and metadata as evidence during trials. In Germany a specific procedure is being developed to receive evidence records for PDF files using e-mail and digital signatures. The client, while being used by an auditor or reviewer, could automatically send the signed result of the verified metadata to an e-mail address that can be used by the judges or attorneys during the trial. The probative force of the data and metadata can be enhanced even further by including identification, authentication and digital signature mechanisms of digital passports carried by scientists.

REFERENCES

[1] Karlsruher Institute of Technology (KIT), "Regeln zur Sicherung guter wissenschaftlicher Praxis im Karlsruher Institut für Technologie (KIT)," http://www.kit.edu/downloads/K_OBP_XX_RI_01_05-10.pdf 04.12.2012.

[2] Deutsche Forschungsgemeinschaft, "Recommendations of the Commission on Professional Self Regulation in Science -Proposals for Safeguarding Good Scientific Practice," January, 1998, http://www.dfg.de/en/research_funding/legal_conditions/good_scientific_practice/ 04.12.2012.

[3] Max-Planck-Gesellschaft (MPG), "Rules of Good Scientific Practice," 2000, http://imprs.ice.mpg.de/ext/fileadmin/imprs/pictures-files/Good_Scientific_Practice_MPG.pdf 04.12.2012.

[4] OECD, "OECD Principles on Good Laboratory Practice," Paris, 1998, http://search.oecd.org/officialdocuments/displaydocumentpdf/?doclanguage=en&cote=env/mc/chem%2898%2917 04.12.2012.

[5] A. Bryaman, "Social Research Methods," Oxford, 2012.

[6] U. Degenhardt, "Requirements for Repository Systems," eScience Seminar, 2009, http://colab.mpdl.mpg.de/mediawiki/images/e/ea/ESci_09_Sem_2_Requirements_from_MPI_f_Dynamics_and_Self-Organization_Degenhardt.pdf 04.12.2012.

[7] L.M. Chan and M.L. Zeng, "Metadata Interoperability and Standardization - A Study of Methodology, Part I," D-Lib Magazine, Vol. 12, Nº. 6, 2006.

[8] Bundesamt für Sicherheit in der Informationstechnik (BSI), "BSI Technical Guideline 03125: Preservation of Evidence of Cryptograohically Signed Documents v.1.1," 2011, https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG03125/TG-03125_main.pdf 04.12.2012.

[9] T. Steinke, "The Universal Object Format – An Archiving and Exchange Format for Digital Objects," in Research and Advanced Technology for Digital Libraries, Springer Berlin, 2006, pp. 552–554.

[10] M. Feijen, "What researchers want," SURFfoundation, 2011, http://www.surf.nl/nl/publicaties/documents/what_researchers_want.pdf 04.12.2012.

[11] M. Rubacha, A.K. Rattan, and S.C. Hosselet, "A Review of Electronic Laboratory Notebooks available in the market today," JALA, vol. 16, Feb.2011, pp. 90–98 , doi:10.1016/j.jala.2009.01.002.

[12] M. Dreyer, N. Bulatovic, U. Tschida, and M. Razum, "eSciDoc – a Scholarly Information and Communication Platform for the Max Planck Society," Proc. German e-Science Conference, 2007.

[13] T. Schlauch and A. Schreiber, "DataFinder – A Scientific Data Management Solution," Proc. PV 2007, 2007.

[14] R. Altenhöner, "Data for the future: The German project 'Co-operative development of a long-term digital information archive' (kopal)," Library Hi Tech, Vol. 24 Iss: 4, 2006, pp. 574–582, doi:10.1108/07378830610715437.

[15] B.A. Weber, H. Yarandi, M.A. Rowe, and J.P. Weber, "A Comparison Study: Paper-based Versus Web-based Data Collection and Management," in Applied Nursing Research, Vol. 18 Iss: 3, 2005, pp. 182–185.

[16] S. Hackel, P.C. Johannes, M. Madiesh, J. Potthoff, and S. Rieger, "Scientific Data Lifecycle – Beweiswerterhaltung und Technologien," Proc. 12. Deutscher IT-Sicherheitskongress (BSI-IT-SEC 2011), SecuMedia, 2011, pp. 403–418.

[17] M.H. Elliott, "Electronic Laboratory Notebooks: A Foundation for Scientific Knowledge Management Edition III," Atrium Research & Consulting LLC, Wilton, CT USA.

[18] P. Boogaard and P. Pijanowski, "Electronic Laboratory Notebooks (ELN) Mean Many Things to Many People," Feb. 09, 2012, http://www.laboratory-journal.com/science/information-technology-it/electronic-laboratory-notebooks-eln-mean-many-things-many-people/ 04.12.2012.

[19] D. Morris, "LIMS vs ELN arch enemies or best of friends?," DDW, 2009, http://www.ddw-online.com/informatics/p146753-lims-vs-elns-arch-enemies-or-best-of-friends-summer-09.html 04.12.2012.

[20] T. Kuipers and J. v. d. Hoeven, "Insight into digital preservation of research output in Europe - Survey report," 2009, http://www.parse-insight.eu/downloads/PARSE-Insight_D3-4_SurveyReport_final_hq.pdf 04.12.2012.

[21] F. Rudolphi and L.J. Goossen, "Electronic Laboratory Notebook: The Academic Point of View," J. Chem. Inf. Model., 2012, 52 (2), pp 293–301 DOI: 10.1021/ci2003895.

[22] S. Fischer-Diskau, "Das elektronisch signierte Dokument als Mittel zur Beweissicherung," Nomos, 2006.

[23] J. Potthoff, S. Rieger, and P.C. Johannes, "Enhancing the Provability in Digital Archives by Using a Verifiable Metadata Analysis Web Service," Proc. 7th ICIW 2012, 2012, pp. 112–117.

[24] Digital Library Federation, "<METS> Metadata Encoding and Transmission Standard: Primer and Reference Manual," Version 1.6 Revised, 2010, http://www.loc.gov/ standards/mets/ 04.12.2012.

[25] M. Gudgin, M. Hadley, N. Mendelsohn, J.J. Moreau, H. Frystyk Nielsen, A. Karmarkar, and Y. Lafon, "SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)," 2007, http://www.w3.org/TR/soap12-part1/ 04.12.2012.

[26] A. Rodriguez, "RESTful Web services: The basics," IBM, developerWorks, 2008, http://www.ibm.com/developerworks/webservices/library/ws-restful/ 04.12.2012.

[27] M. Ney, G. Kloss, and A. Schreiber, "Using Provenance to support Good Laboratory Practice in Grid Environments," in Data Provenance and Data Management in eScience, Springer, 2011.

[28] A. Szalay, "Extreme Data-Intensive Scientific Computing," in Computing in Science Engineering, vol.13 no.6, 2011, pp.34-41.

[29] J. Silbermann, S. Weinert, C. Wernicke, and M. Frohme, "Quality and information management in the laboratory," Logistics and Industrial Informatics (LINDI), 2011, pp. 93-98.

[30] K.T. Taylor, "Evolution of electronic laboratory notebooks," in Collaborative Computational Technologies for Biomedical Research, Wiley, 2011, pp. 303-320.

Parallel Processing of Multiple Graph Queries Using MapReduce

Song-Hyon Kim, Kyong-Ha Lee[†], Hyebong Choi, and Yoon-Joon Lee
Department of Computer Science
KAIST, Daejeon, Republic of Korea
Email: {songhyon.kim, egnever, yoonjoon.lee}@kaist.ac.kr, bart7449@gmail.com[†]

***Abstract*—Recently the volume of the graph data set is often too large to be processed with a single machine in a timely manner. A multi-user environment deteriorates this situation with many graph queries given by multiple users. In this paper, we address the problem of processing multiple graph queries over a large set of graphs. We devise several methods that support efficient processing of multiple graph queries based on MapReduce. Particularly, we focus on processing multiple queries for graph data in parallel with a single input scan. We show that our methods improve the performance of multiple graph query processing with various experiments.**

***Keywords*-parallel processing; MapReduce; graph query; big data;**

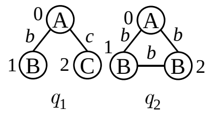
I. Introduction

Graphs are widely used to model complex structures such as chemical compounds, protein interactions, and Web data in many applications [1]. However, many graph data sets are often hard to handle within a single machine because of their size and complexity, e.g., the PubChem project now serves more than 30 million chemical compounds, the storage size of which hits tens of terabytes [2].

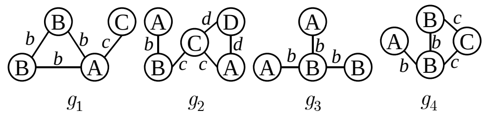
It is required for users to find graphs that contain the patterns in which they are interested from a graph data set. This is formally called a *graph query* or *subgraph isomorphism* problem, which belongs to NP-complete [3]. In a multi-user environment, many users may describe their interesting patterns with their own graph-structured queries. With the massive volume of graph data set and many query graphs, it is more difficult to process graph queries within a single machine in a timely manner.

Meanwhile, MapReduce has gained a lot of attention from both of industry and academia [4]. MapReduce presents a distributed method to processing data-intensive jobs with no hassle of managing the jobs across nodes. In addition, MapReduce has advantages in its scalability, fault-tolerance, and simplicity [5].

In this paper, we address the problem of processing multiple graph queries over a large set of graphs using Hadoop [6], an open source implementation of MapReduce. We first start by discussing a naïve approach which performs all pair-wise subgraph isomorphism tests between a graph query set and a graph data set. Definitely, the naïve approach is very time-consuming. To reduce the number of expensive subgraph isomorphism tests, we introduce a filter-and-verify



(a) Graph query set Q



(b) Graph data set D

Figure 1. A running example

scheme and propose two implementations of the scheme, i.e., *map-side verification* and *reduce-side verification*. In addition, we devise several alternatives considering feature types and feature set comparison, which affect the overall query processing time. We show our experimental results with both synthetic and real data sets. To the best of our knowledge, this is the first work to consider parallel processing of multiple graph queries over a large graph data set with MapReduce.

The rest of the paper is organized as follows. We discuss related work in Section II. Section III presents preliminaries. We propose our methods in Section IV and optimizations for them in Section V. We provide our experimental results in Section VI. Finally, we conclude this paper in Section VII.

II. Related Work

Graph databases are categorized into two types: a *graph-transaction setting* and a *single-graph setting* [7]. In the graph-transaction setting, a graph database consists of a set of relatively small graphs, whereas in the single-graph setting, the data of interest is a single large graph. In this paper, we focus on the graph-transaction setting consisting of tens of millions of graphs.

MapReduce programming model relies on both data parallelism and data shipping paradigms [5]. The MapReduce framework works in two stages: *map* and *reduce*. Input data are partitioned into equal-sized blocks and each of blocks is assigned to a mapper at map stage. Outputs of map stages are stored in local disks, then shuffled and pulled to reducers at reduce stage. This implies I/O inefficiency during processing. Readers are referred to a recent survey for the MapReduce framework and its up-to-date improvements [5].

Blanas et al. [8] compared many join techniques available on MapReduce for analysis of click stream logs at Facebook. Vernica et al. [9] proposed a method to parallelize set-similarity joins with MapReduce by utilizing prefix-filtering scheme. Our work is motivated by some techniques for supporting join processing in the MapReduce programming model.

Some scientists also studied graph processing with MapReduce. Luo et al. [10] solved a single graph query processing problem under a very restricted assumption, i.e., each edge in a query graph must be uniquely identified by labels of its endpoints and itself. Lin et al. [11] proposed several design patterns applicable to iterative graph algorithms such as PageRank [12]. There are also other studies about large scale graph processing [13], [14]. However, they are different from our work in that they focused on processing a single large graph such as Web data or social network, rather than a large set of small graphs. They also do not address the subgraph isomophism problem at all.

III. Preliminaries

We formally define a graph query problem that we deal with in this paper. Then, we describe the way of representing graph data in MapReduce. In addition, we introduce the concept of features in graph query processing.

A. Problem Definition

A graph is denoted by a tuple $g = (V, E, L, l)$, where V is the set of vertices and E is the set of undirected edges such that $E \subseteq V \times V$. L is the set of labels of vertices or edges, and the labeling function l defines the mapping: $V \cup E \rightarrow L$. We also denote the vertex set and the edge set of graph g by $V(g)$ and $E(g)$, respectively. Moreover, we denote the label of $u \in V(g)$ and $(u, v) \in E(g)$ by $l(u)$ and $l(u, v)$, respectively.

Definition 1: Given two graphs $g = (V, E, L, l)$ and $g' = (V', E', L', l')$, g is **subgraph isomorphic** to g', denoted $g \subseteq^s g'$ if and only if there exists an injective function $f : V \rightarrow V'$ such that

1) $\forall u \in V,\ f(u) \in V'$ and $l(u) = l'(f(u))$,
2) $\forall (u, v) \in E,\ (f(u), f(v)) \in E'$ and $l(u, v) = l'(f(u), f(v))$.

Problem Statement: Let $D = \{g_1, g_2, \cdots, g_n\}$ be a graph data set. Furthermore, let $Q = \{q_1, q_2, \cdots, q_m\}$ be a graph query set such that $|Q| << |D|$. For each graph query $q \in Q$, we find all the graphs to which q is subgraph isomorphic from D.

Figure 1 shows a running example which will be used throughout this paper. In the example, the answers of two graph queries are $A(q_1) = \{g_1\}$ and $A(q_2) = \{g_1\}$, where $A(q_i)$ represents a set of answers for q_i such that $A(q_i) = \{g_j | q_i \subseteq^s g_j \wedge g_j \in D\}$.

B. Graph Representation

Data in MapReduce are modeled by a list of `<key, value>` pairs, which are generally typed strings. Thus, all graph data must be serialized to be the `<key, value>` pairs for processing. We use the the following terms to refer to our serialized graph data.

- `gid`: a unique identifier for a single graph g.
- `gcode`: a serialized format of g, which enumerates vertices and edges in g, i.e., $\{|V(g)|$, $|E(g)|$, $l(V(g))$, $E(g))\}$, where $e \in E(g)$ is represented as 'from-`gid`, to-`gid`, $l(e)$'.

We call a pair of `gid` and `gcode` a *graph record*. For example, query graph q_2 in Figure 1(a) is modeled by the pair of `gid` and `gcode` ('q_2', '$3, 3, A, B, B, 0, 1, b, 0, 2, b, 1, 2, b$').

C. Features in graph query processing

A feature is a substructure of a graph, which represents partial structural information of the graph. There are various kinds of features such as path, subtree, and subgraph [15]–[17].

IV. The Proposed Method

In this section, we first discuss a naïve approach for processing multiple graph queries using MapReduce. Two feature set comparison methods are introduced in the following subsection. Finally, we propose two different implementations based on MapReduce, both of which follow a *filter-and-verify* scheme.

A. Naïve approach

A simple solution for parallel processing of multiple graph queries over a massive graph data set is to perform subgraph isomorphism test in parallel for each pair of query graph q and data graph g such that $q \in Q$ and $g \in D$. Since $|Q| << |D|$, we partition D into equal-sized blocks and simultaneously perform the test for each block with query set Q. This is akin to partitioned nested-loop join techniques in parallel DBMS [18]. However, the naïve approach requires to perform subgraph isomorphism test $|Q| \times |D|$ times in total. This consumes a lot of time for query processing.

Therefore, we rather use a filter-and-verify scheme to reduce the number of subgraph isomorphism tests. In the scheme, we first get candidate data graphs then test subgraph isomorphism with only the candidate graphs, rather than directly testing subgraph isomorphism for all data graphs. For this, we filter irrelevant graphs out by comparing a set of features from a query graph with a set of features from a data graph. Although the filtering phase may produce false positives, it reduces the overall execution time significantly since it excludes many graphs in advance so that the number of graphs to be verified is quite reduced.

B. Feature set comparison

In our approach, filtering is a set-inclusion test between two feature sets, one of which comes from a query graph q and the other comes from a data graph g. To become a candidate, a feature set of data graph g must include all items in a feature set of query graph q. The problem here is how quickly to perform the set-inclusion test between two feature sets. A basic approach is akin to nested-loop join i.e., each feature of a query graph is iteratively compared with the features of a data graph one by one until all features of a query graph are tested with features of all data graphs.

To address this problem, we devise a filtering method based on Bloom filter [19], a space-efficient probabilistic structure that is useful for existence test. The detailed procedure is as follows. We first assign each item in a feature set for data graph g a unique number. Then, we convert the feature set to a bitarray by applying multiple hash functions to the unique number associated with each feature. During query processing, if bit positions in the bitarray are set to 1 for all items in the feature set of query graph q, the corresponding data graph becomes a candidate of q. Of course, Bloom filter may generate false positives. We test how such false positives affect the overall performance of our system in experiments. Note that we build Bloom filters only for data graphs, but not for query graphs as the average number of features extracted from query graphs is much less than that from data graphs.

C. Two MapReduce implementations

With the filter-and-verify scheme, we examine if we benefit from where to put the verification step into a MapReduce job: *map-side* and *reduce-side*. In the map-side verification method, both filtering and verification are performed at map stage. The results of map stage are directly emitted into HDFS [6], a distributed filesystem used in Hadoop, and reduce stage works nothing at all. Workload across mappers are expected to be balanced by runtime scheduling scheme in the MapReduce framework [4]. On the contrary, in the reduce-side verification method, mappers perform only the filtering step and reducers perform subgraph isomorphism tests. The map-side verification method is similar to a map-merge join technique in MapReduce except that it requires to perform verification step for each pair of graphs, instead of join operation with two relations [8].

1) Map-side verification: Our map-side verification method is described in Algorithm 1. Reduce stage is omitted, since it contains no action. In MapReduce, input data are first partitioned into equal-sized blocks and each of the blocks is assigned to a mapper at map stage. Each mapper reads a

```
1  class Mapper
2    method initialize()
3      $Q \leftarrow$ load a list of $[gid,\ gcode]$ from HDFS
4      $Q.gfeature[\,] \leftarrow$ generate features from $Q$
5    method map($K$ : null, $V$ : $[gid, gcode]$)
6      $feature \leftarrow$ generate features from $V$
7      foreach query graph $q$ in $Q$ do
8        if $q.gfeature \subseteq feature$ then
9          if SubIsoTest($q.gcode$, $V.gcode$) then
10           emit($q.gid$, $V.gid$)
```

Algorithm 1: The map-side verification method

```
1  class Mapper
2    method initialize()
3      $Q \leftarrow$ load a list of $[gid,\ gcode]$ from HDFS
4      $Q.gfeature[\,] \leftarrow$ generate features from $Q$
5      initialize a buffer $B$
6    method map($K$ : null, $V$ : $[gid, gcode]$)
7      $feature \leftarrow$ generate features from $V$
8      foreach query graph $q$ in $Q$ do
9        if $q.gfeature \subseteq feature$ then
10         $B[V] \leftarrow B[V] \cup \{q.gid\}$
11   method close()
12     foreach data graph $V$ in $B$ do
13       emit($V$, $B[V]$)
14 class Reducer
15   method initialize()
16     $Q \leftarrow$ load a list of $[gid,\ gcode]$ from HDFS
17   method reduce($K'$ : $[gid, gcode]$,
     $V'$ : a list of $gid$s)
18     foreach query id $qid$ in $V'$ do
19       $code \leftarrow$ retrieve gcode with $qid$ from $Q$
20       if SubIsoTest($code$, $K'.gcode$) then
21         emit($qid$, $K'.gid$)
```

Algorithm 2: The reduce-side verification method

block of graph data set D. When launching a MapReduce job, the MapReduce framework delivers a set of graph queries to each mapper via `distributedCache` [6], a facility to cache read-only files in Hadoop. Thus, we load a list of query graphs from a local disk, although Algorithm 1 generally describes loading of query graphs from HDFS (line 3). Each mapper generates features from graph queries in Q (line 4). For each graph record in the block of D, the `map` method also generates features (line 6). Then, the `map` method compares the feature set of each query graph q with the feature set of data graph V (line 8). If contained, the `map` method verifies the candidate by testing subgraph isomorphism (line 9). Note that `SubIsoTest`(q, g) requires `gcode` of two graphs to check whether graph q is subgraph isomorphic to graph g. A feature comparison operator $\subseteq$ is set-inclusion relation in the algorithm, however its details depend on which feature type is used.

2) Reduce-side verification: Algorithm 2 describes the reduce-side verification method. This scheme is more faithful to the MapReduce programming model. Unlikely to the Mapper in Algorithm 1, the verification step moves to reduce stage. Moreover, we perform pre-aggregation of intermediate data in mapper, i.e., *in-mapper combining* [11], to reduce the size of intermediate data delivered to reducers.

The `initialize` method is similar to that of Algorithm 1 except for buffer B, which holds a pair of a data graph V and the corresponding `gids` of its candidate graph queries (line 5). In the `map` method, candidates are identified by comparing features and those are pushed into buffer B for delivery (lines 9–10). When closing the mapper, all pairs of data graph V and the corresponding `gids` of graph queries are emitted (lines 12–13). The `reduce` method reads data graph K' and a list of `gids` of candidate graph queries V' as input (line 17). Here, the input means that the data graph V is a candidate of every graph query in V'. For each graph query, the `reduce` method tests subgraph isomorphism to data graph K' (line 20). Final results are emitted with a pair of `gids` of query graph and data graph (line 21). Figure 2 shows an illustration of the reduce-side verification method. In the figure, *gfeature* represents a set of features. Details about the features will be explained in the following section.

V. Optimization

We discuss two optimization techniques for the proposed methods in this section.

A. Reducing I/Os

As noted in many studies, MapReduce is not optimized for I/O efficiency. Thus I/O cost is a dominant factor for performance in MapReduce [5]. For that reason, we delicately model our data format for I/O efficiency. In Algorithm 2, we make `map` stage emit its result as a pair of a data graph and a list of `gids` of candidate query graphs, instead of a pair of a query graph and its candidate data graph. Two reasons lie in the decision. First, it saves many I/Os which are required to deliver the overall structure of query graphs for each candidate data graph separately. The `reduce` stage eliminates redundant loading of query graphs across different data graphs, then reads once the `gcode` of each query graph from HDFS. Furthermore, since $|Q| << |D|$, the lookup cost of loading graph structure with a given `gid` is reduced when choosing Q rather than D. Second, this strategy generates more groups of intermediate results. It is more suitable for load balancing. If groups of intermediate results are small, sometimes data skewness may involve performance degradation [20].

B. Feature type

The choice of feature types affects the number of candidates. A conventional feature type of a graph g is *edge label* (EL), i.e., $(l(u), l(v), l(u, v))$ for edge $(u, v) \in E(g)$. Edge label must be unique in a feature set, meaning that identical edges are not shown in a feature set. Luo et al. [10] adapts this feature type. Based on edge label, we propose two optional feature types. First, we add the number of occurrences of each edge label in a graph, which is denoted by *edge label with count* (ELC). This is simple but largely drops false positives. The other is to use cycle information in a graph, since cycles are rare in a sparse graph [17]. We call this *edge label with count and cycle* (ELC+CL).

Table I
SUMMARY OF PROPOSED METHODS

Verification position	Feature type	Feature set comparison
- map-side - reduce-side	- edge label - edge label with count info. - edge label with count and cycle info.	- nested-loop - Bloom filter

Example 1: Query graph q_2 in Figure 1(a) has features of *edge label* $\{(A, B, b), (B, B, b)\}$, *edge label with count* $\{(A, B, b, 1), (A, B, b, 2), (B, B, b, 1)\}$ where the last number of each feature is the occurrences of that edge label, and *edge label with count and cycle* $\{(A, B, b, 1), (A, B, b, 2), (B, B, b, 1), cycle(0, 1, 2)\}$ where $cycle(...)$ denotes a cycle consisting of those vertices.

Table I summarizes our methods discussed so far.

VI. Experiments

A. Experimental Setup

Experiments were performed on a 9-node cluster, one of which was designated as a name node. Each data node is equipped with an Intel i5-2500 3.3GHz processor, 8GB memory and a 7200RPM HDD, running on CentOS 6.2. All nodes are connected via a Gigabit switching hub. We select the datasets given by authors of `iGraph` [15], a graph processing benchmark tool. Algorithms were implemented in C++ and executed via Hadoop Pipes [6], a C++ library that provides communications with Hadoop. We use VF2 [21] to test subgraph isomorphism between two graphs. For real graph data sets, we generated various sized chemical data also used in [15], each of which has 23.98 vertices and 25.76 edges in average. For synthetic data sets, we chose the dataset named *Synthetic.10K.E30.D3.L50* from [15], each of which has 14.23 vertices and 30.53 edges with 50 distinct vertex/edge labels in average. The number of graphs in a graph data set, $|D|$, is 10 million. We also randomly generated several sets of graph queries, which contain a various number of queries, i.e., $|Q| = 1, 10, 100$, and 1000.

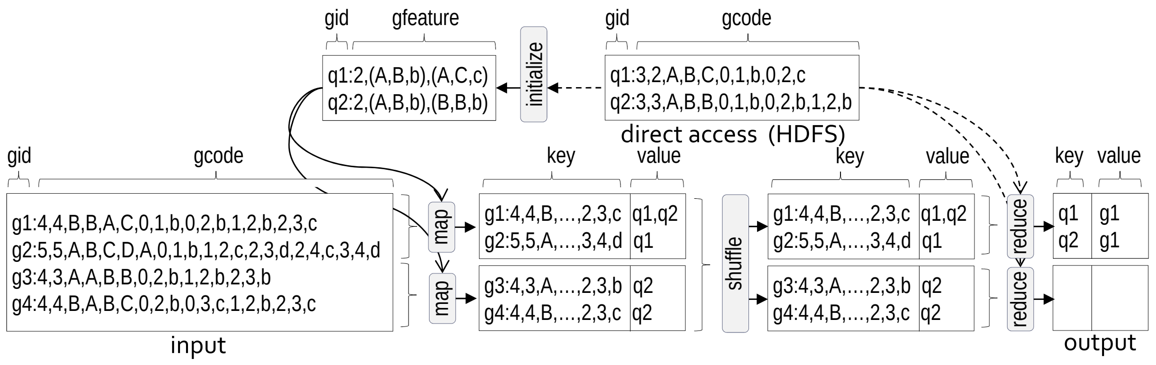


Figure 2. An illustration of parallel processing of graph queries shown in Algorithm 2 with MapReduce

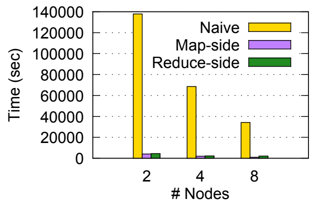


Figure 3. Methods (synthetic)

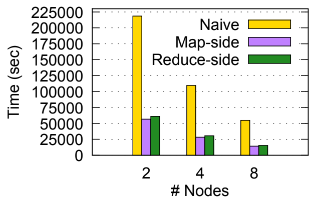


Figure 4. Methods (real)

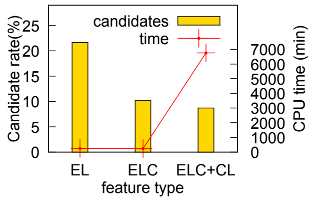


Figure 7. Candidate ratio with different feature types

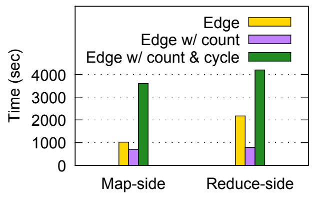


Figure 5. Feature type (synthetic)

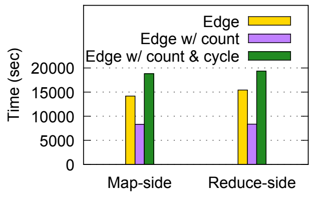
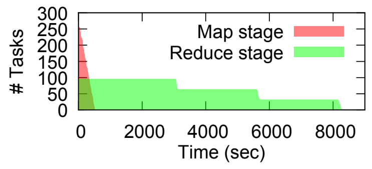
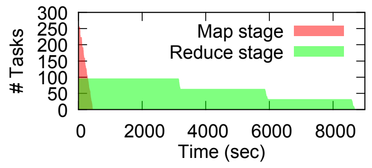


Figure 6. Feature type (real)



(a) Partitioned nested-loop filtering



(b) Bloom filter-based filtering

Figure 8. MapReduce job execution with real dataset

B. Performance Analysis

We first compared three methods, i.e., naïve, map-side verification, and reduce-side verification, of processing graph queries with both real and synthetic datasets. Our results are shown in Figure 3 and 4. The map-side verification method showed the best performance in both cases. In addition, overall execution time decreases linearly with the increasing number of nodes in a cluster. However, The map-side verification could not outperform the reduce-side verification method. The reason is that the size of our intermediate results is marginal, since we gave our best effort to reduce I/Os while delivering intermediate results to reducers. We also compared three feature types as shown in Figure 5 and 6. The feature type of *edge label with count* showed the best performance. Although the most complex feature type, i.e., *edge label with count & cycle* (ELC+CL), has the least number of average candidates, it was not the best since it spent much time in extracting features and testing set-inclusion as shown in Figure 7 (in the figure, CPU time in the right axis means summation of all the elapsed time of mappers in Algorithm 2). Next, we tested partitioned nested-loop and Bloom filter-based filtering schemes. We built a 160-bit length Bloom filter for each $g \in D$ with 4 hash functions. Since $|E(g)|$, $g \in D$ is 25.76 in average, the probability of false positives that the Bloom filter has is 5.2%. Figure 8 explains time taken by map and reduce stages in two filtering schemes running on Hadoop. In nested-loop filtering scheme, map stage took 529 seconds and 8,288 seconds in reduce stage. Bloom filter-based filtering improves the filtering time with 462 seconds. However, overall time was rather extended to 8,723 seconds. The reason is that as the Bloom filter-based filtering scheme generates more candidates, thus it requires more subgraph isomorphism test that is a dominant factor in execution time. Finally, we tested the scalability of our methods on MapReduce, varying the number of data graphs

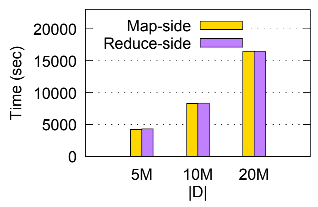


Figure 9. Varying # of data graphs (real)

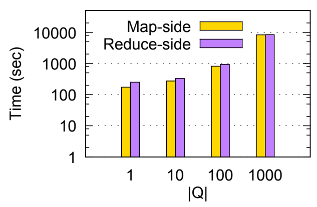


Figure 10. Varying # of queries (real)

and query graphs. As shown in Figure 9 and 10, filter-and-verify scheme on MapReduce scales linearly with both the number of data graphs and query graphs.

VII. Conclusion

In this paper, we applied the MapReduce framework to process multiple graph queries over a large graph data set. Lessons learned in this paper are as follows. Filter-and-verify scheme is better than the naïve approach as expected. Complex features help our system improve its execution time, but there is a tradeoff that feature extraction and comparison overhead may harm the overall execution time. Lastly, reduction of the number of candidates is more crucial for the overall execution time than reduction of the filtering time.

Acknowledgment

This work was partly supported by NRF grant funded by Korea Government (No. 2011-0016282).

References

[1] C. Aggarwal and H. Wang, “Managing and mining graph data,” *Advances in Database Systems*, vol. 40, 2010.

[2] “The pubchem project,” http://pubchem.ncbi.nlm.nih.gov, (accessed November 20, 2012).

[3] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1990.

[4] J. Dean and S. Ghemawat, “Mapreduce: simplified data processing on large clusters,” *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[5] K. Lee, Y. Lee, H. Choi, Y. Chung, and B. Moon, “Parallel data processing with mapreduce: a survey,” *ACM SIGMOD Record*, vol. 40, no. 4, pp. 11–20, 2012.

[6] “Apache hadoop,” http://hadoop.apache.org, (accessed November 20, 2012).

[7] M. Kuramochi and G. Karypis, “Finding frequent patterns in a large sparse graph*,” *Data mining and knowledge discovery*, vol. 11, no. 3, pp. 243–271, 2005.

[8] S. Blanas, J. Patel, V. Ercegovac, J. Rao, E. Shekita, and Y. Tian, “A comparison of join algorithms for log processing in mapreduce,” in *Proceedings of the 2010 international conference on Management of data*. ACM, 2010, pp. 975–986.

[9] R. Vernica, M. Carey, and C. Li, “Efficient parallel set-similarity joins using mapreduce,” in *Proceedings of the 2010 international conference on Management of data*. ACM, 2010, pp. 495–506.

[10] Y. Luo, J. Guan, and S. Zhou, “Towards efficient subgraph search in cloud computing environments,” *Database Systems for Adanced Applications*, pp. 2–13, 2011.

[11] J. Lin and M. Schatz, “Design patterns for efficient graph algorithms in mapreduce,” in *Proceedings of the Eighth Workshop on Mining and Learning with Graphs*. ACM, 2010, pp. 78–85.

[12] L. Page, S. Brin, R. Motwani, and T. Winograd, “The pagerank citation ranking: Bringing order to the web,” in *Stanford InfoLab*, 1999.

[13] G. Malewicz, M. Austern, A. Bik, J. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, “Pregel: a system for large-scale graph processing,” in *Proceedings of the 2010 international conference on Management of data*. ACM, 2010, pp. 135–146.

[14] U. Kang, C. Tsourakakis, and C. Faloutsos, “Pegasus: mining peta-scale graphs,” *Knowledge and information systems*, vol. 27, no. 2, pp. 303–325, 2011.

[15] W. Han, J. Lee, M. Pham, and J. Yu, “igraph: a framework for comparisons of disk-based graph indexing techniques,” *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 449–459, 2010.

[16] X. Yan, P. Yu, and J. Han, “Graph indexing: A frequent structure-based approach,” in *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*. ACM, 2004, pp. 335–346.

[17] P. Zhao, J. Yu, and P. Yu, “Graph indexing: tree+ delta >= graph,” in *Proceedings of the 33rd international conference on Very large data bases*. VLDB Endowment, 2007, pp. 938–949.

[18] H. Lu, K. Tan, and B. Ooi, *Query processing in parallel relational database systems*. IEEE Computer Society Press, 1994.

[19] B. Bloom, “Space/time trade-offs in hash coding with allowable errors,” *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[20] Y. Kwon, M. Balazinska, B. Howe, and J. Rolia, “Skewtune: Mitigating skew in mapreduce applications,” in *Proceedings of the 2012 international conference on Management of Data*. ACM, 2012, pp. 25–36.

[21] L. Cordella, P. Foggia, C. Sansone, and M. Vento, “A (sub) graph isomorphism algorithm for matching large graphs,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 10, pp. 1367–1372, 2004.

On using object-relational technology for querying LOD repositories

Iztok Savnik
Department of Computer Science
Faculty of Mathematics, Natural sciences and Information Technology
University of Primorska, Koper, Slovenia
iztok.savnik@upr.si

***Abstract*—Query engines for managing RDF repositories based on relational technology represent an alternative to query engines based on triple-stores. The paper presents adaptation of object-relational technology for managing RDF data. The architecture of query engine for RDF databases is proposed: i) mapping of RDF graph model to object-relational representation is described, and ii) internal structures and methods used for the implementation of query engine are discussed. While following the architecture of object-relational systems we adapt it for the specific operations defined on RDF data.**

***Keywords*-databases; RDF databases; query processing system; database system implementation.**

I. Introduction

Linked Open Data (abbr. LOD) repositories storing RDF files represent primary means to publish data on the Internet. They serve recently to disseminate large amounts of data in the areas of Life-science, Biology, Chemistry, Medicine, Geography and Media as well as about the institutions such as Governments. It is estimated that the data sets on the above stated areas include more than 31 Giga triples [15].

The first and most common approach to storage and manipulation of RDF data is the use of triple-storage system which is based on few relational tables and a collection of indexes built on these tables. Examples of such triple-store based systems are 3store [13] and Bigdata [3]. SPARQL queries are converted into some form of relational calculus expressions which are further converted to access paths implemented by indexes.

The second approach that is also common due to availability of relational and object-relational systems is the use of ordinary RDBMS by converting triples into relational or object-relational tables with three or four columns. SPARQL queries are converted into relational queries. Indexes can be created as needed and RDBMS uses its optimizer to create fast plans for SPARQL queries. Example of DBMS using such approach is Virtuoso [29].

Both of the above stated approaches to querying RDF repositories treats complete repository as one single database (or table of triples) on top of which indexes are created, either that we use standard object-relational indexes or special indexes that depend on a concrete system.

In this paper we present architecture of a database system for storing RDF triples as classes, properties and concrete objects. RDF triples are converted to specific kind of entities when they are loaded into the database. We follow therefore *conceptual* representation of RDF triples that uses RDFS types or least general types (classes) covering all properties of individual concrete triples, in the case RDFS types are not used. Class extensions now represent collections of concrete objects.

The problems that appear with querying collections of highly structured entities are in a way common to object-oriented database systems [2], object-relational systems [19], [28], XML databases [5] and RDF databases [20]. Among the most important problems that are characteristic for querying complex entities organized in collections are: accessing components of complex entities using path expressions [8], [17], restructuring and creating complex entities [28], [21], and querying complex conceptual schemata [23].

A practical solution to the problems of querying collections of complex objects is presented. Query processing is rooted in relational and object-relational technology. The existing techniques have been adapted and simplified to obtain database system tuned to process collections of complex objects structured by means of RDF data modeling constructs.

In the following section we present design decisions for architecture of query processing system for querying RDF databases. The architecture of query processing system is based on previous work on querying system for internet data sources [25], [23] and on formal definition of the structural part of object-oriented model [24].

Firstly, the storage manager and underlying data model is presented. Further, mapping between RDF and internal model of a database system is described. We overview the architecture of query processor including parsing, type checking, query optimization and query evaluation. Finally, bulk loader for RDF is described. Section III gives a short overview of related work and Section IV presents conclusions and further work.

II. Architecture of query processor

Our work focuses on the design of robust and flexible query execution system for querying and integration of RDF data. The design of query execution system is rooted in the existing work on relational and object-relational query execution systems [11], [10], [9], [14], [6].

A. Storage manager

Every object has an identifier denoted as *object identifier* or *oid*. Objects can have functional or multi-valued attributes. The term "attribute" corresponds to what is also called object component or data member. The range of the functional attribute can be an atomic value: `number`, `string` or `object` (oid). Multi-valued attribute is an array of atomic values. The data model of is formally presented in [24].

From a programmer's perspective, an object is a reference to the associative (hash) array and an attribute is a hash entry which can be either a typed scalar or a reference to an array of typed scalars. Storage manager differentiates between two kinds of objects: class objects and individual objects (instances). Each class object is associated to the set of its sub-classes and its instances. The storage manager implementation is based on the hierarchical and network data models: class, its instances and sub-classes are linked in a ring realized by a double linked list.

1) Types: We distinguish between two sorts of types: *atomic types* and *tuple-structured types*. Atomic types are: `number`, `string` and `object`. Type `object` is oid of the root class object. Its instances are all oids. Tuple-structured types have attributes that are either functional or multi-valued (implemented as refs to arrays). The range of functional and multi-valued attributes can only be atomic types.

As suggested by the above description, types together with the corresponding identifiers form *class objects*. The structure of database can then be viewed uniformly as partially ordered set of objects: class objects and instances (individual objects) are interrelated by the inheritance (isa) and instantiation relationship.

2) Record manager: The record storage manager is based on the Berkeley DB storage system providing access to different storage structures such as for instance hash-based index or B+ tree. Record manager implements a data store for records representing individual and class objects. Records are treated as arrays of bytes, the structure of which is known at the object level. Each record has a record identifier (abbr. rid) implemented as system generated identifier which is used as the key for the hash-based index in Berkeley DB. Therefore, records are stored as oid/value pairs where the values are packed in the sequences of bytes. Record manager includes the methods for the work with object identifiers, routines for reading and writing records, methods for the realization of the hierarchical database model, and access to main memory indexes that are associated to the class object.

3) Object manager: The object manager serves as a cache of objects loaded from Web as well as for storing intermediate results during query processing. Persistent objects are objects that are tied to the database via object manager. Each object has an identifier which is implemented by means of record identifier from the subordinate level. External identifiers which are unique within the datafile can be assigned to objects when they are created. Object cache is realized using LRU (least recently used) strategy for the selection of objects to be removed from cache. The size of object cache can be set as the system parameter via the configuration record of a datafile as well as at run-time.

Object manager handles relationships between instances and class records (objects) in a similar manner as in early network and hierarchical database storage systems. Each class is realized as a list including all instances as well as sub-classes. Ground objects are added at the beginning of the list and the class objects are added at the end of the list.

The implementation of objects is based on associative arrays (or mappings) which map object identifier to the value of the specified attribute. The module includes simple and uniform routines for the manipulation of the instance objects and class objects. The routines can create and delete objects, set and retrieve (get) the values of object attributes, relate objects to inheritance hierarchy and update operations. Two different get and set methods are implemented for reading and updating single and multivalued attributes.

Objects can be accessed either directly using object identifiers, or through scan operator (iterator). When accessed using object identifiers, objects are manipulated through the main memory pointers to objects in cache. User is responsible that object is accessed each time before it is used. Routines for update operation mark the object "changed" when components are altered, unchanged objects do not need to be treated thereafter. When objects are accessed using scans they are obtained by means of iterator. The following scan operators are provided: sequential scan over class extension and sequential scan over all class instances of a class, and hash or B-Tree index scan on class extension as well as the scan over all class instances.

B. Parser

This module includes the implementation of a parser for the algebra expressions. The algebra expressions are checked for syntactical errors and then translated into query trees.

Lexical analysis is implemented using a simple lexer which converts a query into a sequence of tokens including operation codes and constants. Parsing and translation are realized by a top-down examination of the query expressions based on methods for LL(1) grammars.

The query expressions are represented during the query compilation and execution as query trees. A *query tree* is a graph (dag) which vertices are query nodes. The vertices are linked with respect to the parse tree. Two types of nodes are used for the representation of queries: the *query nodes* are tree nodes representing set operations (e.g. operation join), and the *parameter nodes* are tree nodes for the representation of parameter expressions (eg. join parameter expression). The vertices represent all model operations: arithmetic operations, logical operations, comparison operations, schema manipulation operations as well as variables.

The basic skeleton of the query tree is constructed during the parsing process. The variables and the names of the data sources and spans are stored in symbol table `symtab`. The query nodes represent the operations, spans linking rules, and access methods. The data for the different phases of query processing is stored in the same tree nodes. Each particular module (e.g. query optimization) manipulates its own view of query nodes.

1) Representing predicates: The parameters of the set operations select, project and join are abstracted using *parameter nodes*. Parameter node is an abstraction of parameter expression. It stores information about the query subtree corresponding to the parameter expression: it includes references to the root of expression sub-tree, reference to the variables, and references to the data sources of variables. The parameter nodes play vital role for the representation of rules as well as for the rule matching procedure. After a parameter expression is represented by means of a tree and tied to the query tree, a method for the evaluation of parameter node can compute the output from given inputs by means of a tree representing parameter expression.

Symbol table `symtab` is used for handling query variables during parsing and type-checking, and for preparing rules for matching. A symbol name appearing in the query expression can be either the name of data source, query variable or the name of a span. The entry of symbol table includes the name of symbol and the reference to the query node representing data source, variable or span.

2) Path expressions: Path expressions of RDF query language SPARQL [27] called *property paths* are more expressive than the path expressions of classical relational and object-relational systems.

The path expressions that have to be implemented are expressions on properties that are based on regular expressions. Query processing system treats complex path expressions as predicates of relational system. For each objects of a collection complex path expression is evaluated separately.

C. Query optimizer

The design of the query optimizer is based in many aspects on the design of System R [1] and on the work of Graefe [10], [12]. The query optimizer performs a global optimization at master site.

The query expressions are in the query optimizer represented by query trees [14]. The inner vertices of the query tree represent operations comprising the query. The leaves of the query tree represent access paths. The logical equivalences among the query expressions are presented by means of transformation rules [7]. The internal representation of the transformation rules is also based on the query tree representation.

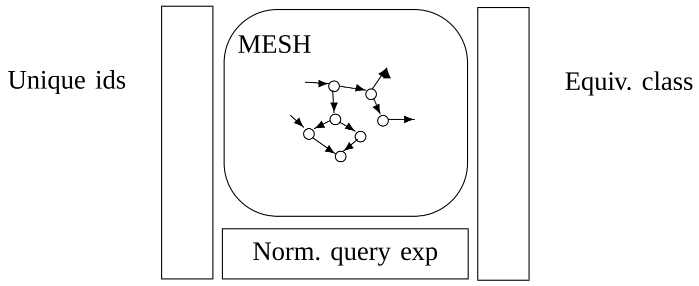


Figure 1. Mesh with access paths

The query optimization subsystem is composed of the following main modules. The query tree manipulation module includes routines for manipulation of query trees, application of rules on the nodes of query trees, and property functions by which the physical and logical properties of the query tree nodes are determined.

The cost function is realized by a separate module. The extensions of the cost functions used for the relational query optimizer are defined. The optimization module comprises the algorithms for the optimization of query expressions. The algorithms which are studied are the exhaustive search and an algorithm based on dynamic programming.

The core of the module is data structure `mesh` for the representation of sets of queries. Common sub-expressions of queries are shared ie. each query is represented in `mesh` only once. Queries are organized into equivalence classes. Let us first present the data structure Mesh.

1) Mesh: `mesh` stores query expressions as query trees organized as a directed acyclic graph (dag). The query trees share common parts hence there is only one representation of a query expression in `mesh`. Further, the query trees are organized into equivalence classes including logically equivalent queries. `mesh` has three entry points.

First, queries can be accessed using the unique identifiers which are created when query is entered in `mesh`. The mapping is realized between query trees (roots of) and the entries in `mesh` holding the queries.

Second, queries in `mesh` can be accessed through the equivalence classes which again have unique identification generated on the creation of the equivalence class from the first query expression. The inverse relationship from the query trees (roots of) to the equivalence classes is also defined.

Third, queries can be accessed using the normalized query expressions (character strings) as the hash index targeting roots of corresponding query trees.

In spite of various access methods to `mesh`, queries are defined free of the cover data structures. Each query node includes solely a reference to the equivalence class to which it belongs.

2) Optimization algorithm: The algorithm for query optimization is based on dynamic programming. The optimization is performed top-down: the procedure starts in the root, descends recursively to sub-trees, computes all logically equivalent queries, places them in the equivalence class, picks the next query from equivalence class and starts at the beginning until all alternatives are considered.

From the point of view of the actual computation of optimal query tree, this query optimization algorithm works bottom-up: the leaves of the query tree are optimized first progressing then upwards toward the root of the query tree.

We use the variant of dynamic programming algorithm called *memoisation* which stores the optimal results of the sub-queries and uses them in the computation of the composed queries. Memoisation is realized in a very simple manner. Every time a query is to be optimized we first check Mesh if the optimal query already exists for a given equivalence class of input query.

At this point we can see the use of Mesh for *plan caching*. If we do not clean Mesh after the execution of a given query that the optimization of the subsequent queries can make use of the existing optimized queries in Mesh. The optimization time is reduced significantly.

Let us now present some aspects of the complexity of query optimization algorithm. The equivalence class comprises a sorted list of queries that are the candidates for the optimization. The cost function is used to order the list. The search space can be reduced by selecting only the most promising queries. This type of optimization is called a *heuristic beam search*. The algorithm is sub-optimal since heuristic function based on cost estimation is not admissible or monotonic.

The practical experiences show that the presented algorithm based on dynamic programming is fast enough for the optimization of complex bushy trees with up to 10 classes.

3) Rules: The *query transformation rules* are used for the transformation of query trees into logically equivalent query trees that have different structure and potentially a faster evaluation method. The logical optimization rules are in Qios specified in a language that follows strictly the syntax and the semantics of the Qios query expressions.

A rule is composed of the *input pattern* and the *output pattern*. The input and output patterns are connected by means of common variables and common data sources. The links among the variables define the meaning of the rule. The input and output patterns, ie. query expressions, are ”terminated” using special operators called *spans*. A span is a virtual operation which can match any algebra operation. The rule (i/o) patterns can then be seen, from the perspective of parse trees, as the upper parts of query trees with the abstract leaves.

The query transformation rules are from the external form (augumented query expressions) transformed into the representation which is based on the query tree representation of

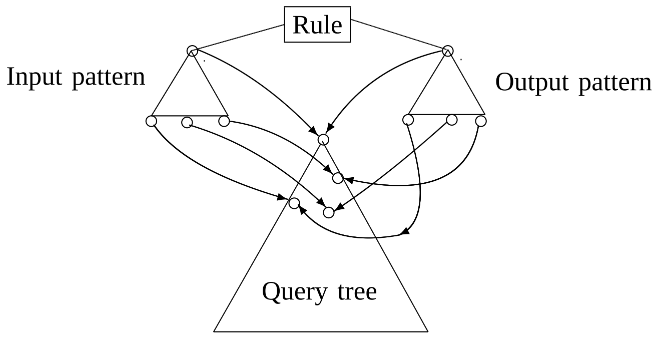


Figure 2. Query transformation

query expressions. The input and output patterns of the rule are represented by the input and output query trees. These are connected by the links relating variables, data sources and parameters of the input and output patterns of rule as presented in Figure 2.

Given a query tree and a rule, the matching procedure can be viewed as an attempt to cover the root of query tree with the input rule pattern ie. query tree. If matching is successful, types of the output pattern of rule are checked in order to verify correctness of query expression. If type checking procedure succeeds the output pattern of the rule is duplicated generating in this way a logically equivalet query of different structure.

D. Query evaluation system

The query evaluation module is based on the iterator-tree representation of the query evaluation plans. The physical query execution plan is computed from the optimized query trees by adding to the existing query nodes information about physical operation that will implement given logical operation (query node). The query nodes already contain information about the statistics, index selection, and cost estimation.

The main strategy which was used in the implementation of query execution is to select reasonably fast access methods on-the-fly without considering alternatives. Simple rules are used for index selection. Firstly, if selection or join is based on equality of attributes than hash-based index is generated. Secondly, if selection or join involves range predicate we use B-tree index. The selection of query execution plan is implemented by the procedure which computes physical operations for all logical operations (query nodes) forming the physical query tree in a bottom-up manner.

The physical algebra operations implemented are sequential scan, nested-loop join, hash-based index, and index-based nested-loop join. The hash-based access methods allow the efficient evaluation of the queries including equality and index-based plan serves for the evaluation of range queries. The procedures for the selection of physical operations are integrated in the routines that open scan operator for query nodes (logical operators).

After the selection of the physical operations the same skeleton of the query tree is employed as the iterator tree for the evaluation of physical query. The query evaluation can be seen as the tree structured pipeline where sub-nodes need to provide the next tuple for the evaluation of the current query node [10], [12]. The result of the logical query optimization are bushy query trees. The results of the inner sub-trees have to be materialized in order to avoid repeating evaluation.

E. RDF loader

RDF loader can read RDF files from Internet or locally. Loader is implemented using RDF::Trine module. Document is loaded into internal data model triple at a time. Schema must be loaded first to construct classes and types of the repository. Further, concrete triples are loaded to construct objects which must be instances of previously loaded classes.

There are many examples of RDF data repositories on the Internet that do not include schema part. For this purpose, RDF loader can compute on-the-fly the least general type (class) covering all instance of a given kind. Algorithm for RDF schema discovery is simple since we build objects on the basis of common object identifier that gathers triples with the same domain.

Currently we do not implement complete RDFS. More study is needed to be able to adequately model properties as objects i.e. instances of classes that can be specialised.

III. Related work

3store is designed to handle up to 100M serialized triples. They use three layered model; RDF representation can be syntactical, model-based in the form of triples and stored in MySql database system. Two tables are used for indexing resources and literals. Index value (hash) serves as unique identifier of resources and literals. These unique identifiers are then used for table representing triples. SPARQL queries are translated into relational calculus that can be directly transformed to SQL.

Bigdata is a triple-store that uses three main relations to store RDF database: Lexicon, Statements (triples) and StatementTypes. Exhaustive set of indexes is created for relation Statements which represents triples. RDF query is based on statement patterns. Bigdata defines a perfect access path for each type of access pattern. System includes also a form of reasoning with triples as well as reasoning on the basis of RDFS statements. Furthermore, the system is scalable to several 100 machines.

Virtuoso is a relational system including functionalities for relational and object-relational data management, XML and RDF data management, Web services deployment, text content management services, full-text indexing as well as Web document server and Linked-data server. RDF data is stored in a table including triples which can be indexed using object-relational system. SPARQL is implemented by translating queries into SQL—equivalent queries can be expressed directly in SQL.

Let us now present the work related to the implementation of the presented query execution system. Firstly, the implementation is closely related to the implementation of Query Algebra originally proposed by Shaw and Zdonik in [26] and implemented by Mitchell [18]. In particular, we have used a similar representation of query expressions by means of query trees. Furthermore, the representation of query expressions in Qios is optimized by using single operation nodes and query trees during all phases of query processing.

The design of the query execution system was based on the design of Exodus optimizer generator [9] and its descendant Volcano [12]. The data structure MESH used in Exodus query optimizator generator is improved by adding additional access paths. The data structure can be accessed through: unique identifier, normalized query expression, and equivalence class. The algorithm for query optimization is rooted in Graefe's work on Volcano optimizer algorithm [12]. This algorithm uses top-down search guided by the possible "moves" that are associated to a query node. The algorithm uses memoisation to avoid repeated optimization of the same query. The search is restricted by the cost limit which is a parameter in optimization.

IV. Conclusions

The paper presents the architecture of query engine used for querying RDF databases. While the architecture is based on relational and object-relational technology there are many specific components and techniques which are implemented to reflect RDF data model and the nature of RDF databases.

Firstly, since RDF model uses triples for modeling conceptual schemata and instances the data model of storage system maintains a single set of object identifiers. Object values are tuples including concrete values while class values include class identifies and represent types. Classes are in many ways treated as ordinary objects reflecting the graph data model of RDF. Secondly, path expressions of RDF data model are more expressive than path expressions of object-relational data model [27]. For this reason treatment of predicates of selection and join operations must be considered much more carefully. Finally, algebra of query processing system have to include operations for grouping and restructuring to be able to support complete functionality of SPARQL.

References

[1] M. Astrahan, Et.al., 'System R: relational approach to database management', ACM Transactions on Database Systems, Vol. 1, Issue 2 June 1976, pp. 97-137.

[2] F. Bancilhon, ‘Object-Oriented Databases’, The Computer Science and Engineering Handbook, 1997.

[3] ‘Bigdata Scale-out Architecture’, Whitepaper, SYSTAP, 2009.

[4] E.F. Codd, ‘A relational model of data for large shared data banks’, Comm. of the ACM, Vol. 13 , Issue 6, June 1970, pp. 377-387.

[5] A. de Jonge, ‘Comparing XML database approaches’, IBM Corporation, 2008.

[6] D. Daniels, P. Selinger, L. Haas, B. Lindsay, C. Mohan, A. Walker, P. Wilms, ‘An introduction to distributed query compilation in R*’, IBM Research Report RJ3497 (41354), June 1982.

[7] J.C. Freytag, ‘A rule-based view of query optimization’, Proc. of ACM Conf. on Management of data, 1987.

[8] G. Gardarin, J.-R. Gruser, Z.-H. Tang, ‘Cost-based selection of path expression processing algorithms in object-oriented databases’, In Proc. Int. Conf. on VLDB, 1996.

[9] G. Graefe, D.J. DeWitt, ‘The EXODUS Optimizer Generator’, Proceedings of ACM SIGMOD international conference on Management of data, 1987, pp. 160-172.

[10] G. Graefe, ‘Query Evaluation Techniques for Large Databases’, *ACM Comp. Surveys*, Vol.25, No.2, June 1993, pp. 73-170.

[11] G. Graefe, ‘Query processing’, Slides, ICDE Influential Paper Award, 2005.

[12] G. Graefe, W. McKenna, ‘The Volcano optimizer generator: extensibility and efficient search’, Proc. of IEEE Conf. on Data Engineering, April 1993, p.209.

[13] S. Harris, N. Gibbins, ‘3store: Efficient Bulk RDF Storage’, Proceedings 1st International Workshop on Practical and Scalable Semantic Web Systems, 2003.

[14] M. Jarke, J. Koch, ‘Query optimization in database systems’, *ACM Comp. Surveys*, Vol.16, No.2, June 1984.

[15] ‘Linked Data - Connect Distributed Data across the Web’, http://linkeddata.org/, 2012.

[16] A. Marathe, ‘Batch Compilation, Recompilation, and Plan Caching Issues in SQL Server 2005’, Microsoft, White paper, Dec. 2006.

[17] T. Milo, D. Suciu, ‘Index structures for path expressions’, In Proc. 7th Int. Conf. on Database Theory, 1999.

[18] G.A. Mitchell, ‘Extensible Query Processing in an Object-Oriented Database’, Ph.D. thesis, Brown University, 1993.

[19] ‘Oracle Database, Documentation Library (11.2)’, Oracle Corporation, 2012.

[20] ‘Resource Description Framework (RDF)’, http://www.w3.org/RDF/, 2004.

[21] M.A. Roth, H.F. Korth, A. Silberschatz ‘Extended algebra and calculus for non 1NF relational databases’, *ACM Trans. Database Systems*, 13(4), 1988, 389-417.

[22] I. Savnik, ‘Algebra for distributed data sources’, Dagstuhl Seminar ”Multimedia Database Support for Digital Libraries”, Schloss Dagstuhl, Germany, Aug 1999.

[23] I. Savnik, Z. Tari, T. Mohorič, ‘QAL: A Query Algebra of Complex Objects’, *Data & Knowledge Eng. Journal*, North-Holland, Vol.30, No.1, 1999, pp.57-94.

[24] I. Savnik, ‘On formalization of object model by unifying intensional and extensional representations’, Information Modelling and Knowledge Bases XXI, Frontiers in Artificial Intelligence and Applications, IOS Press, 2010.

[25] I. Savnik, Z. Tari, ‘QIOS: Querying and Integration of Internet Data’, http://osebje.famnit.upr.si/˜savnik/qios/, Famnit, University of Primorska, Koper, 2009.

[26] G.M. Shaw, S.B. Zdonik, ‘A Query Algebra for Object Oriented Databases’, Proc. of IEEE Conf. on Data Engineering, 1990, pp. 154-162.

[27] S. Harris, A. Seaborne, ‘SPARQL 1.1 Query Language’, W3C Working Draft, http://www.w3.org/TR/sparql11-query/, 2012.

[28] ‘Information technology, Database languages, SQL’, ISO/IEC JTC 1/SC 32, ANSI, 2007.

[29] ‘OpenLink Virtuoso Universal Server: Documentation’, OpenLink Software Documentation Team, 2009.

Using Social Network Information to Identify User Contexts for Query Personalization

Diogo Alves, Marcelo Freitas, Thiago Moura, Damires Souza
Informatics Academic Unit (UAI)
Federal Institute of Education, Science and Technology of Paraiba (IFPB)
Av. 1^{o} de Maio, 720, Jaguaribe, João Pessoa, Brazil
{diogoca, marcello.dudk, tmoura}@gmail.com, damires@ifpb.edu.br

***Abstract*-In recent years, social networks have gained a huge popularity among internet users, serving diverse purposes and communities. Meanwhile, in data-oriented applications, the increasing amount of available data has made it hard for users to find the information they need in the way they consider relevant. To help matters, a user-centric approach may be used to enhance query answering and, particularly, provide query personalization. In this work, we address the issue of personalizing query answers in data-oriented applications considering the user context provided by social network information. To this end, we propose a context-aware plugin named CODI4In. The CODI4In extracts users' social network information regarding their "likes" and use them as context information to provide query personalization. In this paper, we present the developed approach and some experimental results we have accomplished with real users. These results show that by considering the acquired user context really enhances the degree of relevancy of the obtained personalized answers.**

Keywords-Context, User Context Management, Query Personalization, Social Network Information

I. INTRODUCTION

Personalization, in a general sense, means tailoring a product or a medium to a user, according to some identified user personal characteristics [1]. Regarding query answering, in computational settings, it aims to assist users when formulating queries in order to enable them to receive relevant information, where such relevancy is defined by a set of criteria specific to each user [2]. One of the primary ways to achieve query personalization is user profiling, so a query can be related with user preferences stored in a user profile [2]. On the other hand, query personalization may be also considered a machine learning process based on some kind of user feedback or identified usage [3]. In fact, when formulating queries, the user may be found in various contexts, and these contexts may change every time. Meanwhile, the user himself may build his own context, in terms of his specific interests, preferences, relationships and common executed tasks. Considering that, in order to provide query personalization, we argue that it is essential to take into account the user model. Moreover, to build the user model, we should include the *user context*.

The *context* may be understood as the circumstantial elements that make a situation unique and comprehensible [4]. We consider *Context* as a set of elements surrounding a domain entity of interest which are considered relevant in a specific situation during some time interval. The domain entity of interest may be, for instance, a person (e.g., a user) or a task (e.g., a given query). In addition, we use the term *contextual element* (CE) referring to pieces of data, information or knowledge that can be used to define the Context [5]. Regarding the user, his context (e.g., location and preferences) can be exploited by a system either to answer queries or to provide recommendations, so users at different locations or different perceived preferences may expect different results, even from a same formulated query.

Context information may be acquired from diverse sources. Considering online social networks, the users' social profiles are rich sources of information about their preferences (e.g., likes and dislikes) and relationships (friendship) [6][7]. Indeed, in our view, the information extracted from the social user profile provides clues to identify what is relevant to a query submitted by him in another application he is interacting with. For instance, in a data-oriented application, if he is querying about movies and, from his social network profiles the application knows he mostly likes comedy movies, thus retrieved movies from this category can be depicted firstly. With this in mind, we propose a query personalization approach which makes use of social network information as a kind of contextual element.

In order to provide the user context management, we have developed a plugin named CODI4In [8]. The CODI4In manages user context information, providing the persistence and recovery of the contextual elements (CEs) using an ontology. In this work, the CODI4In has been extended. It extracts information from a social network, particularly the Facebook [9], manages this user information as a CE, and uses it as a means to provide personalized answers (in this current version, ranked answers). Experimental results show that by considering the user context provided by social network information really enhances the degree of relevancy and satisfaction of the obtained personalized answers.

Our contributions can be summarized as follows:

(i) We acquire user context information from a social network;

(ii) We manage user context information using an ontology, and a graph-based database as the underlying storage model;

(iii) We present a case-study coupling the CODI4In with a web based data-oriented application; and

(iv) We describe experiments with real users showing the degree of relevancy obtained with the personalized answers produced by considering context information from a social network.

This paper is organized as follows: Section 2 introduces the use of social networks; Section 3 proposes the CODI4In approach; Section 4 describes the developed CODI4In approach and some accomplished experiments. Related work is discussed in Section 5. Finally, Section 6 draws our conclusions and points out some future work.

II. SOCIAL NETWORKS AND USER INFORMATION EXTRACTION

Recently, the use of social networks has gained great popularity. Statistical evidence indicates that not only more people are joining these communities, but also there is an increase in the average amount of time spent [10]. With all this time spent using social networks, more and more information about their users are generated and stored. A social network is usually a place for sharing content in different forms, e.g., short messages on Twitter [11] or interests such as friendship, personal likes and dislikes in Facebook [9]. What is more interesting about that relies in the fact that users themselves are responsible for keeping all these information up to date in the way they consider important.

This introduces the concept of social profiling, which can be valuable to be used by other applications. When a new user connects to a given application for the first time and fewer details about him are available, information gathered from a social network can be used to build an initial user model. A system that has access to social networks can make use of such information in different ways. As an illustration, the SmartObject system considers the user's relationship information to turn on the audio player when friends are online [12].

Most online social networks already give some access to social information, but they limit what data can be accessed and how it can be used [7]. In this work, we deal with information extracted from users of the social network Facebook as a way to build the user context model. The Facebook is particularly interesting because some features extracted from the user profiles regarding “likes” (specially related to movies) provide clues to identify what is relevant to personalize the user queries in a data-oriented application the user interacts with. To make available some information from the users’ profiles, the Facebook provides a personal API [13], thus enabling applications to access information set as public. Examples of such public information are user name, age and gender. It is not possible to extract more information, unless the user make them available as public. In our work, users are asked to allow the CODI4In plugin (to be explained in next section) to access information about the movies that users have liked.

III. THE CODI4IN APPROACH

In this section, we describe the CODI4In approach. To this end, we first present the CODI-User ontology and then we show the main issues underlying the CODI4In architecture.

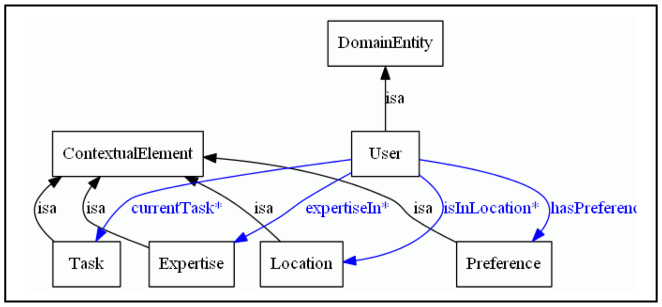


Figure 1. An excerpt from the CODI-User Ontology.

A. *CODI-User: The User Context Ontology*

We represent and store *user context* information in an ontology named CODI-User. The CODI-User ontology includes contextual elements (CEs) regarding *personal*, *environment* and *query* related concepts, which are used to personalize queries. Figure 1 describes some of the CEs that have been specified in order to characterize the domain entity USER. Such view has been produced using OntoViz, a Protégé plug-in [14]. Therefore, *User* is a sub-concept of Domain Entity. *Location*, *Task*, *Interest*, *Expertise* and *Preference* are sub-concepts of Contextual Element. These elements are each one related to User. In this view, we only have metadata, we do not show instances.

To create a simple yet extensible model, we defined diverse CEs that could be useful in different kinds of data-oriented applications. The CEs may be divided into three views: (i) *general query personalization concepts*, (ii) *environment concepts* and, (iii) *personal concepts*. Regarding *the first one*, we consider the user task at hand (in our case, it means a query), the user identification, his interests (e.g., hobbies or work-related interests) and his specific preferences related to the task at hand (i.e., to a query). *Environment concepts* regard the setting where the user interacts and the application is executed. We have primarily chosen the following CEs: the user location (his current geographical position), the kind of connection (his IP address identification), the device at hand and the kind of interface the user is interacting with (e.g., textual, visual). In addition, depending on the kind of application (e.g., e-commerce), the expertise, the group which the user belongs to as well as his *personal information* such as email or birth date are also considered. Although we have defined these three views, the CODI-User ontology may be extended through inheritance and the addition of more concepts, as well as concept instantiation according to the application needs.

B. *Architecture*

After defining the CODI-User concepts, we have been working on a service concerned with the storage and retrieval of the CEs. The CODI4In service has been defined and developed as a plugin in such a way that data-oriented applications can be coupled to it. In this sense, the CODI4In plugin operates as a back-end service of a data-oriented application which works as the front-end, as depicted in Figure 2. The CODI4In supports the persistence and recovery of CEs related to an identified user that interacts with the coupled application. It includes the ability to acquire user context information from multiple sources, e.g., from physical sensors or from explicit information provided by the user. To this end, it provides a common interface so that diverse adapters can be built to gather information from various sources. Particularly, in this work, it acquires context information from the Facebook social network. This information is then persisted in the CODI-User as a kind of CE. Using this information as context relieves the user from the burden of specifying details, focusing on queries he is interested in.

The various user CEs (e.g., location, interests, preferences) required to build the user model may be stored as ontology instances in the CODI-User. The CODI4In populates such ontology and retrieves the CEs when required to identify the user or to personalize a given query. The query personalization may be accomplished as: (i) a query expansion, introducing CEs in the submitted query; or (ii) a post-processing step after query results have been generated. In this work, we personalize user queries following the latter option, using a ranking algorithm on the resulting query answers. Thus, the CODI4In is able to acquire context information from user profiles (in this case, from the Facebook). Then it ranks the query answers according to the CEs it has gathered as relevant to make a decision on the ranking. These ranked answers are forwarded to the application to provide means to present them to the user.

An example of a data-oriented application that can gain from using the CODI4In is a web-based application named MovieShow, which has been developed and coupled to the plugin as our first case study (it will be presented in Section 4). Other applications, e.g., query applications or recommender systems, which need to work with personalized queries over data, may benefit from using it as well.

Although in this current version, we are using context information provided by the Facebook, the population process at the CODI-User ontology may be also accomplished during the user registration (if this is the option underlying the front-end application) or through on-going user interactions, when the user is submitting queries or defining parameters that can be identified as context. Such population process is accomplished in a dynamic and incremental way.

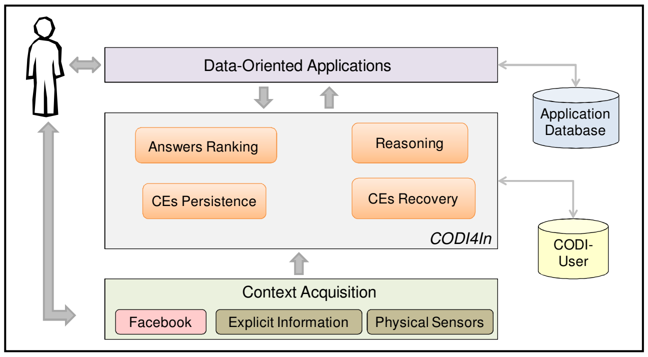


Figure 2. CODI4In Architecture Overview.

IV. IMPLEMENTATION AND RESULTS

In this section, we present the CODI4In implemented with the Facebook-based Context Acquisition, showing a high-level main algorithm. We also describe some implementation issues, a case study and some experimental results.

A. The Algorithm

The principle of our approach is to enhance query personalization by using information from the Facebook user profiles as CEs. Answers produced by the CODI4In algorithm are consistent with what the user has defined as relevant through his "likes". A high level view of the CODI4In main algorithm is sketched in Figure 3. To acquire the movie genres information, we use the IMDB API [15].

In order to provide query personalization, the algorithm performs the following tasks:

I. Once the user has logged in the Facebook and allows the CODI4In to retrieve his personal information and his "likes", the CODI4In gets a specific token (access key) to request the information needed to be dealt with (i.e., the movies that the user has liked).

II. The CODI4In then iterates over the list of obtained movies. To each obtained movie, it recovers all genres using the IMDB API and stores it in an auxiliary list called "preferred genres".

III. When the iteration process is finished, the algorithm ranks this auxiliary list according to the number of occurrences of each genre and removes duplicated names.

IV. The basic information of the user profile and preferred genres captured on the fly are persisted in the CODI-User ontology. These CEs will be used to enhance the ranking of query results, providing a kind of query personalization.

```
getUserContext()
  // When user logins using the Facebook login option and allows
  // to recover his personal data and liked options (e.g., movies)
  token = getUserToken()
  // Retrieves basic information
  user = getFacebookUser(token)
  // Retrieves movies which have been liked
  moviesList = getFacebookLikedMovies(user)
  // New empty collection
  preferredGenres = []
  for each moviesList in movie
    // Retrieves movie genre names from the IMDB API
    genres = getGenresFromImdb(movie.name)
    // Storing
    preferredGenres.push(genres)
    if (at last one of moviesList = movie)
        // Order by number of occurrences
        sortGenresByOccurrence(preferredGenres)
        // Removes duplicate genres
        uniqueGenres(preferredGenres)
        // Storing in plugin
        insertCE(user, preferredGenres)
    end if
  end for
endGetUserContext()
```

Figure 3. High Level View of The CODI4In Main Algorithm.

B. *Implementation Issues*

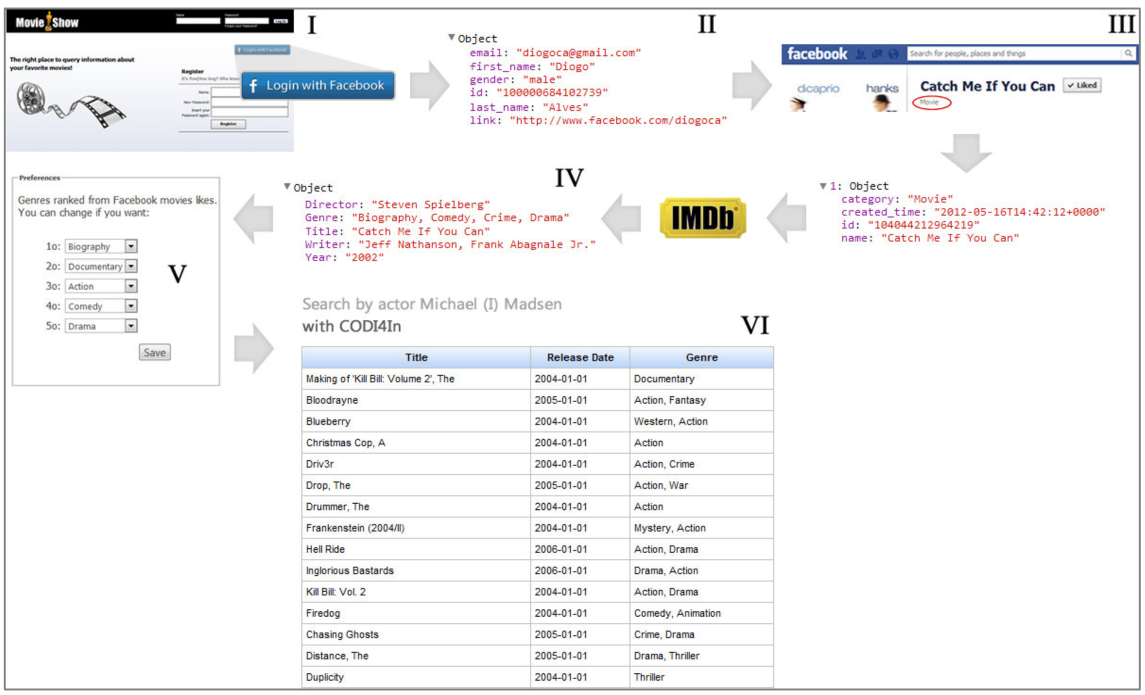
We have developed the CODI4In plugin and the presented algorithm in Java. Since our representation model is an ontology (which may be represented as a graph [16), we have used, as the storage model, a graph-based database called Neo4J [17]. Such database stores the CEs and user instances as the nodes and relationships of a graph, what allows preserving the natural structure of the CODI-User ontology. Besides, to gather information from the Facebook, we have used its specific API and the JavaScript SDK.

As a case study, we have implemented a web based data-oriented application to be used as front-end to the CODI4In back-end service. This application allows users to submit queries about movies and has been named MovieShow. In order to deal with movies information, we have imported data from the IMDB into a local relational database. In this database, we have four tables, as follows: Movie (Id, Title, Release Year, Genre, Actor, Director), Actor (Id, Name, BirthDate) and Director (Id, Name, BirthDate).

When logging to the MovieShow application, the user is required to allow the system to deal with his profile information from the Facebook. If he agrees with that, he is invited to log into the social network. If he does not want to allow such access, he can register in the MovieShow application and log into as well. To help matters, in this work, we are considering only the first option, i.e., the user allows the CODI4In to extract information from his Facebook profile.

When logged in the Facebook and also in the application, the CODI4In retrieves information (in JSON format) from the user profile, as follows: (i) personal information, such as first_name, last_name, email and gender, and (ii) preferences information regarding, in this case, the user "likes" about movies. Particularly, in this case study, the CODI4In acquires the titles of the movies the user has liked. With the set of movies titles at hand, the CODI4In interacts with the IMDB web service and retrieves the list of the given movies genres. These genres are persisted in the CODI-User as user preferences on the movies. Also, the CODI4In sets a ranked list of these preferences on the fly. The user is able to refine this list, but if he does not matter, the CODI4In uses this ranked list to personalize user query results. To this end, it applies an ordering method (Bubble sort). Thereby, the SQL query is not reformulated, only its results are. Figure 4 depicts the overall process to accomplish the context acquisition by using the Facebook user profiles, the CEs management, the query personalization step and the answers presentation.

As an illustration, in Figure 4, the user named "Diogo" logs into the Facebook and into the Movieshow application (step I). Diogo allows the plugin to retrieve his profile information. The CODI4In thus acquires his personal information (step II) and the movies he has liked. In this example, Diogo has liked the movie "Catch me if you can" (step III). With this information at hand, the CODI4In retrieves its genres, using the IMDB web service, set as "Biography, Comedy, Crime and Drama" (step IV). In this example, we illustrate this operation with only one movie, but indeed all the user liked movies are taken into account. Thus, the genre elements of all Diogo's liked movies are processed, i.e., the obtained list of genres is ranked according to the number of occurrences of each genre. Also duplicated genre names are removed. Then, they are persisted in the CODI-User ontology and depicted to the user by means of the MovieShow interface (step V). The movie genres are relevant to personalize queries submitted by Diogo in the MovieShow application. Finally, Diogo submits a query about movies starred by the actor "Michael Madsen". The



```
▼Object
   email: "diogoca@gmail.com"
   first_name: "Diogo"
   gender: "male"
   id: "100000684102739"
   last_name: "Alves"
   link: "http://www.facebook.com/diogoca"
```

```
▼1: Object
   category: "Movie"
   created_time: "2012-05-16T14:42:12+0000"
   id: "104044212964219"
   name: "Catch Me If You Can"
```

```
▼Object
   Director: "Steven Spielberg"
   Genre: "Biography, Comedy, Crime, Drama"
   Title: "Catch Me If You Can"
   Writer: "Jeff Nathanson, Frank Abagnale Jr."
   Year: "2002"
```

Preferences

Genres ranked from Facebook movies likes.
You can change if you want:

1o:	Biography
2o:	Documentary
3o:	Action
4o:	Comedy
5o:	Drama

Search by actor Michael (I) Madsen
with CODI4In

Title	Release Date	Genre
Making of 'Kill Bill: Volume 2', The	2004-01-01	Documentary
Bloodrayne	2005-01-01	Action, Fantasy
Blueberry	2004-01-01	Western, Action
Christmas Cop, A	2004-01-01	Action
Driv3r	2004-01-01	Action, Crime
Drop, The	2005-01-01	Action, War
Drummer, The	2004-01-01	Action
Frankenstein (2004/II)	2004-01-01	Mystery, Action
Hell Ride	2006-01-01	Action, Drama
Inglorious Bastards	2006-01-01	Drama, Action
Kill Bill: Vol. 2	2004-01-01	Action, Drama
Firedog	2004-01-01	Comedy, Animation
Chasing Ghosts	2005-01-01	Crime, Drama
Distance, The	2005-01-01	Drama, Thriller
Duplicity	2004-01-01	Thriller

Figure 4. An Example of the Algorithm Instantiation and a given Query Personalization.

query is executed and its answers are ranked according to the CEs regarding Diogo's preferences on movies genres. A snapshot from the MovieShow application with the set of ranked answers for the user Diogo is shown in Figure 4 (VI). Thereby, the user Diogo firstly receives documentary and action movies as query answers, according to the identified genres preference order. If a retrieved movie genre (from the submitted query) does not match the list of user preferences, it comes at the end of the list.

In this sense, we observe that the CODI4In can change the original query algorithm by integrating a restriction obtained through the identified CEs (in this example, concerned with the genre preferences order). As a result, data presented to the user are ranked according to each user identified context model. In this case study, the user context model has been built using information provided by a social network. This implementation may be extended to consider other CEs related to queries, thus providing more specific personalization.

C. Experiments

We have conducted some experiments to verify the effectiveness of our approach. The goal is to check if enabling the CODI4In plugin would provide benefits to users, i.e., if the produced query answers were really considered as more suitable in terms of results ranking. To this end, we have invited some users (a total of 30) to evaluate our prototype. The users group was composed by undergraduate students in Computer Science as well as from people from other areas (e.g., Engineering). At first, we explained the goal of the evaluation together with the main objective of the CODI4In. We also discussed the usage of the social network as a source of context information and the need of agreement on its access. We asked them to provide "likes" on some movies by using the Facebook, if they had not done it yet.

Then, users received a survey containing questions divided into three main issues: (i) personal information such as age and occupation, (ii) frequency of using the Facebook to "like" movies and (ii) feedback on the relevancy of the obtained answers when the CODI4In was enabled and his "likes" were taken into account as CEs. More specifically, we wanted to know if there is any difference in the query results in terms of its ranking, when considering or not the user context. To answer the survey, they spent a time submitting a number of queries about movies according to actors, directors and release year. They could verify the obtained query answers without enabling the CODI4In as well as by enabling the plugin. When the CODI4In was enabled, the answers were ranked according to the user genre preferences ranking.

As shown in Figure 5, liking movies using the Facebook is not an usual task in this group of users. The reason underlying that is that most of them are more interested in the messages posting instead of this specific option. Nevertheless, they became curious and particularly engaged in this new task.

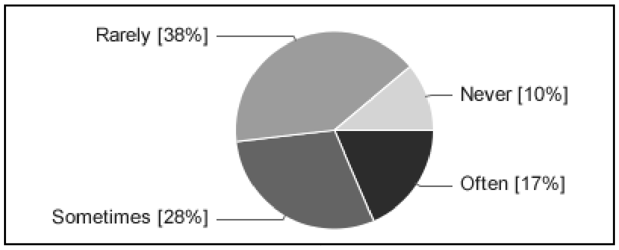


Figure 5. Frequency at which Users "Like" Movies on the Facebook.

Regarding the benefits obtained from the CODI4In enabling, the great majority of them were very satisfied with this new functionality. They considered as very relevant the answers ranked according to their likes (Figure 6). On the other hand, they reported that the response time, when considering the user context, was slower than when the queries were executed without the plugin. In some cases the movie genre has not been returned. This occurred when the Facebook returned a movie with a Portuguese title (not yet translated), thus it could not be found using the IMDB service.

Thus, we could verify the effectiveness of our approach. We have confirmed that not only is personalization highly essential (comparing the ranked results with the original ones), but also that our techniques are promising to proceed.

V. RELATED WORK

Query personalization techniques have been tackled in diverse environments. The works of Koutrika and Ioannidis [2], Stefanidis *et al.* [3] and Arruda *et al*. [18] consider user preferences to personalize queries. The first one [2] provides query personalization in databases based on user profiles. The second one [3] provides a recommendation system that expands query results according to user preferences. This system can compute query results by considering the user history and the current state of the query and the database. Arruda *et al*. [18] implemented a query module in a PDMS (Peer Data Management System) that enables query personalization. This is accomplished by means of the choice of which correspondences (mappings) should be considered when reformulating a query between two peers. As a result, query answers are ranked according to the priority established for the correspondences.

In the field of context-aware systems, there has been significant amount of research effort for modeling and managing context information.

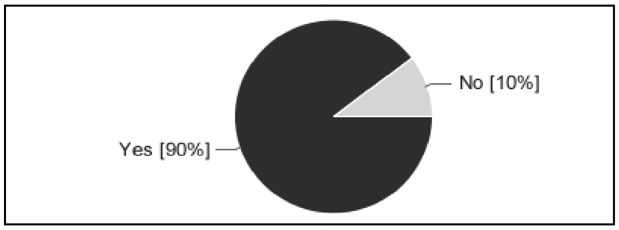


Figure 6. Degree of Relevancy of the Produced Answers when Using the CODI4In

The Context-ADDICT (Context-Aware Data Design, Integration, Customization and Tailoring) project [1] has been working in the development of a framework which, starting from a methodology for the early design phases, supports mobile users through the dynamic hooking and integration of new, available information sources, so that an appropriate context-based portion of data is delivered to their mobile devices.

Kabir *et al*. [6] introduce the SCIMS, a social context information management system which uses an ontology based model for classifying, inferring and storing social context information, in particular, social relationships and status. It also provides an ontology based policy model and language for owners to control access to their information.

The CareDB project [19] addresses the goal of embedding support for context and preference-aware query processing within a database system. It has been developed as a complete relational database system, supporting two main core functionalities: (i) various preference evaluation methods (e.g., skyline) and (ii) integration of surrounding contextual data (e.g., traffic, weather). Both functionalities are included within the query processor.

Karapantelakis and Maguire [20] present a system that uses social network information to recommend Web feeds of related content to users. The system mines data from popular social networks and combines it with information from third party websites to create user profiles. Then, these profiles are matched with appropriately tagged Web feeds and are displayed to users through a mobile device application.

Comparing these works with ours, most of them deal with user profiles, and some of them with context information. In our work, we provide a model to be used in any user context management solution, through an ontology. Differently from these works, our approach is not concerned with only providing context as a means to enhance query personalization, but also, providing a plugin to be coupled in data-oriented applications. Using the CODI4In plugin, the front-end application does not need to take care about the user context. In this current version, our approach lies in the lack of user intervention, since his preferences regarding movies genre are automatically acquired from his profiles in a commonly used social network. The CODI4In is able to identify these preferences from the user tagged “likes” and extract genres from the “liked” movies using information stored in the IMDB on the fly. These obtained preferences are persisted as CEs and used to provide query personalization in the front-end application.

VI. CONCLUSION AND FUTURE WORK

In data-oriented applications, the semantics surrounding queries are rather important to produce results with relevancy according to the users’ context. This work has presented the CODI4In - a plugin for retrieving the user context from social networks and then using this context information to personalize user queries submitted in a data-oriented application. The CODI4In accesses the information from a social network which has been allowed by the user, thus respecting his privacy.

In our case study, when CODI4In is enabled, CEs related to the user (in this case, movies’ likes and referred movies’ genres) are taken into account to rank query answers presented by the application. Experiments carried out with real users have shown that query answers have become more relevant when the context has been considered to rank them.

We are now extending the CODI4In along with a number of directions, including support for reasoning mechanisms and other kinds of query personalization algorithms. We are also specifying another application to be coupled with the CODI4In plugin. It will provide means to accomplish other kinds of experiments, including the usage of other social networks, e.g., the LinkedIn.

REFERENCES

[1] L. Tanca, C. Bolchini, E. Quintarelli, F. Schreiber, and G. Orsi, “Problems and Opportunities in Context Based Personalization,” Proc. VLDB Endowment (PersDB 2011), Vol. 4, No. 11, pp. 1 – 4, 2011.

[2] G. Koutrika and Y. Ioannidis, “Personalized Queries under a Generalized Preference Model,” 21st Intl. Conf. On Data Engineering (ICDE 2005), pp. 841 – 852, Tokyo, 2005.

[3] K. Stefanidis, M. Drosou, and E. Pitoura, “You May Also Like Results in Relational Databases,” Proc. 3rd International Workshop on Personalized Access, Profile Management and Context Awareness in Databases (PersDB 2009), pp. 37-42. Lyon, 2009.

[4] A. Dey, “Understanding and Using Context”. Personal and Ubiquitous Computing Journal, vol. 5 (1), pp. 4-7, 2001.

[5] V. Vieira, P. Tedesco, and A.C. Salgado, “Designing Context-Sensitive Systems: An Integrated Approach,” Expert Systems with Applications, vol. 38(2), pp.1119-1138, 2010.

[6] M. A. Kabir, J. Han, J. Yu, and A. W. Colman, “SCIMS: A Social Context Information Management System for Socially-Aware Applications,” CAiSE, pp.301-317, 2012.

[7] M. Rowe and F. Ciravegna, “Getting to me: Exporting semantic social network from facebook,” Proc. 1st Workshop on Social Data on the Web (SDoW2008), vol. 405, pp. 28-41. Oct. 2008.

[8] M. Freitas, J. Silva, D. Bandeira, A. Mendonça, A. C. Salgado, and D. Souza, “A User Context Management Approach for Query Personalization Settings,” Proc. 6th International Conference on Semantic Computing (ICSC), pp. 333-335, 2012, Palermo, Italy.

[9] Facebook, available at http://www.facebook.com. Accessed on November 26th, 2012.

[10] Nielsen Online. Global Faces and Networked Places. A Nielsen report on Social Networking's New Global Footprint. Available at http://blog.nielsen.com/nielsenwire/wpcontent/uploads/2009/03/nielsen_globalfaces_mar09.pdf.

[11] Twitter, available at http://www.twitter.com. Accessed on November 26th, 2012.

[12] G. Biamino, “Modeling social contexts for pervasive computing environments,” Pervasive Computing and Communications Workshops (PERCOM 2011), pp. 415- 420, 2011.

[13] Facebook documentation. Available at http://developers.facebook.com/docs/reference/api/. Accessed on December 2nd, 2012.

[14] OntoViz documentation. Available at http://protegewiki.stanford.edu/wiki/OntoViz. Accessed on November 26th, 2012.

[15] IMDB API, available at http://www.imdbapi.com. Accessed on November 26th, 2012.

[16] Y. An, J. Mylopoulos, and A. Borgida, “Building Semantic Mappings from Databases to Ontologies,” Proc. Twenty-First National Conference on Artificial Intelligence (AAAI), pp. 1557-1560. Boston, 2006.

[17] Neo4J. Available at http://www.neo4j.org. Accessed on November, 26th, 2012.

[18] T. Arruda, D. Souza, and A.C. Salgado, “PSemRef: Personalized Query Reformulation based on User Preferences,” 12th International Conference on Information Integration and Web-based Applications & Services (iiWas2010), pp. 681-684, Paris, 2010.

[19] J. Levandoski, M. Khalefa, M., “An Overview of the CareDB Context and Preference-Aware Database System”, In. Proc. IEEE Data Eng. Bull. 34(2), pp. 41-46. 2011.

[20] A. Karapantelakis, and G. Q. Maguire Jr, “Utilizing Social Context for Providing Personalized Services to Mobile Users,” Proc. 5th European conference on Smart sensing and context (EuroSSC 2010), pp. 28-41, 2010.

Finding Nearest Neighbors for Multi-Dimensional Data

Yong Shi, Marcus Judd
Department of Computer Science
Kennesaw State University
1000 Chastain Road
Kennesaw, GA 30144
yshi5@kennesaw.edu, mjudd3@kennesaw.edu

***Abstract*–Nearest Neighbor Search problem is an important research topic in data mining field. In this paper, we discuss our continuous work on finding nearest neighbors in multi-dimensional data based on our previous research work. The research work presented in this paper improves our original algorithm by analyzing the distribution of data points on each dimension**.

***Keywords*–Similarity Search, Multi-query, Data Point Weight**

I. Introduction

As one of the important research topics in the data mining field, the nearest neighbor search problem has been studied and various approaches have been proposed in different research scenarios [2], [8], [10], [11], [13], [16]. For example, [11] proposes a scheme for nearest neighbor search by hashing data points from a data set, so that for data points close to each other, the probability of collision is much higher than those that are far from each other. The authors also conducted experiments to demonstrate the accuracy and scalability of their algorithm. [13] presents a multi-step algorithm that produces the minimum number of candidates for nearest neighbor search problem. The algorithm works well for the efficiency requirements of complex high-dimensional and adaptable distance functions. [16] provides a detailed analysis of partitioning and clustering techniques for nearest neighbor search problem. The paper also discusses an alternative organization based on approximations to make the unavoidable sequential scan as fast as possible.

II. Related Work

Traditional approaches apply similarity functions such as Euclidean distance to calculate the distance between two data points in a given data set. Such approaches often have a problem called “curse of dimensionality”, since when dimensionality goes higher, the distance between two data points becomes less meaningful [9], [6], [16], [7]. There are approaches designed for partial similarities analysis [12], [4], [3], but most of them have the problem of lack of flexibility because they require fixed subset of dimensions or fixed number of dimensions as a part of the algorithm input .

In [14], we discuss the fact that in reality, we need to find nearest neighbors to multiple query points with different level of importance. We define the distance between a data point and a set of query points, taking into consideration how important each query point is, and how dimensions should be dynamically chosen for each data point. We apply our algorithm to find nearest neighbors in different subspaces of the original data space.

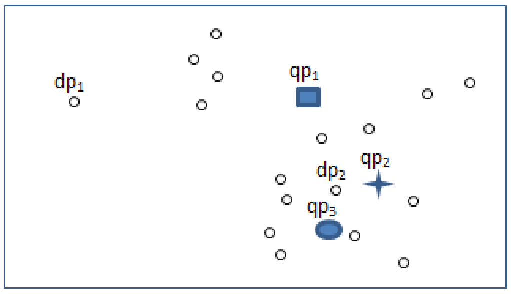


Fig. 1: A 2-dimensional data set with multiple query points where data points have different densities

III. Problem Definition

In this paper, we propose to enhance the process of the algorithm discussed in [14] by analyzing the data point distribution.

For example, figure 1 shows a 2-dimensional data set along with multiple query points. The hollow dots represent the data points. The solid square qp_1, the solid four-point star qp_2, and the solid ellipse qp_3 represent different query points. Among the data points represented by the hollow dots, the data point dp_1 is far from other data points, and the data point dp_2 is close to many data points. For many real-world applications, query points are close to dense data point area, as shown in figure 1, where qp_1, qp_2 and qp_3 are all much closer to dp_2 than to dp_1. For applications of this kind, a data point closer to other data points (such as dp_2) has a higher chance to be one of the K nearest neighbors of multiple query points, compared to dp_1. Based on this observation, we propose to enhance the algorithm in [14] by assigning a weight to each query point which represents how many neighbors it has.

In this paper we will use DS to represent the data set in our approach. DS contains data points in multi-dimensional data space. The size of DS is n and DS has d dimensions: D_1, D_2, ... D_d. Each data point in DS is a d-dimensional vector: $X_i = [x_{i1}, x_{i2}, ..., x_{id}]$ (i=1,2,...,n). The identity of X_i is i. We consider the case that there are multiple query points. Those

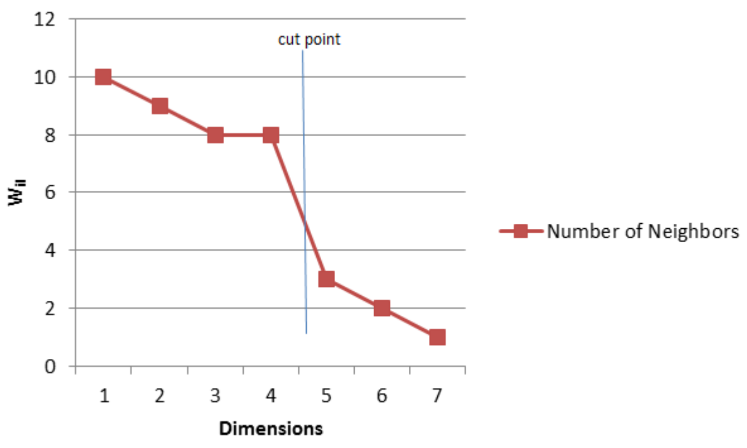


Fig. 2: Dimensions sorted by W_{il}

query points will be in the same data space as data points in DS are: $Q_j = [q_{j1}, q_{j2}, ..., q_{jd}]$. Suppose the set of the query points is Q: $Q = \{Q_1, Q_2, ..., Q_m\}$, with the size of Q being m. Both DS and Q are normalized. Each query point Q_j has a weight WQ_j that represents the importance of the distance to Q_j.

As we discussed previously, we analyze the data distribution and assign a weight W_i to each data point X_i in DS, i=1,2,...,n. The value of weight shows the density of the area X_i is in. The more neighbors X_i has, higher value W_i should have.

We first calculate the weight of X_i on each dimension. For dimension D_l, l=1,2,...,d, we calculate

$$W_{il} = |\{1 \le j \le n||X_{il} - X_{jl}| < \alpha, i \ne j\}| \quad (1)$$

where α is a distance threshold that determines if two data points are close to each other. If X_i is close to many data points on D_l, W_{il} will have a large value; otherwise, W_{il} will have a small value.

Once we have the weight of X_i on each dimension, we can calculate the total weight W_i of X_i in the d-dimensional data space. There are several ways to achieve that. The first solution is to calculate the average of the weights on all the dimensions:

$$W_i = \frac{\sum_{l=1}^{d} W_{il}}{d} \quad (2)$$

The second solution is to select the maximum weight from all the dimensions:

$$W_i = \max_{l=1}^{d} W_{il} \quad (3)$$

Neither of the solutions works well. If we calculate the weight as the average of all the weights, those dimensions on which X_i is not close to many neighbors will weaken the weight of X_i. If we calculate the weight as the maximum value of all the weights, we disregard a lot of useful information from most of the dimensions. To avoid the disadvantages of both solutions, we design the weight W_i of X_i as follows: first we sort the dimensions based on the value of W_{il} in the descending order, shown in figure 2. Next we find the first sharp downward part as the cut point in the second half of the ordered list. All the dimensions before the the cut point are those on which X_i has many neighbors. If there is no sharp point at all in the ordered list, we simply set the cut point as the middle position in the list.

Suppose there are h dimensions before the cut point. We calculate W_i as

$$W_i = \frac{\sum_{p=1}^{h} W_{ip}}{h} \quad (4)$$

W_i is the average of the weights from dimensions on which W_{il} is high. This definition of W_i collects the information of all the dimensions on which X_i has many neighbors.

Thus the final set of the weights for the data points is

$$\mathbf{W} = \{W_1, W_2, ..., W_d\} \quad (5)$$

where W_i i=1,2,...,d is defined in formula (4).

In the next step, we define $\Delta_{ij} = [\delta_{ij1}, \delta_{ij2}, ..., \delta_{ijd}]$ as the array of differences between X_i (i=1,2,...,n) and Q_j (j=1,2,...,m) on all the dimensions (D_1,D_2,...,D_d). δ_{ijl} is calculated as:

$$\delta_{ijl} = WQ_j * |x_{il} - q_{jl}|. \quad (6)$$

IV. Algorithm

In our algorithm, we try to find K nearest neighbors for Q, given a data set DS, a query set Q and the value of K.

We calculate K nearest neighbors for Q on each dimension. The first step is to calculate δ_{ijl} based on equation (6). We next sort the data points based on δ_{ijl} on each dimension D_l, l=1,2,...,d, for each query point Q_j, j=1,2,...,m. We then define KS_{jl} as the set which contains the ids of the first K data points in the sorted list. Let KS'_l = $KS_{1l} \cup KS_{2l} \cup$ ...$\cup$ KS_{ml}. We define KS_l as the set which contains the ids of K data points which appear most frequently in KS'_l.

The next step is to calculate the weight for each data point X_i as we discussed in equation (4). To compute the distance between a data point and the query set, we define

```
Algorithm WMQKNN (DS: data set, Q: query point set, D: dimensions, K: number of data points required, $\omega$: threshold for calculation of query point closeness)
Begin
  For each $X_i \in DS$ and each query point $Q_j \in Q$, calculate $\Delta_{ij} = [\delta_{ij1}, \delta_{ij2}, ..., \delta_{ijd}]$ in which $\delta_{ijl} = |x_{il} - q_{jl}|$;
  Sort the data points in DS based on $\delta_{ijl}$ for each query point $Q_j$ and each dimension $D_l$ ;
  Generate $KS_{jl}$ which contains the first K $ids$ in the sorted list;
  Generate $KS'_l$ as union of $KS_{jl}$ j=1,2,...,m;
  Generate $KS_l$ which contains K $ids$ from $KS'_l$ with the highest frequencies;
  Calculate $W_i$ for each data point $X_i$;
  For each data point $X_i$, generate $B_i = [b_{i1}, b_{i2}, ..., b_{id}]$ in which $b_{il}$ = 1, if $i \in KS_l$; $b_{il}$ = 0, if $i \notin KS_l$;
  Generate $GS$ as union of $KS_l$, l=1, 2, ..., d;
  For each data point $X_i$, where $i \in GS$, calculate $WMSD(X_i, Q)$;
  Sort $GS$ based on $WMSD(X_i, Q)$;
  Let set WMQKS contain the first K $ids \in GS$;
  Return WMQKS.
End.
```

Fig. 3: Proc: Algorithm WMQKNN

a binary array B_i for each data point X_i, i=1, 2, ... n: $B_i = [b_{i1}, b_{i2}, ..., b_{id}]$ in which b_{il} = 1, if $i \in KS_l$; b_{il} = 0, if $i \notin KS_l$. Once we obtain B_i, we calculate the Weighted-multi-query-distance X_i to Q as $WMSD(X_i, Q) = W_i * \frac{\sum_{l=1}^{d} \delta_{il} * b_{il}}{(\sum_{l=1}^{d} b_{il})^2}$, where δ_{il} is the difference between X_i and the average position ($\overline{q_l} = \frac{\sum_{j=1}^{m}(WQ_j * q_{jl})}{\sum_{j=1}^{m}(WQ_j)}$) of Q on D_l, b_{il} is either 1 ($i \in KS_l$) or 0 ($i \notin KS_l$).

From the definition of WMSD(X_i, Q) we can see that only those dimensions on which X_i is close enough to Q are chosen for calculation. We present the WMQKNN algorithm in figure 3.

In this approach, we keep track of the information of all data points and all query points which occupies $O(n+m)$ space. We sort the differences between data points and Q_j on each dimension. The time required is $O(dnmlogn)$.

V. Experiments

In this section we present the experimental results on both synthetic and real data sets, which are run on Intel(R) Pentium(R) 4 with CPU of 3.39GHz and Ram of 0.99 GB.

A. Experiments on synthetic data sets

We design a synthetic data generator to produce data sets with different size and dimensionality. The sizes of the data sets vary from 20,000, 30,000... to 80,000, with the gap of 10,000 between each two adjacent data set sizes. The dimensions of the data sets vary from 10, ... to 80, with the gap of 10 between each two adjacent numbers of dimensions. The value of query set size m varies from 1,2,...,10. The value of K varies from 3,4,...,10.

We conducted various experiments on synthetic data sets. Here we present experiments to demonstrate the performance difference using different way of calculating the weight W_i for each X_i, i=1,2,...,n.

Figure 4 shows the change of algorithm accuracy when the data size increases from 20000 to 80000 using the W_i defined in formula (2). We can see that when there are more data points, more irrelevant information is involved in the calculation.

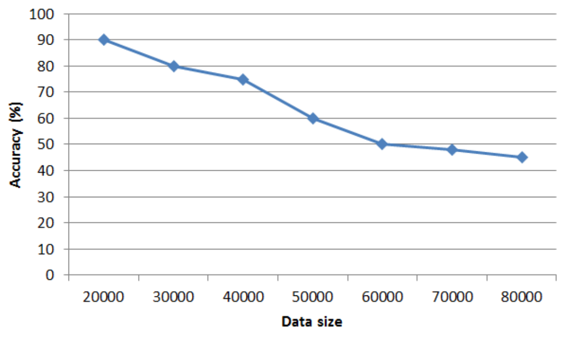


Fig. 4: The change of accuracy when data size increases using formula (2)

Figure 5 shows the change of algorithm accuracy when the data size increases from 20000 to 80000 using the W_i defined in formula (3). The performance is better than the one in figure 4, because we use maximum instead of average. However, we lose a lot of information of most dimensions.

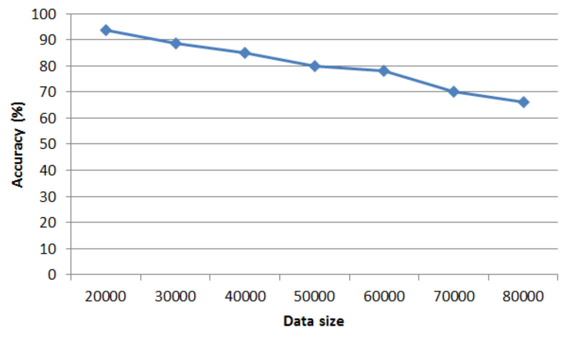


Fig. 5: The change of accuracy when data size increases using formula (3)

Figure 6 shows the change of algorithm accuracy when the data size increases from 20000 to 80000 using the W_i defined in formula (4). The performance is the best compared the previous two, because we only consider the dimensions where the data points have a lot of neighbors.

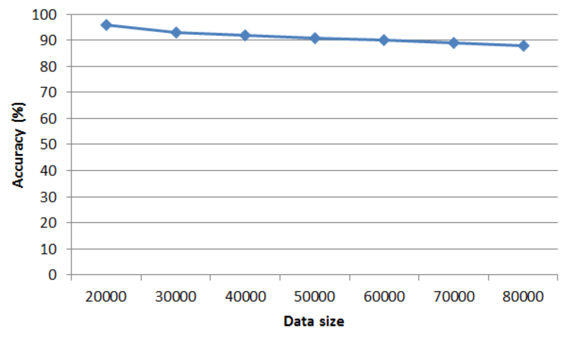


Fig. 6: The change of accuracy when data size increases using formula (4)

B. Experiments on real data set

We next present the experimental results of WMQKNN on real data sets.The real data sets were obtained from UCI Machine Learning Repository [1]. We compare the testing result of these data sets with other algorithms such as IGrid[5] and Frequent K-n-match algorithm [15].

The first data set is Yeast data set. It contains 1484 instances in a 8-dimensional data space. There are 10 clusters in the data set. The second data set is Wine Recognition data set. It contains the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. It contains 178 instances. Each instance has 13 features which means the data set is defined in a 13 dimensional data space. Three clusters are defined with the sizes of 59, 71 and 48. The third data set is Ecoli data set for Protein Localization Sites. There are 336 instances, each of which having 7 features. 8 clusters are contained in the data set.

To generate query points, for each real data set, we randomly select data points as the candidates, and perform our algorithm using K as 6. Query point sets of various sizes are randomly selected, and for each query point, 15 data points are retrieved as the nearest neighbors. If a retrieved data point has the same class with the query point it is associated with, we call it a successful retrieval. Otherwise, we call the data point an unsuccessful retrieval. We calculate how many successful retrievals we have among the results from performing WMQKNN on these query points, and evaluate the accuracy rate. The average accuracy rate of WMQKNN algorithm is 92.6%, which is higher than the accuracy rate of IGrid (87.9%), and that of Freq. K-n-match algorithm, which is 90.8%.

VI. Conclusion and discussion

In this paper, we present our continuous work on finding nearest neighbors for multiple queries. We first analyze the data distribution on each dimension, and calculate the weight of each data point. We discussed various solutions of calculating the total weight of a data point, analyze the disadvantages of two solutions, and choose the one that calculates the average of the weight on selected dimensions. We then apply the weight to calculate the distance between a data point and the query point sets step by step.

We conduct experiments to test our approach on different data sets. We first generate synthetic data sets and demonstrate how the algorithm accuracy changes with the data size using different solutions. We then test our approach on real data sets and compare the experimental results with existing algorithms.

For the future work, we will improve our algorithm by amplifying the effect of data point weight as well as dimension weight. We will modify the way to calculate the distance between a data point and a query point set to improve the performance of our approach.

References

[1] *UCI Machine Learning Archive*. University of California, Irvine, Department of Information and Computer Science. http://kdd.ics.uci.edu. (Retrived: 01/13/2013).

[2] E. Achtert, C. Böhm, P. Kröger, P. Kunath, A. Pryakhin, and M. Renz. Efficient reverse k-nearest neighbor search in arbitrary metric spaces. In *SIGMOD '06*, pages 515–526, New York, NY, USA, 2006. ACM.

[3] C. C. Aggarwal. Towards meaningful high-dimensional nearest neighbor search by human-computer interaction. In *Proceedings of the 18th International Conference on Data Engineering, 26 February - 1 March 2002, San Jose, CA*, pages 593–604. IEEE Computer Society, 2002.

[4] C. C. Aggarwal, A. Hinneburg, and D. A. Keim. On the surprising behavior of distance metrics in high dimensional spaces. In *Proceedings of the 8th International Conference on Database Theory*, ICDT '01, pages 420–434, London, UK, UK, 2001. Springer-Verlag.

[5] C. C. Aggarwal and P. S. Yu. The IGrid index: reversing the dimensionality curse for similarity indexing in high dimensional space. In *Knowledge Discovery and Data Mining*, pages 119–129, 2000.

[6] S. Berchtold, D. A. Keim, and H.-P. Kriegel. The X-tree : An index structure for high-dimensional data. In *VLDB'96*, pages 28–39, Bombay, India, 1996.

[7] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In *International Conference on Database Theory 99*, pages 217–235, Jerusalem, Israel, 1999.

[8] B. Cui, H. T. Shen, J. Shen, and K.-L. Tan. Exploring bit-difference for approximate knn search in high-dimensional databases. In *Proceedings of the 16th Australasian database conference - Volume 39*, ADC '05, pages 165–174, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc.

[9] D. A. White and R. Jain. Similarity Indexing with the SS-tree. In *Proceedings of the 12th Intl. Conf. on Data Engineering*, pages 516–523, New Orleans, Louisiana, February 1996.

[10] R. Fagin, R. Kumar, and D. Sivakumar. Efficient similarity search and classification via rank aggregation. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, SIGMOD '03, pages 301–312, New York, NY, USA, 2003. ACM.

[11] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *The VLDB Journal*, pages 518–529, 1999.

[12] A. Hinneburg, C. C. Aggarwal, and D. A. Keim. What is the nearest neighbor in high dimensional spaces? In *The VLDB Journal*, pages 506–515, 2000.

[13] T. Seidl and K. H.-P. Optimal multi-step k-nearest neighbor search. In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 1998), Seattle, Washington*, pages 154–165, New York,NY,USA, 1998. ACM.

[14] Y. Shi and B. Graham. A similarity search approach to solving the multi-query problems. In *Proceedings of the 2012 IEEE/ACIS 11th International Conference on Computer and Information Science*, ICIS '12, pages 237–242, Washington, DC, USA, 2012. IEEE Computer Society.

[15] A. K. H. Tung, R. Zhang, N. Koudas, and B. C. Ooi. Similarity search: a matching based approach. In *VLDB '06: Proceedings of the 32nd international conference on Very large data bases*, pages 631–642. VLDB Endowment, 2006.

[16] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proc. 24th Int. Conf. Very Large Data Bases, VLDB*, pages 194–205, 24–27 1998.

Parallel Genetic Algorithm Model to Extract Association Rules

Ahmad M. Taleb, Anwar A. Yahya
College of Computer Science and Information Systems
Najran University
Najran, Saudi Arabia
amtaleb@nu.edu.sa
anwaralthari@gmail.com

Nasser M. Taleb
College of Business Administration
Al Ain University of Science and Technology
Al Ain, UAE
nasser.taleb@aau.ac.ae

Abstract— **Over the past generation, the process of discovering interesting association rules in data mining and knowledge discovery has become a cornerstone of contemporary decision support environments. While most of the existing algorithms do indeed focus on discovering high interestingness and accuracy relationships between items in the databases, they tend to have limited scalability and performance. In this paper, we discuss a Parallel Genetic Algorithm Model (PGAM) that has been designed as a scalable and high performance association rules engine. Experimental results demonstrate that the model offers the potential to optimize both scalability and performance in association rules mining.**

Keywords - Data mining; FP-Growth; Multi-objective evolutionary algorithms; scalability; performance; Parallel GA.

I. INTRODUCTION

Data mining and knowledge discovery in databases have been popular targets for researchers over the past 15-20 years, with papers published on a wide variety of related topics. One of the most important tasks in the data mining domain is the association rule mining that aims to find the relationships between the items that frequently appear in the databases' transactions, and additionally, to extract rules of the form IF Condition THEN Predication. The IF clause is called the rule condition that checks if the values of some attributes are true and the THEN clause is called the rule prediction that predicts a value for some goal attribute. In general terms, an association rule is a relation between attributes of the form if X then Y, Where $X \cap Y = \Phi$. It is known that the association rule mining is NP-hard problem because the search space is exponential with the number of itemset.

In 1993, the association rule problem was first introduced by Agrawal et al. [1]. They developed the Apriori algorithm which is the most famous algorithm to solve the association rule problem [2]. This algorithm is based on the support (frequency of the rule in the transactions) and confidence (truth of the rule in the transactions) of the rule. Most of the existing association rules algorithms built upon Apriori-based algorithm, as the Apriori algorithm was well understood and very famous. On the positive side, the improvements of the Apriori algorithm were very impressive in terms of measuring the quality of the generated rules by using a coherent set of multiple measures such as interestingness, comprehensibility, confidence, etc. Unfortunately, such algorithms still make the problem more complex and often provided limited scalability as they are ill-suited to handle massive databases with huge number of attributes and a lot of distinct values for each attribute.

Other approaches are based on Frequent Pattern Growth (FP-Growth) Algorithm [13] to extract association rules. The FP-growth is used to extract the frequent itmesets from the databases in two steps as follows: 1) Build a compact data structure called FP-tree using two passes over the databases and 2) Extract frequent itemsets from the FP-tree by the traversal through the FP tree. After detecting the frequent itemsets, then we can use a user defined parameter called confidence to generate the appropriate association rules. While the FP-Growth algorithm [13] needs only two passes over the datasets, a large amount of memory space is needed because the algorithm generate conditional FP-trees recursively.

For this reason, several parallel algorithms have been proposed in the literature to handle the association rule problem in massive data stores [18]. Scalability on these parallel algorithms was/is indeed acceptable as the partition of large databases and transactions often handles massive data stores. Of course, everything comes at a price and, in the case of parallel algorithms; runtime performance remains a big concern. Specifically, such algorithms often provided poor runtime performance due to the high synchronization and communication overhead and disk I/O cost.

The current paper discusses a parallel genetic-based algorithm to discover association rules called PGAM. The motivation of our design is to provide a high runtime performance and scalable engine for the association rule mining problem. Physically, the architecture is constructed as a federation of largely independent sibling servers, each responsible for a segment of the original transactions. Locally, each server stores, and processes its transactions to extract the association rules using the genetic algorithm that is very well suited to perform global search with less time complexity compared to other algorithms used in data mining problems. A parallel service layer transparently provides global merging and communication services as required. Specifically, the Parallel Genetic Algorithm Model (PGAM) described in this paper is capable of efficiently solving the association rule problems. Experimental evaluation demonstrates that the combination of a shared nothing architecture and heavily optimized local genetic algorithm processing may indeed provide the cost effective scalability/performance pairing that is missing in existing association rule algorithms.

The paper is organized as follows. In Section 2, we briefly review recent work in the area. Basic preliminary material is then outlined in Section 3. We present the details of the parallel model in Section 4, including the genetic algorithm that ties the model together. In Section 5, we discuss the shared nothing architecture and show how the local servers are integrated into a single logical system. Experimental results are then provided in Section 6, with final conclusions in Section 7.

II. RELATED WORK

Association rule mining (ARM) is an important core data mining technique to discover patterns/rules among items in a large database of variable-length transactions. The goal of ARM is to identify groups of items that most often occur together. It is widely used in market-basket transaction data analysis, graph mining applications like substructure discovery in chemical compounds, pattern finding in web browsing, word occurrence analysis in text documents, and so on. Contemporary association rule mining (ARM) research began with the definition introduced by Agrawal et al. [1], an important data mining technique to discover rules among items in massive databases of large number of transactions. Moreover, Agrawal et al. developed the Apriori algorithm to solve the association rule mining problem. This algorithm focuses on the frequent itemsets generation sub-problem and subsequently the generation of the rules with minimum confidence. Subsequently, a number of researchers presented algorithms that are improvements to Apriori algorithm [10], [21]. FP-growth is another famous technique to extract the frequent itemset using minimum support and confidence [13]. More recent work in this area has tended to focus on the quality of the generated rule by considering more measures (mutli-objective algorithms) [10]. In general, scalability and performance were not addressed in the well known Apriori and FP-growth algorithms and their improvements.

Apart from the Apriori and FP-growth algorithms, a significant number of publications focused on generating the association rules using genetic algorithm [5]. Genetic algorithm was first developed by John Holland in 1975. It is based on the idea of survival of the fittest and the greedy approach and performs very well global search with less time. The GA works as follows:

1. An initial population is created. A Population is a group of individuals (Chromosomes) and represents a candidate solution. A Chromosome is a string of genes.
2. Select chromosomes with higher fitness.
3. Crossover between the selected chromosomes to produce new offspring with better higher fitness
4. Mutate the new chromosomes if needed.
5. Terminate when an optimum solution is found.

Ghosh et al. [11] proposed an algorithm to extract frequent itemsets using genetic algorithms. Dou [7] also developed an algorithm to find the maximal frequent itemsets using GA and some defined parameters such as individual identity, individual fitness, upgrade index and upgrade genes that are used in GA. The authors in [15] developed an algorithm for extracting the association rules using GA and without the specification of the user-defined minimum support and confidence. Finally, Hong [14] developed a two-phases GA algorithms to extract the association rules.

With respect to parallel algorithm for the ARM, a number of algorithms were developed based on the parallelization of the Apriori and FP-growth algorithms [3], [12], [16], [19]. Each attempted to effectively exploit the parallel hardware and architecture. Especially, the set of transactions (Databases) is partitioned into a number of subgroups and attempts to utilize the resources of the parallel system efficiently. In other words, each partition is assigned to an independent processor that makes the decision to process and terminate the algorithm. A few other parallel algorithms should be mentioned here. The Hori-Vertical algorithm [18] is a parallel algorithm where no node will be idle because a lot of new independent tasks that can be taken to process. The author in [18] proposed a new database partitioning that is based on dividing the database vertically and horizontally into equivalent parts. Eclat[20] makes vertical database partitioning and is another parallel algorithm to solve the ARM. Limine et al. [4] proposed the "Workload Management Distributed Frequent itemsets mining" (WMDF) algorithm that is based on the horizontal database partitioning and it makes load balancing between system nodes. Parallelizations of the well-known sequential algorithms are discussed with many other parallel algorithms surveyed in [20].

III. PRELIMINARY MATERIALS

We can formally state the task of mining association rules over market basket as follows: let I={I1, I2, …, In} be the set of items/products and T={T1, T2, …, Tn} be the set of transactions in the database. Each of the transaction Ti has a unique ID and contains a subset of the items in I, called itemset. An association rule is an implication among itemsets of the form, $X \rightarrow Y$, where $X \cup Y \subseteq I$ and $X \cap Y = \emptyset$ [1][2]. An itemset can be a single item (e.g. mineral water) or a set of items (e.g. sugar, milk, red tea). The quality of the association rules can be measured by using two important basic measures, support(S) and confidence(C) [17]. Support(S) of an association rule is the percentage of transactions in the database that contain the itemset $X \cup Y$. Confidence (C) of an association rule is the percentage/fraction of the number of transactions that contain $X \cup Y$ to the total number of records that contain X. Confidence factor of $X \rightarrow Y$ can be defined as:

$$\mathbf{Conf}(X \rightarrow Y) = \mathbf{Support}\,(X \cup Y) / \mathbf{Support}\,(X) \qquad (1)$$

Most of the association rule algorithms generate the frequent itemsets –itemsets that are greater than a minimum support—and then generate association rules that have confidence greater than minimum confidence. However, more metrics such as Comprehensibility and Interestingness can be used to have more interesting association rules [10]. Comprehensibility is measured by the number of attributes involved in IF part (condition) of the rule with respect to the THEN (prediction) part of the rule because the rule is more comprehensible if the conditions is less than the prediction. Comprehensibility of an association rule ($X \rightarrow Y$) is measured as:

$$\textbf{Comp } (X \rightarrow Y) = \log(1+|Y|)+\log(1+|XUY|) \qquad \textbf{(2)}$$

where $|Y|$ and $|XUY|$ are the number of attributes in the consequent side and the total rule, respectively.

Interestingness measures how much interesting the rule is [10]. The interestingness of an association rule ($X \rightarrow Y$) is measured as:

$$\textbf{Inter}(X \rightarrow Y) = [\text{Support }(XUY)/\text{Support}(X)] \times [\text{Support }(XUY)/\text{Support}(Y) \times [1 - \text{Support}(XUY)/|D|] \qquad \textbf{(3)}$$

Where $|D|$ is the total number of records/transactions in the database. Using several measures, the association rule problem can be considered as a multi-objective problem.

Fig. 1(a) and (b) depict a small grocery sales database consisting of five products I = {M, R, S, T, W} and six transactions table that illustrates the purchase of items by customers.

Products/items	Abbreviation
Milk	M
Rice	R
Sugar	S
Tea	T
Water	W

(a)

Transaction	Items/products
1	MRTW
2	RSW
3	MRTW
4	MRSW
5	MRSTW
6	RST

(b)

Figure 1. (a) products/items database (b) Database transactions.

Fig. 2(a) shows all frequent itemsets containing at least three products and minimum support 50%. Fig. 2(b) illustrates sample association rules with four metrics (Support, Confidence, Comprehensibility and Interestingness).

In this paper, we choose to deal with the association rule mining as a multi-objective problem rather than one objective problem and to adopt the multi-objective evolutionary algorithm for mining association rules [8], [9] with emphasize on genetic algorithms. Genetic algorithm is an iterative procedure that is appropriate for situations such

Itemsets	Support
R	100%
W, RW	83%
M, S, T, MR, RS, RT, MRW	67%
MT, SW, TW, MRT, MTW, RSW, RTW, MRTW	50%

(a)

Association rules	Support	Confi-dence	Comprehen-sibility	Interest -ingness
$M \rightarrow R$	67%	1	2.58	0.59
$MR \rightarrow T$	50%	0.75	3	0.51
$W \rightarrow RS$	50%	0.75	3	0.51
$R \rightarrow W$	83%	0.83	2.58	0.71

(b)

Figure 2. (a) Frequent Itemsets with minimum support 50% (b) Sample association rules with three metrics

as large and complex search space and optimization problems. In order to use the genetic algorithm, the following points must be addressed [5]:

1. Encoding/decoding schemes of chromosomes (bit string, real-value string, etc.)
2. Population size: how many chromosomes are in population. Good population size is about 30-40.
3. Fitness value: the chromosomes must be ranked according to their fitness values (e.g. support, confidence, comprehensibility, etc. measures can be used to calculate the fitness value of an association rule).
4. Selection: select the chromosomes for next generation by using one of the selection scheme (Roulette wheel, Boltzman, Tournament , Rank, etc.).Note that it is important to use the elitism technique to make sure that the best chromosomes (association rules) that were generated at some intermediate generations will be kept as candidate solutions.
5. Crossover: single point crossover, two point crossover, multi-point crossover, uniform crossover, and arithmetic crossover.
6. Mutation: This to change the new chromosome (offspring) to prevent the algorithm from getting stuck. For binary encoding, the algorithm changes bits from 1 to 0 or vice versa.

IV. MINING ASSOCIATION RULES

The association rule engine described in this paper is a fully parallelized model. That being said, it is physically constructed as series of backend servers, each operates independently to extract the association rules from the database transactions that are housed in each of the nodes. This federated approach allows us to design and build a parallel association rule model by concentrating on the optimization of the single node servers, rather than complex load redistribution policies. The end result is a parallel association rule engine that can directly exploit techniques developed for single node servers. In other words, the current engine can run on a single node, providing good performance that one would expect from a fully optimized association rules mining system. We therefore begin our discussion of the current model by looking at the partitioning of the transactions in the database and the execution model of the sibling (backend) servers. In Section 5, we will return to the question of how to efficiently integrate the individual nodes.

A. Transaction Database Partitioning

We start by looking at the partitioning of the transaction databases (e.g. market-basket problem) across all p backend nodes. Given that our aim is to balance the processing times for mining association rules across all p processors, a good partitioning mechanism is a necessity. We note that our focus in this paper is the database (set of transactions) of the market-basket problem with fix positions of products/items. The set of transactions are stored as bit strings where each string represents a transaction. Table 1 illustrates how the database (set of transactions) looks like.

Table I. Set of transactions (database of market-basket)

		Products/Items								
Transactions		A	B	C	D	E	F	G	H	…
	T1	0	1	1	0	1	0	0	0	…
	T2	1	1	1	0	1	0	0	1	…
	T3	1	0	0	0	1	0	1	0	…
	…									

The stripping technique is described below.

1. **Sort** the original transactions according to the number of items per transaction.

2. **Stripe** the transactions across all processors in a round robin fashion such that successive transactions are sent to the next processor in the sequence. For a network with *p* processors, a database of *n* transactions and *n* mod *p* != 0, a subset of processors receives one additional transaction.

The main goal behind this striping technique is that it dramatically increases the likelihood that the execution time required to extract the association rules will be proportionally distributed across the processors in the multi-computer architecture.

B. Multi-objective Genetic Algorithm

Recall that each backend node operates independently to extract the association rules from the transactions that are housed in each of the backend nodes. Specifically, each backend node executes independently an optimized multi-objective genetic algorithm to extract the association rules for its own transactions database. In our current work, we tried to solve the multi-objective association rule problem with the pareto based [22] genetic algorithm because it is always difficult to find out a single solution for multi-objective problem. Vilferdo Pareto suggested the non-dominance approach to solve multi-objective problems. His approach says " A solution, say a, is said to be dominated by another solution, say b, if and only if the solution b is better or equal with respect to all the corresponding objectives of the solution a, and b is strictly better in at least one objective". We start by discussing the characteristics of the pareto genetic algorithm used in to extract the multi-objective association rules.

The first task in the genetic algorithm is to define what the chromosomes represent (e.g. association rules, frequent itemsets and how (e.g. encoding/decoding). Since we decided to use the Pareto based genetic algorithm in ARM, then the chromosomes will be representing the possible association rules. Two famous approaches (Pittsburg or Michigan) can be used to encode the chromosomes [6]. Pittsburg is very suitable for classification rule mining, while Michigan is more suitable for association rule mining and encodes each part (antecedent and consequent) of the rule separately. More interestingly, Ghosh et al. [10] developed a better scheme for encoding/decoding the rules to/from binary chromosomes. According to Gosh [10], each item or product in the association rule is represented in two bits. If these two bits are 00 the product/item (No need to store the product/item value because the positions of values are fixed) value according to its position appears in the antecedent and if it is 11 then the value of the product appears in the consequent. The other two combinations, 01 and 10 indicate that the absence of the product's value in the rule. In our work, a chromosome represents a possible rule and is represented in binary format as follow: Given six products (ABCDEF), the rule AC → BE will look like 00 11 00 01 11 00. Note that we need k extra bits, where k is the number of items in the database. Note that chromosome data represents a possible association rule that consists whether or not a product/item exists in the association rule (no need to store the actual value of product/item because the positions of products are fixed)**.**

The fitness value for each chromosome is calculated by using a set of three complementary metrics, (1) Confidence (2) Comprehensibility and (3) Interestingness, to filter out the interesting rules. More specifically, an objective fitness function combines these metrics to calculate the fitness value of the chromosomes (possible rules) as the arithmetic weighted average confidence, comprehensibility and interestingness. The fitness function f(x) is defined as follow:

$$\mathbf{f(x)= (W1 * Confidence + W2 * Comprehensibility + W3*Interestingness) /W1+W2+W3 \quad (4)}$$

Where W1, W2, W3 are user defined weights each of the metric and W1+W2+W3 = 100. The weights of the metrics, used to calculate the fitness value, are defined by the user defined parameters (W1, W2 and W3). In other words, the user defined parameters are chosen according to the user's interestingness for each one of the metrics.

As mentioned in equations 1 and 3 that the support of the antecedent part, consequent part and the rule are essential in order to calculate the rule's metrics, as well as, the rule's fitness value. For this reason, we adopted the FP-tree structure [13] to compress a larger database and to avoid the extensively scanning of the raw data set on disk multiple times, as might be done with a naive implementation. If two transactions share a common prefix, then the shared parts will be merged as long as the count is updated properly. There is a better chance that more prefix strings can be shared because the FP-Tree is created by ordering the items by their decreasing support. In our work, the FP-Tree is called SupTree that consists of set of nodes, each of which is defined as **N(value, counter, parentNode, childNode).** A node contains an item value, counter, pointer to the parent node and pointers to its children.

The FP-tree is constructed in two passes over the data-set. In the first pass, scan the data-set and find the support for each item and then sort the items in decreasing order based on their support. In the second pass, Algorithm 1 illustrates the construction of the in-memory tree (SupTree).

The FP-tree (SupTree) usually has a smaller size than the uncompressed data because typically many transactions share items and prefixes and it can fit in the main memory. Moreover, the order of the items by decreasing support minimizes the size of the FP-tree. However, if every transaction has a unique set of values, then the size of the tree is at least as the original database and even higher because of the need to store the pointers between the nodes and the counters.

```
Algorithm 1 (SupTree construction)
Input: Set of Transaction Table with fix positions of values
Output: FP-Tree Structure called SupTree
1: An array A of size n, where n is the total number of items
in the dataset, is created to store the items in decreasing
order with their support. E.g. A[0].item contains the item
with the highest support (A[0].support).
2: Create the root of the SupTree and label it as null.
3: For each transaction in the database [t|T] wher the is first
value and T is the remaining list,
        3.1: Call Insert_Tree ([t|T], SupTree)
4: Function Insert_Tree([t|T], SupTree)
        4.1: If SupTree.root has a child N and N.value =
        t.value Then
                4.1.1: increment the counter of N by 1,and
update Aelse
                4.1.2: create a new node N(value, counter,
parentNode,
                        childNode) and do the following:
                        N.Value = t.value, N.counter = 1;
                        N.parentNode = SupTree.root.childNode;
                        update array A (the parent node of N is
                        linked to SupTree)
        4.2: If T is not empty
                        4.2.1: Call Insert_Tree(T, N)
```

For example, Table 2 shows a sample data-set where 18 transactions and 6 items are exist. First, items in the transactions are sorted in decreasing order by their support. For our example, the order is **(f; a; c; b; d; e)** as shown in Table 3. Fig. 3 illustrates the FP-tree (SupTree) corresponding to the data of Table 2.

Table II. Sample data-set (18 transactions with 6 items)

TID	Items	TID	Items	TID	Items
T1	B, C, D, F	T7	B, D	T13	A, B,C, F
T2	A,D ,F, E	T8	A, C, F	T14	A, B, C, D, F
T3	A, B, C, F	T9	F	T15	A, B, C, D, E
T4	A, C	T10	E, F	T16	A,F
T5	B, F	T11	A, B, C, F	T17	A,D,F
T6	B, C, D	T12	C, F	T18	A,F

Table III. Support for each item

F	A	C	B	D	E
13	11	10	9	7	3

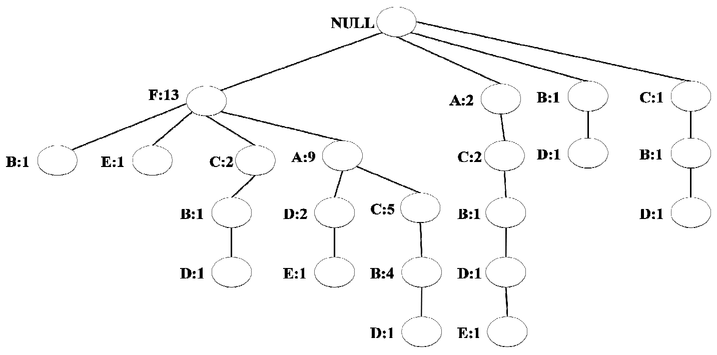


Figure 3. FP-Tree (SupTree) after reading the transactions

In our paper, the FP-tree and the in-memory array (A) will be used to calculate the support of any combinations of items/values without scanning the raw data set multiple times. Recall that each chromosome represents a possible rule and the position of values/items are fixed so that they are not mentioned within the chromosomes. Algorithm 2 shows how the FP-Tree (SupTree) and array (A) are used in very efficient way to calculate the support for any combination of items/values/attributes. In short, the combination of items will be ordered by using the same order of items in array A. We use a top-down process to find the support of the combination. The idea in Algorithm A is to eliminate all sub-trees that will not contribute in the support of the combination (search only sub-trees that may contain the combination of items). For example, given a combination ABC, the algorithm will search only sub-trees rooted at F and A first, then C then B and discard other sub-trees.

After representing the chromosomes and the fitness values, various genetic operators (selection, crossover, mutation, etc.) can be applied to them. Equations 1, 2, 3 and 4 with the FP-tree structure can be used in order to rank the chromosomes according to their fitness values. After ranking the chromosomes, selection operator is used to select the best chromosomes to be in the next population. There are many selection techniques, but, in our paper the chromosomes are selected, for next generation, by the roulette wheel selection scheme. In case of ARM, we need to store the best rules found from the database. However, if we follow the standard genetic operators (selection, crossover, mutation) only, then the final population may not contain better rules that were generated at some intermediates generations. For this reason, we use the elitism technique to replicate the chromosomes ranked as 1 into the next population. If better chromosomes are generated by the standard genetic algorithm operations (selection, crossover and mutation), then replace the dominated chromosomes by the new generated one.

Algorithm 2 Support Calculation

```
Input: FP-Tree (SupTree) and array A and a list of items W
Output: Support of W
1: Order the items of W to be in the same order of items in array A.
2: Call function Find-Support(SupTree, W)
3: Function Find-Support (SupTree, W)
    3.1: set Pos = A[W[0]].postion; Pos is the position of W[0] in Array A (e.g. W[0] = A and Pos = 1)
    3.2 For i = 0 and i<= Pos do
        For each child T in SupTree.children()
            If (T.value() == A[i])
                If(A[i] == W[0])
                    If(W.size() == 1) then
                        return T.counter;
                    Else W=W[1…n]
                        call function Find-Support(T, W)
                    endif
                Else
                    Call function Find-Support (T, W)
                Endif
        endfor endfor
```

Due to the large number of items/products in the market-basket problem, thereby multi-point crossover operator is needed. In short, random positions (crossover points) in the strings (chromosomes) will be chosen and all bits before the first point will be copied from the first parent and all bits after that point and before the next point will be copied from the second parent, and so on until all crossover points are covered. After the crossover is performed, mutation takes place to prevent falling of all solutions in population into a local optimum of the problem. For binary encoding, the mutation procedure changes few randomly chosen bits from 1 to 0 and vice versa. The mutation usually occurs with a very low probability.

V. PARALLEL GENETIC ALGORITHM MODEL (PGAM)

The model in this paper has been designed as a parallel model to extract association rules. The data is partitioned and distributed to all nodes that are operated independently. The database partition is considered as a preprocessing step. In terms of the parallel model, it can be described at a high level as follows. The frontend node serves as an access point for all user defined parameters (mutation probabilities, number of association rules required). Parameters reception and management is performed at this point. The frontend distributes the required parameters to all backend nodes, collect final results from all backend servers, and prepare the final result as per the user requirements. In turn, the backend nodes are fully responsible for extracting the association rules form their local data. In addition, each node houses a *Parallel Service Interface* (PSI) component that allows it to identify its neighborhoods and when and how have to communicate with them. Fig. 4 illustrates the primary components, including the linkage between the sibling servers that are designed according to the ring topology.

With respect to the PSI, we choose to use the open source OpenMPI communication libraries because MPI minimizes the complexity of data transmission and communication within the parallel server. Therefore, utilizing the MPI libraries, the server can be constructed as a single MPI-based application. Specifically, the parallel server consists of a set of nodes (e.g. frontend and backend) that are executed simultaneously and subsequently communicate to each other. Standard precise and reliable operations (send, receive, gather, scatter, broadcast) can then be executed.

As mentioned above, the original set of transactions (datasets) is partitioned and distributed to each one of the backend nodes in round robin fashion. Once the original data (available in the frontend node) is distributed and received by the backend servers, the frontend node broadcasts the user parameters (number of association rules required, number of attributes in the antecedent, number of generations, etc.) and any pre-defined values (crossover and mutation probabilities, size of population, etc.) to all backend nodes. Because of the distribution technique of the original data and replication of the parameters on each of the backend nodes, our parallel mode is said to be load balanced parallel model for mining association rules. At this stage, we are ready to extract the association rules. For this, of course, we require the genetic algorithm along with the FP-Tree services described in Section 4. Algorithm 3 provides a high level description of mining association rules on the backend server instances. In short, identical parameters are sent to each node, where the local server uses these parameters to extract association rules on its local set of transactions. After the execution of all functions and before sending the local results to the frontend, a Parallel Fitness Calculation is performed across the parallel machine. The PSI provides this functionality. Specifically, each node P send the final population R to all other nodes (ring topology) in order to calculate the fitness values of the chromosomes with respect to all transactions found in the original dataset. For example, if P is the current node and N backend nodes exist, then send R to (P+1)%N, (P+2)%N, … , (P+N-1)%N. At the end of this step, each node contains a local population with respect to the local data but the fitness values of the local chromosomes are according to all transactions found in all nodes. Finally, the local population results are returned to the frontend buffers where necessary processing takes place such as merging and ranking of all chromosomes and then return the appropriate association rules as per the user parameters.

following steps then remove the dominated chromosomes from this population.
9: Using the roulette wheel scheme along with the fitness values, select the chromosomes for next generation and replace the chromosomes of old population.
10: Perform multi-point crossover and mutation on these new chromosomes.
11: if the required number of generations is not completed, then go to Step 4

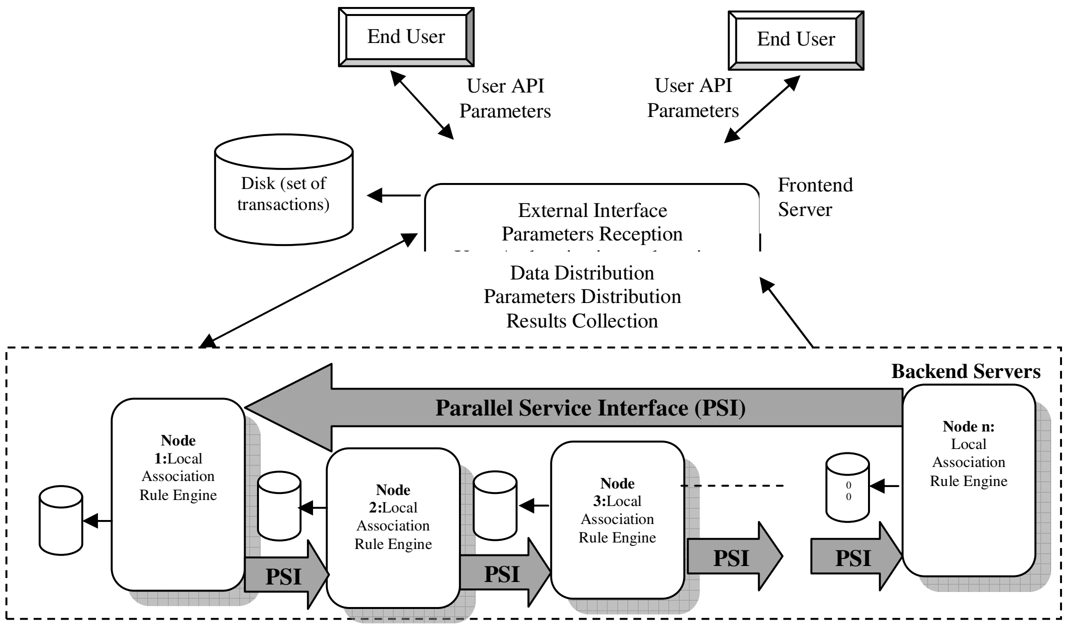

End User
End User
User API Parameters
User API Parameters
Disk (set of transactions)
External Interface
Parameters Reception
Frontend Server
Data Distribution
Parameters Distribution
Results Collection
Backend Servers
Parallel Service Interface (PSI)
Node 1:Local Association Rule Engine
Node 2:Local Association Rule Engine
Node 3:Local Association Rule Engine
Node n: Local Association Rule Engine
PSI
PSI
PSI
PSI

Figure 4: The core architecture of the parallel Association rule engine.

Algorithm 3 Backend Association Rules Engine

Input: A set of parameters received from the frontend and the local set of transactions.
Output: Population with a set of possible association rules sent to the frontend node.

```
1: Receive the user's parameters (uP) and any required pre-defined parameters from the front end (vP)
2: Load the local transaction and create the FP-Tree (SupTree) and array A by calling Algorithm 1.
3: Generate N chromosomes randomly; each chromosome represents a possible rule
4: Decode the chromosomes to get the values of the different items.
5: Using Algorithm 2, find the support of the antecedent side, consequent side and the rule.
6: Using equation 1, 2 and 3, find the confidence, comprehensibility and interestingness
7: Rank the chromosomes by calculating their fitness values (use equation 4).
8: Copy the chromosomes ranked as 1 into a separate population, if better chromosomes are generated from the following steps then remove the dominated chromosomes from this population.
9: Using the roulette wheel scheme along with the fitness values, select the chromosomes for next generation and replace the chromosomes of old population.
10: Perform multi-point crossover and mutation on these new chromosomes.
11: if the required number of generations is not completed, then go to Step 4
12: Do a Parallel Fitness Calculation by sending the final stored population (R) to all other backend nodes in the parallel model (using ring topology design as shown in Fig. 4).
13. Return result R to the frontend (collect R with MPI Allgather()).
```

VI. EXPERIMENTAL RESULTS

In terms of the environment, parallel evaluation was conducted on an 8-node (16 processors), Gigabit Ethernet Linux cluster, with each 2.6 GhZ ProLiant board housing 2 GB of memory. For the test database, we used the Synthetic transactional database generated by IBM Quest Market-Basket Data Generator to synthesize a transaction database. Specifically, we used 1000 unique items to create 10 Million records, each of which has average transaction length of 10. Default values of the genetic parameters are: Population Size = 40, crossover probability = 0.8, mutation probability = 0.02, the values of user-defined weights are chosen to be equal W1=0.33, W2=0.33 and W3=0.33 [11]. In the future, the default values along with the user defined parameters will be changed to evaluate their affects to our algorithm.

We begin by looking at the performance of the proposed parallel genetic algorithm to extract association rules described in this paper. We have directly compared our model with some of the previous parallel association rules algorithms (Elcat, WMDF, HorVertical) using different number of sibling servers (1, 2, 4 and 8 nodes) in the system and minimum support of 0.5%, if needed. Note that the current server has only 8 nodes but our algorithm is designed to work on any parallel server regardless the number of nodes. Fig. 5 shows that the performance of our Parallel Genetic Algorithm Model (PGAM) to extract Association rules does indeed outperform previous parallel association rules algorithms. With respect to our algorithm, the running time consider the time to build and maintain the FP-Tree, communication time, receive results in the front end node and prepare the final list of association rules. This result is due to many reasons. First, our model is based on the Evolutionary Generic Algorithm that is very suitable to such problem (Association rules). Second, finding the support is going to be very fast by adopting the concept of FP-Tree. Third, the encoding/decoding schemes of chromosomes allow us to find the association rules directly without the idea of frequent itemsets. Fourth, the data portioning ensures the load balanced and that all nodes are contributing equally in extracting the association rules. Finally, our approach scans the database only once while the Eclat algorithm scans the database three times and the WMDF algorithm scans the database a lot of Times. Note that HoriVertical scans the database only once but it is not based on evolutionary algorithm.

In production environments, it is quite likely that association rule algorithms will be accessing databases (set of transactions) that are larger than the ones that can be conveniently tested in academic settings. As a result, it is important to provide some understanding of performance as set of transactions (databases) grow. Our scalability assessment begins with a look at performance patterns as the number of transactions increases from 10 million to 40 million records. In this experiment, we use 8 nodes to extract the association rules as the number of transactions vary. Fig. 6 shows the execution time as a function of number of transactions (database size). As can be seen in the figure, the running time is increased by a factor of 1.3 as the number of records in the database increases by a factor of two. The result is expected because the size of the FP-Tree would be almost the same as the number of transactions increases. Consequently, our Parallel Genetic Algorithm Model to extract association rules is very scalable in that an increase in the number of transactions is associated with nearly the same execution time. Note that other algorithms (Elcat, WMDF, HorVertical) focus on the parallel runtime performance to extract association rules therefore we did not compare our algorithm in terms of scalability with their algorithms.

Due to the current capacity of the nodes' memories and processors, we could not make experiments with larger data sets. But in theory the algorithm is designed to support large and big real-world data sets. In the future, we will upgrade the capacity of memory and processor in each one of the nodes to perform experiments with larger datasets.

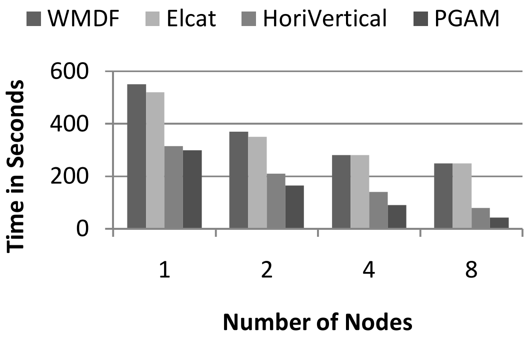


Figure 5. performance of our model (PGAM) versus other parallel algorithms

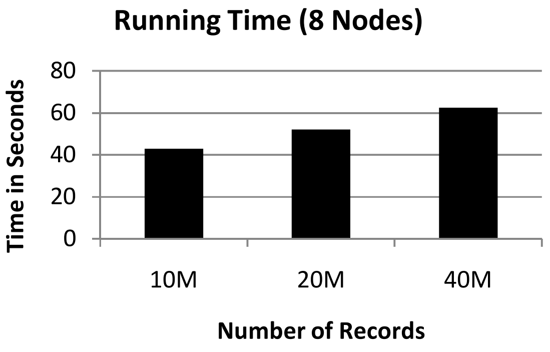


Figure 6. Running time as a function of the number of records

The parallel speedup graph illustrated in Fig. 7 depicts a speedup of approximately 7 (about 88% of optimal). The difference between observed speedup and optimal speedup is due to the Parallel Fitness Calculation used to calculate the fitness values of all chromosomes found in the parallel nodes.

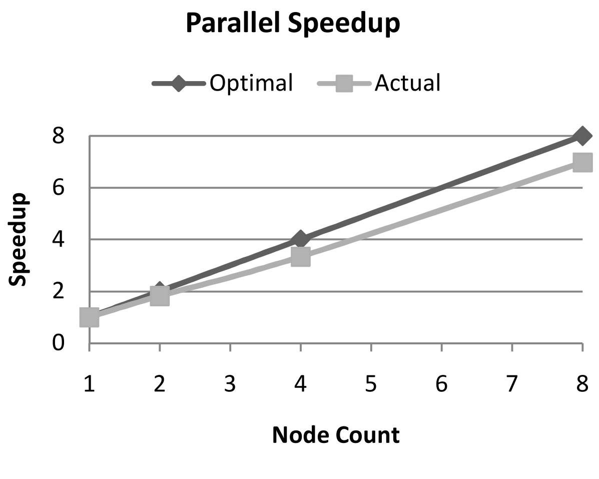


Figure 7. Parallel Speedup

VII. CONCLUSION

A great deal of association rules and frequent itemsets research has been published over the past 15-20 years. For the most part, however, researchers tend to focus on Apriori and FP-Growth algorithms and data structures for single node servers. Given the size of the underlying market-basket databases, coupled with the availability of modestly priced hardware and the advantages of genetic algorithm -- it was proved to perform global search with less time complexity and also very well suited for NP-hard problem such as ARM--, there exists great opportunity for the exploitation of cluster-based data mining servers by using GA. In this paper, we have discussed a Parallel Genetic Algorithm Model (PGAM) for association rules. Constructed as a federation of heavily optimized sibling servers, the current model demonstrates the potential to provide both high performance and scalability. Experimental evaluation in both multi-node scenarios suggests that the current model does indeed have the potential to achieve this objective.

REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules Between Sets of Items in Large Databases," in Proc. of ACM SIGMOID Conf., pp. 207-216, 1993.

[2] R. Agrawal and R. Srikant, "Fast Algorithm for Mining Association Rules," in Proc. of the 20th VLDB Conf., pp. 487-499, 1994.

[3] R. Agrawal and J. Shafer, "Parallel Mining of Association Rules," In IEEE Transactions on Knowledge and Data Engineering, vol. 8, no. 6, pp. 962-969, 1996.

[4] L. Aouad, N. Le-Khac, and T. Kechadi, "Distributed frequent itemsets mining in heterogeneous platforms," Engineering, Computer and Architecture, Volume 1 (2007).

[5] S. Das and B. Saha, "Data Quality Mining using Genetic Algorithm," International Journal of Computer Science and Security, (IJCSS) Volume 3, Issue 2, pp. 105-112.

[6] S. Dehuri, A. Jagadev, A. Ghosh, and R. Mall, "Multi-objective Genetic Algorithm for Association Rule Mining Using a Homogeneous Dedicated Cluster of Workstations," American Journal of Applied Sciences 3 (11): 2086-2095, 2006 ISSN 1546-9239, 2006.

[7] W. Dou, J. Hu, K. Hirasawa, and G. Wu, "Quick Response Data Mining Model Using Genetic Algorithm," SICE Annual Conference, 2008, pp. 1214-1219.

[8] M. Fonesca and J. Fleming, "Multi-objective Optimization and Multiple Constraint Handling with Evolutionary Algorithms," Part I: A Unified Formulation. IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans, 28(1), pp. 26-37, 1998.

[9] A. Freitas, "Survey of Evolutionary Algorithms for Data Mining and Knowledge Discovery," Advances in evolutionary computing: theory and applications, pp 819 – 845, 2003.

[10] A. Ghosh and B. Nath, "Multi−objective rule mining using genetic algorithms," Information Sciences 163, pp 123-133, 2004.

[11] S. Ghosh., S. Biswas., D. Sarkar., and P. Sarkar, "Mining Frequent Itemsets Using Genetic Algorithm," International Journal of Artificial Intelligence & Applications (IJAIA), Vol.1, No.4, October 2010

[12] E. Han, G. Karypis, and V. Kumar, "Scalable Parallel Data Mining for Association Rules," In Proceedings of the ACM SIGMOD International Conference on Management of Data, 1997, pp. 277-288.

[13] J. Han, J. Pei, and Y. Yin, "Mining Frequent patterns without candidate generation," 2000, In Proc. Of ACM-SIGMOD Int. Conf. on Management of Data, pp. 1- 12.

[14] T. Hong, J. Huang., W. Lin, and M. Chiang, "GA-Based Item Partition for Data Mining," 2011 IEEE, pp. 2238-2242.

[15] A. Islam, T. Chung, **"**An Improved Frequent Pattern Tree Based Association Rule Mining Technique," Information Science and Applications (ICISA), 2011 International Conference on 2011.

[16] A. Javed and A. Khokhar, "Frequent Pattern Mining on Message Passing Multiprocessor Systems," In Distributed and Parallel Databases, vol. 16, no. 3, pp. 321-334, 2004.

[17] S. Kotsiantis and D. Kanellopoulos, "Association Rules Mining: A Recent Overview," GETS International Transactions on Computer Science and Engineering, Vol.32(1), 2006, pp.71-82.

[18] H. Marghny and H. Refaat, "Hori-Vertical Distributed Frequent Itemsets Mining Algorithm on Heterogeneous Distributed Shared Memory System," IJCSNS International Journal of Computer Science and Network Security,VOL.10 No.11 , pp. 56–62 , November 2010.

[19] O. Zaïane, M. El-Hajj, and P. Lu, "Fast Parallel Association Rule Mining without Candidacy Generation," In Proceedings of the IEEE International Conference on Data Mining, 2001, pp. 665-668.

[20] M. Zaki, S. Parthasarath, and L. Wei, "A localized algorithm for parallel association mining," In Proceedings of the 9th Annual ACM Symposium on Parallel Algorithms and Architectures, 1997, pp. 321–330.

[21] Q. Zhao and S. Bhowmick, "Association Rule Mining: A Survey," Technical Report, CAIS, Nanyang Technological University, Singapore, No. 2003116 , 2003.

[22] E. Zitzler, K. Deb, and L. Thiel, "An evolutionary Algorithm for Multi-objective Optimization," The Strength Pareto Approach, 1998.

Optimization of SuperSQL Execution by Query Decomposition

Ria Mae Borromeo
Faculty of Information and Communication Studies,
University of the Philippines Open University
Los Baños, Laguna, Philippines
riamae.borromeo@upou.edu.ph

Motomichi Toyama
Department of Information and Computer Science
Faculty of Science and Technology, Keio University
Yokohama, Japan
toyama@ics.keio.ac.jp

***Abstract*— SuperSQL is an extension of SQL that allows formatting and publishing of database contents into various kinds of application data directly as a result of a query. Possible application data output formats include, but are not limited to, HTML, PDF, XML, XLS, and Ajax-driven pages. Originally, the SuperSQL query is directly converted into a single SQL query. In some query cases, this procedure returns a large intermediate table, which typically require a long execution time and consume a lot of memory. To minimize the execution time and memory consumption, query decomposition, the process of dividing a query into sub queries whose result sets' union is equivalent to the result set of the original query, was applied to SuperSQL query processing. Experiments show that for several query cases, there is significant reduction in SuperSQL query execution time and memory consumption.**

Keywords- database applications; database publishing; query processing; query optimization.

I. INTRODUCTION

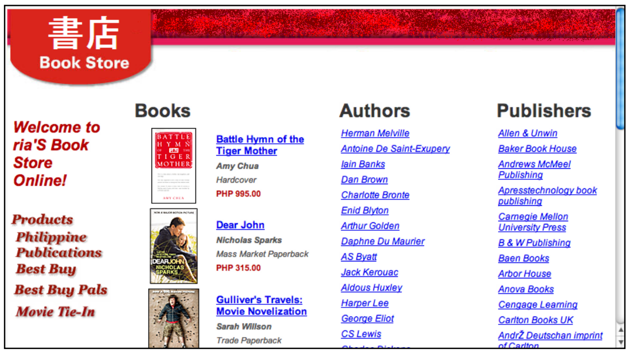
Relational databases are here to stay. Although new database technologies continue to arise and gain popularity, relational databases are far from being obsolete [2]. SQL, the standard query language used for managing and querying relational databases, returns query results to the user in the form of a flat table. To translate this output into a specific application, report writers have been used. However, there is no standard language that covers the specification of such translations into various types of application data [10].

SuperSQL is an extension of SQL that has the capability of generating various kinds of application data directly as a result of a query. Its syntax is similar to SQL with additional formatting capabilities. Currently, it is mainly used to easily create data-driven web pages and applications. Figure 1 is a sample SuperSQL query and Figure 2 is a sample SuperSQL query output. This is a sample page of a bookstore website that lists from left-to-right its records of all books, authors and publishers.

A SuperSQL query is converted into a single SQL query, which is processed by the Database Management System (DBMS). Consequently, one intermediate table is returned and processed by SuperSQL. Depending on the number and size of tables to be accessed, the size of intermediate table can become large even though the actual number of tuples in the desired output is small.

```
GENERATE HTML [
      {"Books"![b.title]!
      {"Authors"![a.name]!},
      {"Publishers"![p.publisher]!}
  ]!
FROM books b, publishers p, authors a
```

Figure 1. Sample SuperSQL Query


書店
Book Store

Welcome to ria'S Book Store Online!

Products
Philippine Publications
Best Buy
Best Buy Pals
Movie Tie-In

Books

Battle Hymn of the Tiger Mother
Amy Chua
Hardcover
PHP 995.00

Dear John
Nicholas Sparks
Mass Market Paperback
PHP 315.00

Gulliver's Travels: Movie Novelization
Sarah Willson
Trade Paperback

Authors

Herman Melville
Antoine De Saint-Exupery
Iain Banks
Dan Brown
Charlotte Bronte
Enid Blyton
Arthur Golden
Daphne Du Maurier
AS Byatt
Jack Kerouac
Aldous Huxley
Harper Lee
George Eliot
CS Lewis

Publishers

Allen & Unwin
Baker Book House
Andrews McMeel Publishing
Apresstechnology book publishing
Carnegie Mellon University Press
B & W Publishing
Baen Books
Arbor House
Anova Books
Cengage Learning
Carlton Books UK
AndrŽ Deutschan imprint of Carlton

Figure 2. Sample SuperSQL Query Output

The SuperSQL query in Figure 1 is converted into the SQL query in Table I a). If the books, authors and publishers tables have 550, 25 and 20 tuples respectively, the resulting SQL query takes the Cartesian product of the three tables. The intermediate table size would have 275,000 tuples. However, in the sample query, there is no relationship between the tables. The desired output only consists of a list of the contents of each table, displayed from left-to-right. Therefore, the desired number of tuples is only 595, which is the sum of the tuples in each table.

Initial experiments using the current SuperSQL version showed that as the intermediate table size increases, the execution time and memory consumption of a SuperSQL query also increase. Thus we aim to reduce the intermediate table returned by the DBMS to reduce execution time and memory consumption of executing SuperSQL queries.

In this study, the concept of query decomposition was applied to SuperSQL queries. Query decomposition is the process of finding several queries wherein the union of the result sets is equivalent to the original result set. Having several queries instead of one eliminates unnecessary Cartesian product and join operations thus reducing the intermediate table size and consequently reducing execution time and memory consumption.

TABLE I. COMPARISON OF RESULTING SQL QUERIES WITH AND WITHOUT QUERY DECOMPOSITION

a) without Decomposition	b) with Decomposition
SELECT DISTINCT b.title, a.name, p.publisher FROM books b, authors a, publishers p	1) SELECT title FROM books; 2) SELECT name FROM authors; 3) SELECT publisher FROM publishers;

If query decomposition is applied to the example, instead of being converted to the SuperSQL query in Table I a), the query is converted into Table I b). The results of the individual queries are combined later on to produce the desired output.

The query decomposition algorithm models the SuperSQL query as an undirected graph where the query's attributes are represented as nodes and the attributes' relationship with each other are represented as edges. The number of connected components in the resulting graph represents the number of possible divisions for the query. The original query is divided into the number of connected components and the resulting queries are executed individually. The results are combined later on to produce the desired output.

II. RELATED WORKS

Query Decomposition is the process of dividing a query into several queries wherein the union of the result sets of the divided queries is equivalent to the result set of the original query. It is widely used in systems wherein a query contains data from different database servers. In such applications, a query is decomposed based on the mapping of data attributes to its sources.

In 2008, Le applied query decomposition to access data from various data repositories [4]. However, since creation of mappings is an expensive process, an input query is automatically decomposed into sub queries without pre-defined mappings. The algorithm traverses from the bottom to the top of a schema tree depending on the structure of local schemas. Compared to top-down approaches, the algorithm can reduce the time for creating the divided queries for local schemas.

Also in 2008, Bonchi applied query decomposition to a document retrieval system [3]. A query is decomposed into a small set of queries whose union of resulting documents corresponds approximately to that of the original query. The goal is to assist users in finding the information they are looking for, by providing them a suitable set of queries as part of the results of their queries.

The problem was instantiated a specific variant of a set cover problem where an efficient greedy algorithm and a clustering algorithm were designed.

In this study, we applied query decomposition to SuperSQL queries. Instead of converting a SuperSQL query into one SQL query, when possible, it is converted into several queries in order to minimize the intermediate table output and reduce SuperSQL execution time and memory consumption.

III. SUPERSQL

SuperSQL extends the functionality of an SQL query by using the Target Form Expression (TFE) processing system. TFE was formerly developed to generate ordinary reports from the contents of relational databases [9]. Currently, it has been extended to generate application data directly from database queries.

A. Syntax

TABLE II. COMPARISON OF SQL AND SUPERSQL SYNTAX

SQL Syntax	SuperSQL Syntax
SELECT <attribute list>	GENERATE <media> <Target Form Expression (TFE)>
FROM <tables>	FROM <tables>
WHERE <condition>	WHERE <condition>

The syntax of SuperSQL is similar to SQL. The major difference is the introduction of the GENERATE keyword, which allows the user to specify the target application data output. Table II shows the syntax comparison of an SQL and a SuperSQL query. Instead of the SELECT keyword, a SuperSQL query starts with the GENERATE keyword. Moreover, the attribute list in SQL is specified in the TFE. As of the latest version of SuperSQL, the following are the possible target media: Actiview [5], Excel, Flash, HTML, HTML5, LaTeX, LDAP, PDF, VRML (X3D) and XML.

B. Target Form Expression

While an ordinary target list in SQL is a comma-separated list of attributes, TFE uses new operators called connectors and repeaters to specify the structure of the document to be generated by the query. For example, in a table element of a webpage, the columns of the table are associated to the first dimension while the rows are associated to the second dimension and the hyperlinks are associated to the third dimension. Binary operators represented by a comma (,), an exclamation point (!) and a percent (%) symbol are used as the connectors of the first three dimensions. They connect objects generated by their operands horizontally, vertically and in the depth direction, respectively [10]. This is illustrated in Figure 3.

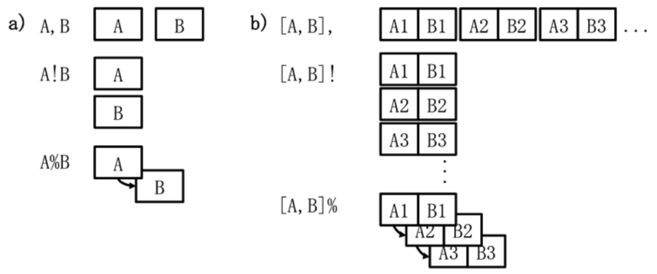


Figure 3. a) Connectors and b) Repeaters

A pair of square brackets ([ ]) followed by any of the above connectors is a repeater for that dimension. It will connect multiple instances in its associated dimension. For example, *[publishers.publisher, books.title, authors.name]!* will connect a publisher, a book title and an author into horizontal direction and connect them vertically as long as there are tuples in the query result.

C. SuperSQL Architecture

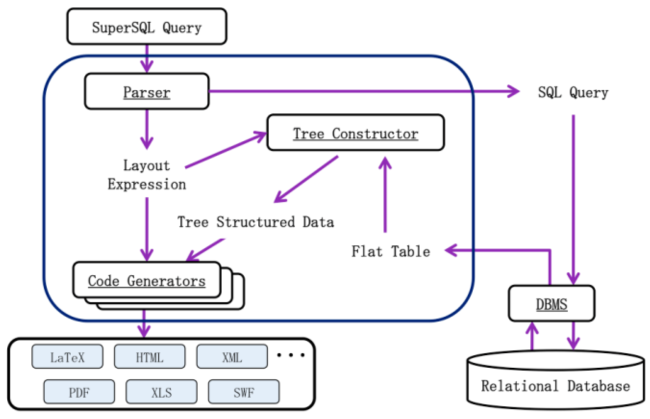


Figure 4. SuperSQL System Architecture

As can be seen in Figure 4, the SuperSQL system has four major components: the Parser, the Tree Constructor, the DBMS and the Code Generators. The Parser is responsible for detecting syntax errors. It extracts the underlying SQL syntax components such as the SELECT, FROM and WHERE clauses. Then an SQL query is created and sent to the DBMS.

The Parser also extracts the layout expression. The layout expression is composed of two parts. The first part is called the schema, which is a tree-structured representation of the layout of the attributes in the query. The second part is called the data formatting, which contains layout information specific to the application data. For example, if the desired application data output is HTML, the data formatting would contain information about HTML formatting such as background color, font size, page title, etc.

The DBMS, which is currently either PostgreSQL or MySQL, executes the SQL query and returns a flat table. The Tree Constructor combines the schema with the flat table containing the SQL query result and outputs a tree-structured data. The Code Generator takes as inputs the data formatting and the tree-structured data and produces the application data output specified in the SuperSQL query.

IV. QUERY OPTIMIZER

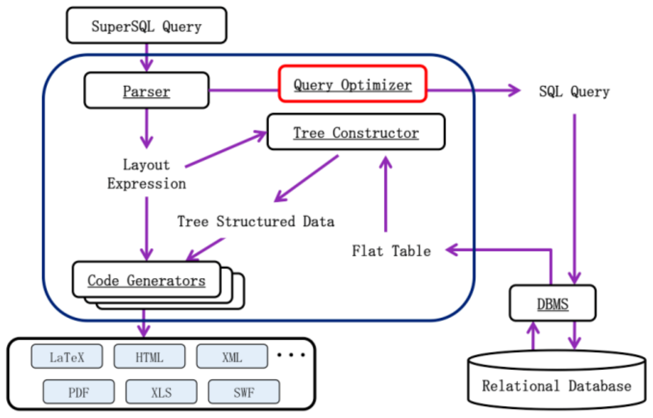


Figure 5. Proposed SuperSQL System Architecture

To implement query decomposition, a Query Optimizer component was added to the system as can be seen in Figure 5. The Query Optimizer is responsible for checking the divisibility of a query and creating single or multiple SQL statements.

The divisibility of a query is determined by modeling the query as an undirected graph. All the attributes found in the schema and the WHERE clause are represented as vertices. The relationships that exist between the attributes are represented as edges.

Edges are added to the graph based on three conditions: first, two attributes are from the same table; second, two attributes are equated in a WHERE condition; and lastly, two attributes are grouped together in the schema.

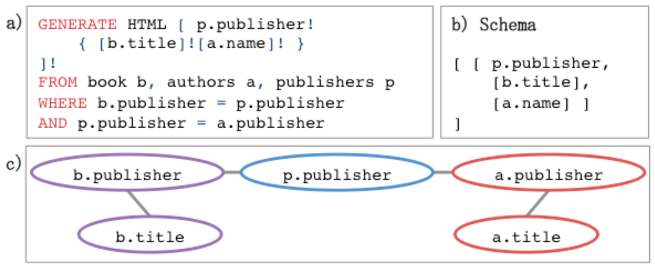

a)
```
GENERATE HTML [ p.publisher!
    { [b.title]![a.name]! }
]!
FROM book b, authors a, publishers p
WHERE b.publisher = p.publisher
AND p.publisher = a.publisher
```
b) Schema
```
[ [ p.publisher,
    [b.title],
    [a.name] ]
]
```

Figure 6. Example: Query graph with Connector Node

In Figure 6 a), we have a SuperSQL query with Figure 6 b) as the schema. From the schema, we added the attributes, *b.title*, *p.publisher* and *a.name* were added as vertices in our graph. From the WHERE clause, *b.publisher* and *a.publisher* were also added as vertices in the graph. Figure 6 c) shows the resulting graph.

Connector nodes are nodes that connect at least two attributes from different tables. A connector node must not be in the same table as any of the attributes it connects. In Figure 6 c), *p.publisher* is a connector node.

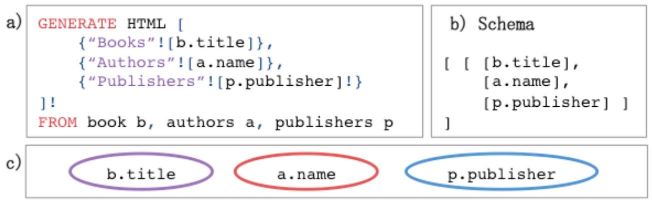
After creating the graph, connected components were identified using Depth-First Search. A connected component is a sub graph that contains a path between all pairs of vertices in the graph. Depth-First Search is a search algorithm that extends the current path as far as possible before backtracking to the last choice point and trying the next alternative path. Each node can only be visited once except for connector nodes, which can be visited as many times as the number of nodes it connects. The Depth-First Search was repeated until all nodes have been visited. The resulting number of paths in the path list is equal to the number of connected components of the graph, which is subsequently equal to the number of possible divisions for the query.

In the example in Figure 6, we get two connected components: *{b.title, b.publisher, p.publisher}* and *{p.publisher, a.publisher, a.name}*. Therefore, the query can be divided into two.

A. *Tree-Structured Data Construction*

The Tree Constructor was modified to handle the results of multiple queries generated by the query optimizer. To create the tree structure, the schema must be combined with SQL query results from different tables. The results of SQL queries are stored in a structure, which can be referenced by the schema. The schema is traversed to create the tree structure. A node in the schema becomes a parent node and the values from the query results are added to the tree structure as its children. Attribute-value pairs are used to ensure the correctness of the parent-children mapping.

B. *Cases Handled*



a)
```
GENERATE HTML [
    {"Books"![b.title]},
    {"Authors"![a.name]},
    {"Publishers"![p.publisher]!}
]!
FROM book b, authors a, publishers p
```

b) Schema
```
[ [ [b.title],
    [a.name],
    [p.publisher] ]
]
```

Figure 7. Example: Trivial Case

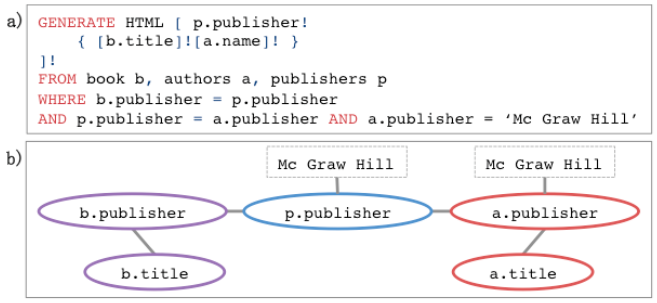
1) *Trivial Case:* Figure 7 illustrates the trivial case. In this example, *p.publisher*, the list of publishers, *b.title*, the list of books and *a.name*, the list of authors, are retrieved. These attributes are from different tables and are not related by any conditions thus they are independent of each other. Originally, the resulting query will generate an intermediate table equal to the Cartesian product of the three tables. However, based on the proposed query decomposition algorithm, the query can be divided into the following sub queries and retrieve smaller intermediate tables.

- *SELECT p.publisher FROM publishers p;*
- *SELECT b.title FROM books b;*
- *SELECT a.name FROM authors a;*

2) *Grouped Columns:* In this case, independent columns are grouped together by a common attribute. This is illustrated in Figure 6. In the given example, *b.title* and *a.name* are grouped together by *p.publisher*. The original query generates an intermediate table which takes the Cartesian product of the *books* and *authors* tables joined with the *publishers* table. However, based on the query decomposition algorithm, this query can be divided into the following sub queries.

- *SELECT b.title, p.publisher*
  FROM books b, publishers
  WHERE b.publisher = p.publisher;

- *SELECT a.name, p.publisher*
  FROM authors a, publishers p
  WHERE a.publisher = p.publisher;



a)
```
GENERATE HTML [ p.publisher!
    { [b.title]![a.name]! }
]!
FROM book b, authors a, publishers p
WHERE b.publisher = p.publisher
AND p.publisher = a.publisher AND a.publisher = 'Mc Graw Hill'
```

Figure 8. Example: Query with String Literal

3) *Queries with String or Literal Conditions:* In Figure 8, *a.publisher* is equated to the string literal, *"McGraw Hill."* Since *a.publisher* is connected to the connector node, *p.publisher*, the string literal, *"McGraw Hill"* is copied to the *p.publisher*. This is done so that when the query is divided, the sub queries will contain the same condition and thus the number of tuples retrieved would be minimized. The resulting queries for this example are:

- *SELECT a.name, p.publisher*
  FROM authors a, publishers p
  WHERE a.publisher = p.publisher
  AND (a.publisher = 'McGraw Hill');

- *SELECT b.title, p.publisher*
  FROM books b, publishers p
  WHERE b.publisher = p.publisher
  AND (p.publisher = 'McGraw Hill');

V. EVALUATION

This study aims to reduce the execution time and memory consumption of SuperSQL queries by reducing the size of the intermediate table returned by the DBMS. This was done by adding a query optimizer, that implements query decomposition, to the SuperSQL system. The resulting SuperSQL system is called the optimizer version.

A. *Input Queries*

To evaluate the effectiveness of the query optimizer, two query cases were used. Query case 1 refers to the query in

Figure 7, which illustrates the trivial case. Query case 2 refers to the query in Figure 6, which illustrates the grouped case. Table III compares the expected number of tuples with and without query decomposition.

TABLE III. COMPARISON OF EXPECTED NUMBER OF TUPLES WITH AND WITHOUT QUERY DECOMPOSITION

Query Case	Expected no. of Tuples w/o Decomposition	Expected no. of Tuples w/ Decomposition
1	|*b.title*| × |*p.publisher*| × |*a.name*|	|*p.publisher*| + |*b.title*| + |*a.name*|
2	|(*b.title* × *a.name*) ⋈ *p.publisher*|	|*p.publisher* ⋈ *b.title*| + |*p.publisher* ⋈ *a.name*|.

Theoretically, the reduction of the intermediate table size results to the following: faster execution of the DBMS, smaller data structure size used to store the intermediate table, and faster selection of tuples to be included in the application data tree structure. In other words, reduction of intermediate table size results to faster execution and less memory consumption, which were proven by the experiments discussed in this section.

B. *Experimental Environment*

The test queries were executed by a machine running Mac OS X version 10.6.8 with 2.3GHz Inter Core i5 processor and 4GB 1333 MHZ DDR3 main memory. The intermediate table size is the independent variable. It is defined as the number of expected tuples without query division. Three intermediate table sizes were used. Small – with tuples ranging from 10 to 100; medium – with tuples ranging from 100 to 1000; and large with tuples ranging from 10000 to 100000. Results were compared to the performance of the SuperSQL src08 package, which was used as the baseline in the experiments.

C. *Data Construction Time*

For the three different size ranges, the data construction time of the optimizer version was compared to the baseline. In the following graphs, the legend Base1 refers to the baseline Query Case 1 (trivial case) and Opt-1 refers to the optimized Query Case 1. Base-2 refers to the baseline Query Case 2 (grouped case) and Opt-2 refers to the optimized Query Case 2.

In the small intermediate table size range, it can be observed in Figure 9 a) that the data construction time of all the cases are almost the same. Since the original query yields only a small intermediate table, its processing time is not significantly different from the total processing time of checking the divisibility of the query, executing and combining results of individual sub queries. Nevertheless, it can be seen that Opt-1 and Opt-2 performed a little faster than their baseline counterparts.

In the medium intermediate table size range, the optimizer version performed significantly faster than the baseline. This can be observed in Figure 9 b). It can also be seen that after processing 800 tuples, a rapid growth was seen in baseline algorithm's data construction time. However, for both trivial and grouped case, the data construction time growth of the optimizer version was linear.

In the large intermediate table size range, the optimizer version also performed significantly faster than the baseline. In Figure 9 c), it can be observed that the growth of the baseline in the grouped case is exponential. The growth of the baseline in the trivial case is also exponential but at a slower level. In the optimizer version, the data construction time of Opt-1 and Opt-2 were almost the same and their growths were both linear.

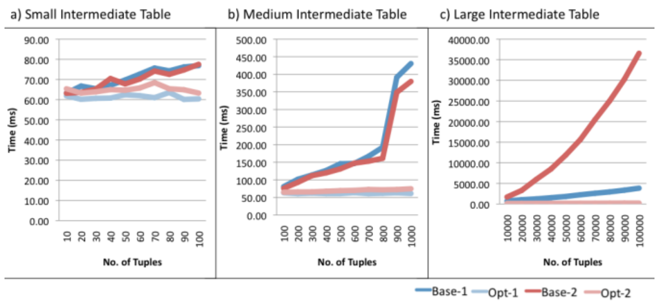


Figure 9. Data Construction Time of Queries

It can be inferred from these results that for the medium and large intermediate table size ranges, the percentage of data construction time reduced is significantly larger than the overhead cost of checking the divisibility of the query and executing and combining results of the sub queries.

D. *Memory Consumption*

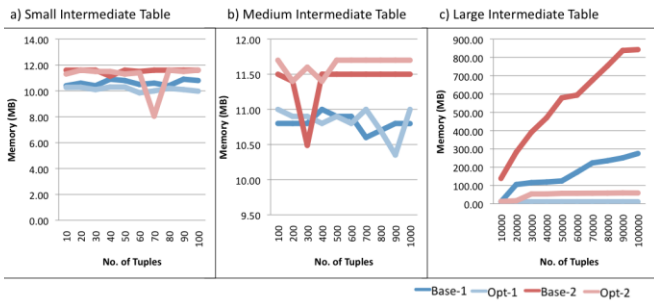


Figure 10. Memory Consumption of Queries

Query Case 1 and Query Case 2 were also used to observe the memory consumption.

For small and medium intermediate table size ranges, the amount of memory used to process both query cases in both the baseline and optimizer version range from 8-12 MB. This can be observed in Figure 10 a) and Figure 10 b). This is because for such intermediate table size ranges, the amount of memory saved from the reduction of the intermediate table size and the overhead memory consumption cost of the graph structure used for checking divisibility are not significantly different.

It can also be observed that the Query Case 2 or the grouped attribute case consumed a little more than the trivial case. This is because the height of its schema tree is higher and thus more memory is needed to represent the tree.

It can be observed in Figure 10 c) that the optimizer version consumes significantly less amount of memory than the baseline for both the trivial case and grouped attribute case. This means that for this size range, the amount of memory saved from the reduction of the intermediate table size is significantly greater than the overhead memory consumption cost of the graph structure used in checking the divisibility of the query.

VI. Conclusions and Future Work

The study proposed a query optimizer for the SuperSQL system based on query decomposition. The main goals were to reduce SuperSQL execution time and memory consumption by reducing the intermediate table size. A SuperSQL query was modeled as a graph wherein vertices are the attributes in the query and edges are the relationships that exist between the attributes. The relationships between the attributes were based on the desired layout of the attributes in the query and the schema, the relational operations in the input query's WHERE clause, and the tables where each attribute belongs. The connected components of the resulting graph were computed by using the Depth-First Search algorithm. The number of connected components is equal to the number of possible divisions for the query. The connected components were converted into SQL queries and executed individually. As a result, for some query cases, the combined size of the intermediate tables of the sub queries was significantly smaller than the size of the intermediate table without query division.

The data construction time and memory consumption of the SuperSQL with the query optimizer were compared to the original SuperSQL version. The comparison was done for three intermediate table size ranges. For queries with intermediate table size of 10 to 100 tuples, the optimizer version did not significantly differ from the original in terms of execution time and memory consumption. For queries with a medium intermediate table size of 100 to 1000 tuples, the optimizer version executed significantly faster, but the memory consumption was not significantly lower. For queries with a large intermediate table size of 10000 to 100000 tuples, the optimizer version executed significantly faster and consumed significantly less memory. Based on these experiments, it can be concluded that the proposed optimizer is effective in reducing SuperSQL execution time and memory consumption for the query cases that it can handle.

The proposed optimizer has already been integrated to the currently working SuperSQL system as a command line parameter for the standalone SuperSQL JAR executable file and as preference setting in the SuperSQL Eclipse plugin.

For future work, the processing of more query cases is deemed necessary to increase the optimization level of the proposed optimizer. The proposed optimizer is still not able to handle all possible query cases. However, it was designed it to fail safely and execute the original process when unhandled cases are encountered.

Acknowledgment

The authors would like to thank the members of the SuperSQL group of Toyama Laboratory for their support and Asian Development Bank – Japan Scholarship Program for the graduate scholarship grant.

References

[1] SuperSQL Website: http://ssql.db.ics.keio.ac.jp/en, [retrieved: November, 2012].

[2] T. Bain: "Are Relational Databases Doomed?", ReadWrite Enterprise, http://www.readwriteweb.com/enterprise/2009/02/is-the-relational-database-doomed.php (2009), [retrieved: November, 2012].

[3] F. Bonchi, C. Castillo, D. Donato, and A. Gionis: "Topical Query Decomposition", In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2008), pp. 52-60.

[4] T. T. T. Le, D. D. Doan, V. C. Bhavsar, and H. Boley: "A Bottom-up Algorithm for Query Decomposition", International Journal of Innovative Computing and Applications Volume 1, Issue 3 (July 2008), pp. 185-193.

[5] Y. Maeda and M. Toyama: "ACTIVIEW: Adaptive data presentation using SuperSQL", In Proceedings of the 27th International Conference on Very Large Data Bases (2001), pp. 695-696.

[6] S. G. Shin: "The Integration of Media Generators in SuperSQL Query Processor", Master's Thesis: Keio University School of Science for Open and Environmental Systems (2002).

[7] A. Silberchatz, H. F. Korth, and S. Sudarshan: "Database System Concepts (6th Edition)", McGraw-Hill, (2010).

[8] S. S. Skiena: "The Algorithm Design Manual", Springer, (2008).

[9] M. Toyama: "Three dimensional generalization of Target List for simple database publishing and browsing", Research and Practical Issues in Database (Proc. 3rd Australian Database Conference), World Scientific Pub. Co. (1992), pp. 139-153.

[10] M. Toyama: "SuperSQL: an extended SQL for database publishing and presentation", In Proceedings of ACM SIGMOD International Conference on Management of Data, (June 1998), pp. 584-586.

Comparison of Subspace Projection Method with Traditional Clustering Algorithms for Clustering Electricity Consumption Data

Minghao Piao[1], Hyeon-Ah Park[1], Kyung-Ah Kim[2], Keun Ho Ryu[1]
[1]Database/Bio informatics Laboratory, Chungbuk National University, Cheongju, South Korea
[1]{bluemhp, hapark, khryu}@dblab.chungbuk.ac.kr
[2]Department of Biomedical Engineering, Chungbuk National University, Cheongju, South Korea
[2]{kimka}@chungbuk.ac.kr

***Abstract*—There are many studies about using traditional clustering algorithms like K-means, SOM and Two-Step algorithms to cluster electricity consumption data for definition of representative consumption patterns or for further classification and prediction work. However, these approaches are lack of scalability with high dimensions. Nevertheless, they are widely used, because algorithms for clustering high dimensional data sets are difficult to implement and it is hard to find open sources. In this paper, we adopt several subspace and projected clustering algorithms (subspace projection method) and apply them to the electricity consumption data. Our goal is to find the strength and weakness of these approaches by comparing the clustering results. We have found that traditional clustering algorithms are better to be used for load profiling by considering global properties and subspace or projected methods are better to be used for defining load shape factors by analyzing local properties without prior knowledge.**

Keywords-subspace projection; traditional clustering; K-menas; SOMs; Two-Step; local property; global propert.

I. Introduction

The knowledge of how and when consumers use electricity is essential to the competitive retail companies. This kind of knowledge can be found in historical data of the consumers collected in load research projects developed in many countries. One of the important tools defined in these projects are different consumers' classes are represented by its load profiles. Load profiling or classification to assign different consumers to the existing classes has been a matter of research during last years. Traditional clustering algorithms like K-means, SOM and Two-Step algorithms are widely used for load profiling [1-10]. However, these approaches just considered the global properties of consumption patterns and ignored local properties during the process even there are many algorithms [11-30] can be used to analyze local properties of such high dimensional data sets.

Traditional clustering algorithms consider all of the dimensions of an input dataset in order to learn as much as possible. In high dimensional data, however, many of the dimensions are often irrelevant. These irrelevant dimensions can confuse clustering algorithms by hiding clusters in noisy data. In very high dimensions, it is common for all of the objects in a dataset to be nearly equidistant from each other.

However, in practice, the data points could be drawn from multiple subspaces, and the membership of the data points to the subspaces might be unknown. For example, trajectory sequences could contain several moving objects, and different subspaces might describe the motion of different objects in the different routing and place. Therefore, there is a need to simultaneously cluster the data into multiple subspaces and find a low-dimensional subspace fitting each group of points. Subspace and projected clustering algorithms localize their search and are able to uncover clusters that exist in multiple, possibly overlapping subspaces.

Another reason that many clustering algorithms struggle with high dimensional data is the curse of dimensionality. As the number of dimensions in a dataset increases, distance measures become increasingly meaningless. Additional dimensions spread out the points until, in very high dimensions, they are almost equidistant from each other.

The final goal of our study is trying to compare the strength and weakness between subspace projection method and traditional clustering algorithms for its application to cluster electricity consumption data.

Remaining paper is organized as following: section 2 lists several studies about clustering application to electricity data; section 3 introduces several subspace projection clustering methods; section 4 describes our experimental result and we made a conclusion on section 5.

II. Related Works

In [1], an electricity consumer characterization framework based on a KDD (Knowledge Discovery from Data) process is presented. The framework is a data mining model composed of load profiling module and the classification module. The load profiling module's goal is the partition of the initial data sample in a set of classes defined according to the load shape of the representative load diagrams of each consumer. This is made by assigning to the same class consumers with the most similar behavior, and to different classes consumers with dissimilar behavior. The first step of the module development was the selection of the most suitable attributes to be used by the clustering model and the best results are obtained by SOM [33] method. In the second step, the K-means algorithm [34] is used to group the weight vectors of the SOM's units and the final clusters are obtained. The load profiles for each class are obtained by averaging the representative load diagrams of the consumers assigned to the same cluster. In the classification module, load shape indexes are derived from each representative load profiles and discretized by using interval equalization method, C5.0 is used to build the tree based and rule based classification models.

In [2, 3], the approach is focusing on identifying typical usage profiles for households and clustering them into a few archetypical profiles with similar kinds of customers grouped together. The work tests the applicability of applying the framework to UK specific data and identifies possible enhancements or modifications to the framework in order to better fit the UK situation. Clustering algorithms are used to derive domestic load profiles that have been successfully used in Portugal and applied it to UK data. The paper found that Self Organizing Maps in Portuguese work is not appropriate for the UK data.

In [4], a novel clustering model is presented, tailored for mining patterns from imprecise electric load time series that are represented by interval numbers. The model consists of three components. First, to guarantee the correctness when comparing two load time series, normalization techniques like Z-score [36] and Max-Min linear normalization [36] are used to handle the differences of baselines and scales. Second, it adopts a similarity metric that uses Interval Semantic Separation based measurement. Third, the similarity metric is used with the k-means clustering method to handle imprecise time series clustering. The model gives a unified way to solve imprecise time series clustering problem and it is applied in a real world application, to find similar consumption patterns in the electricity industry. Experimental results have demonstrated the applicability and correctness of the proposed model.

In [5], Clustering is used to generate groupings of data from a large dataset with the intention of representing the behavior of a system as accurately as possible. To be precise, two clustering techniques K-means and Expectation Maximization (EM) have been utilized for the analysis of the prices curve, demonstrating that the application of these techniques is effective so to split the whole year into different groups of days, according to their prices conduct. Silhouette function is used for selecting number of clusters for K-means and cross validation is used for selecting number of clusters for EM. K-means has been confirmed to be the algorithm more suitable for daily prices classification.

Gabaldón et al. [6] proposed proposed a methodology in order to obtain a better support management decisions in terms of planning of bids and energy offers in real-time energy markets. Specifically, Self-Organizing Maps (SOM) and Statistical Ward's linkage to cluster electricity market prices into different groups. SOM and Ward's clustering provide a similar clustering for the price series in this study.

In [7], a SOM development is presented to achieve the segmentation and demand patterns classification for electrical customers on the basis of database measurements. The objective of this paper is to review the capacity of some of them and specifically to test the ability of Self-Organizing Maps (SOMs) to filter, classify, and extract patterns from distributor, commercializer or customer electrical demand databases. Before the clustering, demand data from time domain was transformed into frequency domain and it shows an improvement in clustering performance.

In [8], various unsupervised clustering algorithms (modified follow-the-leader, hierarchical clustering, K-means, fuzzy K-means) and the Self-Organizing Maps was tested and compared to group customers with similar electrical behavior into one. Furthermore, this paper discussed and compared various techniques like Sammon map [37], principal component analysis (PCA) [38], and curvilinear component analysis (CCA) [40] which are able to reduce the size of the clustering input data set, in order to allow for storing a relatively small amount of data in the database of the distribution service provider for customer classification purposes. The results of the clustering validity assessment performed in this paper show that the modified follow-the-leader and the hierarchical clustering run with the average distance linkage criterion emerged as the most effective ones. Both algorithms are able to provide a highly detailed separation of the clusters, isolating load patterns with uncommon behavior and creating large groups containing the remaining load patterns. The other algorithms tend to distribute the load patterns among some groups formed during the clustering process and, as such, are less effective.

In [9, 10], for deriving the load profiles, they have used K-means algorithm. The data was measured using Automatic Meter Reading (AMR) senses by Korea Electric Power Research Institute. Before the load profiling, the data was normalized into the range of 0 to 1 and the optimal number of clusters is determined by using reproducibility evaluation method.

From the above previous studies, we can see that the tasks of clustering multiple time series stream or many individual time series (i.e., electricity consumption data) have received significant attention, and most papers are using traditional clustering algorithms like K-means, SOM and Two-Step algorithms. However, these methods are not suitable for cluster analysis of time series like electricity data since these approaches are lack of scalability with high dimensions. Nevertheless, they are widely used, because algorithms for clustering high dimensional data sets are difficult to implement and it is hard to find open sources.

III. SUBSAPCE PROJECTION CLUSTERING METHODS

Subspace projection (Subspace clustering or projected clustering) is extension of traditional clustering that seeks to find clusters in different subspaces within a dataset. Subspace clustering algorithms might report several clusters for the same object in different subspace projections, while projected clustering algorithms are restricted to disjoint sets of objects in different subspace. These subspace projections also can be identified into three major paradigms characterized by the underlying cluster definition and parametrization of the resulting clustering [30].

Cell-based approaches search for sets of fixed or variable grid cells containing more than a certain number of objects. Subspaces are considered as restrictions of a cell in a subset of the dimensions. Cell-based approaches rely on counting objects in cells and with their discretization of data. This is similar to frequent itemset mining approaches. The first algorithm for cell-based clustering was introduced by CLIQUE [11]. CLIQUE defines a cluster as a connection of grid cells. Grid cells are defined by a fixed grid splitting each dimension in equal width cells. It consists of three steps: (1) Identification of subspaces that contain clusters, (2) Identification of clusters, (3) Generation of minimal description for the clusters. Base on similarity between mining frequent itemsets and discovering relevant subspace for a given cluster, MINECLUS [12] algorithm adapted FP-growth and employed branch-and-bound techniques to reduce the search space. The quality of the results was further improved by (1) pruning small clusters of low quality, (2) merging clusters close to each other with similar subspaces, and (3) assigning points close to some cluster, else considered as outliers. SCHISM [13] uses the notions of support and Chernoff-Hoeffding bounds to prune the space, and use a depth-first search with backtracking to find maximal interesting subspaces.

Density-based clustering paradigm defines clusters as dense regions separated by sparse regions. As density computation is based on the distances between objects, distances are computed by taking only the relevant dimensions into account in subspace clustering. They can be parametrized by specifying which objects should be grouped together according to their similarities or distances. Density-based approaches are based on the traditional clustering paradigm proposed in DBSCAN [14]. The first approach in this area was an extension of the DBSCAN named SUBCLU [15]. It works in greedy manner by restricting the density computation to only the relevant dimensions. Using a monotonicity property, SUBCLU reduces the search space by pruning higher dimensional projections in a bottom up way, but it results in an inefficient computation. A more efficient solution is proposed by FIRES [16]. It is based on efficient filter-refinement architecture and consists of three steps: first step is the pre-clustering, all 1-D clusters called base clusters are computed; second step is the generation of subspace cluster approximations. The base clusters are merged to find maximal dimensional subspace cluster approximations; third step is post-processing of subspace clusters, refines the cluster approximations retrieved after second step. INSCY [17] is extension of SUBCLU, which eliminates redundant low dimensional clusters which detected already in higher dimensional projections. INSCY is depth-first approach, processes subspaces recursively and prunes low dimensional redundant subspace clusters. Thus, it achieves an efficient computation of density-based subspace clusters. As the maximal high dimensional projection is evaluated first, immediate pruning of all its redundant low dimensional projections leads to major efficiency gains. DOC [18] is a density based optimal projective clustering. It requires each cluster to be dense only in its corresponding subspace. Density conditions refer only to the number of points that project inside an interval of given length, and do not make any assumption on the distribution of points.

Clustering-oriented approaches do not give a cluster definition like the previous paradigms. In contrast to the previous paradigms, clustering-oriented approaches focus on the clustering result by directly specifying objective functions like the number of clusters to be detected or the average dimensionality of the clusters. PROCLUS [19] partitions the data into *k* clusters with average dimensionality l, extending K-Medoids approach, which is called CLARANS [20]. The general approach is to find the best set of medoids by a hill climbing process, but generalized to deal with projected clustering. The algorithm proceeds in three phases: an initialization phase, an iterative phase, and a cluster refinement phase. The purpose of the initialization phase is to reduce the set of points and trying to select representative points from each cluster in this set. The second phase represents the hill climbing process that in order to find a good set of medoids. Also, it computes a set of dimensions corresponding to each medoid so that the points assigned to the medoid optimally form a cluster in the subspace determined by those dimensions. Finally, cluster refinement phase, using one pass over the data in order to improve the quality of the clustering. More statistically oriented, P3C [21] is comprised of several steps. First, regions corresponding to projections of clusters onto single attributes are computed. Second, cluster cores are identified by spatial areas that (1) are described by a combination of the detected regions and (2) contain an unexpectedly large number of points. Third, cluster cores are refined into projected clusters, outliers are identified, and the relevant attributes for each cluster are determined. P3C does not need the number of projected clusters as input. It can discover the true number of projected clusters under very general conditions. It is effective in detecting very low-dimensional projected clusters embedded in high dimensional spaces. P3C is the first projected clustering algorithm for both numerical and categorical data. Defining a statistically significant density, STATPC aims at choosing the best non-redundant clustering [22]. It consists of following steps: Detecting relevant attributes around data points; refining candidate subspaces; detecting a locally optimal subspace cluster; constructing the good candidate subspace; greedy optimization, solving the optimization problem on candidate subsapce by testing all possible subsets is still computationally too expensive in general.

There are also other subspace projection methods like DiSH [23], SUBCAD [24], CLICKS [25], PreDeCon [26], MAFIA [27], DUSC [28], FINDIT [29].

IV. EXPERIMENTS RESULTS AND DISCUSSION

The given test data set is the customers' power consumption data obtained from Korea Electric Power Research Institute (KEPRI). It has 165 instances with 25 attributes (one attribute is class label-contract types), and it is restructured as daily representative vectors:

$$V^{(c)} = \{V_0^C, ..., V_h^C, ..., V_H^C\}, \text{c = customer}, 0 < \text{h} < 24, \text{H} = 24 \qquad (1)$$

Where for each customer *c*, let $V^{(c)}$ denotes the total daily power usage of *c* in one day for 24 hours. The power usage is measured per 1 hour; therefore, each total daily power usages data has 24 dimensions, and dimension names are noted as H1, H2, H3,…, H14, H15,…, H23, H24, e.g., H14 means the measured time 14:00 PM.

Subspace projection algorithms are used with default parameter settings [10] except *Proclus*, the number of clusters of which is set to 11, since there were 11 different contract types in the test data set. From Tables 1 to 3, we can see that when using K-means, SOM and Two-Step to cluster the given data, SOM has achieved better result than K-means and Two-Step since K-means and Two-Step have assigned most number of instances into one cluster. At least, traditional clustering approaches, they have assigned all instances into the clusters. In contrast, for example, FIRES is used with default parameter settings and it resulted in 13 clusters where 92 un-clustered instances are. It means traditional algorithms are able to be used for load profiling to find representative consumption patterns as previous studies while subspace and projected clustering algorithms are not. The reason is that traditional clustering algorithms consider all dimensions which present the global properties of customers' consumption patterns. The given contract types are determined by consumer's consumption activities which characterize the global shapes (global property) of these consumption patterns. Since we have used these contract types as given class information, and subspace projection methods use subspace or projections of whole dimensions instead of all, the result of traditional approaches are better than subspace and projection methods.

TABLE I. CONFUSION MATRIX OF SOM

	Found Clusters										
Contract Types	1	2	3	4	5	6	7	8	9	10	11
1	0	1	0	0	0	0	0	3	0	0	0
2	4	3	1	22	2	0	0	56	2	0	0
3	0	0	0	0	0	0	0	7	0	0	0
4	0	0	0	0	0	0	0	0	0	0	2
5	0	1	2	5	1	0	1	6	0	3	0
6	0	0	1	0	0	1	0	1	0	0	0
7	0	0	0	0	1	0	1	3	2	1	1
8	0	1	0	1	0	0	0	0	0	0	0
9	0	1	0	0	0	0	0	0	0	0	0
10	5	1	1	3	0	0	0	0	0	0	0
11	0	0	1	0	0	0	0	17	0	0	0

TABLE II. CONFUSION MATRIX OF K-MEANS

	Found Clusters										
Contract Types	1	2	3	4	5	6	7	8	9	10	11
1	3	0	1	0	0	0	0	0	0	0	0
2	88	0	2	0	0	0	0	0	0	0	0
3	7	0	0	0	0	0	0	0	0	0	0
4	0	0	0	1	1	0	0	0	0	0	0
5	12	1	3	0	0	1	1	0	0	0	1
6	1	0	1	0	0	0	0	0	0	1	0
7	6	0	0	0	0	1	0	0	1	0	1
8	1	0	1	0	0	0	0	0	0	0	0
9	1	0	0	0	0	0	0	0	0	0	0
10	9	0	0	0	0	0	0	1	0	0	0
11	17	0	0	0	0	0	0	1	0	0	0

TABLE III. CONFUSION MATRIX OF TWO-STEP

	Found Clusters										
Contract Types	1	2	3	4	5	6	7	8	9	10	11
1	0	0	0	0	0	0	0	0	0	3	1
2	1	0	0	0	0	0	0	0	0	88	1
3	0	0	0	0	0	0	0	0	0	7	0
4	0	0	0	0	0	0	0	1	1	0	0
5	2	0	0	2	1	1	0	0	0	11	2
6	0	1	0	0	0	0	0	0	0	1	1
7	0	0	0	0	1	1	0	0	1	6	0
8	0	0	0	0	0	0	0	0	0	1	1
9	0	0	0	0	0	0	0	0	0	1	0
10	0	0	1	0	0	0	0	0	0	9	0
11	0	0	1	0	0	0	0	0	0	17	0

Table 4 shows the dimensions relevant to definition of clusters. These relevant dimensions are describing the local properties of electricity consumption data and it is useful to define load shape factors [31, 32, 33]: for defining load shape factors, we have to find time intervals first which shows big difference of electricity usage during the time intervals. In previous studies, load shape factors are defined by experts according to their experience. From Table 4, we can see dimensions in each cluster which maximize the difference between itself with others. Definition of load shape factors can be done by considering time intervals of these neighboring relevant dimensions. For example, load shape factor for cluster-0 should consider time interval of 01:00 AM ~ 09:00 AM, and for cluster-11, have to consider time interval of 18:00 PM ~ 19:00 PM and 22:00 PM~23:00 PM. There will be some load shape factors overlaps in same time intervals. This problem will not cause serious problems since we can assign weight for each shape factors to cover it.

TABLE IV. DIMENSIONS RELATED TO DEFINITION OF CLUSTERS (DM:DIMENSIONS)

DM	**Cluster Numbers**												
	0	1	2	3	4	5	6	7	8	9	10	11	12
1	1	1											
2	1		1								1		
3	1	1											
4	1		1										
5	1		1										
6	1		1		1								
7	1				1						1		
8	1					1							
9	1				1								
10				1		1	1						
11				1		1	1						1
12				1			1						
13				1		1	1						
14				1		1	1	1	1		1		
15				1		1	1		1		1		
16				1		1	1		1				
17				1			1		1		1		
18				1			1		1		1	1	
19				1			1	1			1	1	1
20				1			1						1
21				1			1	1					
22				1		1	1	1	1			1	
23				1				1		1		1	
24								1			1		1

From Figure 1, we can see some dimensions which is mostly related to the local property of the load usage patterns: often appeared dimensions tend to have most significant discriminating power to differentiate clusters. Therefore, suppose the given count threshold is 4 then dimensions {*11*, *14*, *15*, *16*, *17*, *18*, *19*, *22*, *23*} will be the most useful dimensions to define load shape factors. Combinations of these dimensions also can be most useful time intervals to derive load shape factor if and only the appearance of them are high enough and they are neighboring. Furthermore, we can use some relevance evaluation techniques to evaluate the defined load shape factors.

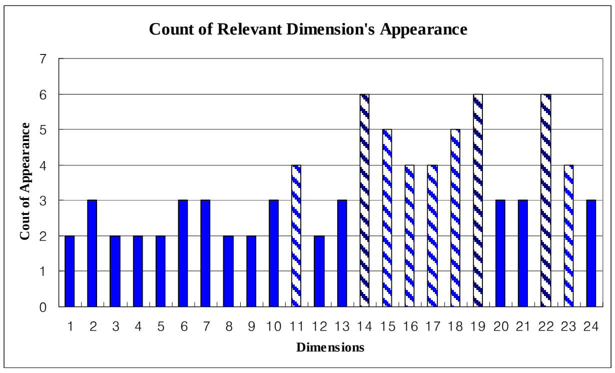


Figure 1. Count of relevant dimension’s appearance from table 3.

V. Conclusion

In this study, we have used small data sets consists of 165 instances with 24 dimensions. The main objective of clustering is to find high quality clusters within a reasonable time. However, we found that several subspace and projected clustering algorithms like P3C, CLIQUE and DOC took too much time to get in the result.

Also, we found that traditional clustering algorithms like K-means, SOM and Two-Step are better than subspace and projected clustering approaches when applied to define representative consumption patterns even they are lack of scalability with high dimensions - some instances are not assigned to the clusters when using subspace and projected clustering algorithms. However, relevant dimensions used to define clusters in subspace and projected approaches are able to be used for extracting load shape factors for classifications works by analyzing local and global properties of electricity consumption data set without prior knowledge.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 2012-0000478).

References

[1] V. Figueiredo, F. Rodrigues, Z. Vale, and J. B. Gouveia, “An electric energy consumer characterization framework based on data mining techniques,” IEEE Transactions on Power Systems, vol. 20, May 2005, pp. 596-602, doi:10.1109/TPWRS.2005.846234.

[2] I. Dent, U. Aickelin, and T. Rodden, “The Application of a Data Mining Framework to Energy Usage Profiling in Domestic Residences using UK data,” Proc. Research Student Conference on “Buildings Do Not Use Energy, People Do?”, June 2011.

[3] I. Dent, U. Aickelin, and T. Rodden, “Application of a clustering framework to UK domestic electricity data,” Proc. The 11th Annual Workshop on Computational Intelligence, 2011, pp. 161-166.

[4] Q. D. Li, S. S. Liao, and D. D. Li, “A Clustering Model for Mining Consumption Patterns from Imprecise Electric Load Time Series Data,” Proc. Fuzzy Systems and Knowledge Discovery, Lecture Notes in Computer Science, vol. 4223, 2006, pp. 1217-1220, doi: 10.1007/11881599_152.

[5] F. Martínez-Álvarez, A. Troncoso, J.C. Riquelme, and J.M. Riquelme, “Partitioning-clustering techniques applied to the electricity price time series,” Proc. The 8th international conference on Intelligent data engineering and automated learning, 2007, pp. 990-999, 2007, doi: 10.1007/978-3-540-77226-2_99.

[6] A. Gabaldón, A. Guillamón, M.C. Ruiz, S. Valero, C. Álvarez, M. Ortiz, and C. Senabre, “Development of a methodology for clustering electricity-price series to improve customer response initiatives,” IET Generation, Transmission & Distribution, vol. 4, June 2010, pp. 706-715, doi:10.1049/iet-gtd.2009.0112.

[7] S. V. Verdu, M. O. Garcia, C. Senabre, A. G. Marin, and F. J. Garcia Franco, “Classification, filtering and identification of electrical customer load patterns through the use of SOM maps,” IEEE Transactions on Power Systems, vol. 21, no. 4, Nov. 2006, pp. 1672-1682, doi:10.1109/TPWRS.2006.881133.

[8] G. Chicco, R. Napoli, and F. Piglione, “Comparisons among Clustering Techniques for Electricity Customer Classification,” IEEE Transactions on Power Systems, vol. 21, no. 2, May 2006, pp. 933-940, doi:10.1109/TPWRS.2006.873122.

[9] M. H. Piao, H. G. Lee, J. H. Park, and K. H. Ryu, “Application of Classification Methods for Forecasting Mid-Term Power Load Patterns,” Proc. Communications in Computer and Information Science, vol. 15, 2008, pp. 47-54, doi:10.1007/978-3-540-85930-7_7.

[10] J. H. Shin, B. J. Yi, Y. I. Kim, H. G. Lee, and K. H. Ryu, “Spatiotemporal Load-Analysis Model for Electric Power Distribution Facilities Using Consumer Meter-Reading Data,” IEEE Transactions on Power Delivery, vol. 26, no. 2, April 2011, pp. 736-743, doi: 10.1109/TPWRD.2010.2091973.

[11] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, “Automatic subspace clustering of high dimensional data for data mining applications,” Proc. ACM SIGMOD international conference on Management of data, vol. 27, no. 2, June 1998, pp. 94-105, doi: 10.1145/276304.276314.

[12] M. L. Yiu and N. Mamoulis, “Frequent-pattern based iterative projected clustering,” IEEE International Conference on Data Mining, Nov. 2003, pp. 689-692, doi:10.1109/ICDM.2003.1251009.

[13] K. Sequeira and M. Zaki, “SCHISM: A new approach for interesting subspace mining,” International Journal of Business Intelligence and Data Mining, Dec. 2005, pp. 137-160, doi: 10.1504/IJBIDM.2005.008360.

[14] M. Ester, H. P. Kriegel, J. Sander, and X. Xu., “A density-based algorithm for discovering clusters in large spatial databases,” Proc. The 2nd International Conference on Knowledge Discovery and Data Mining, 1996. pp. 226-231.

[15] K. Kailing, H.-P. Kriegel, and P. Kroger, “Density-connected subspace clustering for high-dimensional data,” Proc. The 4th SIAM Conference on Data Mining, April 2004, pp. 246-257.

[16] H. P. Kriegel, P. Kroger, M. Renz, and S. Wurst, “A generic framework for efficient subspace clustering of high-dimensional data,” Proc. The 5th International Conference on Data Mining, Nov. 2005, pp. 250-257, doi:10.1109/ICDM.2005.5.

[17] I. Assent, R. Krieger, E. Muller, and T. Seidl, “INSCY: Indexing subspace clusters with in-process-removal of redundancy,” Proc. IEEE International Conference on Data Mining, Dec. 2008, pp. 719-724, doi:10.1109/ICDM.2008.46.

[18] C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. M. Murali, “A Monte Carlo algorithm for fast projective clustering,” Proc. ACM SIGMOD International Conference on Management of Data, 2002, pp. 418-427, doi: 10.1145/564691.564739.

[19] C. Aggarwal, J. Wolf, P. Yu, C. Procopiuc, and J. Park, “Fast algorithms for projected clustering,” Proc. ACM SIGMOD international conference on Management of data, June 1999, pp. 61-72, doi: 10.1145/304182.304188.

[20] R. T. Ng and J. W. Han, “Efficient and Effective Clustering Methods for Spatial Data Mining,” Proc. The 20th International Conference on Very Large Data Bases, 1994, pp. 144-155.

[21] G. Moise, J. Sander, and M. Ester, “P3C: A robust projected clustering algorithm,” Proc. IEEE International Conference on Data Mining, Dec. 2006, pp. 414-425, doi:10.1109/ICDM.2006.123.

[22] G. Moise and J. Sander, “Finding non-redundant, statistically significant regions in high dimensional data: a novel approach to projected and subspace clustering,” Proc. The 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2008, pp. 533-541, doi:10.1145/1401890.1401956.

[23] E. Achtert, C. BÄohm, H. P. Kriegel, P. KrÄoger, I. MÄuller-Gorman, and A. Zimek, “Detection and visualization of subspace clusters hierarchies,” Proc. The 12th International Conference on Database systems for advanced applications, 2007, pp. 152-163, doi: 10.1007/978-3-540-71703-4_15.

[24] G. Gan and J. Wu, “Subspace clustering for high dimensional categorical data,” ACM SIGKDD Explorations Newsletter, vol. 6, Dec. 2004, pp. 87-94, doi:10.1145/1046456.1046468.

[25] M. Zaki, M. Peters, I. Assent, and T. Seidl, “CLICKS: an effective algorithm for mining subspace clusters in categorical datasets,” Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2005, pp. 733-742, doi: 10.1145/1081870.1081965.

[26] C. Bohm, K. Kailing, H.-P. Kriegel, and P. Kroger, “Density Connected Clustering with Local Subspace Preferences,” Proc. IEEE International Conference on Data Mining, Nov. 2004, pp. 27-34, doi: 10.1109/ICDM.2004.10087.

[27] H. Nagesh, S. Goil, and A. Choudhary, “Adaptive grids for clustering massive data sets,” Proc. The 1st SIAM International Conference on Data Mining, 2001, pp. 1-17.

[28] I. Assent, M. Krieger, E.Muller, and T. Seidl, “DUSC: Dimensionality Unbiased Subspace Clustering,” Proc. IEEE International Conference on Data Mining, 2007, pp. 409-414, doi: 10.1109/ICDM.2007.49.

[29] K. G. Woo, J. H. Lee, M. H. Kim, and Y. J. Lee, “FINDIT: a Fast and Intelligent Subspace Clustering Algorithm using Dimension Voting,” Information and Software Technology, vol. 46, no. 4, 20045, pp.255-271, doi:10.1016/j.infsof.2003.07.003.

[30] E. Müller, S. Günnemann, I. Assent, and T. Seidl, “Evaluating Clustering in Subspace Projections of High Dimensional Data,” Proc. The 35th International Conference on Very Large Data Bases (VLDB 2009), vol. 2, Aug. 2009, pp. 1270-1281.

[31] G. Chicco, R. Napoli, P. Postolache, M. Scutariu, and C. Toader, “Electric energy customer characterization for developing dedicate market strategies”, Proc. IEEE Porto Power Tech conference, Sep. 2001, doi:10.1109/PTC.2001.964627.

[32] G. Chicco, R. Napoli, P. Postolache, M. Scutariu, and C. Toader, “Customer characterisation options for improving the tariff offer”, IEEE Transactions on Power Systems, vol. 18, no. 1, Feb. 2003, pp. 381-387, doi:10.1109/TPWRS.2002.807085.

[33] J. B. Lee, M. H. Piao, and K. H. Ryu, “Incremental Emerging Patterns Mining for Identifying Safe and Non-safe Power Load Lines”, Proc. IEEE International Conference on Computer and Information Technology, 2010, pp. 1424-1429, doi: 10.1109/CIT.2010.255.

[34] T. Kohonen, T. S. Huang, and M. R. Schroeder, Self-Organizing Maps, Springer-Verlag, December 2000.

[35] J. B. MacQueen, “Some Methods for classification and Analysis of Multivariate Observations,” Proc. 5th Berkeley Symposium on Mathematical Statistics and Probability, 1967, pp. 281-297.

[36] J. W. Han, M. Kamber, J. Pei, Data Mining: Concepts and Techniques. Third Edition, The Morgan Kaufmann Series in Data Management Systems, July 6, 2011.

[37] J. W. Sammon, “A nonlinear mapping for data structure analysis,”IEEE Transactions on Computers, vol. C-18, no. 5, May 1969, pp. 401-409, doi:10.1109/T-C.1969.222678.

[38] J. E. Jackson, A User’s Guide to Principal Components. New York: Wiley, 1991, pp. 1-25.

[39] P. Demartines and J. Herault, “Curvilinear component analysis: A selforganizing neural network for nonlinear mapping of data sets,” IEEE Transactions on Neural Networks, vol. 8, no. 1, Jan. 1997, pp. 148-154, doi:10.1109/72.554199.

An Approach for Dynamic Materialization of Aggregates in Mixed Application Workloads

Stephan Müller
Hasso Plattner Institute
University of Potsdam
August-Bebel-Str. 88
14482 Potsdam, Germany
stephan.mueller@hpi.uni-potsdam.de

Carsten Meyer
Hasso Plattner Institute
University of Potsdam
August-Bebel-Str. 88
14482 Potsdam, Germany
carsten.meyer@hpi.uni-potsdam.de

Hasso Plattner
Hasso Plattner Institute
University of Potsdam
August-Bebel-Str. 88
14482 Potsdam, Germany
hasso.plattner@hpi.uni-potsdam.de

***Abstract*—Aggregate queries are one of the most resource intensive operations for databases systems. Despite scanning and processing large data sets, they only return relatively small outputs, making them predestined for reuse in future queries. The challenge is to manage the infinite number of possible data areas that can be selected for caching, identify the relevant ones and materialize the corresponding aggregates on a reusable level of granularity. While analytical database systems often save materialized data in cubes of aggregated chunks this is not feasible in systems with transactional insert and change processes. We present the concept *dynamic materialized aggregate views (DMAV)* that materializes aggregates based on mixed application workloads. Using control tables, aggregate query structures as well as selected data areas are captured and kept track of. We evaluate data access characteristics and only materialize aggregates that are read often but at the same time have only few updates resulting in low aggregates maintenance costs.**

***Index Terms*—caching, data aggregation, dynamic view materialization, mixed-workload, OLAP, OLTP.**

I. Introduction

In the last decades, enterprise data management systems have evolved in two directions: transactional (OLTP) and analytical (OLAP) systems. Queries with aggregation functions that scan, read, and finally calculate large amounts of data are very resource intensive for both systems [1], [2], resulting in system-specific work-arounds: In OLTP systems, the applications maintain tables that contain materialized aggregates and deliver by magnitudes faster access to the desired abstraction level compared to aggregating on the original base table. On the downside, every change on the base data must be propagated to the materialized table. OLAP systems on the other hand often contain data schemas that are an aggregated abstraction of the underlying transactional data. The selection and definition of views is either done manually by an administrator or based on sophisticated materialized view selection algorithms. This enables fast response times for pre-defined queries. However, this setting inhibits flexible, ad-hoc queries, hence realistic analytical scenarios.

To overcome these and other drawbacks of the separation and to enable analytical processing directly on the transactional data, in-memory databases such as HYRISE [3] or SAP HANA [4] have evolved in academia and industry. The workload of the combined system is called *mixed workload* and consists of transactional as well as analytical data access patterns [5], [6]. Application scenarios such as dunning, cross selling and availability to promise [7] are an example for such workloads.

Although resource-intensive operations such as aggregations can now be processed directly on the base data and provide acceptable response times, reading a pre-aggregated result is always more efficient, especially in multi-user scenarios and when the same or similar queries are executed repeatedly. When considering the materialized aggregate maintenance costs on the other hand, an on-the-fly aggregation strategy can be more efficient. Nevertheless, the application should not deal with the maintenance of materialized aggregate tables and operate on the base data directly. We believe that the decision of which parts of a table should be materialized as an aggregate should be made in the database layer and not within the application for two reasons: First, this significantly simplifies application development as the aggregations are executed transparently. Second, the materialization of aggregates should be based on the actual workload and not on a static basis.

In this paper we introduce the concept of *dynamic materialized aggregate views*. This concept selectively materializes aggregates, based on the characteristics of a mixed workload, independent from the application. To our best knowledge, there are no techniques available that allow materialization and reuse of generic aggregates in transactional databases which are based on dynamic changing data characteristics. Analyzing these workloads, one can find application-dependent data ranges that do not change. Financial statements of past periods, point of sales data or historic production orders are often described as *cold data*. Still, they can be relevant for analytical queries. This paper proposes to analyze a given mixed workload in order to find those aggregates that are frequently requested and are based on cold data. Stored in dedicated control tables, the database shall be able to keep track of those *hot aggregates*. Instead of calculating hot aggregates over and over again they are materialized and can then be used to answer matching queries faster while, at the same time reducing CPU and memory bandwidth allocation.

We contribute by providing a generic framework that identifies and characterizes aggregate queries in a mixed workload. We describe how historical data relevance and change frequency can be evaluated in order to find important data regions, so called hot spots. Our concept will be provided for dynamically materialized aggregates on database level, as opposed to application based hard coded materialized tables.

The remainder of the paper is structured as follows: Section II provides background information and evaluates related work on materialization in analytical databases, optimization techniques and the nature of aggregate functions. Section III introduces the analysis framework that is used to identify aggregate queries and extract their structure as well as tracking hot spot areas. The core concept of DMAVs is presented and explained using an example in Section IV. It will be shown how different aggregate queries respectively their structural characteristics are stored in a relation and used for view matching and materialization. The number of materialized aggregates will not only be limited to the relevant ones but will also consider changes in the transactional base data. Different ways of optimizing the definition of data areas (hot spots) will be presented in Section V. Finally, in Section VI we preliminarily evaluate our concept with a mixed workload.

II. Background and Related Work

In this section, we present background information for aggregation queries and evaluate related work in the areas of clustering hot and cold data, materialized view selection and maintenance, and cache replacement techniques.

A. Data aggregations

Aggregates are the result of a query that has an aggregation function f() applied to an attribute *A* of an arbitrary relation r. The granularity of aggregates depends on the cardinality of its grouping attributes *G*. The structure of an aggregate query consists of three main aspects:

1) The aggregate function and the attribute *A*
2) The group by clause and its fields *G*
3) The relation *r* and its data selection (where clause structure)

In relational algebra the aggregation operation over a schema (A1, A2, ... An) of relation r is written as follows: **G1, G2, ..., Gm g f1(A1), f2(A2), ..., fk(An') (r)**

Let us assume that we have a table named "Sales" with three columns, namely "Product_Id", "Year" and "Rev". We wish to find the maximum Rev of each Product_Id in Year = 2011. This is accomplished by:
Product_IdgMax(Rev)(Select * Sales where Year = 2011). The relation "r" (Select * Sales where Year = 2011) could also be a relation joined over multiple tables. The table of the attribute "A" is called the *base table* and all rows selected in relation "r" are the *base data* of the aggregates. A materialized aggregate view is defined by the grouping and selection of aggregates on the base data. This data is then materialized using another table in the database.

B. Clustering data in hot and cold areas

Categorizing data areas according to their usage is especially relevant in mixed workload databases. Cold data that is not accessed for updates or maybe even reads can be handled in other ways than hot data that is accessed more frequently. In [8] Funke et al. measure the *temperature* of data, based on memory page granularity. They store attributes column-wise, each attribute vector cut into chunks the size of a memory page. For every chunk they determine its usage from the number of page access but do not distinguish between read and write access. They categorize the usage into *hot*, *cooling*, *cold*, and *frozen* areas. In this paper, chunk based granularity for usage characteristics is not sufficient as hot spots must be described and captured on row level granularity.

C. Materialized view selection and maintenance

Materializing views in order to speed up the processing of queries is not new [9], [10]. Today, all major analytical database systems support the definition of de-normalized, pre-aggregated views [11], [12], [13] for resource-intensive queries. Although static view selection is a common approach in OLAP systems, it contradicts with the dynamic nature of decision support analysis as well as the diverse usage scenarios of enterprise standard software. Kotidis et al. show that the time frame needed to update those views becomes more and more the limiting factor, as the cost per disk decreases while the number of views increases [2]. In their paper they propose a concept for data warehouse systems called *DynaMat* that materializes results based on a goodness measure that is extracted from the workload demand. This measures the relevance of each materialized multi-dimension range fragments and limits the data amount to a certain disk space. In [14] Zhou et al. propose a different approach, but address the same problem. They use key parameters in the where-clauses and an associated control table to filter only the most relevant data. The values (filters) in the control table, dynamically change the number of materialized rows of the view.

This paper builds upon the idea of a workload demand [2] as well as on the concept of control tables [14]. Adapted to a mixed workload, the approach of this paper does not rely on predefined views. Instead, it extracts arbitrary aggregate queries and their structure from a given workload demand. Multi-parameter control tables are used to filter includable and excludable data areas eligible for materialization. This will ensure that only those aggregates are materialized that are selected frequently and do not have frequent updates in their base records, minimizing view maintenance costs.

In [15] Deshpande et al. propose a way to build aggregates based on chunks of aggregates. The idea of chunk based caching [16] requires a simple correspondence between chunks (called closure property) at different levels of aggregation. Without having defined hierarchies, respectively hierarchical dimensions that are used to select aggregates, there is no possibility to aggregate low level chunks up to high level chunks. However, materializing dynamic aggregates shall not require hierarchical dimensions, but adapt to a mixed, changing workload. Still, in section V we will look into ways how simple hierarchies in dimensional- & fact- tables can be used to harmonize the definition of data areas.

The problem of view matching has already received considerable attention in research. Larson and Yang were the first to describe view matching algorithms for SPJ queries and views [9], [10]. In [12] Goldstein and Larson extended SPJ view matching for aggregate queries. They show conditions a view has to fulfill in order to compute a query. A general survey on view matching can be found in [13] by A. Y. Halevy. This paper will only explain those steps, additionally needed to identify aggregate queries that can be answered by a DMAV.

D. Cache replacement

In order to measure and manage the data area definitions (hot aggregates, hot change data) according to their usage and cost, there must be a replacement and admission mechanism for materialization filters in the control tables. In [17] Scheuermann et al. discuss a cache manager (LNC-RA) that employs cache replacement and admission policy that aims at maximizing the query execution cost savings. In comparison to a vanilla LRU algorithm, this one promises to be a good algorithm for managing control tables. Even though the control tables do not manage the caching of data pages but the materialization of aggregates, similar costing functions can be used. With a cost-based ranking, introduced in V-A filters in control table can be compared.

III. Analysis Framework

The framework that is used to analyze the workload is explained within this section. It is shown how the framework is used to identify aggregates including their structure as well as capture relevant data areas as the basis for DMAV. As a generic framework the basic idea is to analyze and identify arbitrary workload characteristics by parsing SQL statements being executed on a database. Today, all major databases (Oracle, DB2, SQL Server, MySQL) are able to trace SQL workloads. This paper shows algorithms that find and save characteristics as well as additional workload information, such as the number of executions, rows processed and the elapsed time in a fact table of an analytical star schema. The framework proposed in this paper was implemented as an database external Microsoft .Net application that can read SQL statements either from exported CSV files, or directly access specific log tables in the database. The application can parse those SQL statements and write commands (facts) into the star schema. As this paper provides a concept for materializing aggregates, the question how this functionality can be integrated into a DBMS is not addressed.

A. Commands

Commands are a substantial component of the analysis framework. Each command describes a certain characteristic of a database table or field that was identified by the parsing code. The used parser is able to distinguish between different

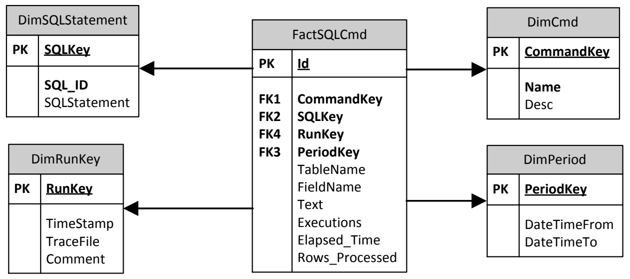


Fig. 1. Star Schema Analysis Framework

statement types (select, insert, delete, update, etc.) , detect projected columns, tables and join relations, predicates and group-by clauses. In general all aspects of an SPJG query can be identified and tracked. The characteristics parsed and needed for the the identification of dynamic, materialized aggregated view are defined as follows:

AGR_SUM SUM aggregates
AGR_AVG AVG aggregates
AGR_CNT COUNT aggregates
AGR_GRP GROUP BY table field
AGR_WCL WHERE clause with values
AGR_WCP WHERE clause w/o field values
UPD_WCL WHERE clause in update statements with values
UPD_WCP WHERE clause in update statements, w/o values
AGR_GCL GROUP BY clause in aggregate queries

B. Star schema

The analysis star schema as illustrated in Figure 1 allows to capture all the parsed information into relational tables. Next to the fact table there are four dimensional tables that describe each captured fact.

DimCmd All commands with their description and version
DimPeriod The time span information of a period
DimRun Holds information for each executed analysis run
DimSQLStatement Stores the actual SQL query information

The analyzer parses each query, looks for certain characteristics, and then creates a fact. The sum aggregate parsing code (AGR_SUM) for example creates a new fact every time it finds a sum aggregate function in a query, while AGR_GRP creates a fact for every group-by table field found in an aggregate function. The captured facts, provided by the workload, which are measurable, are:

TableName Name of the table
FieldName Name of the table field
Executions Number of executions
RowsExecuted Number of rows read, updated, etc.
ElapsedTime Total time in ms used for execution
Text Used to save string values (e.g. values of predicates)

C. Ways to mine those data

Once the workload is parsed and captured in the analysis schema, it is possible to query and mine this data according to certain questions:

TABLE I
WORKLOAD ANALYSIS

Field	Value
SQLKey	5832
Base table T_B	BSEG
View predicate P_V	BSEG.VBELN = VBRK.VBELN
Exec	64
Rows	658
Elapsed Time	87036ms
Aggregate A	AGR_SUM(BSEG.DMBTR)
GroupBy G	BSEG.PSWSL-&&-BSEG.GJAHR
Select predicates P_S	NE(BSEG.AUGDT)-&&-BT(VBRK.FKDAT)-&&–EQ(BSEG.KOART)-&&-EQ(BSEG.SHKZG)-&&-EQ(BSEG.BUKRS)
Filters (Select predicate values) P_{SV}	('0001-01-01','2001-11-01','2002-01-31','D', 'S','6000'), (...), (...)

- How often has which aggregate function been executed,
- On what tables do they base,
- What group-by fields have been used, and
- How do the where-clause predicates look like?

All these questions can be ranked by the provided metrics (executions, rows processed and elapsed time) and additionally joined as required using the SQLKey. In Table I it can be seen, how the identified commands are joined into a usable result set. The set is grouped on: T_B, P_V, A, G, and P_S. From that information, structural information as well the data areas (filters) can be extracted. Besides, all required infrastructure (view definitions, aggregate structure) for DMAVs are derived.

IV. DYNAMIC MATERIALIZED AGGREGATE VIEWS

The idea of DMAV is that aggregates of generic aggregate queries are selectively materialized, limited to hot spot data areas. Therefore a data structure keeping track of generic aggregate functions is needed. As there are no predefined aggregate views, even ad-hoc executed queries are persisted to a central *structure control table* T_S.

Additionally to this structure there are *data control tables* T_D for every base table that holds the hot spot data (as include and exclude filters), dynamically adjusted to the workload. Both structures, T_D and T_S are required to materialize aggregated results, based on the group-by and selection attributes of the aggregate query. They are also both used for view matching (see Section IV-F). Only when a query matches both structure (T_S) and data T_D, it can be answered by the corresponding V_{dA} view. Because V_{dA} contains (is grouped by) the selection attributes of the aggregate query, (see Section: IV-E3) it is possible for the aggregate query to directly select on the V_{dA} instead of the base relation. Because of this grouping the result of an aggregate query cannot be saved to the V_{dA} during runtime but is materialized asynchronously. The concept of DMAV is split into two phases. A periodically executed *update phase* (see Figure 3) that runs asynchronous to the actual database and a second phase (see Figure 2), which is active during query runtime (*online phase*). This separation provides the key requirements that the materialized aggregates can be invalidated and maintained when the base data changes, adapting to a changing workload.

The main task of the online phase is view matching. Every aggregate query is matched against T_S and T_D. When it hits a hot spot, this means the query can be answered by the DMAV V_{dA}. Otherwise the query has to be answered by the base tables. At the same time, unknown structures or newly selected data areas (filters) are added to T_S and to T_D. In both ways (hit or miss) the usage, cost, and the size of the result of a filter is captured in T_D. Thus, a changing workload is recognized as new structures and new filters during the online phase. However, as new aggregates are not directly materialized to the according V_{dA} view during runtime, those new filters are marked as invalid. During the online phase, every change statement to a base table is tracked and captured as an exclude filter in T_S. Additionally, all affected include-filters are invalidated (see: Table II).

Based on change metrics and the system load, the update phase of a T_D is triggered periodically. Those metrics shall not be considered in further detail. Still, they could be dependent on the number and usage of new filters and structures as well as available system resources. Obviously, every T_D table that is updated puts a load on the system and therefore should only be executed when necessary. However, every T_D can be updated separately. Hence, it is easy to scale with the workload demand of different base tables.

During the update phase a V_{dA} view gets materialized. Before that step, new filters in T_S and aggregate structures in T_D are evaluated and optimized. As can be seen in Figure 3 this includes four steps: cost ranking, variable distribution, time correlation, and structural harmonization. Cost ranking calculates a profit for each filter, based on usage statistics captured during online phase: *EXEC* (number of executions), *COST* (time of execution) as well as *FUSE* (first use time) are captured by the runtime manager and the numbers used for profit calculation (see: Section V-A). This ensures that only the relevant filters are part of the control table, vacant includes and excludes are cleaned up and the memory consumption used for materialized aggregates is kept small.

In Section V-C distribution of variables and correlation between current date and filter-variables V-D is used to predict future aggregate selections. In order to improve the view matching, filters are extended during optimization. Thus, include filters are not only based on captured workload selections, but also anticipate aggregates that have not been, but are probable to be selected. E.g.: a correlation between an attribute "posting date" of an exclude filter and the current date and time indicates that changes in a base table always happen in a certain time frame. Harmonization of structures (Section V-B) tries to find selection predicate (P_S) that can be replaced by one unifying attribute in order to keep the number of diverse structures and materialized views V_{dA} small. Finally, the optimized and harmonized filters in control table T_D are used to limit the number of materialized aggregates in view V_{dA}.

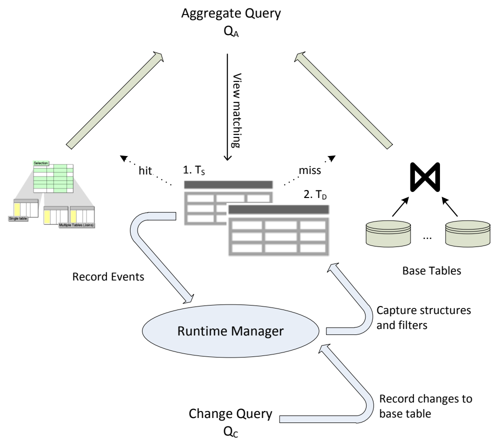


Fig. 2. Runtime Architecture

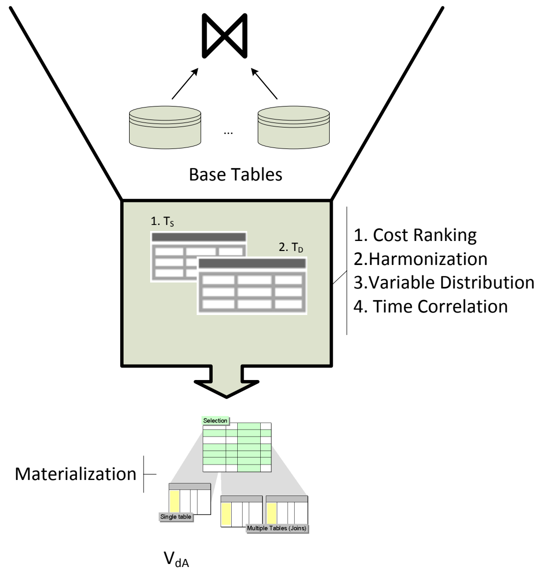


Fig. 3. Update Architecture

The basic steps of each phase are as follows:

1) online phase
   - View matching of aggregate queries
   - Provide aggregates from V_{dA} or refer to base tables
   - Capture filter usage in T_D
   - Add missed aggregates in T_S as structures
   - Add missed data areas in T_D as include filter
   - Add changed data areas in T_D as exclude filter
   - Invalidate aggregates that clash with change areas
   - Trigger update phase
2) Update phase
   - Parse historic workload: identify, save characteristics
   - Parse new structures and data areas
   - Analyze, rank workload characteristics
   - Create materialized aggregate views V_{dA}
   - Create control tables T_D
   - Optimize control table T_D
   - Materialize aggregates V_{dA}

A. Global Structure table T_S

T_S is designed as one global table holding the structures of all aggregate queries that shall be used in a dynamic materialized view V_{dA}. A structure is defined by the aggregate function, its group-by and its predicate-fields. Aggregate queries that have the same predicates, but different group-by fields, can share the same structure, as long as every group-by is covered. However, two aggregate queries, even if they have the same function and grouping, must be handled as two separate structures when they have different selection predicates. This is because two aggregate queries, one selecting using attribute year, the other with a selection on product id produce overlapping, incomparable results.

1) Data Schema: The data schema of T_S consists of the following fields:

SID (int) PK, referenced in T_D
T_B (char) Name of the base table
P_V (char) Predicate that defines base view V_B
A (char) The name of the function and the attribute f(A)
G (char) The group-by clause G_1,G_2...G_n (attribute names)
P_S (char) The where clause of the relation (attribute names)

B. Control table T_D

For every base table, used to build aggregates there is one control table T_D that contains filters, limiting the number of aggregates that are dynamically materialized in its corresponding view V_{dA}. Filters stored in T_D define data ranges that are read (includes) and changed frequently (excludes).

T_D only stores the value characteristics of data areas but not the structure. Therefore, every include filter also references a structure (SID) persisted in T_S . The combined information is needed during materialization, as well as during view matching.

For simplicity reason it is assumed that where-clauses only consists of predicates that are linked with ANDs and that the predicates are all equality predicates. In [14] it is shown how the data structure must be adapted in order to support other predicate types, such as range and expression.

1) Data Schema: There is one dedicated control table for every base table in T_S. Its data schema depends on the number and types of the P_S attributes of the corresponding structures.

FID (int) Primary key - filter id
SID (int) Secondary key, referencing a structure in T_S
SIGN (bit) (I/E) Sign for (I)nclude and (E)xclude filters
FUSE (datetime) A timestamp when filter was first used
LUSE (datetime) A timestamp when filter was last used
EXEC (int) Number of executions of the filter
COST (int) Time used by database cursor for parsing, executing, fetching data of queries referencing the filter
MAT (boolean) A control sign that marks a filter validity
P_{SV} (as original) Filter attributes, store the values of the selection predicates (parameters), derived from P_S in T_S.

2) Hot read data: The runtime manager tracks the usage (number of executions) and cost (elapsed time) for every aggregate query that matches an existing include-filter. Additionally, every aggregate query that does not match an existing filter is captured by the runtime manager as a new include filter in T_D. This filter is saved as invalid until it gets materialized during the update phase.

3) Hot change data: Changes to base table data, such as inserts, updates, and deletes can invalidate materialized aggregates. In order to minimize the aggregates maintenance effort, aggregates based on frequently changing base data are excluded from materialization. Typically, the *hot change* data area shifts over time. Once the data ages, the frequency of change transactions is reduced. In our framework, every change event is recorded by the runtime manager as exclude filter(s). First of all, it must be determined how a change statement can effect an existing aggregate. Changes to attributes that are not part of the view structure V_{dA} can be neglected because they neither change the value of the aggregated attributes, nor does it change the grouping attributes. Changes to a grouping attribute always effect aggregates of two groups: the group of aggregates with the old data and the new one. Inserts and deletes on the other hand can always effect aggregates and results in the creation of an exclude filter. Secondly, the data area that is changed must be identified when selected with a primary key, updates and deletes effect a single row, and when selected with a non-unique attribute they effect a range of rows. In order to make that comparable with the existing include filters (hot read data), every change must be *normalized* to the filter-attributes, as defined in P_S.

4) Data Hotspots: Hotspots are data areas in a base table that are frequently used for aggregations but do not change frequently. Those hotspots change over time and adapt to the workload. The example in Table II shows how hotspots are identified in T_D. The values are extracted from three different aggregate queries (see SID column). SID1 selects its aggregates using attributes Product and Quarter. SID2 uses Product and Month and SID3 aggregates just over the parameter Month. The exclude filters are independent from the SID. The exclude filter FID7 for example could be derived from an insert of Product A in Quarter 3 and Month 8. Because of this, filter FID1 and FID5 are no hotspot areas anymore, even though they have been used to select aggregates. The same happens with FID6 because of FID9. FID9 means that there was a change in Month 7 and therefore aggregates of that selection do not qualify for materialization.

C. Dynamically materialized view V_{dA}

While the structure and filter control tables are maintained *online* during runtime, DMAVs are created and materialized only during the update phase. For every aggregate structure there is one V_{dA} defined and materialized. The algorithm that materializes the views is based on the view structure derived from T_S and the filters provided by T_D. It is important to note that the aggregates of V_{dA} are not only grouped by the G_i attributes of the structure but also by the P_i attributes.

TABLE II
HOTSPOT EXAMPLE OF T_D

FID	SID	SIGN	Product	Quarter	MONTH
1	1	I	A	3	
2	1	I	A	2	
3	1	I	A	1	
4	2	I	A		7
5	2	I	A		8
6	3	I			7
7		E	A	3	8
8		E	B	3	8
9		E	B	3	7

Aggregates are not specific to a certain filter, but comply to all filters. In fact, they can be used among different aggregate queries and different (overlapping) data selections. On the condition that there is a valid, matching filter, different aggregate queries (with different parameter values) can select directly on the aggregated view V_{dA}, instead of the base view. This allows the reuse of aggregates that have overlapping filters V_B.

1) Data Schema: The data schema of an aggregate view consists of (a) column(s) for the aggregated value A_i, (b) the group-bys G_i and (c) the selection attributes P_i.

D. Example for dynamic view materialization

The following example will show two aggregate queries (Q1 and Q2) as well as two change queries (U1 and I1) that represent a mixed workload. Based on a simple table schema, consisting of three tables (one fact table , two dimension tables) it is explained, how the query structure and its read and changed data ranges (includes and excludes) are extracted into the structure table T_S and a control table T_D. The data schema is as follows:

Product (pid, pname, pcategory)
DateH (did, dyear, dquarter, dmonth)
Sales (pid, did, rev)

Suppose Q1 and Q2 (see Listing 1 and 2) are two parameterized SPJG (Select, project, join and group-by) aggregate queries from a given mixed workload. In the first step of the analysis, the base table T_B:(Sales), the view predicate P_V:(Sales.pid = Product.did AND Sales.did = DateH.did) as well as the aggregate function and field A:(sum(Sales.rev)) are identified and written to the T_S table (as illustrated in Figure 3). For both queries they are the same. However their selection-predicates P_S are different; consequently, two separate structures have to be created. In Table III, Q_1 has SID=1 and Q_2 has SID=2. Their grouping attributes are written to G.

By collecting that information, the data structures of the materialized views can be defined as:

$\mathbf{V_{dA1}}$ (SUM(rev), pcategory, dyear dmonth)
$\mathbf{V_{dA2}}$ (SUM(rev), pname, dyear, dquarter)

TABLE III
STRUCTURE CONTROL TABLE T_S

SID	T_B	P_V	A	G	P_S
1	Sales	Sales.pid = Product.did AND Sales.did = DateH.did	SUM(rev)	dmonth	EQ(pcategory)-&&-EQ(dyear)-&&-EQ(dmonth)
2	Sales	Sales.pid = Product.did AND Sales.did = DateH.did	SUM(rev)	dquarter-pname	EQ(pname)-&&-EQ(dyear)

TABLE IV
DATA CONTROL TABLE T_D FOR TABLE SALES: TD_SALES

FID	SID	SIGN	FUSE & LUSE	EXEC	COST	MAT	PNAME	PCATEGORY	DYEAR	DMONTH
1	1	I				true/false		2	2011	3
2	1	I				true/false		4	2011	3
3	2	I				true/false	’A’		2011	
3	2	I				true/false	’A’		2010	
4	-	E				true/false	’B’	1	2010	9
4	-	E				true/false	’A’	2	2010	9
4	-	E				true/false	’A’	1	2011	12

Listing 1. Aggregate Query Q_1

```
SELECT d.dmonth, SUM(s.rev)
FROM Sales AS s, Product AS p, DateH AS d
WHERE s.pid = p.pid
  AND s.did = d.did
  AND (p.cat = 2 OR p.cat = 4)
  AND d.dyear = 2011
  AND d.dmonth = 3
GROUP BY d.dmonth
```

Listing 2. Aggregate Query Q_2

```
SELECT d.dquarter, p.pname, SUM(s.rev)
FROM Sales AS s, Product AS p, DateH AS d
WHERE s.pid = p.pid
  AND s.did = d.did
  AND d.dyear = 2011
  AND p.pname = ’A’
GROUP BY d.dquarter, p.pname
```

Having captured the structure of the aggregate query in T_S, the control table T_D is created. As illustrated in Table IV, the distinct set of attributes (A_i, G_i and P_{Si} of both structures SID= 1 and SID= 2) defines its data structure. Afterwards the values of the selection predicates P_{SV} (parameters) are inserted as include filters FID1-3 in Table III.

Listing 3. Change Query I_1

```
INSERT INTO Sales (pid, did, rev)
     VALUES (2,34,898.99)
```

Listing 4. Change Query U_1

```
UPDATE Sales
   SET rev = 99.99
 WHERE
1st Option: sid = 55
2nd Option: p.pid=’1’ and d.dyear=’2011
```

Listing 3 and 4 are two changing queries: an insert and an update statement with two typical selection predicates. As all their selection predicates cannot be captured in T_D, they need to be normalized, as shown in Listing 5 and 6, and described in Section IV-C.

Listing 5. Normalized Exclude Query I1 language

```
INSERT into control_table
SELECT p.pname, p.pcategory, d.dyear, d.dmonth
FROM Sales s, Product p, DateH d
WHERE s.pid = p.pid AND s.did = d.did AND s.sid = <id>
GROUP BY ...
```

Listing 6. Normalized Exclude Query U1

```
INSERT into control_table
SELECT p.pname, p.pcategory, d.dyear, d.dmonth
FROM Sales s, Product p, DateH d
WHERE s.pid = p.pid AND s.did = d.did AND
1st Option: s.sid = 55
2nd Option: p.pid=’1’ AND d.dyear=’2011’
GROUP BY ...
```

E. View maintenance and materialization V_{dA}

1) Maintenance: Compared to traditional, fully materialized views, dynamic materialization using the control table T_S allows for more efficient maintenance as only the hotspot parts of the view are materialized. Further, as aggregates of base data that is subject to changes is excluded, re-materialization is kept to a minimum.

Many different incremental view maintenance algorithms that compute changes of the base relation to the corresponding views [18] have been discussed in research. The concept used in this paper is built upon the control table T_D. Before a view is materialized, every filter is marked as invalid (mat = false). This status information is also used during view matching in Section IV-F. Aggregate queries that match an invalid filter cannot use the aggregate view, but must be answered by the

TABLE V
DYNAMIC, MATERIALIZED VIEW V_{DA2} FOR STRUCTURE SID 2

REV	PNAME	DYEAR	DQUARTER
2499.12	A	2010	1
3189,81	A	2010	2
3747.15	A	2010	3
1806.07	A	2010	4

base relation. This is true until the view is re-materialized during update phase where all invalid filters are set valid again.

2) Materialization: Let V_B denote a query expression on a base table T_B joining other tables using a predicate P_V. V_{Bi} may be referred to as a base view, defined by the structure SID=i in T_S. A_i shall be its aggregated function on an attribute of V_{Bi}. G_i.* are all of its grouping attributes and P_i.* are the parameters of the where-predicate. For each structure (SID) a dynamic materialized view V_{dAi} is defined. For each V_{Bi} the materialization is controlled by the control table T_D, its own include-control-predicate $P_{ICi}(V_{Bi},T_D)$ and an exclude-control-predicate $P_{EC}(V_{Bi},T_D)$. The algorithm in Listing 7 shows the generic definition of a materialized view. The exists- and not-exists-clause in the definition confine the rows, actually materialized in V_{dAi}, to those satisfying the control predicates $P_{ICi}(V_{Bi},T_D)$ and not to those of $P_{EC}(V_{Bi},T_D)$.

Listing 7. Algorithm for Materialization of $_{dA}$

```
 FOR ALL structures as $i$ in $T_S$ DO
     CREATE VIEW AS
         SELECT $A_i$, $G_i$.*, $P_i$.* FROM $V_{Bi}$
         WHERE
         EXISTS
                 SELECT 1 FROM $T_D$ WHERE $P_{ICi}(V_{Bi},T_D)$)
         AND NOT EXISTS
                 SELECT 1 FROM $T_D$ WHERE $P_{EC}(V_{Bi},T_D)$))
         GROUP BY $G_i$.*, $P_i$
 END FOR
```

In Listing 8 the view definition for SID=2, V_{dA2} is shown. Td_Sales shall be the control table T_D, holding all include and exclude filters. However, most interestingly are the include and exclude predicate controls ($P_{ICi}(V_{Bi},T_D)$, $P_{EC}(V_{Bi},T_D)$) that ensure that only aggregates over the hotspot data areas are materialized.

Listing 8. Materialized View V_{dA} for SID=2

```
SELECT SUM(s.rev), $G_i$.*, $P_i$.*
FROM $V_{Bi}$ AS bV
WHERE EXISTS (
      SELECT *
      FROM Td_Sales f
      WHERE  bV.pcategory = f.pcategory AND bV.pname = f.pname AND
             bV.dyear = f.dyear AND f.sign = I AND f.sid = 1 AND f.
             mat = false) --includes
AND NOT EXISTS (
      SELECT *
      FROM Td_Sales f
      WHERE  bV.pname = f.pname AND bV.ddyear = f.dyear AND f.sign
             = E) --excludes
GROUP BY $G_i$.*,$P_i$.*
```

3) Aggregate granularity: The granularity of an aggregated result set depends on the number and cardinality of its grouping attributes. Aggregates with several group-by attributes have the advantage that they can be reused to calculate coarse-grained ones. On the other hand this increases the size of the result set. This trade-off is considered during cost ranking in Section V-A that calculates the profit of an filter depending on the size of the result set. In the proposed concept, the granularity of the materialized view V_{dA} is also determined by the number and cardinality of the predicate attributes of the base relation. As the aggregates in V_{dA} are not specific for one parameter selection (filter) but should be reusable among overlapping selections (filters), of the same structure, the aggregates are additionally grouped by predicate attributes. This way, different aggregates with different groupings can be selected on the same materialized view.

F. View matching

The question whether an aggregate query or a sub query can be answered by a view (V_{dA}), is a well-known view matching problem. In [12] Goldstein et al. already looked at SPJ queries having an aggregate function followed by a group-by operation. Still, there is one major aspect that makes view matching for DMAVs different to other algorithms: query containment of the aggregate query in the view cannot be tested before execution time. As outlined in [14], the part of view matching that checks if the parameters of a query match a valid include-filter and not an exclude-filter in T_D has to be postponed to execution time. This is being done using a guard condition as described in Listing 9.

Listing 9. Guard Condition

```
EXISTS(SELECT 1 FROM $T_d$ WHERE SID=i AND MAT=true AND SIGN='I' AND
     hotkeys = @parameters)
NOT EXISTS(<exclude-filter>)
```

A guard condition is only executed when there is a valid structure for a query. If the guard condition is positive it is served from V_{dA}, otherwise it must be calculated from the base relation. Besides, aggregate queries can be treated as a SPJ query followed by a group-by. Accordingly, as described in [12] an aggregation query can be computed from a view V_{dA} if there is a structure in T_S that meets the following requirements.

1) All columns required by compensating predicates (if any) are available in the view output.
2) The view contains no aggregation or is less aggregated than the query, i.e., the groups formed by the query can be computed by further aggregation of groups output by the view.
3) All columns required to perform further grouping (if necessary) are available in the view output.
4) All columns required to compute output expressions are available in the view output.

All of these requirements can be satisfied by using the structure control table T_S. They can already be checked during optimization time. Then during execution time the SID of a query is known and the view matching can be limited to the guard condition check.

V. Control Table Management & Optimization

Both tables, T_D and T_S are managed during the online phase and optimized during update phase. During the online phase, the goal is to answer as many queries as possible from the view, because they will be answered a lot faster from V_{dA} than from the base relation. At the same time the runtime manager tracks new queries, usage/ hit statistics, and change events. This allows adapting to new query patterns as well as efficiently utilize the system resources during the update and optimization phase. During this phase, new filters in T_D and structures in T_S are materialized while invalidated filters are refreshed and validated again. Before that the control tables are optimized. The reason for optimization is to materialize only relevant aggregates, reduce the overhead of different aggregate structures and to anticipate future selections and improve the hit ratio. Therefore:

- Every filter is ranked based on a cost function,
- Different query structures of one base table are scanned, harmonized, and merged, and
- Future selection parameters are anticipated and added as filters to T_D.

A. Cost ranking

Every filter that is stored in a control table has a certain cost in terms of storage and materialization resources. Whether a new filter that is captured during online phase replaces existing ones shall depend on the calculated profit that can be compared amongst each other. Following are some metrics that are available for each filter. They are either tracked by the runtime manager or calculated.

- c_i: Sum of time used by the database cursor for parsing, executing, fetching data of all queries that reference the same filter (without materialization)
- FU_i: Time stamp when the filter was used the first time
- LU_i: Time stamp when the filter was used the last time
- s_i: Max. granularity of a view V_{dA}
- e_i: Number of hits (see:EXEC in T_D)

The goal of cost ranking is to have a number that allows the comparison of materialized and un-materialized filters and structures. Based on that number it can be decided which filters to admit to V_{dA} and which ones to replace.

$$profit(i) = \frac{h_i \cdot c_i}{s_i} - (c_i \cdot MAT) \tag{1}$$

$$h_i = \frac{e_i}{t - FU_i} \tag{2}$$

$$s_i = \prod_{j=1}^{n} AG_j \tag{3}$$

$$profit(SID) = \sum_{i=1}^{n} profit(FID_i) \tag{4}$$

In (1) the profit of an include-filter is calculated based on the average hit rate of the filter h_i, the cost of a filter c_i to be retrieved without materialization and the product of the attribute granularity of all view attributes. In order to compare materialized and non-materialized filters, the cost of a onetime materialization is subtracted at the end.

The average hit rate h_i of filter i (2) tells how many times a filter has been hit in average since it was first screened. Including the current time t, is guaranteed that aging of filters is considered. A filter, hit ten times in the last hour has a higher hit rate than a filter, hit ten times in the last two hours. (3) S_i is the maximum possible granularity of the view V_{dA} as defined by its structure. Therefore the cardinality of each attribute must be multiplied. The profit of a structure (4) is calculated by the sum of all filters that reference the structure.

$$relev(i) = \frac{h_i}{t - LU_i} \tag{5}$$

Exclude filters are ranked on their time dependent relevance (5). The overall average hit ratio of the exclude filter divided by its actuality (last time used).

B. Structure harmonization

For each aggregate structure, there is one view V_{dA} defined. They cannot serve aggregate queries with different structures as they have a different predicates clause, different predicate attributes as well as different aggregate granularity. However, if different predicate structures could be harmonized to one structure, this would reduce the number of required aggregates, and improve reuse of materialized aggregates. The example in Listing 10 shows two aggregates (see line 1 and 2). Both queries require their own structure and filters. Accordingly, there would be two materialized views.

In the mentioned example there is one hierarchical dimension: Date(did-year-month-day). The base table is joined with that table on the PK attribute "did". Thus, aggregates can be selected using all attributes of Date. The idea of structure harmonization is to consolidate the selection predicates of one dimension to its unique (composite) key. As can be seen in line 4 of Listing 10, the real selection of data is swapped out to a sub query wrapped with an IN operator. With the unique dimension key "did" every selection in dimension "Date" can be expressed without overlapping sets.

Listing 10. Predicate harmonization

```
WHERE did=d.did AND d.day=x                        1
WHERE did=d.did AND d.year=y AND d.day=z           2
                                                   3
WHERE did IN (Select did from Date WHERE           4
1st Option: d.year=y and d.day=z                   5
2nd Option: d.month=x )                            6
```

Rewriting aggregate queries into such a harmonized form would also simplify the data schema of control table T_D to one attribute "did". The number of filters, however, would increase significantly: E.g.: Assumed that there is one unique key per day, a filter of type: year=2012 would be changed to 365 filters of type "did". Instead, if the key was based on a chronological interval scale, a range control filter could express year=2012 as did=from:1-to:365.

Structure harmonization/ replacement can only be done between attributes of the same dimension table. The new attribute must be on a lower level than the old one. Additionally, only, if there is an overlapping free, n:1 relation between those attribute this harmonization works.

As a downside of this concept, materialized aggregates can become extremely fine grained. Even though the number is filtered by the control table, grouping by a unique dimension key "did" in V_{dA} results in very high numbers of results. At this point, it might happen that the results in views V_{dA} are not aggregated anymore but materialize the base relation.

C. Variable distribution

As there is a big training set of recorded selection parameters in T_S one can check for significant statistical distributions of the single attributes. Those selection characteristics could be used to automatically adjust filters and optimize the materialized view. A continuous distributed variable X (attribute) that has a similar probability for every captured value characteristic should not be limited during materialization to certain values. It is probable that the user might select another value next time, as there is no accumulation of value characteristics. For such attributes existing filters can be replaced so that aggregates are not filtered on that attribute anymore.

D. Time correlation

Often, there are dependencies between time and certain selection parameters. Products might be relevant only in certain seasons, account balances might only be interesting for the last twelve months and so forth. Finding those correlation can be done based on the captured filters in T_D. Not only include-filters could be anticipated in order to improve hit ration on V_{dA}, but also exclude-filters. Based on such knowledge, exclude filters can better adapt to hot change data as it changes over time.

VI. Experiment & Discussion

To evaluate our proposed concept, we have tested it rudimentarily with a mixed workload benchmark. The chosen benchmark [5] was setup on a MySQL database and executed twice with the same settings. The share of read-only OLTP queries was set to 95% and the share of OLAP queries for the mixed client was set to 40%.

In the first run, the benchmark performed all queries with the original schema. In the second run, the OLAP queries where optimized so that they did not accessed the base table anymore, but a DMAV V_{dA} instead. There were four different analytical queries with aggregate functions, two of them had additional sub queries with aggregate function. Hence, there were six different structures and V_{dA} views that the queries could use to select and aggregate. Every run was executed for 15min, plus 2min warm-up time that was not counted. During each run, database performance was monitored using MySQL Enterprise Monitor and the general db log. Afterwards the general log was analyzed using the Analysis Framework III.

Compared with the default benchmark, during the optimized run, the database could serve eight-times more select statements. Hence, the overall output performance increased, also for the OLTP queries. That was because the runtime cost (time in ms) for the OLAP queries was reduced by up to 99,96% each. Obviously, this comparison can only show relative improvements. Performed on a real application, the number of aggregate queries would be even higher. Besides, DMAVs would put an overhead on the database engine in terms of storage consumption and runtime performance. This was not considered in the experiment. Still, it became obvious that the performance can be improved by using DMAVs.

VII. Conclusion & Future Work

In this paper we presented *dynamic materialized aggregate views* (DMAV), a concept to selectively materialize arbitrary aggregate queries in a mixed workload environment. Built upon a central dictionary structure that holds generic aggregate structures and corresponding hot spot tables, it was shown how the execution of analytical queries in transactional databases can be improved. Intelligent selection and materialization of hot aggregates promises to be a way to handle such workload demands in future applications. Even though major parts of the concept can be transferred to asynchronous update phases or can be done during optimization phase it still puts an overhead on the database. Therefore, focus of future research addresses the problem of how and when to integrate the required steps of DMAV into query execution of a database.

References

[1] S. Chaudhuri and K. Shim, "Optimizing queries with aggregate views," *EDBT*, 1996.

[2] Y. Kotidis and N. Roussopoulos, "DynaMat: a dynamic view management system for data warehouses," *ACM SIGMOD Record*, 1999.

[3] M. Grund, J. Krüger, and H. Plattner, "HYRISE: a main memory hybrid storage engine," *Proceedings of the PVLDB*, pp. 105–116, 2010.

[4] V. Sikka, F. Färber, W. Lehner, T. Peh, and C. Bornhövd, "Efficient Transaction Processing in SAP HANA Database – The End of a Column Store Myth," *SIGMOD*, pp. 731–741, 2012.

[5] A. Bog and J. Kruger, "A Composite Benchmark for Online Transaction Processing and Operational Reporting," *BIRTE*, 2008.

[6] R. Cole, F. Funke, L. Giakoumakis, W. Guy, A. Kemper, S. Krompass, H. Kuno, R. Nambiar, T. Neumann, M. Poess, and Others, "The mixed workload CH-benCHmark," in *Proceedings of the Fourth International Workshop on Testing Database Systems*. ACM, 2011, p. 8.

[7] C. Tinnefeld, S. Müller, H. Kaltegärtner, S. Hillig, L. Butzmann, D. Eickhoff, S. Klkauck, D. Taschik, B. Wagner, O. Xylander, A. Zeier, H. Plattner, and C. Tosun, "Available-To-Promise on an In-Memory Column Store," *BTW*, pp. 667–686, 2011.

[8] F. Funke, A. Kemper, and T. Neumann, "Compacting Transactional Data in Hybrid OLTP & OLAP Databases," *VLDB*, 2012.

[9] P. Larson and H. Z. Yang, "Computing Queries from Derived Relations," *VLDB*, 1985.

[10] H. Z. Yang and P.-A. Larson, "Query transformation for PSJ-queries," *VLDB*, pp. 245–254, 1987.

[11] R. Bello, K. Dias, and A. Downing, "Materialized views in Oracle," *VLDB*, 1998.

[12] J. Goldstein and P.-a. k. Larson, "Optimizing queries using materialized views: a practical, scalable solution," *ACM SIGMOD Record*, vol. 30, no. 2, pp. 331–342, 2001.

[13] A. Halevy, "Answering queries using views: A survey," *VLDB*, no. 1999, pp. 270–294, 2001.

[14] J. Zhou, P.-A. Larson, J. Goldstein, and L. Ding, "Dynamic Materialized Views," *ICDE*, pp. 526–535, 2007.

[15] P. M. Deshpande and J. F. Naughton, "Aggregate Aware Caching for Multi-Dimensional Queries," *EDBT*, pp. 167–182, 2000.

[16] P. M. Deshpande, K. Ramasamy, A. Shukla, and J. F. Naughton, "Caching multidimensional queries using chunks," *SIGMOD*, pp. 259–270, 1998.

[17] P. Scheuermann, "WATCHMAN: A Data Warehouse Manager Intelligent Cache," *VLDB*, 1996.

[18] A. Gupta, "Maintaining views incrementally," *ACM SIGMOD Record*, 1993.

About Summarization in Large Fuzzy Databases

Ines BenAli-Sougui
Dept. TIC
Université Tunis El Manar
Ecole Nationale d'Ingénieurs de Tunis
Tunisia
ines.benali@gmail.com

Minyar Sassi-Hidri
Dept. TIC
Université Tunis El Manar
Ecole Nationale d'Ingénieurs de Tunis
Tunisia
minyar.sassi@enit.rnu.tn

Amel Grissa-Touzi
Dept. TIC
Université Tunis El Manar
Ecole Nationale d'Ingénieurs de Tunis
Tunisia
amel.touzi@enit.rnu.tn

***Abstract*—Moved by the need increased for modeling of the fuzzy data, the success of the systems of exact generation of summary of data, we propose in this paper a new approach of generation of summary from fuzzy data called "Fuzzy-SaintEtiQ". This approach is an extension of the SaintEtiQ model to support the fuzzy data. We prove that our approach presents the following optimizations: 1) the minimization of the expert risk, 2) the construction of a more detailed and more precise summaries hierarchy, and 3) the co-operation with the user by giving him fuzzy summaries in different hierarchical levels.**

Keywords-Fuzzy DB; Fuzzy SQL; FCM; FCA; Concept Summary.

I. INTRODUCTION

In the field of the Databases (DB), volumes of the data reached today make necessary a better exploitation of the data.

Several solutions have been proposed to solve this problem and to contribute in database summarization. However, to support massive data evolutionary, formal approaches have been proposed to surround this problem [1][2][3][4].

Several methods of DB summarization have been proposed such as statistical approaches, classification and conceptual classification. Among these data summarization methods, one of the most close to our research tasks, we distinguish the system SaintEtiQ [1] that is inspired primarily by the approach of conceptual classification. This system makes it possible to generate a hierarchy of summaries allowing to cover parts of the database. In [5], we proposed a new system for optimizing the SaintEtiq summarization system. This approach is based on the combination of fuzzy logic, fuzzy clustering and Formal Concept Analysis (FCA).

Moreover, with the evolution of the data processing, the need of modeled fuzzy data has become a necessity for the user [6]. Indeed, in the real world, we are confronted more and more with the situation where applications need to manage fuzzy data and to make profit their users from flexible querying. We speak, then, about flexible querying and Fuzzy Databases (FDB) [6][7][8].

In this paper, we propose an extension of the SaintEtiQ summarization model for modeling fuzzy data. We prove that our approach presents some optimizations: 1) the minimization of the expert risk, 2) the construction of a more detailed and more precise summaries hierarchy, and 3) the co-operation with the user by giving him fuzzy summaries in different levels from the hierarchy. This approach is based on the combination of fuzzy logic, fuzzy clustering and Formal Concept Analysis. For the classification of these fuzzy data, we propose a new algorithm, called Fuzzy FCM. Fuzzy-FCM is an extension of FCM algorithm in order to support fuzzy data.

The rest of this paper is organized as follows: Section 2 presents an overview of some summarization model and the basic concepts of Fuzzy Databases. Section 3 presents an example of fuzzy data. Section 4 presents problems and limits of the existing summarization approach. Section 5 presents our proposed Fuzzy-SaintEtiq system. Section 6 presents a comparison between our summary model Fuzzy-SaintEtiq and others models. We finish this paper with a conclusion and a presentation of some future works.

II. BASIC CONCEPTS

In this section, we present an overview of some summarization model and the basic concepts of Fuzzy Databases.

A. *Overview of the SaintEtiQ summarization model*

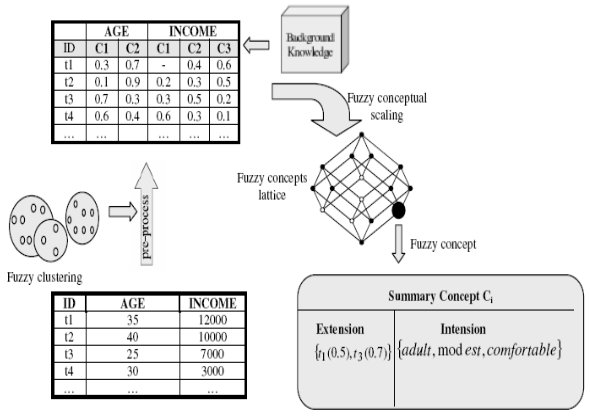
The SaintEtiQ model [1] aims at apprehending the information from a DB in a synthetic manner. This is done through linguistic summaries structured is a hierarchy. The model offers different granularities, i.e., levels of abstraction, over the data. The system architecture and the steps necessary to build a hierarchy are described below. With SaintEtiQ model, the summarization process can be divided into three major steps:

- **A Translation step**: this step allows the system to rewrite DB records in order to be processed by the mining algorithm. This translation step gives birth to candidate records, which are different representations of a single DB record, according to some background knowledge. Background knowledge's are fuzzy partitions defined over attribute domains. Each class of a partition is also labeled with a linguistic descriptor provided by the user or a domain expert. For instance, the fuzzy label young could belongs to a partition built over the domain of the attribute AGE.

- **A data mining step**: it considers the candidate records one at a time, and performs a scalable machine learning algorithm to extract knowledge. Obviously, the intensive use of background knowledge, which supports the translation step, avoids finding surprising knowledge nuggets.
- **A post processing step**: SaintEtiQ model tries to define summaries at different level of granularity. The post-processing step consists in organizing the extracted summaries into a hierarchy, such that the most general summary is placed at the root of the tree, and the most specific summaries are the leaves.

B. *Overview of FCA-based Summary*

In [5], we proposed to extend the SaintEtiQ summarization model [1] by introducing some optimization processes including: i) minimization of the expert risks domain, ii) building of the summary hierarchy from DB records, and iii) cooperation with the user by giving him summaries in different hierarchy levels. With our model, the summarization process can be divided into two major phases as shown on Figure 1.



	AGE		INCOME		
ID	C1	C2	C1	C2	C3
t1	0.3	0.7	-	0.4	0.6
t2	0.1	0.9	0.2	0.3	0.5
t3	0.7	0.3	0.3	0.5	0.2
t4	0.6	0.4	0.6	0.3	0.1
…	…		…	…	…

ID	AGE	INCOME
t1	35	12000
t2	40	10000
t3	25	7000
t4	30	3000
…	…	…

Summary Concept C_i

Extension	Intension
$\{t_1(0.5), t_3(0.7)\}$	$\{adult, \text{mod}\,est, comfortable\}$

Figure 1. The Overall process of proposed FCA-based summary model [5].

C. *Fuzzy Databases*

In this section, we present the basic concepts of Fuzzy Databases.

A Fuzzy Databases (FDB) is an extension of the relational DB. This extension introduces fuzzy predicates under shapes of linguistic expressions that, at the time of a flexible querying, permits to have a range of answers (each one with a membership degree) in order to offer to the user all intermediate variations between the completely satisfactory answers and those completely dissatisfactory [8].

The FDB models are considered in a very simple shape and consist in adding a degree, usually in the interval [0,1], to every tuple.

It allows maintaining the homogeneity of the data in DB. The main models are those of Prade-Testemale[9], Umano-Fukami[10], Buckles-Petry[11], Zemankova-Kaendel[12] and GEFRED of Medina et al. [13].

This last model constitutes an eclectic synthesis of the various models published so far with the aim of dealing with the problem of representation and treatment of fuzzy information by using relational DB.

D. *The GEFRED Model*

The GEFRED model (GEneralised model Fuzzy heart Relational Database) has been proposed in 1994 by Medina et al. [13]. One of the major advantages of this model is that it consists of a general abstraction that allows for the use of various approaches, regardless of how different they might look. In fact, it is based on the generalized fuzzy domain and the generalized fuzzy relation, which include respectively classic domains and classic relations.

In order to model fuzzy attributes we distinguish between two classes of fuzzy attributes: Fuzzy attributes whose fuzzy values are fuzzy sets and fuzzy attributes whose values are fuzzy degrees [6][14].

Fuzzy Sets as Fuzzy Values: These fuzzy attributes may be classified in four data types. This classification is performed by considering the type of referential or underlying domain. In all of them the values Unknown, Undefined, and Null are included:

- **Fuzzy Attributes Type 1 (FTYPE1)**: These are attributes with “precise data”, classic or crisp (traditional, with no imprecision). However, they can have linguistic labels defined over them, which allow us to make the query conditions for these attributes more flexible.
- **Fuzzy Attributes Type 2 (FTYPE2)**: These attributes admit both crisp and fuzzy data, in the form of possibility distributions over an underlying ordered domain (fuzzy sets). It is an extension of the FTYPE1 that does, now, allow the storage of imprecise information.
- **Fuzzy Attributes Type 3 (FTYPE3)**: They are attributes over “data of discreet non-ordered dominion with analogy”. In these attributes some labels are defined (“blond”, “red”, “brown”, etc.) that are scalars with a similarity (or proximity) relationship defined over them, so that this relationship indicates to what extent each pair of labels be similar to each other.
- **Fuzzy Attributes Type 4 (FTYPE4)**: These attributes are defined in the same way as FTYPE3 attributes without the necessity of a similarity relationship to exist between the labels.

Fuzzy Degrees as Fuzzy Values: The domain of these degrees can be found in the interval [0,1], although other values are also permitted, such as a possibility distribution (usually over this unit interval) [13][14]. The meaning of these degrees is varied and depends on their use. The most important possible meanings of the degrees used by some authors are: Fulfillment degree, Uncertainty degree, possibility degree and Importance degree.

E. The FSQL language

The Fuzzy SQL (FSQL) language is an authentic extension of SQL language to model fuzzy queries. It means that all the valid statements in SQL are also valid in FSQL [13][14].

III. EXAMPLE OF FUZZY DATA

In this example, we want to model an employee described by the following information: his *Id* (identifier), his name, his surname, his address, his *Age*, his *Salary*, and his productivity. Attributes *Age*, *Salary* and *Productivity* are described as follows:

- The attribute *Age***,** presented in Figure 2, has the linguistic labels *Young*, *Adult* and *Old*, defined on the trapezoidal possibility distributions as following: *Young(18, 22, 30, 35), Adult(25, 32, 45, 50), Old(50, 55, 62, 70).* An approximate value has a margin of 5. The minimal value to consider two values of this attribute as completely different is of 10.

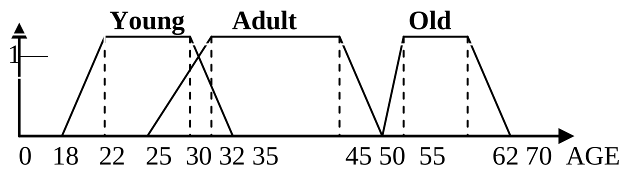


Figure 2. Definition of labels on Age attribute.

- The attribute salary, presented in Figure 3, has the linguistic labels *Low*, *Medium* and *High*, defined on the trapezoidal possibility distributions as following: *Low(50,80,120,180), Medium(150,300,400,550), High(400, 600,800,1000)*. An approximate value has a margin of 10 and the minimal value to consider two values of this attribute as completely different is of 50.

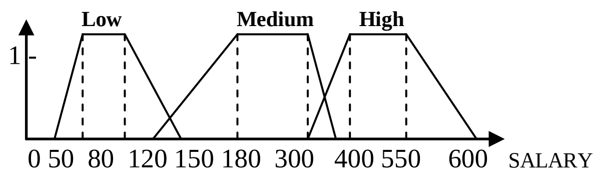


Figure 3. Definition of salary labels.

- The attribute *productivity*, presented in Table I, has the linguistic labels *Bad*, *Regular* and *Good*. In this situation, the data are not quantifiable, but present resemblances in their values. For example, the value *Regular* of the attribute “productivity” resembles to the value *Good* with a degree equal to 0.7.

TABLE I. RELATIONS OF SIMILARITY FOR THE VALUES OF THE PRODUCTIVITY ATTRIBUTE

Similarity degree	**BAD**	**REGULAR**	**GOOD**
BAD	1	0.3	0.2
REGULAR	0.3	1	0.7
GOOD	0.2	0.7	1

While applying the rules of Medina et al., we can say that the *age* attribute is FTYPE2 (5, 10) type, the attribute *salary* is FTYPE1 (10, 50) type and the attribute *productivity* is FTYPE3 (1) type.

An abstract representation of the schema of relation EMPLOYE will be as follows: (ID, NAME, SURNAME, ADDRESS, *AGE, SALARY*, *PRODUCTIVITY*). This description in FSQL script is presented in the Figure 4.

```
CREATE TABLE EMPLOYEE (
ID# VARCHAR(4) NOT NULL,
NAME VARCHAR(20) NOT NULL,
SURNAME VARCHAR(20) NOT NULL,
ADDRESS VARCHAR(40) NOT NULL,
AGE FTYPE2(5,10) NUMBER(3) DEFAULT UNKNOWN
NOT NULL,
SALARY FTYPE1(10,50) NUMBER(7) NOT NULL,
PRODUCTIVITY FTYPE3(1) NOT NULL,
PRIMARY KEY (ID#));
```

Figure 4. FSQL script.

A. Problems and Motivation

We present in the following table a synthesis of the existing summarization techniques.

As Table II depicts, these approaches are applicable only to simple data sets; they do not allow treating fuzzy data, describe with FSQL language, like linguistic labels (string), interval, and approximate values.

In this paper, we propose to define a new approach of summarization allowing treating as well the simple data set or the fuzzy data described with FSQL language.

TABLE II. COMPARATIVE STUDY OF SOME SUMMARIZATION TECHNIQUES

	Understandable	**Sampling**	**Data Nature**	**Huge data**	**Ratio with the original data**	**hierarchical levels**	**Reliability**	**Subject depending**	**Fuzzy DB**
Statistical Model	comprehensible	No	Numeric/ Nominal	No	Lost	No	High	Yes	No
Classification	No	Yes	Numeric	No	Kept	No	Low	No	No
Conceptual classification	Partially	Yes	Numeric	No	Kept	No	Means	No	No
SaintEtiQ	Yes	Yes	Numeric/ Nominal	Yes	Kept	Partially	Means	No	No
FCA-based Summary	Yes	Yes	Numeric	Yes	Kept	Yes	High	No	No

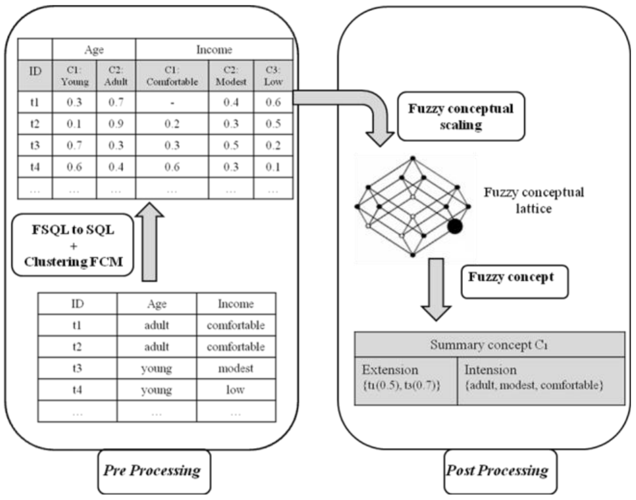
IV. MODEL DESCRIPTION

In this section, we present the architecture of the summarization model, the principal of summaries generation and the formal summary description.

A. *System architecture*

Our summary model takes the database records and provides knowledge.

Figure 5 gives the system architecture. The summarization act considered like a process of knowledge discovery from database, in the sense that it is organized according to two following principal steps.



	Age		Income		
ID	C1: Young	C2: Adult	C1: Comfortable	C2: Modest	C3: Low
t1	0.3	0.7	-	0.4	0.6
t2	0.1	0.9	0.2	0.3	0.5
t3	0.7	0.3	0.3	0.5	0.2
t4	0.6	0.4	0.6	0.3	0.1
…	…	…	…	…	…

ID	Age	Income
t1	adult	comfortable
t2	adult	comfortable
t3	young	modest
t4	young	low
…	…	…

Summary concept C1	
Extension {t1(0.5), t3(0.7)}	Intension {adult, modest, comfortable}

Figure 5. The Overall process of Fuzzy-SaintEtiq.

1) *The preprocessing step*

This step organizes the database records in homogeneous clusters having common properties. This step gives a certain number of clusters for each attribute. Each tuple has values in the interval [0..1] representing these membership degrees according the formed clusters. Linguistic labels, which are fuzzy partitions, will be assigned on attribute's domain.

For the classification on these fuzzy data, we propose a new algorithm, called Fuzzy FCM. Fuzzy-FCM is an extension of FCM algorithm in order to support different types of data represented by GEFRED model. Figure 6 shows the different steps of this algorithm.

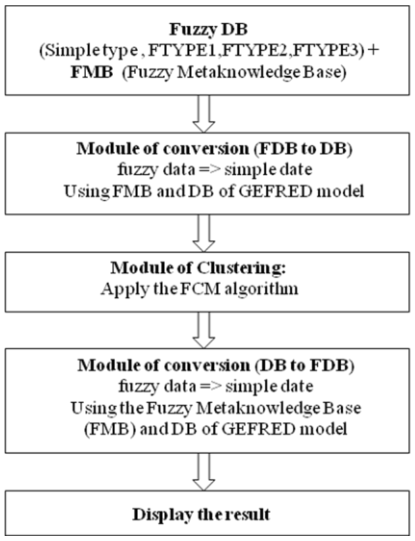

Fuzzy DB
(Simple type, FTYPE1,FTYPE2,FTYPE3) + FMB (Fuzzy Metaknowledge Base)
Module of conversion (FDB to DB)
fuzzy data => simple date
Using FMB and DB of GEFRED model
Module of Clustering:
Apply the FCM algorithm
Module of conversion (DB to FDB)
fuzzy data => simple date
Using the Fuzzy Metaknowledge Base (FMB) and DB of GEFRED model
Display the result

Figure 6. Principe of Fuzzy-FCM algorithm.

The Fuzzy-FCM algorithm allows the user to select attributes according to which he wants to carry out classification. This treatment gives a refined intermediate matrix only formed of the codes of the selected attributes.

Once the selection achieved, the FCM algorithm is applied on the refined table to get a matrix of adherence and a cut is exercised on this matrix of adherence to purify it by eliminating all values lower to the cut.

The main idea of the algorithm is to define an intermediate matrix to model fuzzy data. For this, we define the function $\mathscr{F}$ which permits the construction of this matrix. $\mathscr{F}$ is defined as follows:

Definition 1: Let E be the set of linguistic labels and C the set of numbers.

We define $\mathcal{F}$ as a function which for all e belonging to E, makes correspond a code c belonging to C the set of correspondence codes:

$$\mathcal{F}: \quad E \longrightarrow C$$
$$e \longrightarrow c = \text{Number of Attribute.Threshold}$$

Since the attributes of the type FTYPE1 do not authorize to store fuzzy values they undergo the same treatment as the simple data and thereafter the function $\mathcal{F}$ = id.

For the attributes of the type FTYPE 2 and FTYPE 3, the function $\mathcal{F}$ makes correspond to each linguistic label a code of the form NumberAttribut.Threshold.

We define the Threshold as being the minimal value to be able to consider two values as completely different.

Example: Let us consider the relational DB table Personal, represented in Table III, described by Id, Age, and Experience.

The attribute Age has the linguistic labels definite on the following trapezoidal distributions possibility : Young (18,22,30,35), Adult (25, 32,45,50), Old(50,55,62,70). The minimal value to consider two values of this attribute as completely different is 10.

The attribute Experience has the linguistic labels: Small(2,3,5,6), Good(5,7,10,12), Sufficient(7, 8,15,20), Large(12,15,50,50). These values depend on the numbers of years worked by an employee. The minimal value to consider two completely different Experiences is 5.

TABLE III. PERSONAL DB TABLE

Id	Age	Experience
001	Young	Good
002	Old	Small
003	Adult	Sufficient
004	Young	Large

While applying the rules of Medina et al., we can say that the AGE attributes and Experience attribute is FTYPE2. Thus, the correspondence table is presented by Table IV.

TABLE IV. TABLE OF CORRESPONDANCE

Id	Age	Experience
001	1.1	2.15
002	1.3	2.10
003	1.2	2.20
004	1.1	2.25

For the first attribute Age of this table, the choice of the number 1 translated the number of the attribute on which one works. Here, the Age attribute is the attribute number 1 and thereafter codes corresponding to the linguistic labels start all with 1. The minimal value to consider two values of this attribute as completely different is 10; we fix a step then = 10 in the choice of codes. For the second attribute experience, the choice of the number 2 translated the number of the attribute on which one works. Here, the attribute Experience is the attribute number 2 and thereafter codes corresponding to the linguistic labels start all with 2. The minimal value to consider two values of this attribute as completely different is 5; we fix a step then = 5 in the choice of codes. Moreover, the choice of these codes concord with the semantics of labels.

For example, the small label is nearer "semantically" to the good label than of the big label, thus we chose the codes according to this logic "of ascending order".

2) The post treatment step

This step takes into account the result of the fuzzy clustering on each attribute, visualizes by using the fuzzy concepts lattices. Then, it imbricates them in a fuzzy nested lattice.

Finally, it generalizes them in a fuzzy lattice associating all records in a simple and hierarchical structure. Each lattice node is a fuzzy concept which represents a concept summary.

This structure defines summaries at various hierarchical levels.

This step consists in organizing the summaries within a hierarchy such that the most general concept summary is placed at the root of the fuzzy lattice, and the most specific concept's summaries are the leaves.

This summary model corresponds to prototypical approaches since the intention of a concept summary present for each attribute the various possible values in the form of a fuzzy descriptors and the representativeness of these descriptors within the specified concept summary.

This model will be described formally in subsection C.

B. Principal of summaries generation

The summary model presented here is based on the fuzzy subsets theory with each one of its steps.

1) Generating attribute's clusters

For the generation of the clusters for each attribute, we carry out a fuzzy clustering while benefiting from fuzzy logic. This operation makes it possible to generate, for each attribute, a set of membership degrees. Each cluster of a partition is labeled by linguistic descriptor provided by a domain expert.

For example, the fuzzy label young belongs to a partition built on the domain of attribute AGE.

2) Building the summary hierarchy

After the generation of the clusters of each attribute, data are ready to be summarized. This operation is based on the fuzzy lattices notion.

This very simple sorting procedure gives us for each many-valued attribute the distribution of the objects in the line diagram of the chosen fuzzy scale. Usually, we are interested in the interaction between two or more fuzzy many-valued attributes. This interaction can be visualized using the so-called fuzzy nested line diagrams. It is used for visualizing larger fuzzy concept lattices, and combining fuzzy conceptual scales on-line.

Example: Table V presents the results of fuzzy clustering applied to AGE and INCOME attributes. For INCOME attribute, fuzzy clustering generates three clusters

(C1,C2 and C3). For AGE attribute, two clusters have been generated (C4 and C5).

TABLE V. FUZZY CONCEPTUAL SCALES FOR AGE AND INCOME ATTRIBUTES.

	Age		Income		
	C4	*C5*	*C1*	*C2*	*C3*
t1	0.5	0.4	0.4	0.5	0.5
t2	0.6	-	-	-	0.6
t3	-	-	-	0.7	-
t4	0.4	0.5	0.5	-	0.8
t5	0.4	0.4	0.4	0.6	-
t6	0.3	-	-	0.5	0.5

The minimal value (resp. maximal) of each cluster corresponds on the lower (resp. higher) interval terminal of the values of this last. Each cluster of a partition is labeled with a linguistic descriptor provided by the user or a domain expert.

For instance, the fuzzy labels young and adult could belong to a partition built over the domain of the attribute AGE.

Also, the fuzzy labels miserable, modest and comfortable could belong to a partition built over the domain of the attribute INCOME. So, the table VI can be rewrite as follows:

TABLE VI. FUZZY CONCEPTUAL SCALES FOR AGE AND INCOME ATTRIBUTES.

	Age		Income		
	Young C4	*Adult C5*	*Miserable C1*	*Modest C2*	*Comfortable C3*
t1	0.5	0.4	0.4	0.5	0.5
t2	0.6	-	-	-	0.6
t3	-	-	-	0.7	-
t4	0.4	0.5	0.5	-	0.8
t5	0.4	0.4	0.4	0.6	-
t6	0.3	-	-	0.5	0.5

The corresponding summary hierarchy is illustrated in Figure 7.

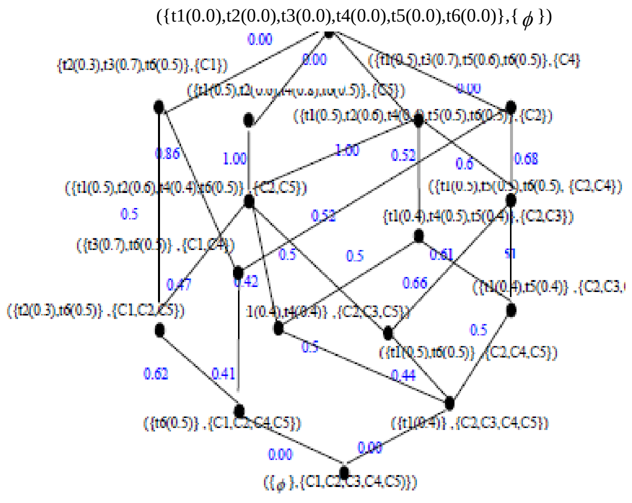

({t1(0.0),t2(0.0),t3(0.0),t4(0.0),t5(0.0),t6(0.0)},{ϕ})
0.00
{t2(0.3),t3(0.7),t6(0.5)},{C1})
({t1(0.5),t3(0.7),t5(0.6),t6(0.5)},{C4})
({t1(0.5),t2(0.6),t4(0.4),t5(0.5),t6(0.5)},{C2})
({t1(0.5),t2(0.6),t4(0.4),t6(0.5)},{C2,C5})
({t1(0.5),t5(0.5),t6(0.5), {C2,C4})
{t1(0.4),t4(0.5),t5(0.4)},{C2,C3})
({t3(0.7),t6(0.5)},{C1,C4})
({t2(0.3),t6(0.5)},{C1,C2,C5})
({t1(0.4),t4(0.4)},{C2,C3,C5})
({t1(0.5),t6(0.5)},{C2,C4,C5})
({t6(0.5)},{C1,C2,C4,C5})
({t1(0.4)},{C2,C3,C4,C5})
({ϕ},{C1,C2,C3,C4,C5})
0.86
1.00
0.52
0.68
0.5
0.47
0.42
0.62
0.41
0.44

Figure 7. A summary Hierarchy.

C. Formal representation of summaries

As shown in Figure 7, each concept summary can be viewed like an n-uplet of a relation R^* whose diagram is the same one as the origin relation R to summarize. Each concept summary z of the set of concept's summary Z is thus a description of a set of n-uplets of R, which jointly form his extension and which is noted by R_Z.

Definition 2. Concept summary: A concept summary is a couple $z=(R_Z,I_Z)$ in which R_Z is the subset of database records involved into the summarization, the extent, whereas the summarized description I_Z of these database records is the intent.

Each concept summary $z=(R_Z,I_Z)$ provides a synthetic view of a part of the database.

Thus, the root contains the summary of all the candidate records, whereas leaves represent only one combination of fuzzy linguistic labels over all the attributes.

Example: *z=({t1(0.5),t5(0.5),t6(0.5), {modest, young})*

Definition 3. Abstraction level: An abstraction level is an abstraction level is regarded as a level in the summary hierarchy generated.

Definition 4. Level: A level L of a summary hierarchy is a set of concept's summary z_k verifying the following property: the majors and the minors of z_k are at the same distance d.

Definition 5. Majors/Minors: Let $(E,\le_E)$ be an ordered set and S a subset of E. Major's elements (successors) and Minor's elements (predecessors) of S are defined by:

$$\text{Majors}(S)=\{x\in E\wedge\forall y\in S, y\le_E x\}$$
$$\text{Minors}(S)=\{x\in E\wedge\forall y\in S, x\le_E y\}$$

Considering the summary hierarchy in Figure 7, we can generate the following levels with the corresponding summaries:

Level 0 { z1=({t1(0.0),t2(0.0),t3(0.0),t4(0.0),t5(0.0),t6(0.0)},{ ϕ })

Level 1 {
z21= {t2(0.3),t3(0.7),t6(0.5)},{miserable})
z22= t1(0.5),t2(0.6),t4(0.8),t6(0.5)},{adult})
z23= ({t1(0.5),t2(0.6),t4(0.4),t5(0.5),t6(0.5)}, {modest})
z24= ({t1(0.5),t3(0.7),t5(0.6),t6(0.5)},{young})

Level 2 {
z31= ({t1(0.5),t2(0.6),t4(0.4),t6(0.5)},{modest, adult})
z32= ({t1(0.5),t5(0.5),t6(0.5),{modest, young})
z33= {t1(0.4),t4(0.5),t5(0.4)},{modest, comfortable})
z34= ({t3(0.7),t6(0.5)} ,{miserable, young})

Level 3 {
z41= ({t1(0.4),t5(0.4)},{modest,comfortable, young})
z42= ({t1(0.4),t4(0.4)},{modest,comfortable, adult})
z43= ({t2(0.3),t6(0.5)},{miserable, modest,adult})
z44= ({t1(0.5),t6(0.5)} ,{modest, young, adult})

Level 4 {
z51= ({t6(0.5)},{miserable, modest, young, adult})
z52= ({t1(0.4)},{modest,comfortable,young, adult})

Level 5 { z61= ({ ϕ },{miserable,modest,comfortable, young,adult})

Levels 0 and 5 are both the root and leaves concept summary.

A concept summary is defined in an extensional manner with a collection of candidate records $R_z = \{t_1, t_2, ..., t_N\}$.

Each t_i is associated to one primitive database records, i.e., an element of R.

Denote by $card(R_z) = \sum_{t \in R_z} w(t)$ the representativity of the concept summary z according to the primary database R. $|R_z|$ the number of candidate records in R_z.

D. About complexity

The space complexity, whatever the number of database records, is thus reduced to a constant value, i.e., about O(1). This characteristic is fundamental in the treatment of the large database in knowledge discovery. Temporal complexity includes the following costs:

- Construction of the attribute's clusters.
- Building the fuzzy lattice.

For cluster's construction, the complexity of fuzzy clustering algorithms is about O(NC^2), where N corresponds to database table records number and C is the maximum number of clusters.

For fuzzy lattice construction, temporal complexity of lattice construction algorithm is about O(N^2) .

V. Comparative study

In recent years, several methods of DB summarization have been proposed such as statistical approaches, classification and conceptual classification. Unfortunately, all these techniques cannot be applied to the large Fuzzy DB.

In this paper, we have proposed a new approach to linguistic summarization for fuzzy databases, called fuzzy-SaintEtiq. This approach is an extension of the SaintEtiQ model to support the fuzzy data.

Based on a hierarchical conceptual clustering algorithm, SaintEtiQ model builds a summary hierarchy from DB records. However, for building hierarchy, this model passes by a three steps which, after their study, each one can be optimized while keeping its hierarchical aspect.

- For the first step, the pre-processing, we have considered a fuzzy clustering allowing generating a membership matrix associating the DB records to generated clusters by means the membership degrees. In this context we have proposed an extension of FCM algorithm, called Fuzzy-FCM, in order to support different types of data represented by GEFRED model.
  This is a form of optimization as much in DB navigation as minimization of the domain expert risks.
- The second and the third steps have been associated together in order to generate the summary hierarchy. So, we have proposed to use fuzzy FCA in order to generate fuzzy hierarchy.
  This step presents an optimization form in building summaries. On the one hand, it cooperates with the user by giving him summaries in different hierarchy levels. In the other hand, it allows the calculation of different measures from possible evaluations.

Table VII gives a comparison between our summary model Fuzzy-SaintEtiq and the other models.

TABLE VII. COMPARATIVE STUDY OF SOME SUMMARIZATION TECHNIQUES

	Understandable	**Sampling**	**Data Nature**	**Huge data**	**Ratio with the original data**	**hierarchical levels**	**Reliability**	**Subject depending**	**Fuzzy DB**
Statistical Model	comprehensible	No	Numeric/ Nominal	No	Lost	No	High	Yes	No
Classification	No	Yes	Numeric	No	Kept	No	Low	No	No
Conceptual classification	Partially	Yes	Numeric	No	Kept	No	Means	No	No
SaintEtiQ	Yes	Yes	Numeric/ Nominal	Yes	Kept	Partially	Means	No	No
FCA-based Summary	Yes	Yes	Numeric	Yes	Kept	Yes	High	No	No
Fuzzy_SaintEtiq	Yes	Yes	Numeric	Yes	Kept	Yes	High	No	Yes

VI. CONCLUSION

With the increasing size of databases, the extraction of data summaries becomes more and more useful, thus, several methods of DB summarization have been proposed.

Unfortunately, all these techniques cannot be applied to the Fuzzy DB. In this paper, we proposed a new approach to linguistic summarization for fuzzy DB, called fuzzy-SaintEtiq. This approach is an extension of the SaintEtiQ model to support the fuzzy data represented by GEFRED model. Although this solution is based on the fuzzy model GEFRED, it can be applied to other fuzzy models.

To validate our approach, we currently plan to develop this approach with JAVA language.

As futures perspectives of this work, we mention essentially 1) to test our approach on the large fuzzy data set and 2) to describe a new approach for Knowledge Discovery in Fuzzy Databases (KDFDB) described with FSQL language. While basing on the summary hierarchy, generated by fuzzy-SaintEtiq, we proceed to discover the Knowledge in a hierarchical way. Thus, according to the degree of detail required by the user, this approach proposes a level of knowledge and different views of this knowledge.

REFERENCES

[1] G. Raschia and N. Mouaddib, “SaintEtiQ: A fuzzy set-based approach to database summarization,” Fuzzy Sets and Systems, vol. 129, no. 2, pp. 137–162, 2002.

[2] RR. Yager, “A new approach to the summarization of data. Information Sciences,” vol. 28(1), pp. 69–86, 1982.

[3] P. Bosc, O. Pivert, and L. Ughetto, “On data summaries based on gradual rules,” Proc. International Conference on Computational Intelligence, Theory and Applications: Fuzzy Days”, pp. 512–521, 1999.

[4] H.J. Lenz and A. Shoshani, “ Summarizability in OLAP and statistical databases,” Proc. International Conference on Scientific and Statistical Database Management, pp. 132–143, 1997.

[5] M. Sassi, A. Grissa-Touzi, H. Ounelli, and I. Aissa, “About Database Summarization,” International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 18(2), pp. 133-151, 2010.

[6] J.Galindo, A.Urrutia, and M. Piattini, Fuzzy databases: modeling, design and implementation. USA: Idea Group Publishing Hershey. 2006.

[7] M.A Ben Hassine, A. Grissa Touzi, J. Galindo, and H. Ounelli, “How to Achieve Fuzzy Relational Databases", in Handbook of Research on Fuzzy Information Processing in Databases”, Ed, Information Science Reference, pp. 351- 380, 2008.

[8] P. Bosc, L. Liétard, and O. Pivert, “Bases de données et Flexibilité : Les requêtes Graduelles,” Techniques et Sciences informatiques, vol. 7(3), pp. 355-378, 1998.

[9] H. Prade and C. Testemale, “Fuzzy Relational Databases: Representational issues and Reduction Using Similarity Measures,” *J. Am. Soc. Information Sciences*, vol. 38(2), pp. 118-126, 1987.

[10] M. Umano, S. Fukami, M. Mizumoto, and K. Tanaka, “Retrieval Processing from Fuzzy Databases,” Technical Reports of IECE of Japan, vol. 80(204), pp. 45-54, 1980.

[11] B. P. Buckles, and F. E. Petry, “A Fuzzy Representation of Data for Relational Databases,” Fuzzy Sets and Systems, vol. 7, pp. 213-226, 1982.

[12] M. Zemankova-Leech, and A. Kandel, “Implementing Imprecision in Information Systems,” Information Sciences, vol. 37, pp. 107-141, 1985.

[13] J.M. Medina, O. Pons, and M.A.Vila, “GEFRED. A Generalized Model of Fuzzy Relational DataBases”, Information Sciences, vol. 76(1-2), pp. 87-109, 1994.

[14] J.Galindo, “New characteristics in FSQL, a fuzzy SQL for fuzzy databases”. WSEAS Transactions on Information Science and Applications 2, vol. 2(2), pp. 161-169, 2005.

Provenance Policies for Subjective Filtering of the Aggregated Linked Data

Tomáš Knap
Department of Software Engineering
Charles University in Prague
Prague, Czech Republic
Email: tomas.knap@mff.cuni.cz

***Abstract*—As part of LOD2.eu project and OpenData.cz initiative, we are developing an ODCleanStore framework (1) enabling management of governmental linked data and (2) providing web applications with a possibility to consume cleaned and integrated governmental linked data; the provided data is accompanied with data provenance and a quality score based on a set of policies designed by the governmental domain experts. Nevertheless, these (objective) policies fail to express subjective quality of the data as perceived by various data consumers and different situations at their hand. In this paper, we describe how consumers can define their own situation-specific policies based on the idea of filtering certain data sources due to certain aspects in the data provenance records associated with these sources. In particular, we describe how these policies can be (1) constructed by data consumers and (2) applied as part of the data consumption process in ODCleanStore. We are persuaded that provenance policies are an important mechanism to address the subjective dimension of data quality.**

Keywords-provenance; provenance policies; linked data; linked open data; data aggregation; data quality

I. Introduction

Allover the world, governments are connecting to the uprising trend of publishing governmental data as open data [6]; open data is original non-aggregated machine readable data which is freely available to everyone, anytime, and for whatever purpose. As a result, citizens paying the government are able to see and analyze the performance of the government by observing the raw data or using third-party applications visualizing the data; companies can use the data to run their business.

Cannot we do more than just opening the data to simplify the data exploration and creation of applications on top of open data? If global identifiers were used in the form of HTTP URLs for the data exposed as open data, data could be published on these URLs and data consumers could then use the current Web infrastructure to obtain relevant information about any resource by simply inserting the HTTP URL of the resource to the browser. Furthermore, if the open data was represented as RDF (Resource Description Framework) [15] triples or quads (quads are RDF triples with a forth field representing the context – named graph [11] – to which the triple belongs), we could link data (e.g., the public contract) to other data (e.g., the supplier or price) and, thus, create a huge web of interconnected data. The idea described is precisely the idea of *linked data* [9].

The advent of linked data [9] accelerates the evolution of the Web into an exponentially growing information space (see the linked open data cloud [1]), where the unprecedented volume of data will offer information consumers a level of information integration and aggregation agility that has up to now not been possible. Consumers can now "mashup" and readily integrate information for use in a myriad of alternative end uses. Indiscriminate addition of information can, however, come with inherent problems, such as the provision of poor quality, inaccurate, irrelevant or fraudulent information. All will come with an associate cost of the data integration which will ultimately affect data consumer's benefit and linked data applications usage and uptake.

To overcome these issues, as part of the *OpenData.cz initiative* [4] and *LOD2 project* [2], we are developing the *ODCleanStore framework* [3] (1) enabling management of linked data – RDF data cleansing, linking, transforming, and quality assessing – and (2) providing consumers with a possibility to consume cleaned and integrated RDF data supplemented with data quality and provenance metadata.

The overall picture of ODCleanStore is depicted in Figure 1; data filtering module is depicted in red in Figure 1 to denote that it is not part of the current ODCleanStore state of the art and is discussed as one of the contributions further in this paper. ODCleanStore processes RDF *data feeds* and stores it to the *staging database*; feeds can be uploaded to the staging area by any third-party application registered to ODCleanStore. The RDF data feed is a set of named graphs including the RDF *data graph* (the main named graph of the feed) and graphs holding descriptive and provenance metadata. Based on the pipeline identifier within the feed's descriptive metadata, ODCleanStore engine launches the particular *data processing pipeline* containing an execution of the sequence of *data transformers (data processing units)*, which may normalize the data, deduplicate the data against the *raw data mart* or link the data to the data in the raw data mart, assess the quality of the data, or execute an arbitrary data transformation. After executing all the transformers on the given pipeline, the data feed is stored to the raw data mart together with any auxiliary data

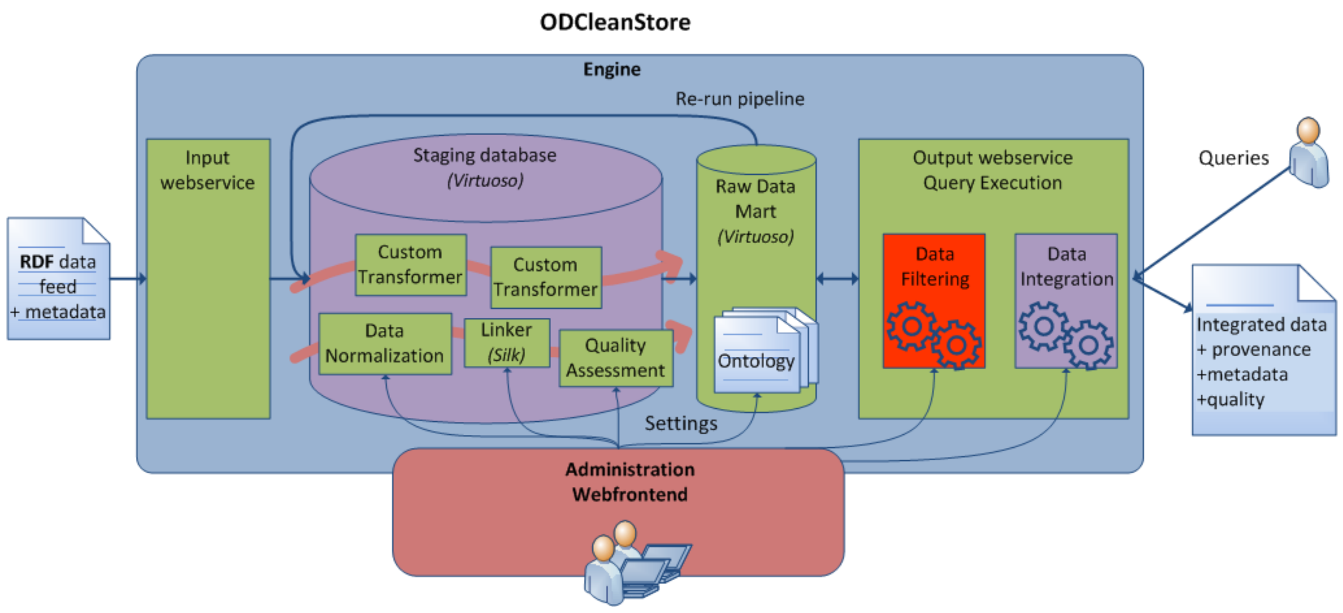


Figure 1. ODCleanStore Framework with Provenance Data Filtering Component

created during the pipeline execution, such as links to other resources or metadata about the feed's quality.

Consumers can query the raw data mart (using the *output web service*). In further text, we highlight two types of queries: *uri* queries and *keyword* queries. Suppose that the set Q_q contains quads from the raw data mart relevant to the consumer's query q; for an uri query q, Q_q contains the quads $(x,*,*,*)$,$(*,*,x,*)$, where x is the URI in the consumer's query or URI being `owl:sameAs` with the URI in the consumer's query; for a keyword query q with a sequence of keywords kw, Q_q contains the quads $(*,*,l,*)$, where the literal l contains the keywords kw. Sample keyword query may be: "Give me all you know about the Ministry of Finance of the Czech Republic".

Since the same resource can be described by various sources, using different schemas (vocabularies), data integration is necessary to provide the data consumer with an integrated view on the data. Data integration consists of three phases [10] – *schema mapping* (the detection of equivalent schema elements in different data graphs), *duplicate detection* (detection of equivalent resources) and *data fusion* (fusion the descriptions of equivalent resources).

Data integration module in ODCleanStore addresses all the integration phases – it takes into account the mappings between different schemas stored in the raw data mart and links deduplicating resources generated by the proper transformers on the data processing pipelines and then fuses the data. When fusing data about the same resource from multiple sources (data graphs), data conflicts may arise and should be solved. Thus, as part of the data fusion phase, data integration module in ODCleanStore applies certain conflict handling strategies which resolve, ignore, or avoid the data conflicts in the resulting RDF data. Provided the conflict handling strategy is to resolve the conflicts, consumer can also specify the *conflict resolution policies* driving the data fusion.

Furthermore, the resulting RDF data outputted by the data integration module in a form of quads is supplemented with a *quality score* influenced by the quality of the feed the data originates from (which is quantified on the transforming pipeline by the quality assessment transformer), the score of the data publisher (e.g., a domain http://wikipedia.org), and by the applied conflict resolution policies [16].

Thus, the ODCleanStore framework is able to addressed the objective part of the data quality by employing quality assessment transformer. Nevertheless, the information quality must be always considered w.r.t the specific (subjective) requirements of the consumer [8, 17, 20] for his particular task at hand. For example, the consumer might want to prefer data from the Czech Business Register when looking for data about companies, or use only sources verified by his boss.

These needs are partially supported – the resulting data is accompanied also with *data provenance* of the data graphs (sources) the data originates from, providing the necessary contextualization for the information consumer to analyze the (subjective) quality of the information [12, 13, 19]. However, the manual examination of such provenance and manual filtering of the resulting data based on its provenance is rather tedious work for information consumers.

Therefore, in this paper, we describe the concept of *provenance policies* which will enable the data consumer to express his subjective preferences for certain data sources based on the provenance data associated with these sources; such policies must be taken into account automatically during the query execution - as a result, only data from certain sources appear in the result on the consumer's query q. We describe in the paper the format of the provenance policies and how they are applied during the query execution as part of the data filtering module in ODCleanStore to refine the data being provided to data consumers.

II. Data Provenance

We use in this paper the definition of provenance, which is based on the definition introduced by W3C Provenance Group [7] and takes some aspects from the "provenance as annotations" approach [19]: "Provenance of a resource r is a record that contains resources, agents, or processes (and their properties contextualizing them) involved in producing and delivering or otherwise influencing the resource r. "

W3C Provenance Group is defining the core provenance terms for tracking provenance on the Web; in [12], we described a W3P provenance model for the Web.

In further text, *provenance graph* is the named graph $g^p \in G$ (set of triples belonging to that graph) containing provenance information about the data graph $g \in G$; data feed inserted to the staging area always contains the data graph g and may contain provenance graph g^p.

```
<http://source.com/1> a prov:Entity ;
  dc:creator <http://purl.org/knap#me> ;
  dc:created "2011-11-18"^^xsd:date ;
  dc:source <http://source.com/2> ;
  p:hadPrimarySource <...> .
```

Listing 1. Sample Provenance Graph

Listing 1 depicts the sample provenance graph holding provenance data about the data graph `<http://source.com/1>`. As you can see, the provenance graph holds (1) the creator of the source, (2) the creation time, and (3) the primary sources from which that source was obtained (e.g., extracted); in Listing 1, the prefix `dc:` stands for `http://purl.org/dc/terms/`, `p:` stands for `http://www.w3.org/ns/prov#`.

III. Provenance Policies

We define a provenance policy $p \in P$ as a tuple $(cond, weight)$, where $cond \in C$, C is a set of all valid `GroupGraphPatterns` in the SPARQL language [5]; function $w(p)$, $w : P \to [-1, 1] \setminus \{0\}$ quantifies the $weight$ of the policy p, $w(p) \in (0, 1]$ determines a *positive policy* and $w(p) \in [-1, 0)$ determines a *negative policy* p.

A provenance policy $p = (cond,\ weight) \in P$ can be successfully applied to the provenance named graph g^p if and only if a SPARQL query "ASK FROM NAMED g^p WHERE $\{cond\}$" returns $true$. The successful application is expressed as $a(p, g^p) = true$; otherwise, if the policy was not successfully applied, $a(p, g^p) = false$; $a : P \times G \to \{true,\ false\}$. If $a(p, g^p) = true$, then the policy p changes the *provenance score* of the graph g according to $weight$. Positive policy always increases the provenance score, negative policy decreases.

The condition $cond \in C$ of a policy $p = (cond,\ weight) \in P$ may use the variable `odcs:graph` which is replaced by the particular data graph g being processed before the query is sent to the underlying SPARQL engine; full URL for the odcs prefix is: http://ld.opendata.cz/infrastructure/odcleanstore/. Suppose a condition $cond$ = {`odcs:graph dc:creator <http://purl.org/knap#me>`}; such condition is matching all the graphs g^p containing the triple with the subject being the URI of the graph g, the predicate `dc:creator` and the object `<http://purl.org/knap#me>`, i.e., all graphs created by the agent `<http://purl.org/knap#me>`.

We define a function $s_{prov} : G \times \mathcal{P}(P) \to (0, 1]$ quantifying the *provenance score* of the graph g based on the weights of the policies $P_a = \{p \in P \mid a(p, g^p) = true\}$ successfully applied to g^p as:

$$s_{prov}(g, P_a) = min\{\frac{\prod_{p \in P_a}(1 + w(p))}{C}, 1\}$$

The constant $C \in \mathbb{N}$ defines the upper boundary for the influence of the positive policies; if $\prod_{p \in P_a}(1 + w(p)) > C$, the provenance score $s_{prov}(g, P_a)$ is equivalent to the case when $\prod_{p \in P_a}(1 + w(p)) = C$. The constant C should be set based on the average proportion of positive and negative policies and the average absolute number of positive policies applied to the graphs. Furthermore, the default provenance score of any graph to which no policy was successfully applied should be equal to $1/C$.

IV. Applying Provenance Policies

The application of provenance policies is part of the data filtering component depicted in Figure 1. The data filtering component is executed during query execution after fetching the quads, Q_q, relevant for the uri or keyword query q from the raw data mart. An output of the data filtering component is the collection of quads, $\widetilde{Q_q} \subseteq Q_q$, which is created as defined further in Algorithm 1; such output is used as the input to the data integration component in Figure 1. The quality score computed in the data integration module of ODCleanStore [16] also takes into account the provenance score s_{prov} computed for the graphs involved in $\widetilde{Q_q}$.

The inputs to Algorithm 1 are (1) the quads, Q_q, being fetched as a result of the consumers query q, (2) policies, $P_c \subset P$, defined by the consumer c executing the query, (3) constraints, $F_q \subset F$, customizing the behavior of the algorithm for the given query q, and (4) the desired threshold, $\kappa \in [0, C]$, for the provenance score. The output is the refined set of quads, $\widetilde{Q_q}$, $|Q_q| \geq |\widetilde{Q_q}|$, belonging to data graphs g having the provenance score above the threshold κ. The algorithm can enforce the following constraints F on

Algorithm 1 Provenance Policies Application

```
Input: $Q_q$, $P_c \subseteq P$, $F_q \subseteq F$, $\kappa$
Output: $\widetilde{Q_q} = applyProvPolicies(Q_q,\ P_c,\ F_q,\ \kappa)$
 1: $\widetilde{Q_q} \leftarrow \emptyset$
 2: $G_q \leftarrow \{g \mid \exists(*,*,*,g) \in Q_q\}$
 3: for all graphs $g \in G_q$ do
 4:     $P_a \leftarrow \emptyset$, $flagResult \leftarrow true$
 5:     for all policies $p \in P_c$ do
 6:         if $a(p, g^p)$ then
 7:             $P_a \leftarrow P_a \cup \{p\}$
 8:         end if
 9:     end for
10:     for all flags $f \in F$ do
11:         $flagResult \leftarrow flagResult \wedge eval(P_a, f)$
12:     end for
13:     if $flagResult$ then
14:         if $s_{prov}(g, P_a \geq \kappa_{pp}$ then
15:             $\widetilde{Q_q} \leftarrow \widetilde{Q_q} \cup$ {(*,*,*,g)}
16:         end if
17:     end if
18: end for
19: return $\widetilde{Q_q}$
```

the provenance graphs of the data graphs whose triples are included in $\widetilde{Q_x}$:

- NoNeg - Negative policy shall not be successfully applied to the provenance graph.
- ExistsPos - A positive policy shall be successfully applied to the provenance graph.
- PosMajority - Number of positive policies successfully applied to the provenance graph shall prevail over the number of negative policies.
- PolMandatory - At least one policy shall be successfully applied to the provenance graph.

In Lines 3 – 18 of Algorithm 1, the provenance policies are successively applied to the graphs $g \in G_q = \{g \mid \exists(*,*,*,g) \in Q_q\}$. In Lines 5 – 9, the set P_a of successfully applied policies is constructed; based on that, in Lines 10 – 12, the function $eval$, $eval : \mathcal{P}(P) \times F \rightarrow \{true, false\}$, progressively checks the satisfaction of all the constraints F_q w.r.t. to the set of polices P_a and directly influences the construction of $\widetilde{Q_q}$. The function $eval(P_a, f)$ is defined as follows for the set of successfully applied policies $P_a \subset P_c \subset P$ and the constraint $f \in F$:

$$eval(P_a, NoNeg) = \begin{cases} true\,\{\nexists p \in P_a : w(p) < 0\} \\ false\,\{otherwise\} \end{cases}$$

$$eval(P_a, ExistsPos) = \begin{cases} true\,\{\exists p \in P_a : w(p) > 0\} \\ false\,\{otherwise\} \end{cases}$$

$$eval(P_a, PosMajority) = \begin{cases} true\left\{|\{p|w(p) > 0\}| > \frac{|P_a|}{2}\right\} \\ false\,\{otherwise\} \end{cases}$$

$$eval(P_a, PolMandatory) = \begin{cases} true\,\{|P_a| > 0\} \\ false\,\{otherwise\} \end{cases}$$

In Lines 14 – 16, if all the flags F_q are satisfied for the given P_a, the algorithm tests whether the condition on the threshold $\kappa \in [0, C]$ of the provenance score is satisfied; if yes, the quads of the data graph g are added to $\widetilde{Q_q}$ in Line 15. Otherwise, the quads associated with the processed graph g are not included in $\widetilde{Q_q}$.

The time complexity of Algorithm 1 is $O(|Q_q| + |G_q||P_c|O(a(p, g^p)))$, where $O(|Q_q|)$ yields from loading the quads to the memory and $O(a(p, g^p))$ is the time complexity of applying a single policy $p \in P_c$ to the provenance graph g^p. Space complexity is at least $O(|Q_q|)$, because the quads has to be loaded to the memory, but depends also on the SPARQL queries being executed as part of the policy application.

V. Related Work

Researchers have developed and investigated various policy languages to describe trust, quality, and security requirements on the Web, such as [8, 14]; a variety of access control mechanisms generally based on policies and rules have been developed, such as [18]. In linked data framework WIQA (Web Information Quality Assessment Framework) [8], users can specify policies in the form of RDF graph patterns using the WIQA-PL policy language; they can filter the information in their local storage according to the selected policy, and get justifications for ”why” a given information satisfies a set of policies. WIQA-PL and our policy language are both based on the SPARQL language; the grammar for WIQA-PL is not aligned with the latest SPARQL specification. A WIQA-PL policy enables to define which information is filtered positive; ODCleanStore supports both positive and negative filtering. WIQA does support the provision of justifications by extending the SPARQL language with the construct EXPL; in ODCleanStore, justifications are represented by the list of policies being applied to the resulting data.

VI. Conclusions and Future Work

In this paper, we described the concept of provenance policies – motivation for provenance policies, how these policies can be constructed, and how they can be applied as part of the data filtering component in ODCleanStore when the query is prepared for the data consumer. We are persuaded that provenance policies are an important mechanism to address the subjective dimension of data quality on the Web.

Future work involves the evaluation of the provenance policies' usability. To that end, we will develop a linked data browser [9] using the ODCleanStore's output web service's results and supporting data consumers with simple user interface to (1) manage provenance policies and (2) select iteratively relevant provenance policies which should be applied to the data resulting from the given query. Future work also involves examining the reasoning possibilities within provenance graphs, which affects the efficiency of provenance policies application.

VII. Acknowledgments

The work presented in this article has been funded in part by GAUK 3110, project of the internal grant agency of the Charles University.

References

[1] Linked Open Data Cloud. http://richard.cyganiak.de/2007/10/lod/ (Online, retrieved: January, 2013).

[2] LOD2 Project. http://lod2.eu (Online, retrieved: January, 2013).

[3] ODCleanStore. http://sourceforge.net/p/odcleanstore (Online, retrieved: January, 2013).

[4] OpenData.cz Initiative. http://opendata.cz (Online, retrieved: January, 2013).

[5] SPARQL Query Language - Group Graph Pattern. http://www.w3.org/TR/rdf-sparql-query/#rGroupGraphPattern (Online, retrieved: January, 2013).

[6] The Open Data Handbook. http://opendatahandbook.org/en/ (Online, retrieved: January, 2013).

[7] W3C Provenance Working Group. http://www.w3.org/2011/prov (Online, retrieved: January, 2013).

[8] C. Bizer and R. Cyganiak. Quality-driven Information Filtering Using the WIQA Policy Framework. *Web Semantics*, 7(1):1–10, 2009.

[9] C. Bizer, T. Heath, and T. Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.

[10] J. Bleiholder and F. Naumann. Data fusion. *ACM Comput. Surv.*, 41(1):1:1–1:41, Jan. 2009.

[11] J. J. Carroll, C. Bizer, P. Hayes, and P. Stickler. Named graphs, Provenance and Trust. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 613–622, New York, NY, USA, 2005. ACM.

[12] A. Freitas, T. Knap, S. O'Riain, and E. Curry. W3P: Building an OPM based provenance model for the Web. *Future Generation Comp. Syst.*, 27(6):766–774, 2011.

[13] O. Hartig. Provenance Information in the Web of Data. In *Linked Data on the Web (LDOW 2009), http://events.linkeddata.org/ldow2009/papers/ldow2009_paper18.pdf*, April 2009.

[14] L. Kagal, T. Finin, and A. Joshi. A Policy Based Approach to Security for the Semantic Web. In *The SemanticWeb - ISWC 2003, Florida, USA*, pages 402–418, 2003.

[15] G. Klyne and J. J. Carroll, editors. *Resource Description Framework (RDF): Concepts and Abstract Syntax.* W3C Recommendation. World Wide Web Consortium, Feb. 2004.

[16] T. Knap, J. Michelfeit, and M. Necaský. Linked Open Data Aggregation: Conflict Resolution and Aggregate Quality. In *COMPSAC Workshops, Izmir, Turkey*, pages 106–111, 2012.

[17] S. A. Knight and J. Burn. Developing a Framework for Assessing Information Quality on the World Wide Web. *Informing Science Journal*, 8:159–172, 2005.

[18] K. Lawrence and C. Kaler. WS-Trust Specification. Technical report, 2007, http://docs.oasis-open.org/ws-sx/ws-trust/200512.

[19] L. Moreau. The Foundations for Provenance on the Web. *Found. Trends Web Sci.*, 2(2&#8211;3):99–241, Feb. 2010.

[20] F. Naumann and C. Rolker. Assessment Methods for Information Quality Criteria. In *Proceedings of the International Conference on Information Quality*, pages 148–162, 2000.

Merging Multidimensional Data Models: A Practical Approach for Schema and Data Instances

Michael Mireku Kwakye, Iluju Kiringa, Herna L. Viktor
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Ontario, Canada.
mmire083@uottawa.ca, {kiringa, hlviktor}@eecs.uottawa.ca

***Abstract*—Meta-model merging is the process of incorporating data models into an integrated, consistent model against which accurate queries may be processed. Within the data warehousing domain, the integration of data marts is often time-consuming. In this paper, we introduce an approach for the integration of relational star schemas, which are instances of multidimensional data models. These instance schemas represented as data marts are integrated into a single consolidated data warehouse. Our methodology which is based on model management operations focuses on a formulated merge algorithm and adopts first-order Global-and-Local-As-View (GLAV) mapping models, to deliver a polynomial time, near-optimal solution of a single integrated data warehouse.**

Keywords-Schema Merging; Data Integration; Model Management; Multidimensional Merge Algorithm; Data Warehousing

I. INTRODUCTION

Schema merging and data integration are important research areas with many practical applications. Some of the application areas are federated database systems, Enterprise Information Integration (EII), bioinformatics data integration, and financial information integration. Schema merging involves the integration of instance schema of meta-data models using the mappings between the elements of the instance schemas [1]. Data integration, on the other hand, involves the consolidation of the instance data within the framework of a merged instance schema to deliver efficient query solutions [2]. Most procedures that involve these concepts have focused on traditionally identifying the independent data sources and the associated element mapping correspondences. Recent studies have emphasized the importance of inferring the semantic meaning of the data source elements during integration. Some problems that are associated with the procedural methodologies for these concepts are the identification of prime meta-models, and the formulation of algorithms for specific meta-models and their schema and data instances.

The conceptual processes of data integration and schema merging largely come from the fundamental operations of model management [3] [4]. Some of these operations are namely, *match schemas* (expressed as schema matching), *compose mappings* and *apply mappings* (both expressed as schema mapping discovery), and *merge schemas* (expressed as schema merging) [3]. In line with multidimensional data integration for data warehouses, a number of studies have been investigated. Cabibbo and Torlone [5] [6] introduce and address dimension algebra and dimension compatibility in relation to data marts integration. Riazati et al. [7] also propose a solution for integration of data marts where they infer on the aggregations in the hierarchies of the dimension. Although these studies and others attempt to address this integration problem, they fail to investigate in detail areas such as an elaborate merge algorithm, element conflict management, technical merge requirements, amongst others.

In this paper, we introduce an integration procedure for both instance schema and instance data of multidimensional data models. Our motivation is to employ the concept of model management to address the above-mentioned shortcomings of merge algorithm, conflict management and technical merge requirements for integration of data marts. Our key contribution in this paper is the formulation of a novel well-defined algorithm capable of delivering an efficient integrated data warehouse. Our presentation focuses on the proposition of star schema instances in our analyses. We deal with different procedures starting with finding of mapping correspondences to a more complex procedure of merging. Our work subsumes and extends prior work on generic models [1], to present a practical solution for merging schema instances of multidimensional data models.

The technical contributions may be summarized as follows. We adopt a hybrid form of schema matching, in which we use both instance schema structure and instance data and extension algorithms to deliver correct attribute mapping correspondences. To this end, we employ first-order Global-and-Local-As-View (GLAV) mapping models in the mapping discovery procedure. We identify and resolve specific conflicts that are exposed as a result of the integration of data marts. We further define technical qualitative merge correctness requirements which serve to validate the formulation of our merge algorithm.

This paper is organized as follows. In Section II, we discuss our integration methodology. We present the multidimensional instance schema and data merging in Section III. In Section IV, we address the implementation and evaluation analysis of the merge methodology. In Section V, we conclude, discuss the open issues, and the areas of future work.

II. INTEGRATION METHODOLOGY

Our approach for generating a single integrated data warehouse from independent, but related, multidimensional star schemas extends from the above-mentioned concept of model management.

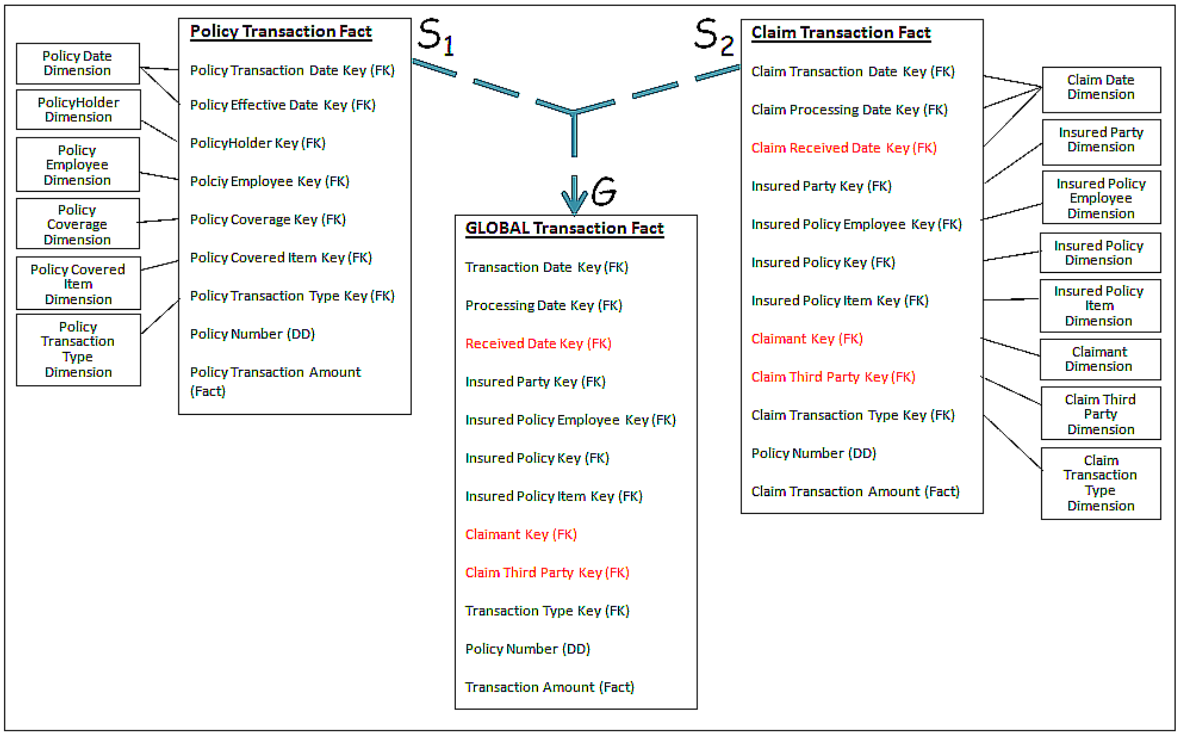

Policy Transaction Fact
Policy Transaction Date Key (FK)
Policy Effective Date Key (FK)
PolicyHolder Key (FK)
Polciy Employee Key (FK)
Policy Coverage Key (FK)
Policy Covered Item Key (FK)
Policy Transaction Type Key (FK)
Policy Number (DD)
Policy Transaction Amount (Fact)
Policy Date Dimension
PolicyHolder Dimension
Policy Employee Dimension
Policy Coverage Dimension
Policy Covered Item Dimension
Policy Transaction Type Dimension
S1
S2
G
GLOBAL Transaction Fact
Transaction Date Key (FK)
Processing Date Key (FK)
Received Date Key (FK)
Insured Party Key (FK)
Insured Policy Employee Key (FK)
Insured Policy Key (FK)
Insured Policy Item Key (FK)
Claimant Key (FK)
Claim Third Party Key (FK)
Transaction Type Key (FK)
Policy Number (DD)
Transaction Amount (Fact)
Claim Transaction Fact
Claim Transaction Date Key (FK)
Claim Processing Date Key (FK)
Claim Received Date Key (FK)
Insured Party Key (FK)
Insured Policy Employee Key (FK)
Insured Policy Key (FK)
Insured Policy Item Key (FK)
Claimant Key (FK)
Claim Third Party Key (FK)
Claim Transaction Type Key (FK)
Policy Number (DD)
Claim Transaction Amount (Fact)
Claim Date Dimension
Insured Party Dimension
Insured Policy Employee Dimension
Insured Policy Dimension
Insured Policy Item Dimension
Claimant Dimension
Claim Third Party Dimension
Claim Transaction Type Dimension

Figure 1. Logical and Conceptual Multidimensional Schema Merge

In line with this meta-data conceptual assertion, we present an overview of our integration methodology, as depicted in Figure 1. The figure shows a logical and conceptual merging of the fact and dimension tables from the *Policy* and *Claims* data marts, of an Insurance industry, to form an enterprise data warehouse. We explain further our motivation using Example 1 and Figure 1.

Example 1. *Suppose we have 2 data marts from an Insurance industry – Policy Transactions and Claims Transactions – and we have to integrate these data marts into an enterprise-wide data warehouse, as illustrated in Figure 1. The existence of corresponding attributes will enable the possibility of integrating the attributes of the fact and dimension tables of these data marts. A merge algorithm can be applied to the corresponding mappings to generate the integrated data warehouse needed in answering queries, as it will be posed to the independent data marts.* ■

A. Overview of Integration Methodology

We outline our methodology based on 3 main streamlined procedures. These are finding mapping correspondences, mapping models discovery, and the formulation of merge algorithm. Figure 2 illustrates a description of our methodology and framework architecture in a workflow chain. Here, we describe the step-wise procedures and processes, algorithm executions, and the generated outputs, as well as, query analyses. We further describe into detail the first 2 procedures (Finding Mapping Correspondences and Mapping Models Discovery & Modelling) and give also a detailed description of procedure 3 (Merge Algorithm) in Section III.

B. Finding Mapping Correspondences

In our methodology, we adopt a hybrid form of schema matching which aim to deliver efficient schema attribute correspondences. Our adoption of this hybrid approach uses the logical and conceptual features of the multidimensional schema structure in schema-based matching and the instance data and extensions in instance-based matching, to find attribute correspondences. We adopted schema-based algorithms in the form of Lexical Similarity and Semantic Names. The Lexical Similarity uses schema string names and text, equality of names, synonyms, homonyms, and similarity of common substrings. The Semantic Names, on the other hand, uses schema data types, constraints, value ranges, relationship types, amongst others to match attributes [10]. We use Example 2 to illustrate the schema-based form of finding mapping correspondences.

Example 2. *Following up on Example 1, suppose we want to merge the dimensions of DimPolicyHolder and DimInsuredParty from Policy and Claims data marts, respectively. The application of Lexical Similarity algorithm will produce mapping correspondences, such as:*

1. $PolicyHolder.PolicyHolderKey \approx InsuredParty.InsuredPartyKey$

2. $PolicyHolder.FullName \approx InsuredParty.FamilyName,$ $InsuredParty.GivenName, InsuredParty.CityName$

3. $PolicyHolder.Address \approx InsuredParty.StreetAddress,$ $InsuredParty.EmailAddress$

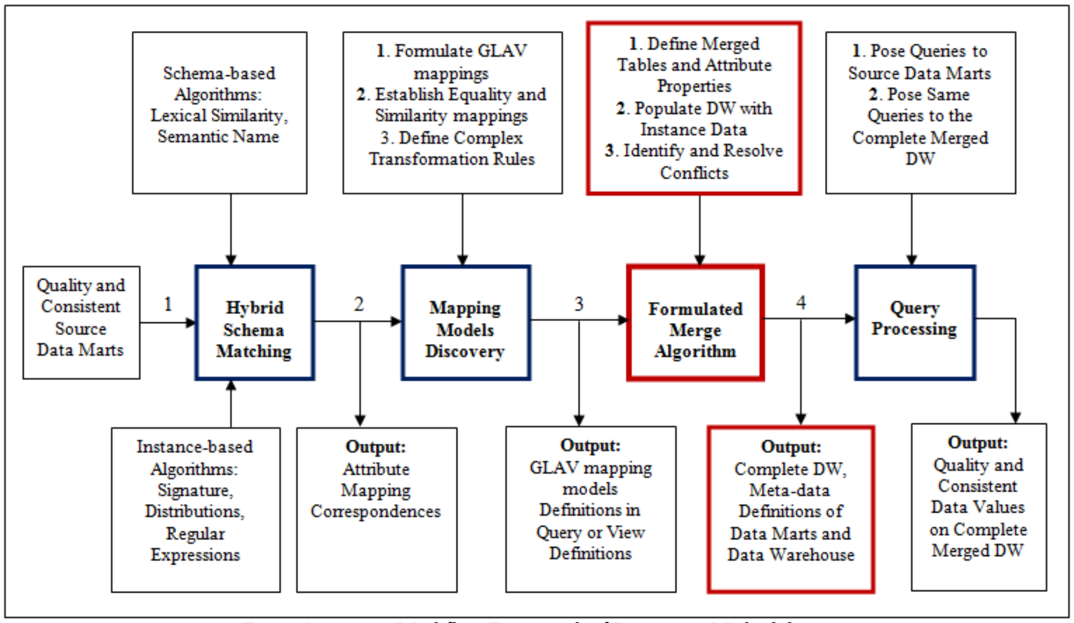

Schema-based Algorithms: Lexical Similarity, Semantic Name
1. Formulate GLAV mappings 2. Establish Equality and Similarity mappings 3. Define Complex Transformation Rules
1. Define Merged Tables and Attribute Properties 2. Populate DW with Instance Data 3. Identify and Resolve Conflicts
1. Pose Queries to Source Data Marts 2. Pose Same Queries to the Complete Merged DW
Quality and Consistent Source Data Marts
1
Hybrid Schema Matching
2
Mapping Models Discovery
3
Formulated Merge Algorithm
4
Query Processing
Instance-based Algorithms: Signature, Distributions, Regular Expressions
Output: Attribute Mapping Correspondences
Output: GLAV mapping models Definitions in Query or View Definitions
Output: Complete DW, Meta-data Definitions of Data Marts and Data Warehouse
Output: Quality and Consistent Data Values on Complete Merged DW

Figure 2. Workflow Framework of Integration Methodology

*Moreover, the application of the Semantic Names algorithm will offer an improved schema matching. This matching eliminated **InsuredParty.CityName** in the 2nd matching to deliver mapping correspondence, as in:*

2. $PolicyHolder.FullName\ [varchar(60)] \approx InsuredParty.FamilyName[varchar(30)], InsuredParty.GivenNames[varchar(40)]$ ■

The instance-based algorithms that were adopted are Signature, Distributions, and Regular Expressions. The Signature algorithm uses the similarity in the actual data values contained in the schemas based on data sampling. The Distributions algorithm, on the other hand, uses the common values and frequent occurrences of data values based on sampling. The Regular Expressions algorithm uses textual or string searches based on regular string expressions or pattern matching [10]. We use Example 3 to illustrate a generalized form of instance-based algorithm.

Example 3. *Following up on Examples 2, we complement the results of the initial schema-based mapping correspondences with a generalized instance-based mapping to produce a final semantically correct mapping correspondence for the 3rd matching, as in:*

3. $PolicyHolder.Address \approx InsuredParty.StreetAddress$

This final matching was attained because of the data values and extensions from the dimension attributes. Some of the instance data values contained in PolicyHolder.Address are {39 Baywood Drive, 178 Flora Ave., 79 Golden Rain St.}, where as data values contained in InsuredParty.StreetAddress and InsuredParty.EmailAddress are {40 Roslyn St., 68 Hastings Drive, 48 Whitehall Avenue} and {amartens@cybserv.com, drice@vipe2k.com, jtausig@fitexes.com}, respectively. ■

C. Mapping Models Discovery and Modelling

Definition 1. (First-Order Mapping): Let $\mathcal{M} = (S, T, f)$ represent a mapping model from Source, S and Target, T schemas. Let $a \in \{S \cup T\}$ represent disjoint variable element where a denotes $\{a_1, a_2, \ldots, a_n\}$. The mapping assertion, $\mathcal{M}$ is said to be in first-order if $f: \{\forall a\ (S(a) \rightarrow T(a))\}$, where f represents the logical view from the Source to the Target. ■

We adopted first-order Global-and-Local-As-View (GLAV) mapping model formalisms in the mapping discovery procedural step. Our motivation is based on the expressiveness of the correspondences that exist between the attributes of the schemas [2]. This mapping model combines mapping formalisms from both the Local-As-View (LAV) and Global-As-View (GAV) mappings. It expresses mapping views where the extensions of the source schemas provide any subsets of tuples satisfying the corresponding view over the global mediated schema. Moreover, an equivalent number of attribute view definitions are expressed in both the LAV and GAV queries [2]. One other unique feature is the expression of multi-cardinality mappings between mapping elements. This enables the expression of complex transformation formula which is much useful in our integration methodology [12].

Definition 2. (Equality Mapping): Let $\mathcal{M} = (S, T, f)$ represent a mapping for Source, S and Target, T schemas. The assertion $f: \{\forall x \forall y\ (S(x, y) \rightarrow \exists z\, T(x, z))\}$ for disjoint variable elements x, y, z is an Equality mapping such that $y =$ z. ■

Definition 3. (Similarity Mapping): Let $\mathcal{M} = (S, T, f)$ represent a mapping for Source, S and Target, T schemas. For disjoint element variables x, y, z the assertion $f: \{\forall x \forall y\ (S(x, y) \rightarrow \exists z\, T(x, z))\}$ is a Similarity mapping such that $g(y) = z$ where g denotes or encloses a complex transformation expression. ∎

In this second step of mapping discovery and modelling, 2 forms of mapping relationships were adopted. These are equality and similarity mapping relationships. It should be emphasized that these defined classifications were based on expressive characterization of relationship cardinality, and the attribute semantic representation, amongst others [11]. We used these forms of mapping relationships in a GLAV mapping model, as explained in Example 4.

Example 4. *Continuing on Example 1, suppose we want to integrate the DimPolicyHolder and DimInsuredParty dimensions from Policy and Claims data marts, respectively, into DimInsuredPolicyHolder dimension. The Datalog queries for the GLAV mapping model will be expressed as:*

InsuredPolicyHolder *(InsuredPhKey, InsuredPhID, InsuredPhName, BirthDate, StateProvince, Region, City, Status):-*
PolicyHolder *(PolicyHolderKey, PolicyHolderID, PolicyHolderFullName, DateOfBirth, State, City, Status),*
InsuredParty *(InsuredPartyKey, InsuredPartyID, InsuredFamilyName, InsuredGivenName, BirthDate, Province, Region, CityName)*

In this Datalog query, the existence of corresponding attributes in both dimensions are automatically expressed in the merged dimension, as well as, local attributes of ***Status*** *and* ***Region*** *from Policy and Claims data marts, respectively, are included in the global or merged dimension.* ∎

III. MULTIDIMENSIONAL INSTANCE SCHEMA AND DATA MERGING

In this section, we present the technical qualitative requirements necessary for producing an efficient single consolidated data warehouse. We further outline and describe an elaborate merge algorithm (Algorithm 1) for integrating the instance schema and data of data marts fact and dimension tables. We finally describe the resolution of identifiable conflicts associated with the integration of the data marts.

A. Merge Correctness Requirements

The single consolidated data warehouse that is generated as a result of the implementation of the merge algorithm needs to satisfy some requirements, to ensure the correctness of the data values from the queries that would be posed to it. These qualitative technical requirements describe the properties that the data warehouse schema should exhibit.

Drawing on the propositions in the requirements defined by the authors in [1] for merging generic meta-models, we performed a gap analysis on their propositions in relation to generating a data warehouse. Hence, we formulate and describe a set of correctness requirements in relation to merging of multidimensional star schemas. These technical requirements extend the requirements already proposed in [1], in order to address star schemas. We outline the set of *Merge Correctness Requirements* (MCR) that validates the formulated merge algorithm needed for the generation of a global data warehouse.

Dimensionality Preservation. For each kind of dimension table connected to any of the integrating fact tables, there is a representation of corresponding dimension also connected to the merged fact table.

Measure and Attribute Entity Preservation. All fact or measure attribute values in either of the integrating fact tables are represented in the merged fact table. Additionally, all other attribute values in each of the dimension tables are represented through an equality or similarity mapping. Finally, there is an automatic inclusion for non-corresponding attributes in the merged fact (dimension) tables based on the condition of no attribute redundancy or duplication.

Slowly Changing Dimension Preservation. Slowing Changing Dimension is the occurrence where an entity in a dimension has multiple representations based on the changes in instance data values in some key attributes. For such dimensional entity occurences, the merged dimension should offer an inclusion of all the instance representations from each integrating dimension. Hence, we enforce an automatic inclusion of attributes that contribute to the dimensional change in the merge dimension.

Attribute Property Value Preservation. The merged attribute should preserve the value properties of the integrating attributes, whether the mapping correspondence is an equality or similarity mapping. Equality mapping should be trivially satisfied by the *UNION* property for all equal attributes. For a similarity mapping, the transformation expression should have the properties to be able to satisfy the attribute property value of each integrating dimension attribute.

Definition 4. (Surrogate Key): Let $\mathcal{D}_i$ represent a dimension table for a multidimensional model, $\mathcal{B}$ such that $\mathcal{D}_i \in \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n\}$ for $i \leq n$. Let $\mathcal{E}$ represent each entity of a dimension, $\mathcal{D}_i$ such that $\mathcal{E} \in \mathcal{D}_i$. The identifier, $\mathcal{K}$ is said to be a Surrogate Key for $\mathcal{E}$ such that $\mathcal{K}_m \equiv \mathcal{E}_m$ ∎

Tuple Containment Preservation. The single consolidated data warehouse should offer the containment of all unique tuples as they are valuable in returning correct answers to queries posed. This ensures the preservation of all *Surrogate Keys* needed in identifying each dimensional entity.

B. Merge Algorithm

The merge algorithm (Algorithm 1) is formulated and designed to generate the single consolidated data warehouse from different related data marts, modelled as star schemas instances. The algorithm primarily performs 2 levels of integration. Firstly, the integration of the instance schema structure which comprises the attribute relationships and properties for the fact and dimension tables. These procedures are described in Steps (1) to (9).

```
Algorithm 1:    Multidimensional Instance Schema and Data Merging
Input:
(a)  A set of star schema data marts, A and B
(b)  A set of first-order GLAV mapping model; $Mapping_{AB}$, consisting of $factMapping_{AB}$ and $dimMapping_{AB}$
(c)  An optional designation of a data mart, A or B, as the $preferredDataMart$;
Output:
(a)  A single consolidated star schema instance data warehouse free of duplicate and redundant schema and instance data.
(b)  A metadata consisting of data definition of the integrating data marts and the single consolidated data warehouse.
Procedure:
Initialization
 (1)  Let $mergeDataWarehouse \leftarrow NULL$
Generate Merged Table
 (2)  For each $correspondingMappingType \in factMapping_{AB}$ do
        (a) If $corrrespondingMappingType = NULL$ then
            i.  Return $mergeDataWarehouse \leftarrow NULL$
        (b) Else
            i.  Let $mergeDataWarehouse \leftarrow mergeFactTable \in \{factTableA, factTableB\}$
 (3)  Repeat Step (2) for each $mergeDimTable$ using $dimMapping_{AB}$, add $\{nonCorrespondingDimTable\}$
 (4)  Return $mergeDataWarehouse \supset \{mergeFactTable, \{mergeDimTable, nonCorrespondingDimTable\}\}$
Merged Table Attribute Relationships
 (5)  For each $correspondingMappingType \in factMapping_{AB}$ do
        (a) Let $mergeFactTable \leftarrow NULL$
        (b) If $correspondingMappingType =$ "Equality" then
            i.  Let $mergeFactAttribute \leftarrow definedAttribute \in \{factMapping_{AB} \in preferredDataMart\}$
        (c) Else If $correspondingMappingType =$ "Similarity" then
            i.  Let $mergeFactAttribute \leftarrow definedAttribute \in factMapping_{AB}$
 (6)  For each $nonCorrespondingAttribute \in \{factTableA, factTableB\}$ do
        (a) If $nonCorrespondingAttribute \notin \{mergeAttribute\}$ then
            i.  Let $mergeFactAttribute \leftarrow nonCorrespondingAttriubte$
        (b) Return $mergeFactTable \supset \{mergeFactAttribute, nonCorrespondingAttribute\}$
 (7)  For each $correspondingMappingType \in dimMapping_{AB}$ do
        (a) Repeat Step (3) for each $correspondingAttribute \in \{dimTableA, dimTableB\}$
        (b) Repeat Step (4) for each $nonCorrespondingAttribute \in \{dimTableA, dimTableB\}$
        (c) Return $mergeDimTable \supset \{mergeDimAttribute, nonCorrespondingAttribute\}$
Merged Table Attribute Properties
 (8)  For each $mergedFactAttribute \in mergeFactTable$ do
        (a) Let $mergeAttributeTypeValue \leftarrow definedAttributeType \in factMapping_{AB}$
 (9)  Repeat Step (6) for each $mergeDimTable$ using $dimMapping_{AB}$
Dimension Tables Data Population
 (10) For each $mergeDimTable$ do
        (a) If $(keyIdentifierConflict$ OR $multipleEntityRepresentation) = TRUE$ then
            i.  Let $entityKeyIdentifier \leftarrow surrogateKey \in preferredDataMart$
        (b) Else
            i.  Let $entityKeyIdentifier \leftarrow (newSurrogateKey \equiv primaryKey) \in nonPreferredDataMart$
Fact Table Data Population
 (11) For each $mergeFactTable$ do
        (a) Load fact records using $entityKeyIdentifier \in \{surrogateKey, newSurrogateKey\}$
 (12) Let $mergeDataWarehouse \supset \{mergeFactTable, \{mergeDimTable, nonCorrespondingDimTable\}\}$
 (13) Return $mergeDataWarehouse$
```

Steps (1) to (4) initialize and generate the integrated fact and dimension tables. Steps (5) to (7) describe the generation of attributes for the integrated tables. Finally, Steps (8) and (9) describe the derivation of attribute property values of the merged fact and dimension tables.

Secondly, the algorithm performs integration of the instance data contained in the star schema data marts. This involves the population of these instance data from the data marts fact and dimension tables into the merged tables in the data warehouse. Steps (10) to (13) describe these procedures of data population.

We further summarize the merge algorithm in fulfilment of the technical *Merge Correctness Requirements* (MCRs) outlined in Section III.A.

a) Step (2) satisfies Dimensionality Preservation: Each fact and dimension table is iterated to form the Merged Fact Table.

b) Steps (3), (4), (5) satisfy Measure and Attribute Entity Preservation: All the attributes contained in the Fact or Dimension Tables are represented in the Merged Table (Fact or Dimension) through equality or similarity mapping.

c) Steps (6) and (7) satisfy Attribute Property Value Preservation: Value properties of attributes are represented for each of the Fact or Dimension Tables.

d) Step (8) satisfies Slowly Changing Dimension Preservation and Tuple Containment Preservation: Entity representations from the different data marts are included in the merged dimensions.

e) Steps (9), (10) satisfy Tuple Containment Preservation: Tuple data values from each of the data marts are populated in the merged data warehouse.

C. Conflicts Identification and Resolution

The integration of meta-data models are generally coupled with different forms of conflicts in either the instance schema structures or instance data. These conflicts are resolved through different propositions from the algorithm and based on the semantic representation of the meta-data models and their instance schemas. In our integration approach, we identify and propose resolution measures for these conflicts that are encountered during merging.

Identifier Conflicts. These conflicts arise as a result of the same identifier for different real-world entities in the merged dimension. These categories of conflicts are practically exposed as a result of the possibility of different entities from the integrating data marts having the same surrogate key identifier in their individual dimensions. A resolution measure for these conflicts is explained in Example 5.

Example 5. *Suppose we aim to merge the employee dimensions into a single merged dimension, using DimPolicyEmployee and DimInsuredPolicyEmployee from Policy and Claims data marts, respectively. In such an integration procedure, it happens that Employee P from DimPolicyEmployee and Employee Q from DimInsuredPolicyEmployee have the same identifiers of a surrogate key. There is the need to resolve such a conflict, in the algorithm, by preserving the surrogate key identifier in the preferred data mart and re-assign a new surrogate key identifier for the non-preferred data mart(s).* ■

Entity Representation Conflicts. These conflicts arise as a result of the multiple representations of the same real-world entity in the merged dimension by the different identifiers. This occurrence is traced to different representations of surrogate key identifiers from different dimensions for the same real-world entity in the merged dimension. Following on Example 1, a proposed resolution measure, outlined in the merged algorithm, will be to perform a de-duplication of the conflicting entities, by preserving the entity from the preferred data mart as the sole representation of the real-world entity in the merged dimension.

Attribute Property Type Conflicts. These forms of conflicts occur as a result of the existence of different attribute property values from the integrating attributes into a merged attribute. Using Example 5, a merged attribute for *HireStatus* and *EmployeeStatus* from *DimPolicyEmployee, DimInsuredPolicyEmployee*, respectively, will hold a data type value of, say *varchar(1)*, being the *UNION* of integrating attribute data types for *char(1)* and *bit* data types from *HireStatus* and *EmployeeStatus*, respectively. We resolve these conflicts by using the attribute data types as defined in the mapping model.

IV. IMPLEMENTATION AND EVALUATION

In this section, we discuss the implementation and evaluation work based on the integration methodology and formulated merge algorithm. We present our implementation framework and the procedures we followed, and we discuss and analyze the evaluation results.

A. Implementation

We implemented our methodology using 2 different data warehouses, for the Insurance business and Transportation services. The Insurance data consisted of 2 initial data marts. These were Policy and Claims data marts. The Policy and the Claims data marts contained 7 and 10 dimension table schemas, respectively. The Policy fact table schema contained instance data of 3070 tuples of data, whilst the Claims fact contained 1144 tuples of data. The Transport data set contained 3 data marts. These were Frequent Flyer, Hotel Stays, and Car Rental data marts. Their fact tables contained 7257, 2449, 2449 tuples of data for Frequent Flyer, Hotel Stays, and Car Rental, respectively. The data marts resided in a Microsoft SQL Server DBMS. Each entity representation in the dimensions was identified by a unique surrogate key and with a clustered indexing as created on the primary key.

The schema matching and mapping models discovery procedural steps were implemented using IBM Infosphere Data Architect [9] [10]. This tool incorporated the schemas of the data mart source repositories, together with their contained instance data. The schema matching step was implemented using the set of algorithms incorporated in the application software. The algorithms were configured by sequentially manipulating the order of execution, configuration of rejection threshold, sampling size and sampling rate. The manipulations of these configurations for finding mapping correspondences were based on an iterative procedure of inspection. With regards to the mapping models discovery and modelling step, the adoption of GLAV mappings enabled the inclusion of all attributes for each mapping formulation of fact and dimension table attributes. Moreover, complex transformation expressions were derived for multi-cardinality mappings. An output file in a Comma Separated Values (.csv) format was later generated, which contained the mapping definitions based on the tables, their attributes, and the attribute property values from each of the data marts. The merge algorithm was implemented using C# .Net programming.

B. Evaluation

Our evaluation analyses were primarily based on the single consolidated data warehouse from the formulated merge algorithm, in Section III.B, as against the independent data marts. We compared both the outputs of the query processing on the data marts and the generated data warehouse. We first ran a formulated query on one or more data marts, and afterwards ran the same query on the generated data warehouse. With this ordering, we are able to effectively compare the results from the data marts and the single consolidated data warehouse.

Evaluation Criteria and Analysis. We evaluate the outcome of the experiments conducted based on a set of criteria based on the guidelines proposed by Pedersen et al. [8]. We performed a gap analysis on their study and then adapted the correctness of data values, dimensionality hierarchy, and rate of query processing as criteria.

The metrics that we used in evaluating these criteria for query processing were recall, precision, and accuracy. These were proposed by Junker et al. [13] to evaluate the performance of database query processing and information retrieval. Recall is computed by the number of tuples retrieved from a data mart divided by the number of tuples that should have been retrieved from the generated data warehouse from each original data mart. Precision is computed by the number of tuples retrieved from a data mart divided by the number of tuples that were retrieved from the single consolidated data warehouse, per the data mart. Accuracy is determined by the degree of validity or exactness of the data values generated from a query posed to the data warehouse in comparison to the data values retrieved from a data mart.

All formulated queries that were posed to the data warehouse were based on fact and dimension attributes from all the data marts. For recall, an evaluation of 100% was trivially attained and verified. The verification was based on the assertion that the merge algorithm fulfilled the MCRs of measure and attribute entity preservation and tuple containment preservation.

Precision evaluation was very important, as it measured the proportion of relevant and non-relevant tuples that were retrieved based on a formulated query. This gives us insight into the composition of our merged data warehouse, in terms of the level of integration of related data from multiple sources. Deducing from the precision values, a higher rate was attained for all formulated queries that were posed against the data warehouse. For cases of dimensions that were only related to some specific data marts, a formulated query against the fact and these dimension tables yielded a very high precision rate. This was as a result of the retrieval of few non-relevant tuples. An example query was, "*What insurance claimant employment type receives the most claims processed for the current Calendar Season*"? Conversely, for queries on dimensions that related to all data marts, an average precision rate was observed where a considerable number of non-relevant tuples were retrieved in reference to a particular data mart. An example query was, "*What type of Policy Coverage is most popular? What are the trends since the 2nd Calendar Quarter.*"

Figures 3 and 4 show the precision evaluation for Insurance and Transportation data warehouses, respectively. In Figure 3, an average rate of 86% was achieved for the queries posed to dimensions only related to the Claims data mart. The precision rate increases significantly with an increase in the tuples in these dimensions, as more relevant tuples are generated. This is evident in queries 1 to 7. In terms of corresponding dimensions for all data marts, processed queries generated an average rate of 51% and 49% for Claims and Policy data marts, respectively, as highlighted in queries 8 to 12. In Figure 4, an average precision rate of 72%, 74%, and 83% were attained for Hotel Stays, Car Rental, and Frequent Flyer data marts, respectively, for the set of formulated queries posed. In summary, we were able to provide the user with details regarding the proportion of the data in the merged data warehouse that originate from a specific source. This holds important practical value, for data warehouse practitioners, who want to be able to have statistics regarding the composition of the merged data.

In terms of accuracy, we achieved a 100% return rate of valid and exact data values from the data warehouse in comparison to each individual data mart. This was affirmed based on the merge algorithm fulfilling MCRs of Tuple Containment Preservation and Measure and Attribute Entity Preservation. Additionally, the adoption of GLAV mapping model enabled the processing of exact and sound queries on the data warehouse.

We also analyzed the rate of query processing to ensure that queries posed to the data warehouse are of optimal rate. With an integration of instance data from the data marts, a considerable volume of expected data cannot be overemphasized in the data warehouse. We recorded the query response time for an average of 20 query executions for each of the data sets. These queries were processed on a single 3.20 GHz processor with a 4 GB of RAM.

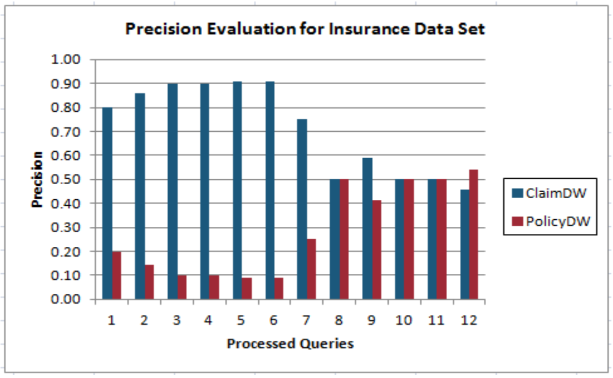


Figure 3. Precision for Insurance Data Set

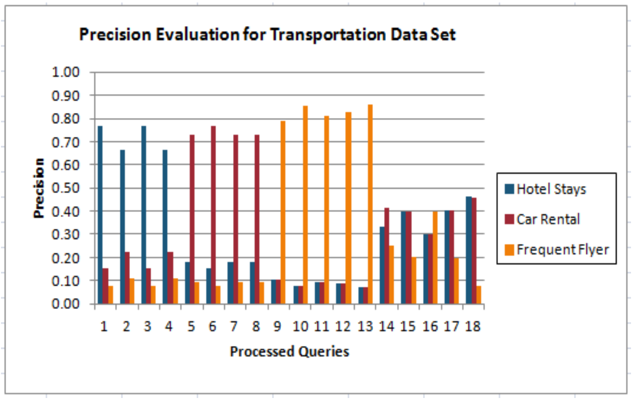


Figure 4. Precision for Transportation Data Set

TABLE I. SUMMARY OF AVERAGE QUERY RESPONSE TIME AND VARIANCES

Data Set	**Average Query Response Time and Variances**		
	Data Mart / Data Warehouse	***Avg. Query Response (ms)***	***Variance From Integrated Data Warehouse (ms)***
Transportation	Car Rental	26.70	63.95
Transportation	Hotel Stays	27.10	63.55
Transportation	Frequent Flyer	70.95	19.70
Transportation	DataWarehouse	90.65	0.00
Insurance	Policy	29.65	19.60
Insurance	Claim	13.75	35.50
Insurance	DataWarehouse	49.25	0.00

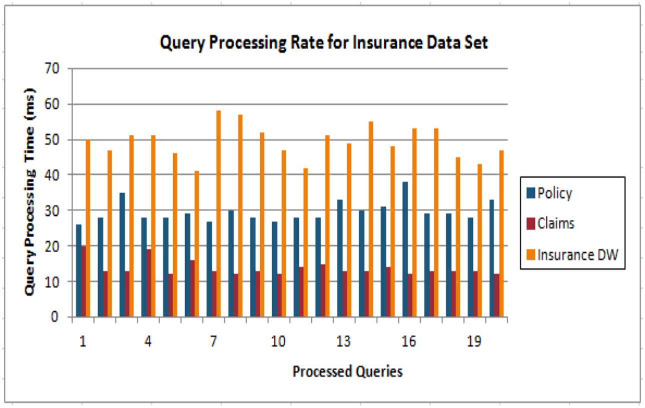


Figure 5. Query Processing Rate for Insurance Data Set

We further computed the variance of the average query rate per data mart as it quantitatively differs from its consolidated data warehouse. Our evaluation showed that queries generally ran at almost the same rate or slightly higher than when posed against the data mart sources.

The query execution durations for the data marts and data warehouses for the Insurance data set are shown in Figure 5. It can be deduced from these data values that the query rate for the data warehouses were appreciable taking note of the compared values generated from the data marts. In some cases, such as queries 7 and 8, the rates were a bit higher due to higher level of aggregation and increased number dimension attributes involved in data values retrieved. We present a summary of the variances in the average query response time for the data marts in comparison to the respective data warehouse. Table 1 shows the query response (in milliseconds) for the Insurance and Transportation data sets.

V. CONCLUSION

This paper presents a methodology for the merging of multidimensional data models using star schemas instances. We formulated a merge algorithm for integrating disparate data marts into a single consolidated star schema data warehouse. We further identified and outlined the resolution of likely conflicts that may be encountered when merging data marts. Moreover, we outlined the satisfaction of some technical merge correctness requirements for integrating data marts into a data warehouse.

Analyses of our evaluation showed that the rates of recall, precision and accuracy of the data values retrieved from the generated data warehouse are high and noticeable. Our approach, thus, provides data warehouse researchers and practitioners with procedures, criteria, and exact measures as to how successful an integration process is achieved.

A number of future research directions remain. The potential enrichment of the mapping language by modelling the functional dependencies between the attributes of the fact and dimension tables is an interesting future direction. Additionally, incorporation of data mart level integrity constraints into the data warehouse needs to be investigated further. We also envisage the extension of the methodology to handle snowflake and fact constellation multidimensional schema models.

ACKNOWLEDGEMENT

This work was supported by grants from the Natural Sciences and Engineering Research Council (NSERC) Strategic Network on Business Intelligence (BI).

REFERENCES

[1] R. A. POTTINGER and P. A. BERNSTEIN, "Merging Models Based on Given Correspondences," VLDB 2003: 826-873 & Microsoft Research 2000: MSR-TR-2000-53.
[2] M. LENZERINI: Data Integration, "A Theoretical Perspective," PODS 2002:233-246.
[3] P. A. BERNSTEIN and S. MELNIK, "Model Management 2.0: Manipulating Richer Mappings," SIGMOD 2007:1-12.
[4] S. MELNIK, "Generic Model Management: Concepts and Algorithms," Springer LNCS 2967. (2004).
[5] L. CABIBBO and R. TORLONE, "Integrating Heterogeneous Multidimensional Databases," SSDBM 2005:205-214.
[6] L. CABIBBO and R. TORLONE, "Dimension Compatibility for Data Mart Integration," SEBD 2004:6-17.
[7] D. RIAZATI, J. A. THOM and X. ZHANG, "Inferring Aggregation Hierarchies for Integration of Data Marts," DEXA 2010:96-110.
[8] T. B. PEDERSEN, C. S. JENSEN and C. E. DYRESON, "A Foundation for Capturing and Querying Complex Multidimensional Data," Elsevier Sci. 26(5):383-423 (2001).
[9] R. FAGIN, L. M. HAAS, M. A. HERNÁNDEZ, R. J. MILLER, L. POPA and Y. VELEGRAKIS, "Clio: Schema Mapping Creation and Data Exchange," Conceptual Modelling: Foundations and Applications 2009:198-236.
[10] IBM: IBM Infosphere Data Architect 7.5.3.0 – Finding Relationships.http://publib.boulder.ibm.com/infocenter/idm/v2r1/index.jsp?topic=/com.ibm.datatools.metadata.mapping.ui.doc/topics/iiymdadconfiguring.html (Accessed–Dec. 8, 2012).
[11] B. TEN CATE and P. G. KOLAITIS, "Structural Characterizations of Schema-Mapping Languages," ICDT 2009:63-72.
[12] D. KENSCHE, C. QUIX, X. LI, Y. LI and M. JARKE, "Generic Schema Mappings for Composition and Query Answering," Data Knowl. Eng. (DKE). 68(7):599-621 (2009).
[13] M. JUNKER, A. DENGEL and R. HOCH, "On the Evaluation of Document Analysis Components by Recall, Precision, and Accuracy," ICDAR 1999:713-716.

Considerations about Implementation Variants for Position Lists

Andreas Schmidt*† and Daniel Kimmig†
* *Department of Computer Science and Business Information Systems,*
Karlsruhe University of Applied Sciences
Karlsruhe, Germany
Email: andreas.schmidt@hs-karlsruhe.de
† *Institute for Applied Computer Science*
Karlsruhe Institute of Technology
Karlsruhe, Germany
Email: {andreas.schmidt, daniel.kimmig}@kit.edu

***Abstract*—Within "traditional" database systems (*row store*), the values of a tuple are usually stored in a physically connected manner. In a *column store* by contrast, all values of each single column are stored one after another. This orthogonal storage organization has the advantage that only data from columns which are of relevance to a query have to be loaded during query processing. Due to the storage organization of a *row store*, all columns of a tuple are loaded, despite the fact that only a small portion of them are of interest to processing. The orthogonal organization has some serious implications on query processing: While in a traditional *row store*, complex predicates can be evaluated at once, this is not possible in a column store. To evaluate complex conditions on multiple columns, an additional data structure is required, the so-called *Position List*. At first glance these *Position Lists* can easily be implemented as an dynamic array. But there are a number of situations where this is not the first choice in terms of memory consumption and time behavior. This paper will discuss some implementation alternatives based on (compressed) bitvectors. A number of tests will be reported and the runtime behavior and memory consumption of the different implementations will be presented. Finally, some recommendation will be made as to the situations in which the different implementation variants for *Position Lists* will be suited best. Their suitability depends strongly on the selectivity of a query or predicate.**

***Keywords*–Column stores; PositionList implementation variants; bitvector; compression; run length encoding; performance measure**

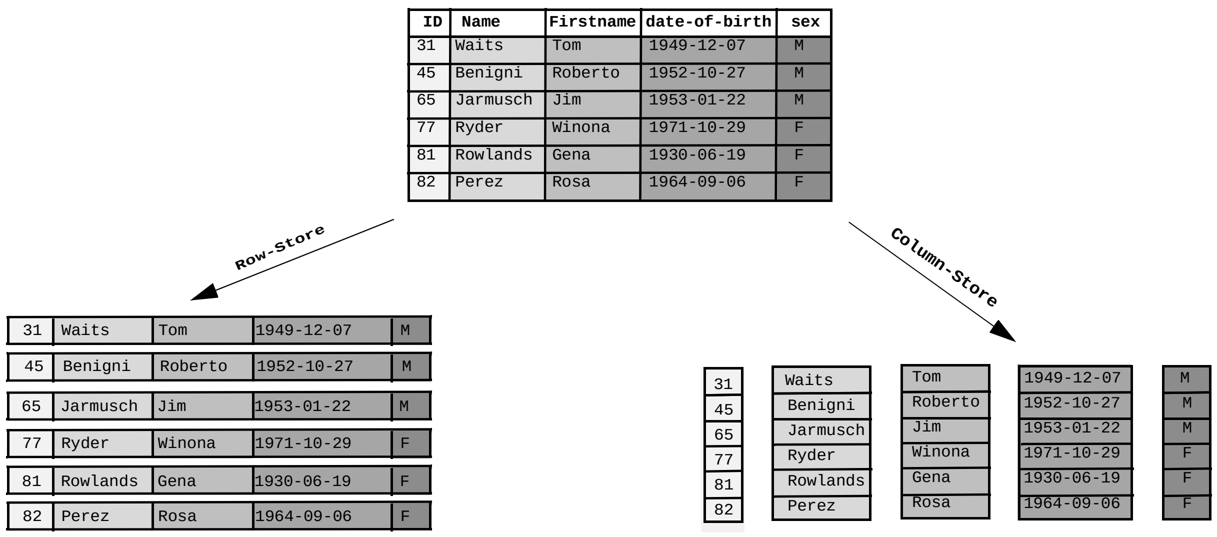
I. Introduction

Nowadays, modern processors utilize one or more cache hierarchies to accelerate access to main memory. A cache is a small and fast memory which resides between main memory and the CPU. In case the CPU requests data from the main memory, it is first checked, whether these data are already contained in the cache. In this case, the item is sent directly from the cache to the CPU, without accessing the much slower main memory. If the item is not yet in the cache, it is first copied from the main memory to the cache and then sent to the CPU. However, not only the requested data item, but a whole cache line, which is between 8 and 128 bytes long, is copied into the cache. This prefetching of data has the advantage of requests to subsequent items are being much faster, because they already reside within the cache.

Depending on the concrete architecture of the CPU, the speed gain when accessing a data set in the first-level cache is up to two orders of magnitude compared to regular main memory access [1]. This means that when a requested data item is already in the first-level cache, the access time is much faster compared to the situation, when the data item must be loaded from the main memory (this situation is called a cache miss). The use of special data structures which increase cache locality (the preferrred access of data items already residing in the cache) is called *cache-conscious* programming.

Column stores take advantage of this prefetching behavior, because values of individual columns are physically connected. Therefore, they often already reside in the cache when requested, as the execution of complex queries is processed column by column rather than tuple by tuple. This difference between a "traditional" *row store* and a *column store* is illustrated in Figure 1. In the upper part of the figure, a relation, consisting of six tuples, each with five columns, is shown. The lower part of the figure shows the physical layout of this relation on disk or in the main memory. On the left side, the row store layout, is represented. The row store stores all values of one tuple in a physically connected manner. In contrast to this a column store contains all values of each single column one after another.

This also means that the decision whether a tuple fulfills a complex condition on more than one column is generally delayed until the last column is processed. Consequently, additional data structures are required to administrate the status of a tuple in a query. These data structures are referred to as *Position List*s. A *Position List* stores information about matching tuples. The information is stored in the form of a Tuple IDentifier (TID). The TID is nothing more than the position of a value in a column. Execution of a complex query generates a *Position List* with entries of the qualified tuples for every simple predicate.



ID	Name	Firstname	date-of-birth	sex
31	Waits	Tom	1949-12-07	M
45	Benigni	Roberto	1952-10-27	M
65	Jarmusch	Jim	1953-01-22	M
77	Ryder	Winona	1971-10-29	F
81	Rowlands	Gena	1930-06-19	F
82	Perez	Rosa	1964-09-06	F

Figure 1. Comparison of the layouts of a row store and a column store (from [2])

Complex predicates on multiple columns can be evaluated in two different ways. First, as shown in Figure 2, the predicates can be evaluated separately, and in a subsequent step, the resulting *Position Lists* can be merged. The advantage of this variant is, that the evaluation of the predicates can be done in parallel. A drawback of this solution is, that all column values must be evaluated.

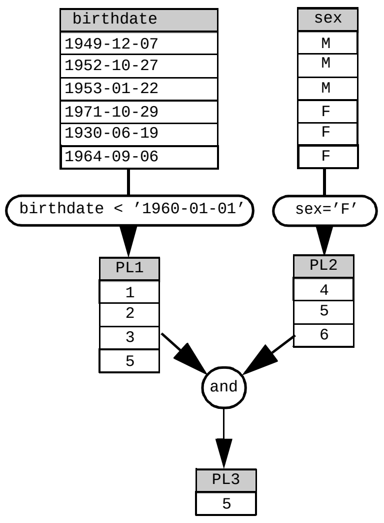


Figure 2. Isolated evaluation of predicates on their corresponding *Position Lists* and subsequent merging of the resulting *Position Lists* (from [2])

In contrast to this, the evaluation of the query can also be done sequentially as shown in Figure 3. In this case, a *Position List*, representing the result of a previously evaluated predicate, is an additional input parameter for the evaluation of the second predicate. Not all column values have to be evaluated, but only those for which an entry in the first *Position List* exists. The drawback of this solution is the strict sequential program flow and a slightly more complex execution which may probably cause more cache misses compared to the parallel version.

Which of the variants is better suited depends on the boundary conditions of the query.

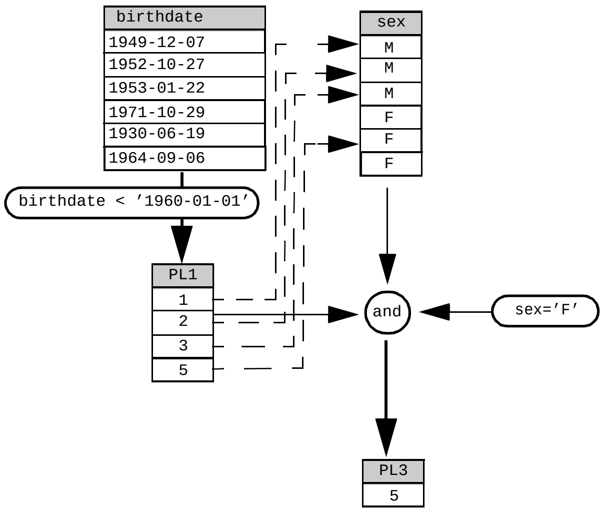


Figure 3. Iterative evaluation of predicates, using *Position Lists* as additional parameters

In previous work, we developed the Column Store Toolkit (CSTK) [2] and also used this toolkit as a starting point for further research in the field of optimizing SQL queries, based on a column store architecture [3].

The main objective of this paper is to present an in-depth analysis of different implementation variants of *Position Lists* and to demonstrate their advantages and disadvantages in different situations in terms of runtime behavior and memory consumption.

The paper is structured as follows: In the next section, we will discuss some specific details of *Position Lists*. Then, the most important components of the CSTK will be introduced. After that, a number of experiments with respect to runtime behavior and memory consumption will be performed in the main part. Finally, results will be summarized, and an outlook will be given on future research activities.

II. Related Work

Various publications compare the performance of column stores with that of row stores for different workloads [4], [5], [6]. In contrast to this, [7] examines different execution plan variants for column stores, while [8] considers the impact of compression. Following the work in [7], we examine different implementation variants for the underlying data structures and algorithms of the operations used in the execution plan of a query.

III. Position Lists

From a logical point of view, a *Position List* is nothing else than an array or list with elements of the data type *unsigned integer* (UINT) as far as structure is concerned. However, it has a special semantics. The *Position List* stores TIDs. A *Position List* is the result of a query via predicate(s) on a *Column*, where the actual values are of no interest, whereas the information about the qualified data sets is desired. *Position List*s store the TIDs in ascending order without duplicates. With other words, a *Position List* stores the information for each tuple no matter whether if it belongs to a result (so far) or not.

A. *Operations on* Position Lists

The fundamental logical operations on *Position Lists* are appending TIDs at the end (write operation), iterating over the list of TIDs (read operation), and performing *and/or* operations on complete lists.

Further operations that are mainly based on this basic functionality, include the materialization [7] of the corresponding values from the requested columns, the storage of the whole list or parts of it in a file and the import from a file.

B. *Implementation Variants*

Based on the logical structure and behavior discussed above, the first intuitive implementation of a *Position List* is using a dynamic array (an array of flexible size) of unsigned integer values. The advantage of this variant is, that the implementation is straight-forward and the storage of the TIDs is cache-conscious [9], [10] in the context of the above-mentioned operations like iterating, storing, and *and/or* operations.

As *Position List*s store the TIDs in ascending order without duplicates, typical *and/or* operations are very fast, as the cost for both operations is $O(|Pl_1| + |Pl_2|)$.

One big drawback of the implementation as a dynamic array is the fact, that the lists may be very large. This is especially true for predicates over multiple columns, where no predicate has a *high selectivity*. *High selectivity* means, that only a small number of tuples qualify the condition. A *low selectivity*, by contrast, means that a lot of tuples satisfy the condition. Typical predicates of low selectivity are the “family status” or the “gender” of a person. Let us consider a conjunctive query consisting of 6 predicates on different columns. Each single predicate has a selectivity of up to 50% (i.e. gender, family status, etc.). The overall selectivity of the query is about 1.5% of the original number of tuples, but the size of the cardinality of the individual *Position Lists* is up to 50% of the original table. Starting with the predicate of the highest selectivity and iteratively examining the values of tuples from the subsequent columns which qualified previously (see Figure 3) can reduce this problem. However, if no or only vague information about the selectivity of the different predicates is available, this can be a serious problem. Figure 4 shows the size of a *Position List* in megabytes with respect to the selectivity (density) of the predicate for a 100 million tuple table. In the worst case, the resulting *Position List* can be bigger than the original column (e.g. for columns with binary values or a small number of possible values only).

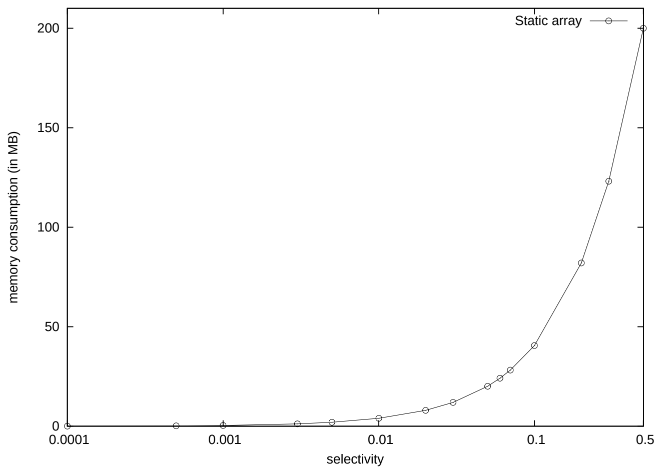


Figure 4. Memory consumption of a *Position List* implemented as array (logarithmic scale on x-axis)

The problem of the unpredictable size of the intermediate *Position Lists* can be prevented by using a bitvector to represent the *Position List*. Here, every tuple is represented by one bit. A value of ’1’ means, that the tuple belongs to the (intermediate) result, a value of ’0’ means that the tuple does not belong to the result.

This has the advantage of the size of a *Position List* being exactly predictable, independently of the selectivity of the predicate. The selectivity only has impact on how many bits are set to '1'. Moreover, the two important operations *and* and *or* can be mapped on the respective primitive processor commands, which makes the operations fast. If *Position List*s are sparse, bitvectors can also be compressed very well using run length encoding (RLE) [11]. The idea behind RLE encoding is, that if only a small number of bits are one, the '0' bits are not stored physically, but only the number of '0' bits are stored.

The *Word Aligned Hybrid*-algorithm (WAH) [12] uses this principle and distinguishes between two word types: *fills* and *literals*. The two word types are distinguished by the most significant bit, so 31 (63) bits remain for the stored bits per word or the length field. A *literal* is a word consisting of 31 (63) bits of which at least one bit is '1'. A *0-fill* consists of a multiple of 31 (63) '0' bits which are stored in one word. The maximum number of '0' bits which can be stored in one word is $31 * 2^{31}$ (resp. $63 * 2^{63}$ for the 64-bit version).

The necessary operations like iterating, *and*, *or* can be performed on the compressed lists, thus avoiding a temporary decompression of the compressed representation. In the context of this paper, the bitvector implementation of the WAH algorithm and a simple plain uncompressed bitvector implementation are used. The WAH algorithm is considered to be one of the fastest algorithms for performing logical *and/or* operations on compressed bitmaps.

IV. The Column-Store-Toolkit

The Column Store Toolkit (CSTK) was deveolped as a toolkit with a minimum amount of basic components and operations required for building column store applications. These basic components were used as catalysts for further research into column store applications and for building data-intensive, high-performance applications with minimum expenditure.

The main focus of our components is on modeling the individual columns, which may occur both in the secondary store as well as in main memory. Their types of representation may vary. To store all values of a column, for example, it is not necessary to explicitly store the TID for each value, because it can be determined by its position (dense storage). To handle the results of a filter operation, however, the TIDs must be stored explicitly with the value (sparse storage).

Another important component is the already discussed *Position List*. Just like columns, two different representation forms are available for main and secondary storage. In this paper, it is concentrated on the main memory behavior of the *Position Lists*.

To generate results or to handle intermediate results consisting of attributes of several columns, data structures are required for storing several values (so-called multi columns). These may also be used for the development of hybrid systems as well as for comparing the performance of row and column store systems.

The operations mainly focus on writing, reading, merging, splitting, sorting, projecting, and filtering data. Predicates and/or *Position List*s are applied as filtering arguments.

Figure 5 presents an overview of the most important operations and transformations among the components. The arrows show the operations among the different components (ColumnFile, Dense-/Sparse ColumnArray, PositionList, and PositionListFile). For a detailed description of the operations, see [2].

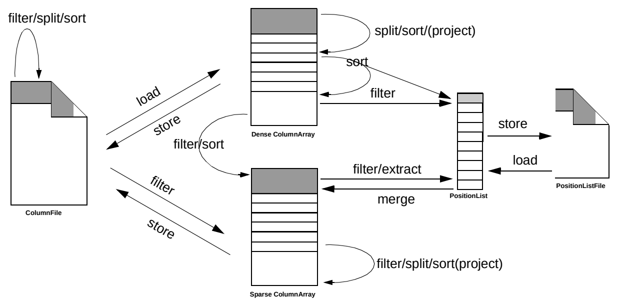


Figure 5. CSTK: Components and Operations (from [2])

V. Measurements

A. Elemental Operations

1) Memory consumption: In a first experiment we compare the size of the different data structures with respect to memory consumption. As shown in Figure 6, the behavior of the dynamic array implemetation is quite good for very small selectivities, but changes for the worse for medium and high densities. Uncompressed bitmaps (plain bitvector, WAH-uncompressed) behave independently for all densities, their size is determined by the number of tuples in a table only. Compressed bitmaps show a very good behavior for all densities. If the selectivities get low, they behave like uncompressed bitmaps (compared to a pure uncompressed implementation of a bitvector, there will be a slight overhead of 1/32 resp. 1/64.). From a selectivity of about 3% the array has a higher memory consumption than the uncompressed bitvector.

2) Iterating over TIDs: In the next experiment, we examine the runtime behavior of the two elemental operations:

- Appending TIDs on a *Position List*
- Iterating over the TIDs in a *Position List*.

These two operations are heavily used in the implementation of the CSTK components.

We implement a simple bitvector class by our own (without compression facility), and also use the well-known WAH algorithm. The overhead of the uncompressed representation of WAH is quite small, in terms of both runtime and memory consumption.

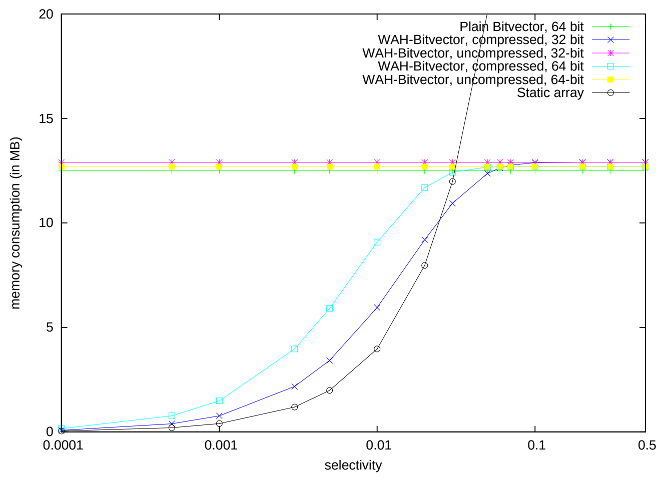


Figure 6. Memory consumption of different implementation alternatives for *Position Lists*

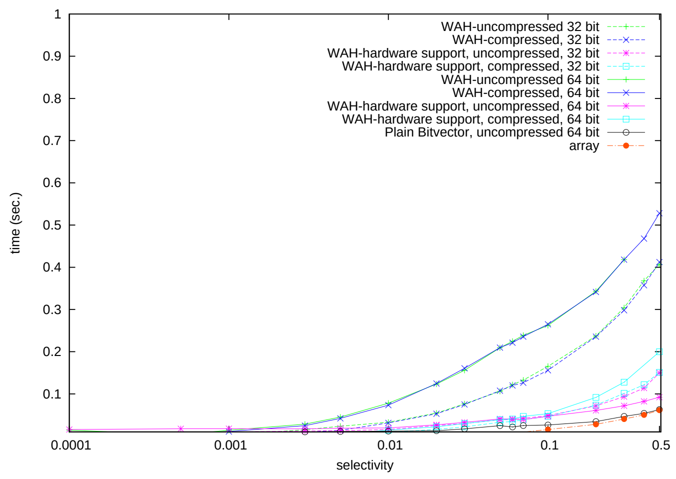


Figure 7. Measured time to iterate over 100 million data sets with different densities

In contrast to the original implementation of the WAH algorithm, we also use hardware support for special operations. The Leading Zero Count Instruction (LZCNT) is used to find the '1' bits inside a processor word. This leads to a performance advantage of a factor of 3 compared to the orginal WAH version.

In our first experiment we take a table of 100 million tuples and formulate predicates with different selectivities between 0.0001 and 0.5. The TIDs of the qualified tuples are then stored in the different representation forms (plain bitvector, WAH bitvector uncompressed/compressed with 32 and 64 bit word size, array). After that, we measured the time to iterate over all the stored TIDs.

Figure 7 presents an overview of the runtime behavior for our different implementations:

The fastest implementation for all selectivities is the dynamic array. In contrast to this, the worst runtime behavior is reached from the standard WAH iterator (both 32- and 64 bit version), which therefore will not be considered any further. More interesing values come from the iterators which use the *__builtin_clzl* instruction from the gnu compiler family, which is mapped on the LZCNT instruction if available (the plain bitvector implementation was the fastest).

Two more detailed graphs are given in Figure 8 and Figure 9. Here the static array implementation and the LZCNT supported iterators are considered for high and low selectivity, respectively.

While Figure 8 shows the details for selectivities between 0.0001 and 0.05, Figure 9 shows the lower selectivities between 0.05 and 0.5. One interesting point is, that with low selectivity (Figure 9) the hardware-supported iteration behaves differently for the 32 and 64 bit WAH version. While the compressed version is faster for the 32 bit version, the opposite is true for the 64 bit version. This behavior can be founded with the better compression ratio of the 32 bit version for lower selectivities, which leads to a smaller amount of memory which has to be loaded into the CPU.

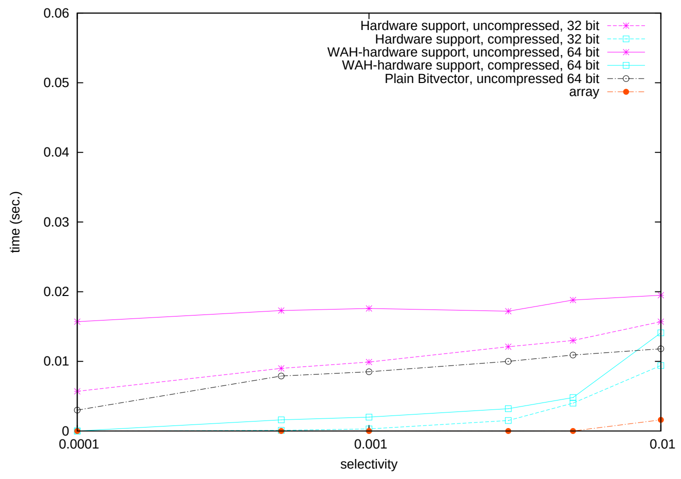


Figure 8. Measured time to iterate over 100 million data sets with high selectivity

It is obvious that the time for the uncompressed bitvector versions is the least dependent on the selectivity. This can be explained by the dominating time for loading the data from the main memory into the CPU. For all other implementations the influence of the descending selectivity is higher.

Although the static array implementation is faster by a factor of five for some selectivities, we also have to consider that in absolute values, the time of iterating over a bitlist of 50 million entries (selectivity: 0.5) is between 0.08 seconds (array) and 0.26 (64-bit, hardware supported, uncompressed). This is not bad and probably not such a dominating factor compared to the memory consumption of the different implementations shown in Figure 6.

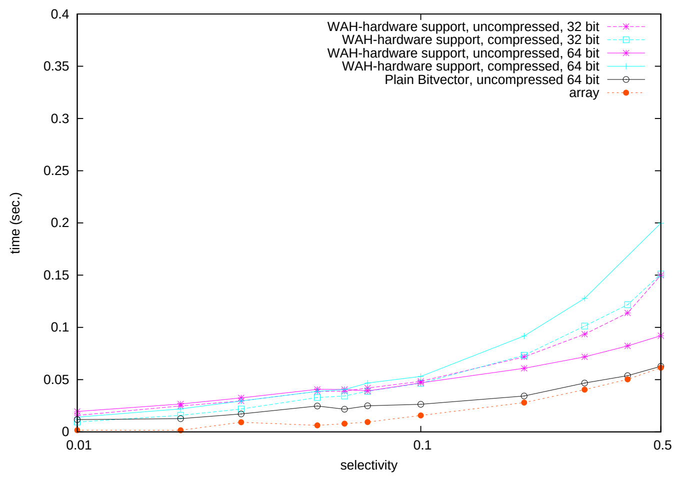


Figure 9. Measured time to iterate over 100 million data sets with low selectivity

3) Writing TIDs: In our next experiment we analyse the time to write TIDs in the different implementation variants. This operation is done every time, when a predicate is evaluated against a column value and found to be "true".

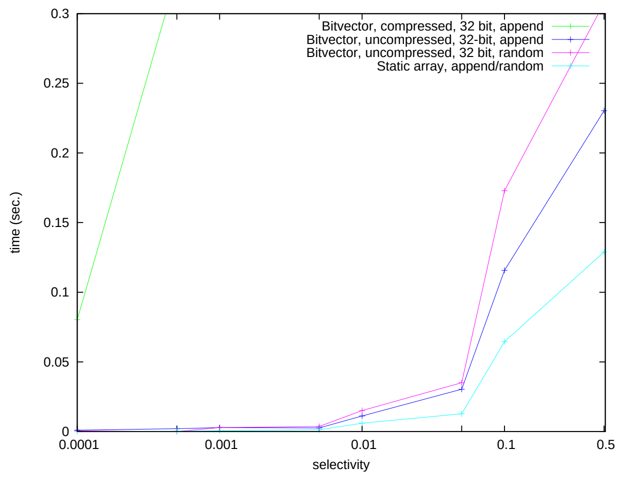


Figure 10. Measured time to write TIDs in different implementation variants for *Position Lists*

As a basic condition we can assume that writing of TIDs is mostly done in in the append mode. The reason is, that when evaluating a predicate on a column, this is done sequentially value by value with increasing TID values. In some complex situations, however, TIDs must be written in random order (i.e. after a previous sort operation on a column).

The results for this experiment are shown in Figure 10. Again, the storage as an array of UINT values is the fastest solution for all selectivities. This is true for the append mode and the random order mode (from the implementation point, there is no difference between the two variants). The uncompressed bitvector turned out to be the second best solution. Based on the implementation, the solution in append mode is slightly faster than the random write mode. This can be motivated by the fact that the number of cache misses is lower in the append mode, than in the random mode. This characteristic increases with decreasing selectivity (0.05 and above), because the probability of the next TID being close enough to the previous TID and so the corresponding memory segment (the bit) being already in the cache, increases.

Compressed bitvectors behave worse. The reason for random access is that with every insertion of a TID, the compressed bitvector must be reorganized, which often has an influence up to the end of the whole compressed bitvector. This behavior occurs in the append and random mode for the WAH implementation (the WAH implementation has no special append mode, but only a *setBit(uint pos, bool value)* method to set a bit at an arbitrary position). However, the append mode could be implemented in a much more efficient way. The basic concept for the algorithm is represented in Figure 11. The idea of this implementation is, that in the append mode only the last two words (LL: Last Literal, LF: Last Fill) must be considered: The last but one word, which is a literal, and the last, which is a 0-fill. Either the TID sets a bit in the last literal word, or the last fill must be splitted into two fills, with a literal in between (with holds the TID). From the time behavior, we expect that this solution has about half of the performance of the uncompressed version (where we only have to jump to the corresponding word and set the bit), but is by far better than the general purpose *setBit* method from the WAH implementation.

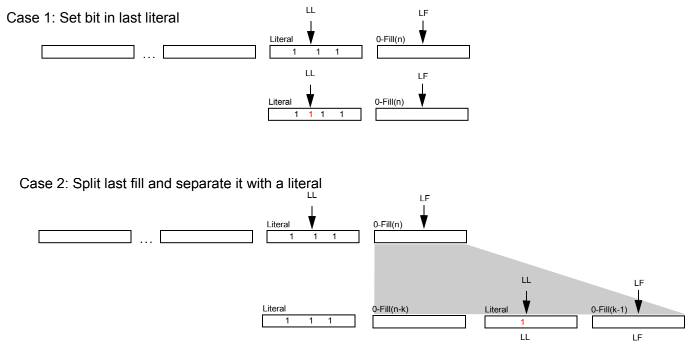


Figure 11. Appending TIDs in a compressed bitvector

4) AND *operations on* Position Lists*:* Next, we perform an experiment to measure the time for *AND* operations. This is one of the basic operations performing the "WHERE" part of a query on a column store, where two or more *Position Lists* are *AND*ed (same with *OR*).

Figure 12 shows the results for the *AND* operation. As you can see, the time for *AND*ding two uncompressed bitvectors (both, the plain bitvector implementation and the uncompressed WAH bitvector) is mostly independent of the selectivity. This can easily be understood, because the length of the vector is also independent of the selectivity and so the the *AND* operation consists of a constant number of *and* instructions in the CPU. The slight overhead of the WAH implementation can be explained by the more complex algorithm and the additional memory consumption of 1/32 compared to the plain uncompressed bitvector.

The more interesting lines are the compressed bitvector and the array. While the array performs best for selectivities of 0.02 and higher, it degrades for lower selectivities. This is a little surprising, because the array implementation was one of the fastest in the previous experiments (iterating and writing TIDs). The degeneration can be explained by the caching strategies of modern CPUs. In the case of low selectivities, the two arrays grow and there is a cut-throat competition for places in the processor cache, which is why many cache misses result.

The compressed bitvector outperforms the uncompressed version for high selectivities (0.007 and above), because of its more compact representation and the ability to skip all the fill words completely. With lower selectivities the fills get shorter and disappear later on. Hence there is no advantage compared to the uncompressed representation. In this situation, the more complex algorithm is another drawback and leads to more instruction cache misses compared to the uncompressed version.

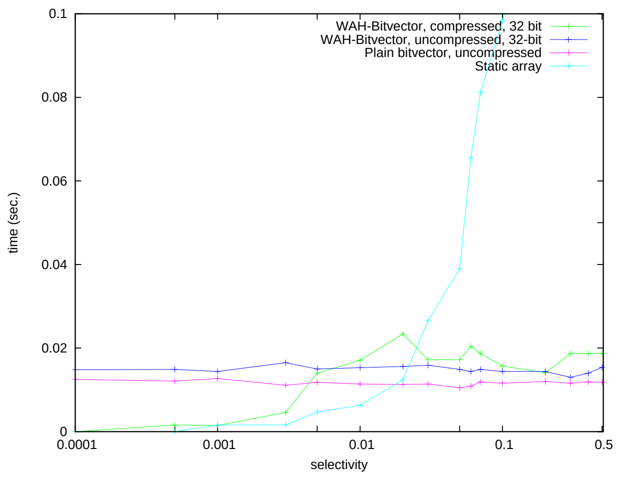


Figure 12. Measured time for *AND*ing two *Position Lists* with different implementations

VI. Conclusion and Future Work

The choice of the right data structure and algorithm for implementing *Position Lists* is not an easy task. It largely depends on the selectivity of the predicates and the operations to perform. Especially for low selectivities, the choice of the right solution is critical as was shown by the experiments.

The data structure of a dynamic array of unsigned integer values is outperformed by the uncompressed bitvector implementations by up to two orders of magnitude for low selectivities. On the other hand, it is very good choice at high selectivities.

Uncompressed bitvectors have a predictable behavior for all selectivities, but are again outperformed by compressed bitmaps and arrays for very high selectivities.

If no information about the expected selectivity is available, using an uncompressed bitvector probably is a good choice. Depending on the selectivity and the used algorithm, the execution time ranges over three orders of magnitude and the uncompressed bitvector is of moderate performance.

Next, the “append-mode” will be used for setting bits in a compressed bitvector. With this implementation. we will be able to perform more experiments with repect to the runtime behavior of complex conditions in both the sequential (Figure 3) and parallel (Figure 2) mode. After finishing this task, we will perform some more experiments using the different implementations together with our toolkit components to measure the time behavior of ouf our components with more complex queries like those from the TPC-H [13] benchmark.

Another interesting point is the usage of different data structures in one query. This may sound strange, but the conversion of compressed into uncompressed bitvectors and vice versa is very fast compared to the penalty of using the wrong algorithm/data structure. After evaluating the first predicates using uncompressed bitmaps (which perform well for low selectivities), the overall selectivity will increase and use compressed bitvectors could be advantageous.

It has to be kept in mind that the ultimate goal is the development of a query optimizer for a *column store* [3].

References

[1] P. A. Boncz, M. Zukowski, and N. Nes, “Monetdb/x100: Hyper-pipelining query execution,” in *CIDR*, 2005, pp. 225–237.

[2] A. Schmidt and D. Kimmig, “Basic components for building column store-based applications,” in *DBKDA’12: Procceedings of the The Fourth International Conference on Advances in Databases, Knowledge, and Data Applications*. iaria, 2012, pp. 140–146.

[3] A. Schmidt, D. Kimmig, and R. Hofmann, “A first step towards a query optimizer for column-stores,” in *DBKDA’12: Procceedings of the The Fourth International Conference on Advances in Databases, Knowledge, and Data Applications*. iaria, 2012.

[4] M. Stonebraker, D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madden, E. O’Neil, P. O’Neil, A. Rasin, N. Tran, and S. Zdonik, “C-store: a column-oriented dbms,” in *VLDB ’05: Proceedings of the 31st international conference on Very large data bases*. VLDB Endowment, 2005, pp. 553–564.

[5] D. J. Abadi, S. R. Madden, and N. Hachem, “Column-stores vs. row-stores: How different are they really,” in *In SIGMOD*, 2008.

[6] N. Bruno, “Teaching an old elephant new tricks,” in *CIDR 2009, Fourth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA*. www.crdrdb.org, 2009.

[7] D. J. Abadi, D. S. Myers, D. J. Dewitt, and S. R. Madden, “Materialization strategies in a column-oriented dbms,” in *In Proc. of ICDE*, 2007.

[8] D. J. Abadi, S. R. Madden, and M. Ferreira, “Integrating compression and execution in column-oriented database systems,” in *SIGMOD*, Chicago, IL, USA, 2006, pp. 671–682.

[9] T. M. Chilimbi, B. Davidson, and J. R. Larus, “Cache-conscious structure definition,” in *PLDI ’99: Proceedings of the ACM SIGPLAN 1999 conference on Programming language design and implementation*. New York, NY, USA: ACM, 1999, pp. 13–24.

[10] S. Manegold, P. A. Boncz, and M. L. Kersten, “Optimizing database architecture for the new bottleneck: memory access,” *The VLDB Journal*, vol. 9, no. 3, pp. 231–246, 2000.

[11] M. Nelson, *The Data Compression Book*. New York, NY, USA: Henry Holt and Co., Inc., 1991.

[12] K. Wu, E. J. Otoo, and A. Shoshani, “Optimizing bitmap indices with efficient compression,” *ACM Trans. Database Syst.*, vol. 31, no. 1, pp. 1–38, 2006.

[13] “TPC Benchmark H Standard Specification, Revision 2.1.0,” Transaction Processing Performance Council, Tech. Rep., 2002.

Aspects of Append-Based Database Storage Management on Flash Memories

Robert Gottstein
Databases and Distributed Systems Group
TU-Darmstadt Germany
gottstein@dvs.tu-darmstadt.de

Ilia Petrov
Data Management Lab
Reutlingen University, Germany
ilia.petrov@reutlingen-university.de

Alejandro Buchmann
Databases and Distributed Systems Group
TU-Darmstadt, Germany
buchmann@dvs.tu-darmstadt.de

***Abstract*—New storage technologies, such as Flash and Non-Volatile Memories, with fundamentally different properties are appearing. Leveraging their performance and endurance requires a redesign of existing architecture and algorithms in modern high performance databases. Multi-Version Concurrency Control (MVCC) approaches in database systems, maintain multiple timestamped versions of a tuple. Once a transaction reads a tuple the database system tracks and returns the respective version eliminating lock-requests. Hence under MVCC reads are never blocked, which leverages well the excellent read performance (high throughput, low latency) of new storage technologies. Upon tuple updates, however, established implementations of MVCC approaches (such as Snapshot Isolation) lead to multiple random writes – caused by (i) creation of the new and (ii) in-place invalidation of the old version – thus generating suboptimal access patterns for the new storage media. The combination of an append based storage manager operating with tuple granularity and snapshot isolation addresses asymmetry and in-place updates. In this paper, we highlight novel aspects of log-based storage, in multi-version database systems on new storage media. We claim that multi-versioning and append-based storage can be used to effectively address asymmetry and endurance. We identify multi-versioning as the approach to address data-placement in complex memory hierarchies. We focus on: *version handling*, (physical) *version placement*, *compression* and *collocation* of tuple versions on Flash storage and in complex memory hierarchies. We identify possible read- and cache-related optimizations.**

***Keywords*-Multi Version Concurrency Control; Snapshot Isolation; Version; Append Storage; Flash; Data Placement.**

I. INTRODUCTION

New storage technologies such as flash and non-volatile memories have fundamentally different characteristics compared to traditional storage such as magnetic discs. Performance and endurance of these new storage technologies highly depend on the I/O access patterns.

Multi-Version approaches maintaining versions of tuples, effectively leverage some of their properties such as fast reads and low latency. Yet, asymmetry and slow in-place updates need to be addressed on architectural and algorithmic levels of the DBMS. Snapshot Isolation (SI) has been implemented in many commercial and open-source systems: Oracle, IBM DB2, PostgreSQL, Microsoft SQL Server 2005, Berkeley DB, Ingres, etc. In some systems, SI is a separate isolation level, in others used to handle serializable isolation.

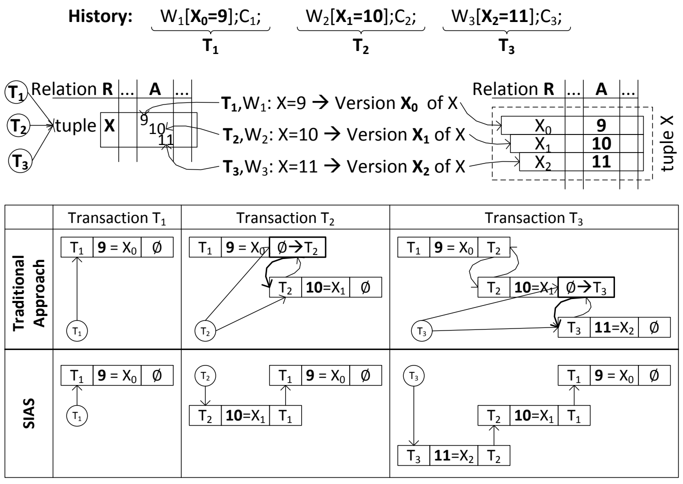

History: W1[X0=9];C1; W2[X1=10];C2; W3[X2=11];C3;
T1 T2 T3
Relation R ... A ...
tuple X
T1,W1: X=9 → Version X0 of X
T2,W2: X=10 → Version X1 of X
T3,W3: X=11 → Version X2 of X
tuple X
Transaction T1
Transaction T2
Transaction T3
Traditional Approach
SIAS
9 = X0
10=X1
11=X2
∅→T2
∅→T3

Figure 1. Invalidation in SI and SIAS

Under the concept of *Append-based storage* management any newly written data appended is at the logical head of a circular append log. This way, random writes are transformed into sequential writes and in-place update operations are reduced to a controlled append of the data, which is an effective mechanism to address the assymmetric performance of new storage technologies (see Section III)

In SIAS [1], we combine snapshot isolation and append storage management (with tuple granularity) on Flash. Under TPC-C workload SIAS achieves up to 4x performance improvement on Flash SSDs, a significant *write overhead reduction* (up to 38x), better *space utilization* due to denser version packing per page, better *I/O parallelism* and up to 4x lower disk I/O execution times, compared to traditional approaches. SIAS aids better *endurance*, due to the use of out-of-place writes as appends and write overhead reduction.

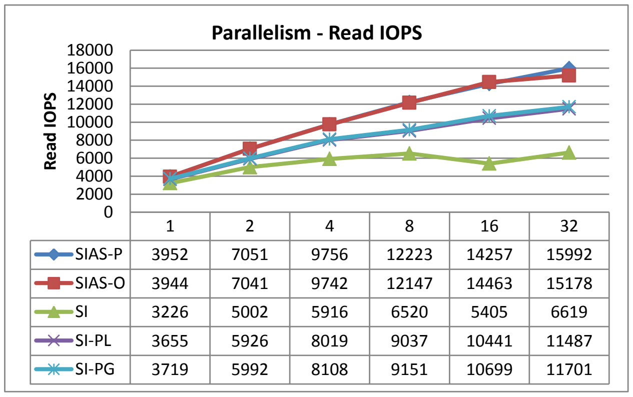
SIAS implicitly invalidates tuple versions by creating a successor version; thus, avoiding in-place updates. SIAS manages tuple versions of a single data item as simply linked lists (chains), addressed by a virtual tuple ID (VID). Figure 1 illustrates the invalidation process in SI and SIAS. Transactions $T1$, $T2$, $T3$ update data item X in serial order. Thereafter, the relation contains three different tuple versions of data item X. The initial version X_0 of X is created by $T1$ and updated by $T2$. The *traditional approach (SI)* invalidates X_0 in-place by physically setting the invalidation timestamp and creating X_1. Analogously, $T3$ updates X_1 with the physical in-place invalidation of X_1. *SIAS* connects tuple versions using the VID where the newest tuple version is always known. Each tuple maintains a backward reference to its predecessor, which does not need to be updated in place. Hence, updating X_0 leads to the creation of X_1.

We report our work in progress on data placement and summarize key findings and the preliminary results of SIAS (published in a previous work). In this paper, we focus on novel aspects of *version handling*, (physical) *placement* and *collocation* on append-based database storage manager using flash memory as primary storage.

In the next section we present the related work. Section III provides a brief summary of the properties of flash technology. Section IV introduces the SIAS approach, aspects of *version handling*, (physical) *placement* and *collocation*. Section V concludes the paper.

II. Related Work

SIAS organizes data item versions in simple chronologically ordered chains, which has been proposed by Chan et al. in [2] and explored by Petrov et al. in [3] and Bober et al. in [4] in combination with MVCC algorithms and special locking approaches. Petrov et al. [3], Bober et al. [4], Chan et al. [2] explore a log/append-based storage manager. The applicability of append-based database storage management approaches for novel asymmetric storage technologies has been partially addressed by Stoica et al. in [5] and Bernstein et al. in [6] using page-granularity, whereas SIAS employs tuple-granularity much like the approach proposed by Bober et al. in [4], which, however, invalidates tuples in-place. Given a page granularity the whole invalidated page is remapped and persisted at the head of the log, hence no write-overhead reduction. In tuple-granularity, multiple new tuple-versions can be packed on a new page and written together. Log storage approaches at file system level for hard disk drives have been proposed by Rosenblum in [7]. A performance comparison between different MVCC algorithms is presented by Carey et al. in [8]. Insights to the implementation details of SI in Oracle and PostgreSQL are offered by Majumdar in [9]. An alternative approach utilizing transaction-based tuple collocation has been proposed by Gottstein et al. in [10]. Similar chronological-chain version organization has been proposed in the context of update intensive analytics by Gottstein et al. in [11]. In such systems data-item versions are never deleted, instead they are propagated to other levels of the memory hierarchy such as HDDs or Flash SSDs and archived. Any logical modification operation is physically realized as an append. SIAS on the other hand provides mechanisms to couple version visibility to (logical and physical) space management. SIAS uses transactional time (all timestamps are based on a transactional counter) in contrast to timestamps that correlate to logical time (dimension). Stonebraker et al. realized the



	1	2	4	8	16	32
SIAS-P	3952	7051	9756	12223	14257	15992
SIAS-O	3944	7041	9742	12147	14463	15178
SI	3226	5002	5916	6520	5405	6619
SI-PL	3655	5926	8019	9037	10441	11487
SI-PG	3719	5992	8108	9151	10699	11701

Figure 2. I/O Parallelism on Intel X25-E SSD - 60 Minute TPC-C

concept of TimeTravel in PostgreSQL [12].

III. Flash Memories

We briefly sum up common properties of flash storage and compare them to traditional storage technologies (RAM, spinning disks): (i) *read/write asymmetry* – reads are much faster than writes, up to an order of magnitude; (ii) *low random write throughput* – small random writes are 5-10x slower than reads; (iii) *endurance issues and wear*; (iv) *suboptimal mixed load performance*: mixing reads/writes or random/sequential patterns leads to performance degradation.

Table I
SIAS and SI results on Intel X25-E SSD [1]

Queue Depth 1

Trace	read IOPS	write IOPS	read MB	write MB	time (sec)
SIAS-O(I)	4476	20	19713	89.96	563.675
SIAS-P (I)	4499	19	20666	89.96	587.873
SI (I)	3771	322	19901	1624	721.843
SIAS-O(II)	3947	13	11542	39.76	374.204
SIAS-P (II)	3953	13	11562	39.76	374.341
SI (II)	3656	432	11852	1395	414.869

Queue Depth 32

Trace	read I/O	write I/O	read MB	write MB	time (sec)
SIAS-O(I)	14500	66	19713	89.96	174.01
SIAS-P (I)	14642	63	20666	89.96	180.658
SI (I)	3360	264	19901	1624.9	805.193
SIAS-O(II)	15981	55	11542	39.76	92.44
SIAS-P (II)	15722	54	11562	39.76	94.128
SI (II)	11365	1338	11852	1395	133.478

IV. SIAS - Snapshot Isolation Append Storage

In this section we provide a short summary of the SIAS approach [1]. SIAS manages versions as simply linked lists (chains) that are addressed by using a virtual tuple ID (VID). On creation of a new version it implicitly invalidates the old one resulting in an out-of-place write – implemented as a logical append – and avoiding the in-place update of the predecessor. SIAS is coupled to an append-based storage manager, appending in units of tuple versions. Table I shows our test results with SIAS. Two traces containing all accessed and inserted tuples were recorded under PostgreSQL running TPC-C instrumented with different parameters. *Trace I* was instrumented using 5 warehouses with four hours runtime and *Trace II* using 200 warehouses and 90 minutes runtime. Both traces were fed into our database storage simulator which generated SIAS-O/P and SI traces, containing read and written DB-pages to be used as input for the FIO benchmark which executed them on an Intel X25-E SSD. SIAS-O is a simulation with and SIAS-P without caching of the SIAS data structures, where SI is the classic Snapshot Isolation using in-place updates on the invalidation. The conclusions of our results are: *(i) SI reads more than SIAS-O but less than SIAS-P; (ii) SI writes more gross-data than SIAS-O/P; (iii) SIAS-O/P reads with more IOPS than SI; (iv) SIAS needs less runtime than SI; and (v) SIAS-O/P scales better than SI with higher parallelism.* We also conducted tests using SI and page-wise append, performing a remapping of all pages which either appends pages local at each relation (SI-PL) or at a global append area (SI-PG) with the results displayed in Figure 2. We found that while each of both outperforms original SI by 15 to 76%, both themselfes are outperformed by SIAS-O/P by 6 to 36%. Our results empirically confirm our hypothesis that (a) appends are more suitable for Flash, (b) append granularity is crucial to performance and (c) appending in tuples and writing in pages is superior to remapping of pages. In the following sections we describe our approaches to merging of pages and physical tuple version placement as well as compression and indexing.

A. Merge

One key assumption of append based storage is that once data was appended it is never updated in-place. In a multi-version database old and updated versions inevitably become invisible which leads to different tuple versions of the same data item, most likely located at different physical pages. Hence pages *age* during runtime and contain visible and invisible tuple versions. In a production database running 24x7 it is realistic to assume that *net amount of visible tuples* on such pages is low and that an ample amount of outdated *dead* tuple versions is transferred, causing cache pollution. Once a certain threshold of dead tuples per page is reached it is beneficial to re-insert still visible tuples and mark the page as invalid. Dead tuples may be pruned or archived. Since a physical invalidation of the old page would lead to an in-place update, we suggest using a bitmap index providing a boolean value per page indicating its invalidation. The page address correlates to the position in the bitmap index, therefore the size is reasonably small. A merge therefore includes the re-insertion of still visible tuples into a new page and the update of the bitmap index. On the re-insertion the placement of the tuples may be reconsidered (Sect. IV-B).

Space reclamation of invalidated pages is also known as garbage collection in most MVCC approaches. On flash memories, a physical erase can only be executed in erase unit granularities, hence it makes sense to apply reclamation in such granules and to make use of the *Trim* command. Pruning a single DB-page with the size smaller than an erase unit will most likely cause the FTL to create a remapping within the it's logical/physical block address table and postpones the physical erasure. This may result in unpredictable latency outliers due to fragmentation and postponed erasures [3]. Using the bitmap index, indicating deleted/merged pages (prunable), a consecutive sequence of pruned pages within an erase unit can be selected as a victim altogether. If the sequence still contains pages which have not been merged yet, they can be merged before the reclamation.

SIAS uses data structures to guarantee the access to the most recent committed version X_v of a data item X. If only the most recent committed version has to be re-inserted (i.e. no successor version exists), nothing but the SIAS data structure has to be updated. It is theoretically possible that the tuple version is still visible and invalidated. In this case a valid successor version to that tuple exists which has to be re-inserted as well: Let P_m be the victim page, X_i an invalidated tuple version of data item X, where $X_i \in P_m$ and $X_v \in P_k$, $P_m \neq P_k$. X_v is the direct successor to X_i physically pointing to X_i. The merge of P_m leads to a re-insertion of X_i as X_i* which leads to a re-insertion of X_v as X_v*, pointing to X_i*. The SIAS data structures are updated such that the most recent committed version of X know is X_i*. It is not necessary to merge P_k as well, since X_v simply becomes an orphan tuple version which is not reachable by the SIAS data structures. Phantoms cannot occur since X_v* and X_v yield the same VID and version count. Nevertheless, it is most likely that X_i will become invisible during the merge since OLTP transactions are usually short and fast running.

B. Tuple Version Placement

In SIAS, each relation maintains a private append region and tuples are appended in the order they arrive at the append storage manager. Tuples of different relations are not stored into the same page and pages of different relations are not stored into the same relation regions. Appending tuple versions in the order they arrive may be suboptimal, since merged, updated and inserted tuples usually have different access frequencies. Collocation of tuples according to their access frequency can be benefitial since the net amount of actually used tuples per transferred page is higher [10]. Using temperature as a metric, often accessed tuples are hot and seldom accessed tuples are cold. The goal of tuple placement is to transfer as much hot tuples as possible with one I/O to reduce latency and to group cold tuples such that archiving and merging is efficiently backed. Visibility meta-information also contributes to access frequency, since tuples need to be checked for visibility. This creates yet another dimension upon which tuples can be related apart from the attribute values. Even if the content is not related the visibility of the tuples may be comparable.

Under the working set assumption and according to the 80/20 rule - both are the key drivers of data placement - (80% of all accesses refers to 20% of the data – as in OLTP enterprise workloads [13]) statistics can be used during an update to inherit access frequencies to the new tuple version.

In SIAS, the length of the chain describes the amount of updates to a data item (amount of tuple versions). Hence, a long chain is correlated to a frequently updated data item. A page containing frequently updated tuple versions will likely contain mostly invisible tuples after some runtime, hence simplifying the merge/reclamation process.

Version Meta Data Placement: Version metadata embodying a tuple's visibility/validity is stored on the tuple itself in existing MVCC implementations. An update creates a new version and version information of the predecessor has to be updated accordingly. SIAS benefits largely from the avoidance of the in-place invalidation. Further decoupling visibility information and raw data would be even more benefitial. Raw data becomes stale and redundancies caused by, e.g., tuples that share the same content but different visibility information are reduced or vanish completely. A structure that separately maintains all visibility information, enables accessing only needed data (payload) on Flash memory. This principle inherently deduplicates tuple data and creates a dictionary of tuple values. Visibility meta-information can be stored in a column-store oriented method, where visibility information and raw tuple data form a n:1 relation. This facilitates usage of compression and compactation techniques. A page containing solely visibility meta-information can be used to pre-filter visible tuple versions which subsequently can be fetched in parallel utilizing the inherent SSD parallelism, asynchronous I/O and prefetching.

C. Optimizations

A number of optimization techniques can be derived from observation that in append based storage a page is never updated, yet: compression, optimization for cache and scan efficiency, page layout transformation etc. Generally these facilitate analytical operations (large scans and selections) on OLTP systems supporting archival of older versions.

Compression. Most DBMS store tuples of a relation exclusively on pages allocated for that very relation. In a multi version environment, versions of tuples of that relation are stored on a page. Since all these have the same schema (record format) and differ on few attribute values at most, the traditional light-weight compression techniques (e.g., dictionary- and run-length encoding) can be applied.

Page-Layout and Read Optimizations. Since the content of a written page is immutable and only read operations can access the page, a number of optimizations can be considered. If large scans (e.g. log analysis) are frequent, cache efficiency becomes an issue hence the respective page-layouts can be selected. Furthermore it is possible to use analytical-style page layout (e.g., PAX) for the version data and traditional slotted pages for the temporary or update intensive data such as indices.

V. Conclusion and Future Work

We propose the combination of multi-version databases and append-based storage as most beneficial to exploit new storage media. We have prototypically implemented SIAS in PostgreSQL and validated the reported simulation results. The highest performance benefit can be achieved by the integration of the append storage principle directly into a multi-version DBMS, reducing the update granularity to a tuple-version, implementing all writes out-of-place as appends, and coupling space management to version visibility. In contrast page remapping append storage manager does not fully benefit of the new storage technology. SIAS is a Flash-friendly approach to multi-version DBMS: (i) it sequentialises the typical DBMS write patterns, and (ii) reduces the net amount of pages written. The former has direct performance implications the latter has long-term longevity implications. In addition SIAS introduces new aspects to data placement making it an important research area. We especially identify version archiving, selection of hot/cold tuple versions, separation of version data and version meta-data, compression and indexing as relevant research areas.

In our next steps we focus on optimizations such as compression of tuple versions to further reduce write overhead by 'compacting' appended pages, placement of correlated tuple versions to increase cache efficiency as a 'per page clustering' approach and an efficient indexing of multi-version data using visibility meta-data separation.

Acknowledgment

This work was supported by the DFG (Deutsche Forschungsgemeinschaft) project "Flashy-DB".

References

[1] R. Gottstein, I. Petrov and A. Buchmann, "SIAS: Chaining Snapshot Isolation and Append Storage," submitted.

[2] A. Chan, S. Fox, W.-T. K. Lin, A. Nori, and D. R. Ries, "The implementation of an integrated concurrency control and recovery scheme," in *19 ACM SIGMOD Conf. on the Management of Data, Orlando FL*, Jun. 1982.

[3] I. Petrov, R. Gottstein, T. Ivanov, D. Bausch, and A. P. Buchmann, "Page size selection for OLTP databases on SSD storage," *JIDM*, vol. 2, no. 1, pp. 11–18, 2011.

[4] P. Bober and M. Carey, "On mixing queries and transactions via multiversion locking," in *Proc. IEEE CS Intl. Conf. No. 8 on Data Engineering, Tempe, AZ*, feb 1992.

[5] R. Stoica, M. Athanassoulis, R. Johnson, and A. Ailamaki, "Evaluating and repairing write performance on flash devices," in *Proc. DaMoN 2009*, P. A. Boncz and K. A. Ross, Eds., 2009, pp. 9–14.

[6] P. A. Bernstein, C. W. Reid, and S. Das, "Hyder - A transactional record manager for shared flash," in *CIDR*, 2011, pp. 9–20.

[7] M. Rosenblum, "The design and implementation of a log-structured file system," U.C., Berkeley, Report UCB/CSD 92/696, Ph.D thesis, Jun. 1992.

[8] M. J. Carey and W. A. Muhanna, "The performance of multiversion concurrency control algorithms," *ACM Trans. on Computer Sys.*, vol. 4, no. 4, p. 338, Nov. 1986.

[9] D. Majumdar, "A quick survey of multiversion concurrency algorithms."

[10] R. Gottstein, I. Petrov, and A. Buchmann, "SI-CV: Snapshot isolation with co-located versions," in *in Proc. TPC-TC*, ser. LNCS. Springer Verlag, 2012, vol. 7144, pp. 123–136.

[11] J. Krueger, C. Kim, M. Grund, N. Satish, D. Schwalb, J. C. H. Plattner, P. Dubey, and A. Zeier, "Fast updates on read-optimized databases using multi-core CPUs," in *Proceedings of the VLDB Endowment*, vol. 5, no. 1, sep 2011.

[12] M. Stonebraker, L. A. Rowe, and M. Hirohama, "The implementation of postgres," *IEEE Trans. on Knowledge and Data Eng.*, vol. 2, no. 1, p. 125, Mar. 1990.

[13] S. T. Leutenegger and D. Dias, "A modeling study of the TPC-C benchmark," in *Proc. ACM SIGMOD Conf.*, Washington, DC, May 1993, p. 22.

Mapping XML to Key-Value Database

Pavel Strnad
Czech Technical University, FEE
Prague, Czech Republic
Email: pavel.strnad@fel.cvut.cz

Ondrej Macek
Czech Technical University, FEE
Prague, Czech Republic
Email: ondrej.macek@fel.cvut.cz

Pavel Jira
Czech Technical University, FIT
Prague, Czech Republic
Email: jirap1@fit.cvut.cz

***Abstract*—XML is a popular data format used in many software applications. Native XML databases can be used for persistence of XML documents or the document can be stored in a relational database.**

In this paper we propose an alternative storage for XML documents based on key-value databases. We propose three general algorithms for XML to key-value database mapping. The algorithms are optimized using specific features of key-value database Redis. Finally a performance comparison of implemented algorithms and common native XML databases is provided.

***Keywords*-XML; key-value database; Redis; RedXML; XML mapping**

I. Introduction

XML format is a popular data format used in many software applications, where the XML documents can be stored as files or in native XML databases, but often they are stored in Relational Database Management Systems (RDBMS).

In this paper we propose an alternative to mentioned storages of XML documents based on key-value databases, which are popular in last days. The next reason why to consider key-value databases as possible storage of XML documents is the speed of queries in key-value databases, which can improve the speed of XML querying. Therefore we have proposed algorithms for XML to key-value database mapping and implemented them using key-value database Redis [1]. Next a performance testing was performed on two basic scenarios of loading and saving an XML document. Results of these benchmarks and their comparison to already existing XML storages show if the key-value databases are a next alternative for storing XML data.

The paper is organized as follows: first related work in the area is discussed in Section II, then the algorithms for XML to key-value database mapping are introduced in Section III and their implementation and optimization is described in Section IV. Next, there is the performance comparison in Section V and finally the future work is discussed in SectionVI.

II. Related work

XML to Key-Value Database mapping: Vaidya and Gopal [2] presented an algorithm for mapping an XML document to a two dimensional array of hashes. In contrast with this work we introduce more sophisticated mapping algorithms and we provide a performance comparison with native XML databases. At the moment we are not aware of any other extensive research in this field, therefore we focus on mapping XML to RDBMS.

XML enabled databases: The concept of XML-enabled databases allows to manipulate data stored in relational database as an XML document. The approach is based on a special column type or on a mapping of an XML document to a relational schema [3] [4].

XML to RDBMS mapping: The problem of mapping XML document to RDBMS is well known issue. Many algorithms of mapping an XML structure to a relational database schema were proposed. Some of them are based on a structure of XML document [5] [6], others use an additional information about document querying to create more effective mapping [7] [8]. The issue of retrieving XML documents from RDBMS is addressed in [9] where several mapping algorithms are introduced and theirs efficiency and scalability is evaluated.

The mapping algorithms proposed in this paper allows key-value databases to be used as XML-enabled ones. All proposed algorithms are based only on structure of an XML document. The optimization of proposed algorithms is based on features specific for Redis database.

III. Algorithms

In this section we describe three basic principles of mapping XML documents (elements, attributes, etc.) into a key-value database. As an example key-value database platform we have chosen Redis. Each of proposed solutions differs in the way of mapping and some of them are dependent on specific features of Redis database. All mappings mentioned in this section are demonstrated on document books2.xml shown in Figure 1.

A. Redis Database Structure

Before we propose transformations for XML to key-value database mapping we introduce the structure of Redis database as the target of all mapping algorithms. The Redis database consists of environments containing collections, which can be nested, so it is possible to create trees of collections.

The environment is the main logical unit in the database which contains collections, documents and sequences for generating identifiers. Each environment defines a context for various database settings. The environment information are stored in keys:

- **info** – contains only the field <*iterator*> representing the value of sequence for generating identifier for a new environment.
- **environment** – contains mapping from environment name to identifier.
- **IDenvironment<info** – is similar to the **info** key as it contains the <*iterator*> for collections and documents stored in the environment.
- **IDenvironment<collections** – for collections in the concrete environment it contains the mapping from the collection name to its identifier.

To organize the content of the database the collections can be used. Each collection can contain a document or other collection. The collections are represented by following keys:

- **IDenvironment:IDcollection<info** – contains information about the collection - its name (*name*) and a parent identifier (*parent_id*). The parent identifier is used for effective navigation inside the collection trees and the environment.
- **IDenvironment:IDcollection:documents** – contains mapping from documents' names to theirs identifiers.
- **IDenvironment:IDcollection:collections** – contains mapping from sub-collections' names to theirs identifiers.

B. Naive Mapping

The first designed mapping of an XML document to key-value database is Naive mapping. This mapping is a straightforward solution and we introduce it as a reference solution which we compare to other proposed solutions.

The basic idea of Naive mapping is to store most of the information inside a key name which refers to a specific part of an XML document. Hence, the knowledge of the key provides important information without a need of querying a database. This finding is important for optimal implementation of querying languages such as XQuery [10]. The example of this key is following:

various::ebooks::books2.xml::
catalog::book>1::genre>2

For better understanding of the key above an equivalent XPath query is:

doc("books2.xml")/catalog/book[1]/genre[2]

Informally we can describe this key as: This key points to an element "genre" that is the second descendant node of

```
<?xml version="1.0" standalone="yes"
      encoding="UTF-8" ?>
<catalog>
   <book id="bk101" ISBN="123456"
         count="10">
      Author:
      <author>Gambardella, Matt</author>
      <title>XML Guide</title>
      <genre>Computer</genre>
      <genre>Technical</genre>
      <genre>Various</genre>
      <price>44.95</price>
      <publish_date>2000-10-01
      </publish_date>
      This book is nice.
   </book>
   <book id="bk102" ISBN="123457"
         count="9">
      Author:
      <author>Ralls, Kim</author>
      Title:
      <title>Midnight Rain</title>
      <genre>Fantasy</genre>
      <genre>Various</genre>
      <price>5.95</price>
      <publish_date>2000-12-16
      </publish_date>
   </book>
</catalog>
```

Figure 1. Example XML docuement books2.xml

an element "book" that is the first descendant node of a root element "catalog" and the root element "catalog" is located in "books2.xml" in "ebooks" collection and in a database named "various".

As we can see from the example above every key stores a lot of information inside its name. This can be very useful in XPath access optimization. The separator of two colons "::" must not be appear in th names of elements, collections or files.

The previous example shows how keys used in the mapping looks like. The following paragraph describes utilized structures by Naive Mapping that can be referenced by a key.

Element mapping: The key pointing to an element contains a list of children keys. This is useful for recursive descent along children elements. Information about parent is stored directly in the child's key. The prefix various::ebooks::books2.xml:: is omitted for better readability.

Key:

various::ebooks::books2.xml::catalog::book>1

A corresponding XPath Query:

doc("books2.xml")/catalog/book[1]

Content:

{catalog::book>1::name>1,
catalog::book>1::genre>1}

Attribute mapping: The key refers to a hash containing attributes of an element. For Attribute mapping we do not provide a corresponding XPath Query because XPath queries can not generate only attributes into the result.

Key:

various::ebooks::books2.xml::catalog::book>1<attributes

Content:

{'ISBN'=>'123456', 'count'=>'10'}

Text mapping: The key refers to a string representing mapped text. The same way of mapping is used for comments or CDATA sections. We only change the keyword "text" to "comment" or "cdata" respectively.

Key:

various::ebooks::books2.xml::catalog::book>1>text>2

Content:

"This book is nice."

XML declaration and file information mapping: The content of this key provides basic information about the document. This information contains the root element name, XML declaration and so on. We define a new delimiter "<" that is used to recognize special keys known as properties. In the next example we present a special key "info".

Key:

various::ebooks::books2.xml<info

Content:

{"root" => "catalog", "version" => "1.0", "encoding" => "UTF-8", "standalone" => "yes"}

The important advantage of Naive mapping is an easy implementation and high information density of keys' names. Main disadvantages of this approach are:

- The length of key names is enormous for deep nested elements.
- Rename operation of a document, element or collection is very inefficient, because every key containing old name has to be renamed.
- Delete operation is also inefficient.

The following approaches were developed to remove (or minimize) these disadvantages of Naive mapping.

C. Abbreviated Key Mapping

Abbreviated Key Mapping (AKM) is a mapping approach that eliminates the enormous length of keys of deep nodes and simplifies rename and delete operations. The basic idea of AKM is based on mapping of all elements, collections and files to unique identifier ID. In AKM the rename operation executes in a constant time ($O(1)$). We only change the name which is mapped to an ID. ID remains unchanged. IDs are used instead of names in keys.

AKM needs a new hash map which maps names to IDs. The last change is abbreviation of long identificators: *'attributes' => 'a', 'text' => 't', 'cdata' => 'd', 'comment' => 'c'*.

We have to define a new structure in the AKM's implementation. This structure is a hash map which maps original names to unique IDs. The example of the key is:

1:2:3:1:2>2:3>3

And an unbreviated version using Naive Mapping is:

various::ebooks::books2.xml::
catalog::book>2::genre>3

As you can see all names are replaced by identifiers. This key is much shorter than unabbreviated one. The AKM helper structure which maps names to IDs is stored for each document in the database independently. The name of this key is "emapping". The example of the mapping is:

Key:

1:2:3<emapping

Content:

{"catalog" => "1", "book" => "2", "genre" => "3"
"<iterator>" => "3"}.

Field " $< iterator >$ " is used for counting element IDs. If a new element is added " $< iterator >$ " is incremented and its value is used as ID of the new element. Hence, each document has its own iterator. This design decision was made with regards to the length of keys. Shorter keys are better because they are better aligned in a memory. AKM compared to naive mapping is much better performing in rename operation (in a constant time $O(1)$). On the other hand, performance of delete operation is still poor. Hence, we tried to find a better solution.

D. Centralized Hash Mapping

Centralized Hash Mapping (CHM) approach is based on a hash structure. The basic idea is to store the whole document in one hash structure to allow easy removing of documents from a database.

Redis database allows storing of string values into a hash structure. This lead to a change of a representation of documents in the database. The key which identifies this hash structure is:

1:2:3<content

The hash fields represent keys the same way as in previous mappings. The difference is that CHM does not use a document prefix to uniquely identify an element. The example of this key is:

1:2>1

This key contains following array of children:

"1:2>1:3>1 | 1:2>1:3>2"

As we mentioned above Redis allows only string values. Hence, we have to store children elements in a string delimited by "|".

Attributes are stored in a similar way:
Key:

1:2>1<a

Content:

1"value1"2"value2"3"value3

As we can see attribute names are also replaced by IDs. In this case we used quote symbol (") as a delimiter. This delimiter is conflict–free because quote must not be included in an attribute's value according to XML specification [11]. Full EBNF grammar of the proposed mapping is presented on page 69 in Appendix E in Jíra's Master's Thesis [12].

CHM provides many advantages in contrast to previous approaches. Rename, delete and move operations are executed in a constant time $O(1)$.

IV. Implementation

The implementation of algorithms mentioned in previous section is covered in project named RedXML. RedXML is written in Ruby language and is available as an open-source project at github [13]. As a storage layer RedXML uses Redis database.

This section provides information about the performance optimization of the database structure.

A. Key Cutting

Key Cutting is a mapping approach that utilizes a memory optimization techniques [14] implemented in Redis. Key Cutting is built on a fact that storing hashes in Redis can be much more memory efficient than storing key/string pairs. This approach optimizes memory consumption of the database.

For example $value1932 => "value"$ consumes more memory than this hash representation: $value19 => \{"32" => "value"\}$.

Main advantages of Key Cutting approach are:

- a decreasing of memory consumption by using hashes instead of strings,
- fast load operation of the whole document.

This approach has also following disadvantages:

- if there exist many keys in a database, the delete operation is slow ($O(n)$),
- decreasing of memory consumption is hardly predictable, since it differs from document to document.

B. Algorithms Optimization

The mapping algorithms were introduced in Section III, as we try to improve performance of the implementation we decided to optimize these mappings. The main idea of the optimization is to reduce the number of database queries, therefore we introduce following new keys:

1) **IDenvironment:IDcollection:IDdocument<namespaces** – is a special key for document namespaces. Its motivation is easier manipulation of a whole document (where the namespaces are not important) and faster querying (the namespace can be found by its key when needed).
2) **IDenvironment:IDcollection:IDdocument<info** – represents the content of the document. The elements and theirs content can be accessed by:
    - **IDroot:IDelement>order** – contains information about a single element - its attributes, text content and ordered set of its children keys.
    - **IDroot:IDelement>order>c>order** – contains the comment as a text.
    - **IDroot:IDelement>order>d>order** – contains the content of CDATA section.

This mapping allows us to query a document effectively as an XPath query can be rewritten directly into keys. Nevertheless some XPath queries (such as "//books") still need to go through the whole document structure.

V. Experiments

This section introduces results of several performance experiments. Theirs aim was to show the capabilities of proposed mappings and their limits and to compare RedXML with native XML databases. We have chosen to compare RedXML with native XML databases, because they are very close to RedXML mapping techniques and granularity of stored information is similar. The RedXML was compared to the eXist [15], Berkeley DB XML [16] and BaseX [17]. These databases were chosen as they represent the state of the art in the field of XML databases. According to the measurement results we can choose the best way for future optimizations.

Performance tests were made on two scenarios – document loading and document saving. We used XML generator XMLGen [18] to generate documents used for the performance testing. XMLGen tool provides a switch *-f* to set a size factor of generated document. Documents sizes are shown in Table I.

Table I
THE SIZE OF GENERATED DOCUMENTS USED IN THE EXPERIMENTS. THE FACTOR COLUMN REPRESENTS VALUE OF THE XMLGEN'S SWITCH *-f*.

Document name	Size	Factor
0-0-5.xml	1.4MB	0.01
0-1.xml	14.5MB	0.1
0-1-5.xml	21.7MB	0.15
0-2.xml	29.3MB	0.2

The main aim of the tests was to measure performance capabilities and memory usage of Redis database according to proposed algorithms. Both, an optimized and non-optimized, versions of algorithms were used so the optimization contribution can be verified. The measurement was done in two steps. First we did evaluation of the proposed mapping techniques. Second we compared these mappings to common used native XML database systems.

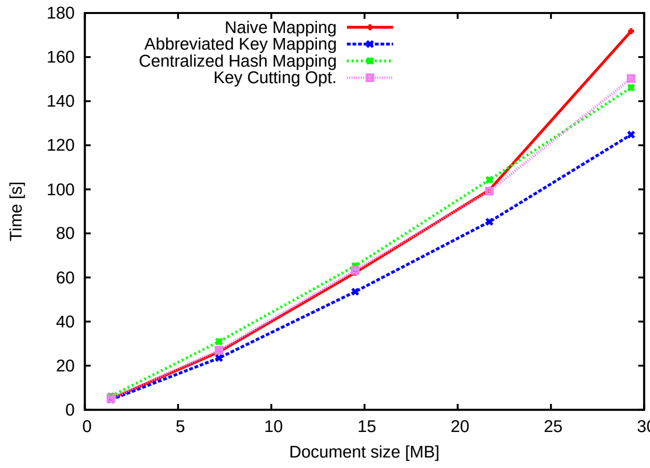


Figure 2. Saving documents of different sizes.

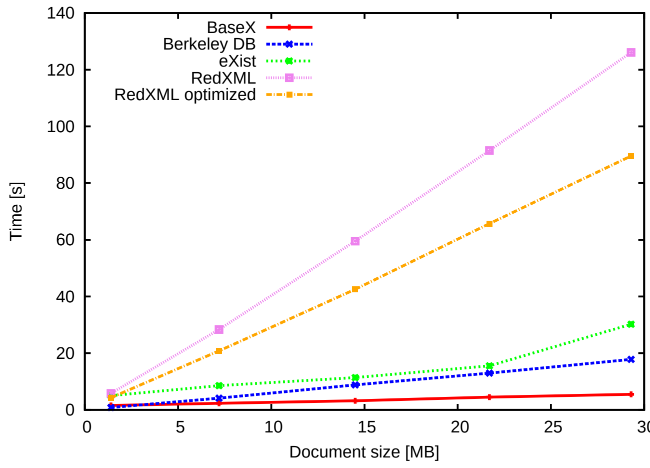


Figure 3. Saving documents of different sizes.

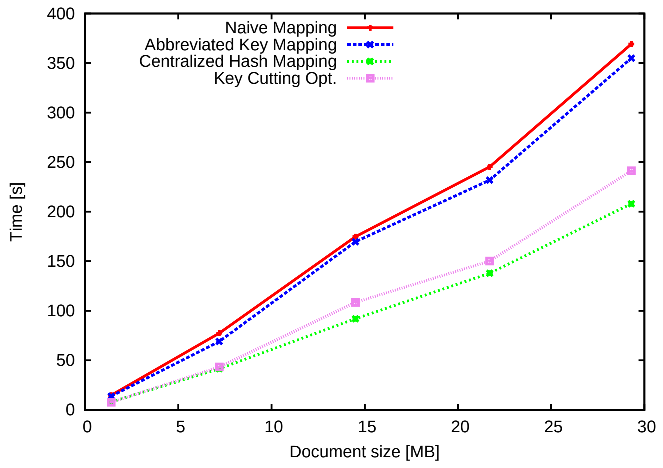


Figure 4. Loading documents of different sizes.

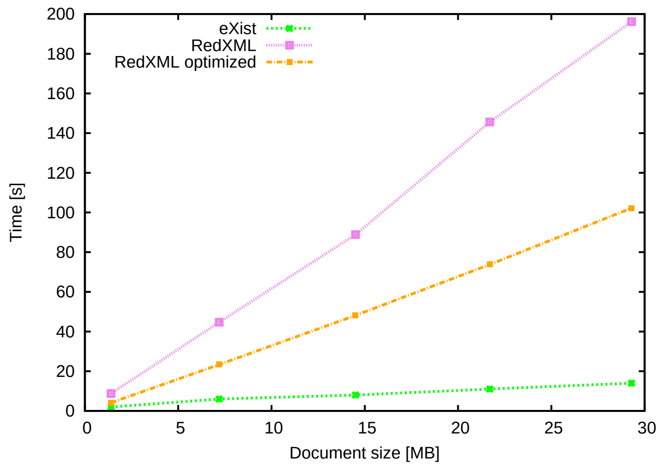


Figure 5. Loading documents of different sizes.

Hardware Configuration: In our experiments we used the following hardware and software configuration:

- Processor: Intel(R) Core(TM) 2 Duo T5500 1.66 GHz
- Memory: 1.5 GB
- Operating System: Debian 6.0.4, Kernel 2.6.32-5-amd64
- Redis version: 2.4.8

A. Databases Performance Comparison Results

The results of the performance tests for the document saving scenario of proposed approaches are shown in Figure 2. It is obvious that Abbreviated Key Mapping is most effective approach when saving large documents. All other mappings perform almost the same way.

Figure 3 shows the performance of RedXML compared to common native XML databases when saving documents. The RedXML performance in optimized version is 25% more effective than unoptimized version. On the other hand

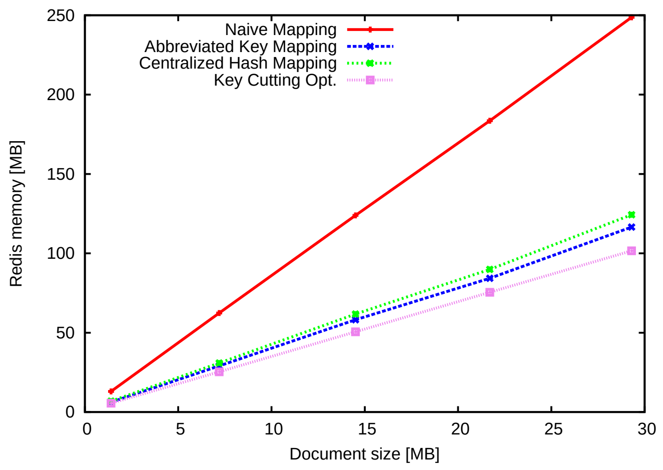


Figure 6. Memory consumption of Redis according to different mapping approaches.

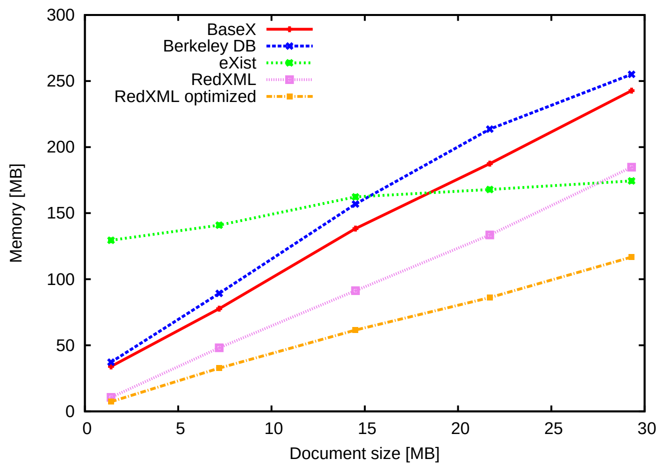


Figure 7. Memory consumption of different DB systems according to document size.

other databases are much more effective when saving a document. This is caused by an early stage of the RedXML project and by the fact that RedXML creates a lot of indices during saving, because of planned support of XQuery queries.

The results for the document loading scenario of proposed mappings are in Figure 4. Centralized Hash Mapping and Key Cutting Optimized algorithms are performing much faster than Naive Mapping and Abbreviated Key Mapping. It shows that the design of CHM and KC was successfuly implemented and it performs as intended.

The comparison with other databases is shown in Figure 5. BaseX and Berkeley DB XML databases were not measured because their document loading time was too short, it can be caused by lazy loading of documents or because mentioned databases saves a whole document so they are able to return it very fast when needed. The eXist database is again faster than RedXML or optimized RedXML, this can be partially caused by the performance difference between Java (eXist) and Ruby (RedXML). To speed up the document loading in the RedXML database the lazy loading or saving a copy of a whole document can be implemented.

The results of memory usage tests for the document saving are in Figure 6; the most effective algorithm is Key Cutting. It is 2.5x faster than Naive Mapping. Centralized Hash Mapping and Abbreviated Key Mapping have also good performance.

Figure 7 shows that database eXist is very effective in memory representation of XML documents in contrast with other databases for large documents. The optimized version of RedXML is most effective from memory usage perspective, even for large documents up to 30 MB it can be more effective than Bekeley DB XML, BaseX and eXist, but we would like to prove this hypothesis for larger documents in the near future. This optimization was focused on memory consumption and this benchmark verifies its implementation.

VI. Conclusion and Future Work

In this paper we have proposed four mappings of XML documents to a key-value database, next we have provided their implementation and optimization for key-value database Redis. Resulting XML-enabled database is called RedXML. RedXML can be easily deployed on multiple computers to achieve high scalability thanks to Redis platform. According to solutions provided in related-work we provide the solution that is dependent only on a few Redis commands which have specified time complexity. Hence, if we optimize these commands we can achieve better performance of RedXML.

The RedXML database was compared to several native XML databases from performance and memory consumption point of view. The results are promising especially according to memory consumption. In other benchmarks RedXML is not performing as well as other database systems.

The future work on RedXML will focus on the optimization of XML document querying as the main goal of the project is to provide effective XQuery implementation. Next research will focus on optimization of mapping an XML document into key-value database according to known queries over the mapped XML document.

References

[1] "Redis," Available: http://redis.io 21.9.2012.

[2] A. Vaidya and A. Gopal, "XML Representation using Hash Key for Performance Improvement," *International Journal of Computer Science and Application*, no. 2010, 2010.

[3] "Oracle XML-Enabled Technolgy," available: http://docs.oracle.com/cd/B10464_05/web.904/b12099/adx01int.htm 27.9.2012.

[4] “IBM - DB2 database software,” Available: http://www-01.ibm.com/software/data/db2/ 27.09.2012.

[5] A. Deutsch, M. Fernandez, and D. Suciu, “Storing semistructured data with STORED,” in *Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, ser. SIGMOD ’99. New York, NY, USA: ACM, 1999, pp. 431–442.

[6] D. Florescu and D. Kossmann, “A Performance Evaluation of Alternative Mapping Schemes for Storing XML Data in a Relational Database,” INRIA, Rapport de recherche RR-3680, 1999.

[7] M. Klettke and H. Meyer, “Xml and object-relational database systems - enhancing structural mappings based on statistics,” in *ACM SIGMOD Workshop on the Web and Databases*, 2000, pp. 63–68.

[8] P. Bohannon, J. Freire, P. Roy, and J. Simeon, “From XML Schema to Relations: A Cost-Based Approach to XML Storage,” *ICDE*, vol. 00, p. 64, 2002.

[9] A. Chebotko, M. Atay, S. Lu, and F. Fotouhi, “Xml subtree reconstruction from relational storage of xml documents,” *Data Knowl. Eng.*, vol. 62, no. 2, pp. 199–218, 2007.

[10] S. Boag, D. Chamberlin, M. F. Fernández, D. Florescu, J. Robie, and J. Siméon, “XQuery 1.0: An XML Query Language,” 2007.

[11] T. Bray, F. Yergeau, J. Cowan, J. Paoli, C. M. Sperberg-McQueen, and E. Maler, “Extensible Markup Language ({XML}) 1.1,” 2004.

[12] P. Jíra, “Transformation of XML into key-value database Redis,” Master’s thesis, Czech Technical University in Prague, Czech Republic, 2012. [Online]. Available: https://dip.felk.cvut.cz/browse/pdfcache/jirapave_2012dipl.pdf

[13] P. Jíra and P. Strnad, “RedXML Concept,” Available: https://github.com/jirapave/RedisXmlConcept 21.9.2012.

[14] “Redis Memory Optimization,” Available: http://redis.io/topics/memory-optimization 21.9.2012.

[15] “eXist Project Homepage,” Available: http://www.exist-db.org 21.9.2012.

[16] M. A. Olson, K. Bostic, and M. Seltzer, “Berkeley db,” in *Proceedings of the annual conference on USENIX Annual Technical Conference*, ser. ATEC ’99. Berkeley, CA, USA: USENIX Association, 1999, pp. 43–43.

[17] C. Grün, S. Gath, A. Holupirek, and M. H. Scholl, “Xquery full text implementation in basex,” in *Database and XML Technologies, 6th International XML Database Symposium, XSym 2009, Lyon, France, August 24, 2009. Proceedings*, ser. Lecture Notes in Computer Science, Z. Bellahsene, E. Hunt, M. Rys, and R. Unland, Eds., vol. 5679. Springer, 2009, pp. 114–128.

[18] A. R. Schmidt, F. Waas, M. L. Kersten, D. Florescu, I. Manolescu, M. J. Carey, and R. Busse, “The XML Benchmark Project,” CWI, Amsterdam, The Netherlands, Tech. Rep. INS-R0103, 2001.

Typing XPath Subexpressions With Respect to an XML Schema

Yasunori Ishihara[*], Kenji Hashimoto[†], Atsushi Ohno[*], Takuji Morimoto[*] and Toru Fujiwara[*]
[*] *Graduate School of Information Science and Technology, Osaka University, Suita, Japan*
Email: ishihara@ist.osaka-u.ac.jp, fujiwara@ist.osaka-u.ac.jp
[†] *Graduate School of Information Science, Nara Institute of Science and Technology, Ikoma, Japan*
Email: k-hasimt@is.naist.jp

***Abstract*—This paper discusses typing XPath subexpressions with respect to an XML schema, which is a new static analysis problem of XPath expressions. More formally, the typing problem is to decide whether there exists an XML document conforming to a given XML schema such that the nodes of the document matching to given subexpressions of a given XPath expression are of the given types. Deciding this problem is useful for query rewriting induced by schema evolution or integration. The contribution of this paper includes a decision algorithm for the typing problem, provided that XPath expressions include no path union operator. Moreover, it is shown that the typing problem is reducible to the XPath satisfiability problem in the presence of DTDs, for which many tractability results are known.**

***Keywords*-XPath; static analysis; type; XML schema**

I. Introduction

Static analysis of XPath expressions is one of the major theoretical topics in the field of XML databases. XPath is a query language for XML documents, where an XML document is often regarded as an unranked labeled ordered tree. An XPath expression specifies a pattern of (possibly branching) paths from the root of a given XML document. The answer to an XPath expression for an XML document t is a set of nodes v of t such that the specified path pattern matches the path from the root to v.

The most popular subtopic of static analysis may be *XPath satisfiability*, where a given XPath expression p is *satisfiable* under a given XML schema S if there is an XML document t conforming to S such that the answer to p for t is a nonempty set. Many tractable combinations of XPath classes and XML schema classes have been investigated so far [1], [2], [3]. Another popular one is *XPath containment* [4], [5], [6]. Moreover, *XPath validity*, which is a dual of XPath satisfiability, is investigated recently [7], [8].

This paper discusses a new problem of static analysis: *typing XPath subexpressions with respect to an XML schema*. We explain it by an example first.

Example 1: Consider the following fragment of an XML schema:

$$\begin{array}{ll} T_e \to \texttt{teachers}(M_t^*), & S \to \texttt{students}(M_s^*), \\ M_t \to \texttt{member}(N\,T_i), & M_s \to \texttt{member}(N\,G_e\,G_r), \\ N \to \texttt{name}(\cdots), & G_e \to \texttt{gender}(\cdots), \\ T_i \to \texttt{title}(\cdots), & G_r \to \texttt{grade}(\cdots). \end{array}$$

T_e, M_t, etc. are *types* defined in this XML schema, while `teachers`, `member`, etc. are *tag names* or *labels*. Note that the type of `member` is not unique in this schema.

Now, consider the following two XPath expressions:

$$p_1 = \downarrow^*\!::\texttt{member}/\downarrow::\texttt{name},\; p_2 = \downarrow^*\!::\texttt{member}/\downarrow::\texttt{gender}.$$

The types of the subexpression $\downarrow^*::$ `member` of p_1 are M_t and M_s, because in both cases `member` has `name` as its child. On the other hand, the type of $\downarrow^*::$ `member` of p_2 is only M_s, because `member` of type M_t has no `gender` as its child.

More formally, the typing problem is to decide whether for a given XML schema S, an XPath expression p, and a sequence $(\alpha_1, X_1), \ldots, (\alpha_k, X_k)$ of pairs of positions of p and types of S, there exists an XML document t conforming to S such that the node of t matching to the subexpression of p at α_i is of type X_i for each i $(1 \leq i \leq k)$.

The typing problem can be viewed as a natural extension of the XPath satisfiability problem, and is useful especially for query rewriting and optimization induced by schema evolution or integration. For example, consider again the fragment of the schema in Example 1. Suppose that $M_t \to$ `member`(NT_i) has evolved into $M_t \to$ `t_member`(NT_i). In order to keep the behavior of queries unchanged, p_1 must be rewritten to $(\downarrow^*::$ `member` $\cup \downarrow^*::$ `t_member`$)/\downarrow::$ `name` while p_2 is not necessarily rewritten. As another example, suppose that $M_t \to$ `member`(NT_i) has evolved into $M_t \to$ `member`(NT_iG_e). Then, the associated types of the subexpression $\downarrow^*::$ `member` of p_2 have been changed (i.e., the types are now both M_t and M_s). In this case, p_2 must be rewritten to an expression, say $\downarrow^*::$ `students`$/\downarrow::$ `member`$/\downarrow::$ `gender`, so that the subexpression $\downarrow^*::$ `member` is associated with only M_s.

Under the assumption that XPath expressions include no path union operator, this paper adopts two approaches to developing decision algorithms for the typing problem. The first one is a direct approach. In this approach, a given XML schema and a given XPath expression are translated into finite tree automata, and the associated types are analyzed by computing their intersection automaton. The second one is a reduction-based approach. In this approach, it is shown that the typing problem is reducible to the XPath satisfiability problem in the presence of DTDs, for which many tractability results are known. Moreover, a part of this result is extended so that all the possible combinations of types of subexpressions can be efficiently enumerated.

The rest of this paper is organized as follows. In Section II several preliminary definitions are provided. Sections III and IV present the direct and the reduction-based approaches, respectively. Section V summarizes the paper.

II. Definitions

A. Trees and XML documents

An XML document is represented by an unranked labeled ordered tree $t = (V_t, \lambda_t)$, where

- V_t is a prefix-closed, finite set of sequences of positive integers such that if $v \cdot i \in V_t$ and $i > 1$ then $v \cdot (i-1) \in V_t$; and
- λ_t is a mapping from V_t to a set Σ of labels.

Each element in V_t is called a *node* of t. The empty sequence $\epsilon \in V_t$ is called the *root* of t. The parent-child relation and sibling relation between nodes are defined in an ordinary way. We extend λ_t to a function on sequences, i.e., for a sequence $v_1 \cdots v_n$ of nodes, let $\lambda_t(v_1 \cdots v_n) = \lambda_t(v_1) \cdots \lambda_t(v_n)$. Attributes are not handled in this paper.

For any tree structure t (which is not necessarily representing an XML document) and its node v, let $t|_v$ denote the subtree of t rooted at v. For any trees $t_1, \ldots, t_n$ with the same node set V and $v \in V$, let $(t_1 \cdots t_n)|_v$ denote $t_1|_v \cdots t_n|_v$. Also, for any mapping f which returns a tree structure, let $f|_v$ denote a mapping such that $f|_v(x) = f(x)|_v$.

B. Regular expressions

A regular expression over an alphabet Σ consists of constants ϵ (empty sequence) and the symbols in Σ, and operators $\cdot$ (concatenation), $*$ (repetition), and $|$ (disjunction). We exclude $\emptyset$ (empty set) because we are interested in only nonempty regular languages. The concatenation operator is often omitted as usual. The string language represented by a regular expression e is denoted by $L(e)$. The size of a regular expression is the number of constants and operators appearing in the regular expression.

C. Finite tree automata and XML schemas

A *finite tree automaton* TA is a quadruple (N, Σ, B, P), where

- N is a finite set of states,
- Σ is a finite set of labels,
- $B \in N$ is the initial state, and
- P is a finite set of transition rules in the form of $X \to a(e)$ or $X \to Y$, where $X, Y \in N$, $a \in \Sigma$, and e is a regular expression over N called *content model*.

A finite tree automaton $TA = (N, \Sigma, B, P)$ is *local* if for any pair of rules $X \to a(e)$ and $X' \to a'(e')$ in P, $a \neq a'$ whenever $X \neq X'$.

An *interpretation* I_t^{TA} of a tree t for a finite tree automaton $TA = (N, \Sigma, B, P)$ is a mapping from V_t to the set of finite sequences over N satisfying the following conditions:

- $\mathit{first}(I_t^{TA}(\epsilon)) = B$ (note that ϵ is the root of t), where $\mathit{first}(x)$ denotes the first element of sequence x; and
- for any node v with n children ($n \geq 0$), there are transition rules $X_1 \to X_2$, $X_2 \to X_3, \ldots, X_k \to a(e)$ in P such that
  - $I_t^{TA}(v) = X_1 X_2 \cdots X_k$,
  - $\lambda_t(v) = a$, and
  - $\mathit{first}(I_t^{TA}(v \cdot 1)) \cdots \mathit{first}(I_t^{TA}(v \cdot n)) \in L(e)$ (note that $v \cdot i$ is the i-th child of v).

A tree t is *accepted* by a finite tree automaton TA if there is an interpretation of t for TA. Let $TL(TA)$ denote the set of trees accepted by TA.

An *XML schema* S is a finite tree automaton such that

- S has no rule in the form of $X \to Y$, and
- S has no pair of rules $X \to a_1(e_1)$ and $X \to a_2(e_2)$ where $a_1 \neq a_2$.

A *DTD* is an XML schema which is also local. In a DTD there is a one-to-one correspondence between N and Σ, so we often use a triple (N, B, P) to mean a DTD. A tree t *conforms* to an XML schema S if t is accepted by S. In this paper, we assume that every XML schema $S = (N, \Sigma, B, P)$ contains no useless states. That is, for each $X \in N$, there are a tree $t \in TL(S)$ and its interpretation I_t^S such that $I_t^S(v) = X$ for some node v of t. Each state in an XML schema is referred to as a *type*. The *size* $|S|$ of an XML schema S is the sum of the size of all content models.

D. XPath expressions

The syntax of an XPath expression p is defined as follows:

$$\begin{array}{lll} p & ::= & \chi :: l \mid p/p \mid p \cup p \mid p[p], \\ \chi & ::= & \downarrow \mid \uparrow \mid \downarrow^* \mid \uparrow^* \mid \rightarrow^+ \mid \leftarrow^+, \end{array}$$

where $l \in \Sigma$. Each $\chi \in \{\downarrow, \uparrow, \downarrow^*, \uparrow^*, \rightarrow^+, \leftarrow^+\}$ is called an *axis*. A subexpression in the form of $\chi :: l$ is said to be *atomic*. The *size* $|p|$ of an XPath expression p is defined as the number of atomic subexpressions in p.

A *position* of an XPath expression p is a finite sequence of positive integers representing a node of a parse tree of p. Precisely, the subexpression $p|_\alpha$ of p at position α is defined as follows:

- $p|_\epsilon = p$.
- If $p|_\alpha = p_1/p_2$, then $p|_{\alpha \cdot 1} = p_1$ and $p|_{\alpha \cdot 2} = p_2$.
- If $p|_\alpha = p_1 \cup p_2$, then $p|_{\alpha \cdot 1} = p_1$ and $p|_{\alpha \cdot 2} = p_2$.
- If $p|_\alpha = p_1[p_2]$, then $p|_{\alpha \cdot 1} = p_1$ and $p|_{\alpha \cdot 2} = p_2$.

Next, we define the satisfaction relation of an XPath expression p by a tree t with a witness mapping w, which is a partial mapping from the set of positions of p to the set of pairs of nodes of t. Intuitively, $w(\alpha)$ is a pair of nodes that satisfies $p|_\alpha$. Note that $w(\alpha)|_1$ and $w(\alpha)|_2$ are the first and the second components of $w(\alpha)$, respectively:

- $t \models (\downarrow :: l)(w(\alpha))$ if $w(\alpha)|_2$ is a child of $w(\alpha)|_1$ and $\lambda_t(w(\alpha)|_2) = l$.

- $t \models (\uparrow :: l)(w(\alpha))$ if $w(\alpha)|_2$ is the parent of $w(\alpha)|_1$ and $\lambda_t(w(\alpha)|_2) = l$.
- $t \models (\downarrow^* :: l)(w(\alpha))$ if $w(\alpha)|_2$ is $w(\alpha)|_1$ or a descendant of $w(\alpha)|_1$, and $\lambda_t(w(\alpha)|_2) = l$.
- $t \models (\uparrow^* :: l)(w(\alpha))$ if $w(\alpha)|_2$ is $w(\alpha)|_1$ or an ancestor of $w(\alpha)|_1$, and $\lambda_t(w(\alpha)|_2) = l$.
- $t \models (\rightarrow^+ :: l)(w(\alpha))$ if $w(\alpha)|_2$ is a following sibling of $w(\alpha)|_1$ and $\lambda_t(w(\alpha)|_2) = l$.
- $t \models (\leftarrow^+ :: l)(w(\alpha))$ if $w(\alpha)|_2$ is a preceding sibling of $w(\alpha)|_1$ and $\lambda_t(w(\alpha)|_2) = l$.
- $t \models (p_1/p_2)(w(\alpha))$ if $t \models p_1(w(\alpha \cdot 1))$, $t \models p_2(w(\alpha \cdot 2))$, $w(\alpha \cdot 1)|_2 = w(\alpha \cdot 2)|_1$, and $w(\alpha) = (w(\alpha \cdot 1)|_1, w(\alpha \cdot 2)|_2)$.
- $t \models (p_1 \cup p_2)(w(\alpha))$ if $t \models p_i(w(\alpha \cdot i))$ and $w(\alpha) = w(\alpha \cdot i)$ for some $i \in \{1,2\}$. Moreover, if $j \in \{1,2\}$ does not satisfy $t \models p_j(w(\alpha \cdot j))$, then for any position α' whose prefix is $\alpha \cdot j$, $w(\alpha')$ is undefined.
- $t \models (p_1[p_2])(w(\alpha))$ if $t \models p_1(w(\alpha \cdot 1))$, $t \models p_2(w(\alpha \cdot 2))$, $w(\alpha \cdot 1)|_2 = w(\alpha \cdot 2)|_1$, and $w(\alpha) = w(\alpha \cdot 1)$.

If $t \models p(w(\epsilon))$, we say that t *satisfies* p *with witness* w and write $(t, w) \models p$. Note that if $(t, w) \models p$ and p does not include path union operator $\cup$, then w is a total mapping.

E. Typing problem

Suppose that an XML schema S, an XPath expression p without path union operator $\cup$, and a sequence $(\alpha_1, X_1), \ldots, (\alpha_k, X_k)$ of pairs of positions of p and types of S are given. The *typing problem* is to decide whether there exist $t \in TL(S)$, an interpretation I_t^S of t for S, and a mapping w such that

- $w(\epsilon)|_1 = \epsilon$ (i.e., the root node of t),
- $(t, w) \models p$, and
- $I_t^S(w(\alpha_i)|_2) = X_i$ for each i $(1 \leq i \leq k)$.

III. A Direct Approach

This section provides an algorithm which directly decides the typing problem. First, the algorithm translates a given XPath expression p into a finite tree automaton TA_p, maintaining the information on the structure of p as the states of TA_p. Then, the algorithm analyzes the correspondence between subexpressions of p and the types of a given schema S, by taking the intersection of TA_p and S.

A. Translating XPath expressions into finite tree automata

For a given XPath expression p without path union $\cup$, we construct a finite tree automaton TA_p with two distinguished sets NC_p and ND_p of states. Roughly speaking, TA_p accepts an arbitrary tree t satisfying p. States in NC_p and ND_p are associated with the "start" and "goal" nodes of t matching p.

First, we provide the definitions of $TA_p = (N_p, \Sigma, B_p, P_p, NC_p, ND_p)$ for $p = \chi :: l$, where $B_p = B$, $NC_p = \{C\}$, and $ND_p = \{D\}$.

- If $p = \downarrow :: l$, then P_p consists of
  - $A \to \sigma(A^*)$ for each $\sigma \in \Sigma$,
  - $B \to \sigma(A^* B A^*)$ for each $\sigma \in \Sigma$,
  - $B \to C$,
  - $C \to \sigma(A^* D A^*)$ for each $\sigma \in \Sigma$, and
  - $D \to l(A^*)$.
- If $p = \downarrow^* :: l$, then P_p consists of
  - $A \to \sigma(A^*)$ for each $\sigma \in \Sigma$,
  - $B \to \sigma(A^* B A^*)$ for each $\sigma \in \Sigma$,
  - $B \to C$,
  - $B' \to \sigma(A^* B' A^*)$ for each $\sigma \in \Sigma$,
  - $B' \to D$,
  - $C \to \sigma(A^* B' A^*)$ for each $\sigma \in \Sigma$,
  - $C \to D$, and
  - $D \to l(A^*)$.
- If $p = \rightarrow^+ :: l$, then P_p consists of
  - $A \to \sigma(A^*)$ for each $\sigma \in \Sigma$,
  - $B \to \sigma(A^* B A^*)$ for each $\sigma \in \Sigma$,
  - $B \to B'$,
  - $B' \to \sigma(A^* C A^* D A^*)$ for each $\sigma \in \Sigma$,
  - $C \to \sigma(A^*)$ for each $\sigma \in \Sigma$, and
  - $D \to l(A^*)$.

For axes $\uparrow$, $\uparrow^*$, and $\leftarrow^+$, the tree automata are defined by swapping states C and D (and their associated labels) for the ones of $\downarrow$, $\downarrow^*$, and $\rightarrow^+$, respectively.

Consider $TA_p = (N_p, \Sigma, B_p, P_p, NC_p, ND_p)$ where $p = p_1/p_2$. TA_p is defined as the intersection [9] of TA_{p_1} and TA_{p_2}, except that the states in ND_{p_1} overlaps only the states in NC_{p_2}, and vice versa. More precisely,

$$N_p = (ND_{p_1} \times NC_{p_2}) \cup ((N_{p_1} - ND_{p_1}) \times (N_{p_2} - NC_{p_2})).$$

Moreover, $B_p = (B_{p_1}, B_{p_2})$, $NC_p = NC_{p_1} \times N_{p_2}$, and $ND_p = N_{p_1} \times ND_{p_2}$. TA_p for $p = p_1[p_2]$ can be constructed in a similar way.

Lemma 1: TA_p satisfies Properties 1 and 2 below. Note that each state of TA_p has the same tree structure as p:

Property 1: For each mapping w such that $(t, w) \models p$, there is an interpretation $I_t^{TA_p}$ of t for TA_p such that for each position α of p, $I_t^{TA_p}(w(\alpha)|_1)|_\alpha$ contains a state in $NC_{p|_\alpha}$ and $I_t^{TA_p}(w(\alpha)|_2)|_\alpha$ contains a state in $ND_{p|_\alpha}$.

Property 2: Conversely, for each interpretation $I_t^{TA_p}$ of t for TA_p, there is a mapping w such that $(t, w) \models p$ and for each position α of p, $I_t^{TA_p}(w(\alpha)|_1)|_\alpha$ contains a state in $NC_{p|_\alpha}$ and $I_t^{TA_p}(w(\alpha)|_2)|_\alpha$ contains a state in $ND_{p|_\alpha}$.

Proof: The lemma is shown by the induction on the structure of p. The case where $p = \chi :: l$ is easy: If $(t, w) \models p$, then there is an interpretation $I_t^{TA_p}$ of t for TA_p such that $I_t^{TA_p}(w(\epsilon)|_1) = C$ and $I_t^{TA_p}(w(\epsilon)|_2) = D$. Conversely, if $I_t^{TA_p}$ is an interpretation of t for TA_p, then there are unique nodes v and v' of t such that $I_t^{TA_p}(v) = C$ and $I_t^{TA_p}(v') = D$. By letting $w(\epsilon) = (v, v')$, we have $(t, w) \models p$, $I_t^{TA_p}(w(\epsilon)|_1) = C$, and $I_t^{TA_p}(w(\epsilon)|_2) = D$.

Consider the case where $p = p_1/p_2$. Suppose that TA_{p_1} and TA_{p_2} satisfy the two properties and $(t, w) \models p_1/p_2$. Then, by the definition of $\models$, we have $t \models p_1(w(1))$, $t \models p_2(w(2))$, $w(1)|_2 = w(2)|_1$, and $w(\epsilon) = (w(1)|_1, w(2)|_2)$. Let $I_t^{TA_{p_1}}$ and $I_t^{TA_{p_2}}$ be interpretations of t for TA_{p_1} and TA_{p_2}, respectively, that satisfy Property 1. Define $I(v) = (I_t^{TA_{p_1}}(v), I_t^{TA_{p_2}}(v))$ for any node v of t. Then, by the definition of TA_p, I is an interpretation of t for TA_p and satisfies Property 1. Conversely, suppose that $I_t^{TA_p}$ is an interpretation of t for TA_p. Then, by the definition of TA_p, $I_t^{TA_p}|_1$ and $I_t^{TA_p}|_2$ are interpretations of t for TA_{p_1} and TA_{p_2}, respectively. Let w_1 and w_2 be the mappings determined by $I_t^{TA_p}|_1$ and $I_t^{TA_p}|_2$, respectively, which satisfy Property 2. Define $w(1 \cdot \alpha) = w_1(\alpha)$, $w(2 \cdot \alpha) = w_2(\alpha)$, and $w(\epsilon) = (w_1(\epsilon)|_1, w_2(\epsilon)|_2)$. Then $(t, w) \models p_1/p_2$ and Property 2 is satisfied.

The case where $p = p_1[p_2]$ can be shown similarly. ■

B. Analyzing correspondence between XPath expressions and schemas

Let $TA_{S \cap p}$ be an intersection automaton of an XML schema S and TA_p, except that the states in NC_p overlaps only the initial state of S, and vice versa. $TA_{S\cap p}$ accepts t if and only if t satisfies p at its root node and t conforms to S. Moreover, for each interpretation $I_t^{TA_{S\cap p}}$ of t for $TA_{S\cap p}$, $I_t^{TA_{S\cap p}}|_1$ is an interpretation of t for S and $I_t^{TA_{S\cap p}}|_2$ is an interpretation of t for TA_p. Since TA_p satisfies Property 2, it holds in turn that there is a mapping w such that $(t, w) \models p$ and for each position α of p, $I_t^{TA_{S\cap p}}|_2(w(\alpha)|_2)|_\alpha$ contains a state in $ND_{p|_\alpha}$. Conversely, by Property 1, if $(t, w) \models p$, then there is an interpretation $I_t^{TA_p}$ of t for TA_p such that for each position α of p, $I_t^{TA_p}(w(\alpha)|_2)|_\alpha$ contains a state in $ND_{p|_\alpha}$. It holds in turn that for any interpretation I_t^S of t for S , mapping I defined as $I(v) = (I_t^S(v), I_t^{TA_p}(v))$ is an interpretation of t for $TA_{S\cap p}$.

This observation naturally induces the following algorithm for deciding the typing problem. Suppose that an XML schema S, an XPath expression p, and a sequence $(\alpha_1, X_1), \ldots, (\alpha_k, X_k)$ of pairs of positions of p and types of S are given. For each i $(1 \le i \le k)$, eliminate all the states (X, Y) (and associated rules) of $TA_{S\cap p}$ such that $X \ne X_i$ and $Y|_{\alpha_i} \in ND_{p|_{\alpha_i}}$. Then decide the emptiness of the tree language accepted by the resultant finite tree automaton. The answer of the typing problem is "yes" if and only if the tree language is not empty.

C. Discussion

It is easy to see that in the worst case, this algorithm runs at least in exponential time in the size of p because of the intersection operation on finite tree automata. However, this algorithm is expected to run reasonably fast in many cases. Indeed, finite tree automata can be translated into formulas in a variant of μ-calculus [10], and then the typing problem can be solved by fast decision procedures for μ-calculus formulas. An experimental analysis is left as future work.

IV. A Reduction-Based Approach

In this section, it is shown that the typing problem is reducible to XPath satisfiability in the presence of DTDs. Then, using this result, we provide a condition where all the possible combinations of types of atomic subexpressions can be efficiently enumerated.

Let p be an XPath expression without path union $\cup$. Then, for each subexpression $p|_\alpha$ of p, there is an atomic subexpression $p|_{\alpha'}$ of p such that for any t and w such that $(t, w) \models p$, we have $w(\alpha)|_2 = w(\alpha')|_2$. Therefore, in the rest of this section, we assume without loss of generality that all the given subexpressions of the typing problem are atomic.

A. Reduction to XPath satisfiability in the presence of DTDs

Let $S = (N, \Sigma, B, P)$ be an XML schema. Define a mapping φ as follows:

$$\begin{aligned}\varphi(S) &= (N, N \times \Sigma, B, \varphi(P)),\\ \varphi(P) &= \{X \to (X, a)(e) \mid X \to a(e) \in P\}.\end{aligned}$$

It is easy to show that $\varphi(S)$ is local.

For any $t \in TL(S)$ with an interpretation I_t^S, let $\varphi(t, I_t^S)$ be a tree such that $V_{\varphi(t,I_t^S)} = V_t$ and $\lambda_{\varphi(t,I_t^S)}(v) = (I_t^S(v), \lambda_t(v))$. It is easy to see that $\varphi(t, I_t^S) \in TL(\varphi(S))$. On the other hand, for any $t' \in TL(\varphi(S))$, let $\varphi^{-1}(t')$ denote the tree such that $V_{\varphi^{-1}(t')} = V_{t'}$ and $\lambda_{\varphi^{-1}(t')} = \lambda_{t'}|_2$. It is also easy to see that $\varphi^{-1}(t') \in TL(S)$ with interpretation $\lambda_{t'}|_1$.

Lemma 2: Let p be an XPath expression without path union $\cup$. Also, let p' be an XPath expression obtained by replacing each atomic subexpression $\chi_\alpha :: l_\alpha$ of p at α with $\chi_\alpha :: (X_\alpha, l_\alpha)$ for some $X_\alpha \in N$.

- Suppose that $t \in TL(S)$ with interpretation I_t^S and $(t, w) \models p$. Also suppose that for each position α of p such that $p|_\alpha$ is atomic, $I_t^S(w(\alpha)|_2) = X_\alpha$. Then, $(\varphi(t, I_t^S), w) \models p'$.
- Conversely, suppose that $t' \in TL(\varphi(S))$ and $(t', w') \models p'$. Then, $(\varphi^{-1}(t'), w') \models p$, and interpretation $\lambda_{t'}|_1$ of t for S satisfies that for each position α of p such that $p|_\alpha$ is atomic, $\lambda_{t'}|_1(w'(\alpha)|_2) = X_\alpha$.

Proof: The lemma is shown by induction on the structure of p. Consider the case where $p = \chi_\epsilon :: l_\epsilon$. Suppose that $t \in TL(S)$ with an interpretation I_t^S, $(t, w) \models p$, and $I_t^S(w(\epsilon)|_2) = X_\epsilon$. Since $p' = \chi_\epsilon :: (X_\epsilon, l_\epsilon)$, we have $\varphi(t, I_t^S) \models p'(w(\epsilon))$. Hence, the first condition holds. It is obvious that the second condition holds.

Consider the case where $p = p_1/p_2$. Suppose that $t \in TL(S)$ with an interpretation I_t^S and $(t, w) \models p_1/p_2$. Also suppose that for each position α of p such that $p|_\alpha$ is atomic, $I_t^S(w(\alpha)|_2) = X_\alpha$. Define two mappings w_1 and w_2 so that $w_1(\alpha) = w(1\cdot\alpha)$ and $w_2(\alpha) = w(2\cdot\alpha)$. Then, $(t, w_1) \models p_1$

and $(t, w_2) \models p_2$. By inductive hypothesis, $(\varphi(t), w_1) \models p'_1$ and $(\varphi(t), w_2) \models p'_2$, where $p' = p'_1/p'_2$. Hence $(\varphi(t), w) \models p'$. Conversely, suppose that $t' \in TL(\varphi(S))$ and $(t', w') \models p'_1/p'_2$. Define two mappings w'_1 and w'_2 so that $w'_1(\alpha) = w'(1 \cdot \alpha)$ and $w'_2(\alpha) = w'(2 \cdot \alpha)$. Then, $(t', w'_1) \models p'_1$ and $(t', w'_2) \models p'_2$. By inductive hypothesis, $(\varphi^{-1}(t'), w'_1) \models p_1$ and $(\varphi^{-1}(t'), w'_2) \models p_2$, and hence $(\varphi^{-1}(t'), w') \models p_1/p_2$. Moreover, interpretation $\lambda_{t'}|_1$ satisfies that for each position α of p_i such that $p_i|_\alpha$ is atomic, $\lambda_{t'}|_1(w'_i(\alpha)|_2) = X_\alpha$ ($i \in \{1, 2\}$). Hence the second condition holds.

The case where $p = p_1[p_2]$ can be shown similarly. ∎

Theorem 1: The typing problem for an XPath class $\mathcal{X}$ with respect to an XML schema S is reducible in polynomial time to satisfiability for XPath class $\mathcal{X}$ plus path union in the presence of a DTD $\varphi(S)$.

Proof: Let $(\alpha_1, X_1), \ldots, (\alpha_k, X_k)$ be given pairs of positions of an XPath expression p and types of $S = (N, \Sigma, B, P)$, where $N = \{Y_1, \ldots, Y_n\}$. Let $\varphi(p)$ denote the XPath expression obtained by replacing each atomic subexpression $\chi_i :: l_i$ of p at α_i with $\chi_i :: (X_i, l_i)$, and other atomic ones $\chi :: l$ with $\chi :: (Y_1, l) \cup \cdots \cup \chi :: (Y_n, l)$.

Suppose that there exist $t \in TL(S)$, an interpretation I_t^S, and a mapping w such that $w(\epsilon)|_1 = \epsilon$, $(t, w) \models p$, and $I_t^S(w(\alpha_i)|_2) = X_i$ for each i ($1 \le i \le k$). Then, by Lemma 2, we have $(\varphi(t, I_t^S), w) \models \varphi(p)$.

Conversely, suppose that there exist $t' \in TL(\varphi(S))$ and a mapping w' such that $w'(\epsilon)|_1 = \epsilon$ and $(t', w') \models \varphi(p)$. Then, by Lemma 2 again, we have $(\varphi^{-1}(t'), w') \models p$ and the interpretation $\lambda_{t'}|_1$ of $\varphi^{-1}(t')$ for S satisfies that $\lambda_{t'}|_1(w'(\alpha_i)|_2) = X_i$ for each i ($1 \le i \le k$). ∎

By Theorem 1, many known results on XPath satisfiability in the presence of DTDs can be used to realize tractable combinations of classes of XPath expressions and XML schemas. For example, from the result in [1], the typing problem for an XPath class consisting of downward axes with respect to an arbitrary XML schema S is tractable.

In what follows, we focus on the known results on *disjunction-capsuled DTDs* [3], or *DC-DTDs* for short, since satisfiability of wide XPath classes including path union $\cup$ is tractable under DC-DTDs. A regular expression e is *disjunction-capsuled*, or *DC* for short, if e is in the form of $e_1 e_2 \cdots e_n$ ($n \ge 1$), where each e_i ($1 \le i \le n$) is either

- a symbol in Σ, or
- in the form of $(e'_i)^*$ for a regular expression e'_i.

An XML schema $S = (N, \Sigma, B, P)$ is *disjunction-capsuled*, or *DC* for short, if for each transition rule $X \to a(e)$ in P, e is disjunction-capsuled. Immediately from the results in [3], we have the following corollary of Theorem 1:

Corollary 1: The typing problem for an XPath class $\mathcal{X}$ with respect to a DC XML schema S is tractable if

- $\mathcal{X}$ consists of $\downarrow$, $\downarrow^*$, $\rightarrow^+$, $\leftarrow^+$, and [ ]; or
- $\mathcal{X}$ consists of $\downarrow$, $\downarrow^*$, $\uparrow$, $\uparrow^*$, $\rightarrow^+$, and $\leftarrow^+$.

The known time complexities are $O(|p||S|^4)$ for the former case and $O(|p|^3|S|^3)$ for the latter case.

B. *Efficient enumeration of types of subexpressions*

The first case of Corollary 1 can be extended so that all the possible combinations of types of atomic subexpressions (precisely speaking, the witness mappings ξ introduced below) can be efficiently enumerated. To demonstrate this, we first briefly explain how satisfiability can be determined in this case. We introduce a *schema graph* of a given DC-DTD, which represents parent-child relationship as well as the possible positions of the children specified by the DC-DTD. We also define a satisfaction relation between schema graphs and XPath expressions. The satisfaction relation coincides the satisfiability under DC-DTDs and is decidable efficiently. In what follows, let $D = (N, B, P)$ be a DC-DTD and p be an XPath expression without upward axes. For each $A \in N$, let $P(A)$ denote the content model of the unique rule in P whose left-hand side is A. Moreover, for a DC regular expression $e = e_1 e_2 \cdots e_n$, let $len(e)$ denote the number n of subexpressions of the top-level concatenation.

Definition 1: The *schema graph* [3] $G = (U, E)$ of a DC-DTD $D = (N, B, P)$ is a directed graph defined as follows:

- A node $u \in U$ is either
  - $(\perp, 1, -, B)$, where $\perp$ is a new symbol not in N, or
  - (A, i, ω, A'), where A, $A' \in N$ and $1 \le i \le len(P(A))$ such that A' appears in the i-th subexpression e_i of $P(A)$, and $\omega =$ "$-$" if e_i is a single symbol in N and $\omega =$ "$*$" otherwise.

  The first, second, third and fourth components of u are denoted by $\lambda_{par}(u)$, $pos(u)$, $\omega(u)$, and $\lambda(u)$, respectively.
- An edge from u to u' exists in E if and only if $\lambda(u) = \lambda_{par}(u')$.

If $t \in TL(D)$, then each node of t can be associated with a node of the schema graph of D. More precisely, there exists a mapping θ, called an *SG mapping* of t, from the set of nodes of t to the set of nodes of the schema graph of D with the following properties:

- θ maps the root node of t to $(\perp, 1, -, B)$.
- Let v be a node of t with n children. Then, $\theta(v \cdot j) = (\lambda_t(v), i_j, \omega_{i_j}, \lambda_t(v \cdot j))$, where $1 \le i_j \le len(P(\lambda_t(v)))$, $\omega_{i_j} =$ "$-$" if the i_j-th subexpression of $P(\lambda_t(v))$ is a single symbol in N and $\omega_{i_j} =$ "$*$" otherwise, and $i_j \le i_{j'}$ if $j \le j'$. Moreover, for every maximum subsequence $(v \cdot j) \cdots (v \cdot j')$ such that $i_j = \cdots = i_{j'}$, the i_j-th subexpression of $P(\lambda_t(v))$ matches $\lambda_t((v \cdot j) \cdots (v \cdot j'))$.

A satisfaction relation $\models_{\mathrm{DC}}$ of an XPath expression p by a schema graph G with a witness mapping ξ, which is from the set of positions of p to the set of pairs of nodes of G, is defined as follows (some cases are omitted because of the space limitation):

- $G \models_{\mathrm{DC}} (\downarrow :: l)(\xi(\alpha))$ if there is an edge from $\xi(\alpha)|_1$ to $\xi(\alpha)|_2$ in G and $\lambda(\xi(\alpha)|_2) = l$.

- $G \models_{\mathrm{DC}} (\rightarrow^{+}:: l)(\xi(\alpha))$ if $\lambda_{par}(\xi(\alpha)|_1) = \lambda_{par}(\xi(\alpha)|_2)$, $\lambda(\xi(\alpha)|_2) = l$, and $pos(\xi(\alpha)|_1) < pos(\xi(\alpha)|_2)$ if $\omega(\xi(\alpha)|_1) =$ "$-$" and $pos(\xi(\alpha)|_1) \leq pos(\xi(\alpha)|_2)$ if $\omega(\xi(\alpha)|_1) =$ "$*$".
- $G \models_{\mathrm{DC}} (p_1/p_2)(\xi(\alpha))$ if $G \models_{\mathrm{DC}} p_1(\xi(\alpha \cdot 1))$, $G \models_{\mathrm{DC}} p_2(\xi(\alpha \cdot 2))$, $\xi(\alpha \cdot 1)|_2 = \xi(\alpha \cdot 2)|_1$, and $\xi(\alpha) = (\xi(\alpha \cdot 1)|_1, \xi(\alpha \cdot 2)|_2)$.
- $G \models_{\mathrm{DC}} (p_1 \cup p_2)(\xi(\alpha))$ if $G \models_{\mathrm{DC}} p_i(\xi(\alpha \cdot i))$ and $\xi(\alpha) = \xi(\alpha \cdot i)$ for some $i \in \{1, 2\}$. Moreover, if $j \in \{1, 2\}$ does not satisfy $G \models_{\mathrm{DC}} p_j(\xi(\alpha \cdot j))$, then for any position α' whose prefix is $\alpha \cdot j$, $\xi(\alpha')$ is undefined.

If $G \models_{\mathrm{DC}} p(\xi(\epsilon))$, we say that G *satisfies* p with witness ξ and write $(G, \xi) \models_{\mathrm{DC}} p$.

Theorems 3 and 4 in [3] imply the following theorem:

Theorem 2: If $(t, w) \models p$ with an SG mapping θ, then $(G, \theta \circ w) \models_{\mathrm{DC}} p$. Conversely, if $(G, \xi) \models_{\mathrm{DC}} p$, then there is an SG mapping θ such that $\xi = \theta \circ w$ and $(t, w) \models p$.

Hence, ξ such that $(G, \xi) \models_{\mathrm{DC}} p$ has enough information to give one possible combination of the types of atomic subexpressions of p.

Let S be a given DC XML schema and p be a given XPath expression without path union and upward axes. Let $\varphi'(p)$ denote the expression obtained by replacing each atomic subexpression $\chi :: l$ of p with $\chi :: (Y_1, l) \cup \cdots \cup \chi :: (Y_n, l)$, where $Y_1, \ldots, Y_n$ are all the states of S. Now, the enumeration algorithm is as follows. First, construct the schema graph G of DC-DTD $D = \varphi(S)$. Then, compute the set $\Xi(\alpha)$ of all the pairs (u, u') such that $G \models_{\mathrm{DC}} \varphi'(p)|_\alpha(u, u')$ for each position α of p, in a bottom-up manner with respect to the structure of p. Finally, by traversing Ξ in a top-down manner with respect to α, construct each ξ such that $(G, \xi) \models_{\mathrm{DC}} p$. Each ξ can be enumerated with worst-case delay $O(|p||S|^4)$ time.

V. Conclusion

This paper has discussed typing XPath subexpressions with respect to an XML schema. An algorithm which directly decides the typing problem has been proposed. Moreover, it has been shown that the typing problem is reducible to the XPath satisfiability problem in the presence of DTDs, for which many tractability results are known.

In the definition of the typing problem, we have excluded the path union operator $\cup$ from XPath expressions. Actually, we have found that handling path union is a challenging task. For example, consider an XPath expression $p_1 \cup p_2$. If (ϵ, X) is specified, then we have to check whether X is associated with p_1 *or* X is associated with p_2. On the other hand, if $(1, Y)$ and $(2, Z)$ are specified, then we have to check whether Y is associated with p_1 *and* Z is associated with p_2 simultaneously. In this sense, the meaning of $\cup$ changes depending on the specified pairs of positions and types. Now we are trying to incorporate path union operator and to find a wider condition where the typing problem is solvable efficiently. However, we are also conjecturing that polynomial-time reduction to XPath satisfiability in the presence of DTDs is possible only if atomic subexpressions are specified in the typing problem. It is also interesting to investigate whether efficient enumeration is possible in the second case of Corollary 1.

Acknowledgment

The authors thank the anonymous reviewers for their helpful comments to improve the paper. This research is supported in part by Grant-in-Aid for Scientific Research (C) 23500120 from Japan Society for the Promotion of Science.

References

[1] M. Benedikt, W. Fan, and F. Geerts, "XPath satisfiability in the presence of DTDs," *Journal of the ACM*, vol. 55, no. 2, 2008.

[2] M. Montazerian, P. T. Wood, and S. R. Mousavi, "XPath query satisfiability is in PTIME for real-world DTDs," in *Proceedings of the 5th International XML Database Symposium, LNCS 4704*, 2007, pp. 17–30.

[3] Y. Ishihara, T. Morimoto, S. Shimizu, K. Hashimoto, and T. Fujiwara, "A tractable subclass of DTDs for XPath satisfiability with sibling axes," in *Proceedings of the 12th International Symposium on Database Programming Languages*, 2009, pp. 68–83.

[4] P. T. Wood, "Containment for XPath fragments under DTD constraints," in *Proceedings of the 9th International Conference on Database Theory*, 2003, pp. 297–311.

[5] G. Miklau and D. Suciu, "Containment and equivalence for a fragment of XPath," *Journal of the ACM*, vol. 51, no. 1, pp. 2–45, 2004.

[6] F. Neven and T. Schwentick, "On the complexity of XPath containment in the presence of disjunction, DTDs, and variables," *Logical Methods in Computer Science*, vol. 2, no. 3, 2006.

[7] H. Björklund, W. Martens, and T. Schwentick, "Optimizing conjunctive queries over trees using schema information," in *Proceedings of the 33rd International Symposium on Mathematical Foundations of Computer Science*, 2008, pp. 132–143.

[8] K. Hashimoto, Y. Kusunoki, Y. Ishihara, and T. Fujiwara, "Validity of positive XPath queries with wildcard in the presence of DTDs," in *The 13th International Symposium on Database Programming Languages*, 2011. [Online]. Available: http://www.cs.cornell.edu/conferences/dbpl2011/papers/dbpl11-hashimoto.pdf

[9] M. Murata, D. Lee, M. Mani, and K. Kawaguchi, "Taxonomy of XML schema languages using formal language theory," *ACM Transactions on Internet Technology*, vol. 5, no. 4, pp. 660–704, 2005.

[10] P. Genevès, N. Layaïda, and A. Schmitt, "Efficient static analysis of XML paths and types," in *Proceedings of the ACM SIGPLAN 2007 Conference on Programming Language Design and Implementation*, 2007, pp. 342–351.

Modeling a Data Storage System (DSS) for Seamless Real-Time Information Support from Information Manufacturing System (IMS)

Mohammad Shamsul Islam
Faculty of Engineering & Computing
Dublin City University (DCU)
Dublin, Ireland
Email: mohammad.islam6@dcu.ie

Paul Young
Faculty of Engineering & Computing
Dublin City University (DCU)
Dublin, Ireland
Email: paul.young@dcu.ie

***Abstract*—Nowadays, a large number of enterprises operate in a business time schedule of 24×7. These enterprises need to deliver information as fast as possible for information support. Therefore, the information manufacturing system of the enterprises should have the ability for seamless real-time information support. Data storage system in the information manufacturing system plays the role of providing non interrupted real-time information support. Therefore, 2-Data Storage System oriented information manufacturing system is developed for providing real-time information support. This 2-Data Storage System oriented information manufacturing system can provide real-time information support for a short period of time. However, it is not possible to provide seamless real-time information support by this information manufacturing system. Hence, modeling a data storage system for seamless real-time information support from the information manufacturing system is the purpose of this paper.**

Keywords-data loading; indexing; query processing.

I. Introduction

An IMS (Information Manufacturing System) is an information system that manufactures information from the raw data [23]. The most important component of the IMS is a DSS (Data Storage System). The DSS integrates multiple sources of the system and so contains raw data from multiple sources. Data come from multiple sources are processed by the refreshment function of the availability of data in the DSS. Available data in the DSS are then delivered as information by the execution of query function.

Traditionally, IMS works in the non real-time environment. Single or cluster (replication) DSS oriented IMS is used for providing information support for this non real-time environment. The DSS is updated periodically, typically in a daily, weekly or even monthly basis in the non real-time environment [24]. The DSS needs to update continuously for providing real-time information support with most recent data. Update is done with the refreshment function in the DSS. Continuous execution of the refreshment function (single DSS) and non simultaneous update (cluster DSS) can cause of the poor quality information support from the IMS [6]. More specifically, the poor quality information support occurs for not executing the refreshment and query function simultaneously (single DSS) or the propagation delay for updating the DSS (cluster DSS) of IMS. Therefore, these DSS oriented IMS are not suitable for real-time information support.

Enterprises such as stock brokering, e-business, online telecommunication, health system and traffic systems need to deliver information as fast as possible to knowledge workers or decision-makers who make a decision in a real-time or near real-time environment, according to the new and most recent data captured by an organization's IMS [12]. Therefore, Santos and Berardino [18] as well as Hanson and Willshire [10] developed a 2-DSS oriented IMS for providing real-time information support. However, this 2-DSS oriented IMS cannot provide real-time information support seamlessly. Nowadays, some enterprises need to operate in a business time schedule of 24 × 7 for providing information support in real-time environment. Therefore, the purpose of this research is for modeling a data storage system in the IMS that can provide non-interrupted real-time information support for the business time schedule of 24 × 7. The modeled data storage system is 3-DSS for serving the purpose.

The remaining part of this paper is organized as follows: Section 2 presents the related research of the data storage system. Section 3 describes the 3-DSS. Section 4 shows the regulating procedure of the tasks of the refreshment and query function in the 3-DSS. Section 5 presents the management of the system at the down period of principal 3-DSS and the execution of tasks for restarting the principal 3-DSS again in the system. Experimental evaluation as well as conclusion and future work are shown in Section 6 and Section 7 respectively.

II. Related Research

So far, some researches have been done over DSS (DW, distributed DW, etc.). Bouzeghoub et al. [1] and Vavouras et al. [21] presents the modeling of the data warehouse refreshment process. They explain the difference between loading and refreshment process to these papers. Analyzing the information manufacturing system of many organizations, Mannino and Walter [13] identify that timeliness and availability of data in the DSS are responsible for bad quality information in the IMS. They also find that the refresh period of a system influences the timeliness and availability of data. Theodoratus and Bouzeghoub [20] discuss the data currency quality factors in data warehouses and propose a DW design that considers these factors. An important issue for near real-time data integration is the accommodation of delays, which has been investigated for (business) transactions in temporal active databases in [17]. Vrbsky [22] developed a model to get approximate information from the IMS within a certain time in real-time environment. McCarthy and Risch [16] provide the data structure for execution of real-time queries. Capiello et al. [6] shows that multiple scattered DSS has the lowest degree of integration. It is evident that there is a data quality problem as a result of both long refresh period and propagation delay. On the other hand, single DSS in the IMS has the highest degree of integration, therefore, it does not make the data quality problem. Santos and Berardino [18] present a table structure replication technique to ensure fresh decision support for the real-time or frequently changing data. The table structure replications have two tables, the permanent table and the temporary table. The data stored in a temporary table are transferred to a permanent table for the deterioration of the query response . At the time of transfer, no access is possible in the table of data storage for the information support. Hanson and Willshire [10] developed a faster data warehouse model providing an auxiliary structure for quick query response. This is also a 2-DSS oriented IMS. It has a temporary table as well where only data will be loaded and no administrative overhead such as indexing will be done. In this model storage capacity of temporary tables is limited as it is installed in the non-volatile NVRAM. After fulfilling the 95% of the temporary tables, data is transferred to the permanent table. Therefore, no data access will be possible in this period. As a result, 24 × 7 services will be not possible with these 2-DSS oriented IMS seamlessly.

The data storage model of this research does not need to transfer data by pushing the system to offline or by stopping the system. Therefore, it will be possible to provide 24 × 7 services seamlessly with this DSS model. Further, it will update the data in the DSS in real-time manner for providing the real-time information support.

III. 3-Data Storage System (3-DSS)

According to [1][13][18], refreshment and query function execute in a data storage system of the IMS to make the data available and for the information support respectively.

Refreshment Function: This is a complex process comprising the tasks, such as data loading, indexing and propagation of data for synchronizing data in the information manufacturing system (IMS) [1][13][18].

Data loading: Key activities of data loading include extraction, transformation, integration, cleaning etc. Therefore, storage of manipulating [insert, update] data are to extract from the sources , then, transformed data if the source data are in the different format. After that, extracted and transformed data are to integrate and to clean for loading data in the data storage system [1][13][18].

Indexing: Update the index for newly loaded data or delete data to align the data in the data storage system [18]. Indexing determines the effective usability of data collected and aggregated from the sources and increases the performance of the data storage system for information support [1][13].

Propagation of Data: Data is propagated through the refreshment process for synchronizing the data of multiple DSSs of the system.

Query Function: This function of data storage system in IMS is done by the query processing task. A requested query of the user is processed in the data storage system for delivering the information to the user.

In the 3-DSS, three individual DSS are mutually interconnected with each other. The tasks of the refreshment and the query function of data storage system work simultaneously in three individual DSS . Therefore, data loading and indexing with updated data propagation task of the refreshment function work in two individual DSS of 3-DSS and another DSS of 3-DSS executes the task of the query function at the same time. After each successive period, the tasks of the refreshment and the query function of data storage system will interchange with cyclic order. As, propagation of manipulated data in the DSS is done simultaneously at the working period of the task of the functionalities, there may not have propagation delay. Therefore, 3-DSS will hold exact the same data. The 3-DSS is shown in Figure 1.

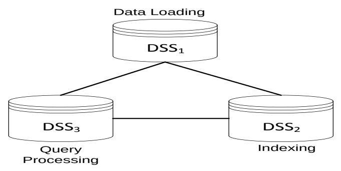


Figure 1. 3-DSS

In the following sub-sections, the execution process of 3-DSS, protocol of 3-DSS and the partitioning procedure of 3-DSS will be discussed.

A. Execution Process of 3-DSS

Suppose, DB1, DB2 and DB3 is three individual DSS for 3-DSS. These three DSS are mutually interconnected with each other. Now, if DB1 store some data from operational data sources, DB2 and DB3 must have the same data. The tasks of the query and refreshment function work simultaneously in these three data storage systems. There must have a synchronization of starting and finishing time of the tasks of the query and refreshment function of these three data storage systems. Manipulated data from operational data sources will load into one database. At the same time, another database will do the indexing and updated data propagation task for synchronizing data with other two DSS and the third one will be used for query processing. The indexed and query processing database lead the process. When the indexing with the propagation of updated data and the query processing are finished, the tasks of the refreshment and query function of data storage system will interchange with cyclic order. The rotation algorithm for the interchanging process is given in the Figure 2.

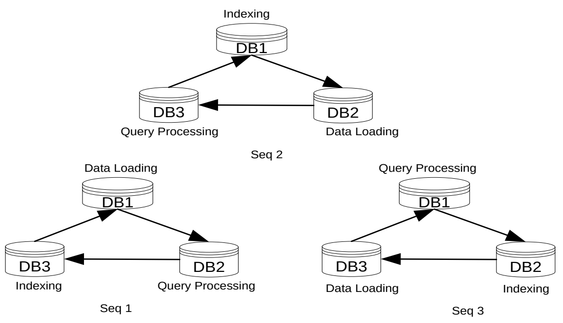


Figure 2. Rotation of the tasks of functionalities in 3-DSS

The algorithm for the rotation of function (Data Loading, Indexing and Query Processing) in data storage system is shown in Figure 3.

Rotation of Tasks of Functionalities in 3-DSS (Data Loading, Indexing with update propagation, Query Processing)

```
DB1 = Database 1, DB2 = Database 2, DB3 = Database 3

Step   1: DB1=>Data Loading, DB2=>Nothing, DB3=>Nothing

Step 2: DB1=>Indexing and send updated data to DB2, DB2=>Data Loading, DB3=>Nothing

Step 3: DB2=>Indexing and send updated data to DB3, DB3=>Data Loading, DB1=>Query Processing

Step 4: DB3=>Indexing and send updated data to DB1, DB1=>Data Loading, DB2=>Query Processing

Step 5: DB1=>Indexing and send updated data to DB2, DB2=>Data Loading, DB3=>Query Processing

Step 6: Repeat Step 3, 4 and 5.
```

Figure 3. Algorithm for Rotation of Tasks of Functionalities in 3-DSS

In the algorithm, Steps 1 and 2 indicate the initialization of the system. Data come from multiple sources are loaded into DB1 in Step 1. DB1 executes the indexing task and send the updated data to DB2 and DB2 loads the updated data and source data simultaneously in Step 2. Each task of the functionalities of data storage system works simultaneously in Steps 3, 4 and 5. As, data have been loaded into DB2 in Step 2, DB2 is indexed in Step 3 and send the updated data to DB3. At the same time, DB3 loads the manipulated data including the updated data of DB2. Further, DB1 provides the information by processing the query request of the system in Step 3. Steps 4 and 5 will follow the same process but interchange the roles of each DB of the system. Therefore, Steps 3, 4 and 5 will continue repeatedly in the system.

B. 3-DSS Protocol

A protocol is a set of rules for regulating a system [9]. 3-DSS is the data storage system where multiple tasks will execute simultaneously. Therefore, 3-DSS are to maintain a set of rule for avoiding the cumbersome operations of the 3-DSS in the IMS.

1. N, 2N, 3N,...............NN amount of manipulating new data will be loaded each time to be available in the DSS after completion of the task of the refreshment function.
2. Three individual DSS of the 3-DSS will be located contiguously in the same place.
3. Functionalities do not interchange if the indexing task is in progress. It means that functionalities do not interchange before the completion of indexing task.
4. Functionalities do not interchange if the propagation of update data from one DSS to another DSS is in progress. Otherwise, the propagated data receiver DSS can not load the all new data of the sender DSS.
5. Functionalities do not interchange if query processing task is in progress. Otherwise, the query can process an incomplete query result.
6. Throughput of the three interconnected network link of 3-DSS should be same.
7. Rotation of the tasks of functionalities will be clockwise cyclic order like the Figure 2.
8. Previous DSS of indexing DSS will always process the query and next DSS of indexing DSS always load the manipulated data. It is seen in Figure 2 that in every sequence (seq) previous DSS of indexing DSS process the query.

9. Previous DSS of query processing DSS will always load the manipulated data and next DSS of query processing will always indexed the loaded data of the DSS. It is seen in Figure 2 that in every sequence (seq) previous DSS of query processing DSS load the manipulated data.
10. Previous DSS of data loading will always indexed the loaded data of the DSS and next DSS of data loading DSS will always process the query. It is seen in Figure 2 that in every sequence (seq) previous DSS of data loading DSS indexed the loaded data.
11. Interchange of the tasks of functionalities will be done after the completion of the indexing and query processing tasks.

C. *3-DSS Partitioning*

Partitioning is the technique of fragmenting large relations (tables) into smaller ones. In the large DSS, (tables), if manipulation of data (insertion, update, delete) is done, it needs more time to rebuild the indices of the large DSS (tables). As a result, a requested query may not execute in time or may provide a poor query result. The partitioning of a large table can resolve this problem. In a partitioning DSS (tables), the problem that created at the time of index rebuilding for the manipulation of data is limited only in a particular partition. Therefore, except the partitions where data is being manipulated, other partitions of the DSS (tables) can provide the query result for the requested query as the index rebuilding process is limited to the certain partition. Additionally, it needs less time to rebuild the index than the non-partitioned DSS (tables) as the volume of data of each partition will be certain. There are two ways to partition a DSS: vertically and horizontally. Vertical partitioning involves splitting the attributes (columns) of a DSS (tables), placing them into two or more DSS (tables) linked by the DSS (tables) primary key. Horizontal partitioning involves splitting the tuples of a DSS (table), placing them into single or more DSS (tables) with the same structure. For keeping the certain volume of data in the DSS (table), horizontal partitioning will be used in the 3-DSS. There are two types of horizontal partitioning: primary and derived. Primary horizontal partition (HP) of a DSS (table) is performed using attributes defined on that DSS (table). On the other hand, derived horizontal partition is the fragmentation of a DSS (table) using the attributes defined on another DSS (tables) [4]. Horizontal partitioning for 3-DSS is discussed in below,

Let, F is the primary or fact table, D is the derived or dimensional table. Example of a primary (fact) relational table and derived (dimensional) table are depicted in Figure 4.

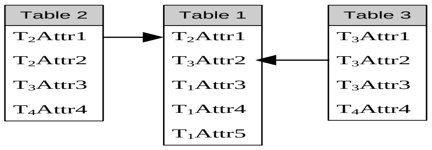


Figure 4. Primary (fact) relational table and derived (dimensional) table

In Figure 4, Table 2 and Table 3 is primary table. On the other hand, Table 1 is derived table. Fragmentation of Table 1 depends on Table 2 and Table 3. If primary Table 2 and Table 3 are manipulated, Table 1 must have to be manipulated. Therefore, tuple of Table 1, Table 2 and Table 3 will be horizontally partitioned simultaneously considering the instruction of the predicate.

A predicate is the Boolean expression over the attributes of a relational table and constants of the attribute's domains. Horizontal partitioning can be defined as a pair (T, Φ), where T is a relation and Φ is a predicate. This predicate partitions T into at most 2 fragments with the same set of attributes. The first fragment includes all tuples of t of T which satisfy Φ, i.e., t = Φ. The second fragment includes all tuples t of T which does not satisfy Φ, i.e., t ≠ Φ. It is possibly one of the fragments to be empty if all tuples of T either satisfy or do not satisfy Φ.

Let Φ = (counted tuple = N), which results into fragment horizontally where tuple of a relational table will be counted. If the condition of the predicate Φ is true, relational table will be fragmented into 2 partitions. The first partition will hold N numbers of the tuple. If manipulation of data in the information manufacturing system (IMS) is stopped after being partitioned, the second partition will remain empty. When manipulation of data in the IMS is continued and the total insertion of tuple reaches to N number, this partition will again fragment into two pieces. This partitioning process will continue as long as the information manufacturing system is not stopped. This single table partitioning process can be applied to the multiple relation tables of the DSS in the IMS. The horizontal Partitioning algorithm and the partitioning algorithm of the 3-DSS are given in Figure 5 and Figure 6, respectively.

```
Horizontal Partition ( DSS )
Existing Tuple := M,      New Tuple := N
Delete Frequency := D,      Insert Frequency := I
Partition Tuple Limit := T

1. Count Existing Tuple M
2. M := | M | - | D |
3. If ( M == T )
4. Create Partition
5. Else
6. Insert New Tuple N
7. Count New Tuple N
8. N := | I |
9. Count Total Tuple O
10. O := | M | + | N |
11. If ( O == T )
12. Create Partition
13. Else
14. Insert More Tuple Until ( O == T )
```

Figure 5. Horizontal Partitioning algorithm for 3-DSS

```
Partitioning ($DSS_1$, $DSS_2$, $DSS_3$)
1. Locate the DSS that execute L ( )
2. If ( DSS == $DSS_1$ )
3. Execute Horizontal Partition ( DSS) for $DSS_1$ in Loading Period L(t)
4. Else If ( DSS == $DSS_2$ )
5. Execute Horizontal Partition ( DSS) for $DSS_2$ in Loading Period L(t)
6. Else
7. Execute Horizontal Partition ( DSS) for $DSS_3$ in Loading Period L(t)
8. Go On Step1 and Repeat
```

Figure 6. Partitioning algorithm for 3-DSS

Partitioning of DSS of the 3-DSS will be done by following the partitioning algorithm given in Figure 6. According to this algorithm, DSS will be located for the partitioning in the executing period of data loading of a particular DSS. Then, the horizontal partitioning algorithm will be applied for partitioning the DSS of the 3-DSS. The horizontal partitioning algorithm is shown in Figure 5. In this algorithm, Existing and new tuples have to calculate for the horizontal partitioning. Deleting of a tuple from DSS will detect the tuple from the existing tuple. On the other hand, insertion of the tuple will count as a new tuple. A DSS will be partitioned if total tuples of the DSS is reached in the partition limit N. Therefore, the existing tuple will be counted by deducting the deleted tuple from the DSS. Hence, the number of existing tuples will be counted for checking whether the existing tuple is in partition limit or not. If, it is in partition limit, the partition will be created. Otherwise, new tuple will be inserted in the DSS. Therefore, new tuple will be counted by the number of new insertions. After that, total tuple will be counted by adding existing tuple with new tuple. Henceforth, it will be checked for whether a counted number of total tuples are in partition limit or not. If total tuple equals to the partition limit, the partition will be created. Otherwise, more tuple has to be inserted until the total tuple reaches to the range of the partition limit of the tuple.

IV. REGULATING PROCEDURE OF THE TASKS OF REFRESHMENT & QUERY FUNCTION OF 3-DSS

3-DSS executes three individual DSS simultaneously in the information manufacturing system (IMS). The tasks of the functionalities of data storage system work in these three individual DSS of the 3-DSS. These tasks also need to interchange in the 3-DSS. Further, this 3-DSS is to give an assurance of quick update of data for the real-time information support. As a result, DSS of the 3-DSS is fragmented with a partitioning procedure. Therefore, there must have a coordination and communication among the tasks of the functionalities of the 3-DSS for doing the simultaneous operation, interchanging of the tasks of the functionalities and the partitioning of the DSS. The regulator algorithm will play the role for making the coordination and communication among the tasks of the functionalities with the help of some other algorithms. The regulator algorithm is given in Figure 7.

```
Regulator ($DSS_1$, $DSS_2$, $DSS_3$)
(Loading), (Indexing and propagation) and (query processing) is
represented as L, Ix and IS respectively
3-DSS is controlled by a controller thread represented as CT

Step 1: Create Thread 1, Thread 2, Thread 3 and Controller Thread as
$T_1$, $T_2$, $T_3$ and CT
Step 2: $T_1 \rightarrow$ Ix ( ),  $T_2 \rightarrow$ IS ( ),  $T_3 \rightarrow$ L ( )
Step 3: $T_1 \rightarrow$ L ( ),  $T_2 \rightarrow$ Ix ( ),  $T_3 \rightarrow$ IS ( )
Step 4: $T_1 \rightarrow$ IS ( ),  $T_2 \rightarrow$ L ( ),  $T_3 \rightarrow$ Ix ( )
Step 5: CT $\rightarrow$
        I.  Execute Synchronizing Agent Algorithm
        II. Execute Partitioning Algorithm
Step 6: Go on step 2 and repeat from step 2
```

Figure 7. Regulator algorithm for 3-DSS

In the regulator algorithm of the 3-DSS, four threads will be created for the execution of the operation of 3-DSS. Thread 1, thread 2 and thread 3 are created for the simultaneous operation of the tasks of the functionalities of the 3-DSS in three individual DSS. On the other side, controller thread CT is constructed for the execution of the synchronizing agent and the partitioning algorithms. Synchronizing agent and partitioning algorithms are shown in Figure 8 and Figure 6 respectively. As thread 1, thread 2 and thread 3 work simultaneously, so, when thread 1 executes indexing function, thread 2 and thread 3 will execute the query and loading function, respectively. This will continue until thread 1, thread 2 and thread 3 get a message from the synchronizing agent of controller thread to change their tasks of the functionalities. If thread 1, thread 2 and thread 3 get a message from the synchronizing agent of controller thread to change their activities, then, thread 1 executes the function for loading task, thread 2 and thread 3 execute function for indexing and query processing tasks respectively. These activities of threading will continue until these threads do not get any message to interchange their activities. As soon as, each thread gets the message to interchange their activities, thread 1 will start query processing, thread 2 will start loading of data and thread 3 will start the indexing task. Hence, the sequence of activities of among threads will continue until the system is stopped by the user or any other reasons. For the shortening of indexing period, DSS will be partitioned by partitioning algorithm after a certain number of storage data. This partitioning process will provide the service from the controller thread together with synchronizing agent. Details of the synchronizing agent algorithm are given in Figure 8.

```
Synchronizing Agent ($DSS_1$, $DSS_2$, $DSS_3$)
1. Ix ( ), L ( ) and IS ( ) is executing simultaneously in $T_1$, $T_2$ and $T_3$ by rotation.
2. Ix ( ) And IS ( ) inform CT of its completion status
3. L ( ) sends a message to CT to get the status information of Ix ( ) And IS ( )
4. Waiting for the reply of CT about the status of Ix ( ) And IS ( )
5. IS ( ) sends a message to CT after its completion to get the status information of Ix ( )
6. Waiting for the reply of CT about the status of Ix ( )
7. Ix ( ) sends a message to CT after its completion to get the status information of IS ( )
8. Waiting for the reply of CT about the status of IS ( )
9. Completion of the status of Ix ( ) And IS ( ) = Boolean Value
10. If Boolean Value of Ix ( ) = False AND Boolean Value of IS ( ) = False
    10.1    Continue Current Functions in $T_1$, $T_2$ and $T_3$
11. Else If Boolean Value of Ix ( ) = True AND Boolean Value of IS ( ) = False
    11.1    Continue Current Functions in $T_1$, $T_2$ and $T_3$
12. Else If Boolean Value of Ix ( ) = False AND Boolean Value of IS ( ) = True
    12.1    Continue Current Functions in $T_1$, $T_2$ and $T_3$
13. Else
    13.1    $T_1$ : Move to Next Function and Execute
    13.2    $T_2$ : Move to Next Function and Execute
    13.3    $T_3$ : Move to Next Function and Execute
14. Set
    14.1    Next Function $\longrightarrow$ Current Function in $T_1$
    14.2    Next Function $\longrightarrow$ Current Function in $T_2$
    14.3    Next Function $\longrightarrow$ Current Function in $T_3$
```

Figure 8. Synchronizing Agent algorithm for 3-DSS

Figure 8 presents the algorithm for the interchange process among of the tasks of the functionalities of the 3-DSS. It shows, how the tasks of the functionalities of the 3-DSS communicate with each other for providing their service rotationally in three individual DSS of the 3-DSS. According to the regulator algorithm of the 3-DSS, each task of the functionalities of the 3-DSS (indexing, loading and query processing) executes simultaneously in three separate threads by rotation. Further, each individual DSS of the 3-DSS changes role after a certain period of time. Query processing and indexing tasks are not possible to stop in the middle of the execution or before the completion of these tasks. Therefore, the indexing and the query processing tasks can be called dependent tasks. On the other hand, it is possible to stop the loading of data at any moment of time. Therefore, this task could be called independent task. Hence, the interchanging process of the tasks of the functionalities in the 3-DSS depends on both the indexing and the query processing tasks. In line 1 of algorithm indicates that step 1, step 2 and step 3 of regulator algorithm will be executed simultaneously by rotation. In line 2, function of the indexing and the query processing tasks will inform their current status to the controller thread. Then, the function of the loading task will send the message to the controller thread to know the current status of the indexing and the query processing function in line 3. The controller thread will deliver a reply about the status of the indexing and the query processing in line 4. In line 5 and 6, the query processing function will send a message to the controller thread to know the status of the indexing function after the completion its task and wait for the reply. Similarly, in line 7 and 8, indexing function will do the same and wait for the reply about the status of the query function. Now, from line 10 to line 13 shows that whether the tasks of the functionalities of the 3-DSS will be interchanged or not. If the completion status of the query processing or the indexing function is false, the tasks of the functionalities of the 3-DSS will not be interchanged. So, from line 10 to line 12, current function is continued in thread 1, thread 2 and thread 3. In line 13, the completion status of both the query processing and the indexing function is true, so, the tasks of the functionalities of the 3-DSS is interchanged. For this reason, current function of each thread is stopped and move to the next function. Current function move to the next function in the regulator algorithm mean that indexing, loading and query processing function execute in step 1, step 2 and step 3 respectively in thread 1; query processing, indexing and loading function execute in step 1, step 2 and step 3 respectively in thread 2 and loading, query processing and indexing function execute in step 1, step 2 and step 3 respectively in thread 3. Now, if the current function of step 1 of thread 1, thread 2 and thread 3 are indexing, query processing and loading function respectively, next function will be the function of thread 1, thread 2 and thread 3 of step 2 and so on for step 2 and step 3. Therefore, when the next function will be prepared for execution, it will be executed as current function. Finally, in line 14, next function is set and executed as the current function in thread 1, thread 2 and thread 3. The whole process executes repeatedly to continue the interchanging of the tasks of the functionalities in three individual DSS simultaneously in the 3-DSS.

V. Additional Tasks for Handling the System for the Failure of Principal 3-DSS

Two 3-DSS can be installed in IMS for the real-time information support seamlessly. One 3-DSS can be said principal 3-DSS. Another can be told alternative 3-DSS. The principal 3-DSS can be down at any moment of time in the system for the crashing or other difficulties for any of the DSS of the principal 3-DSS. An alternative 3-DSS can facilitate the seamless real-time time information support at the down period of the principal 3-DSS. Therefore, some of the additional tasks have to include in the regulator algorithm of Figure 7 for handling the system for the failure of the principal 3-DSS. These tasks will be executed in the controller thread. The additional tasks that will be included in the controller thread are,

Sending the Source Data to Alternative 3-DSS: The source data will be stored in the alternative 3-DSS at the same time of loading data in the principal 3-DSS. It is done by sending source data to one DSS of the alternative 3-DSS. This DSS then replicates the data to other two DSS of the alternative 3-DSS.

Recovery System: The log based or the shadow paging recovery system described in [19] can be used for recovering the data for crashing of 3-DSS.

Activation of Alternative 3-DSS: Alternative 3-DSS will be activated for information support just after the failure of principal 3-DSS. This alternative 3-DSS will then work just like the principal 3-DSS.

Storage of Data in Temporary DSS: The source data will also be stored in the temporary DSS for supporting the principal 3-DSS. It will store data as long as principal 3-DSS will be down.

Transferring the Temporary DSS Data to Principal 3-DSS: After fixing the problem of the principal 3-DSS, the stored data in the temporary DSS will now be transferred to the principal 3-DSS.

Restart the Principal 3-DSS: Now, the principal 3-DSS will be restarted and the activities of principal 3-DSS will be released from the alternative 3-DSS. This alternative 3-DSS will then perform its general task.

Now, the controller thread of the regulator algorithm in Figure 7 can be written as:

```
Controller Thread (CT)

If Principal 3-DSS ≠ Fail
    CT ──────►
        I. Execute Synchronizing Agent Algorithm
        II. Execute Partitioning Algorithm
        III. Sending the Source Data to Alternative 3-DSS

Else
    CT ──────►
        I. Execute Recovery System for Principal 3-DSS
        II. Activation of Alternative 3-DSS
        III. Execute Synchronizing Agent Algorithm
        IV. Storage of Data in Temporary DSS
        V. Transferring the Temporary DSS Data to Principal 3-DSS
        VI. Restart the Principal 3-DSS
```

Figure 9. Additional tasks in controller thread for handling both principal and alternative 3-DSS

In Figure 9, if principal 3-DSS is not in failing state, then, the controller thread will execute the general 3-DSS tasks and Sending the source data to alternative 3-DSS task. On the other hand, if the principal DSS is in failing state, then, the controller thread will execute recovery system for principal 3-DSS. Then, it will activate the alternative 3-DSS. At the same time, synchronizing agent algorithm will start to work on the alternative 3-DSS. Storage of data in temporary DSS, transferring the temporary DSS data to principal 3-DSS and restarting the principal 3-DSS tasks will also execute in the controller thread. Once the principal 3-DSS restarts, the controller thread will again execute the tasks of not failing state.

VI. EXPERIMENTAL EVALUATION

The experiments have been done for the 2-DSS and 3-DSS oriented IMS. 2-DSS is used as the benchmark for showing the real-time information support of 3-DSS. Temporary DSS is considered for the experiment of 2-DSS. Only loading of data task work in the temporary DSS of the 2-DSS for providing the real-time information support . The tasks of the DSS functionalities work in three individual DSS simultaneously in the 3-DSS. Four machines were used for doing the experiments. Among the four machines, three machines were used for implementing the 3-DSS, another is used as a server for controlling the 3-DSS, inserting data from the sources to the 3-DSS and sending the query request to the 3-DSS. Server machine and one more machine were used for the experiment of the 2-DSS. Multi core 2.2 Ghz processors, 4GB RAM and 5400 r.p.m hard drive were used for the server machine. The rest of the machine used single core 1.69 Ghz processor, 1GB RAM and 5400 r.p.m hard drive. SQL server was the database software for creating the data storage system.

For doing three experiments, 1GB data was stored in both the 2-DSS and the 3-DSS oriented IMS. Fifty thousand new rows (tuple) were extracted from the sources and stored in the 2-DSS and the 3-DSS with the refreshment function at the time of each individual experiment for delivering the data for the query request. One experiment of 3-DSS was done without storing 1GB data in the 3-DSS. In the real world, user may send the query request in the refreshment period. For this reason, query and refreshment functions executed simultaneously in these experiments. Query request for retrieving all newly inserted data was sent repeatedly after each single minute. Therefore, the query result was delivered for each respective query request. These query results were measured to get the result for both the 2-DSS and the 3-DSS oriented IMS. Start of data insertion time means the first data insertion time from the source to the DSS and query delivery time is the time the query result is delivered. Therefore, the distance between start of data insertion time and query delivery time is measured by subtracting query delivery time from the start of data insertion time. Volume of query result data is measured by dividing the number of inserted data in the DSS with the total data for insertion. The results are given in Table I, Table II, Table III, and Table IV.

TABLE I. EXPERIMENTAL RESULT OF 2-DSS (BENCHMARK DSS)

Query Request No.	Distance between Start of Data Insertion Time and Query Delivery Time (Minute)	Volume of Query Result Data (%)
1	1	11.00
2	2	20.80
3	3	31.20
4	4	41.27
5	5	50.30
6	6	61.16

TABLE II. EXPERIMENTAL RESULT OF 3-DSS (NO EXISTING DATA)

Query Request No.	Distance between Start of Data Insertion Time and Query Delivery Time (Minute)	Volume of Query Result Data (%)
1	1	10.70
2	2	20.30
3	3	30.60
4	4	41.15
5	5	50.00
6	6	60.80

TABLE III. EXPERIMENTAL RESULT OF 3-DSS (1GB EXISTING DATA)

Query Request No.	Distance between Start of Data Insertion Time and Query Delivery Time (Minute)	Volume of Query Result Data (%)
1	1	10.30
2	2	18.20
3	3	26.37
4	4	33.32
5	5	41.60
6	6	49.78

TABLE IV. EXPERIMENTAL RESULT OF 3-DSS (PARTITIONING)

Query Request No.	Distance between Start of Data Insertion Time and Query Delivery Time (Minute)	Volume of Query Result Data (%)
1	1	10.90
2	2	20.40
3	3	30.38
4	4	41.00
5	5	50.10
6	6	61.10

Table I is representing the experimental result for the 2-DSS. Table II, Table III and Table IV are showing the experimental result for the non-partitioned 3-DSS, the non-partitioned 3-DSS with 1 GB existing data and the partitioned 3-DSS with 1 GB existing data respectively. Query result of Table I and Table II is almost the same. There is a big difference between the query result of Table I and Table III. Indexing of data was the cause for this difference. It was not visible in the Table II for the low volume of data (only newly data was inserted and no existing data were there). As, partition was done after 1GB of data , Table IV presents almost the same volume of query result for each query request like Table I. Therefore, it can be said that 3-DSS can provide real-time information support. Further, as 3-DSS does not need the data transfer from non-indexed DSS to indexed DSS, it can provide information support seamlessly.

VII. Conclusion and Future Work

This paper showed that the 3-DSS model can provide seamless real-time information support from the IMS. It is quite possible to down any of the DSSs of the 3-DSS at the period of execution in the real world. Therefore, there should have a redundant or replication system of 3-DSS to provide information support service in the down period. Additionally, a recovery system needs to develop for this 3-DSS for recovering the data for the failure of the system for crashing or some other reasons. Replication and recovery system are described briefly in this paper. Further, details work is needed on the dynamic partitioning system. Therefore, future work will be conducted on the replication system of the 3-DSS, the recovery system of the 3-DSS and the dynamic partitioning system of the 3-DSS in a broader aspect. Additionally, comparison of 3-DSS oriented IMS with the 2-DSS oriented IMS will also be the future research work.

References

[1] M. Bouzeghoub, F. Fabret and M. Matulovic-Broqué, “Modeling Data Warehouse Refreshment Process as a Workflow Application”, Proceedings of the International Workshop on Design and Management of Data Warehouses, 1999, pp. 6.1-6.12.

[2] D.P. Ballou and H.L. Pazer, “Modeling data and process quality in multi-input, multi-output information systems”, Management Science, vol. 31, 1985, pp. 150–162.

[3] C. Batini and M. Scannapieco, “Data Quality: Concepts, Methodologies and Techniques”, Publisher: Springer, Berlin, Germany, 2006.

[4] L. Bellatreche, K. Karlapalem, M. Mohania and M. Schneider, “What can partitioning do for your data warehouses and data marts?”, IEEE, 2000, pp. 437-445.

[5] C. Cappiello, C. Francalanci and B. Pernici, “Data Quality and Multichannel Services”, PhD Thesis. Politecnico di Milano, 2005.

[6] C. Cappiello, C. Francalanci and B. Pernici, “Time-Related Factors of Data Quality in Multichannel Information Systems”, Journal of Management Information Systems, vol. 20, 2003, pp. 71-92.

[7] C. Cappiello, C. Francalanci and B. Pernici, “A Self-monitoring System to Satisfy Data Quality Requirements”, Springer Verlag, vol. 3761, 2005, pp. 1535-1552.

[8] C. Cappiello and M. Helfert, "Analyzing Data Quality Trade-Offs in Data-Redundant Systems", Interdisciplinary Aspects of Information Systems Studies , Physica-Verlag HD, 2008, pp. 199-205.

[9] B.A. Forouzan, C. Coombs and S.C. Fegan, "Data Communications and Networking", Publisher: Tata Mcgraw-Hill, 2003.

[10] J.H. Hanson and M.J. Willshire**, "**Modeling a Faster Data Warehouse", IEEE, 1997, pp. 260-265.

[11] Y. Hu, S. Sundara and J. Srinivasan, "Supporting Time-Constrained SQL Queries in Oracle", Proceedings of the 33rd international conference on Very large data bases, 2007, pp. 1207-1218.

[12] W.H. Inmon, R.H. Terdeman, J. Norris-Montanari and D. Meers, "Data Warehousing for E-Business", J. Wiley & Sons, 2001.

[13] M.V. Mannino and Z. Walter, "A framework for data warehouse refresh policies". Decision Support Systems, vol. 42, 2006, pp. 121-143 .

[14] C.L. Pape and S. Gancarski¸ "Replica Refresh Strategies in a Database Cluster", Springer-Verlag, LNCS vol. 4395, 2007, pp. 679-691.

[15] C.L. Pape, S. Gancarski and P. Valduriez, "Refresco: Improving Query Performance Through Freshness Control in a Database Cluster", Springer-Verlag, vol. 3290, 2004, pp. 174-193.

[16] T. Padron-McCarthy and T. Risch, "Performance-Polymorphic Execution of Real-Time Queries**.** Workshop on Real-Time Databases: Issues and Applications", Newport Beach, California, USA, 1996. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.48.2149. [Accessed 2nd January 2013].

[17] J.F. Roddick and M. Schrefl, "Towards an Accommodation of Delay in Temporal Active Databases", 11th Australasian Database Conference (ADC), 2000.

[18] R.J. Santos and J. Bernardino, "Real-Time Data Warehouse Loading Methodology", ACM, vol. 299, 2008, pp. 49-58.

[19] A. Silberschatz, H.F. Korth and S. Sudarshan, "Database System Concepts". Publisher: Mcgraw-Hill, 1997.

[20] D. Theodoratus and M. Bouzeghoub, "Data Currency Quality Factors in Data Warehouse Design", Int. Workshop on Design and Management of Data Warehouses (DMDW), 1999.

[21] A. Vavouras, S. Gatziu and K.R. Dittrich, "Modeling and Executing the Data Warehouse Refreshment Process",IEEE, 2000, pp. 66-73.

[22] S.V. Vrbsky, "A data model for approximate query processing of real-time databases", ACM Data & Knowledge Engineering, vol. 21, 1996, pp. 79-102.

[23] R.Y. Wang, M. Ziad and Y.W. Lee, "Data Quality", Publisher: Kluwer Academic, 2001.

[24] T. Zurek and K. Kreplin, "SAP Business Information Warehouse From Data Warehousing to an E-Business Platform", 17th Int. Conf. on Data Engineering (ICDE), 2001.

[25] K. Zdenek , M. Kamil, M. Petr and S. Olga, "On updating the data warehouse from multiple data sources", Springer, vol. 1460, 1998, pp. 767-775.

Knowledge Enhanced Framework for designing e-Workflow Systems

Farhi Marir, John Ndeta
Knowledge Management Research Centre,
Faculty of Computing, London Metropolitan University,
London, UK
e-mail: f.marir@londonmet.ac.uk, ndetaj@hotmail.co.uk

***Abstract*- Traditional workflow framework in implementing e-business processes has ignored the worker knowledge, the user feedback and problems resulting of incompatibility of inter-organisation e-business processes. The limitation of this framework comes from its behavioural perspective, which focuses only on monitoring and detecting problems in the execution of homogenous intra-organisation business processes. As a result, e-business processes designed and implemented using this traditional workflow framework become increasingly inadequate in the new e-business environment which is characterised by an increasing pace of mobility, business collaboration between complementary organisations, discontinuous and unforeseen change in the business processes. This paper presents results of a research investigation confirming these limitations and proposes a knowledge enhanced framework for design e-workflow that captures the dynamics of e-business process.**

Keywords-e-workflow; framework; e-business process; knowledge management; design pattern

I. INTRODUCTION

Workflow management systems are designed to support business process modelling. Business process modelling is an effective tool for managing organisational change and is known to have brought benefits to many organisations. Organisations and their business processes undergo changes from time to time, and in some cases these changes are dynamic, discontinuous and complex. Organisations change either through anticipation of surprise (proactive) efforts to become more competitive or in response to a need to maintain competitiveness in a changing business environment [1]. During the last decade, workflow technology has become readily available [2]. They are built using a traditional framework for workflow modelling “Fig. 1” which is composed of four perspectives: functional, informational, organisational and behavioural Functional perspective decomposes process functionality into a task hierarchy that can be allocated to actors (human or software agents). Informational perspective however, describes the information objects that are consumed and produced. It also describes the business data, documents, and electronic forms that are transported between actor and the files and databases that store persistent application information. Organisational perspective specifies the roles and actors that are involved in the workflow execution and describes how the organisation configures its resources to perform business processes. Behavioural perspective specifies when and under which conditions a workflow is executed in which routes define the way the information and the knowledge are channelled through the different steps of the process and the rules reflecting the constraints of the business policies and practices.

A major limitation of this traditional framework is that it can typically only support simple, static and predictable business processes, such as insurance and travel claim processing. It is well understood for traditional intra-organisational workflow applications, which span only one organisation and are characterised by structured and predictable business processes and business environment. However, it is increasingly inadequate in the new online business (e-business) era that is characterised by increasing complexities, unpredictable and discontinuous change. Also, with the emergence of the Internet, e-workflow design and evolution within e-business environments where most organisations cooperates or outsource parts of their internal process, a new framework with additional perspectives and modelling concepts are needed for specifying the processes and knowledge flow between these organisation [1][3][4][5]. Furthermore, the perspectives of the traditional framework are not parametrically modelled and as a result they are not flexible and difficult to change.

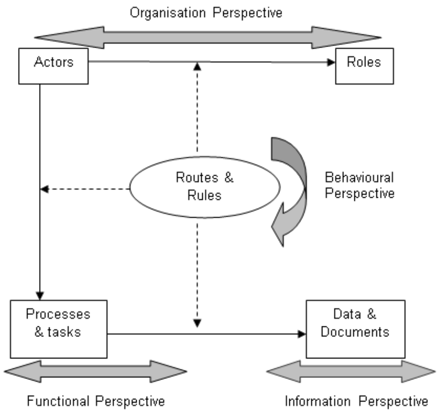


Figure 1. Traditional Framework.

These limitations are consolidated by several research works. Manolescu [6] started from the observation that current workflow systems do not provide the workflow functionality required in object-oriented applications; so, developers are forced to build custom workflow solutions. Furthermore, the thesis claimed that traditional workflow architectures are based on the requirements and assumptions that do not hold in the context of contemporary object-oriented software development. This mismatch makes current workflow systems unsuitable for developers who need workflow within their applications. Zhuge et al. Reference [7] cited the fact that traditional workflow approaches are too rigid to adapt to the changes of domain business and are not useful in rapid development of virtual organisations. Once the domain business has changed, the system has to be re-designed. Chung et al. [1] stated that, while existing workflow management systems are widely used for the streamlined management of 'administrative' business processes, current systems are unable to cope with the more dynamic situations encountered in ad hoc and collaborative processes. Furthermore, the author's also stated that a major limitation of traditional workflow systems is that they can, typically, only support simple, predictable, but not the dynamically changing and complex processes that are present in many organisations. Ndeta and Marir [8, 9] confirmed through questionnaires and case studies the limitations of traditional framework when it comes to the development of e-workflow to support e-business processes.

This paper addresses the above limitation by enhancing traditional framework with a knowledge perspective that provides workflow modellers with processes and methods to dynamically capture explicit and tacit knowledge generated within or outside the boundary of the organisation. This captured knowledge will play a proactive role as a change manager monitoring and updating the other perspectives actions to reflect the dynamic changes of e-business processes and to increase the flexibility of workflow design and evolution.

The paper is organised as follows; section two is devoted to literature review and related work. Section three will present the proposed knowledge enhanced framework for e-workflow design, the workflow design pattern repository to support the proposed knowledge perspective and also the structure and access to the repository, and finally section four devoted to conclusion and future work.

II. RELATED RESEARCH WORK

Following the limitations of the traditional Workflow Management Systems (WFMS) as presented in the previous section, there have been numerous attempts to tackle these limitations. However, little of the published research work proposed to enhance the traditional framework instead most of the research works have focussed on the improvement of the workflow development within that traditional framework. Their approach focuses more on implementation issues and do not provide clear procedure on how to incorporate knowledge aspects in their e-workflow design.

Ndeta [9] integrated knowledge-based techniques with a workflow management approach to support processes change, trace decisions and to provide notification mechanism. This approach was applicable in civil and software engineering and focuses more on implementation issues and does not look at the other perspectives of workflow modelling e.g., informational and organisational perspective. Dellen et al. [10] provides process-oriented architecture for modelling inter-organisational workflow for e-commerce. It uses Petri-net formalism to model behavioural/dynamic aspects of inter-organisational workflows and the concept of soundness to verify the workflow model. The work failed to show how to model the other perspectives, which are essential during workflow specification and modelling i.e., organisational and information perspectives. The TBPM project in is based on a work carried out in the Enterprise project and centres on an intelligent workflow engine that includes an interactive process planner [11]. This approach focuses more on the functional perspective and implementation issues of workflow application development.

A number of research works use ontology as a tool for explanation and provides an algorithm to resolve the heterogeneity amongst the interfaces of web services and integrate it with workflow tasks. The approach focuses on the functional and information perspective of e-workflow modelling and fails to mention organisational and behavioural perspectives, which are essential if a complete picture of e-workflow modelling is to be exemplified.

Zhuge et al. [7] proposed a simulation-based development framework for establishing virtual organisations. The framework consists of a federation-agent-workflow (FAW) model, a set of rules for establishing the mapping from the domain organisation into the virtual organisation, a set of management services, and a macro development process. The framework unifies the traditional domain organisation and information system model into a virtual organisation model, and allows users to develop intuitive virtual organisations from the viewpoint of the domain. The virtual level is separated from the implementation level that consists of a runtime-support mechanism and a behaviour repository. The approach focuses on implementation issues in the form of a simulation framework. There are no clear procedure and step on how to model some of the workflow perspectives, i.e., functional and information perspectives. Chung et al.'s [1] their adaptive workflow approach uses the ontology and intelligent agents and knowledge based planning techniques to provide support for developing flexible workflow for the development of new product in the chemical industries, where the modelling focuses only on the functional perspective. Medeiros et al. [12] presented the on-going research on their scientific workflow framework called WOODSS. Conceived for supporting scientific work in environmental planning and distributed web-based applications for other scientific domains. The core of this work is based on creating repositories containing workflow specifications and providing mechanisms for accessing this repository for reuse. This work does not provide generic solution for e-workflow development by ignoring the different perspective of the framework and focussing on the implementation of the repository. Van der Aalst et al. [13] used workflow-mining techniques to create a feedback loop to adapt the workflow model to changing circumstances and detect imperfections of the workflow design and also find explicit representations for a broad range of process models. The approach focuses more on the functional, organisational, behavioural and case perspectives and on implementation or operational issues in the form of reverse engineering. There is no clear procedure and steps on how to develop a workflow application. It is also not clear how the informational perspective of workflow modelling is dealt with in this approach.

There are several issues that have driven the traditional workflow approach into obsolescence. Workflow for e-business processes should be characterised by anticipating surprises, self-control in management, creation and renewal of knowledge, unstructured organisation, intangible assets and proactive work style instead of prediction, compliance, and utilisation of knowledge, tangible assets, and structured organisation and reactive respectively..

III. KNOWLEDGE ENHANCED FRAMEWORK

Workflow modelling and design is well understood for traditional intra-organisational workflow applications, which span only one organisation with structured and predictable business processes and business environment. Traditional framework shown in “Fig. 1” for designing workflow is appropriate for modelling business processes of such organisation as the focus is on the internal work processes and information processes perspectives within the organisation. Thus, designing workflows in the traditional framework is limited to modelling four internal perspectives of the organisation: (i) decomposing organisation process functionality into task hierarchy that can be allocated to existing actors (human or software agents) of the organisation (functional perspective), (ii) describing the organisation information objects (business data, documents and product specs) which are consumed and produced within the organisation (informational perspective), (iii) specifying the roles and actors which are designated by the organisation management to be involved in the workflow execution and describes how the organisation configures its resources to perform business processes (organisational perspective), and (iv) specifying the way the information and the knowledge are channelled through the different steps of the process and the rules and constraints of the organisation business policies and practices (behavioural perspective). If traditional framework is successful in designing workflow to support traditional business processes, it is increasingly inadequate to model e-business processes as it requires not only modelling internal organisation perspectives but coping with unpredictability and continually changes brought by e-business environment [1][6][7][10].

Furthermore, a questionnaire put by the team in collaboration with Workflow Management Coalition found that around 70% of workflow developers have noticed the limitations of traditional framework when designing e- workflow systems for organisation e-business processes [9]. This confirms that a new framework for designing e-workflow systems became a necessity due to the impact of e-business processes on organisation survival and growth in this global and knowledge economy.

This paper presents a new knowledge enhanced framework for designing flexible e-workflow systems, as shown in “Fig. 2” below.

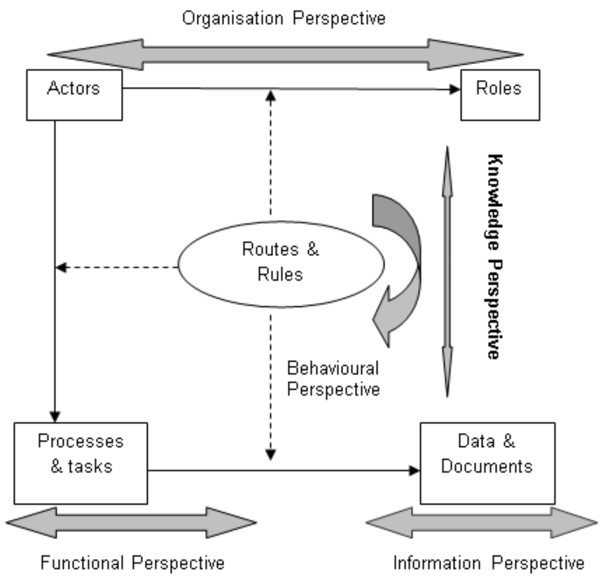


Figure 2. Knowledge Enhanced Framework for e-Workflow.

This new framework enhances traditional framework with a new knowledge perspective and provides new methods for developing e-workflow systems to support dynamic e-business processes. The proposed framework shown in “Fig. 2” borrows concepts and solution from both design and rule patterns to support flexibility during workflow modelling, design and evolution in the new e-business environment. Traditional approaches to workflow modelling mainly deal with the use of rules in specific contexts serving as a mechanism for workflow enactment and evolution [14]. The proposed framework focuses on providing abstraction mechanisms to represent in an implementation-independent manner the knowledge and experience associated with workflow design. The additional knowledge perspective is set to capture explicit and tacit knowledge generated within or outside the boundary organisation. The knowledge perspective will use the captured knowledge to trigger changes needed to both the organisation e-business processes and the four perspectives of the traditional framework to reflect and satisfy the new requirements of internal and external stakeholders of the organisation. Some examples of the role of this knowledge perspective could be extracting knowledge from data and information flowing between the other perspectives, memorising successful stories and best practices within or outside the boundary of the organisation or recording views of external stakeholders such as customer on the organisation e-business processes. The captured knowledge could be fed back into the organisation to be used to undertake changes to their e-business processes and improve their e-workflow systems to respond to the e-business new requirements.

To support the knowledge enhanced framework, the new knowledge perspective will be consolidated with an underlying knowledge repository, which maintains a history of process structure, relating each structure to the types of tasks for which it is a suitable method.

A. *Workflow Patterns-Based Knowledge Repository*

The knowledge repository is designed using the concept of design patterns successfully used in object-orientation and component software engineering to suggest well-proven design patterns [15] in order to implement solutions for typical recurring problems. Patterns are a way for the designer to specify the workflow schema (process definition) by reusing previous experience and knowledge of the best e-business designers. This work is contributing to research work in the direction of formalising patterns in the workflow domain where substantial effort in the form of workflow patterns [2], workflow data patterns and workflow resource patterns [16] have been investigated. The proposed workflow design pattern repository is one of the possible ways to assist knowledge workflow designers and users in building their workflow models efficiently. This approach will avoid them re-inventing already existing solutions of problems, which are common in the domain context, thus enhances the competitive edge of the organisation. In addition to memorising previous successful workflow design, the knowledge repository will also memorise feedback and views from external stakeholders and competitors' successful e-business processes to trigger changes required in the organization and update their e-workflow and e-business processes [9]. In the sub-sections below, we present the workflow design pattern scheme that composes the knowledge repository and how it is indexed and retrieved. [9].

B. *Workflow Design Pattern Scheme*

Each workflow design pattern specifies a set of tasks, together with the ordering constraints and object flows between them. Thus, a workflow design pattern represents one possible means of achieving a given type of task by breaking it down into a particular structure of sub-tasks. Each workflow design pattern specifies a single level of structural decomposition. However, the decomposition is decomposed into a further set of tasks, for each of which further workflow design patterns may exist in the repository. These workflow design patterns may in turn be selected to specialise/instantiate the sub-tasks, and so a multi-level hierarchical process structure may be generated by composing many design patterns.

For any given task, there may be multiple possible workflow design patterns, expressing different ways of breaking the task down for different situations. In this research we focus on the patterns in business problem domain, where control flow and data flow interact. We select UML as an implementation language as it allows modelling of the various perspectives, i.e., process, organisation, behaviour and data. A workflow schema shown in "Fig. 3" depicts a simple workflow definition (schema) for the business process of booking a travel service from a network centric travel service provider.

The workflow schema has three other sub-workflow schemas relating to partner (associate) business service providers (BSP), airline reservation, hotel reservation and car rental reservation. Three additional sub-workflow design patterns are required, each for achieving the airline, hotel and car rental reservation sub-workflows task.

C. *Workflow Design Patterns Indexing and Retrieval*

Two main interconnected indexing schemes are provided in the workflow design pattern repository; workflow design pattern indexing scheme and the sub-workflow design pattern indexing schemes as shown in "Fig. 4".

Each of the indexing schemes is composed of two types of indexes with different functionalities. This is to reflect general and local context-sensitive of the workflow design patterns and its sub workflow design patterns. Classification indexes represent the global and local context features of workflow design patterns and their sub-workflow design patterns that represent the acquired knowledge and experience in the workflow domain. These indices are considered as difference-based indexing scheme by their main function of differentiating a workflow design pattern from another similar workflow design pattern. However, these indexes are mainly used to classify and direct the retrieval to context-sensitive workflow design patterns and sub-workflow design patterns. This reflects the importance given to the context information which domain expert use to retrieve and adapt patterns. It also add an advantage to the proposed e-workflow approach by reducing the scope of the retrieval search space of classes of similar workflow design patterns rather than the whole workflow design pattern repository

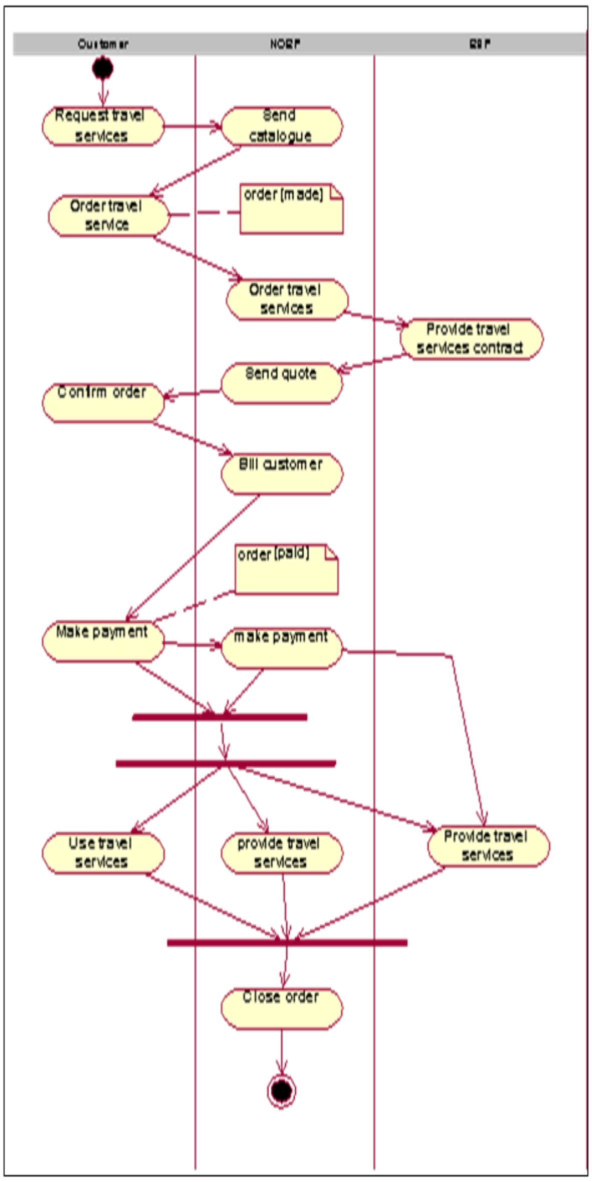


Figure 3. Travel Booking service workflow definition

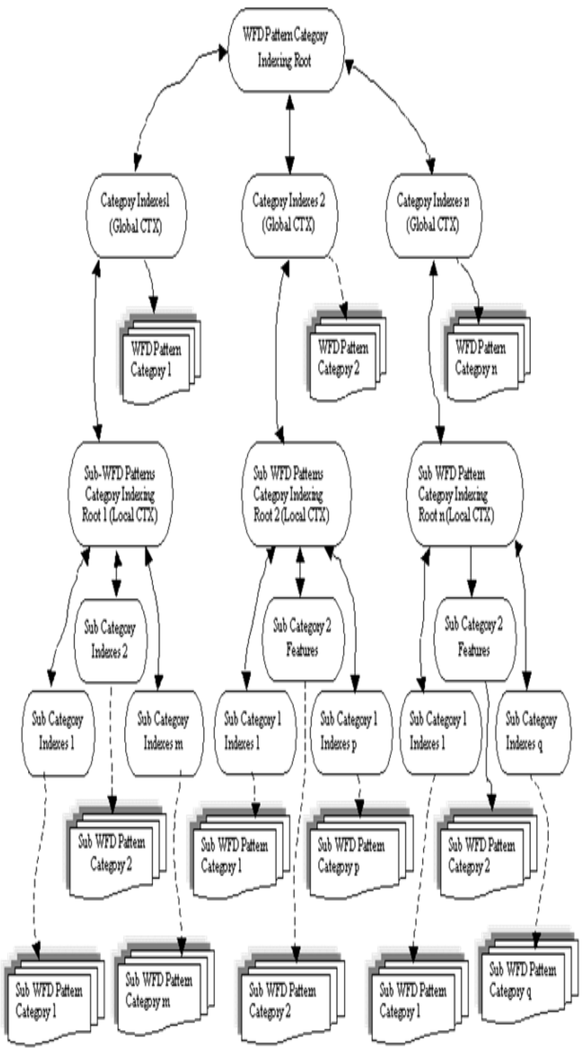

WFD Pattern Category Indexing Root
Category Indexes1 (Global CTX)
Category Indexes 2 (Global CTX)
Category Indexes n (Global CTX)
WFD Pattern Category 1
WFD Pattern Category 2
WFD Pattern Category n
Sub-WFD Patterns Category Indexing Root 1 (Local CTX)
Sub WFD Patterns Category Indexing Root 2 (Local CTX)
Sub WFD Pattern Category Indexing Root n (Local CTX)
Sub Category Indexes 2
Sub Category 2 Features
Sub Category 2 Features
Sub Category Indexes 1
Sub Category Indexes m
Sub Category 1 Indexes 1
Sub Category 1 Indexes p
Sub Category 1 Indexes 1
Sub Category Indexes q
Sub WFD Pattern Category 2
Sub WFD Pattern Category 1
Sub WFD Pattern Category p
Sub WFD Pattern Category 2
Sub WFD Pattern Category 1
Sub WFD Pattern Category m
Sub WFD Pattern Category 2
Sub WFD Pattern Category 1
Sub WFD Pattern Category q

Figure 4. Indexing schema for Workflow design patterns.

.

IV. EVALUATION OF THE PROPOSED FRAMEWORK

The proposed framework for designing e-workflow from scratch requires less work than the traditional approaches do because of the process of reuse provided by the knowledge repository. Another advantage is that previous workflow cases can be adapted easily to new domain of business. Compared to traditional workflow approaches our proposed approach provides a consistent development process, so it meets the needs of virtual organisations for rapid, low cost and flexible development of workflow systems. The approach also takes into consideration all the perspectives involved in the design and modelling of e-workflow applications [17].

The traditional workflow systems evolving around the traditional framework principles have failed to provide support for knowledge workers in order to assist them in performing their daily tasks. This often causes delays in the work process when an exception occurs due to the fact that the workflow participant is not empowered with context-specific knowledge about the task within the work process [13]. This also causes delays for workflow designers during workflow design and evolution in the new e-business environment.

For instance, in traditional framework, the functional perspective decomposes process functionality into a task hierarchy that can be allocated to e-actors (human or software agents) on the Internet. These functions are difficult to adapt when there is a change in the business process or business environment. This is largely due to the fact that the workflow designers or users are not provided with sufficient business processes related knowledge that is necessary to facilitate workflow process adaptation and maintenance if there is a change in the business environment. However, in the proposed framework, the knowledge repository that underlines the knowledge perspective is there to provide relevant information and knowledge for the workflow designers to adapt the task at hand.

The majority of conventional workflow systems have been designed to enable information management activities that target the capture of data and information as opposed to enabling knowledge management for harvesting knowledge itself [13]. Thus traditional workflow development frameworks and approaches focuses on information processing (or transaction led) rather than the knowledge creation and innovation which is vital for survival and growth in the new world of e-business and the emerging global and knowledge economy.

In the traditional workflow framework and development approaches, external roles are not well defined as the system fails to provide process related knowledge and knowledge flow for all the parties involved. If there is a breakdown during the execution of a workflow process, it is sometimes difficult to know who is responsible for the breakdown since the system fails to provide efficient flow of process related knowledge between the stakeholders involved in the execution of the e-workflow process. While in the traditional framework modelling the behavioural perspective was based on a high level of structure and control reflecting the internal organisation policy, the dynamics of the new e-business environment in which more than one organisation are sharing that e-business process require a different approach to organisational and workflow design. The knowledge perspective provided in the proposed framework allows e-workflow designer or users to develop flexible rules agreed between organisations who share the e-business processes.

The proposed framework has been tried successfully by a travel agent to assess their e-business processes. One of the e-business processes assessed was Flight + Hotel booking web based service. In the initial flight + hotel booking service, the customer while booking the flight is not given the possibility to alter the number of days required to spend in the hotel. We changed the Flight + Hotel booking service by providing on the web a facility for the user to put their comment why they did not proceed with the booking. When we analysed the comments of the customers we found that a large number of customers would like to book the flight with fewer days to spend in the hotel as they would like to spend some of the days with local friend or families within or outside the city deserved by the flight. Once, we changed the Flight + Hotel e-business allowing customers the flexibility to book Flight + Hotel with the possibility to alter the number of days to spend in the hotel, no customer has raised again that problem which might means that more customers have booked for Flight + Hotel service.

V. CONCLUSION AND FUTURE WORK

In this paper, we have presented a new framework that takes into consideration the knowledge dimension for the development of e-workflow to support continual and unpredictable changes of current e-business processes. A significant contribution of our proposed framework is not only limited to adding a knowledge perspective to traditional framework but by setting new relationship and dynamics between this knowledge perspective and the existing functional, organisational, behavioural and informational perspectives. In the proposed knowledge enhanced framework, all perspectives are integrated and all of them harbour some knowledge through common access to a knowledge repository that memorise best practices, previous experiences and stakeholders experiences. Furthermore, the paper has presented one realisation of the knowledge repository in the form workflow design pattern repository. We have also provided an input format for pattern creation, which would enable patterns content in the workflow design pattern repository to be structured and predictable. The paper has also provided an appropriate hierarchical structured indexing scheme for the repository to be able to retrieve most similar workflow and sub workflow design patterns. Future research work will focus on the architecture of the e-workflow design pattern providing mechanisms for accessing and managing the workflow design pattern repository and the use of exceptions as a basis for managing dynamic workflow evolution.

REFERENCES

[1] Chung, L.P.W., Cheung, H., Stader, J., Jarvis, P., and Moore. J., Macintosh, A. “Knowledge-based process management - an approach to handling adaptive workflow”, KBS, 2003, 16, pp. 149-160

[2] Van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B. and Barros, A.P. (2003) “Workflow Patterns”. Distributed and Parallel Databases, 14(1):5-51, 2003.

[3] Fahey, L., Srivastava, R., Sharon, J.S. and Smith D.E. (2001) “Linking e-business and operating processes: the role of Knowledge management”, IBM Syst. J. 40 (4) pp. 889-906

[4] Madhusudan, T., Zhao, J.L. and Marshall, B. (2004) “A case-based reasoning framework for workflow model management”. Data & Know. Engineering, 50, pp. 87-115

[5] R-Moreno, D.M., Borrajo, D., Cesta, A. and Oddi, A. “Integrating planning and scheduling in workflow domains”. Expert Systems with Applications, 2007, 33, pp. 389-406.

[6] Manolescu, D. “Micro-workflow: A Workflow Architecture Supporting Compositional Object- oriented Software Development. Ph.D. thesis, Graduate College of the University of Illinois at Urbana-Champaign. Urbana, Illinois, 2001.

[7] Zhuge, H., Chen, J., Feng, Y., and Shi, X. “A federation-agent-workflow simulation framework for virtual organisation development”, Information & Management, 2002, 39, pp. 325-336.

[8] Ndeta, J., Marir, F., and Choudhury, I. ”Knowledge enhanced e-workflow modelling – a pattern-based approach for the development of Internet workflow systems”, Proceedings of the 7th European Conference on Knowledge Management, 2006, Budapest, Hungary.

[9] Ndeta, J. “Knowledge Enhanced Framework for the Design and Development of e- workflow Systems”, PhD Thesis, London Metropolitan University, 2008, UK.

[10] Dellen, B., Maurer, F., and Pews, G. “Knowledge-based techniques to increase the flexibility of workflow management”, Data & Know. Eng., 1997, 23, pp. 269-295.

[11] Van der Aalst. W.M.P. “Process-oriented Architecture for Electronic Commerce and Interorganisational Workflow”, Information Systems 1999, Vol. 24, No. 8, pp. 639-671

[12] Medeiros, C. B., Perez-Alcazar, J., Digiampietri, L., Patorello, Jr. G. Z, Santanche, A., Torres, R. S., Madeira, E. and Bacarin, E. “WOODSS and the Web: annotating & reusing scientific workflows”. 2005, SIGMOD Record, Vol. 34, No. 3, pp. 18-23.

[13] Van der Aalst, W.M.P., Reijers, H.A., Weijters, A.J.M.M., Van Dongen, B.F., Alves de Medeiros, A.K., Song, M. and Verbeek, H.M.W. “Business process mining: An industrial application”, Inf. Systems, 2007, 32, pp. 713 – 732

[14] Casati, F., Fugini, M. and Mirbel, I. “An Environment for designing Exceptions in Workflows”, Information Systems, 1999, Vol. 24, No.3, pp. 255-273

[15] Gamma, E., Helm, R., Johnson, R and Vlissidesn, J., “Design Patterns: Elements od Reusable Object-Oriented Software”, 1995, Addison-Wesley Professional Computing Series. Addison-Wesley Publishing Company, New York.

[16] Russell, N., ter Hofstede, A.H.M., Edmond, D. and Van der Aalst, W.M.P. “Workflow Data Patterns”. QUT Technical report, 2004, FIT-TR-2004-01, Queensland University of Technology, Brisbane.

[17] Ndeta, J., and Marir, F. “Towards the Development of a Conceptual Framework fro Knowledge Enhanced e-Workflow Modelling”, Proc. 6th European Conference on KM, University of Limerick, Ireland, 2005.

Modeling and Simulation Versions of Business Process using Petri Nets

Fatma Ellouze, Mohamed Amine Chaâbane, Rafik Bouaziz
University of Sfax/ Faculty of Economics and Management of Sfax
Route de l'aéroport, BP1088 Sfax, Tunisia
fatma_ellouze@hotmail.com; {Ma.chaabane, Raf.bouaziz}@fsegs.rnu.tn

Eric Andonoff
University of Toulouse/ IRIT
2 Rue de Doyen Gabriel Marty, 31042 Toulouse Cedex, France
andonoff@univ-tlse1.fr

Abstract — This paper proposes a solution for modeling and simulating flexible Business Processes (BP) using version concepts. It advocates a Model Driven Architecture approach to handle versions of BP. The paper presents, first, a meta model (VBP2M: Versioned Business Process Meta model) for modeling versions of BP considering six perspectives of business processes: process, functional, operation, informational, organizational and intentional perspectives. Then, it proposes an extension of the meta model of Petri nets (PN) to support the version concepts. Finally, it defines mapping rules to translate versions of BP modeled in accordance with the VBP2M to PN models, with graphical representations, in order to be able to simulate the behavior of each version of BP.

Keywords-Flexible Business Process; Version; VBP2M; Petri nets; Mapping rules; MDA framework.

I. INTRODUCTION

Nowadays, the importance of Business Processes (BP) in organizations' Information Systems (IS) is widely recognized and these last few years, there has been a shift from data-aware IS to process-aware IS [1] [2] [3]. However, despite important advances in Business Process Management (BPM), several issues are still to be addressed. Among them, the business process flexibility issue is a really a relevant and challenging one. Indeed, the competitive and dynamic environment in which organizations and enterprises evolve leads them to frequently change their business processes in order to meet new production or customer requirements, new legislation or business rules. We define flexibility of BP as *the ability of BP to deal with both foreseen and unforeseen changes in the environment in which they operate* [4].

This issue has mainly been addressed using two main approaches: a declarative (decision oriented) approach and a procedural (activity oriented) approach. The declarative approach consists in defining a set of constraints defining BP execution rules [5] [6] [7] [8]. In this approach, dealing with flexible BP seems very easy as we just need to add and/or remove constraints to define new BP execution rules. However this approach is only available in the Declare BPM suite [13], which is an academic solution unknown in the industry. At the opposite, the procedural approach is widely used in industrial Workflow management systems and BPM suites. Thus BPM community has to deal with procedural BP flexibility. In this approach, a BP is defined as a set of activities coordinated by using control patterns [9]. Several techniques have been introduced to address BP flexibility and among them we quote late binding, late modeling and versioning [10] [11].

In a previous work [12], we advocated to use versioning for BP flexibility. Indeed, using this technique, it is possible to deal with the different flexibility types defined in [11] as it is possible to handle, at the same time, different schemas (versions) of a given BP.

Our proposition [12] extends the three main contributions about BP versions ([16] [17] [18]) considering, in addition to the process and functional perspectives of BP. four other perspectives aiming to have a comprehensive description of BP [9] [10]. These perspectives are: (i) the operation perspective which defines actions to be achieved within an atomic activity, (ii) the informational perspective which describes the structure of information consumed and/or produced by the BP, (iii) the organizational perspective which details roles, organizational units and actors invoked by the BP and (iv) the intentional perspective which explains the context of use of a BP.

To sum up, our previous works [19][20] deal with BP flexibility issue adopting the procedural paradigm and using the versioning technique. It introduced VBP2M (Versioned Business process Meta model) for BP version modeling. However, we did not addressed the simulation and verification of the modeled BP versions in order to check their behavior. As a consequence, the aim of this paper is to simulate and verify the behavioral dimension of the modeled versions of BP using conventional Petri nets. Conventional Petri nets have been chosen since they are recognized as a perfect mean for simulating and verifying distributed applications and they are widely used in BPM [14] [15]. More precisely, this paper:

- advocates an MDA approach for dealing with BP simulation;
- extends PN meta model to integrate BP version concepts;
- defines a transformation process including (i) the mapping between concepts of VBP2M and extended PN meta model and (ii) translation related rules;
- presents a tool implementing this transformation.

The remainder of this paper is organized as follows. Section 2 shows how we model versions of BP. Section 3 introduces the MDA framework we propose to handle versions of BP. Firstly, this section details the three models (CIM, PIM and PSM) of the framework. Secondly, it gives mapping rules allowing to represent a version of BP, modeled according the previous meta model, as a tree. Thirdly, it explains mapping rules to translate a version of BP represented as a tree to a Petri Net model. Section 4 illustrates our proposals within a case study. Section 5 presents the tool implementing our contributions. Finally, Section 6 recaps our contributions, discusses them according related work, and gives some perspectives for our future works.

II. MODELING VERSIONS OF BUSINESS PROCESS

This section briefly presents the Versioning Business Process Meta model (VBP2M) we have proposed to model versions of business process [12][19][20]. More precisely, first it introduces the version concept and then it details the VBP2M for versions of BPs.

A. *The version concept*

A real world entity characteristic that may evolve during its life cycle: it has different successive states. A version corresponds to one of the significant entity states. So, it is possible. Hence, it is possible to manage several entity states. The entity versions are linked by derivation link; they form a derivation hierarchy. When created, an entity is described by only one version. The definition of every new entity version is done by derivation from a previous one. Such versions are called derived versions. Several versions may be derived from the same previous one. They are called alternative version. Fig. 1 illustrates a derivation hierarchy to describe entity evolution.

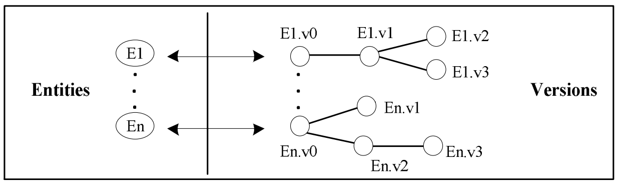


Figure 1. Derivation hierarchy

A version is either frozen or working. A frozen version describes a significant and final state of an entity. A frozen version may be deleted but not updated. To describe a new state of this entity, we have to derive a new version (from the frozen one). A working version is a version that describes one of the entity states. It may be deleted or updated to describe a next entity state. The previous state is lost to the benefit of the next one.

B. *VBP2M: A meta model for versions of BPs*

The VBP2M is result from merging of two layers; a BP meta model for classical BP (which not evolve on time) modeling, and versioning pattern to make some classes of the BP meta model versionable (i.e., classes for which we would like to handle versions). Because of space limitation, in this we focus on the VBP2M only. Intersected reader can consult our previous work [19] [20] to have additional information about these two layers and the way we merge them to obtain the VBP2M.

Fig. 2 below present the VBP2M in terms of classes and relationships between classes. This figure visualizes in gray versionable classes (i.e., classes for which we handle versions), and non-versionable classes (i.e., classes for which we do not handle versions). The VBP2M considers the six perspectives (Functional, operation, process, informational, organizational, and intentional perspectives).

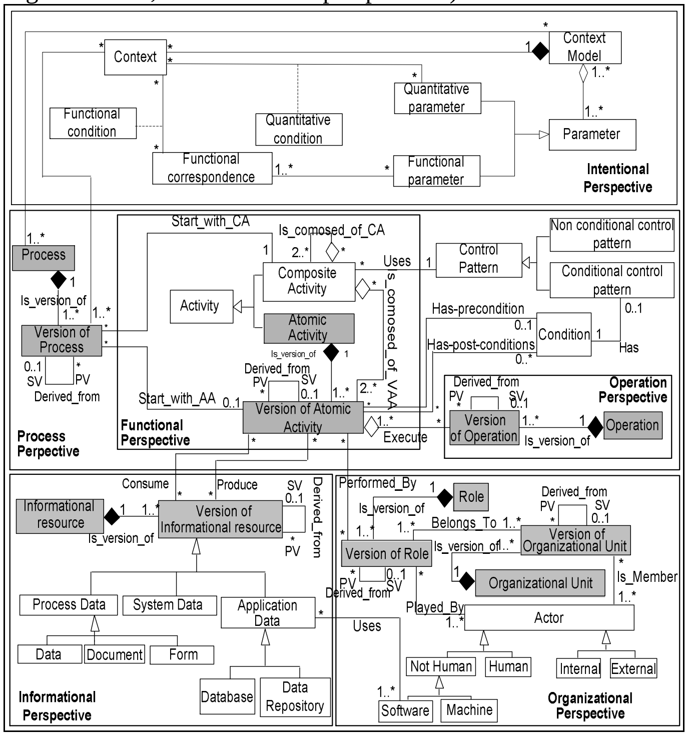


Figure 2. Versioning Business Process Meta model

1) *Main concepts of VBP2M*: The main concepts of the VBP2M are Process, Activity, Control Pattern, Operation, Informational resource, Role and Context concepts. A process performs activities, which are atomic or composite. Only the first of these activities is explicitly indicated in the meta model. At the composite activity, we keep its component activities, which are coordinated by control patterns. In our meta model, the main control patterns described in the literature are provided. Some of them are conditional (e.g., if, while, etc.), while others are not (e.g., sequence, etc.). An atomic activity can have precondition (or start condition), post-condition (or end condition) and execute one or several operations. It is performed by role, which can play by several actors belonging to organizational units (organizational perspective). Moreover, an atomic activity consumes and/or produces informational resources (informational perspective). A use context is associated for each version of process.

2) *Taking into account versions:* The underlying idea of our proposition to take into account versions of BP is to describe, for each versionable class, both entities and their

corresponding versions as indicated in “Fig. 1”. As consequence, each versionable class is described using two classes: the first class is called “…”, to model entities and a second one, called “version of …”, whose instances are versions. For instance, versions of processes are modeled within two classes: the Process class contains all modeled BP while the Version of process contains versions of the modeled BP. These classes are linked together by two relationships: the “is_version_of” relationship links a versionable class with its corresponding “Version of…” class and the “Derived_from” relationship describes version derivation hierarchies between versions of a same entity. This latter relationship is reflexive and the semantic of both sides of this relationship are: (i) a version (SV) succeeds another one in the derivation hierarchy and, (ii) a version (PV) precedes another one in the derivation hierarchy. Moreover, we introduce in the “Version of…” classes, classical properties for versions i.e., version number, creator name, creation date and state [21].

3) Versionable class: Finally, it is possible to manage versions both at the schema and the instance levels. In the Business Process context, it is only interesting to consider versions at the schema level (i.e., versions of BP schemas), and the notion of version must be applied to all the perspectives defined at the schema level. In our proposition, and unlike related work (e.g., [16] [17] [18]), which consider only two perspectives (functional and process perspectives), we take into account the five main perspectives of BPs, i.e., the process, functional, operational, organizational and informational perspectives, which are considered as relevant for BP modeling and execution [9] [10]. More precisely, regarding the process and functional perspectives, we think that it is necessary to keep versions for only two classes: the Process and the Atomic activity classes. It is indeed interesting to keep changes history for both processes and atomic activities since these changes correspond to changes in the way that business is carried out. More precisely, at the process level, versions are useful to describe the possible strategies for organizing activities while, at the activity level, versions of atomic activities describe evolution in activity execution. We defend the idea that versioning of processes and atomic activities is enough to help organizations to face the fast changing environment in which they are involved nowadays. Regarding the other perspectives, it is necessary to handle versions for the Operation class of the operational perspective, for the Informational resource class of the informational perspective, and for the Role and Organizational Unit classes of the organizational perspective.

III. MDA FRAMEWORK TO HANDEL VERSIONS OF BUSINESS PROCESSES

After modeling, verification of version of business process (VBP) can be done by (i) semantic verification and (ii) behavioral verification. The aim of the semantic verification is to verify the presence of VBP’s activities, their coordination, the invoked roles and the used informational resources. This verification can be ensured by the graphical languages and notations (i.e., BPMN, Yawl.) Regarding the behavioral verification, we verify some behavioral properties such as the liveness (i.e., the absence of global or local deadlock situation), the consistency (i.e., the existence of cyclic behavior for some marking), etc. This verification can be done with languages which have a simulator as Petri Nets. In our previous work [4], we interested by the semantic verification using BPMN. More precisely, we have generated a BPMN specification from a VBP obtained by instantiation of the VBP2M. Using this specification, we can visualize a version of a BP model in order to approve it. In this paper we deal with the behavioral verification issue. Especially, we propose mapping rules to translate automatically a VBP modeled according to VBP2M to a Petri nets specification. Then we use the Platform Independent Petri net Editor 2 "PIPE2" (which is an open source tool that contains a simulator and analyzer. It conforms to Petri Net Markup Language "PNML") to verify the behavior of the modeled VBP.

This section is organized as follows: first, we detail a MDA framework we propose to consider VBPs from modeling to execution. Second, we propose mapping rules allowing the translation from a VBP to VBP-tree. Finally, we define mapping rules to generate a Petri-net from the VBP-Tree.

A. *MDA framework*

An MDA (Model Driven Architecture) [22] framework specifies three levels of models: (i) the CIM (Computation Independent Model) refers to a business or domain model, (ii) the PIM (Platform Independent Model) is an independent model from all execution platforms and (iii) the PSM (Platform Specific Model) gives a textual description which can be used by execution platforms. To automate the transition from modeled VBP to the execution step, we propose an MDA framework where the CIM model contains the VBP2M and its instances (modeled VBP). The PIM model contains the specification of these VBP using a graphical specification language. At PIM model user can choose a language from (BPMN, Yawl, PN, etc.) to visualize in order to approve the modeled VBP. Regarding the PSM model an execution description (i.e., XPDL, BPEL.) is generated. This description belongs to specific platform (i.e Bonita, PETALS). This paper proposes mapping rules to translate from CIM model to PIM model. This transformation is operated according two steps: (i) consist of the querying the VBP2M in order to obtain the components elements of a VBP. These elements are organized in a tree named VBP-Tree (Versionned Business Process Tree). This step is common for all the graphical languages from PIM model (ii) allows the generation of graphical representation according to a chosen specification language using mapping rules. Fig. 3 shows the propose framework.

VBP-Tree is used to optimize the execution and the development time as the common operations will be done only once with all specification languages. For example, if we have a graphical representation of a VBP with BPMN then we want to visualize this same VBP with PN, so we just reuse the VBP-tree and just make mapping rules from VBP-Tree to BPMN.

VBP-Tree contains two types of nodes:

- Terminal node (leaves): represented by ellipses and correspond to versions of atomic activities of a VBP.
- Non terminal node: represented by rectangles and correspond to composite activities.

Fig. 4 below presents the meta model of VBP-Tree in terms of classes and relationships.

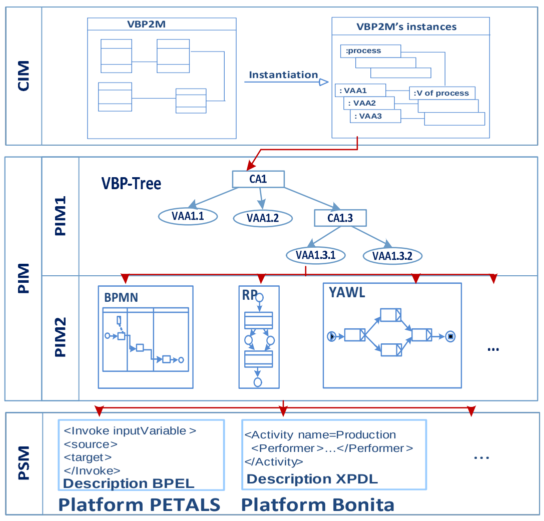


Figure 3. Framework MDA used

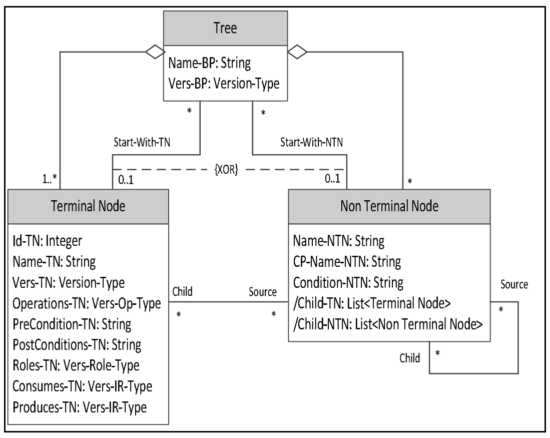


Figure 4. VBP-Tree meta model

Table I describes properties of a terminal node.

TABLE I. PROPERTIES OF TERMINAL NODE

Properties	Description
Id-TN	ID of the Terminal Node
Name-TN	Name of the Terminal node
Vers-TN	Properties of version of the atomic activity that the Terminal Node represents, it refers to another class type that contains the following properties number, Creator name, Creation Date and state of the version
Operations-TN	List of operations that are executed by the version of atomic activity that the Terminal Node represents
PreCondition-TN	Condition that must be evaluated to true to make the execution of the version of atomic activity that the Terminal Node represents
PostConditions-TN	Conditions associated to the version of atomic activity, after the execution of operations of the Terminal Node
Roles-TN	List of role able to execute the version of atomic activity that the Terminal Node represents
Consumes-TN	List of informational resources required to execute operations of the version of atomic activity that the Terminal Node represents
Produces-TN	List of informational resources produced after executing operations of the version of atomic activity that the Terminal Node represents

Properties of a non terminal node are described in Table II:

TABLE II. PROPERTIES OF NON TERMINAL NODE

Properties	Description
Name-NTN	Name of the Non Terminal Node
CP-Name-NTN	Name of the control pattern used for the composite activity that the Non Terminal Node represents
Condition-NTN	Optional property associated to conditional control patterns
Child-TN	List of Terminal Node that compose the Non Terminal Node
Child-NTN	List of Non Terminal Node that compose the Non Terminal Node

In the remainder, we represent firstly mapping rules from VBP2M to VBP-Tree. Secondly, we generate a PN with mapping rules from VBP-Tree to PN.

B. *Mapping rules from VBP2M to VBP-Tree*

To translate from a VBP to a VBP-Tree, we propose three mapping rules detailed in Table III below.

TABLE III. MAPPING RULES FROM VBP2M TO VBP-TREE

N°	VBP2M concepts	VBP-Tree concepts
1	Version of Business Process	Tree
2	Version of Atomic Activity	Terminal Node
3	Composite Activity	Non Terminal Node

The function implementing the mapping from a VBP to VBP-Tree (cf. Fig. 5) use a set of functions permitting the handling version of processes and nodes

- StartWithVAA (VP): indicates if the VP start with Version of Atomic Activity class;
- BuildTN (A, Tree): build a Terminal Node with properties of the Version of Atomic Activity (A) and its relations (performed-by, consumes, etc.), then add it in the Tree;

- BuildNTN (A, Tree): build a Non Terminal Node with properties of the composite Activity and its relations (uses, etc.), then add it in the Tree;
- Children (Node): return all the children of the composite activity (Node).

```
Function BuildVBP-Tree (A: Activity): VBP-Tree
Local n: Node
Global Tree: VBP-Tree
Begin
  If StartWithVAA (A)
     n: BuildTN (A, Tree)
  Else
     n: BuildNTN (A, Tree)
     For each child in Children (A)
         BuildVBP-Tree = BuildVBP-Tree (child)
     Next child
  End If
End.
```

Figure 5. Function BuildVBP-Tree

C. *Mapping rules from VBP-Tree to PN*

Firstly, we explain more what is PN? In fact, Petri Net is a formal tool that is composed of:

- A set of places (P1, P2, ..., Pn) which represents triggering conditions;
- A set of transitions (T1, T2, ..., Tn) which represent activities;
- A set of oriented arcs. Two types of arcs are distinguished: input arcs which link place to transition and output arcs which link transition to place. For the input arcs, we distinguish also two types which are normal (arc with an arrow at the end, that can contains a token) and inhibitor (arc with a small circle at the end, that cannot contains a token) arcs.

The meta model of PN is shown in the UML diagram of Fig. 6.

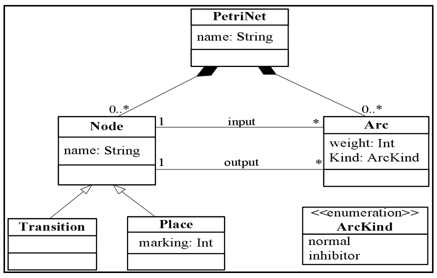


Figure 6. Petri net meta model

In order to support concepts of the VBP2M, we propose to extend Petri net meta model by adding five classes:

- Version class: that contains version number, creator name, creation date and state attributes. This class linked by the Node class and the Petri Net class in order to specify their versions;
- Operation class: contains an attribute which specify its name and has a relationship named "op-vers" that is linked to Version class;
- Informational resource class: specify the type of the Place class. It contains two attributes: type and nature of the resource;
- Role class: specify the type of the Place class;
- Organizational unit class: specify the type of the Place class;

Besides these new classes, we add also two attributes in the Transition class that concerns the precondition and the post conditions and one relationship named "has-vers-op" linked to Version class. The aim of this extension is to increase the semantic dimension by specifying each node with its version and non version information. Fig. 7 shows the extended meta model of PN. These new classes are represented with gray color.

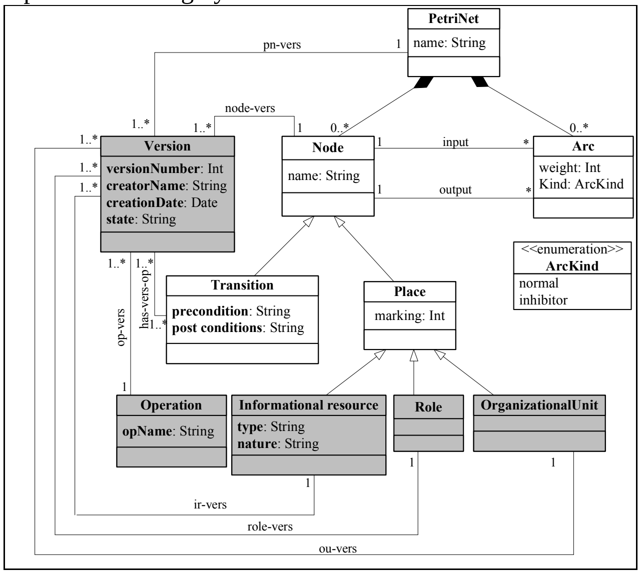


Figure 7. Petri net meta model extended

To generate a PN from VBP-Tree we propose mapping rules (cf. table IV) between VBP-Tree properties and the extended PN properties. After applying these rules, it becomes possible to verify the behavioral of the obtained PN. This verification ensured using the simulator of PIPE2.

TABLE IV. MAPPING RULES FROM VBP-TREE TO PN

VBP-Tree concepts	**VBP-Tree properties**	**PN properties**
Terminal Node	Name-TN	Name of a transition
	Name-Role of the attribute Roles-TN	Name of an input Role or Organizational unit place drawn with an inhibitor arc
	Name-IR of the attribute Consumes-TN, when Type-IR is “internal” or “external”	Name of an input Informational resource place drawn with a normal arc
	Name-IR of the attribute Consumes-TN, when Type-IR is “position”	Name of an input Informational resource place drawn with an inhibitor arc
	Name-IR of the attribute	Name of an output

	Produces-TN	Informational resource place drawn with a normal arc
Tree	Name-BP and Vers-BP	Name of the Petri net

We use the inhibitor arc for resources (Role and/or Information) that are not consumed by an atomic activity. These mapping rules are implemented by the function "Build PN" detailed in Fig. 8:

```
Function BuildPN (n: Node): PN
Local child: Node
Begin
  If IsTN(n)
    t: BuildTransitionPlace (n)
  Else
    Case pattern (n)
      When sequence: t: BuildTranPlacSeq(n)
      When parallel: t:BuildTranPlacPar(n)
      When …
    End case
    For each child in children(n)
      BuildPN(child)
    Next child
  End If
End.
```

Figure 8. Function BuildPN

This function uses a set of functions:

- IsTN(n): indicates if a node n is a terminal node;
- BuildTransitionPlace (n): build and add the corresponding transition and places of the terminal node n;
- *Pattern* (n): return the used pattern if n is a non terminal node;
- *BuildTranPlacSeq*(n) : build and add the transition and places of the terminal node n into the PN according to a sequence control pattern;
- *BuildTranPlacPar* (n): build and add the transition and places of the terminal node n into the PN according to a parallel control pattern.

Because of space limitation, we do not specify other control pattern such choice, iteration, etc.

IV. CASE STUDY

In order to illustrate our approach, we propose the process of the participation in a business tender named BTP .

The first version of this process, represented in Fig. 9(a), contains three activities (Acquisition of tender specifications, Preparation of the offer, Submission of the offer).

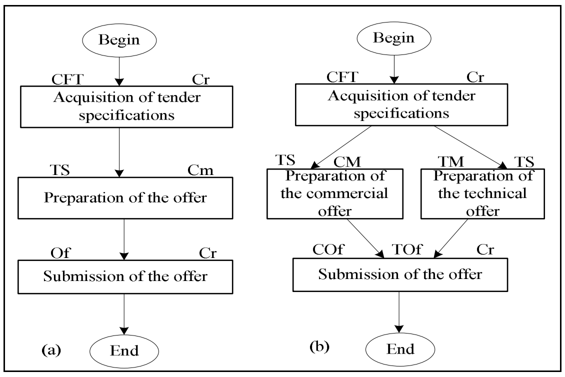

Begin
CFT
Cr
Acquisition of tender specifications
TS
Cm
Preparation of the offer
Of
Cr
Submission of the offer
(a)
End
Begin
CFT
Cr
Acquisition of tender specifications
TS
CM
TM
TS
Preparation of the commercial offer
Preparation of the technical offer
COf
TOf
Cr
Submission of the offer
(b)
End

Figure 9. Versions of process

- "*Acquisition of tender specifications*" which is triggered by the presence of a call for tender (CFT) and produces a tender specifications (TS). This activity is achieved by a courser (Cr).
- "*Preparation of the offer*" which is triggered by the availability of specifications tender and produces an offer (of). This activity is done by a committee of offer preparation (Cm).
- "*Submission of the offer*" which is triggered by the prepared offer and produces a coupon. This activity is realized by the courser.

Fig. 10, gives an extract of an instantiation of the VBP2M according to the first version of BTP process.

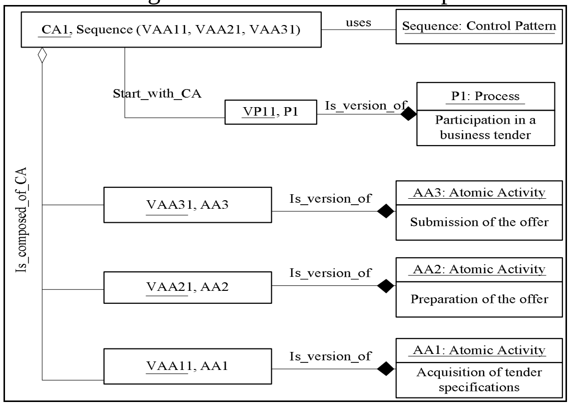


Figure 10. Instantiation of the VBP2M for the first version of BTP process

Fig. 11 presents the VBP-Tree of this version with a simplified view (nodes are not described in details).

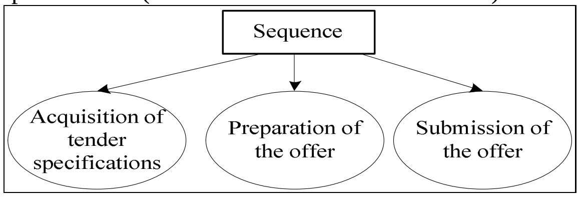


Figure 11. VBP-Tree for the first version of the BTP process

In the second version of this process, represented in Figure 9(b), the second activity "*Preparation of the offer*" will be divided: (i) "*Preparation of the technical offer*" activity which is realized by a technical manager (TM) (ii)

"*Preparation of the commercial offer*" activity which is realized by a commercial manager (CM). These two activities are done in parallel and executed respectively by a *Technical service* and *Commercial service*. The VBP-Tree of this version is illustrated in Fig. 12

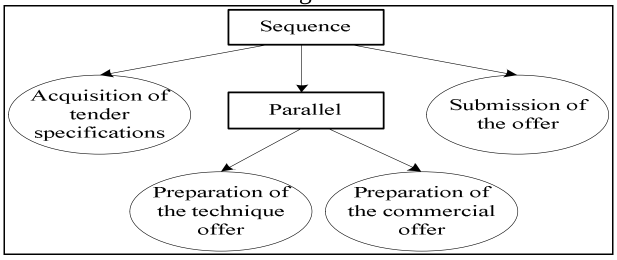

Sequence
Acquisition of tender specifications
Parallel
Submission of the offer
Preparation of the technique offer
Preparation of the commercial offer

Figure 12. VBP-Tree for the second version of BTP process

V. Implementation

We use PIPE2 (Platform Independent Petri net Editor 2) [23] [24] tool to implement our propositions.

Two steps are considered to implement our propositions: (i) extend the Petri net tool "PIPE2" in order to augment the existing dialogues that open when you currently click on the Place or Transition objects to display proprieties relative to our context and (ii) implement the proposed mapping rules.

In fact, we add new package "vffs" that contain:

- Four forms: One associate to the Transition class and the three others associate to the Place class (Role Place, Informational resource Place and Organizational unit Place);
- A class: contains methods that fill these forms.

We modify also:

- The "PlaceHandler.java" and "TransitionHandler.java" classes in the package "pipe.gui", especially the method of "mouseclicked";
- The "Place.java" and "Transition.java" in the package "pipe.dataLayer", especially the method of "showEditor".

After extending PIPE2, we implement mapping rules of detailed in sections III.B and III.C. Fig. 13 shows further the steps to generate automatically PN from the VBP2M, passing through the VBP-Tree.

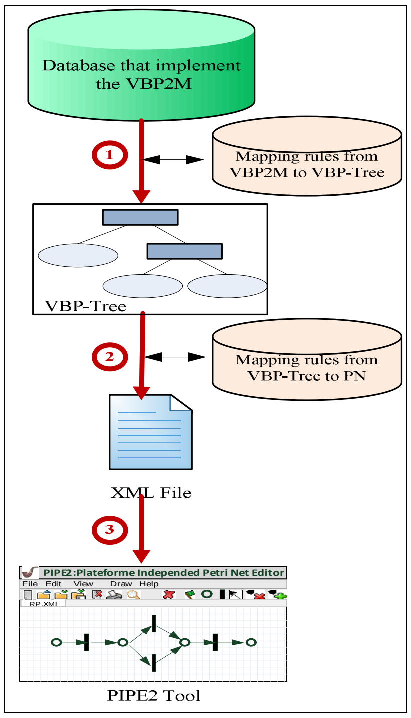
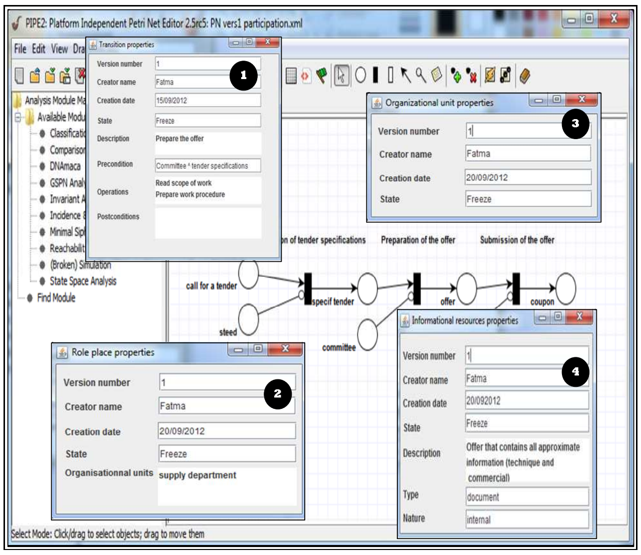

Database that implement the VBP2M
1
Mapping rules from VBP2M to VBP-Tree
VBP-Tree
2
Mapping rules from VBP-Tree to PN
XML File
3
PIPE2:Plateforme Independed Petri Net Editor
File Edit View Draw Help
RP.XML
PIPE2 Tool

Figure 13. Steps of implementation

In fact, these steps are:

- 1: This step allows to translate from VBP2M to VBP-Tree according the mapping rules detailed in § III.B.
- 2: As PIPE2 save/load its Petri nets in XML file, we generate this file from the VBP-Tree according the mapping rule explained in § III.C. To create the XML file we used the JDOM API (which enables to parses, manipulates, and outputs XML using standard Java constructs).
- 3: Finally, we open the created XML file by PIPE2 to verify by simulation the nets that belongs to a specific VBP.

After choosing a VBP and building the VBP-Tree, we choose PN to visualize and simulate our VBP. The result is shown in Fig. 14.



Transition properties

Version number	1
Creator name	Fatma
Creation date	15/09/2012
State	Freeze
Description	Prepare the offer
Precondition	Committee ^ tender specifications
Operations	Read scope of work Prepare work procedure
Postconditions	

Organizational unit properties

Version number	1
Creator name	Fatma
Creation date	20/09/2012
State	Freeze

Role place properties

Version number	1
Creator name	Fatma
Creation date	20/09/2012
State	Freeze
Organisationnal units	supply department

Informational resources properties

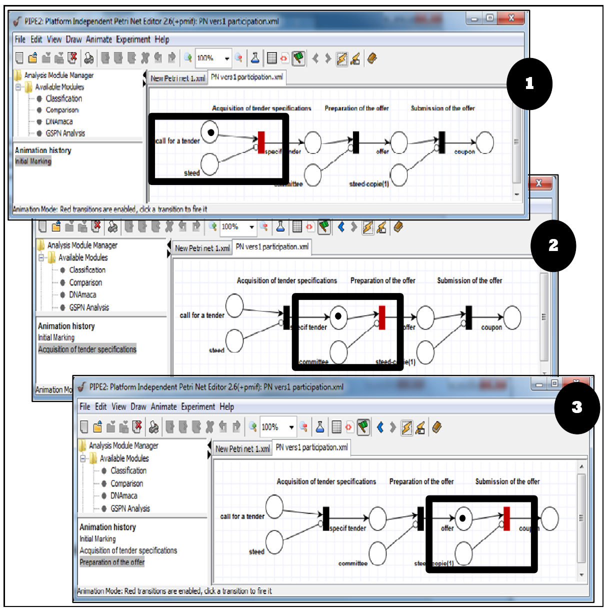
Version number	1
Creator name	Fatma
Creation date	20/092012
State	Freeze
Description	Offer that contains all approximate information (technique and commercial)
Type	document
Nature	internal

Select Mode: Click/drag to select objects; drag to move them

Figure 14. The generation of the first version of participation in a business tender process

- When we click in a transition we obtain on (1) a form displaying its information such version number, creator name, creation date, state, description, precondition, operations and post conditions.
- When we click in a role place we obtain on (2) a form displaying its version information such version number, creator name, creation date, state and organizational units that the role place belongs.
- When we click in an organizational unit place we obtain on (3) a form displaying its version information.
- When we click in an informational resource place we obtain on (4) a form displaying its information such version number, creator name, creation date, state, type and nature.

Fig. 15 shows the simulation result of this VBP visualized in Fig. 14. In fact, we can conclude that all transitions are attainable. So, the reachability property is verified. There are many other properties that can be verified such as the liveness property, the boundedness property, etc. We can also use the analysis techniques that the PIPE2 tool provides.



Animation Mode: Red transitions are enabled, click a transition to fire it

Figure 15. The simulation result of the first version of participation in a business tender process

VI. CONCLUSION AND FUTURE WORK

This paper has presented a solution to simulate the behavioral dimension of versions of BP using an extension of PN. This solution is integrated into a more general framework supporting a process designer for modeling and specifying flexible business processes using the version concept. This framework advocates a MDA approach considering (i) at the CIM level, a specific meta model, the Version Business Process meta model (VBP2M) for modeling versions of BPs (ii) at the PIM level, an extension of the PN meta model for validating by simulation the behavioral dimension of modeled BP versions, and finally, (iii) at the PSM level, several meta models for implementing versions of BPs (e.g., XPDL and BPEL meta models). This paper mainly focuses on the automatic mapping from the CIM level onto the PIM level (i.e., the extension of the PN meta model). Its contributions are the following:

- The specification of an extension of PN in order to support the versions of BPs.
- An automatic mapping of versions of BP modeled according the VBP2M onto BP versions modeled with extended PN meta model.

An implementation of this mapping extending the PIPE2 tool in order to take into account version specificities. Regarding related work, main contributions in BPs [16][17][18] only considered two perspectives (functional and process) and did not consider four other perspectives (operation, informational, organizational and intentional) which are considered as relevant for BP [19][20]. Moreover, theses contributions do not address the mapping from the modeled versions to their graphical representation.

Our future work will take two directions. First perspective, an extensive study of control pattern in PN because we only considered sequence, parallelism and choice patterns. Other more long perspective, we will map versions of BP modeled using the extended PN meta model onto versions of BP described using language relevant from the PSM level of our MDA-based framework: XPDL and BPEL, which are the de-facto standards for implementing BP. Second, we will address execution of specified BP.

References

[1] H. Smith and P. Fingar, "Business Process Management: the Third Wave business process modelling language (bpml) and its pi-calculus foundations," in the Information and Software Technology 45(15), 2003, pp. 1065-1069.

[2] M. Dumas, W. van der Aalst and A. ter Hofstede, "Process-Aware Information Systems: Bridging People and Software through Process Technology," Wiley-Interscience, 2005.

[3] W. van der Aalst, B. Benatallah, F. Casati, F. Curbera and E. Verberk, "Business Process Management: Where Business Processes and Web Services Meet," in the Int. Journal on Data and Knowledge Engineering, 61(1), 2007, pp. 1–5.

[4] I. Ben said, M.A. Chaâbane and E. Andonoff, "A Model Driven Engineering Approach for Modelling Versions of Business Processes using BPMN," in the Int. Conference on Business Information System (BIS'10), Berlin, Germany, May 2010, pp. 254–267.

[5] F. Casati, S. Ceri, B. Pernici and G. Pozzi. "Workflow Evolution," in the Int. journal of Data and Knowledge Engineering. 24(3), 1998, pp. 211–238.

[6] G. Faustmann, "Enforcement vs. Freedom of Action - An Integrated Approach to Flexible Workflow Enactment," in the ACM SIGGROUP Bulletin, 20(3),1999, pp. 5–6.

[7] P. Kammer, G. Bolcer, R. Taylor and M. Bergman, "Techniques for supporting Dynamic and Adaptive Workflow," in the Int. Journal on Computer Supported Cooperative Work, 9(3/4), 2000, pp. 269–292.

[8] S. Rinderle, M. Reichert and P. Dadam, "Disjoint and Overlapping Process Changes: Challenges, Solutions and Applications," in the Int. Conference on Cooperative Information Systems, Agia Napa, Cyprus, 2004, pp.101–120.

[9] W. van der Aalst, A. ter Hofstede, B. Kiepuszewski and A. Barros, "Workflow Patterns," in the Int. Journal on Distributed and Parallel Databases, 2003, 14(1), pp. 5–51.

[10] S. Nurcan, "A Survey on the Flexibility Requirements related to Business Process and Modelling Artifacts," in the Int. Conference on System Sciences, Waikoloa, Big Island, Hawaii, USA, January 2008, pp. 378–387.

[11] H. Schoneneberg, R. Mans, N. Russell, N. Mulyar and W. van der Aalst, "Process Flexibility: A Survey of Contemporary Approaches," in the Int. Workshop on CIAO/EOMAS, at Int. Conf. on Advanced Information Systems, Montpellier, France, June 2008, pp. 16–30.

[12] M. A. Chaâbane, E. Andonoff, L. Bouzguenda and R. Bouaziz. "Versions to Address Business Process Flexibility Issue," in the East-European Conference on Advances in Databases and Information Systems (ADBIS 2009), Riga, 07/10/2009 – 10/10/2009, Janis Grundspenkis, Tadeusz Morzy, Gottfried Vossen (Eds.), Springer Berlin, Heidelberg, September 2009, pp. 2–14.

[13] M. Pesic, H. Schonenberg and W. van der Aalst, "Constraint-Based Workflow Models: Change Made Easy," in the Int. Conference on Cooperative Information Systems, Vilamoura, Portugal, November 2007, pp. 77–94.

[14] W. van der Aalst "Three Good reasons for Using a Petri-net-based Workflow Management System, " In proceedings of the International Working Conference on Information and Process Integration in Entreprises (IPIE'96), Cambridge, Massachusetts, USA, November 1996, pp. 179 – 201.

[15] S. Narayanan and S. McIlraith, "Simulation, Verification and Automated Composition of Web Services, " in the 11th Int. World Wild Web Conference, Honolulu, Hawaii, 2002, pp. 77–88.

[16] M. Kradofler and A. Geppert, "Dynamic Workflow Schema Evolution based on Workflow Type Versioning and Workflow Migration," in the Int. Conference on Cooperative Information Systems, Edinburgh, Scotland, 1999, pp. 104–114.

[17] X. Zhao and C. Liu, "Version Management in the Business Change Context," in the Int. Conference Business Process Management, Brisbane, Australia, September 2007, pp. 198–213.

[18] B. Weber, M. Reichert and S. Rinderle-Ma, "Change Patterns and Change Support Features -Enhancing Flexibility in Process-Aware Information Systems," in Data and Knowledge Engineering 66 (3), September 2008, pp. 438–466.

[19] M. A. Chaâbane, E. Andonoff, L. Bouzguenda and R. Bouaziz, "Dealing with Business Process Evolution," in the Int. Conference on E-Business (ICE-B 2008), Porto, Portugal, July 2008, pp. 267–278.

[20] M. A. Chaâbane, E. Andonoff, R. Bouaziz and L. Bouzguenda, "Modélisation Multidimensionnelle des Versions de Processus," Ingénierie des Systèmes d'Information, Numéro spécial Modélisation d'entreprise, RTSI ISI 15(5), 2010, DOI:10.3166, Lavoisier, Paris, pp. 89–114.

[21] E. Sciore. "Versioning and Configuration Management in Object-Oriented Databases," in the Int. Journal on Very Large Databases, 3(1), 1994, pp. 77–106.

[22] OMG, MDA Guide Version 1.0.1, Document Number: omg/2003-06-01. OMG, Juin 2003.

[23] PIPE Homepage. http://pipe2.sourceforge.net/

[24] P. Bonet, C. M. Llado, R. Puigjaner and W. J. Knottenbelt, "PIPE v2.5: a Petri Net Tool for Performance Modeling," in 23rd Latin American Conference on Informatics, October 2007.

Parameterized Attribute and Service Levels Semantic Matchmaking Framework for Service Composition

Salem Chakhar
CRAD, ESAD, University Laval, Québec, QC, Canada
Email: salem.chakhar@crad.ulaval.ca

***Abstract*—The paper presents a parameterized and highly customizable semantic matchmaking framework. The matchmaking approach on which this framework is based distinguishes three types of matching: (i) functional attribute-level matching, (ii) functional service-level matching, and (iii) non-functional matching. This paper focuses only on functional matching. A series of algorithms is advised for both attribute-level and service-level matching. These algorithms are designed to support a customizable matching process that permits the user to control the attributes matched, the order in which attributes are compared, and the way the sufficiency is computed for both attribute and service levels. Experimental results show that both the proposed algorithms and the original one perform pretty much in the same way.**

***Keywords-web service*; *service composition*; *matchmaking*; *attribute-level matching*; *service-level matching*.**

I. Introduction

Individual web services are conceptually limited to relatively simple functionalities modeled through a collection of simple operations. However, for certain types of applications, it is necessary to combine a set of individual web services to obtain more complex ones, called *composite* or *aggregated* web services [1]. One important issue within web service composition is related to the selection of the most appropriate one among the different candidate composite web services. The selected web service(s) should *mach* at best the specifications provided by the user. Most of existing matchmaking frameworks such as [2] [3] [4] utilize a strict capability-based matchmaking, which is proven [5] to be inadequate in practice. Some recent proposals including [6] [7] propose to use semantics to enhance the matchmaking process but most of them still consider capability attributes only. To avoid the shortcomings of strict capability-based matchmaking, Doshi et al. [5] present a parameterized semantic matchmaking framework that exhibits a customizable matchmaking behavior. One important shortcoming of [5] is that the sufficiency condition defined by the authors is very strict since it requires that all the specified conditions hold at the same time. This seems to be very restrictive in practice, especially for attributes related to the Quality of Service (QoS).

The objective of this paper is to propose a semantic matchmaking framework for web service composition. In this framework, we distinguish three types of matching: (i) functional attribute-level matching, (ii) functional service-level matching, and (iii) non-functional matching. The functional attribute-level matching concerns capability and property attributes. The functional service-level matching supports attribute-level matching and adds a service similarity measure that should be satisfied by the advertised service as a whole. The non-functional matching deals with the QoS attributes. This paper focalizes on functional matching only.

The rest of the paper is as follows. Section II sets the background. Section III addresses functional attribute-level matching. Section IV deals with functional service-level matching. Section V presents experimental results. Section VI discusses related work. Section VII concludes the paper.

II. Background

A. Example Scenario

For the purpose of illustration, we consider a web service use case concerning travel reservation. This example is freely inspired from a use case scenario described in the WSC Web Services Architecture Usage Scenarios [8].

A company (travel agent) offers to people the ability to book complete vacation packages: plane/train/bus tickets, hotels, car rental, excursions, etc. Service providers (airlines, bus companies, hotel chains, etc) are providing web services to query their offerings and perform reservations.

The user gets the location of a travel agent service via an unspecified way (search engine, service directory, etc).

The user provides a destination and some dates to the travel agent service. The travel agent service inquires airlines about deals and presents them to the user.

B. Basic Definitions

We introduce some basic definitions of a service and other service-specific concepts. Some ones are due to [5].

*Definition 1 (**Service**):* A service S is defined as a collection of attributes that describe the service. Let $S.A$ denotes the set of attributes of service S and $S.A_i$ denotes each member of this set. Let $S.N$ denote the cardinality of this set.$\diamond$

Example 1: The travel agent company provides a web service, **bookVacation**, that is defined by the following attributes: service category, input, output, preconditions, postconditions, response time, availability, cost, security, and geographical location.$\diamond$

*Definition 2 (**Service Capability**):* The capability of a service $S.C$ is a subset of service attributes ($S.C \subseteq S.A$), and includes only functional ones that directly relate to its working.$\diamond$

Example 2: The capability of **bookVacation** is: $S.C = \{\text{input, output, preconditions, postconditions}\}$.$\diamond$

*Definition 3 (**Service Quality**):* The quality of a service $S.Q$, is a subset of service attributes ($S.Q \subseteq S.A$), and includes all attributes that relate to its QoS.$\Diamond$

Example 3: The Service Quality of **bookVacation** is: $S.Q = \{$response time, availability, cost, security$\}.\Diamond$

*Definition 4 (**Service Property**):* The property of a service, $S.P$, is a subset of service attributes ($S.P \subseteq S.A$), and includes all attributes other than those included in service capability or service quality.$\Diamond$

Example 4: The property of **bookVacation** is: $S.P = \{$service category, geographical location$\}.\Diamond$

C. Service Matching Types

The input for web service composition is a set of specifications describing the capabilities of the desired service. These specifications can be decomposed into two groups:

- Functional requirements that deal with the desired functionality of the composite service,
- Non-functional requirements that relate to issues like cost, performance and availability.

The non-functional requirements are often called QoS attributes. The QoS attributes can be subdivided into two groups [1]:

- *Hard attributes*: These correspond to QoS attributes for which some obligatory constraints are imposed by the client. Instances that fail to meet these constraints are automatically eliminated,
- *Soft attributes*: These correspond to QoS attributes that should be optimized, i.e., maximized or minimized, according to the user desires.

Hard QoS attributes are very useful in practice since they permit to reduce the computing time. They should be used in a *preliminary analysis step* [1] to reduce the number of candidate compositions. Soft QoS attributes, in the contrary, are used to compute the overall QoS of the remaining candidate compositions.

Furthermore, we may distinguish three types of service matching [1]:

- **Functional attribute-level matching**: This type of matching concerns capability and property attributes and consider each matching attribute independently of the others.
- **Functional service-level matching**: This type of matching concerns capability and property attributes but the matching operation implies the service as a whole.
- **Non-functional matching**: This type of matching concerns QoS attributes. It uses soft QoS attributes.

In the rest of this paper, we will focalize only on functional attribute-level and service-level matching. The non-functional QoS-oriented matching is addressed in [9].

III. Functional Attribute-Level Matching

Functional matching may be defined as the process of discovering a service advertisement that *sufficiently* satisfies a service request [5]. Functional matching is based on the concept of *sufficiency*. The latter relies on the *similarity* measure.

A. Similarity Measure

A semantic match between two entities frequently involves a similarity measure. The similarity measure quantifies the semantic distance between the two entities participating in the match. Following [5], a similarity measure is defined as follows.

*Definition 5 (**Similarity Measure**):* The similarity measure, μ, of two service attributes is a mapping that measures the semantic distance between the conceptual annotations associated with the service attributes. Mathematically,

$$\begin{aligned}\mu : A \times A \rightarrow \quad & \{\text{Exact, Plug-in, Subsumption,} \\ & \text{Container, Part-of, Disjoint}\}\end{aligned}$$

where A is the set of all possible attributes.$\Diamond$

The mapping between two conceptual annotations is called:

- **Exact** map: if the two conceptual annotations are syntactically identical,
- **Plug-in** map: if the first conceptual annotation is specialized by the second,
- **Subsumption** map: if the first conceptual annotation specializes the second,
- **Container** map: if the first conceptual annotation contains the second,
- **Part-of** map: if the first conceptual annotation is a part of the second,
- **Disjoint** map: if none of the previous cases applies.

A preferential total order may now be established on the above mentioned similarity maps.

*Definition 6 (**Similarity Measure Preference**):* Preference amongst similarity measures is governed by the following strict total order:

$$\textbf{Exact} \succ \textbf{Plug-in} \succ \textbf{Subsumption} \succ \textbf{Container} \succ \textbf{Part-of} \succ \textbf{Disjoint}$$

where $a \succ b$ means that a is preferred over b.$\Diamond$

This definition of similarity measure is due to [5]. Other matchmaking frameworks (e.g., [10] [5] [3] [4]) utilize an idea similar to μ, but label it differently.

B. Matchmaking Supported Customizations

The functional attribute-level matching algorithms proposed in this paper extend [5]'s matchmaking algorithm. Indeed, to enhance the work of [5], we propose an additional customization by allowing the user to specify the way the similarity measures are logically aggregated. Indeed, in [5]'s matching algorithm, the match must hold conjointly for all attributes. However, it may be useful to allow the user to specify different types of logical expressions. In fact, other logical connectors than "AND" may be used to combine attribute-level similarity measures.

Accordingly, we distinguish three types of functional attribute-level matching: (i) conjunctive matching based on the use of "AND" connector, (ii) disjunctive matching based on the use of "OR" connector, and (iii) complex matching that uses different logical connectors, especially

"AND", "OR" and "NOT" operators. For the purpose of this paper, only conjunctive and disjunctive functional attribute-level matching are considered. Complex matching is addressed in [9].

C. Functional Attribute-Level Conjunctive Matching

As specified above, functional matching concerns only capability and property attributes. Let S^R be the service that is requested, and S^A be the service that is advertised. A first customization of functional matching is to allow the user to specify a desired similarity measure for each (capability and property) attribute in $S^R.A$. In this case, a sufficient match exists between S^R and S^A in respect to attribute $S^R.A_i$ if there exists an identical attribute of S^A and the values of the attributes satisfy the desired similarity measure. A second customization of the matching process is to allow the user specifying which attributes of the service requested should be utilized during the matching process, and the order in which the attributes must be considered for comparison.

In order to support these customizations, we use the concept of Criteria Table, introduced by [5], that serves as a parameter to the matching process.

*Definition 7 (**Criteria Table**):* A Criteria Table, C, is a relation consisting of two attributes, $C.A$ and $C.M$. $C.A$ describes the service attribute to be compared, and $C.M$ gives the *least preferred similarity measure* for that attribute. Let $C.A_i$ and $C.M_i$ denote the service attribute value and the desired measure in the ith tuple of the relation. $C.\eta$ denotes the total number of tuples in C.$\diamondsuit$

Example 5: Table I shows a Criteria Table example.$\diamondsuit$

Table I
AN EXAMPLE CRITERIA TABLE

$C.A$	$C.M$
input	Exact
output	Exact
precondition	Subsumes
postcondition	Subsumes

A sufficient functional attribute-level conjunctive match between services can now be defined as follows.

*Definition 8 (**Sufficient Functional Conjunctive Match**):* Let S^R be the service that is requested, and S^A be the service that is advertised. Let C be a Criteria Table. A sufficient *conjunctive* match exists between S^R and S^A if for *every* attribute in $C.A$ there exists an identical attribute of S^R and S^A and the values of the attributes satisfy the desired similarity measure as specified in $C.M$. Formally,

$$\forall_i \exists_{j,k}(C.A_i = S^R.A_j = S^A.A_k) \wedge \mu(S^R.A_j, S^A.A_k) \succeq C.M_i \\ \Rightarrow \text{SuffFunctionalConjunctiveMatch}(S^R, S^A) \quad 1 \leq i \leq C.\eta.\diamondsuit \qquad (1)$$

The functional attribute-level conjunctive match is formalized in Algorithm 1. This algorithm follows directly from Sentence (1).

D. Functional Attribute-Level Disjunctive Matching

A less restrictive definition of sufficiency consists in using a disjunctive rule on the individual similarity measures. The attribute-level disjunctive matching is defined as follows.

*Definition 9 (**Sufficient Functional Disjunctive Match**):* Let S^R be the service that is requested, and S^A be the service that is advertised. Let C be a Criteria Table. A sufficient *disjunctive* match exists between S^R and S^A if for at *least one* attribute in $C.A$ it exists an identical attribute of S^R and S^A and the values of the attributes satisfy the desired similarity measure as specified in $C.M$. Formally,

$$\exists_{i,j,k}(C.A_i = S^R.A_j = S^A.A_k) \wedge \mu(S^R.A_j, S^A.A_k) \succeq C.M_i \\ \Rightarrow \text{SuffFunctionalDisjunctiveMatch}(S^R, S^A).\diamondsuit \qquad (2)$$

The functional attribute-level disjunctive match is formalized in Algorithm 2. This algorithm follow directly from Sentence (2). Algorithms (1) and (2) differ only at the level of the last *while* loop.

Algorithm 1: FunctionalConjunctiveMatching

```
Input  : $S^R$, // requested service.
         $S^A$, // advertised requested.
         $C$, // criteria table.
Output: Boolean // success: true; fail: false.
while ($i \leq C.\eta$) do
    while $\left(j \leq S^R.N\right)$ do
        if $\left(S^R.A_j = C.A_i\right)$ then
            Append $S^R.A_j$ to rAttrSet;
        end
        Assign $j \longleftarrow j + 1$;
    end
    while $\left(k \leq S^A.N\right)$ do
        if $\left(S^A.A_k = C.A_i\right)$ then
            Append $S^A.A_k$ to aAttrSet;
        end
        Assign $k \longleftarrow k + 1$;
    end
    Assign $i \longleftarrow i + 1$;
end
while ($t < C.\eta$) do
    if ($\mu(rAttrSet[t], aAttrSet[t]) \prec C.M_t$) then
        return fail;
    end
    Assign $t \longleftarrow t + 1$;
end
return success;
```

E. Computational Complexity

Algorithms 1 and 2 have the same complexity. Generally, we have $S^A.N \gg S^R.N$, hence the complexity of the first outer *while* loop is $O(C.\eta \times S^A.N)$. Then, the worst case complexity of our algorithms is $O(C.\eta \times S^A.N) + \alpha$ where α is the complexity of computing μ. The value of α depends on the approach used to infer $\mu(\cdot,\cdot)$. As underlined in [5], inferring $\mu(\cdot,\cdot)$ by ontological parse of pieces of information into facts and then utilizing commercial rule-based engines which use the fast Rete [11] pattern-matching algorithm leads to $\alpha = O(|R||F||P|)$ where $|R|$ is the number of rules, $|F|$ is the number of facts, and $|P|$ is the average number of patterns in each rule. In this case, the worst case complexity of Algorithms 1 and 2 is $O(C.\eta \times S^A.N) + O(|R||F||P|)$. Furthermore, we observe, as in [5], that the process of computing μ is the most "expensive" step of the algorithms. Hence, we obtain: $O(C.\eta \times S^A.N) + O(|R||F||P|) \asymp O(|R||F||P|)$.

Algorithm 2: FunctionalDisjunctiveMatching

```
Input  : $S^R$, // requested service.
         $S^A$, // advertised requested.
         $C$, // criteria table.
Output: Boolean // success: true; fail: false.
while $(i \leq C.\eta)$ do
    while $\left(j \leq S^R.N\right)$ do
        if $\left(S^R.A_j = C.A_i\right)$ then
            Append $S^R.A_j$ to rAttrSet;
        end
        Assign $j \longleftarrow j + 1$;
    end
    while $\left(k \leq S^A.N\right)$ do
        if $\left(S^A.A_k = C.A_i\right)$ then
            Append $S^A.A_k$ to aAttrSet;
        end
        Assign $k \longleftarrow k + 1$;
    end
    Assign $i \longleftarrow i + 1$;
end
while $(t < C.\eta)$ do
    if $(\mu(rAttrSet[t], aAttrSet[t]) \succeq C.M_t)$ then
        return success;
    end
    Assign $t \longleftarrow t + 1$;
end
return fail;
```

IV. Functional Service-Level Matching

The service-level matching allows the client to use two types of desired similarity: (i) desired similarity values associated with each attribute in the Criteria Table, and (ii) a global desired similarity that applies to the service as a whole. In order to define the sufficient functional service-level match, we need to introduce the concepts of aggregation rule and sufficient single attribute match.

A. Aggregation Rule

The computing of service-level similarity measure requires the definition of an aggregation rule to combine the similarity measures associated with attributes into a single measure relative to the service as a whole.

*Definition 10 (**Aggregation Rule**):* An aggregation rule ζ is a mean to combine the similarity measures into a single similarly measure. Mathematically,

$$\zeta : \mathcal{F}_1 \times \cdots \times \mathcal{F}_\eta \rightarrow \begin{array}{l}\{\text{Exact, Plug-in, Subsumption,} \\ \text{Container, Part-of, Disjoint}\}\end{array}$$

where $\mathcal{F}_j$={Exact, Plug-in, Subsumption, Container, Part-of, Disjoint} $(j = 1, \cdots, \eta)$; and η is the number of attributes included in the Criteria Table.$\Diamond$

The similarity maps given in Section III-A still apply here. Furthermore, the preference amongst similarity measures is governed by the same strict total order given in Definition 6.

Since the similarity measures are defined on an ordinal scale, there are only a few possible aggregation rules that can be used to combine similarity measures:

- **Minimum**: picks out the minimum similarity measure,
- **Maximum**: picks out the maximum similarity measure,
- **Median**: picks out the similarity measure corresponding to the median (in terms of order).
- **Floor**: picks out the similarity measure corresponding to the floor of the median values.
- **Ceil**: picks out the similarity measure corresponding to the ceil of the median values.

The Floor and Ceil rules apply only when there is an even number of similarity measures (which leads to two median values).

B. Sufficient Single Attribute Match

The sufficient single attribute match is defined as follows.

*Definition 11 (**Sufficient Single Attribute Match**):* Let S^R be the service that is requested, and S^A be the service that is advertised. Let C be a Criteria Table. A sufficient match exists between S^R and S^A in respect to attribute $S^R.A_i$ if there exists an identical attribute of S^A and the values of the attributes satisfy the desired similarity measure as specified in $C.M_i$. Formally,

$$\begin{array}{l}\exists_{j,k}(C.A_i = S^R.A_j = S^R.A_k) \wedge \mu(S^R.A_j, S^A.A_k) \succeq C.M_i) \\ \Rightarrow \text{SuffSingleAttrMatch}(S^R, S^A, A_i).\Diamond\end{array} \quad (3)$$

The single attribute matching is formalized in Algorithm 3. This algorithm follows directly from Sentence (3).

Algorithm 3: SuffSingleAttrMatching

```
Input  : $S^R$, // requested service.
         $S^A$, // advertised requested.
         $C$, // criteria table.
         $A_i$, // service attribute.
Output: Boolean // success: true; fail: false.
while $\left(j \leq S^R.N\right)$ do
    if $\left(S^R.A_j = C.A_i\right)$ then
        Append $S^R.A_j$ to rAttrSet;
    end
    Assign $j \longleftarrow j + 1$;
end
while $\left(k \leq S^A.N\right)$ do
    if $\left(S^A.A_k = C.A_i\right)$ then
        Append $S^A.A_k$ to aAttrSet;
    end
    Assign $k \longleftarrow k + 1$;
end
if $(\mu(rAttrSet[i], aAttrSet[i]) \succeq C.M_i)$ then
    return success;
end
return fail;
```

C. Functional Service-Level Matchmaking Algorithm

The service-level similarity measure quantifies the semantic distance between the requested service and the advertised service entities participating in the match by taking into account both attribute-level and service-level desired similarity measures.

*Definition 12 (**Sufficient Functional Service-Level Match**):* Let S^R be the service that is requested, and S^A be the service that is advertised. Let C be a Criteria Table. Let β be the service-level desired similarity measure. A sufficient service-level match exists between S^R and S^A if (i) for every attribute in $C.A$ there exists an identical attribute of S^R and S^A and the values of the attributes satisfy the desired similarity measure as specified in $C.M$, and (ii) the value of overall similarity measure satisfies the desired overall similarity measure β. Mathematically,

$$\begin{aligned}&[\forall i \quad (\text{SuffSingleAttrMatch}(S^R, S^A, A_i)) \quad 1 \leq i \leq C.\eta] \quad \wedge \\ &\quad [\exists j_1, \cdots, j_i, \cdots, j_\eta \quad (\zeta(s_{1,j_1}, \cdots, s_{i,j_i}, \cdots, s_{\eta,j_\eta}) \succeq \beta)] \\ &\Rightarrow \text{SuffFunctionalServiceLevelMatch}(S^R, S^A) \end{aligned} \tag{4}$$

where ζ is an aggregation rule; and for $i = 1, \cdots, \eta$ and $j_i \in \{j_1, \cdots, j_\eta\}$:

$$s_{i,j_i} = \mu(S^R.A_i, S^A.A_{j_i}).\Diamond$$

The service-level matching is formalized in Algorithm 4. This algorithm follows directly from Sentence (4).

Algorithm 4: FunctionalServiceLevelMatching

```
Input  : $S^R$, // requested service.
         $S^A$, // advertised requested.
         $C$, // criteria table.
         $\zeta$, // aggregation rule.
         $\beta$, // overall similarity threshold.
Output: Boolean // success: true; fail: false.
while ($i \leq C.\eta$) do
    if ($NOT$(SuffSingleAttrMatch($S^R, S^A, A_i$))) then
        return fail;
    end
end
while ($i \leq C.\eta$) do
    while ($j \leq S^R.N$) do
        if ($S^R.A_j = C.A_i$) then
            Append $S^R.A_j$ to rAttrSet;
        end
        Assign $j \longleftarrow j + 1$;
    end
    while ($k \leq S^A.N$) do
        if ($S^A.A_k = C.A_i$) then
            Append $S^A.A_k$ to aAttrSet;
        end
        Assign $k \longleftarrow k + 1$;
    end
    Assign $i \longleftarrow i + 1$;
end
if
($\zeta(\mu(rAttrSet[1], aAttrSet[1]), \cdots, \mu(rAttrSet[C.\eta], aAttrSet[C.\eta])) \succeq \beta$)
then
    return success;
end
return fail;
```

D. Computational Complexity

The complexity of the two first *while* loops in Algorithm 3 is equal to $O(S^R.N) + O(S^A.N)$. Generally we have $S^A.N \gg S^R.N$, hence the complexity of the two first *while* loops in Algorithm 3 is equal to $O(S^A.N)$. Then, based on the discussion given in Section III-E, we may conclude that the worst case complexity of Algorithm 3 is $O(|R||F||P|)$, where R, F and P have the same definition as in Section III-E.

Based on the discussion in Section III-E, we can set that the complexity of the first *while* loop in Algorithm 4 is equal to $O(C.\eta \times (|R||F||P|))$ and the complexity of the second *while* loop is equal to $O(C.\eta \times S^A.N)$. Finding the minimum, maximum, median, floor and ceil of a list of n values are $O(n)$ worst-case linear time selection algorithms. Then, the overall complexity of Algorithm 4 is equal to $O(C.\eta \times (|R||F||P|)) + O(C.\eta \times S^A.N) + O(n)$. Then based on the discussion of Section III-E, the worst case complexity of Algorithm 4 is $O(|R||F||P|) + O(n)$.

V. Illustration and Experimental Results

A. Illustration

Let consider the travel agent scenario example given in Section II. Fig. 1 shows a fragment of the Travel Agent Ontology. We consider an *Advertisement* for the travel agent web service who permits the client to book *Accommodation* in the specified *Destination*. Assume that the *Advertisement* has the following Inputs and Outputs:

Inputs:	*Destination*
Outputs:	*Accommodation*, *Sport*, *Cost*

We consider a *Query* from a client who planning a vacation. The client wants to make reservations for a *Hotel* and an *Excursion* at the specified *Destination*. As shown in Fig. 1, the *Hotel* is a subclass of the concept *Accommodation* and *Excursion* is a subclass of the concept *Entertainment*. The Query has the following Inputs and Outputs:

Inputs:	*Destination*
Outputs:	*Hotel*, *Excursion*

Let now focalize on the process of matching *Query* outputs. The same reasoning applies to the process of matching the inputs. The matching algorithm iterates over the list of output concepts of the *Query* and looks to find a match to an output concept in the *Advertisement*. Intuitively, any concept in the output represents a candidate for the match. In this particular example, the initial candidate list is {*Accommodation*, *Sport*, *Cost*}. The matching algorithm first looks to see if there is a match for *Hotel*. Based on Fig. 1, the match between *Hotel* and *Accommodation* will be flagged Exact. Then, the algorithm looks a match for *Excursion*. Based on Fig. 1, the match for *Excursion* with respect to *Accommodation*, *Sport* and *Cost* will fail (since there is no relationship between these concepts and *Excursion* concept). Based on the same reasoning, we obtain an Exact match for the input concept *Destination*.

Assume now that the Criteria Table is as in Table II. The desired similarity specified by the Criteria Table holds only for *Destination* and *Hotel*. Hence, the attribute-level conjunctive matching algorithm will fail but the attribute-level disjunctive matching algorithm will success.

Table II
Criteria Table

$C.A$	$C.M$
input	Exact
output	Exact

Suppose now that the *Advertisement* has the following Inputs, Outputs and Categories:

Inputs:	*Destination*
Outputs:	*Accommodation*, *Entertainment*, *Cost*
Categories:	*AirplaneModel*

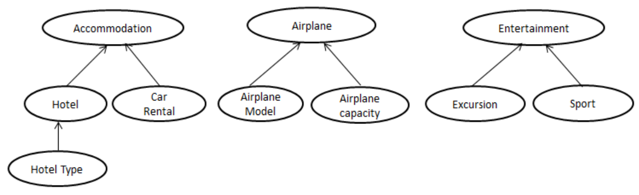


Figure 1. Travel Ontology

Suppose also that *Query* has the following Inputs, Outputs and Categories:

Inputs:	*Destination*
Outputs:	*HotelType*, *Excursion*
Categories:	*AirplaneModel*

Based on the same reasoning as earlier, the match for *Destination*, *Excursion* and *AirplaneModel* concepts will be flagged Exact; and the match for concept *HotelType* will be flagged Plug-in. Assume that the Criteria Table is as follows:

Table III
CRITERIA TABLE

C.A	*C.M*
input	Exact
output	Exact
service category	Subsumption

The instantiation of the last instruction in Algorithm (4) will then look like ζ(Exact, Plug-in, Exact, Exact). Then, the result of Algorithm (4) according to different aggregation rules (except Median which does not apply here) is as follows:

Table IV
RESULT OF AGGREGATION

Aggregation rule (ζ)	Minimum	Maximum	Floor	Ceil
Result	Plugin	Exact	Plugin	Exact

B. Implementation and Experimental Results

For the purpose of implementation, we used the same architecture proposed by [10]. We used the Protege editor [12] to browse and edit OWL [13] ontologies. The Mindswap OWL-S API [14] has been used to load the OWL ontologies into the Knowledge Base and parse the OWL-S [13] Queries and Advertisements. We used the Pellet reasoner [15] to classify the loaded ontologies and Jena API [16] to query the reasoner for concept relationships. To test the algorithms, we used 10 ontologies from the OWTLS-TC (service retrieval test collection from SemWebCentral) in our Knowledge Base. About 500 advertisements from OWLS-TC were loaded into the advertisement repository. Experiments were designed to measure execution time of the algorithms. All measurement points shown are average results taken from 50 runs. The data sets for clients and providers were randomly generated. Fig. 2 shows the execution time of all algorithms.

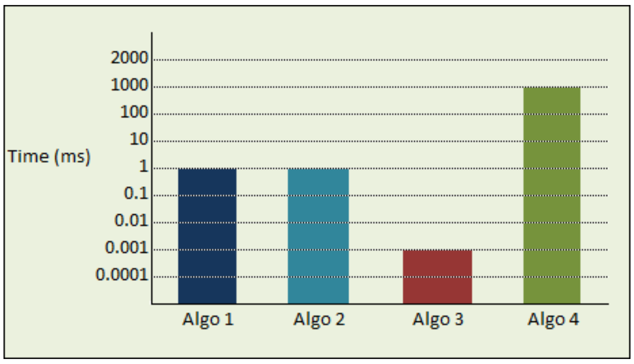


Figure 2. Execution time of algorithms

VI. RELATED WORK

Existing matchmaking frameworks such as [2] [3] [4] utilize a strict capability-based matchmaking, which is proven [5] to be inadequate in practice. Some recent proposals [6] [7] propose to use semantics to enhance the matchmaking process but most of them still consider capability attributes only. Doshi et al. [5] present a parameterized semantic matchmaking framework that exhibits a customizable matchmaking behavior. This framework has three main merits. First, it uses semantic information in the matching process which has been proven in information retrieval [17], fuzzy database [18] and semantic web services [19] [20] [21] to be more useful than simple syntactic search. Second, it uses both functional and non-functional matchmaking. Third, by enabling the user to specify matchmaking criteria and conditions. One important shortcoming of [5] is that the sufficiency condition defined by the authors is very strict since it requires that all the specified conditions hold at the same time.

In [22], Fu et al. transform the problem of matching web services to the computation of semantic similarity between concepts in domain ontology. They developed a semantic distance measure to provide a quantitative similarity measures to support matching in semantic web services. Bellur and Kulkarni [10] improve [3]'s matchmaking algorithm and propose a greedy-based algorithm that relies on the concept of matching bipartite graphs. Their idea permits to avoid problems faced with the original [3]'s matchmaking algorithm.

In [9], we extended the work given in this paper by adding generic attribute-level and non-functional QoS matching. The non-functional matching permits to categorize web services into different ordered QoS classes. The user should then select one web service from the highest QoS class for implementation.

We think that the matchmaking framework presented in this paper extends and improves the works of [10] [5] [3]. Specifically, the proposed framework adds more customization, relaxes matchmaking conditions and supports three types of matching. An important addition of the proposed framework is the service-level matching which, to the best knowledge of the author, previous studies have not dealt with it.

VII. Conclusion and Future Work

In this paper, we presented a parameterized and highly customizable semantic matchmaking framework. A series of highly customizable algorithms is advised for both attribute-level and service-level matching. These algorithms are designed to support a customizable matching process that permits the user to control the attributes matched, the order in which attributes are compared, as well as the way the sufficiency is computed for both attribute and service levels. In this paper, we addressed functional attribute-level conjunctive matching, functional attribute-level disjunctive matching and functional service-level matching. We extended the work given in this paper in [9] by adding generic attribute-level and non-functional QoS matching.

The matchmaking framework proposed in this paper and its extended version given in [9] constitutes a basic component of a layered system, called QoSeBroker, for web service composition that we have proposed in [1]. In the future, we plan to conduce additional experimental tests in order to analyze the performance of the matching algorithms. We also intend to implement the subsequent components of QoSeBroker system.

References

[1] S. Chakhar, "QoS-enhanced broker for composite web service selection," in *Proc. of the 8th International Conference on Signal Image Technology & Internet Based Systems*, Sorrento - Naples, Italy, Nov. 2012, pp. 533–540.

[2] L. Li and I. Horrocks, "A software framework for matchmaking based on semantic web technology," in *Proc. of the 12th International World Wide Web Conference*, Budapest, Hungary, May 1999, pp. 331–339.

[3] M. Paolucci, T. Kawamura, T. Payne, and K. Sycara, "Semantic matching of web services capabilities," in *Proc. of the First International Semantic Web Conference on The Semantic Web*, Sardinia, Italy, Jun. 1999, pp. 333–347.

[4] K. Sycara, M. Paolucci, M. van Velsen, and J. Giampapa, "The retsina mas infrastructure," *Autonomous Agents and Multi-Agent Systems*, vol. 7, no. 1-2, pp. 29–48, 2003.

[5] P. Doshi, R. Goodwin, R. Akkiraju, and S. Roeder, "Parameterized semantic matchmaking for workflow composition," IBM Research Division, Tech. Rep. RC23133, Mar. 2004.

[6] S. Ben Mokhtar, A. Kaul, N. Georgantas, and I. Valérie, "Efficient semantic service discovery in pervasive computing environments," in *Proc. of the ACM/IFIP/USENIX 2006 International Conference on Middleware*, Melbourne, Australia, Nov. 2006, pp. 240–259.

[7] R. Guo, J. Le, and X. L. Xiao, "Capability matching of web services based on OWL-S," in *Proc. of the 16th International Workshop on Database and Expert Systems Applications*, Copenhagen, Denmark, Aug. 2005, pp. 653–657.

[8] H. Hao, H. Haas, and D. Orchard. (2004) Web services architecture usage scenarios. [Online]. Available: http://www.w3.org/TR/ws-arch-scenarios/ [retrieved: Nov., 2012]

[9] S. Chakhar, "QoS-aware parameterized semantic matchmaking for web service composition," in *Proc. of the 9th International Conference on Web Information Systems and Technologies*, Aachen, Germany, May 2013, unpublished.

[10] U. Bellur and R. Kulkarni, "Improved matchmaking algorithm for semantic web services based on bipartite graph matching," in *Proc. of the IEEE International Conference on Web Services*, Salt Lake City, Utah, USA, Jul. 2007, pp. 86–93.

[11] C. Forgy, "Rete: A fast algorithm for the many patterns/many objects match problem," *Artificial Intelligence*, vol. 19, no. 1, pp. 17–37, 1982.

[12] Protege: Ontology editor and knowledge-base framework. [Online]. Available: http://protege.stanford.edu/ [retrieved: Nov., 2012]

[13] G. Antoniou and F. van Harmelen, "Web ontology language: OWL," in *Handbook on Ontologies*, 2nd ed., S. Staab and R. Studer, Eds. Berlin / Heidelberg, Germany: Springer-Verlag, 2009, pp. 91–110.

[14] MINDSWAP: Maryland information and network dynamics lab semantic web agents project, OWL-S API. [Online]. Available: http://www.mindswap.org/2004/owl-s/api/ [retrieved: Nov., 2012]

[15] Pellet: An OWL DL reasoner. [Online]. Available: http://pellet.owldl.com/ [retrieved: Nov., 2012]

[16] Jena: Java framework for building semantic web. [Online]. Available: http://jena.sourceforge.net/ [retrieved: Nov., 2012]

[17] A. K. Kirykov and K. S. Iv, "Ontologically supported semantic matching," in *Proc. of NoDaLiDa'99: Nordic Conference on Computational Linguistics*, Trondheim, Norway, Dec. 1999, pp. 9–10.

[18] R. Bouaziz, S. Chakhar, V. Mousseau, S. Ram, and A. Telmoudi, "Database design and querying within the fuzzy semantic model," *Information Sciences*, vol. 177, no. 21, pp. 4598–4620, 2007.

[19] J. Cardoso, J. Miller, and S. Emani, "Web services discovery utilizing semantically annotated WSDL," in *Reasoning Web*, C. Baroglio, P. Bonatti, J. Maluszynski, M. Marchiori, A. Polleres, and S. Schaffert, Eds. Berlin / Heidelberg, Germany: Springer-Verlag, 2008, pp. 240–268.

[20] C. Mateos, M. Crasso, A. Zunino, and M. Campo, "Supporting ontology-based semantic matching of web services in movilog," in *Advances in Artificial Intelligence*, S. R. J. Sichman, H. Coelho, Ed. Berlin / Heidelberg, Germany: Springer-Verlag, 2006, pp. 390–399.

[21] P. V. C. E. Wu, C., "Latent semantic analysis - the dynamics of semantics web services discovery," in *Advances in Web Semantics I*, R. M. K. S. T. Dillon, E. Chang, Ed. Berlin / Heidelberg, Germany: Springer-Verlag, 2009, pp. 346–373.

[22] P. Fu, S. Liu, H. Yang, and L. Gu, "Matching algorithm of web services based on semantic distance," in *Proc. of the 2009 International Workshop on Information Security and Application*, Qingdao, China, Nov. 2009, pp. 465–468.

Sales Prediction with Parametrized Time Series Analysis

Michael Schaidnagel*, Christian Abele*, Fritz Laux†, Ilia Petrov†
Fakultät Informatik
Reutlingen University
D-72762 Reutlingen, Germany
{*michael.schaidnagel*|*christian.abele*}@*student.reutlingen-university.de**
{*fritz.laux*|*ilia.petrov*}@*reutlingen-university.de*†

***Abstract*—When forecasting sales figures, not only the sales history but also the future price of a product will influence the sales quantity. At first sight, multivariate time series seem to be the appropriate model for this task. Nontheless, in real life history is not always repeatable, i.e. in the case of sales history there is only one price for a product at a given time. This complicates the design of a multivariate time series. However, for some seasonal or perishable products the price is rather a function of the expiration date than of the sales history. This additional information can help to design a more accurate and causal time series model. The proposed solution uses an univariate time series model but takes the price of a product as a parameter that influences systematically the prediction. The price influence is computed based on historical sales data using correlation analysis and adjustable price ranges to identify products with comparable history. Compared to other techniques this novel approach is easy to compute and allows to preset the price parameter for predictions and simulations. Tests with data from the Data Mining Cup 2012 demonstrate better results than established sophisticated time series methods.**

***Keywords*-sales prediction, multivariate time series, price-sales correlation, parametrized predictor.**

I. Introduction

Sales prediction is an important goal for any time series based analysis [1], [2]. The task consists of forecasting sales quantities given the sales history. This can be achieved by extending the time series into the future.

The prolongation of the time series into the future is determined by the underpinning time series model [3]. If this model is not well supported by the empirical data it is likely that the accuracy of the forecast is low. So the challenge is to find data build ”similar” products or situations (e.g. time or price) and form clusters of products for building a prediction model. If a major sales factor like the product price changes, the model based solely on previous sales will lead to wrong forecasts. Therefore, it is important to include the price as parameter into the model in addition to the sales history.

Standard solutions to this problem have been to supply a long history of sales with sufficient data to validate the model and to correlate the sales data with the variable product price. The mathematical tools of choice for analyzing multiple time series simultaneously are multivariate statistical techniques like Vector AutoRegressive (VAR) models [4], [5] or such as the Vector ARIMA (AutoRegressive Integrated Moving Average) [6]. The model parameters are estimated with least square or Yule-Walker functions [4, p. 181]. The accuracy of the estimator depends on the number of observations and its ”strength” of correlation.

To illustrate the process consider an excerpt from the Data Mining Cup 2012 [7] dataset (Table I:

Table I
Sample Data, Data Mining Cup 2012

day	Prod#	price	quantity
1	1	4.73	6
1	2	7.23	0
1	3	10.23	1
1	4	17.90	0
...	...	...	...
1	570	7.91	0
2	1	4.73	12
2	2	7.23	1
...	...	...	...
42	569	9.83	2
42	570	7.84	0
43	1	5.35	?
43	2	7.47	?
...	...	...	...
43	570	7.84	?
...	...	...	...
56	570	8.12	?

The information provided comprises a collection of 570 products whose history of sales and prices are given over a period of 42 days. The task was to predict the sales quantities for the next 14 days where the daily sales price was preset. The majority of products produced only low quantity sales. Comparisons with other sales data showed a similar distribution [8] [9] which indicates that the sample is typical for larger collections.

When we tried to predict the future sales with commercial ARIMA products we experienced a low prediction quality with a relative accuracy of only 47%.

The disappointing results from professional tools implementing ARIMA encouraged us to look for a simpler and better prediction model. First of all we assumed that the future price is causally influenced and should not be treated as stochastic variable. Second, it might be helpful to filter out cyclic behavior from the "white noise" in case of low volume sales. Third, the proposed approach is suitable for online scenarios: (i) it has low computational overhead, the underlying model is simple and maintainable; (ii) the algorithm can operate on partial datasets and can be used for incremental forecasting.

A. *Structure of the Paper*

In the next subsection follows a discussion of related work and we contrast it with our contribution.

The rest of the paper is structured as follows: The data profile under investigation and the research problem will be described formally in Section II. In Section III we present our parametrized time series algorithm that predicts sales volumes with variable product prices and low data support. The following Section IV gives a description of the technical framework for the implementation of the prototype. The results are discussed in Section V and compared with standard methods found in commercial products like ARIMA. From these experiences we draw our conclusion in the last section.

B. *Related Work*

Adaptive correlation methods for prognostic purposes have been proposed early in the 1970^{th} by Griese [10] and more specifically as AutoRegressive Moving Average (ARMA) method by Box and Jenkins [11]. As ARMA is constrained to a stationary stochastic process the ARIMA is of more practical use as it can handle time series with a linear trend and is therefore widely implemented.

The idea behind ARMA and ARIMA is that the model adapts automatically to a given history of data. A natural extension is to include other influential factors beside the prognostic value itself. This leads to multivariate models, namely Vector Autoregressive (VAR) models [6]. The development of the model was influenced and motivated by critiques of Sims [12] and Lucas [13]. In essence, their statement is: every available data is potentially correlated.

If the model is extended to cover the influence from correlated data this leads to a vectorial stochastic model $(\mathbf{X}_t(\pi, \pi_r))$ that allows not only the serial time dependence t of each component but also the interdependence of products π and product prices π_r. We prefer to use parentheses () for a stochastic process instead of braces $\{\}$ because it is rather a sequence of stochastic variables than a set.

To estimate the parameters of such a multivariate ARMA process the following equation ([14, p. 417], [4, p. 167]) has to be solved:

$$\Phi(L)X_t(\pi,\pi_r) \quad = \quad \Theta(L)Z_t \qquad (1)$$

where L denotes the backshift (lag) operator and

$$\Phi(x) \quad := \quad I - \Phi_1 x - \Phi_2 x^2 - \ldots - \Phi_p x^p \qquad (2)$$
$$\Theta(x) \quad := \quad I + \Theta_1 x + \Theta_2 x^2 + \ldots + \Theta_q x^q \qquad (3)$$

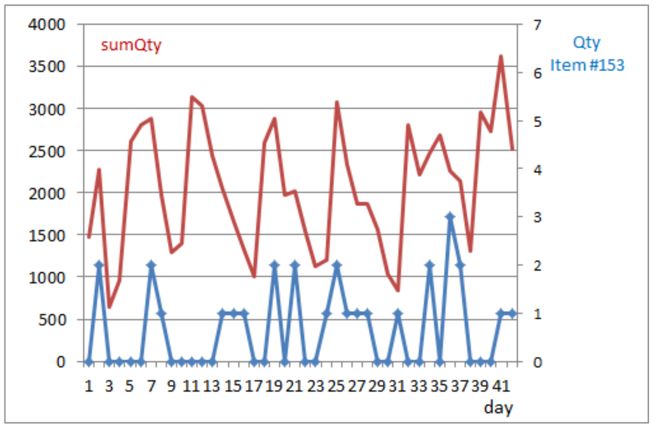


Figure 1. Seven day periodicity for the overall sales data (DMC2012) and a typical low selling product (item # 153)

are matrix-valued polynomials with dimensions of p (order of regression) and q (order of moving average). Z_t denotes a multivariate "white noise" process.

There is one major drawback to this approach in our problem setting. The model treats all historical input values as stochastic variables. However, the product price does not vary *stochastically*, its value is preset by the vendor. Economic models assume a causal dependency between the price of a product and its sales quantities (see Arnold [15, chap. 17]). Variations in consumer demand are caused by various factors like price, promotions, etc [16]. This causal dependency is not modeled by VAR methods. This is an issue for the multivariate model.

Another complication arises from the noisy periodicity resulting from low volume sales. The low sales quantity introduces a kind of random pattern that makes it hard to find even a known periodicity. In the sample data the overall sales history shows a clear 7-day periodicity (see Figure 1) but not for individual products.

Cyclic sales quantities are a typical behavior for short shelf-life products and are important for building a causal sales model. Doganis et al. [17] investigated the sales quantity of fresh milk (a short shelf-life product) in Greece. They used a genetic algorithm applied to the sales quantities of the same weekday of last year. Our approach is only similar in that we take corresponding weekdays but it differs in how we analyse the weekly periodicity and correlate it with the sales prices.

To recapitulate, there are two general arguments against the multivariate VAR approach sketched above: Granger and Newbold [18] showed that simpler models often outperformed forecasts based on complex multivariate models. And Lucas [13] criticized that the economic models are too static and that "any change in policy will systematically alter the structure of the econometric model". Applied to the sales forecast situation the variation of the price does not play a

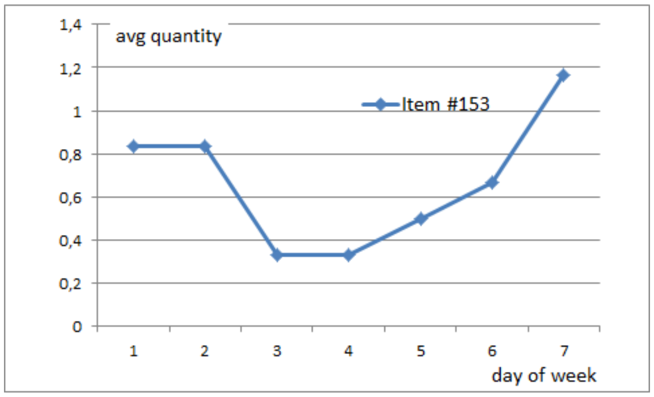


Figure 2. Seven day periodicity for the sales of item # 153

stochastic, but a *systematic*, i.e. a functional, role.

Our idea is to filter the seasonality by a period-based ”folding” of the sales quantity, i.e. the aggregation of sales quantities for the same weekdays. This cancels the stochastic variation and accumulates the seasonal effect. Applying such a model improves the prediction coverage and accuracy for low volume data with a cyclic behavior.

II. Problem Description and Contribution

In Section I we pointed out that the nature of the data and its sales profile play an important role for the time series analysis. In particular, the influence of price and periodicity are dominant factors as we will see in the following. For this reason we present first the characteristics of the data.

A. Data Profile

The 570 products of the sample data realized a total of 86641 units sold. The average price ranged between 14.46 and 15.92 over the period of 42 days. The maximum price variability of a single product is ±48%, but on average the price varies only by ±9%. However, for high selling products ($>$ 500 units) the variability stands at ±15%.

The total sales quantity per product ranged from 17 to 2083 over the 6 weeks. Broken down to the day level the product price ranged from 0.24 to 152.92 and the sales quantities between 0 and 193. The sample had average sales per product of 152 units with a standard deviation of 257 which indicates a high sales variability of the items.

This conjecture is confirmed by the product sales ranking that roughly follows a shifted hyperbolic distribution (see Figure 3) which supports that low volume sales contribute significantly to the overall sales and may not be neglected. 506 products out of 570 sell less than 250 units in total but contribute with approximately the same quantity sold (43991 units) as the 64 high selling products.

The low volume sales (sum of sales $<$ 250) showed a strong positive trend ($\approx$ 40% increase over 42 days) whereas the high volume sales (sum of sales $\geq$ 250) had a more stationary behavior. In the sample data are more than 100

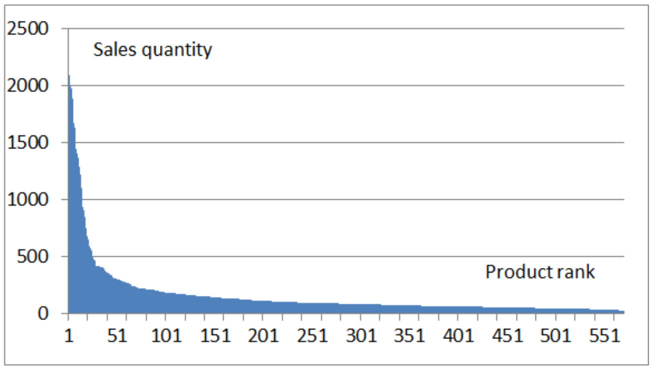


Figure 3. Sales quantity ranking of sample data (DMC2012)

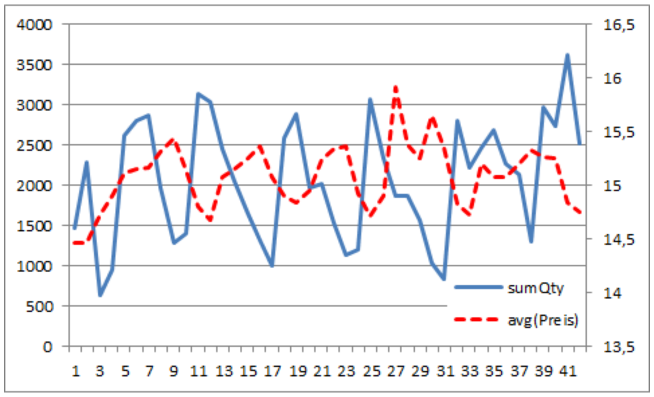


Figure 4. Sales quantity and average price time series of sample data (DMC2012)

products that sell less than six units a day. Nearly all of them sell none at half of the time.

The above properties require that an adequate forecasting algorithm, which has to allow for low volume sales with high variability and be able to accustom for some price variability.

A 7 day sales periodicity was highly visible over the whole sample (see Figure 1) but not on the product level. Tests with sample low selling products with an assumed periodicity of 7 days produced better results than without this assumption. Again, the cumulative values of sales quantities and prices plotted over time give clear indication for a negative price-quantity correlation and a 7 day periodicity except for the first week (Figure 4). These results indicate that our algorithm has to deal with hidden periodicity.

In order to avoid the risk that our method is tailored for a particular data set we investigated a second data set with a completely different profile. The data records contain Piglet Market Data (PMD), including price and quantity sold for weekly auctions over a period of six years. The PMD does not show any clear periodicity but the market tended lower towards the end of the year. It is characterised by the following quantities: (i) Min. Qauntity - 701; (ii) Max. Quantitiy - 2063; (iii) Min. Price - 20 €; (iv) Max. Price -

71.5 €; (v) the trace sequence length spans 311 data points; (vi) Qantity standard deviation - 207.97; (vii) price standard deviation - 10.38 € which depicts a similar variance as the previous time series but without periodicity. We will show in Section V that our method produces a more accurate prediction.

B. Formal Problem Description

The problem consists of developing a parametrized time series model that is able to forecast future sales quantities depending on the given sales history and a price parameter. The solution of the stochastic Equation (1) is a multidimensional mapping

$$\begin{aligned} F: \quad (\mathbf{\Pi}, \mathbf{T}) &\longrightarrow (\mathbb{R}^+, \mathbb{N}_0) \\ (\pi, t) &\longmapsto (\hat{\pi}_r, \hat{x}_t) \end{aligned} \tag{4}$$

where $\mathbf{\Pi}$ is the set of products and $\mathbf{T}$ are consecutive time intervals. A product $\pi \in \mathbf{\Pi}$ is described by its identification number π_i and its price π_r. The mapping F computes sales quantity $\hat{x}_t$ and price $\hat{\pi}_r$ for every product π and time interval t .

The vector time series $(\hat{\pi}_r, \hat{x}_t)$ is a concrete realisation of the stochastic process (X_t) of Equation (1). The mapping F has to be adjusted so that the process (X_t) explains best a given realisation. This can be done by various estimator functions: least square error, Yule-Walker, maximum-likelihood, or Durbin-Levison algorithms. This is where our approach differs from the traditional because in real business the price is not a stochastic variable but is preset by the vendor. Instead of predicting the future price $\hat{\pi}_r$ we use the price as input parameter.

Having fixed the model in this way it is possible to transform the mapping F to the following form:

$$\begin{aligned} F_r: \quad (\mathbb{N}, \mathbb{R}^+, \mathbf{T}) &\longrightarrow \mathbb{N}_0 \\ (\pi_i, \pi_r, t) &\longmapsto \hat{x}_t \end{aligned} \tag{5}$$

With this predictor $F_r(\pi_i, \pi_r, t)$ it is possible to forecast the sales quantities for future time periods $t > T$ (T is the present time) of a product $\pi \in \Pi$ using the future price π_r as input.

C. Contribution

By restricting our approach to model a linear trend, seasonality, and using historic and future prices as causal parameter leads to a predictor function that is easy to compute and explain. It yields higher accuracy for data with hidden periodicity and variable prices than the ARIMA model. The novelty of our contribution comprises:

- a model that has a causal explanation
- where the future price is a major input factor and
- the overall periodicity is respected by individual items.

The prediction function can also be used for simulation to see how the price will influence the sales quantity.

III. The Parametrized Time Series Model

For a causal predictor function F_r we need to identify and quantify all influencing factors. Therefore we analyzed different correlations of the attributes quantity, price and time. We used the standard Pearson Correlation [19] as a measure to determine the linear dependence between two time series. It is widely used and can range between -1 and $+1$. This section will present the relations which have been analyzed.

A. Price-Sales Correlation

The main conjecture was that the price has a causal influence on the quantity. This is justified by the price elasticity of demand theory by Alfred Marshall [20]. As the correlation coefficients of all 570 products ranged from -0.6515 to $+0.3471$, we expected that the products with strong correlation exhibit a better prediction accuracy. Surprisingly this seemed not to be the case.

A systematic analysis with three synthetic time series lead to an explanation. The first series had a growing price trend, the second and third had a cyclic price development where one product responded immediately and the other responded with a delay. ARIMA did recognize the price trend but forecasted a constant quantity instead of a decreasing one. This was the result of the low integer sales numbers that produced a monotone decreasing step function. Our approach managed to forecast the right quantities as long as a matching price was present in the history.

Surprisingly ARIMA could not deal well with the systematic cyclic price development and a detailed analysis showed that the step function of the price (price was kept constant for two days) was the reason. Figure 5 shows the result of the ARIMA compared to our F_r algorithm (see equation 5). The lower Qty (Figure 5) values at the extrema produced by our algorithm results from the delay in the response to the price change. Without lag, no damping of extrema occurs in F_r.

B. Price Similarity

We analyzed correlations between the price development of different products. The assumption was to find product bundles which are linked together via their price development. For the analysis the prices were first normalized such that we could easily compare the different price levels. Several bunches of products were linked together via their prices. But the corresponding sales figures of these products were not related. This is why we ignored the possible cross price influence from other products for the forecast.

C. Sales Periodicity

One of the most interesting properties of the given data was the periodicity of the total sales curve. It showed a clear 7 day period (Figure 1). This period was not directly observable in most sales time series of individual products.

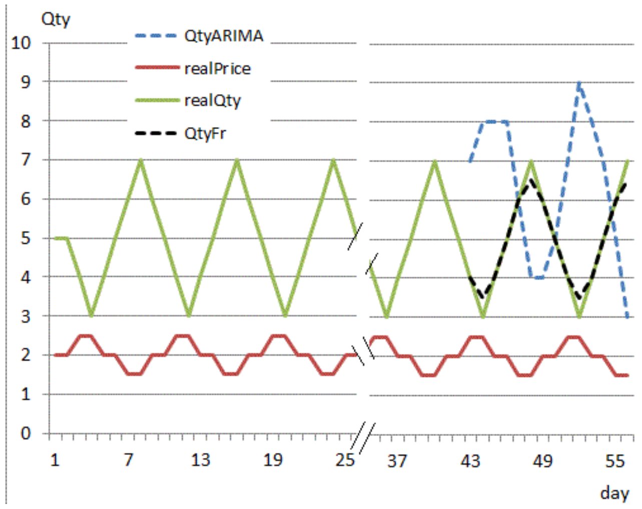


Figure 5. Forecast of synthetic time series with delayed price-sales dependency

Also the Pearson Correlation between the sum curve and the single products was too low to draw any conclusions. Nevertheless, since the total sales curve consists of all products, there must be a hidden periodicity within the individual products.

When we used this 7-day periodicity and summed up the sales of corresponding days most products confirmed the 7-day periodicity. Cumulative sales quantities over matching weekdays reveal hidden cyclic pattern even for low volume sales products. Figure 2 shows the periodicity pattern of product #153 that cannot be seen from its original time series (Figure 1).

Using this 7-day periodicity we were able to improve the forecast quality of our method. The algorithm takes only prices from the same days of each period (see Figure 7 for an example) and computes its trend ($trend(Qtylist)$). The trend computation can be performed in different way; for our purses we use linar trend. A more detailed explanation will follow in the next subsection.

A systematic spectral analysis discovered not only the dominant weekly patterns but weaker 4, 5 and 14 days patterns.

D. The Parametrized Predictor Function

Putting all correlation observations together the result is a function F_r whose pseudocode is shown in Figure 7. As an input, it takes the price π_r of a product π at the prediction day t, the periodicity and a price range δ. The upper and lower price limits are set to $\pm\delta$ percent. Using the periodicity from the previous analysis the algorithm looks for prices $\pi_r(w)$ that occur on days $w = t$ modulo $period$. For example, if the prediction day is 43 and the periodicity is 7 days, only the information from the days 36, 29, 22, 15, 8 and 1 will be considered (see Figure 6). If the price

```
**Input:** $t > T$ // prediction day
    $\pi_r$ (input) // price at day t
    $\delta$ (input) // price range (e.g. $\pm$ 10%)
    $period$ (input)
**Def:** $u, l$ // upper & lower price limit
    QtyList // list of sales quantities
    $w$ // = t - n * period
    $\hat{x_t}$ // predicted quantity at day t
**for each** $\pi \in \Pi$ {
    $u := \pi_r(1 + \delta/100);\ l := \pi_r(1 - \delta/100)$
    $w := t - period$
    **while** $(w \geq 1)$ {
      **if** $(\pi_r(w) < u) \,\&\, (\pi_r(w) > l)$
        QtyList.add($p(w)$) // $p(w)$ is Qty on day $w$
      $w := w - period$
    }
    **if** (QtyList $\neq \emptyset$)
      $\hat{x_t} := trend$(QtyList)
      **return** $\hat{x_t}$
    **else**
      **return** nil
}
```

Figure 7. Parametrized Sales Prediction Algorithm F_r

on such a day is outside of the upper or the lower limit (day 1 in our example), the sales quantity is ignored. If the price is within the bounds, the corresponding quantity on that day ($p(w)$) is inserted into the quantity list ($QtyList$). After all matching quantities have been selected, the forecast quantity is computed as linear trend ($trend(Qtylist)$) of these quantities.

If all prices are outside of the upper and lower limit, no forecast is produced. The procedure may be repeated with enlarged upper and lower limits if needed. This algorithm defines a simple forecasting model that takes into account the sales trend, the periodicity, and the price influence to predict sales quantities.

IV. Technical Framework and Infrastructure

This section covers some technical details about execution and implementation of the two models discussed in this paper.

A. ARIMA Model Execution

The Microsoft Visual Studio 2008 and Microsoft SQL Server 2008 were used to apply the ARIMA model on the given data set. In order to run the ARIMA mining models a OLAP cube was build. It consists of the dimensions *price*, *product* and *time*. In the corresponding time series mining model we used *itemId* and *day* as key attributes and the *price* attribute as input. The *quantity* was set as predictable attribute. Most model parameters were left as default, except the minimum series value and the periodicity hint. The

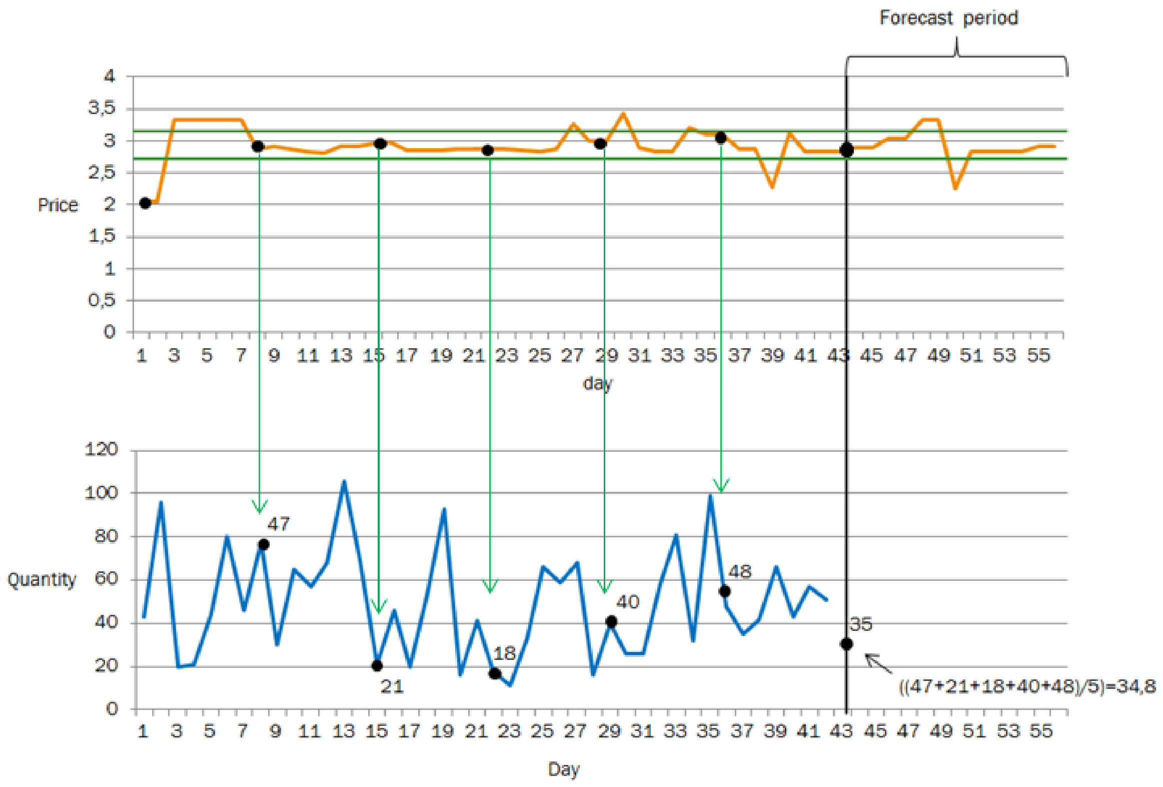


Figure 6. Illustration of prediction concept using trend, periodicity, and a parametrized price

following listing shows all model parameters used:

AUTO_DETECT_PERIODICITY	0.6
FORECAST_METHOD	ARIMA
HISTORIC_MODEL_COUNT	1
HISTORIC_MODEL_GAP	10
INSTABILITY_SENSITIVITY	1.0
MAXIMUM_SERIES_VALUE	$+1E308$
MINIMUM_SERIES_VALUE	0
MISSING_VALUE_SUBSTITUTION	None
PERIODICITY_HINT	7
PREDICTION_SMOOTHING	0.5

B. *Implementation of the Suggested Model in Java*

Our own approach is implemented in Java. We used Eclipse (Version: Indigo Service Release 1) with Java Platform Standard Edition 6.0 (JRE6). The data was stored in a MySQL database on an Apache web server (2.2.21). During execution time the data is queried from the database, the model parameters computed and the forecast results are instantly stored in the corresponding result table in the database. The model was developed using the standard java.sql.* package which was used to interface with the database and for SQLException handling.

V. RESULTS

The absolute prediction error was measured as $|realQty - predictQty|$. The performance of F_r was measured against ARIMA and the results are given as sum of absolute error numbers over 14 prediction days. The F_r algorithms benefited from two input parameters: the hidden periodicity that was calculated in a previous step and the predefined future price. The hidden sales periodicity contributed for an improvement of about 20%. The overall forecast was improved by 26.7%. The price influence was less dominant than expected, but was determinant for a cluster of 26 products. Cluster characteristics: (i) correlation < -0.25; (ii) relative standard error < 0.25; (iii) sales quantity > 160; and (iv) price variation $((max(\pi_r) - min(\pi_r)) > 4)$. In total, F_r could forecast this cluster 36.4% better than ARIMA.

A. *Comparison of Results with ARIMA*

In this section we compare the results of ARIMA and our F_r method. The following table shows the prediction error points of both ARIMA and F_r. The prediction error is calculated as sum of the absolute difference between real and predicted values ($\sum \mid realQty - predictedQty \mid$, $predictedQty \in \{predictedARIMA, predictedF_r\}$). The total error of all 570 products was 30152 for the ARIMA and 22093 for F_r. This is an improvement of 26.7% compared to ARIMA (Table II). For further analysis we splitted the products into disjoint sets according to different criteria. This allowed us to find the strengths and weaknesses of our algorithm in terms of total sales quantity, sales sparsity, and price.

In the case of PMD we predicted the sales quantities for a full year. The result analysis for this forcast yields an improvement of 20% for the F_r alogrithm over ARIMA. This is less than for the DMC set but with more then 20% still substantial.

Table II
COMPARISON OF ARIMA PREDICTION ERROR WITH F_r ALGORITHM

Class	ARIMA	F_r	improvement
All products	30512	22093	26.7%
quantity < 500	24338	17248	29.1%
quantity ≥ 500	6174	4845	21.5%
(quantity = 0) in $< 1/3$ time	19178	15942	16.9%
(quantity = 0) in $\geq 1/3$ time	11334	6151	45.7%
avg(π_r) < 20	26756	18711	30.1%
avg(π_r) ≥ 20	3756	3382	10.0%
top 100 items	15805	6131	61.2%
least 470 items	16167	15962	1.2%
PMD	10841	8612	$20,6\%$

VI. CONCLUSION AND FUTURE WORK

The broad range of products with its hidden periodicity made the analysis difficult. The low volume sales further complicated the analysis of the influence of the price on the sales quantities. The conclusions drawn from the above results can be reduced to the following three statements:

1) Data profiling is crucial for choosing the best time series model
2) Low sales volume can hide a cyclic sales behavior and the price should be treated as input parameter
3) Simple models for sales forecasting based on causal parameters can outperform some sophisticated stochastic models.

Cross influence from other products should be further investigated. This could lead to clusters of products that behave coherently. Sometimes products are complementary. This has not been investigated. The overall sales had a significant periodicity length of 7, the available data covered only 56 days. Hence, a longer time history would be needed to verify this and to look for overlaying periodicities. A spectral analysis on a individual product level could further improve the prediction accuracy. For products with a strong monotone price development our approach to look for similar prices is not well suited. The price trend should be computed instead.

The F_r algorithm can be used with incomplete time series. This is particularly useful for real-time analysis used in recommender systems and should be further investigated.

REFERENCES

[1] August-Wilhelm Scheer, *Absatzprognosen* [engl. Sales Forecasting], Springer Verlag, Berlin, 1983

[2] Manfred Hüttner, *Markt- und Absatzprognosen* [engl. Market and Sales Forecasting], Kohlhammer, Stuttgart, 1982

[3] Yang Lan and Daniel Deagu, ”A New Approach and Its Applications for Time Series Analysis and Prediction Based on Moving Average of n^{th}-Order Difference”, in: Dawn E. Holmes and Lakhmi C. Jain (Eds.), *Data Mining: Foundations and Intelligent Paradigms*, Vol 2: Statistical, Bayesian, Time Series and other Theoretical Aspects, pp. 157 - 182, Springer Berlin Heidelberg, 2012

[4] Klaus Neusser, *Zeitreihenanalyse in den Wirtschaftswissenschaften* [engl. Time Series Analysis in Economic Sciences], B. G. Teubner Verlag, Wiesbaden, 2006

[5] Alan Julian Izenman, *Modern Multivariate Statistical Techniques - Regression, Classification, and Manifold Learning*, Springer Science + Business Media, New York, 2008

[6] Helmut Lütkepohl, *New Introduction to Multiple Time Series Analysis*, corr. repr., Springer Verlag, Berlin Heidelberg, 2007

[7] N. N., DATA MINING CUP 2012, prudsys AG, Zwickauer Str. 16, D-09113 Chemnitz, [Online] http://www.data-mining-cup.de/en/review/dmc-2012, last access: 13.01.2013

[8] Sam Oches, ”Top 50 Sorted by Total Units”, Special Report of QSR Magazine, Journalistic Inc., August 2011, [Online] http://www.qsrmagazine.com/reports/top-50-sorted-total-units

[9] Economist Intelligence Unit, ”Denmark: Market Indicators and Forecasts”, [Online] http://datamarket.com/data/set/1wmo/ (indicators: Private consumption, Consumer goods), last access: 30.08.2012

[10] Joachim Griese, *Adaptive Verfahren im betrieblichen Entscheidungsprozess* [engl. Adaptive Methods in Operational Decision-Making], Physica Verlag, Würzburg - Wien, 1972

[11] Georg E. P. Box and Gwilym M. Jenkins, *Time Series Analysis: Forecasting and Control*, 1^{st} rev. ed., Holden Day,Oakland, San Francisco, 1976

[12] Christopher A. Sims, ”Macroeconomics and Reality”, in: Econometrica, Vol. 48, No. 1, pp. 1 - 48, 1980

[13] Robert E. Lucas, ”Econometric policy evaluation: A critique”, In: K. Brunner and A. H. Meltzer (Eds), *The Phillips Curve and Labor Markets*, Vol. 1, Carnegie-Rochester Conference Series on Public Policy, pp. 19 - 46, Amsterdam, North-Holland, 1976

[14] Peter J. Brockwell and Richard A. Davis, *Time Series: Theory and Methods*, 2^{nd} Edition, Springer Science + Business Media, New York, 2006

[15] Roger A. Arnold, *Economics*, 9^{th} ed., South-Western College Publ., 2008

[16] Jack G. A. J. van der Vorst, Andrie J. M. Beulens, W. de Wit, Paul van Beek, ”Supply chain management in food chains: Improving performance by reducing uncertainty”, in: International Transactions in Operational Research, Vol. 5(6), pp 487 - 499, 1998

[17] Philip Doganis, Alex Alexandridis, Ranagiotis Patrinos, and Haralambos Sarimveis, ”Time series sales forecasting for short shelf-life food products based on artificial neural networks and evolutionary computing”, Journal of Food Engineering 75, pp. 196 - 204, Elsevier Ltd., 2006, [Online] http://www.elsevier.com/locate/jfoodeng, last access: 30.08.2012

[18] Clive W. J. Granger and Paul Newbold, ”Economic Forecasting: The Atheist’s Viewpoint”, in: G.A. Renton (ed.), *Modelling the Economy*, pp. 131 - 148, Heinemann, London, 1975

[19] Horst Rinne and Katja Specht, *Zeitreihen - Statistische Modellierung, Schätzung und Prognose* [engl. Time Series - Statistical Modelling, Estimation and Prediction], Verlag Vahlen, München, 2002

[20] Alfred Marshall, *Principles of Economics*, 8^{th} ed., Cosimo Classics, 2009, first publ. 1890

Enhancing Distributed Data Mining performance by multi-agent systems

María del Pilar Angeles, Francisco Javier García-Ugalde
Facultad de Ingeniería
Universidad Nacional Autónoma de México
México, D.F.
pilarang@unam.mx, fgarciau@unam.mx

Jonathan Córdoba-Luna
Posgrado en Ciencia e Ingeniería de la Computación
Universidad Nacional Autónoma de México
México, D.F.
jel_154@comunidad.unam.mx

***Abstract*— This research work presents a Multi-Agent Distributed Data Mining framework and its implementation on a prototype in order to improve performance on the data mining process and maintain the underlying information systems security.**

Keywords- Distributed Data Mining; Multi-Agent System; Multi Agent Data Mining; Agent BasedDistributed Data Mining

I. INTRODUCTION

Data mining (DM) is focused on identifying patterns and trends from massive data integrated within a data warehouse. However, a single data mining technique has not been proven appropriate for every domain and data set [1]. Data mining is a computationally intensive process involving very large datasets, affecting the overall performance and data confidentiality because data might change rapidly and is located at different sites. Distributed Data mining (DDM) has emerged as an approach to performance and security issues because DDM mines data sources regardless of their physical locations, avoiding the transference across the network of very large volumes of data and the security issues occasioned from network transferences. Multi-agent Systems (MAS) are a collection of software entities (agents) that are intended to cooperate to undertake some processing task [2]. Therefore, MAS has revealed opportunities to improve distributed data mining systems in a number of ways [1]. This approach is also known as Multi-Agent Data Mining (MADM).

The present paper is organized as follows: The next section is focused on previous work on data mining and its role within de process of knowledge discovering databases (KDD), the most representative data mining tasks and components. The third section details cluster analysis by describing the K-Means and the agglomerative hierarchical algorithms, besides a set of criteria to assess the algorithms performance. The forth section describes a multi-agent based system architecture and how is mainly implemented.

The fifth section presents the implemented framework describing the multi-agents, the scope and limitations of the agents. The sixth section shows the experimentation plan and the four scenarios considered. The seventh section analyses the experiment results and the last section concludes the main topics achieved and the future work to be done.

II. RELATED WORK

The present section is aimed to briefly describe the related work on data mining.

A. Data Mining

According to Han and Kamber in [3], data mining is related to the extraction or mining of knowledge from very large data sources.

Witten and Frank in [4] relate data mining as the process of data pattern discovery. The process has to be automatic or semi-automatic.

The discovered patterns must be meaningful enough to provide a competitive advantage, mainly in terms of business. However, Hand in [5] proposed data mining as a complex data set analysis aimed to discover unsuspected data interrelations in order to summarize or classify data in different and understandable forms that should be useful to the data owner.

For Sumathi and Sivanandam in [6] data mining is related to the process of discovery of new and significant correlations, patterns and tendencies mined from very large data sources by using statistics, machine learning, artificial intelligence and data visualization techniques. According to the evolution of the techniques implemented for data mining, we consider data mining as the process of extraction of new and useful information from very large data sources by considering a number of multidisciplinary technics, such as statistics, artificial intelligence and data visualization aimed to make informed decisions that provide business advantage.

The Process of Knowledge Discovery (KDD) is a set of processes focused in discovering knowledge within databases, while data mining is the application of a number of artificial intelligence, machine learning and statistics techniques to data. Furthermore, data mining is one of the most important processes within KDD

The following Section is focused on the data mining process.

B. The Process of Data Mining

The process of data mining is focused in two main objectives: prediction and description. The main goals within a knowledge discovery project shall be already determined and they will determine if descriptive or predictive models would be applied.

The availability of an expert or supervisor would determine the type of learning (supervised or unsupervised) that will apply during the data mining process. The predictive model learns under the control of a supervisor or expert (supervised learning) who determines the desired answer from the data mining system [6], whereas the descriptive model execute clustering and association rules tasks to discover knowledge by unsupervised learning, in other words, with no external influence that establish any desired behavior within the system [6].

The next task within data mining shall be the identification of methods and their corresponding algorithms. The study of past data behavior thorough the implementation of algorithms for classification, clustering, regression analysis, or any other method allows building a model that describes and distinguishes data within classes or concepts.

Classification is used mostly as a supervised learning method, whereas clustering is commonly used for unsupervised learning (some clustering models are for both). The goal of clustering is descriptive; that of classification is predictive [9].

As our proposal will be implemented with no external supervision, Section III is aimed to briefly explain only the implemented algorithms and metrics involved in our clustering analysis.

III. Cluster Analysis

The term cluster analysis encompasses a number of different algorithms and methods for grouping objects of similar kind into respective categories. Such algorithms or methods are concerned with organizing observed data into meaningful structures. In other words, cluster analysis is an exploratory data analysis tool which aims at sorting different objects into groups in a way that the degree of association between two objects is maximal if they belong to the same group and minimal otherwise. Given the above, cluster analysis can be used to discover structures in data without providing an explanation/interpretation.

There are a number of classifications of clustering algorithms; this research takes a basic but practical classification that allows organizing the existing algorithms. Such algorithms are divided into two categories: Partition based algorithms and hierarchical algorithms.

A. Partition based clustering algorithms

Given a data set with *n* data objects to identify *k* data partitions, where each partition represents a cluster and $k \leq n$. There is a good partitioning if the objects within a cluster are close to each other (cohesion), or they actually are related to each other, and at the same time they are far from the objects that belong to other cluster. This Section will explain the partition based clustering k-means algorithm [10].

The k-means algorithm represents each cluster by the mean value of the data objects in the cluster.

Given an initial set of *k* means (centroids) $m_1^{(1)},\dots,m_k^{(1)}$, the algorithm proceeds by alternating between two steps:

1. Assignment step : Assign each observation to the cluster with the closest mean.
2. Update step: Calculate the new means to be the centroid of the observations in the cluster.
3. The algorithm is deemed to have converged when the assignments no longer change.

K-means is a classical partitioning technique of clustering that clusters the data set of *n* objects into *k* clusters with *k* known a priori.

Many clustering methods use distance measures to determine the similarity or dissimilarity between any pair of objects. It is useful to denote the distance between two instances xi and xj as: d(xi,xj). A valid distance measure should be symmetric and obtains its minimum value (usually zero) in case of identical vectors. This section describes three distance measure for numeric attributes: Minkowski, Euclidean and Manhattan. The distance between two data instances can be calculated using the Minkowski metric (Han and Kamber, 2001):

d(xi, x j) = (|xi1−x j1|g+|xi2−x j2|g+. . .+ |xip−x jp|g)1/g

The commonly used Euclidean distance between two objects is achieved when g = 2. Given g = 1, the sum of absolute paraxial distances (Manhattan metric) is obtained.

B. Hierarchical clustering algorithms

These algorithms consist of joining two most similar data objects, merge them into a new super data object and repeats until all merged. There is a graphical data representation by a tree structure named dendrogram to illustrate the arrangement of the clusters produced by hierarchical clustering. There are two ways of creating the graphic, the agglomerative algorithm or divisive algorithm [11]. Agglomerative hierarchical clustering is a bottom-up clustering method where clusters have sub-clusters, which in turn have sub-clusters, etc.

The key operation of agglomerative hierarchical clustering algorithm is the computation of the proximity between two clusters. However, cluster proximity is typically defined with a particular type of cluster. The cluster proximity in this section will refer to the single link, complete link and group average respectively.

For the single link, the proximity of two clusters A, B is defined as the minimum of the distance (maximum of the similarity) between any two points x, y in the two different clusters. For the complete link, the proximity of two clusters A, B is defined as the maximum of the distance (minimum of the similarity) between any two points x,y in the two different clusters. For the group average, the proximity of two clusters Cx and Cy are of size Sx and Sy, respectively, is expressed as the average pairwise proximity among all pairs of points in the different clusters.

C. Clustering Evaluation

In most cases, a clustering algorithm is evaluated using a) some internal evaluation measure like cohesion, separation, or the silhouette coefficient (addressing both, cohesion and separation), b) some external evaluation measure like accuracy, precision. In some cases, where evaluation based on class labels does not seem viable, c) careful (manual) inspection of clusters shows them to be a somehow meaningful collection of apparently somehow related objects [12].

There are a number of important issues for cluster validation, such as the cluster tendency of a set of data, the correct number of clusters, whereas the cluster fit the data without reference to external information or not, and determining which cluster is better [13]. The first three issues do not need any external information.

The evaluation measures are classified into unsupervised, supervised and relative. We have implemented the unsupervised evaluation.

Unsupervised validation: In the case of cluster cohesion is concerned to how closely relate the objects in a cluster are. In the case of cluster separation is aimed to determine how distinct a cluster is from other clusters, these internal indices use only information from the data set [13].

Cluster Cohesion: Measures how closely related are objects in a cluster. Cohesion can be defined as the sum of the proximities to the cluster centroid or medoid.

Cluster Separation: Measures how distinct or well-separated a cluster is from other clusters. Therefore,
Separation is measured by the sum of the weights of the links from points in one cluster to points in the other cluster.

Given a similarity matrix for a data set and the cluster labels from a cluster analysis, it is possible to compare this similarity matrix against an ideal similarity matrix on the basis of cluster labels. An ideal cluster is one whose points have a similarity of 1 to all points in the cluster and a similarity of 0 to all points in other clusters.

In the case of unsupervised evaluation of hierarchical based clustering algorithms, we discuss the cophenetic correlation.

In the agglomerative hierarchical clustering process, the smallest distance between two clusters is assigned, and then all points in one cluster will have the same value as a cophenetic distance with respect to the points in other cluster. In a cophenetic distance matrix, the entries are the cophenetic distances between each pair of objects.

If any of single link clustering, complete link or group average is applied, the cophenetic distances for each point can be expressed in cophenetic distance matrix. Thus, the cophenetic correlation coefficient is the correlation between the entries of this matrix and the original dissimilarity matrix and is a standard measure of how well a hierarchical clustering fits the data.

IV. Introduction to Multi-Agent Systems

An agent is a computer system that is capable of autonomous action on behalf of its user or owner. An agent is capable to figure out what it is required to be done, rather than just been told what to do [15].

An intelligent agent must be reactive, pro-active, and social. A reactive agent maintains an ongoing interaction with its environment, and responds in time to changes that occur in it. A proactive agent attempts to achieve goals, not only driven by events, but also taking the initiative. However, at the same time a social agent takes into account the environment, in other words, some goals can only be achieved by interacting with others. The social ability in agents is the ability to interact with other agents (and possibly humans) via cooperation, coordination, and negotiation. Agents have the ability to communicate, to cooperate by working together as a time to achieve a shared goal. Agents have the ability to coordinate different activities. Agents shall negotiate to reach agreements taking into consideration the environment in order to react, to negotiate, to coordinate, etc. The environments are divided in accessible, inaccessible, deterministic, non-deterministic, episodic, static and dynamic.

A multi-agent system is one that consists of a number of agents, which interact with one-another.

In the 1990s, Bailey proposed in [16] a multi-agent clustering system to achieve the integration and knowledge discovered from different sites with a minimum amount of network communication and maximum amount of local computation by a distributed clustering system where data and results can be moved between agents. There was proposed a distributed density based clustering algorithm the Peer to Peer model in [17]

These previous approaches were aimed to improve security by a distributed data mining. However, there were no measurements of general performances by considering distributed agents against centralized clustering techniques within a data warehouse.

V. Multi Agent System for Distributed Data Mining Framework

The present research proposes a framework for implementing a Multi-agent Distributed Data mining, which is based on [2] and extended by additional agents such as performance, validating and coordinating agents in order to address performance and security issues within the disparate information systems that conform the distributed data mining system. The involved agents are:

a) A user agent is responsible for the interaction between end-users and the coordinating agents in order to accomplish the assigned tasks.

b) Coordinating agent is focused on the correct message transmission among the agents within the network. It takes the user requirements and sends them to the corresponding agent.

c) Coordinating Algorithm Agent is focused on the interaction between clustering agents. This agent receives the processed information from the clustering agents and executes the algorithm globally in order to guarantee a better clustering quality.

d) Clustering agent is concerned with a clustering algorithm. Once the clustering agents have done their task, they send local processed information to the algorithm coordinator agent. The clustering algorithms are the most commonly used and keep the same structure utilized within a centralized approach but they can be sent to other sites where is required to perform clustering avoiding data transference in order to enhance performance and enforce security.

e) Data agent is in charge of a data source; it interacts and allows data access. There is one data agent per data source.

f) Validation agent is responsible for the quality assessment of the clustering results. There is validation agent per a measuring technique of a given cluster configuration. These agents consider either cluster cohesion or cluster separation. In the case of the hierarchical clustering, the cophenetic distance is utilized to measure the proximity within the hierarchical agglomerative clustering algorithm. This distance helps to determine the precision. Therefore is required to compute the similarity matrix and the cophenetic matrix. The cophenetic distance can be seen as a correlation between the distance matrix and the cophenetic matrix. If the computed value is close to 100%, the quality of clustering is enough.

g) Performance agent is focused on the measurement of operating system resources in order to obtain the overall performance of the processing algorithms in terms of data transmission, data access and data process as follows:

Memory used: physical memory consumed by the algorithm when it has been executed. The resulting value is given in megabytes (MB).

Elapsed Processing Time: the amount of time the algorithm took to process. The resulting value is given in nanoseconds (ns).

Amount of data transmitted: A quantity in MB which determines the total size of all data processed and transferred.

PC-LAN Broadband: Amount of information that can be sent over a network connection in a given period of time. The bandwidth is usually given in bits per second (bps), kilobits per second (kbps) or megabits per second (mps).

Elapsed response time: Time interval from which the request is made by the user until the result set is presented.

Transmission-time: time of the node-to-node data transfer.

Total Response Time: The total result of the processing time + transmission time + response time.

Physical reads: total number of data blocks read from disk.

Logical reads: total number of data blocks read from the main memory (RAM/cache).

All these measures are stored within a table as a log from which the data agent can access and inform the performance agent. Therefore, when a user request is submitted, it will be evaluated according to the historical information stored in the log, and an execution strategy will be developed. If the amount of data to be processed is small, the performance agent will establish a "low status", thus the creation of a single clustering agent to perform clustering analysis shall be enough. If the amount of data is considerably high, the performance agent establishes a "medium status", in order to create two agents to process the data and obtain the clustering analysis. If the amount of data is very large, the performance agent establishes a "high status", in order to create three clustering agents for clustering analysis. This status is sent to the coordinating agent, which is responsible for building the agents requested. In order to improve the clustering results and the performance of data mining across the distributed system, there has been implemented negotiation among agents by a communication protocol. For instance, considering the amount of data to analyze, there is a negotiation of which clustering method is the best by asking each clustering agent if it is able to perform the task according to the resources of the site where that agent resides.

The framework proposes an agent performance which according to the status established from negotiation and statistics; it is able to determine the strategy to implement the algorithms through clustering agents running on parallel.

Fig. 1 shows de Multi-Agent System for Distributed Data Mining Framework.

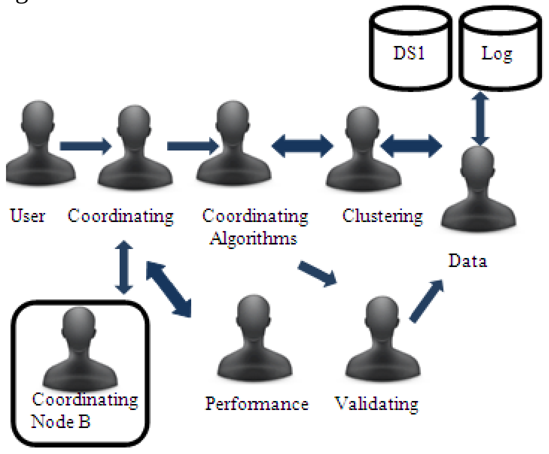


Figure 1. Multi-Agent System for Distributed Data Mining

VI. Implemented Framework

The present work proposes the implementation of the Multi-Agent System for Distributed Data Mining framework described previous section by the development of a web platform through Agent-Oriented Programming paradigm (AOP).

We have developed such framework with Java Agent DEvelopment (JADE) [18], which integrates a library called "jade. gateway" for the agent programming within a web interface. JADE is compliant to the Foundation for Intelligent, Physical Agents (FIPA) [19]. FIPA specifications represent the most important standardization activity conducted in the field of agent technology. JADE is composed by a native Agent Communication Language (ACL), which incorporates an Agent Manager System (AMS) and a Directory Facilitator (DF). The Agent Communication Language may be modified according to system requirements. Message Transport Service (MTS) is a service provided to transport FIPA-ACL messages between agents in any given agent platform and between agents on different agent platforms. The Agent Management System is responsible for managing the operation of an agent platform, such as the creation, deletion, status, overseeing and migration of agents. The Directory Facilitator provides yellow pages services to other agents, maintaining a list of agents and providing the most current information about agents in its directory to all authorized agents.

In order to implement negotiation among agents, we have utilized a number of communicative acts and protocols for effective communication of agents:

OneShotBehaviour: This type of behaviour is executed only once and with no interruption.

CyclicBehaviour: Represents a behavior that should be executed a number of times.

CompositeBehaviour: Behavior based on the composition of other behaviours or sub-behaviours, the implementation of the framework proposed contains the following CompositeBehaviour subclasses:

SequentialBehaviour: executes a series of sub-behaviours sequentially, and is considered finished when all its sub-behaviours have been completed .

ParallelBehaviour: executes a series of behaviors concurrently and ends when a certain condition is met upon completion of the sub-behaviours:

The following communication protocols have been implemented:

FIPA-Request: Allows an agent to request another agent to perform an action. The messages exchanged are:
"Request" followed by the request, "Agree", if the request is accepted, "Refuse" in case the request is rejected. "Failure", if an error occurred in the process, "Inform", to communicate the results.

FIPA-Query: Allows an agent to request another agent an object by a "Query-ref()" message or a comparison value by an if() message, depending on what type of request it will be a query-if (test of truth). The messages exchanged are: "Agree", Refuse", "Failure" and "Inform".

The class ContractNet implements a protocol behaviour where a initiator sends a proposal to several responders and select the best proposal. The messages exchanged are: CFP (Call For Proposal) in order to specify the action to perform. Therefore, the responders may send a "Refuse" to deny the request, a "Not-Understood" if there was a failure in communication, or "Propose" to make a proposal to the originator. The initiator evaluates the proposals received and sends "Reject-Proposal" or "Accept-Proposal. Responders whose proposal was accepted send a "Failure" if something went wrong, an "Inform-Done" if the action was successful or an "Inform-Result" with the results of the action if appropriate.

The web application architecture is as follows:

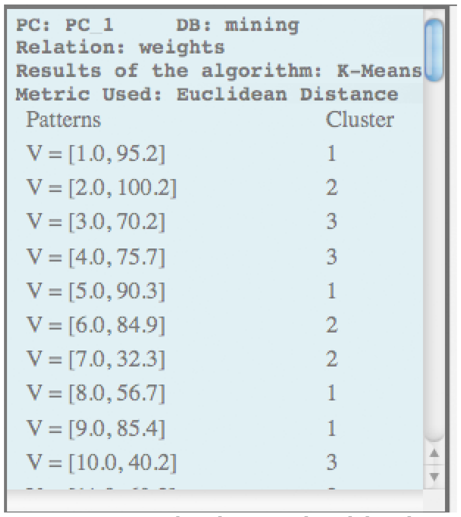
a) The Web interface allows users to interact with the Multi-Agent System through a web browser by sending request of data mining tasks and receiving the corresponding results.

b) Data repositories, which consist of file folders or PostgreSQL databases.

c) Clustering Repository with all the clustering and validation algorithms.

d) The System engine for the involved agent management, data preprocessing, connection to the Database Management Systems (DBMS), and sites communication languages.

The web interface calls the user agent, which in turn allows users the specification of the node, the data source from which the clustering is required. User agent asks the data agent to connect to the distributed database system and to retrieve information from a specific database table or file within a remote or local site. Once obtained the node, the database and table the data mining system requires the specification of the clustering algorithm, the K number of clusters and the metric. Fig. 2 corresponds to the results K-means algorithm with 5 clusters and the metric Euclidean distance.



```
PC: PC_1     DB: mining
Relation: weights
Results of the algorithm: K-Means
Metric Used: Euclidean Distance
 Patterns                      Cluster
 V = [1.0, 95.2]               1
 V = [2.0, 100.2]              2
 V = [3.0, 70.2]               3
 V = [4.0, 75.7]               3
 V = [5.0, 90.3]               1
 V = [6.0, 84.9]               2
 V = [7.0, 32.3]               2
 V = [8.0, 56.7]               1
 V = [9.0, 85.4]               1
 V = [10.0, 40.2]              3
```

Figure 2. K-means with 5 clusters and Euclidian distance

VII. Experiments and Results

In order to assess the framework proposed in Section V, we have identified a set of experiments according to the following scenarios:

a) Centralized Data Scenario: A typical data mining system, composed by a centralized data mining process with no multi-agents.

b) Multi-agent Centralized Data Scenario: A Multi-agent centralized data mining system.

c) Distributed Scenario: A Distributed data mining system with no multi-agents.

d) Multi-agent distributed data mining Scenario: A Distributed data mining system with multi-agents.

The identified independent variables are: a) clustering methods; b) metrics; c) number of clusters; d) data sources

The identified dependent variables are: a) data access time; b) data transmission time; and c) processing time.

For each scenario a set of 9 data sources have been processed, the corresponding results are presented as follows:

a) Centralized data scenario

Table 1 presents the results obtained from processing 9 data sources by the k-means algorithm, considering no agents, 10 clusters and a transfer rate of 500 kb/s. For instance, the process of mining a table called agency with 35000 rows takes 7.83E+09 nanoseconds, and 7.11 Mb of memory used.

TABLE 1. CENTRALIZED, K-MEANS, 10 CLUSTERS SCENARIO

Table name	Rows	Data Transfer (Mb)	Data Transfer Time (ns)	Memory Used (Mb)	Processing Time (ns)
agency	**35000**	**0.200272**	**3.13E+08**	**7.11**	**7.83E+09**
school	500	0.003893	6.08E+07	1.22	3.08E+08
supermarket	150	0.001001	1.56E+07	1.10	3.06E+08
weights	70	0.000476	7.44E+06	0.76	2.77E+08
substance	800	0.003338	5.22E+07	1.31	4.36E+08
articles	500	0.002538	3.97E+07	1.22	3.56E+08
survey	300	0.005728	8.95E+07	1.51	3.13E+08
population	300	0.002251	3.52E+07	1.15	2.87E+08
school_age	1200	0.008817	1.38E+08	1.45	5.44E+08

Table 2 presents the results obtained from processing 9 data sources by the hierarchical algorithm, considering no agents and 10 clusters. In the case of table Agency, there were memory problems from the JVM. Therefore, the maximum number of rows processed by this algorithm was of 1600 tuples, which in turn the corresponding processing time was of 1.12E+10 nanoseconds.

TABLE 2. CENTRALIZED, HIERARCHICAL, 10 CLUSTERS SINGLE LINK SCENARIO

TableName	Rows	Processing Time
agency	**35000 (1600)**	**1.12E+10**
school	500	7.15E+08
supermarket	150	3.89E+08
weights	70	2.33E+08
substance	800	1.69E+09
articles	500	6.80E+08
survey	300	4.33E+08
population	300	4.31E+08
school_age	1200	4.28E+09

b) Multiagent centralized data

Table 3 presents the results obtained from processing 9 data sources by the k-means algorithm, considering multi-agents and 10 clusters. For instance, the process of mining a table called agency with 35000 rows takes 7790887000 nanoseconds.

TABLE 3. MULTI-AGENT, CENTRALIZED, K-MEANS, 10 CLUSTERS SCENARIO

TableName	Rows	Processing Time
agency	**35000**	**7.79E+09**
school	500	2.74E+08
supermarket	150	2.71E+08
weights	70	2.43E+08
substance	800	4.02E+08
articles	500	3.21E+08
survey	300	2.79E+08
population	300	2.53E+08
school_age	1200	5.10E+08

Table 4 presents the results obtained from processing 9 data sources by the hierarchical algorithm, considering no agents and 10 clusters. In the case of table Agency, there were memory problems from the JVM. Therefore, the maximum number of rows processed by this algorithm was of 1600 tuples, which in turn the corresponding processing time was of 11118830000 nanoseconds.

TABLE 4. MULTI-AGENT, CENTRALIZED, HIERARCHICAL, SINGLE LINK, 10 CLUSTERS SCENARIO

TableName	Rows	Processing Time
agency	**35000 (1600)**	**1.11E+10**
school	500	6.81E+08
supermarket	150	3.55E+08
weights	70	1.99E+08
substance	800	1.66E+09
articles	500	6.46E+08
survey	300	3.99E+08
population	300	3.97E+08
school_age	1200	4.25E+09

c) Distribuited dta scenario

Table 5 presents the results obtained from processing the Agency table distributed on two partitions stored on node A and node B. The Agency table was processed by the k-means and hierarchical algorithms, with no consideration of agents. For instance, the process of mining 36000 rows by the k-means algorithm takes 775756400 nanoseconds agency, whereas processing only 1600 rows from the same table by hierarchical algorithm takes 11116402300 nanoseconds.

TABLE 5. DISTRIBUTED AGENCY TABLE ON TWO PARTITIONS, NO AGENTS SCENARIO

Data rows Node A	Data rows Node B	Algorithm	Total Processing Time
18000	18000	kMeans	7.76E+08
800	800	Hierarchical	1.11E+10

d) Multi-agent distributed data mining scenario

Table 6 presents the results obtained from processing the Agency table distributed on two partitions stored on Node1 and Node2. The Agency table was processed by the k-means and hierarchical algorithms, with multi-agents. For instance, the process of mining 36000 rows by the k-means algorithm takes 748213000 nanoseconds agency, whereas processing only 1600 rows from the same table by hierarchical algorithm takes 11085513000 nanoseconds.

TABLE 6. MULTI-AGENT, DISTRIBUTED AGENCY TABLE, 2 PARTITIONS

Data rows Node1	Data rows Node 2	Algorithm	Total Time Processing
18000	18000	kMeans	7.48E+08
800	800	Hierarchical	1.11E+10

Table 7 presents the results obtained from processing a set of 9 data sources with multi-agents, distributed environment and clustering algorithm k-means. Comparing this table with Table 1, we can conclude that the amount of memory used in multi-agent, distributed environment was less than the memory required for the no-agent, centralized environment in all cases.

TABLE 7. MULTI-AGENT, DISTRIBUTED, K-MEANS

Relation	Number of Rows	Memory Used Agent 1	Memory Used Agent 2	Memory Used Agent 3	Memory Used Total
agency	35000	2.33	2.33	2.33	6.99
school	500	0.36	0.36	0.36	1.08
supermarket	150	0.33	0.33	0.33	0.99
weights	70	0.21	0.21	0.21	0.63
substance	800	0.40	0.40	0.40	1.20
articles	500	0.36	0.36	0.36	1.08
survey	300	0.34	0.34	0.34	1.02
population	300	0.34	0.34	0.34	1.02
School_age	1200	0.44	0.44	0.44	1.32

e) Analysis of Results

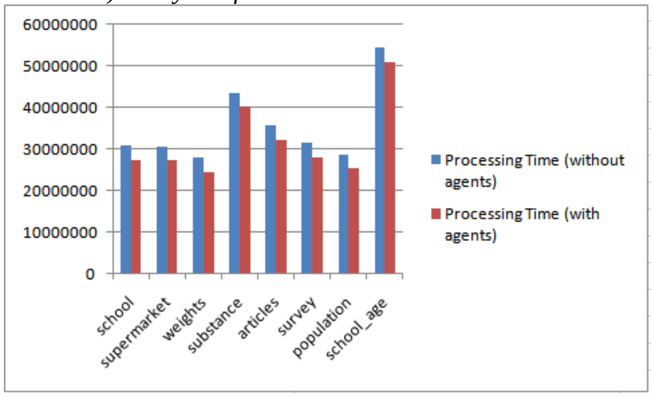


Figure 3. Centralized no agents vs. multi-agents with k-means algorithm

Fig. 3 shows a slight advantage in the use of multi-agent systems to process data with the K-means algorithm, the K value represents the number of clusters. Processing the data partitions with multi-agents, and merge the results, allows faster data processing. If the amount of data is significantly large, data can be shared among n agents, reducing response time. However, a disadvantage could be that by sharing data between n agents the quality of the clusters may decrease.

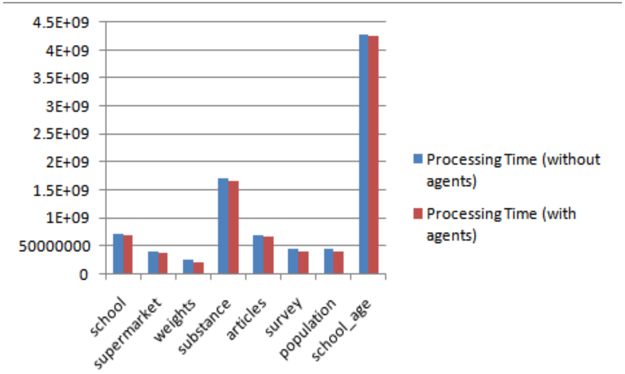


Figure 4. Centralized no agents vs. multi-agents with hierarchical algorithm

Fig. 4 shows a slight advantage in the use of multi-agent systems to process data with the hierarchical algorithm. In the case of large data sets, this algorithm might be a good strategy since the distance matrix has to be calculated and multi-agents offer a slightly better data processing. In this implementation we have used a single link clustering criteria. However, using a different technique might affect the processing because other criteria require an average of the data in clusters.

We can conclude that agents reduce response time by partitioning data into n subsets. As data grows, a better strategy might be distributing workload among agents. If the amount of memory is a limit, data shall be partitioned between few agents, because each agent runs on its own thread generating a significant overhead. Otherwise, a larger number of agents and parallel processing is recommended. Furthermore, negotiation and parallelization of agents is an alternative for hierarchical algorithms.

Regarding the centralized and distributed scenarios, there is a significant advantage in the use of agents, since the design of agents is intended for distributed systems.

Considering the distributed multi-agent scenario, where all the existing nodes process data locally and send a result which can be wrapped by another agent, allows a significant data processing optimization. Considering the distributed no-agents scenario we have utilized RMI (Java Remote Method Invocation) for remote methods invocation. This offers the advantage of exporting java objects. However, is not fast enough on distributed tasks, compared to a fully distributed tool as Jade

VIII. CONCLUSION AND FUTURE WORK

Nowadays, organizations that operate at global level from geographically distributed data sources require distributed data mining for a cohesive and integrated knowledge. Such organizations are characterized by end users localized geographically separated from the data sources. The MDD is a relatively new research field, so a considerable number of research problems lie, relatively unaddressed.

We have proposed a Multi-Agent Distributed Data Mining System in order to improve data mining performance and data security considering negotiation and a metadata for further information and better decision regarding how many and agents and where they are required.

Nowadays k-means and agglomerative hierarchical clustering algorithms with their corresponding metrics such as Euclidean distance, Minkowski distance, Manhattan distance and single link are utilized. However, the present implementation could be improved by incorporating new algorithms.

According to the results of the experiments we can conclude that there is a better performance in terms of response time, and processing distribution comparing with no agents or centralized environments.

The process of clustering can lose precision when data is partitioned and processed locally; the coordinating algorithm agent merges only the results into a single cluster in the case of hierarchical clustering algorithm. However, there is a better performance and cutbacks in memory space used. There has to be further experiments and analysis to achieve a better balance between the number of desired clusters, the memory resources and response time.

Regarding the information stored within the log, the present implementation utilizes tables containing numerical data; the creation of further agents in order to transform data into numerical ratings would be an improvement as part of future work.

REFERENCES

[1] Vuda S.R.: Multi agent-based distributed data mining, an overview, International Journal of Reviews in Computing, ISSN: 2076-3328, E-ISSN: 2076-3336, pp 83-92.

[2] CHAIMONTREE, S., ATKINSON, K., COENEN, F. (2012) .A Multi-Agent Approach To Clustering: Harnessing The Power of Agents. Springer.

[3] Jiawei Han and Micheline Kamber, Data Mining: Concepts and Techniques, 2nd ed.The Morgan Kaufmann Series in Data Management Systems, Jim Gray, Series Editor Morgan Kaufmann Publishers, 2006. ISBN 1-55860-901-6

[4] Frank, E., Paynter, G. W., Witten, I. H., Gutwin, C., and Nevill-Manning, C. G. (1999), Domain-specific keyphrase extraction. In Proceedings of the 16th International Joint Conference on Artificial Intelligence, pages 668–673. Morgan Kaufmann.

[5] Han, J. and Kamber, M. Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers, 2001.

[6] S.Sumathi,S.N. Sivavavdam Introduction to Data Mining and its applications. Editor in Chief Janusz K.Studies in Computational Intelligence, Springer verlag, 2006.

[7] Adriaans, P., and Zantinge, D., Data Mining. Addison-Wesley, New York, 1996.

[8] The Datamining and Knowledge Discovery Handbook http://www.cse.hcmut.edu.vn/~chauvtn/data_mining/Texts/%5B9%5D%202010%20Data%20Mining%20and%20Knowledge%20Discovery%20Handbook.pdf, (retrieved: January,2013).

[9] Veyssieres, M.P. and Plant, R.E. Identification of vegetation state and transition domains in California’s hardwood rangelands. University of California, 1998

[10] . Chaimontree, S., Atkinson, K., Coenen, F.: Multi-Agent Based Clustering: Towards Generic Multi-Agent Data Mining. In: Perner, P. (ed.) ICDM 2010. LNCS, vol. 6171, pp. 115–127. Springer, Heidelberg (2010)

[11] International Journal of Image and Data Fusion,Volume 3, Issue 3, 2012, Special Issue: Image Information Mining for EO Applications, Hierarchical data representation structures for interactive image information mining, DOI: 10.1080/19479832.2012.697924, Lionel Gueguen & Georgios K. Ouzounis, pages 221-241.

[12] Evaluation of Multiple Clustering Solutions Hans-Peter Kriegel, Erich Schubert, Arthur Zimek, Ludwig-Maximilians-Universität München, Oettingenstr. 67, 80538 München, Germany,http://www.dbs.ifi.lmu.de{kriegel,schube,zimek}@dbs.ifi.lmu.de, (retrieved: January,2013).

[13] Tan, Steinbach Kumar Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, Vipin Kumar,Addison-Wesley Companion Book Site.

[14] Chaimontree, S., Atkinson, K., Coenen, F.: Clustering in a Multi-Agent Data Mining Environment. In: Cao, L., Bazzan, A.L.C., Gorodetsky, V., Mitkas, P.A., Weiss, G., Yu, P.S. (eds.) ADMI 2010. LNCS, vol. 5980, pp. 103–114. Springer, Heidelberg (2010).

[15] An Introduction to MultiAgent Systems - Second Edition,by Michael Wooldridge, Published May 2009,by John Wiley & Sons,ISBN-10: 0470519460,ISBN-13: 978-0470519462http://www.csc.liv.ac.uk/~mjw/pubs/imas/distrib/pdf-index.html, (retrieved: January,2013).

[16] . Bailey, S., Grossman, R., Sivakumar, H., Turinsky, A.: Papyrus: A system for data mining over local and wide area clusters and super-clusters. IEEE Supercomputing (1999)

[17] Klusch, M., Lodi, S., Moro, G.: Agent-Based Distributed Data Mining: The KDEC Scheme. In: Klusch, M., Bergamaschi, S., Edwards, P., Petta, P. (eds.) Intelligent Information Agents. LNCS (LNAI), vol. 2586, pp. 104–122. Springer, Heidelberg (2003).

[18] Bellifemine, F., Bergenti, F., Caire, G., Poggi, A.: JADE: a java agent development framework. In: Bordini, R.H. (ed.) Multi-agent Programming: Languages,Platforms, and Applications, p. 295. Springer, New York (2005).

[19] FIPA: Communicative Act Library Specification. Tech. Rep. XC00037H, Foundation for Intelligent Physical Agents (2001), http://www.fipa.org, (retrieved: January,2013).

An Implementation Model for Managing Data and Service Semantics in Systems Integration

Miklós Kasza, Vilmos Szűcs, Vilmos Bilicki, Gábor Antal, András Bánhalmi
Department of Software Engineering, University of Szeged
Szeged, Hungary
{kaszam,vilo,bilickiv,antalg,banhalmi}@inf.u-szeged.hu

***Abstract*—The ubiquity of data, services and computing devices demands a higher level of understanding of their real nature if one wants to create value-added services based on them. The human brain can understand the concepts behind these artifacts and find appropriate conceptual links easily, however, by applying semantic technologies (ontologies, concept mapping, inferencing, etc.) computer programs can also be taught to behave similarly. Semantic solutions based on these technologies can lead to powerful value-added services for various domains. A generic domain that can be addressed successfully with the help of these technologies is systems integration. In this paper we introduce a generic implementation model that was developed to serve as a basis of integration solutions in various real-life projects.**

***Index Terms*—semantic knowledge representation; ontologies; ontology mapping; ontology merging; systems integration; device management**

I. Introduction

Data and computing is everywhere nowadays. Specialized data providers are collecting and producing data in different domains of each and every aspect of our lives. The bare availability of the enormous amount of data does not make it usable in itself. For providing valuable services to the masses, somebody has to convert it to consumable information (or more importantly knowledge) that can be understood and acted upon by people easily. The ubiquity of computing makes this picture even more complex. A myriad of devices exist on the market possessing various capabilities for creating and accessing data. Data and device providers and service integrators have to work hand in hand to provide the appropriate value-added services, since people demand more and more sophisticated services and want them to be integrated with each other as seamlessly as possible. These facts lead to a scene filled with heterogeneous information sources, channels, consumers and computing devices.

Heterogeneity and diversity lead to a wide-scale interpretation of concepts. Considering two different computer programs dealing with the same domain-specific problems, the representation of common concepts can vary heavily. For instance, an author of a business-related document can be stored explicitly in the document's meta-data section, while in the case of an electronic mail the author can be the sender of the e-mail. Basically, the relation of the author and the sender concepts in this case are not straightforward for computer programs, but can be interpreted easily by humans and thus computers can be "taught" to act similarly. However, the manual interpretation of each domain concept and finding the relationships between each other can use up huge amount of human power and thus is not cost-effective.

Recently the growing availability of computing power created the possibility of advanced information processing. One of the key aspects is the identification and representation of the information's semantic content. The computer programs have to provide the means for capturing the meaning of various pieces of information. Knowledge representation is as old as computer science itself. Multiple approaches exist for representing human knowledge in the form of machine-processible artifacts. Basic information (e.g., the birth date of a person, topic of a university course) can be represented easily; however, the description of the meaning of the information snippets stored in a computer system is a hard problem. Still, it is an important problem, since the attachment of semantic information to the stored knowledge leads to new ways of information management.

As an ideal vision, one can imagine a world of autonomous, co-operating services that have the knowledge of a common concept set and the meaning of various concepts included in them, as well. Studying the field of semantic knowledge representation and processing brings us closer to this idealistic state; where the mapping of various concepts takes place automatically; the information is filtered based on the interest of the target audience and can connect pieces of information based on the meaning.

This paper provides insights on some key problems in the field of semantic information processing and a possible implementation model that can be used while solving the problems. The insights and the model are based on real-life project-based experience. The paper is structured as follows. Section II presents some research projects related to complex integration problems and introduces several R&D projects the authors base their experience on and highlights the key problems that were identified and partially solved during the execution of these projects. Section III describes the problem of semantic information representation and a model for it, which results from the projects that are strongly connected to this field. Section IV explains some practically feasible and effective ways to collect source data for semantic applications. Section V provides a brief description of ontology matching, merging and mapping and their practical applications. Section VI discusses various aspects of the automatization possibilities of semantic information processing and real-life implementations dealing with automatization. Section VII concludes the paper and identifies some interesting problem areas that are targeted to be further studied and improved in the future.

II. Related Work

There are several research projects which use ontologies and semantics to solve complex integration and communication problems. In the TMTFactory [1] tourism research project [2] the main goal was the integration of several semantic services (e.g., museum, cinema, restaurant searching services) using the ALIVE [3] architecture to create a service collection which can be used by tourists. The project included a Streetbox application which can be used on interactive street displays to find points of interest in a specific area. The improvement of service discovery, system stability and maintainability gave the motivation to use semantic technologies in the application. The Tripcom [4] project aimed the change of Web Service architectures using machine-to-machine communications and ontologies. The TripleSpace technology of the project gives the users a global space of web services to use with the ORDI (Ontology Representation and Data Integration) middleware which can help the modification of the system knowlegde base. The project has an e-Health use-case with the integration of several health services (e.g., data of patients, doctors, hospitals, specialists) to make them easier to use.

The Department of Software Engineering, University of Szeged, took part in several semantic information management-related projects during the course of the recent years. These projects include EU-funded R&D projects, projects funded by the Hungarian Government and projects executed jointly with industrial partners. The common aspect of these projects is the use of semantic technologies in various research and development areas. Out of these projects came several valuable findings that can be used to enhance the practical application of semantic technologies in real-life scenarios. This section covers the scope and overview of these projects.

A. The CONVERGE Project

The electronics industry in Europe faces strong competition not only with companies located in the United States but recently companies in far eastern countries endanger the competitiveness of their European counterparts as well. This challenge can be efficiently targeted by the European industry only if taking the altered circumstances in account and collaborating effectively with each other. Low geological distances and the availability of high-level industrial technology can help them to do so. In order to enhance the efficiency of European-level collaborations, so called non-hierarchical supply chains are being formed. These supply chains are sticked together by decisions made on novel levels; therefore novel approaches are required to address the information sharing issues of the companies. The distinction between sharable and non-sharable information is of key importance in this field. Because of the novelty of the new approaches, the appropriate methodology and tooling is still missing [5], [6]. The CONVERGE project aims at eliminating this imperfection by providing the appropriate methodology and toolkit for supporting decision making on strategic and tactical levels in non-hierarchical supply chains.

In addition to the scientific and technological experts, four industrial partners took part in the consortium executing the project. This fact significantly enhances the acceptance of the project results in the industry, since the industrial partners collaborating in the project provide the field experience that can be exploited to enhance the viability of the emergent methodology and tooling. The solution developed in the project is based on a non-centralized decision support system that enhances the process of production planning and resource optimization by utilizing a novel reference model directly targeted at inter-organizational decision making and existing inter-organizational relationships.

B. Telenor Smart Environment System – Device Integration

The growing availability of ubiquitous computing capabilities can enhance life quality. New smart sensor devices are constantly appearing in various M2M markets. The functions provided by such modern sensor devices enable system developers to create systems that were unimaginable earlier. More and more complex monitoring devices provide functions that can ease the life of humans. However, this rich set of smart devices pose challenges to system developers and specialists as well. To improve the quality and cost effectiveness of smart home systems and services, the necessary devices and sensors must be selected carefully to keep the system available for a low price. However, this can lead to dealing with a diverse set of hardware manufacturers and communication protocols. In addition, the various structures of data coming from the involved devices must be supported by the system. These issues all affect the design and development of the data model and device integration process. Furthermore, the final result of the development process has an impact on the flexibility, the reusability, and the performance of the developed system.

Smart sensor integration processes pose a difficult and complex task for developers. The process of device integration starts at the studying of the protocol used by the given device. The protocol is usually given by the structure of messages (based on a given communication protocol) constructed by the device to send observation or measurement data to a specific server. The structure of the messages is usually defined in a protocol specification document in the development documentation of the device. The messages typically consist of key-value pairs which can be defined by parameter names (keys) as well as data types and possible constraints refer to parameter values. These value sequences provide the exact data that should be forwarded. Besides the parameters some additional information is needed about the place of the data in the message, the type of communication (simple message, acknowledged message, complex communication process) and security. In the project we inspect the device integration processes more deeply and we describe our novel methodology to resolve current issues in the field of smart sensor integration based on semantic protocol and data representations and mappings.

C. R&R Application Service Network

The goal of the Application Service Network project is to create an application service integration platform, in which the integration is not done by IT experts. The end users, who access the services, are to be made capable of assembling integrated services from Internet-enabled service systems that are compatible with the platform. The end users can access the applications (which are assembled directly to meet their demands) as services residing "in the cloud".

The network under construction can unify the telecommunications networks and the IT-services. This way, novel business applications can emerge based on vertical service integration. The target result of the project is a model that is:

- available as a service in the cloud;
- enables the assemblement and usage of custom applications;
- enables the expansion of application components and
- takes the demands of the players in the service network into consideration.

Besides the model, in the project, a model implementation prototype is provided. Based on this implementation prototype the the model is validated using user scenarios. For these purposes, two main scenarios were selected: the *Medical Attendance* scenario and the *Semantic Map* scenario. A brief description of these scenarios follows.

1) Medical Attendance: A telemedicine system provides functions for the health care industry that make medical attendance faster and more reliable using an appropriate telecommunications environment. These services usually solve emergency or non-emergency problems occurring due to large distances. This way, patients can get medical assistance even while being at their homes with the help of various visualisation or data collection devices. On the other hand, telemedicine services ease the communications between medical staff, and thus urgent consultations and the sharing of medical records become possible.

In emergency cases, when patients are not capable of communicating directly, telemedicine systems can save lives. A use case of such a telemedicine system is providing the emergency units with optimal routes to the nearest appropriate hostpital. An algorithm for this problem requires the availability of parameters that lead to more relevant results. Such relevant parameters are the amount of free space in the hospital, the facilities of the hospital, the patient's health-related history records, etc. However, acquiring such vital data is not straightforward.

A system of this type can be relied upon by almost the entire health industry. These systems have to support integration with external systems on a high level. Applying semantic information representation seems an ideal solution for these problems.

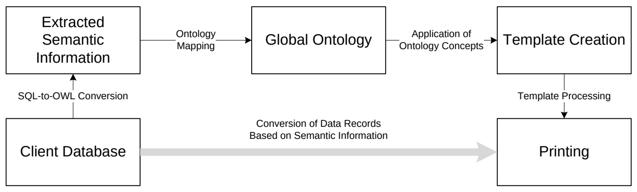

Extracted Semantic Information
Ontology Mapping
Global Ontology
Application of Ontology Concepts
Template Creation
SQL-to-OWL Conversion
Template Processing
Client Database
Conversion of Data Records Based on Semantic Information
Printing

Fig. 1. High level overview of the POS Printing process.

Maintaining semantic context in telemedicine systems can lead to better and safer services.

One part of the project targets the aforementioned scenario in a fully automatized manner. It investigates the possibilities of transferring telemedicine and geo-information services to the world of semantic information management and thus the possibilities for creating a web application that is optimized and automatized more than the existing ones. Therefore, the project's goal was to build the aforementioned solution from ground up to the highest abstract levels using the ALIVE framework [3].

2) Semantic Map: Another representative scenario for the Application Service Network is the *Semantic Map*. In this scenario, the data set originates from a geo-information database containing places, points of interest and paths. The project investigates the valid scope of integration between geo-information data and semantic web technologies.

D. POS Printing

In large supermarkets, the management of the products and the corresponding product-related marketing material can be a cumbersome task. Usually, the preparation, printing and distribution of the marketing material is done by an appropriate service provider other than the supermarket company. The client base of these printing providers are not limited to one client only and thus they have to deal with product- and service-related data originating from various data sources. The POS (Point Of Sale) Printing project targets a printing provider by applying semantic solutions in order to reduce human work. The main aspect of the project is the semantic annotation of existing data stored in the clients' databases and mapping it to a common (global) ontology on the provider's side. This way, the management (design, printing, delivery, etc.) of the marketing material can be based on a common set of product and service store-related concepts, while the integration of different clients' different database schemas can be done in a semantic way. The process of integrating one client's information base is depicted on Figure 1.

The goal of the project is to provide methodology and the appropriate tooling for executing the process in an automatized manner (or as much automatized as is possible).

III. Semantic Information Modelling

In recent years, the term of ontology enjoys a growing popularity in the IT world as it promises an appropriate basis to provide the IT tools for human thinking and decision making ability. The ontology-based interoperability across heterogeneous systems can be achieved to build business logic and complex data flow between processes without the need of human intervention. Ontologies are formal representations of knowledge as sets of concepts within domains, and describe the relationship between them by providing semantic meaning for syntactic terms. The semantic description of data and services allows the automatic understanding and perception of them to achieve collaboration and orchestration between services.

Ontologies can serve as tools for sharing and reusing the existing knowledge in the form of semantically rich structures. Obviously, current computing capacity of modern computers is not enough to store and process information about the world in its entirety. This problem is addressed by partitioning the knowledge into more specific domains. This way, usually ontologies can store knowledge about objects only in several well-defined domains. Despite the differences between various domains, most ontologies provide vocabularies (containing terms that are meaningful in the target domain) and definitions [10].

First of all, the semantic information has to be modeled in some way to be processable by computers. Various models exist for these purpose (RDF, OWL, etc.) In our projects we experimented with some of these models and captured the pros and cons of them (primarily from practical aspects). In the first wave of the projects, the integrated models were semantically annotated by hand (labelling, Java annotations, etc.). This lead to run-time evaluation of the annotated models and the semantic descriptions were generated during run-time. This solution proved to be quite unstable, since it did not make the fine definition of semantic content possible. This way, in the second wave of the projects, we used ontology modelling tools (TopBraid Composer Free Edition [8], Protégé [9]) to define semantics. With the help of these tools the semantic information emerged in the form of OWL-documents. These documents can be stored in generic repositories (file systems, relational databases) or ontology-specific repositories. This solution leads to development-time ontology definitions, however, the ontology mappings are done during run-time.

In the CONVERGE project a given data source with specific metadata was matched to some conventional ontologies, e.g., FOAF [11] and Dublin-Core [12]. In the POS Printing project local ontologies were generated using SQL schemas and matched to a manually annotated global ontology.

IV. Data Collection

By having a model appropriate for our purposes, a well-defined collection methodology had to be developed to populate the model with data. As we found, the collection methodology is a very important area of semantic information management, since the whole semantic ecosystem is viable only if the semantic information can be extracted from existing information sources. Otherwise, the population of the data model itself would take lots of efforts.

First of all, in each project, we had to find the available information repositories that could be used as data sources. We found that various public thematic repositories (accessible via Internet) can serve as bases for several domain-specific aspects. Additionally, other non-public information sources (such as local databases, mail boxes, file systems, etc.) can be used to refine the set of available information.

In the introduced projects we used adapter-based solutions in all cases. As an example, the high-level architecture of CONVERGE's data mediator subsystem is shown on Figure 2. As it can be seen, the architecture is based upon an extensible modular structure that can be extended by introducing new system adapter modules. In the project, adapters have been created for IMAP-based mailboxes, network shares and for CAS Software AG's CASOpen platform. Similar approaches were followed in the case of other projects, as well.

In other projects the data adapter components are called gateways. These gateways mediate the data between the adapted and the target systems by:

- receiving data in the format of the integrated external systems;
- adapting the data to the schema of the target system based on semantic mappings;
- transferring the adapted data to the target system;

In most of the projects HTTP-based gateways were used, however, in CONVERGE, adapters for IMAP-based mail boxes and generic file shares were also developed. In most of the projects the gateway modules are automatically generated from the available semantic information.

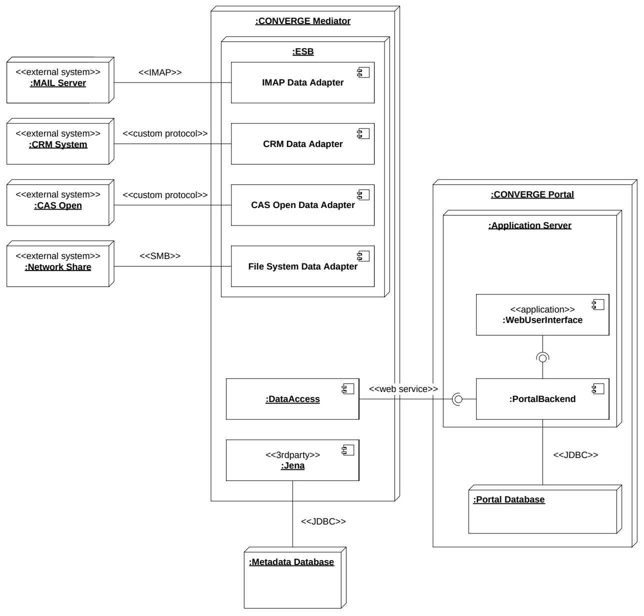


Fig. 2. High-level overview of the CONVERGE's data mediator subsystem.

V. Matching, Mapping and Merging

The application of ontologies in knowledge representation is quite straightforward, as long as we do not try to manage multiple ontologies and we do not try to integrate our ontology into a heterogeneous system. In the latter cases we can face incompatibility and heterogeneity issues. This is a common problem, since usually multiple ontologies can be utilized in distributed systems. The most problematic cases appear when domains of the different ontologies are overlapping, i.e., they represent similar types of knowledge but the syntaxes of them are different partially or in their entirety [6]. However, ontology mapping (or matching) solutions can be provided in order to resolve these issues. These tools can find the rules on which concepts in one ontology can be mapped to concepts in other ontologies [13], [14], [15], [16], [17], [18], [19].

Generally, ontology mappings can be classified as follows:

- mapping between a global and multiple local ontologies (*global-local* mappings)
- mapping between multiple local ontologies (*local-local* mappings)
- ontology *merging* and *alignment*

Global-local mappings can be used as the means of ontology integration, i.e., they can describe the rules of mapping various local ontologies to an integrated global ontology [20], [21]. Local-local mappings map the local contents of each ontology on the basis of semantic relationships without the existence of a global ontology. Ontology merging techniques enable the creation of a single, coherent ontology based on multiple existing ontologies dealing with the same domain. The new, merged ontology contains information about each source ontology in a more-or-less unchanged form. Ontology alignment's main purpose is to find a link between two separately stored ontologies when they become inconsistent [22], [23].

After some investigation it was determined that for the purposes of the CONVERGE project (i.e., to create a knowledge model that is capable of storing heterogeneous information available all around in an enterprise), ontology merging was a viable solution to use. However, before being able to merge the available knowledge, first we had to

1) gather the data from external systems and transform it to an ontology-based presentation format
2) find mappings between various concepts used in external systems in order to be able to integrate different concepts originating from different systems but having the same semantic meaning.

During our experimentation we evaluated and compared five different ontology matching tools based on some functional and subjective non-functional metrics: WSMT Mapping [24], COMA++ [25], PROMPT [26], MAFRA toolkit [27] and PyOntoMap [28]. On the functional side we created some similar ontologies and found the optimal mappings by hand. These sets of optimal mappings served as the baseline that was used to evaluate the goodness of the tools' results. Because of the hand-made mappings we used relatively small ontologies, they contained 8.25 concepts and 21.5 attributes on average.

TABLE I
Efficiency of automatic matching tools in a sample ontology-mapping scenario.

Toolkit	S	d_s	L	d_l	C	d_c
WSMT	12	$\frac{12}{20}$	13	$\frac{13}{23}$	7	$\frac{7}{20}$
COMA++	15	$\frac{15}{20}$	17	$\frac{17}{23}$	9	$\frac{9}{20}$
PROMPT	12	$\frac{12}{20}$	5	$\frac{5}{23}$	1	$\frac{1}{20}$
MAFRA	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*
PyOntoMap	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*

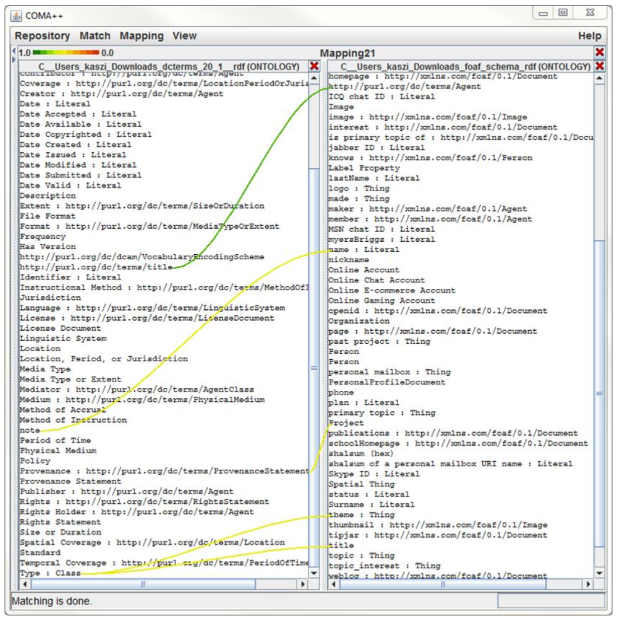
We used 3 metrics for evaluating the goodness of each tool:

- d_s: the ratio of mappings found by the matching tool when the two ontologies contain only structural differences
- d_l: the ratio of mappings found by the matching tool when the two ontologies contain only lexical differences
- d_c: the ratio of mappings found by the matching tool when the two ontologies contain both lexical and structural differences

In each case, the number of optimal mappings were 20, 23 and 20 for structural, lexical and complex problems, respectively. The results found by the evaluations are summarized in Table I. As it can be seen in the table, the MAFRA Toolkit and PyOntoMap frameworks did not work on the sample ontology set.

With regards to the non-functional metrics, COMA++ proved to be the best choice due to its speed, integrability and automation possibilities. The WSMT Mapping tool seemed to be an accurate tool, however, its functionality is automatizable only partially, because it is built upon the availability of user activity. PROMPT proved to be relatively imprecise, it found only a small part of mappings, and it also lacks the proper automatization functions. The MAFRA Toolkit currently does not include a usable semi-automatic ontology mapping, this way it can not be automatized. PyOntoMap is easy to use, however it is also imprecise and can map only concepts, not attributes. Based on our evaluation, the toolkits under investigation provide API-s for semi-automatic or automatic mappings, however, they are very poorly documented.

After the initial evaluation of the tools, we decided to apply COMA++ to the vocabularies used in our knowledge representation model. Despite the good results in the artificial tests, even COMA++ achieved poor results. It found mappings between totally unrelated concepts and missed almost all the reasonable mappings. In Figure 3 a sample mapping can be seen between the Dublin Core and FOAF vocabularies. It can be seen that it found mappings only on the basis of lexical similarities, however, the lexical mappings were even false (found mapping between *note* and *name* or *Type* and *theme* concepts). Correct mappings were missing in the case of mapping by structural similarities, as well. After some deeper investigation, it turned out that the unsuccessful application of the tool to the real vocabularies can be deducted to some


COMA++
Repository
Match
Mapping
View
Help
1.0
0.0
Mapping21
C__Users_kaszi_Downloads_dcterms_20_1__rdf (ONTOLOGY)
C__Users_kaszi_Downloads_foaf_schema_rdf (ONTOLOGY)

```
Contributor : http://purl.org/dc/terms/Agent
Coverage : http://purl.org/dc/terms/LocationPeriodOrJuris
Creator : http://purl.org/dc/terms/Agent
Date : Literal
Date Accepted : Literal
Date Available : Literal
Date Copyrighted : Literal
Date Created : Literal
Date Issued : Literal
Date Modified : Literal
Date Submitted : Literal
Date Valid : Literal
Description
Extent : http://purl.org/dc/terms/SizeOrDuration
File Format
Format : http://purl.org/dc/terms/MediaTypeOrExtent
Frequency
Has Version
http://purl.org/dc/dcam/VocabularyEncodingScheme
http://purl.org/dc/terms/title
Identifier : Literal
Instructional Method : http://purl.org/dc/terms/MethodOfI
Jurisdiction
Language : http://purl.org/dc/terms/LinguisticSystem
License : http://purl.org/dc/terms/LicenseDocument
License Document
Linguistic System
Location
Location, Period, or Jurisdiction
Media Type
Media Type or Extent
Mediator : http://purl.org/dc/terms/AgentClass
Medium : http://purl.org/dc/terms/PhysicalMedium
Method of Accrual
Method of Instruction
note
Period of Time
Physical Medium
Policy
Provenance : http://purl.org/dc/terms/ProvenanceStatement
Provenance Statement
Publisher : http://purl.org/dc/terms/Agent
Rights : http://purl.org/dc/terms/RightsStatement
Rights Holder : http://purl.org/dc/terms/Agent
Rights Statement
Size or Duration
Spatial Coverage : http://purl.org/dc/terms/Location
Standard
Temporal Coverage : http://purl.org/dc/terms/PeriodOfTime
Type : Class
```

```
homepage : http://xmlns.com/foaf/0.1/Document
http://purl.org/dc/terms/Agent
ICQ chat ID : Literal
Image
image : http://xmlns.com/foaf/0.1/Image
interest : http://xmlns.com/foaf/0.1/Document
is primary topic of : http://xmlns.com/foaf/0.1/Docu
jabber ID : Literal
knows : http://xmlns.com/foaf/0.1/Person
Label Property
lastName : Literal
logo : Thing
made : Thing
maker : http://xmlns.com/foaf/0.1/Agent
member : http://xmlns.com/foaf/0.1/Agent
MSN chat ID : Literal
myersBriggs : Literal
name : Literal
nickname
Online Account
Online Chat Account
Online E-commerce Account
Online Gaming Account
openid : http://xmlns.com/foaf/0.1/Document
Organization
page : http://xmlns.com/foaf/0.1/Document
past project : Thing
Person
Person
personal mailbox : Thing
PersonalProfileDocument
phone
plan : Literal
primary topic : Thing
Project
publications : http://xmlns.com/foaf/0.1/Document
schoolHomepage : http://xmlns.com/foaf/0.1/Document
shalsum (hex)
shalsum of a personal mailbox URI name : Literal
Skype ID : Literal
Spatial Thing
status : Literal
Surname : Literal
theme : Thing
thumbnail : http://xmlns.com/foaf/0.1/Image
tipjar : http://xmlns.com/foaf/0.1/Document
title
topic : Thing
topic_interest : Thing
weblog : http://xmlns.com/foaf/0.1/Document
```

```
Matching is done.
```

Fig. 3. Sample COMA++ mapping.

important issues:

- the lack of a proper thesaurus worsens the mapping accuracy: as it can be seen, COMA++ can not map differently named concepts having the same meaning automatically. A domain-specific thesaurus could help this issue.
- the "flatness" of the applied ontologies interferes with structural mapping approaches: the mapping could be enhanced by taking structural similarities into account. However, our target ontologies do not have a complex structure, only several subclassings are included, and this fact together with the lexical differences leads to poor mapping results.

Obviously, the matching capabilities of the currently available solutions can be enhanced. Some possible solutions for enhancing ontology matching:

- Refining literal comparisons: resolving abbreviations; learning notational conventions; identifying frequent prefixes and suffixes; word swappings and mixed language notations);
- Enhancing structural comparisons [29], [30]: identifying relations between concept hierarchies and class properties; dealing with transitive and inversible relationships;
- Aggregating similarity metrics: introducing adaptive aggregation, threshold refinement [31] and weighing [32]; introducing methods using new types of discriminative machine learning algorithms or decision trees [32]; shifting to fuzzy aggregation [33], [34];
- Enabling human interventions for refinement.

These solutions would solve problems using methods from other fields (e.g., artificial intelligence, natural language processing, machine learning) and using the given approaches together, taking practical points of view into account. Practical applicability of each enhancement is currently under further investigation and is subject of future work.

VI. AUTOMATIZATION OF PROCESSING

When thinking about automatization of information processing tasks, the first question is about the subject material for automatic processing. Since different data repositories and manageable systems usually expose different interfaces for data access and represent data in different formats, the first candidates for automatic processing are data schemas and data access methods. While the data access methods can usually be defined by the appropriate standardized or proprietary protocols and data formats, the interpretation of the data schemas is usually a harder task. The structural adoption of an external data set is therefore based on well-defined rules that are specific to the system to be integrated. The target representations can be formats specially designed to hold semantically enriched information, such as RDF or OWL graphs.

The integration on this level is a well-studied problem and as such, there exist well-elaborated solutions for it.

It must be noted that these adapters can deal only with the data access related differences of the different systems, but do not consider the schema-related differences and specificities. As it was discussed, the automatization of the schema mappings can be donesuccessfully only if the appropriate semantic annotations exist for the adapted schema. In these cases, human interaction is involved only on the data schema/service side. Once they are annotated correctly, human interaction is not required. Unfortunately, the lack of properly annotated data sets and services enforces human interactions in other stages as well.

This way, we identified three models for human interaction involvement:

- No additional human interaction required (Medical Attendance)
- Human interaction is required during development phase (Semantic Map, TSES-DI, CONVERGE, Factory)
- Human interaction is required during run-time

The first approach is the idealistic one, it has been covered earlier. The second approach requires intervention to the processes during development time. In this case, the domain-specific knowledge is inserted into the system during the development of adapter components. For these purposes, the appropriate development tools have to be provided for the developers and domain experts to be able to express their knowledge easily. In the projects we developed tools in the form of Eclipse-plugins that make these tasks easier. Figure 4 shows the wire-frame design of the tool for defining concept matchings.

In most of the projects we used development-time human interactions, since this solution proved to be the most viable at the time. However, the tools used during development time can be elevated to a level, on which end users are made capable of defining semantic properties of the domain. This makes the third approach feasible.

The third option for involving human interaction exposes tools for domain experts during run-time. This way the experts can insert their knowledge into the system without having the need for rebuilding the involved components.

It has to be noted, that the two later cases do not require a priori annotation of the target data sets and services, however the annotation has to be done before the data sets and services can be used by the system. Compared to the first solution, this annotation process takes place later in the process.

Since a primary goal of automatization is enhancing productivity, we made productivity measurements in the Semantic Map scenario of the R&R Application Service Network project. In these measurements the productivity of classic development was compared with semantics-based development (in which automatic code generation can take place with the availability of semantic information). The steps of the compared processes are depicted in Figure 5. As it can be seen, the development was broken up to three individual stages based on the nature of the work required on the stage: design, modelling and implementation. The left side of the picture shows the steps a developer had to take using the semantics-based development, while the right side describes the process of classical development. In both cases a new data source (with the appropriate components) had to be developed.

By looking at the number of steps required using each approaches, one can see that the classic development method requires more human tasks to be done. After the measurements (results shown in Figure 6) we found that the productivity in the design phase does not differ significantly, since in both methods, the developers have to understand the tasks and the domain. The productivity of this phase can be enhanced by assigning tasks to developers familiar with the target domain. The first significant difference shows up in the modelling phase. In this phase, the ontology-based development approach lets the developers concentrate on the important domain-related concepts and their relationships and thus frees the developers from doing manual design in the cumbersome areas. Finally, the results from the implementation phase show

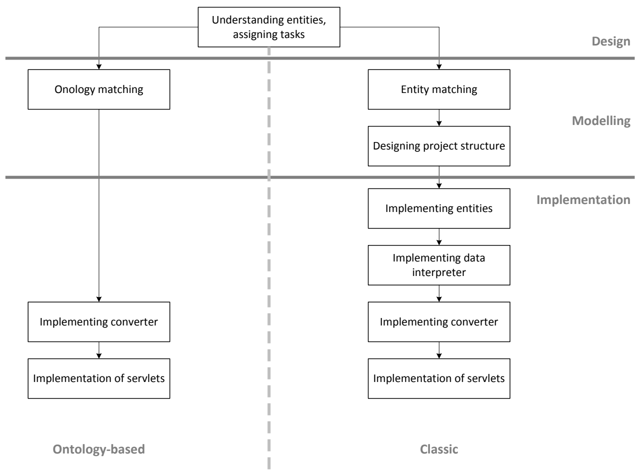
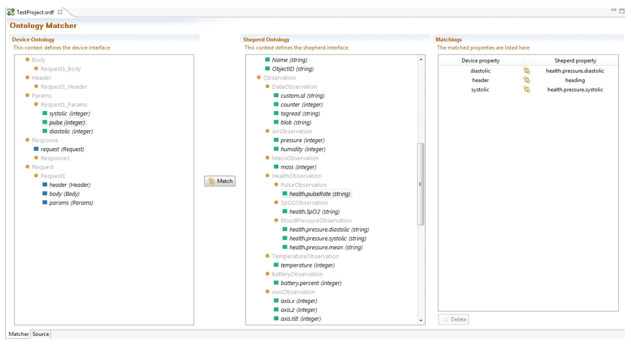

Understanding entities, assigning tasks
Design
Onology matching
Entity matching
Modelling
Designing project structure
Implementation
Implementing entities
Implementing data interpreter
Implementing converter
Implementation of servlets
Implementing converter
Implementation of servlets
Ontology-based
Classic

Fig. 5. Processes followed for measuring productivity-related differences between classic and ontology-based development approaches

the real benefit of using the ontology-based approach over the classic one. In this phase the productivity was much higher, since the developers did not have to deal with the details of the program code. Therefore there are much less mistakes caused by the developers and heterogeneities caused by misunderstandings. The generated code is much easier to improve, correct and maintain.

VII. Conclusion and Future Work

During the execution of the projects dealing with semantic information management, we revealed that the information required for successful systems integration can leverage semantic additions. However, we identified some problem areas that can be enhanced in order to make semantic processing more automatic and requiring less human interactions.



Ontology Matcher

Device Ontology
This context defines the device interface

- Body
  - Request1_Body
- Header
  - Request1_Header
- Params
  - Request1_Params
    - systolic (integer)
    - pulse (integer)
    - diastolic (integer)
- Response
  - request (Request)
  - Response1
- Request
  - Request1
    - header (Header)
    - body (Body)
    - params (Params)

Sheperd Ontology
This context defines the shepherd interface

- Name (string)
- ObjectID (string)
- Observation
  - DataObservation
    - custom.id (string)
    - counter (integer)
    - tagread (string)
    - blob (string)
  - AirObservation
    - pressure (integer)
    - humidity (integer)
  - MassObservation
    - mass (integer)
  - HealthObservation
    - PulseObservation
      - health.pulseRate (string)
    - SpO2Observation
      - health.SpO2 (string)
    - BloodPressureObservation
      - health.pressure.diastolic (string)
      - health.pressure.systolic (string)
      - health.pressure.mean (string)
  - TemperatureObservation
    - temperature (integer)
  - BatteryObservation
    - battery.percent (integer)
  - AxisObservation
    - axis.x (integer)
    - axis.z (integer)
    - axis.tilt (integer)

Matchings
The matched properties are listed here

Device property	Sheperd property
diastolic	health.pressure.diastolic
header	heading
systolic	health.pressure.systolic

Fig. 4. User interface of an ontology matching development tool prototype

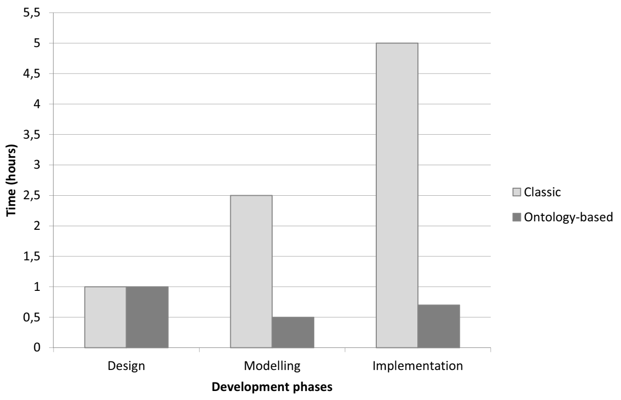


Fig. 6. Results of productivity measurements

A. Availability of a Priori Semantic Annotations

Real-time processing lacks the possibility to find properly annotated services and data sets available. By introducing end-user programming via domain specific languages, this constraint can be released, but in this case the appropriate tooling and generic processing engine must be available.

B. Better Matching Algorithms

The automatic matching of domain concepts and their true relationships requires more flexible matching algorithms. Matching based on simple lexical and structural properties of the concepts can lead to incomplete matching that require human intervention to make it usable. Applying advanced techniques can lower the requirement of human intervention.

C. Productivity

The application of semantics-based modelling and automatic code generation in systems integration makes the productivity of the development tasks more effective. This productivity boost can be introduced in run-time semantics definitions with the help of end-user programming and domain-specific languages.

ACKNOWLEDGMENT

Parts of this work are done in the *CONVERGE - Collaborative Communication Driven Decision Management in Non-Hierarchical Supply Chains of the Electronics Industry* project which is funded by the European Union under grant number 228746 in FP7-NMP.

Parts of this work are funded by the Government of Hungary under grants GOP-2009-1.1.1 and KMOP-2009-1.1.1 and by R&R Software Ltd.

Parts of this work are funded by the Government of Hungary under grants GOP-2011-1.1.1 and by Factory Creative Studio Ltd.

Parts of this work are funded by Telenor Hungary.

REFERENCES

[1] TMT Research & Innovation. Available at: http://research.tmtfactory.com/

[2] K. Alonso, M. Zorrilla, R. Confalonieri, J. Vzquez-Salceda, H. Inan, M. Palau, J. Calle and E. Castro, *Ontology-Based Tourism for All Recommender and Information Retrieval System for Interactive Community Displays*, Information Science and Digital Content Technology (ICIDT), 2012 8th International Conference, 26-28 June 2012.

[3] IST-Alive Project, Available at: http://www.ist-alive.eu/.

[4] The official website of the TripCom project. Available at: http://www.tripcom.org/

[5] B. Scholz-Reiter, J. Heger, C. Meinecke, D. Rippel, M. Zolghadri, R. Rasoulifar, *Supporting Non-Hierarchical Supply Chain Networks in the Electronics Industry*

[6] C. Meinecke and D. Rippel (eds.), *Decision-Making Model, Data Mapping and Integration Roadmap*, Project Deliverable document, CONVERGE, retrieved from http://www.converge-project.eu/images/stories/pd/D2.2-Decision Making Model & Data Mapping.pdf on 2 October, 2011.

[7] K. Kalaboukas (ed.), *System Requirements, Data Sharing concept and System Architecture*, Project Deliverable document, CONVERGE, retrieved from http://www.converge-project.eu/images/stories/pd/D3.1-Specification.pdf on 2 October, 2011.

[8] TopQuadrant TopBraid Composer Free Edition. Available at: http://www.topquadrant.com/products/TB_Composer.html.

[9] The Protg Ontology Editor and Knowledge Acquisition System. Available at: http://protege.stanford.edu/.

[10] J. D. Heflin, *Towards the Semantic Web: Knowledge Representation in a Dyncamic, Distributed Environment*, PhD thesis, Faculty of the Graduate School of the University of Maryland, College Park. 2001. Retrieved from http://www.cs.umd.edu/fs/www/projects/plus/SHOE/pubs/heflin-thesis-orig.pdf in 2 October, 2011.

[11] FOAF Vocabulary Specification 0.98. Available at: http://xmlns.com/foaf/spec/.

[12] DCMI Home: Dublin Core Metadata Initiative (DCMI). Available at: http://dublincore.org/.

[13] E. Rahm and P. A. Bernstein, A survey of approaches to automatic schema matching, In *The VLDB Journal, vol. 10 (2001)*, pp. 334–350.

[14] P. Shvaiko and J. Euzenat, A Survey of Schema-based Matching Approaches, In *Journal on Data Semantics IV (2005)*, pp. 146–171.

[15] A. Gal and P. Shvaiko, Advances in Ontology Matching, In *Advances in Web Semantics I (2009)*, pp. 176–198.

[16] S. M. Falconer, N. F. Noy, and M-A. Storey, Ontology Mapping — A User Survey, In *Proceedings of the Workshop on Ontology Matching (OM 2007)* at ISWC/ASWC 2007, pp. 113–125, Busan, South Korea (2007)

[17] X. Su, *Semantic Enrichment for Ontology Mapping*, PhD thesis, Norwegian University of Science and Technology, October 2004.

[18] P. Shvaiko and J. Euzenat, Ten challenges for ontology matching, In *Proceedings of the OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008. Part II on On the Move to Meaningful Internet Systems (2008)*, pp. 1164–1182.

[19] M. Sabou, M. d'Aquin, and E. Motta, Using the Semantic Web as Background Knowledge for Ontology Mapping, In *Proceedings of the International Workshop on Ontology Matching (OM-2006)*, pp. 1–12.

[20] H. S. Pinto and J. P. Martins, A methodology for ontology integration. In *K-CAP '01: Proceedings of the 1st international conference on Knowledge capture (2001)*, pp. 131-138

[21] C. M. Keet, *Aspects of Ontology Integration*, Technical report, School of Computing, Napier University, January 2004.

[22] J. de Bruijn, M. Ehrig, C. Feier, F. Martín-Recuerda, F. Scharffe, and M. Weiten, Ontology mediation, merging and aligning, In *Semantic Web Technologies (July 2006)*

[23] T. C. Hughes and B. C. Ashpole, The Semantics of Ontology Alignment, In *I3CON. Information Interpretation and Integration Conference (2004)*.

[24] M. Kerrigan, A. Mocan, M. Tanler, and W. Bliem, *Creating Semantic Web Services with the Web Service Modeling Toolkit (WSMT)*. Available online at http://www.sti-innsbruck.at/fileadmin/documents/papers/creating-semantic-web-services-wsmt.pdf. Accessed on 2 October 2011. The Web Service Modeling Toolkit (WSMT), http://www.sourceforge.net/projects/wsmt

[25] *Schema and Ontology Matching with COMA++*, Retrieved from http://dbs.uni-leipzig.de/Research/coma.html on 2 October 2011.

[26] N. Noy, Prompt, In the *Protégé Community of Practice Wiki*. Retrieved from http://protege.cim3.net/cgi-bin/wiki.pl?Prompt on 2 October, 2011.

[27] N. Silva and J. Rocha, *Semantic Web Complex Ontology Mapping*, Retrieved from http://sourceforge.net/projects/mafra-toolkit/files/mafra-toolkit/0.2/SemanticWebComplexOntologyMapping.pdf/download on 2 October, 2011.

[28] P. Besana, *Using Demster-Shafer for Combining Ontology and Schema Matchers*, Retrieved from http://pyontomap.sourceforge.net/UsingDSforOntoMap.pdf on 2 October, 2011.

[29] J. Euzenat and P. Shvaiko, *Ontology Matching, 1st ed.* Springer Publishing Company, Inc., 2010.

[30] A. K. Alasoud, *A Multi-Matching Technique for Combining Similarity Measures in Ontology Integration*, Phd Thesis, Concordia University Montral, Qubec, Canada, 2009.

[31] T. Kohonen, Learning Vector Quantization, in *The Handbook of Brain Theory and Neural Networks*, MIT Press, Cambridge, MA, 1995, o. 537-540.

[32] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach, 2nd ed.* Prentice-Hall, Englewood Cliffs, NJ, 2003.

[33] J. Dombi, Towards a General Class of Operators for Fuzzy Systems, *IEEE T. Fuzzy Systems, vol. 16,* pp. 477–484, 2008.

[34] M. Detyniecki, *Fundamentals on Aggregation Operators*, University of California, Berkeley, 2001.

Explanations of Recommendations

Considering Human Factors in Recommender Systems

Muhammet Tugberk Isyapar
Computer Engineering
Middle East Technical University
Ankara, Turkey
tugberk.isyapar@metu.edu.tr

***Abstract*— Traditional evaluation metrics based on statistical formulas employed to assess the performance of a recommender system are now considered to be inadequate when utilized solely. New metrics considering the quality of user-system interaction alongside with traditional ones have been proposed in evaluation process in order to arrive at more adequate results. Generating explanations for recommendations is a research topic that has emerged as a way to evaluate the system with respect to various criteria considering users' opinions and feelings. This paper presents the state-of-the-art with respect to the explanation of a recommendation.**

***Keywords*— Recommender systems; recommendation algorithms; design and evaluation of recommender systems; explanations; human factors.**

I. Introduction

The objective of recommendation technology is to help the end-user making sense of large and growing amounts of data. Recommender systems have been developed for various problem domains including automatic movie recommendation. People keen on cinema would like to discover yet-unseen movies that suit their tastes and avoid the ones that they would probably regret watching. Before deciding to watch a movie, they may also like some kind of prediction about the item, since there are usually many alternatives to choose and dedicating a considerably long time and other resources to a bad movie is likely to be annoying. Movie recommender systems for the domain of cinema including MovieLens [7] and Netflix [8] have been devised to provide the demanded facilities. Other problem domains for which recommender systems have been developed include online shopping, news filtering, academic paper discovery, and social networking.

The recommendation problem, as commonly formulated, is the problem of estimating ratings for yet-unseen items by a user. The estimation is called *prediction* and is based on the ratings given by the user to seen items. As soon as predictions for unseen items are generated, the system can recommend to the user several items with the highest ratings, which is commonly named in the literature as *top-N recommendations* [1][3]. The user of the recommender system is referred to the *active user* in this paper.

There are various methods for estimating ratings for the yet-unrated items and recommender systems are classified according to the approach chosen for prediction generation. Current research poses three categories [1]:

- *Collaborative recommendations:* The active user is recommended items similar to the ones that are liked by other users with similar tastes without considering item contents.
- *Content-based recommendations:* The active user is recommended items that are similar to the ones he/she liked in the past depending on the contents of the items.
- *Hybrid approaches:* These imply developing methods that combine benefits yet avoid disadvantages of collaborative and content-based methods.

Recommender systems technology proposes several statistical metrics to measure the coverage, accuracy and precisions of generated recommendations. It is currently thought that these metrics can only partially evaluate the systems [2][3][4]. User satisfaction, serendipity, diversity and trust are now considered among important evaluation criteria [3] since recommender systems are deployed with well-designed user interfaces and the quality of recommendations tends to increase by providing better user-system interaction through interfaces. Generating explanations for the recommendations made via the interface to the user has emerged as an idea to compensate for the new evaluation criteria. Explanations have several aims determined by the characteristics of the problem domain and these aims could be utilized in evaluating explanations. Throughout the rest of the paper, definitions, aims, evaluation, types, design and usage of explanations will be described at sufficient detail to show the current trends in the technology and the effects of explaining recommendations.

II. Aims and Evaluations of Explanations

Explanation facilities propose several aims to be attained [3]. Good explanations tend to increase user satisfaction, give users trust about the system and inspire loyalty, persuade them to buy or use the recommended item or correctly guess the user's possible rating about the item, and make it easier and quicker for users to find what they want. Distinct aims may hold only in particular domains since coexistence of particular aims cannot be true by definition. Seven possible aims of employing explanations, evaluation of explanations with respect to these aims, ways of presenting recommendations with explanations, and facilities providing user-system interaction will be described in the rest of this section of the paper.

A. Aims of Explanations

Transparency is the first aim of explanations. Explanations making a system transparent help the user understand how a system works, i.e., why a particular recommendation is generated instead of others. The user receiving the explanation can then understand the mechanisms of the system and act accordingly.

The second aim of explanations is *flexibility*. A flexible system can correct itself whenever the user spots an incorrect assumption made about them. Transparency and clarity could be regarded as two aims to be realized in a cyclic fashion. Explanations should provide insights into the system as a first step and should next allow the user to correct reasoning, that is, explanations should make the system flexible.

The third aim of explanations is inspiring *trust*. Good explanations tend to increase users' confidence in the system and users are loyal to systems that they regard as trustworthy. Trust as an aim has bounds with transparency. Whenever the system is unsure about the recommendation it generates, it should state that with appropriate explanations. Users' trust for the system increases whenever they are provided information about the quality of recommendations they receive. Moreover, as most commercial recommender systems come with user interfaces, the design of the interface is an important factor that affects users' trust.

Persuasiveness is the fourth aim of explanations. Appropriate explanations of recommendations could contribute to the system's persuasiveness by convincing user to try or buy the recommended item. If the quality of generated explanations is adequate, users may be affected by the rating predicted by the system and may believe that they would give the same rating even if not provided recommendations. However, a balance should be taken into account as too persuasive explanations carry the risk of convincing the user to buy a bad item which may end up in a decrease of user's trust and loyalty to the system.

Effectiveness is the fifth aim of explanations. An explanation may help the user to make better decisions in the sense that the predicted rating of the item will actually reflect the user's own preferences and tastes. To put it in other words, a recommender system with effective explanations assists a user to reach the items that they will like in the end. Therefore effectiveness has bounds with the accuracy of the recommendation algorithm and could be thought as a user-based extension to the statistical accuracy utilized in the literature.

The sixth aim of explanations is *efficiency*. Explanations may make it quicker for users to decide which recommended item fits their needs best. As recommendation process itself is finding the most valuable information out of huge amounts of data, it is also important that the user could reach what they are looking for in the least possible amount of time. Explanations should provide interaction with user in order to reduce the search time and make the system efficient.

The seventh and the last aim of explanations is *satisfaction*. Satisfaction is a broad and abstract category yet in the context of recommender systems it could be considered as making the use of system fun. Explanations could serve as means that increase users' satisfaction with the system. The quality of explanations is crucial since poor explanations are likely to decrease the user's interest or acceptance of the system. Moreover, explanations are deployed as an integral part of the user interface of the recommender system. It is known [3][4] that users tend to like more features included in the interface; therefore, including an explanation facility in commercial recommender systems is an important contribution to users' overall satisfaction with the system.

B. Evaluation of Explanations

In this section, we explore the criteria used to evaluate a recommendation. The aims of explanations could be utilized as criteria to evaluate how good an explanation is. More criteria based on combinations of the seven aims could be generated. The appropriate evaluation technique regarding each criterion will be described below.

To evaluate an explanation with respect to the *transparency* criterion, one could ask users if they believe the recommendations they have acquired are based on similar tastes with other users or items to discover if the users could understand the insides of the recommendation process. An implicit way of evaluating how transparent an evaluation is could be conducting tests based on particular tasks involving users. An example could be affecting the system in a particular way to see if the behavior changes as expected. To be concrete, one can set up a task in which the user affects the system by giving ratings only to items with a particular characteristic to see whether the recommended items will also have the same characteristic or not.

The second way of evaluating explanations is checking how good they are with respect to the *flexibility* criterion. To measure the performance of an explanation according to this criterion, one has to make use of task-based scenarios in which users give feedback via the user interface to the system stating, for example, that they no longer want to get recommendations about items with particular characteristics. The time for the system to complete such a task could be utilized as a quantitative measure unless the user interface is problematic when providing feedback.

Explanations could be evaluated according to the *trust* criterion. To measure how explanations affect the trust of users' one can use questionnaires with users. Yet, such explicit tools could be misleading and it could be a better idea to also keep track of variables related with the trust including users' loyalty (for how long and at which frequencies the user has been using the system, etc.) and sales profile if the recommender system was deployed for commercial purposes.

As the fourth criterion, explanations could be evaluated with respect to *persuasiveness*. Persuasion by evaluations could be measured as the difference in likelihood of selecting a recommended item before and after the recommendation has been delivered to the user. If the user rates the item more highly following the recommendation, one can deduce that the user is convinced by the system with the help of explanations. Another way to measure the degree of persuasiveness due to explanations could be observing if the user tends to buy or try more recommended items by using a recommender system with explanations than a system without explanations facility. The overall persuasiveness of the system could also be measured implicitly by analyzing sales profile to see if there is a significant increase.

In order to evaluate explanations according to the *effectiveness* criterion, one could measure how much a recommended items is liked by the user before and after receiving the system's prediction about the item. If the degree to which the user likes the item does not change significantly, then one can conclude that the system has been effective by providing the user the accurate recommendation. In order to analyze the role of explanations in effectiveness, one could set up tests with two recommender systems, one with and the other without an explanations facility and see how the user's degree of liking changes in both cases.

As the sixth criterion, one can evaluate explanations with respect to *efficiency*. The time to spend until the desired recommendations is delivered to the user through the interface, namely the completion time, could be used as a quantitative measure. Indirect measures including the number of inspected explanations could also be devised.

The last criterion with respect to which explanations could be evaluated is *satisfaction*. To measure users' satisfaction, one can form questionnaires to investigate if the users tend to like the system with or without explanations. One can also measure satisfaction indirectly by keeping track of users' loyalty. A qualitative way to measure users' satisfaction with the recommendation process could be observing characteristics of the users' experiences with the system until they eventually locate the desired item(s) in the interface.

Choosing the criteria exhibits certain tradeoffs. As one can observe from the definitions of the criteria, some contradict with each other like persuasiveness and effectiveness. Another example could be that the systems providing high degrees of transparency may lack having much efficiency. In design of explanations, the goal of the system should be taken into consideration together with certain properties of the problem domain. As an example, persuasiveness (balanced by trust) could be a more important criterion in online shopping than in a movie recommender system since for the latter effectiveness (together with user satisfaction) may be regarded as crucial to reach the goal of introducing the user with items both unseen and also similar to their tastes.

C. Presenting Recommendations and Explanations

The way the recommendations are presented affect the explanations and some particular recommendation representations combine the recommendations and the explanations altogether.

1) *Top item:* The best item is presented to the user with an explanation. For example, a user who is keen on sports and swimming, in particular, could be recommended recent news about the results of a swimming contest together with an explanation like *"You have been following a lot of sports news, and swimming in particular. This is the most popular and recent item from the championship."*

2) *Top-N items:* The top-N items with highest predicted ratings are presented to the user. Suppose that the user in the previous example is also interested in politics but not at as much as sports. Therefore, the system might present sports news together with a couple of recent political analysis to the user with an explanation like *"You have watched a lot of sports and politics news. You might like to see the results of the local swimming contests and the featured article of the day about the intervention in Libya."*

3) *Similar to top item(s):* The system might list similar items to the already listed ones with an explanation. This approach is generally adopted in online shopping. Customers who bought a number of items could be recommended to buy other items by presenting an explanation such as *"People who bought these also bought …"* or *"You might also like to buy … which is similar to the ones you have already bought."*.

4) *Predicted ratings for all items:* Instead of presenting the user a limited number of items as recommendations, the system may allow them to see the predicted ratings for all items, i.e. the items with low predicted ratings as well with explanations. This way the user may also receive explanations about why an item is predicted to have a low rating. If the user of the previous examples does not like football, they could receive an explanation like *"This is a sports item, but it is about football. You do not seem to like football!"* about a football story.

5) *Structured overview:* In order to allow displaying trade-offs between recommended items, the best item suiting user's needs and/or characteristics could be listed at the top and below it other alternatives having particular trade-offs could be listed certainly with explanations. This representation combines recommendations and explanations integrally. A user of an online shopping system could be recommended a camera that best fits their needs, and the rest of the cameras could be listed as *"[this camera]... is cheaper but has less resolution and poorer zooming capacity."* by explicitly stating the trade-off. Structured overview presents users several items of a particular category and increases efficiency by easing navigation and user comprehension of available options.

The recommender system may present the user with recommendations they might already know about to inspire trust, or may supply them more serendipitous recommendations to increase user satisfaction. Recommender systems could be bold in the sense that they know that the user will like to item to a certain degree, or they could state that they are sure about the recommendation they have made. These factors are part of the recommendation process and should be taken into account while presenting explanations as well.

D. Interacting with The System

There are various ways in which a user can give feedback to the system to take part in the recommendation flow.

- *The user specifies their requirements directly:* By implicitly participating in data collection process the user could tell the system what they demand. Another way of doing that could be providing the required facilities in the user interface of the system so that the user could succeed in interacting with the system. Such a facility could even allow natural language processing and the user could specify their requirements through conversations with the system.
- *The user asks for an alteration:* Like in the structured overview presentation, the user might demand the system recommends them another item having more or less a particular characteristic than the already recommended item. For example the user might desire to be recommended a similar laptop to the one they are already presented but one that is cheaper and that could have less processing power via an online shopping system interface.
- *The user rates items:* This is the most typical way of giving feedback to the system. Most movie recommender systems require that users have to rate a certain number of movies in order to start receiving recommendations about unseen items.
- *The user states their opinion:* If the user likes an item they are trying, they could ask through the interface to be recommended more items of this type. The interface could provide additional options including that the user could specify if they would like receiving more recommendations about similar items currently or later. In the same way, the user could specify that they do not want items like the recommended one. They could ask for a diversification or could state total rejection for the particular type. Finally the user could demand to be recommended a serendipitous item as well.

The way the recommendations and explanations are presented could further be extended with facilities providing user-system interaction as indicated above. Following the general framework outlined up to this point, we will continue with mechanisms that generate explanations in the next section.

III. TYPES OF EXPLANATION

Recommendations provide user the items they might like or predictions about items that the user queries about. Explanations of recommendations deliver the user the adequate information why they might like the recommended items or why they are given a particular prediction about an item. As shown in the previous section, there are various ways in which the user could receive explanations.

There are various types of explanations with relations to the mechanisms that generate recommendations and explanations [5][6]. The direct relations between users and items are unknown. In generating explanations particular *intermediary entities* are utilized to understand the relations between the active user and the item of interest. This technique is illustrated in Figure 1.

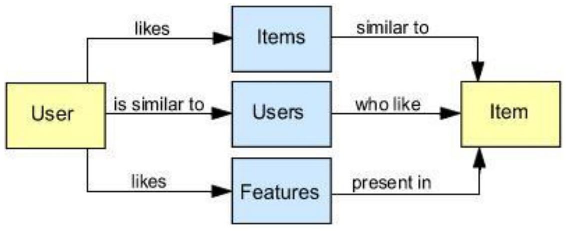


Figure 1. The user is related to the item via intermediary entities.

Explanations are categorized according to the intermediary entities they utilize:

- *Item-based explanations:* Other items rated by the user which are similar to the item for which the prediction is generated are utilized as intermediary entities to form explanations, as depicted on the first line in Figure 1. The active user receives explanations like *“(...) because you rated similarly [items used as intermediary entities]”*. This approach is adapted by particular movie recommender systems such as Netflix.
- *User-based explanations:* Other users similar to the active user who rated the item for which the prediction generated are utilized as intermediary entities to form explanations like *“(...) because [users used as intermediary entities] like you rated the item similarly”* as depicted on the middle line in Figure 1. This kind of explanations is adapted in certain scientific researches.
- *Feature-based explanations:* Particular features of the recommended item that the active user likes are utilized as intermediary entities to form explanations like “(…) because you like [features used as intermediary entities] present in the item.”, as depicted on the last line in Figure 1. This approach could be utilized in the movie recommendation domain where users may receive why they have been delivered particular recommendations on the basis of their degree of liking the director, the genre, the actors and other characteristics of recommended movies.

There are several benefits and shortcomings of each approach. Item-based explanations improve users’ satisfaction with the recommendation and help users to make more accurate decisions, yet users receiving this kind of explanations may not understand the relations between the recommended item and the explaining items. User-based explanations contribute to persuasiveness, yet they are less effective in helping users make accurate decisions. Feature-based explanations pose several challenges due to limited content analysis whenever multimedia items are recommended, and the results of the content analysis could occasionally be regarded as too low level since there are utilized particular techniques to extract the features by keeping track of frequencies of keywords contained in the item.

IV. EXPLAINING RECOMMENDATIONS USING TAGS

Vig et al. [5] propose utilizing tags formed by users to generate explanations. Their approach called *tagsplanation* makes use of a tag or a set of tags as intermediary entities. This type of explanations is developed for MovieLens, the online movie recommender system which has been in use since the second half of 1990's. The system includes millions of ratings for thousands of movies by thousands of users. MovieLens has been one of the pioneer systems developed through the recommender systems research.

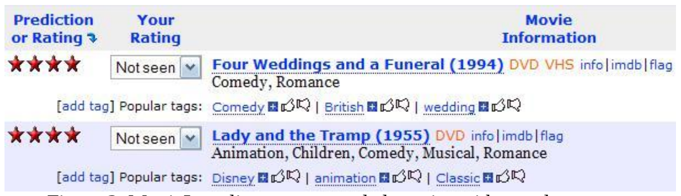
Tag usage is currently popular online. Most web systems allow their users to tag items they present. However, tags pose certain challenges:

- *Tag relevance:* The relationship of the tag with the item is a key component of tagsplanations. Tag relevance specifies the degree to which the tag can represent the item.
- *Tag preference:* The relationship of the tag with the user is the other key component of tagsplanations. Tag preference implies how much the user likes the category marked by the tag considering the item which has the tag.

Tag-based explanations are inspired by certain benefits of the existing approaches and the adapted approach tries to abstain from the shortcomings described previously. Tag-based approach adapts the rating scale utilized in the item-based approach to be applied on tags by users yet through making use of user tags about items it avoids confusions of users about the explanations generated. Tagsplanations try to address a solution for the lack of effectiveness in user-based explanations. Tag-based explanation generation is similar to feature-based approach by utilizing tags with ratings and frequencies as features. However, it is different from the latter in the sense that it also deals with eliminating low quality or redundant tags and user tags can be said to have better quality than the keywords obtained through limited content analysis of items. Tags offer the possibility to generate explanations having more cognitive values since they reflect users' understandings of movies by their nature.

The aim of tagsplanations is providing *justifications* rather than *descriptions*. Descriptions reveal the actual mechanism that generates recommendations and are means of ensuring high degrees of transparency, yet they may be irrelevant, confusing, or too complex for the purposes of users. On the other hand justifications convey a conceptual model that may differ from the insides of the algorithm. Although adapting that way one has to keep considerations over transparency low, choosing justifications versus descriptions provides a degree of freedom in designing the mechanisms that generate explanations than the recommendation algorithm. This is especially useful as designing explanations can be performed as a module and integrating the module to the rest of the recommender system can be managed without increasing complexities of the recommendation algorithms. Moreover explanations gain more importance as they are more meaningful than crude descriptions of algorithmic mechanisms when they are generated to provide justifications about the generated recommendations.

Most popular tags are presented with recommendations on MovieLens website, as depicted in Figure 2.


Prediction or Rating | Your Rating | Movie Information

★★★★ | Not seen | Four Weddings and a Funeral (1994) DVD VHS info|imdb|flag
Comedy, Romance
[add tag] Popular tags: Comedy | British | wedding

★★★★ | Not seen | Lady and the Tramp (1955) DVD info|imdb|flag
Animation, Children, Comedy, Musical, Romance
[add tag] Popular tags: Disney | animation | Classic

Figure 2. MovieLens lists recommended movies with popular tags.

Users could vote for or against the adequacies of the tags presented under each recommended movie. Users' votes determine interactively the *tag popularity*. They could insert new tags for movies through the interface depicted in Figure 2 as well.

Tag preference could be measured by directly asking users their opinions about the tags presented in particular movies. Yet practical rejections may be raised against adapting this kind of approach because even though adequate features were provided by the interface, users do not have to use them and even when they use them they may not rate a substantial numbers of tags.

Tag preference could be inferred based on the ratings each user has given to movies. First a weighted average of the user's ratings of movies with the particular tag is computed. Next *tagshare* of the tag, the number of time the tag is applied to a movie divided by the total number of tags of the movie, is calculated to be used as the weight factor. User's preference of the tag is computed according to a formula which returns a manipulation of the tagshare of the tag and the user's ratings for the movies with the tag in (0, 5) interval similar to the rating scale. If the user has not rated any movies with the tag, then the tag preference is unknown.

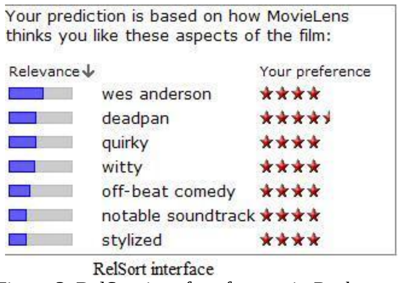
Tag relevance is computed by calculating the similarity between the tag preference of the user and the rating of the movie by the user via applying Pearson correlation formula, which is a well-known and extensively-utilized similarity metric in CF algorithms [1]. This approach enables the employment of a continuous scale rather than a binary one such as <relevant, not relevant> since it is more meaningful to concern the degree of relevance in a more detailed fashion.

Tagpslanations differ from traditional feature-based explanation techniques in the sense that tag filtering is an important component of their designs. Filtering tags is realized based on the quality of the tag, tag redundancy, and the usefulness of the tag for explanation. To deduce the quality of a tag, it is checked against particular constraints including adequate popularity, and a minimum threshold related to total number of times it is rated by users. If these constraints do not hold for the tag, it is eliminated. In order to understand whether a tag is redundant, it is checked against its possible synonyms such as (film, movie) pair and different words for the category it implies such as (violence, violent) pair. One of these tags is eliminated and the user forming the eliminated tag is supposed to form the other tag as a consequence. Lastly, tags whose preferences are undefined or relevance is very small are eliminated as they are not useful for generating explanations.

Tag-based explanations have been evaluated by conducting experiments involving users of MovieLens [7]. Users are asked questions about the explanations presented in distinct interfaces depicted in Figure 3, Figure 4, Figure 5, and Figure 6. The questions asked users to rate three proposals about each interface to evaluate the system with respect to the following criteria:

- The proposal to be rated by the participants of the experiment in order to evaluate the system with respect to *justifiability* is *"This explanation helps me understand my predicted rating."*
- To evaluate the system with respect to *effectiveness*, the users are asked to rate the proposal *"This explanation helps me determine how well I will like this movie."*
- In order to evaluate the system according to the *mood-compatibility*, which measures how well the generated explanation fits with the user's temporal feelings, situation, etc, the participants are asked to rate the proposal *"This explanation helps me decide if this movie is right for my current mood."*

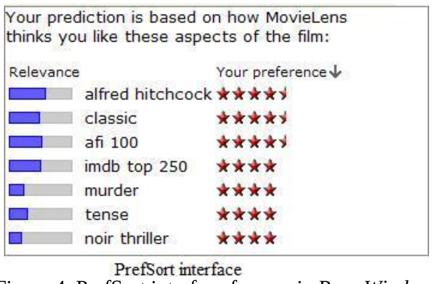
In Figure 3, tags of recommended movie *Rushmore* utilized in the generated explanation are sorted with respect to relevance and for each tag user's preference is depicted using a 5-star representation. The interface is called *RelSort*. In Figure 4, tags used in the explanations for movie *Rear Window* are sorted according to preference and the relevance is also included in the interface called *PrefSort*. In Figure 5, tags for movie *The Bourne Ultimatum* are shown only according to relevance in *RelOnly* interface and in Figure 6, only the preferences of the tags for movie *The Mummy Returns* are depicted in the interface *PrefOnly*.


Your prediction is based on how MovieLens thinks you like these aspects of the film:

Relevance↓		Your preference
	wes anderson	★★★★
	deadpan	★★★★½
	quirky	★★★★
	witty	★★★★
	off-beat comedy	★★★★
	notable soundtrack	★★★★
	stylized	★★★★

RelSort interface

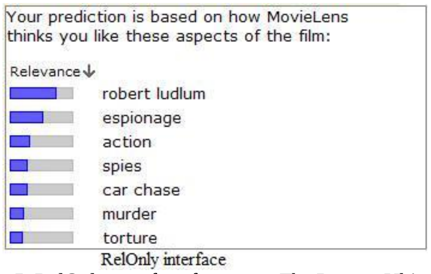
Figure 3. RelSort interface for movie *Rushmore*.


Your prediction is based on how MovieLens thinks you like these aspects of the film:

Relevance		Your preference↓
	alfred hitchcock	★★★★½
	classic	★★★★½
	afi 100	★★★★½
	imdb top 250	★★★★
	murder	★★★★
	tense	★★★★
	noir thriller	★★★★

PrefSort interface

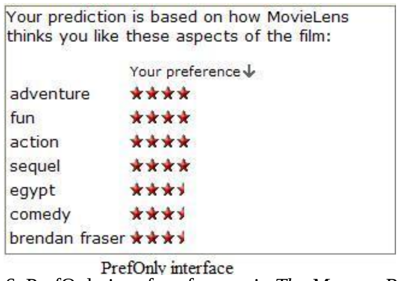
Figure 4. PrefSort interface for movie *Rear Window*.


Your prediction is based on how MovieLens thinks you like these aspects of the film:

Relevance↓	
	robert ludlum
	espionage
	action
	spies
	car chase
	murder
	torture

RelOnly interface

Figure 5. RelOnly interface for movie *The Bourne Ultimatom*.


Your prediction is based on how MovieLens thinks you like these aspects of the film:

	Your preference↓
adventure	★★★★
fun	★★★★
action	★★★★
sequel	★★★★
egypt	★★★½
comedy	★★★½
brendan fraser	★★★½

PrefOnly interface

Figure 6. PrefOnly interface for movie *The Mummy Returns*.

Results are listed in Figure 7 based on the percentages to which the users either *strongly agree* or *agree* with the proposals given for the criteria.

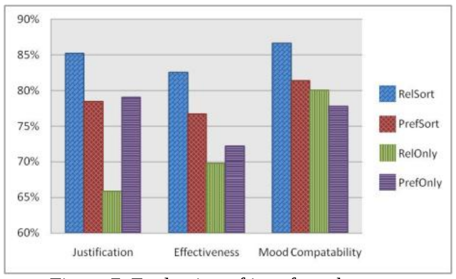


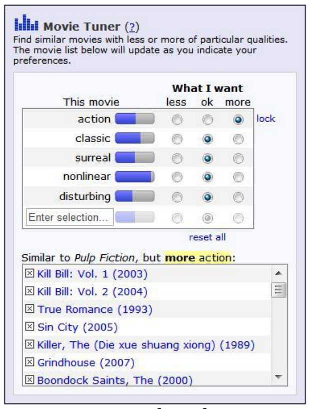
Figure 7. Evaluation of interfaces by users

Users' evaluation of explanations imply that tag preference is more important than tag relevance for justifying recommendations since the interface PrefOnly attains higher percentages than RelOnly according to the justification criterion. However, users preferred the tags to be sorted by relevance as RelSort has a higher percentage than other interfaces. According to effectiveness criterion, tag preference and tag relevance appear to have roughly equal importance as PrefOnly and RelOnly interfaces are evaluated to have close percentages. Users evaluated RelSort interface as the most effective one. Although tag relevance and tag preference appear to be equally important according to mood-compatibility criterion, it can be deduced that relevance plays its most important role in mood-compatability since RelOnly attains its highest percentages in that evaluation metric. Participants of the experiment rated the RelSort interface better than others according to three criteria.

J. Vig et al. also conducted an experiment to evaluate which kind of tags the users tend to like most in generated explanations. The results have shown that users find subjective tags more important than factual tags in all categories. However, in certain cases factual tags outperformed subjective ones such as the users preferred the factual tag *sexuality* more important than the subjective tag *sexy*. Users consider general factual tags like *World War II* more important than specific factual tags like *Manhattan*, and descriptive subjective tags like *surreal* and *dreamlike* more important than subjective tags with sexual themes or tags with opinions without descriptions like *magnificent* and *brilliant*.

Almost 82% of the participants rated the generated explanations as good overall. Thus it can be concluded that tagsplanation as an extension of the existing explanation generation approaches is successful in fulfilling its aims of justifiability and effectiveness and brings additional value to MovieLens.

J. Vig (2010) discusses the requirement of certain future work in order to extend tagsplanations to achieve other aims including scrutability [4]. MovieLens develops a new interface called the Movie Tuner that allows users to change the recommendations they receive. A sample screenshot is included in Figure 8.


Movie Tuner (?)

Find similar movies with less or more of particular qualities. The movie list below will update as you indicate your preferences.

	What I want			
This movie	less	ok	more	
action			●	lock
classic		●		
surreal		●		
nonlinear		●		
disturbing		●		
Enter selection...		●		

reset all

Similar to *Pulp Fiction*, but **more** action:

- Kill Bill: Vol. 1 (2003)
- Kill Bill: Vol. 2 (2004)
- True Romance (1993)
- Sin City (2005)
- Killer, The (Die xue shuang xiong) (1989)
- Grindhouse (2007)
- Boondock Saints, The (2000)

Figure 8. Movie Tuner interface for movie *Pulp Fiction*.

Movie Tuner interface injects a conversational aspect to MovieLens. Users are presented tags that are relevant to them as explanations. Tag relevance is also used to sort movies in order to answer queries like *"more action than Pulp Fiction"* more quickly. Movie Tuner chooses and lists tags utilizing a regression-based machine learning technique. This is an improvement achieved over the approach based on similarity computation to calculate the tag relevance previously. In order to select candidate tags to be displayed in user critiques (in the interface), an *entropy-based approach* to divide the space of neighbouring movies when used in critiques and a *relevance-based approach* to choose the tags that have the highest relevance to the recommended movie. A new algorithm to provide scrutability is developed to ensure that newly recommended items will significantly have the characteristics implied by the tag more or less as specified by the user.

In Figure 8, tags selected by the system are depicted to the user for movie *Pulp Fiction*. Users could query about other tags existing in the system through entering them in the text box *"Enter selection"*. The interface enables users to demand other movies having *more* or *less* of the category implied by a particular tag to be recommended to them. They could also combine critiques by clicking *lock* in the interface to demand for example *"more classic and less surreal"* movies than Pulp Fiction to be recommended.

Movie Tuner interface has brought additional aims that could be achieved by utilizing explanations, it particularly allows users to criticize the recommendations they have received and to ask for an alteration based on the particular criteria they supply. This is an achievement of tag-based explanations as it has been shown that they could unite important aims to be accomplished together. The degree of success of Movie Tuner has not been announced yet, as it is being evaluated by users according to various criteria.

V. Justifiable and Accurate Recommendations

P. Symeonidis et al. (2009) develop a new movie recommender system which they call MoviExplain [2]. MoviExplain combines collaborative filtering with content-based filtering to adapt a hybrid recommendation algorithm and similarly generates explanations by combining *influence* (user-based or item-based) and *keyword* (feature-based) explanation techniques.

MoviExplain relies on user's ratings of movies. Through ratings it infers users' possible votes about particular features of the rated movies. By using these features, it builds *feature profiles* for users. The clusters for users such as *users that prefer comedies* are generated to reason about collective preferences of whole communities. The generated explanations of MoviExplain are of the form *"Movie X is recommended because it contains features a, b ... which are also included in movies Z, W ... you have already rated"*. If these features occur frequently in the user's feature profile, than it could be utilized as evidence for justifying recommendations. Feature extraction is performed by making use of the Internet Movie Database (IMDB) as the knowledge-base.

The recommendation algorithm applies in stages. First user groups are created. Next the feature-weighting is performed and the neighborhood is formed. Lastly the recommendation and justification (explanation) lists are generated. These lists are presented in MoviExplain's interfaces online.

MoviExplain is evaluated with respect to statistical precision and recall and the results are compared to the ones obtained by evaluating particular hybrid recommender systems which proved to be successful previously. MoviExplain is claimed to attain better precision than similar systems [2] as it uses a clustering approach and detects particular matches among the preferences of users. Furthermore, MoviExplain achieves better explain coverage values than other systems because it is based on the notion of groups of users whilst other systems work on individual users.

Explanations generated by the system are evaluated by surveys conducted by users as well. The participants are asked to rate five movies before receiving recommendations. Then they are asked to rate each recommendation based on influence, keyword and hybrid explanation approaches. Lastly they are asked to rate each recommended movie after receiving the explanation. Results obtained through the survey indicate that the hybrid explanation enables both accuracy and justifiability.

Developing hybrid approaches for generating recommendations has been discussed to improve overall accuracy. The study over the system MoviExplain has shown that hybridizing existing approaches could help in explanation technology since it is possible to attain both accurate and justifiable recommendations through generating hybrid explanations.

VI. Diversification Based On Recommendations

Recommender systems are occasionally faced with problems of *overspecialization*, that is recommended items are too similar to each other and the aim of introducing users with yet-unseen items that they may not encounter themselves is not satisfied. In order to overcome this problem a “flavor” of *diversity* should be added to the list of recommended items.

The goal of *recommendation diversification* is to recommend items that are dissimilar with each other but still fit with user’s tastes and preferences. Therefore certain trade-offs are taken into account so that diversification will not result in recommending users irrelevant items. C. Yu et al. (2009) show that explanations could be utilized in performing diversification [6].

The notion of similarity of explanations between distinct items can be conceptualized as the *diversity distance*. Certain correlations between the recommendation algorithm and explanation-based diversity for a list of recommended items exist. Applying the known similarity metrics utilized in the recommendation algorithm to the explanations for distinct items, one can calculate the diversity distance since explanations consist of a list of similar items and similar users.

Based on the notion of diversity distance between items, Yu et al. develop efficient algorithms for generating recommendations which achieve a good balance between relevance and diversity. Details of the algorithms and evaluation techniques are described in [6]. The results of their evaluation indicate that the proposed method indeed achieves its goals.

VII. Conclusions

Traditional statistical methods to evaluate the performance of recommender systems are not considered to be adequate. Since recommender systems are widespread and deployed with well-designed user interfaces, user-centered criteria should be devised to test the user’s relations with the system.

Generating explanations for recommendations has emerged to compensate for the need of increasing human-system interaction and bringing cognitive aspects to the recommender systems. Explanations provide several aims and distinct problem domains for which recommender systems could be developed. In order to evaluate explanations, surveys will be utilized to be carried out with a critical mass of real users.

Diversification could be attained using algorithms based on the notion of similarity between explanations or allowing flexibility by designing interfaces which include facilities through which users can state their opinions and ask for alterations to the recommended items. Interfaces providing flexibility have additional benefits as they increase the importance of cognitive aspects in recommender systems.

Explanation technology is open to contributions including new approaches such as tag processing, hybrid approaches and the notion of similarity between explanations to solve problems arising from recommendation algorithms. The state-of-the-art implies that explanations will be an integral part of all large scale commercial recommender systems both to increase results obtained by users’ evaluation of the system and provide material to improve particular problems exhibited by extensively-used recommendation algorithms.

References

[1] G. Adomavicius and A. Tuzhilin, “Towards the next generation of recommender systems: a survey of the state-of-the-art and possible extensions,” IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 6, 2005, pp. 734-749.

[2] P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos, “MoviExplain: a recommender system with explanations,” Proceedings of the 3rd ACM Conference on Recommender Systems, 2009, pp. 317-320.

[3] N. Tintarev and J. Masthoff, “A survey of explanations in recommender systems,” Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering: Workshop on Recommender Systems and Intelligent User Interfaces, 2007, pp. 801-810.

[4] J. Vig, “Intelligent tagging interfaces: beyond folksonomy,” Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology, 2010, pp. 371-374.

[5] J. Vig, S. Sen, and J. Riedl, “Tagsplanations: explaining recommendations using tags,” Proceedings of the 13th International Conference on Intelligent User Interfaces, 2009, pp. 47-56.

[6] C. Yu, L. Lakshmanan, and S. Amer-Yahia, “It takes variety to make a world: diversification in recommender systems”, Proceedings of the 12nd International Conference on Extending Database Technology: Advances in Database Technology, 2009, pp. 368-378.

[7] MovieLens web site, http://www.movielens.org, [retrieved: 01, 2013].

[8] Netflix web site, http://www.netflix.com, [retrieved: 01, 2013].

Data and provenance management for climate effect studies

Adaption of climate data with distribution based scaling for hydrological simulations.

Lena Strömbäck, Kean Foster, Jörgen Rosberg
Swedish Meteorological and Hydrological Institute (SMHI)
Norrköping, Sweden
e-mail: {lena.stromback, kean.foster, jorgen.rosberg}@smhi.se

***Abstract* — Climate effect studies are currently of high interest to predict the impact of a changing climate. The results of such studies are used by decision makers as a basis for planning how to mitigate and adapt to the effects of expected climate changes. However, these studies require heavy computations on large sets of data in several steps. This combination of heavy computation and results being basis for important decisions makes it extremely important to have an efficient and well documented process for computations, to provide accurate results in an efficient way. In this paper we describe the problem and present our DBS (Distribution Based Scaling) tailoring tool that has been implemented to support the process. We discuss the problem and our solution in relation to scientific workflow systems and provenance engines in general.**

Keywords-workflow systems; climate studies; hydrology; data management; quality insurance; DBS tailoring tool; provenance

I. Introduction

Today there is a huge demand for knowledge on climate change and how this has an impact on our environment. Information on possible outcomes of climate change comes from the numerical global circulation models (GCMs). The GCMs model the climate for the entire globe, from the past into the future. Different assumptions on how the greenhouse gas emissions will evolve can thus be tested within this framework. However, the scale of the information obtained from those models is often too coarse for any impact study on a regional scale. To downscale the information from the GCMs either statistical methods or regional climate models (RCM) are used. However, to be able to use this data for hydrological predictions we need realistic input data to the hydrological model about future occurrences of rain and temperature. This requires that the regional information is even further downscaled and bias corrected to ensure that the provided data represents a realistic distribution of precipitation and temperature in time and space. Therefore historical simulations of temperature and rain are compared with historical observations to calibrate an adjustment schema which is applied to future predictions. This process is called Distribution Based Scaling, DBS.

From a computer science perspective this process involves a number of interesting challenges. First of all climate studies involves time series of daily values with a high geographic resolution. This means that we need to consider gigabytes of data that need to be efficiently stored and processed. Secondly, the input data from RCMs can have different representation, meaning that there is a need to manage and translate these huge datasets between different data formats. In addition, we need methods for quality insurance i.e. to detect and correct faulty data or errors in the processing. Finally, there is a need to document the process for further scientific development of the procedures, data and models. An important aspect of the problem is recording provenance, i.e. to record the history of the correction process making a new result comparable to older runs.

The problem in many ways resembles problems in other scientific disciplines where derived results are dependent on data from many different data sources, versions of data, and versions derived from analyses and simulations. For all these scientific areas it is extremely important to keep track of all steps in the process. One of the most common approaches for recording provenance is scientific workflows. In this field a number of workflow based tools have been implemented ([1] and [2] gives an overview.)

In this paper we will discuss the issues and requirements around DBS applications, present our current system for the process and discuss how it relates to scientific workflows and scientific workflow tools in general. The paper starts with a more thorough introduction to the DBS process. After this we give a more thorough discussion of the computational challenges and how they relate to problems addressed in general by scientific workflows. Finally, we present the main features of our implemented system and discuss how this system relates to scientific workflow engines in general.

II. Distribution based scaling

As explained in the introduction, even though output data from climate scenarios give accurate predictions on the expected long term changes in climate, they do not provide the accurate detailed information needed as input for hydrological simulations. For instance, even though the total amount of precipitation is the same for a longer period, rainfall in the climate models tend to be much less varied on a daily basis than can be expected if we compare with current observations. Therefore this data need to be adjusted according to an observed reference period to provide more accurate input.

In the case of precipitation, the DBS approach uses two steps: (1) spurious drizzle generated by the climate model is removed to obtain the correct percentage of wet days and (2)

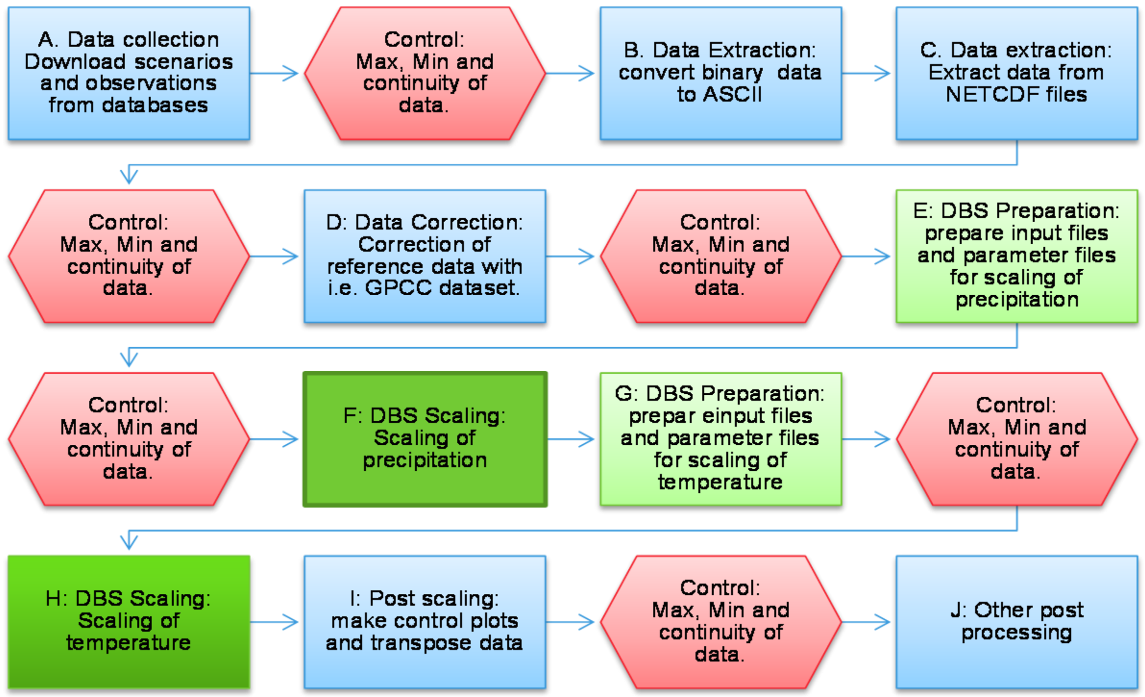

A. Data collection Download scenarios and observations from databases
Control: Max, Min and continuity of data.
B. Data Extraction: convert binary data to ASCII
C. Data extraction: Extract data from NETCDF files
Control: Max, Min and continuity of data.
D: Data Correction: Correction of reference data with i.e. GPCC dataset.
Control: Max, Min and continuity of data.
E: DBS Preparation: prepare input files and parameter files for scaling of precipitation
Control: Max, Min and continuity of data.
F: DBS Scaling: Scaling of precipitation
G: DBS Preparation: prepar einput files and parameter files for scaling of temperature
Control: Max, Min and continuity of data.
H: DBS Scaling: Scaling of temperature
I: Post scaling: make control plots and transpose data
Control: Max, Min and continuity of data.
J: Other post processing

Figure 1. A schematic figure of the general scaling process. The dark green boxes (F, H) correspond to the actual DBS scaling, while the lighter green boxes (E, G) correspond to preparation of statistical information (step 1, 2, 4 and 5 in the process).. Blue (A, B, C, D, I and J) represent other necesseray preparation and conversion of the raw input data to the process as well ass post processing needed to prepare the data for different hydrological engines. Finally, the read diamonds represent data quality controls that are necessary to detect errors in the data transformation process.

the remaining precipitation is transformed to match a sample for current observed frequency distribution. To obtain the percentage of wet days correctly, a threshold is identified for each sub-basin and season. The sub-basin represents the geographical resolution and for Europe we typically work with around 40 000 sub-basins. Days with precipitation amount larger than the threshold value were considered as wet days and all other days as dry days [3], [4], [5]. After this the temperature from the simulation is adjusted based on the new precipitation and an observed reference period for the temperature.

This means that a typical DBS process can be broken down into six general steps:

1. Calculate statistical parameters for the wet day threshold and distribution of the observed precipitation.
2. Calculate statistical parameters for the wet day threshold and distribution of the precipitation provided by the climate model.
3. Scaling of the precipitation from the climate model.
4. Calculate statistical parameters related to observed temperature.
5. Calculate statistical parameters related to the temperature provided by the climate model.
6. Scaling of the temperature from the climate model.

In practice the process includes several steps of data conversion as climate simulations and reference data may occur in different formats. In addition, to ensure a correct result from the process it is also important to perform quality control between the steps in order to detect errors as early as possibly in the processing. A schematic picture describing the general process is given in figure 1.

III. Problems and Challenges

In this section we discuss the main challenges we face for an efficient data management for the DBS scaling process.

A. *Large datasets*

For each step in the processing chain we are faced to manage and transform very large volumes of data. As an example, our hydrological model [6] represents Europe by approximately 40 000 sub-basins. The model uses time series representing daily values for temperature and precipitation as input for simulating hydrological conditions. For climate impact studies a typical time series represents 100 years of daily values. This results in gigabytes of data that need to be efficiently processed. This large volumes of data put high requirements on data storage as well as efficient processing.

B. *Long processing chains and processing time*

As seen in the schematic picture in figure 1, the processing chain consists of several steps. In reality, most of the included boxes can be further broken down to several individual steps. In many of the steps the processing time is long resulting of computation times of days for the whole chain.

C. Data formats and data conversion

The data formatting and conversions can involve several different tasks; for example, to select relevant data in time and space from the simulated climate model, to convert data between different geographic representation (e.g. grid vs. sub-basin), to convert data between different data representations (e.g. NetCDF vs. ASCII). As the data quantities are large each of these processes has to be efficient. Moreover, in many cases the order of conversions affects the efficiency of the process.

D. Quality assurance

As we are working with research, data and processes are under constant development. This means that quality assurance is extremely important. Errors may occur from faults in the observed or climate data, but also by mistakes in the selection or data conversion process. In many cases strange values of data origins from new geographical conditions and it is important to analyze to further improve the scaling method. Thus, detecting and determining the source of error or strange value is critical. As the total computation time for the chain is long it is important to do this as soon as possible to avoid delays in producing results. Therefore, it is crucial to check data quality after each of these steps.

E. Reproducibility

The final result of the DBS process is used for hydrological simulations and in many cases published or exported to a customer. For comparison with other similar results, future reruns with updates of the model or discussions about the validity of results, it is extremely important to store a record of the whole processing chain, i.e. the provenance of the final result.

F. Cooperation

Due to the long processing chains there is often a need for cooperation of researchers to produce one result. In many cases we need to keep records of old data and details of a run to train new researchers in performing the scaling process. Therefore recording of details around the process is very important.

These challenges in many ways address the same problems as scientific workflows or provenance management systems [1], [2]. However, there are also some main differences. Although our process chains include variation, it is in general more static than processes represented in scientific workflows systems. Moreover, the large volumes of data and long execution times for our process must be taken into account when designing a tool. We will further elaborate on these differences after presenting the main features of our implemented system.

IV. THE DBS TAILORING SYSTEM

The DBS tailoring system is used for facilitating bias correction/downscaling using the DBS approach described above. The DBS tailoring system is used both to prepare the different data and control files needed to perform a DBS bias correction/downscaling and run the DBS motor. As described in section II a typical DBS job can be broken down into six general steps. Around these main steps we need supporting modules for data conversion and quality control. The DBS tailoring system helps to create the required files for the different steps and controls the process. In this section we give an overview of the system, for a more detailed description see [7].

A. General Architecture

The general architecture for the system is, as described in figure 2, a general process engine and a set of different modules for the different processing steps or tasks. In the general engine, the user can define the processing chain and put together the different tasks as desired. The processing chain is described by XML files that define how to perform the different tasks in the chain. These XML files can be directly defined by the user but there is also a graphical user interface for support. This means that the different modules are loosely coupled and that we achieve a high flexibility in how to combine the processes to achieve the desired functionality.

In addition, the design of the system gives a high flexibility in supporting new modules. The general process engine is implemented in Java, and processes can be implemented in any programming language that can be called from Java. Currently the process modules are implemented in Java, for data conversion, or Fortran for the core DBS scaling. However, the general architecture makes it easy to integrate new process modules in the system independently on whether these are fetched from existing libraries or whether they are developed by our research team.

B. XML process scripts

One of the most important features of the system is the process chain description used for controlling the execution. The process script are used to document a run, but is also the key for reproducibility, as it contains all details about the process, such as versions of data and processes, that is needed to rerun a process. The example in figure 4 shows the principles behind the process description. In the example we can find the six steps involved in the DBS scaling, parameter preparation and scaling for the precipitation and then corresponding for the temperature. Between these main tasks

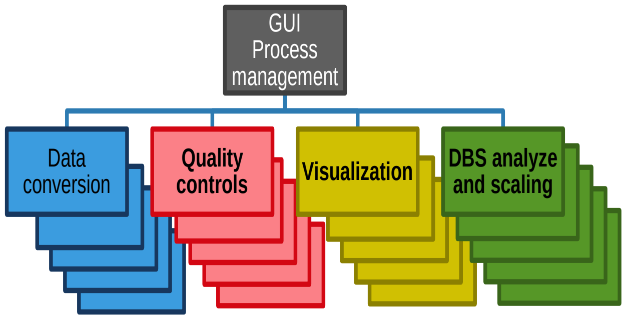


Figure 2. A schematic description f the general architecture for the DBS tailoring system. The system consist of a graphical user interface and process managementt system and a large number of modules for data conversion, quality control, visualization and the DBS scaling. The user can use the GUI to design the desired process chain.

```
- <Tasklist>
+ <Task Tasktype="LOAD" Taskgroupid="1" Taskorder="1">
  …
+ <Task Tasktype="EXPORT" Taskgroupid="1" Taskorder="4">
  …
- <Task Tasktype="DBS" Taskgroupid="1" Taskorder="9">
  <Critical_task>true</Critical_task>  >
  <DBStype>REFP</DBStype>  </Task>
  …
- <Task " Tasktype="DBS" Taskgroupid="1" Taskorder="12">
  <Critical_task>true</Critical_task>
  <DBStype>SIMP</DBStype> </Task>
  …
- <Task Tasktype="DBS" Taskgroupid="1" Taskorder="15">
  <Critical_task>true</Critical_task>
  <DBStype>SCALET</DBStype> </Task>
  …
- <Task Tasktype="DBS" Taskgroupid="1" Taskorder="18">
  <Critical_task>true</Critical_task>
  <DBStype>REFT</DBStype> </Task>
  ….
- <Task Tasktype="DBS" Taskgroupid="1" Taskorder="21">
  <Critical_task>true</Critical_task>
  <DBStype>SIMT</DBStype>  </Task>
  …
- <Task Tasktype="DBS" Taskgroupid="1" Taskorder="24">
  <Critical_task>true</Critical_task>
  <DBStype>SCALET</DBStype> </Task>
  …
- <Task Tasktype="PLOT" Taskgroupid="1" Taskorder="28">
  …
- <Task Tasktype="PLOT" Taskgroupid="1" Taskorder="31">
- <Task Tasktype="FLIPPER" Taskgroupid="1"
        Taskorder="32">
- <Task Tasktype="FLIPPER" Taskgroupid="1"
        Taskorder="33">
  </Tasklist>
```

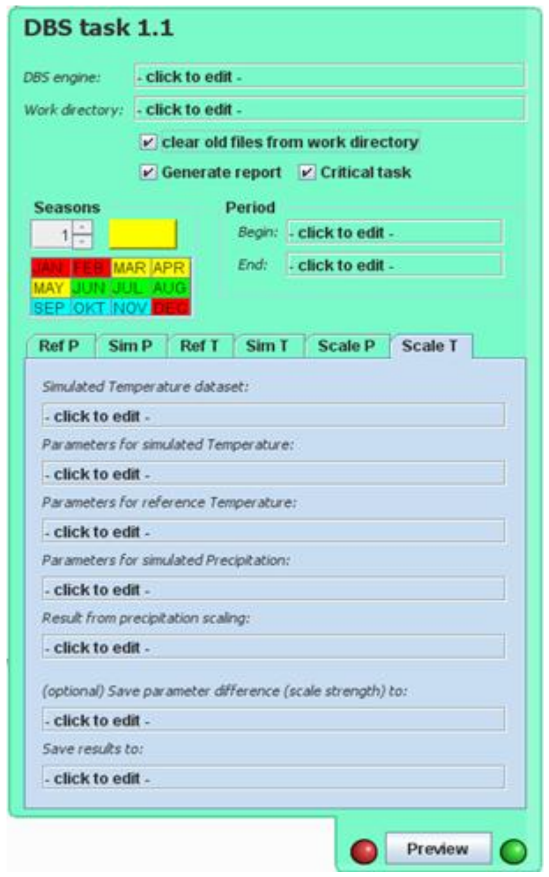
Figure 4. An example of the XML script describing the DBS Tailoring process. The example shows a typical process schema for DBS scaling. The example have been shortened to save space and improve readability. In principle detailes about the rpocesses have been removed. In addition we have removed many LOAD and export taks, these placement of these is marked with dots.

there are a number of supporting processes. Here LOAD and EXPORT is used to define and extract data needed for the process, PLOT derives maps for validating the results and finally FLIPPER provides matrix transposition of data, as needed to prepare it for the hydrological simulation.

C. *User Interface*

The DBS tailoring system provides a user interface to manage the XML script files. From the user interface the user can define the sequence of modules by selecting the desired module types. For each module type the interface provides a menu that prompts the user with required values and data. Figure 3 shows an example from the he final DBS step, the scaling of the temperature.

To further aid the user old XML process files can be loaded into the system and used as templates for defining new jobs. This is beneficial as the processes are similar. In principle there are a few types of template processes depending on the data types of input and output data and for such cases, the end user only need to alter paths to actual data files and periods for calculations.



DBS task 1.1

DBS engine: - click to edit -

Work directory: - click to edit -

[✓] clear old files from work directory

[✓] Generate report [✓] Critical task

Seasons: 1

JAN	FEB	MAR	APR
MAY	JUN	JUL	AUG
SEP	OKT	NOV	DEC

Period

Begin: - click to edit -

End: - click to edit -

Simulated Temperature dataset: - click to edit -

Parameters for simulated Temperature: - click to edit -

Parameters for reference Temperature: - click to edit -

Parameters for simulated Precipitation: - click to edit -

Result from precipitation scaling: - click to edit -

(optional) Save parameter difference (scale strength) to: - click to edit -

Save results to: - click to edit -

Figure 3. The user interface for defining last of the DBS tasks scaling the temperature dataset. The user need to enter where the DBS engine and the work folder are located. As calculations is season dependent he also need to define the seasons. The user also must enter the period for this task, i.e. dates between which calculations need to be done. Finally, the user enters he files the scaling is dependent on, note that, in this case, several files are needed more as the temperature scaling is dependent information from the precipitation files.

The interface also provides an easy environment for running the processes and monitors the results.

D. *Data conversion and efficiency*

The current implementation support geographical and time series data represented as shape files, netCDF files or plain ASCII files. ASCII files can be of three types; Discrete – non continuous data such as parameter files; Time series – ‘time’ orientated time series i.e. they have time on the x-axis; and Id series – ‘id’ orientated time series i.e. they have id on the x-axis. The implemented functionality for managing these files is LOAD and EXTRACT.

The load task allows loading file information for the different file types into the system. This step creates an index file called a POSMAP file which contains information regarding the data contained in the file i.e. number of data points, data ranges etc. These POSMAP files allows the system to identify where in the ASCII files the desired data is stored which allows for faster reading of the data.

The export task allows you to export data from a file. The task can be divided into two main types of export i.e. using geographic information and using unique ids to select the relevant data. The former type is mostly used to extract data from netCDF files for the scaling process. This task uses the geographic data in the shape file to select the nearest corresponding point in space from the netCDF file and assigns the id number from the shape file to the data at that point.

The latter export type is more versatile and can be used for a number of different tasks, for instance, to export statistics for the data to a shape file for analysis purposes, or extract a subset from an existing dataset.

This machinery gives a flexible management of data conversion. It is efficient since the load functionality with POSMAP files avoids duplicating data Also choices in conversion orders allows for optimization choices. As an example, it is often preferable to keep the grid representation of geographic information as long as possible as it has a lower geographic resolution than the sub-basin representation of data.

E. *Quality insurance*

Within the framework we typically use two kinds of quality checks, automatic statistical checks and generation of maps that can be manually inspected by an expert. The second type is demonstrated by the plot functionality in the example in figure 4. This functionality allows for plotting for instance maximal and minimal values for precipitation and temperature for each sub-basin. This allows an expert to easily inspect that the results are valid.

Another kind of control, not demonstrated by the example is statistical computations for finding flaws in the data. Examples of these are; too high or too low values; or sudden jumps in the measured or simulated values which indicate that something has gone wrong in the process.

This is important as errors can occur, for instance, by having to few observed values for one single sub-basin and period for a point. Such data can make the statistical predictions uncertain which indicate that the process has to be rerun with other input data or different parameter settings.

V. RELATED WORK

As the implemented system covers many areas there are a lot of interesting related work, for instance, within data management and efficient processing. However, in this paper we will focus on how it relates to provenance and scientific workflows. Scientific workflows and workflow based systems [1], [2], [8], [9], [10] have emerged as an alternative to ad-hoc approaches for documenting computational experiments and designing complex processes. They provide a simple programming model whereby a sequence of tasks (or modules) is composed by connecting the outputs of one task to the inputs of another. Workflows can thus be viewed as graphs, where nodes represent modules and edges capture the flow of data between the processes.

The actual features and representation of a scientific workflow differ between the systems, due to varied needs from the application areas and users they are designed for.

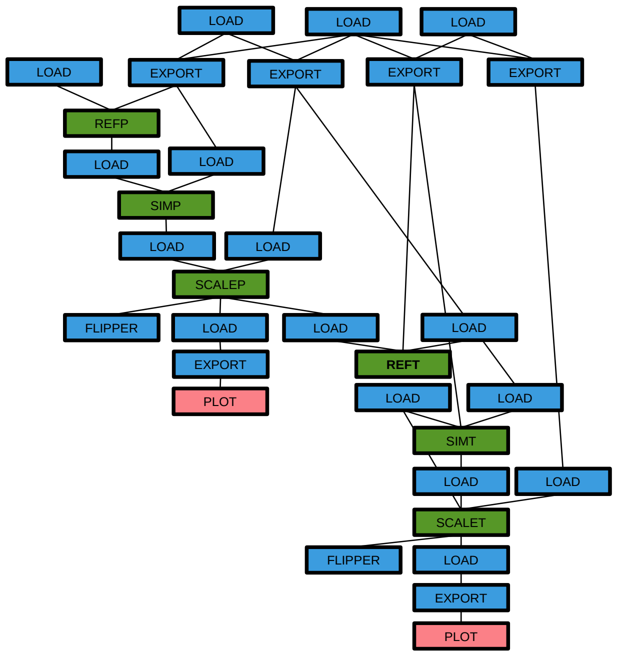


Figure 5. The sample DBS tailoring process represented as a scientific workflow. The figure gives an overview that shows the general flow of data between the processes. As in figute 1, the actual DBS scaling correspond to the green boxes, whike control plots are red. The blue boxes represent different kinds of data conversions. The blue LOAD typically creates an index for more efficient processing of the file, needed before each step in the process.

Therefor, systems tend to work in their own internal format and it is becoming common to provide conversion to other formats, e.g., the Open Provenance Model [11] and mediation approach [11]

There is a very strong relationship to our problem and scientific workflows. As discussed already in section III the basic requirements, i.e. documenting the scientific process and provenance of data is the same for our application and scientific workflows. In addition, the structure of our process chain is similar to the graph structure used for scientific workflows. Figure 5 shows schematically how the process chain in figure 2 could be represented as a scientific workflow. Although our current implementation uses an XML format and user interface that is tailored for our needs, it would be possible to translate this representation to the Open Provenance Model and thus, it can be imported to several scientific workflows systems. This would in give access to the features implemented in many of the available workflow systems. Here, we will discuss some of the features provided by VisTrails [10] which is one of the most advanced available systems.

VisTrails supports exploratory computation tasks. It has a graphical user interface that is used for the composition and execution of workflows. Data and workflow provenance is uniformly captured to ensure reproducibility of results by others. Workflows can be composed by program libraries (Python) or by external web services. VisTrails has been used in the fields of biology and earth science.

One of the most important features of VisTrails is its ability to document the provenance of the development process. This means that the system records whether one workflow is developed based on other used solutions. Even though workflows in general support cooperation between researchers, this feature further enhances cooperation as the relation between different versions is visualized by the tool. This is something not provided by our implemented system, but very useful for recording provenance and exploring the difference between workflows. Therefore it is interesting to explore how this or something similar can be included in our system in the future.

In addition to this VisTrails contains a number of interesting features, such as; the possibility to explore the parameter space of a workflow; comparing two workflows and applying the changes between them to another similar workflow; and various search and presentation facilities. All these features are very powerful when working with large collections of workflows and of interest also for our application.

VI. Future Work

The current implementation of the DBS tailoring system is in use and supports the process of preparing data used for hydrological impact studies. In a near future we will run several of these studies, on different geographic areas and based on different climate models. During this phase we will use the system for documenting the process and follow up the quality. This is a perfect opportunity to test the ability of the system and learn where it can further improve.

As the current system gives a good documentation of each process chain, but lacks information on relations between different chains this is a particular point of interest for us. Here the functionalities for scientific workflow systems in general and in particular VisTrails will be very interesting to explore further to see whether they can be adapted to our settings. In principle, two solutions are possible, extending our implementation with these features or exporting our process description to make use of an existing tool, such as VisTrails.

One of the most interesting issues to explore is how the inherent properties of our application, i.e. large data sets, long processing times and relatively static processing chains compared to many other applications where scientific workflows are used affects how these feature is realized. For instance, how can the recorded information be used for avoiding duplication of data and rerun of expensive computation processes in an optimal way.

VII. Conclusion

Climate effect studies require heavy computation and the results are being the basis for important decisions in society which makes it extremely important to have an efficient and well documented process for computations. This paper gave an overview of the problem and our DBS tailoring tool that has been implemented to support the process and compare it with scientific workflow tools in general. The comparison shows that our tool and scientific workflow tools has many common properties, such as documentation of the process, even though scientific workflows systems in general have a number of additional functionality. In the future we will investigate how we can reuse some of these features in our setting to further improve our computation process.

Acknowledgment

The development of the DBS tailoring system has been done as a part of several projects involving hydrological effect studies. The work in this paper has been funded by the Swedish Research Council (Vetenskapsrådet), The Swedish Environmental Protection Agency, The Swedish Research Council Formas and the European Commission. We are also grateful to Jonas Olsson for valuable comments on the work.

References

[1] J. Freire, D. Koop, E. Santos, and C. Silva, Provenance for Computational Tasks: A Survey. *Computing in Science and Engineering*, 2008

[2] S. Davidson, and J. Freire J: "Provenance and scientific workflows: challenges and opportunities", SIGMOD, 1345-1350, 2008

[3] D.S. Wilks, 1995 "*Statistical Methods in the Atmospheric Sciences: An Introduction."* Academic Press, INC, Burlington,MA, p. 86, 1995

[4] M. R. Haylock, G. C. Cawley, C. Harpham, R.L. Wilby, C. M. Goodess, "Downscaling heavy precipitation over the United Kingdom: a comparison of dynamical and statistical methods and their future scenarios", *Int. J. Climatol*. 26, 1397–1415, 2006

[5] W. Yang,, J. Andréasson, L.P. Graham, J. Olsson, J. Rosberg, F. Wetterhall "Improved use of RCM simulations in hydrological climate change impact studies", *Hydrol. Res*., 41, 211-229, 2010

[6] G. Lindström, C. Pers, J. Rosberg, J. Strömqvist, B. Arheimer, "Development and test of the HYPE (Hydrological Predictions for the Environment) model - A water quality model for different spatial scales", *Hydrol. Research 41* (3-4): 295-319, 2010

[7] K. Foster "*DBS Tailoring System An operators manual*". CLEO project report. SMHI, 2012

[8] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. Pocock, P. Li, *et al.*, "Taverna: a tool for building and running workflows of services*." Nucleic Acids Research*, 2006.

[9] T. Oinn, M. Greenwood, M. Addis, N. Alpdemir, J. Ferris, K. Glover, et al. "Taverna: lessons in creating a workflow environment for the life sciences," *Concurrency and Computation: Practice and Experience*, 2006.

[10] L. Bavoil, S. P. Callahan, P. J. Crossno, J. Freire, C. E. Scheidegger, C. T. Silva, *et al.*, "Vistrails: Enabling interactive multiple-view visualizations," In Proceedings of IEEE Visualization, 2005. Information Sciences Institute, "Pegasus:home,"

[11] L. Moreau, J. Freire, J. Futrelle, R. E. McGrath, J. Myers, and P. Paulson, "The open provenance model," 2008. [Online]. Available: http://eprints.ecs.soton.ac.uk/14979/1/opm.pdf (Last visited 2012-12-06)

[12] T. Ellkvist, D. Koop, J. Freire, C. Silva, and L. Strömbäck, "Using mediation to achieve provenance interoperability," in IEEE Workshop on Scientific Workflows, 2009. [Online]. Available: http://vgc.poly.edu/~juliana/pub/provinterop-swf2009.pdf (Last visited 2012-12-06)

Dr Warehouse - An Intelligent Software System for Epidemiological Monitoring, Prediction, and Research

Vladimir Ivančević, Marko Knežević,
Miloš Simić, Ivan Luković
University of Novi Sad, Faculty of Technical Sciences
Novi Sad, Serbia
e-mail: dragoman@uns.ac.rs,
marko.knezevic@uns.ac.rs,
milossimicsimo@gmail.com, ivan@uns.ac.rs

Danica Mandić
University of Novi Sad, Medical Faculty
Novi Sad, Serbia
e-mail: mandiceva88@yahoo.com

***Abstract*—We present Dr Warehouse, an extensible intelligent web-based system for epidemiological analyses. It features a data warehouse containing medical data about registered disease cases and relevant demographical data. There is also a segment of the system that is devoted to presentation and analysis of epidemiological data collected in the data warehouse. The main objectives that we set out for Dr Warehouse include intuitive visualization of epidemiological data, discovery of epidemiological information, and prediction of epidemic dynamics. In the context of epidemiological knowledge discovery, we present a rationale for developing such a system, system architecture of Dr Warehouse, its functionalities, short review of similar systems, and ideas for future development. Furthermore, we describe in more detail choices regarding data modelling, as well as some of the featured predictions and data mining based analyses.**

Keywords-data warehouse; data mining; business intelligence; epidemiological analysis; absenteeism; disease outbreak prediction.

I. Introduction

Frequent epidemics and various diseases continue to persist in modern world despite great medical discoveries and numerous countermeasures. However, the increase of medical knowledge has helped in the improvement of the overall quality and length of human life. One of the methods for battling diseases includes collection of epidemiological knowledge and its use in the prevention of outbreaks. Our main goal is to contribute to public health by building a software system that could help in the prevention and control of epidemics. This would be possible through the application of results of data analyses featured in the system. Such analyses would be executed on disease case records gathered in the system from various sources.

By following this idea and applying the latest advancements in information technology to epidemiological domain, we created Dr Warehouse – a closed source software system that supports storing of epidemiological data and offers descriptive, as well as predictive, analyses of disease outbreaks. All necessary data are stored in a specially designed data warehouse, while supported analyses include various data visualization techniques, statistical methods, data mining algorithms, and epidemic models. Results of the analyses may be accessed through a rich web client, which offers all of the analyses included in the system, or a mobile device client, which offers a subset of analyses that might be of interest to non-experts. Given the rapid rate of discovery of new analysis methods and epidemic models, we made the system extensible and ensured that new types of analyses may be easily added.

The rest of this paper is organized as follows. Section II looks into our motivation for building such a system. The overview of the system and its components is given in Section III. Some of the predictive analyses supported by the system are presented together with sample results in Section IV. Section V offers a review of similar software systems and their comparison to Dr Warehouse. Section VI includes concluding remarks and ideas for further research.

II. Motivation

A system that could provide its users with a piece of information important in the prediction of epidemics or understanding of disease dynamics would offer many indirect benefits including saving of lives, reduction in treatment costs, and decrease of everyday stress. However, we are also motivated by two more specific reasons: modernization of the healthcare system in Serbia and impact of absenteeism on the economy.

As outlined in the national development strategy [1], the Serbian healthcare system is undergoing a significant transformation. Many segments of that system are being modernized and redesigned to rely more on electronic records as opposed to traditional paper records. Moreover, the expected interconnection of healthcare centres would allow a better electronic access to medical data and consequently better conditions for data analyses, as in the case of the health information system (HIS) for the Serbian Ministry of Defence [2]. In such circumstances, Dr Warehouse could be integrated into the main healthcare system and used for epidemiological analyses. The main system would only be utilized as a data source in the extraction of necessary data, which, after several processing steps, would be stored in the data warehouse within the Dr Warehouse system. Dr Warehouse has been developed also as a potential pilot solution that should demonstrate advantages of using a business intelligence (BI) system in the healthcare domain. It is primarily applicable in activities of institutions that concern themselves with disease prevention in a population, such as institutes of public health.

Besides the fluctuation of labour, absenteeism, which is defined as “failing to report for scheduled work” [3], is the most important parameter that should be monitored by human resources managers in order to increase production potential. This is the case because high absenteeism has negative impact not only on colleagues and superiors, who must cope with greater workloads, but on the profit of a company as well. According to the research from 2009 led by the Chartered Institute of Personnel and Development (CIPD) from Great Britain [4], the most important reasons for the short-term absence from work (4 weeks as maximum) are: colds, influenza, stomach problems, headaches, migraines, injuries of the muscular and skeletal system, as well as pain in the lower back part. Most of these conditions are preventable non-communicable diseases (NCDs) whose rate reduction includes scientifically based cost-effective measures. According to data from population surveys, NCDs are a major health problem in Serbia. Although they are to a great extent preventable, there is no adequate prevention and control of NCDs in Serbia [5, p. 41].

There are several groups of potential users who might benefit from the system that we describe in this paper. Users in healthcare institutions that are dealing with epidemiological data could utilize our software system, which is specially tailored to the epidemiological domain, instead of relying on solutions that are intended for generic statistical analyses. An expected advantage of having a domain-specific system would be an increase in user productivity. Large amounts of data that are typical of modern HISs may be well utilized owing to the well-tried approach incorporated into our system – a data warehouse for data storing and data mining for efficient analyses. In this manner, the main system load may be reduced by running analyses primarily on data stored in the Dr Warehouse system. The second group of users includes scientists whose research is related to epidemiology. By utilizing the Dr Warehouse system, they may create, test, and improve epidemic models through adding, running, and modifying new extensions. New visualization techniques for epidemiological data may be similarly employed and evaluated. Furthermore, the system may also target users who are not medical experts but are interested in latest disease trends, forecasts, or results of some specific analysis.

Bearing in mind the facts concerning the adverse health of the population in Serbia and the “white space” in terms of medical services aimed at predicting occurrence of certain diseases, our decision to develop a system that would allow the use of BI technologies in such a context is both socially and economically justified.

III. System Overview

In this section, we present the system and give an overview of its architecture and functionalities. The featured data warehouse, which represents a foundation for data analyses, is explained in more detail. We also elaborate on the built-in support for adding new functionalities.

A. *System Architecture*

There are four principal components in the system: (i) database server, which has a built-in support for extensibility and includes subcomponents: relational database management system, services for data mining and multidimensional analysis, and services for extracting, cleansing, transforming, and loading data from various sources; (ii) application server, which supports extensibility and acts as an intermediary between database server and clients; (iii) web client application, which supports extensibility and smart card reading; and (iv) mobile device client.

The system may fit into existing HISs and provide various services to other similar solutions. This architecture allows the possibility of having the database server and application server reside at different physical locations. Furthermore, in order to increase the scalability and performance of the system, the data mining and analysis services (currently implemented using Microsoft SQL Server Analysis Services [6]) may be located separately from the database server. In future versions of the system, the architecture may be extended to include terminals that would be publicly available and offer a set of functionalities similar to those in the existing web client application (currently implemented in Microsoft Silverlight [7]).

B. *Data Warehouse*

The data warehouse is modelled using a star schema, which consists of eight dimensions, two of which are role-playing dimensions, and one fact table (Fig. 1). The fact table keeps track of events which lead to absenteeism, disease occurrences and time measured in days that person spent away from duty or workplace. Each dimension represents the context of disease occurrence and absence. Therefore, we can observe these events in the context of time (when an event occurred or ended), gender of the person involved, place where it happened, person’s profession, data source, absence cause, person’s age, and diagnosis that was established. Dimensions concerning diagnosis, place, and time have several hierarchical levels modelled as a fully denormalized structure, which enables multi-level classification of factual data. In the time dimension, we have two hierarchies: one defined as calendar year, quarter, month, and day, and the other one as calendar year, week, and day. The diagnosis dimension has three levels of hierarchy for diagnosis, disease subcategory, and disease category, while community (place) dimension has four levels of hierarchy for community, state, region, and continent. Although the normalization of our schema would remove redundant data and hence become easier to maintain and change, our initial considerations of the schema type led us to choose the star schema. Denormalization, which is typical for the star schema, helped us to reduce the number of foreign keys and to reduce the query execution time. As the system was designed to be used by a wide variety of users, ease of use was one of our priorities. For end users, the star schema is more comprehensible than snowflake schema and less complex queries are needed to satisfy their information needs. Since this is a pilot project, advanced cost-benefit analysis of normalizing our star schema into the showflake schema is a matter of our future work. The unavailability of a larger and more complex absenteeism data set was a major reason for simplifying the initial schema design and focusing on the aforementioned fact and dimensions.

The data warehouse was implemented using Microsoft SQL Server 2008 [8]. It includes the following dimensions: *DimCause*, *DimDiagnosis*, *DimGender*, *DimProfession*, *DimCommunity*, *DimDataSource*, *DimTime*, and *DimAge*. *DiseasePresence* is the only fact table in the system. Each of these tables contains a surrogate primary key which allows us to deal with changes in natural key in a more convenient way and track slowly changing dimensions.

Taking into consideration that the data in the system are expected to reflect the actual state of the health of a population, it is necessary to support acquisition and integration of medical data from multiple sources. We developed a solution within Microsoft Integration Services [9], which allows us to extract, clean, transform, and load (ECTL) the necessary data. We perform incremental extraction, i.e., we consider only data that were added to the HIS of a public health institute after the previous extraction. Extracted data serve as an input for a series of transformations in which we detect and eliminate errors and inconsistencies: (i) different domains of semantically equivalent attributes (as in the case of the attribute *GenderName*); (ii) different encodings of textual data (*ProfessionTitle*); (iii) different granularity of semantically equivalent attributes (*DiagnosisCode*). Diagnosis codes that are used in the source HIS are shorter versions of the codes that are featured in the 10th revision of International Classification of Diseases (ICD 10) [10]. We created a transformation that relies on regular expressions to resolve this issue. In this manner, we extended disease information with the disease name, subcategory and category. At the moment, there is only support for data insertion. Since the data set in the current version of the system is only a sample taken from a HIS, we decided to keep all data in the data warehouse, while leaving the implementation of a deletion policy for obsolete data, data that have little or no impact on the system output, to be included in the future version.

In order to meet the needs for efficient and flexible consumption of valuable information produced by the system, we developed an online analytical processing (OLAP) database, which contains rich metadata. The OLAP cube makes our data organized in a way that facilitates non-predetermined queries for aggregated information. As we used Kimball Method [11] to implement the dimensional model in the relational database, the OLAP design step was a straightforward translation from the existing design. The relational database serves as the permanent storage of the cleaned and conformed data, and feeds data to the OLAP database. Data mining structures and models are stored in the third database, which, together with the OLAP database, resides at the Analysis Server – the primary query server.

C. System Functionalities

The communication between the application server and clients is done via web services. At present, the client applications possess functionalities concerning: access to medical records stored in the data warehouse; access to the data cube and use of some of the cube's advanced analytical operations; execution of advanced analyses and forecasts, as well as result retrieval; services specially tailored for mobile device client; upload of extensions; and their invocation.

The system, whose public resources are available at [12], may be accessed via a web application in which each page groups a number of similar functionalities. Within *Home page*, users may access information about the most common causes of absenteeism for the current month. *Analysis page* contains functionalities regarding the execution of advanced analysis and forecasts that identify the most probable diseases (or causes of work absence) and the most frequent

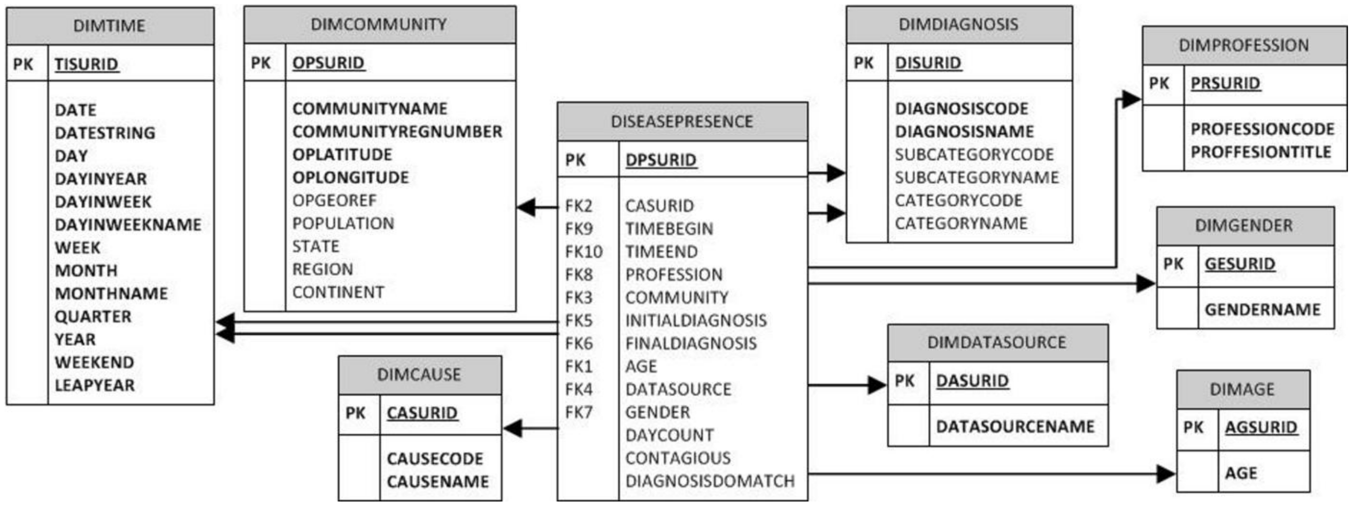


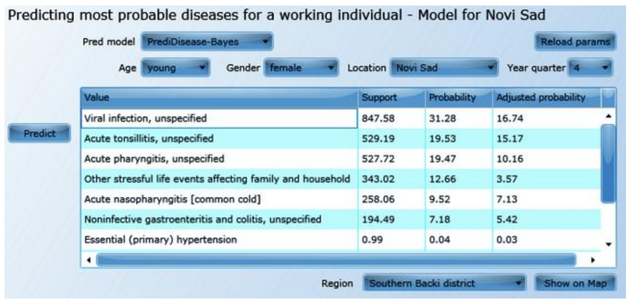
Figure 1. The star schema of the data warehouse.

diagnosis mismatches. Through this page, a user is able to generate predictions concerning a selected subpopulation for a particular quarter of a year. The subpopulation may be specified by selecting an age group, gender, and municipality (Fig. 2). *What about me? page* is a location from which we may generate and retrieve results of the personalized predictions concerning the most probable diseases (or causes of work absence). In order to generate these predictions, all a user needs to do is insert his or her identity card (ID card) into the attached smart card reader and a report is automatically generated. Execution of analytical operations and access to historical data is provided within *Health Reports page*. Users may perform operations such as dice and slice in order to analytically process the available data. *Upload page* offers functionalities regarding uploading of server and client extensions. Within *Extensions page* users may activate and run uploaded extensions.

Some of the aforementioned functionalities are also available via a mobile application for Microsoft Windows Phone [13]. These include disease predictions for a selected location (or the current location of a mobile device) and personalized predictions similar to those featured in *What about me? page* in the web client.

D. *Extensibility*

New functionalities may be added to the system in the form of extensions. The support for extensibility was implemented using Managed Extensibility Framework (MEF) [14]. A user may upload an extension, which then becomes immediately available for use without a need to restart the system. There are two types of extensions: (web) client extensions and (application) server extensions. Both may be uploaded to the application server through the web client. A web client extension is automatically downloaded from the application server to a web client machine, where it is then executed. This is done upon the first invocation of the extension at the client side. Such extension is actually a Silverlight web page that is generally expected to act as a user interface to the built-in or user-added (via server extensions) queries and analyses. On the other hand, server extensions reside on the application server, where they are also executed upon the invocation initiated at the client side. These extensions are functions generally responsible for data operations, analyses, and epidemic models.


Predicting most probable diseases for a working individual - Model for Novi Sad

Pred model: PrediDisease-Bayes

Age: young Gender: female Location: Novi Sad Year quarter: 4

Value	Support	Probability	Adjusted probability
Viral infection, unspecified	847.58	31.28	16.74
Acute tonsillitis, unspecified	529.19	19.53	15.17
Acute pharyngitis, unspecified	527.72	19.47	10.16
Other stressful life events affecting family and household	343.02	12.66	3.57
Acute nasopharyngitis [common cold]	258.06	9.52	7.13
Noninfective gastroenteritis and colitis, unspecified	194.49	7.18	5.42
Essential (primary) hypertension	0.99	0.04	0.03

Region: Southern Backi district

Figure 2. Section from the Analysis page in the web client.

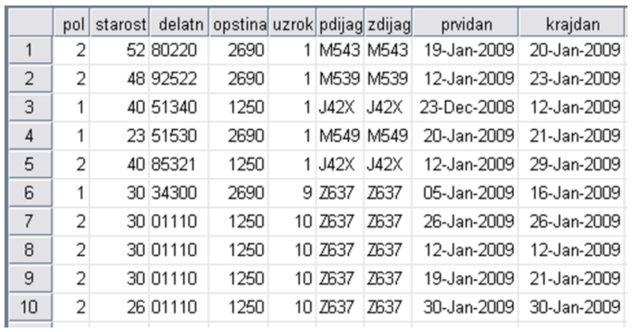
IV. FEATURED EPIDEMIOLOGICAL FORECASTS

In this section, we present two types of epidemiological forecasts that are available in Dr Warehouse: forecasts that rely on data mining and forecasts that rely on compartmental models. In addition to describing a data set that was used, we offer exemplary results of these forecasts.

A. *Data*

Data set used in the testing of the system during the development is acquired from the HIS of The Institute of Public Health of Vojvodina in Novi Sad, Serbia. The obtained sample (an excerpt is featured in Fig. 3) has approximately 8,500 records about workplace absences that ended in 2009. It contains depersonalized information including: gender (represented by the variable *pol*), age (*starost*), municipality code (*opstina*), absence cause (*uzrok*), start (*prvidan*) and end date (*krajdan*) of absence, disease codes for initial (*pdijag*) and final (*zdijag*) diagnosis, and business activity code (*delatn*) of a person involved.

Gender is represented by numbers 1 and 2 referring to the male or female respectively. Business activity code is represented by a five-digit code indicating sector, division, branch and group of a business activity in accordance with the classification of activities as defined by the corresponding law of the Republic of Serbia. Municipality code is a unique identifier of the municipality in which an absence was recorded. The cause of the absence is denoted by numbers from 1 to 12 that respectively correspond to: disease, isolation, accompanying sick person, maintenance of pregnancy, tissue and organ donor, injury at workplace, injury outside of workplace, occupational disease, nursing a child under 3 years, nursing a child over 3 years, care of other sick person, and maternity leave. Initial and final diagnosis codes are obtained by reducing the appropriate diagnosis codes defined by the 10th revision of International Classification of Diseases (ICD 10) to four characters. Codebooks of diseases, business activities, causes and municipalities may be gathered from official Internet sites of organizations that are responsible for their maintenance and distribution. Credibility of the data depends largely on the credibility of data sources. Therefore, we rely on sources that can guarantee the integrity and validity of provided data.



	pol	starost	delatn	opstina	uzrok	pdijag	zdijag	prvidan	krajdan
1	2	52	80220	2690	1	M543	M543	19-Jan-2009	20-Jan-2009
2	2	48	92522	2690	1	M539	M539	12-Jan-2009	23-Jan-2009
3	1	40	51340	1250	1	J42X	J42X	23-Dec-2008	12-Jan-2009
4	1	23	51530	2690	1	M549	M549	20-Jan-2009	21-Jan-2009
5	2	40	85321	1250	1	J42X	J42X	12-Jan-2009	29-Jan-2009
6	1	30	34300	2690	9	Z637	Z637	05-Jan-2009	16-Jan-2009
7	2	30	01110	1250	10	Z637	Z637	26-Jan-2009	26-Jan-2009
8	2	30	01110	1250	10	Z637	Z637	12-Jan-2009	12-Jan-2009
9	2	30	01110	1250	10	Z637	Z637	19-Jan-2009	21-Jan-2009
10	2	26	01110	1250	10	Z637	Z637	30-Jan-2009	30-Jan-2009

Figure 3. Excerpt from a data set used in the generation of predictions.

B. *Forecasts based on Data Mining*

In Dr Warehouse, we utilize three classification algorithms that are supported by Microsoft SQL Server 2008 R2 Analysis Services: decision trees, naive Bayes, and neural network classifier. These classifiers are trained to estimate the individual share of each disease in all work absences attributed to the 15 most common diseases, as determined by examining the available data set, for a selected year quarter and subpopulation, as defined by age group and gender, in a selected municipality. In this manner, we may form coarse predictions of the distribution of the most common diseases in a selected subpopulation. In Fig. 4, we give a set of predictions for male employees in the city of Novi Sad who are between 40 and 61 years old. This example demonstrates how a share of some common diseases in that subpopulation may change throughout a year. These estimates are generated using the naive Bayes classification algorithm for Novi Sad.

Predicted shares indicate that essential hypertension, dorsalgia (thoracic region), and lumbago with sciatica may be causes of a larger percentage of absence in quarters 2 and 3 (spring and summer), while their share substantially decreases during quarters 1 and 4 (winter and autumn). On the other hand, viral infection is most responsible for absences in quarter 4 (autumn).

C. *Forecasts based on Compartmental Models*

Compartmental models are a group of epidemic models that are used to predict dynamics of an epidemic by dividing an analysed population into several compartments (subpopulations) and calculating the changes in compartment sizes given some initial conditions [15-17]. These conditions include sizes of compartments (generally expressed as percentages of a whole population) at a single moment in time. Population compartments correspond to infected, recovered, or some other group of individuals in a population. Furthermore, there are disease-related parameters that are needed in the calculation of changes in compartments sizes: contact rate, recovery rate, death rate, etc. Different models from this family feature different compartments and may be used to obtain forecasts for different diseases. Actual spread of a disease (transition of individuals between different compartments) is modelled by a system of differential equations.

As an example of how Dr Warehouse may support standard epidemic models, we implemented the SIR (Susceptible/Infected/Recovered) model as an extension pair consisting of: (i) a client extension, which is used to set parameters, invoke model execution, and present results; and (ii) a server extension, which is an invoked function that numerically solves the system of equations and prepares results for the client side.

The implementation is an adaptation of the model version featured in [18]. The name of the model is derived from the three compartments that are used to model a population struck by a disease: susceptible (S), infected (I), and recovered (R). A susceptible individual from the S compartment may become infected through contact with an infected individual from the I compartment, while an infected individual may become a member of the R compartment after a recovery period. A rate at which a disease is transmitted from an infected to a susceptible individual is the contact rate β, while a rate at which an infected individual recovers is the recovery rate γ. Actual values for the rates β and γ depend on a disease that is being modelled. Three ordinary differential equations describe the dynamics:

$$dS / dt = -\beta I S, \quad (1)$$
$$dI / dt = \beta I S - \gamma I, \quad (2)$$
$$dR / dt = \gamma I. \quad (3)$$

Our implementation approximates the solution of this system by using the 4th order Runge-Kutta method for solving a system of ordinary differential equations. In Fig. 5, we give an example of a prediction that was generated by a chronological simulation using our implementation of the SIR model. The presented chart demonstrates a typical situation when equilibrium in a population is gradually reached after a peak in the number of infected individuals.

Other compartmental models may be implemented in a similar manner. The only differences would be changes in a set of differential equations that model a disease and addition of new parameters or compartments. By adding the support

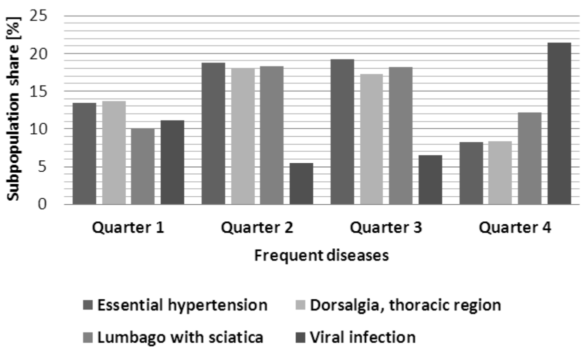


Figure 4. Example of percentage disease shares for male employees in Novi Sad aged between 40 and 61 years, as predicted using a naive Bayes classifier.

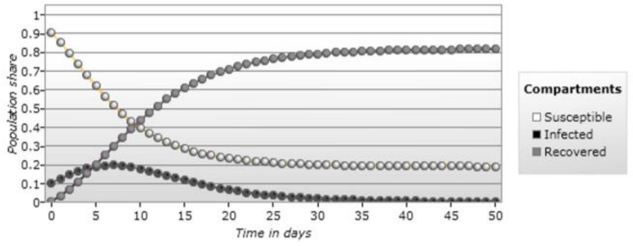


Figure 5. Example of a disease forecast obtained using the SIR model.

for the SIR model in the form of a pair of extensions, we have demonstrated that Dr Warehouse may be used to predict the rate of spread of any disease for which there is an adequate compartmental model.

V. RELATED WORK

There are many software systems for epidemiological analyses and monitoring. One group of such systems provides mostly statistical procedures that are often used in epidemiology. Open Source Epidemiologic Statistics for Public Health (OpenEpi) [19] is an example of a freely available system that may be run in a web browser [20] because it is implemented in HyperText Markup Language (HTML) and JavaScript. It focuses on statistical calculations: calculation of confidence interval and sample size, estimation of power for different types of studies, execution of various statistical tests, etc. Another free solution is WinPepi [21], which is a set of desktop applications that are similar to OpenEpi and offer many statistical procedures that are useful in epidemiology. When compared to Dr Warehouse, both OpenEpi and WinPepi are projects of a narrower scope because they ignore data storage and management. Furthermore, they put emphasis on statistics and a large number of calculation modules whose input is mostly a small set of summarized values. Unlike Dr Warehouse, they do not support data mining, visual representation of data, epidemiological maps, nor user extensions. However, the source code of OpenEpi may be directly modified to include new procedures.

The second group of epidemiological systems includes data storing and manipulation capabilities in addition to analysis procedures. Epi Info [22] is one such example of a desktop software application with a wider range of functionalities than OpenEpi and WinPepi. What sets it apart from other software systems for epidemiology is the support for form creation. A user may design custom forms through an integrated editor and later use them for data entry. Besides basic and advanced statistical procedures, this system supports data import and export, as well as basic data selection and transformation. It has good data visualization capabilities and offers various types of charts, tables, and even map overlay. Its main strengths with respect to Dr Warehouse are support for form creation, direct data entry, data transformation and data import/export for various types of data sources. However, there is a conceptual difference between these two systems regarding data storage. Epi Info is a tool that may be used over any data (in the supported file or database format) and, therefore, provides transformation functions, which a user utilizes in order to prepare data for analyses. On the other hand, Dr Warehouse features a data warehouse with a fixed set of facts and dimensions, and a carefully designed ECTL process, which is automatically executed. Therefore, there is generally no need for manual data import and transformation because data preparation is done automatically. In other words, Dr Warehouse may be seen as a more specialized and more automated solution in which the data warehouse has a prominent role. Our system relies on a strong dependency between the data warehouse schema and analyses, which helps to simplify the analysing process. This, in turn, alleviates much of the burden concerning data preparation, which is usually the longest activity in analysis projects. Some of the main features of Dr Warehouse that Epi Info lacks are data mining procedures and the support for adding user extensions. We consider data mining to be an essential part of the system because, unlike most statistical procedures, it is well suited for analysing large quantities of data that are efficiently stored in a data warehouse. We may summarize this comparison by generally classifying Epi Info as a solution that offers a fixed set of analyses for any set of data attributes and Dr Warehouse as a solution that features a fixed set of data variables but an extensible set of techniques for data presentation and analysis.

The third group consists of typically web-based systems that focus on epidemiological monitoring and publicly presenting latest disease outbreak data for different regions throughout the world. They primarily rely on data from numerous Internet-related sources, which may be informal or official. HealthMap [23] provides a world map with the latest information on outbreaks by automatically collecting and integrating data mostly from several online news sources and reports from eyewitnesses and officials. There is also a mobile version of the system with similar functionalities. Another web system with a support for mobile devices is Outbreak Watch [24]. It does real-time analyses of data in social networks by evaluating keywords that are considered to be indicators of outbreaks. In this manner, the system tracks changes in the number of reports concerning relevant diseases. Google Flu Trends [25] was created as an attempt to estimate actual flu activity in various countries by analysing aggregated Google search queries that are related to flu. Since there is a relationship between an actual number of flu cases and search queries about flu, as confirmed by the overall match between the official surveillance data and the calculated estimates, this service offers near real-time results, which may help in preparing a response to a flu outbreak. Dr Warehouse is similar to these systems, as it may offer latest epidemiological data and forecasts in the form of charts, tables, and maps. In addition to supporting web access, it also features a mobile version with a selected set of services. On the other hand, the principle difference lies in the selection of data sources. The three monitoring systems use data that are available on the Internet (HealthMap and Outbreak Watch) or from web search queries (Google Flu Trends), while Dr Warehouse displays only data present in the data warehouse, which was planned to include credible data collected in healthcare institutions. However, the ECTL process in Dr Warehouse may be extended in the future to include data from public web sources.

When compared to the three aforementioned groups of epidemiological software, Dr Warehouse is a complex system that possesses traits typical of all three because: (i) it may offer any statistical procedure that has been added as an extension; (ii) data management is one of the key segments of the system; and (iii) collected data are constantly available to users via web and mobile client, which makes the system suitable for epidemiological monitoring.

VI. Conclusion and Future Work

In this paper, we introduced a software system that may be used in epidemiology for data collection, data mining, analyses and research. We expect that this system may have an important role in the activities concerned with epidemic control and better understanding of temporal and spatial disease patterns. In addition to a general description of the system's structure and components, a special attention was given to the implementation of its data warehouse and data analyses. The presented examples of analyses illustrate just some of the results that may be obtained through the current version of the system. In order to support application of the latest advances in epidemiology and evaluation of new epidemic models, we incorporated extensibility that allows addition of new functionalities to the system.

There are numerous ideas for future work and research on the presented system. We may modify the existing analyses and make them more generic so that they could support a greater number of queries. The data warehouse schema may be altered and extended in order to support additional analyses. We may also enforce a strict security policy by introducing user roles and separating the set of functionalities into subsets better suited for various user categories. A new version of the system could be implemented using open (and free) technologies, which could lead to a creation of a completely open version of the system. Due to the prominence of spatio-temporal and epidemiological data in Dr Warehouse, best practices from geographic information systems and constraint databases are topics also worth exploring in the future. Furthermore, significant additions to the system would be construction of an epidemiological knowledge base, which could be regularly updated or consulted during data analyses, and creation (or selection) of a convenient ontology. In this manner, the semantics may be expressed and the new version of the system could communicate with other systems that follow the idea of the Semantic Web.

Acknowledgement

The research was supported by Ministry of Education and Science of Republic of Serbia, Grant III-44010. The authors are most grateful to The Institute of Public Health of Vojvodina in Novi Sad for the provided absenteeism data sample and valuable comments.

References

[1] *Strategija razvoja informacionog društva u Republici Srbiji do 2020. godine* [The Strategy for the Development of Information Society in the Republic of Serbia until the Year 2020], (in Serbian), Službeni glasnik Republike Srbije, vol. 51, 2010.

[2] M. Fimić, M. Radulović, I. Vulić, and S. Atanasijević, "Zdravstveni informacioni sistem Ministarstva odbrane Republike Srbije – generičko rešenje za integraciju institucija" [The Health Information System of the Ministry of Defense of the Republic of Serbia – A Generic Solution for Institution Integration], (in Serbian), in Proceedings of YU INFO 2012, pp. 511-516.

[3] G. Johns, "absenteeism," in *The Blackwell Encyclopedia of Sociology*, G. Ritzer, Ed. Oxford, UK: Blackwell Publishing, 2007, pp. 4-7.

[4] "Absence management 2009 – Survey Reports - CIPD," http://www.cipd.co.uk/hr-resources/survey-reports/absence-management-2009.aspx [Dec. 7, 2012].

[5] Đ. Jakovljević and P. Mićović, *Zdravstveno stanje i zdravstvene potrebe stanovništva Srbije* [Health Status and Health Needs of the Population of Serbia], (in Serbian), http://www.palgo.org/files/leaflet/brosura_zdravstvo.pdf [Dec. 7, 2012].

[6] "SQL Server Analysis Services," http://technet.microsoft.com/en-us/sqlserver/cc510300.aspx [Dec. 7, 2012].

[7] "Microsoft Silverlight," http://www.microsoft.com/silverlight/ [Dec. 7, 2012].

[8] "Microsoft SQL Server," http://www.microsoft.com/sqlserver/ [Dec. 7, 2012].

[9] "Microsoft Integration Services," http://msdn.microsoft.com/en-us/library/ms141026%28v=sql.105%29.aspx [Dec. 7, 2012].

[10] "International Classification of Diseases," http://www.cdc.gov/nchs/icd/icd10cm.htm [Dec. 7, 2012].

[11] J. Mundy, W. Thornthwaite, and R. Kimball, *The Microsoft Data Warehouse Toolkit: With SQL Server 2008 R2 and the Microsoft Business Intelligence Toolset*, 2nd ed., Indianapolis, IN: Wiley Publishing, Inc., 2011.

[12] "Dr Warehouse," http://www.acs.uns.ac.rs/sr/node/237/1140/ [Dec. 7, 2012].

[13] "Microsoft Windows Phone," http://www.microsoft.com/windowsphone/ [Dec. 7, 2012].

[14] "Managed Extensibility Framework," http://msdn.microsoft.com/en-us/library/dd460648.aspx [Dec. 7, 2012].

[15] W.O. Kermack and A. G. McKendrick, "A contribution to the mathematical theory of epidemics," *Proceedings of the Royal Society of London*, vol. 115, no. 772, pp. 700-721, August 1927.

[16] R.M. Anderson and R.M. May, "Population biology of infectious diseases: Part I," *Nature*, vol. 280, no. 5721, pp. 361–367, August 1979.

[17] R.M. May and R.M. Anderson, "Population biology of infectious diseases: Part II," *Nature*, vol. 280, no. 5722, pp. 455–461, August 1979.

[18] M.J. Keeling and P. Rohani, *Modeling Infectious Diseases in Humans and Animals*. Princeton, NJ: Princeton University Press, 2007.

[19] K.M. Sullivan, A. Dean, and M.M. Soe, "OpenEpi - a web-based epidemiologic and statistical calculator for public health," *Public Health Reports*, vol. 124, no. 3, pp. 471-474, May-June 2009.

[20] "Open Source Epidemiologic Statistics for Public Health," http://www.openepi.com/ [Dec. 7, 2012].

[21] J.H. Abramson, "WINPEPI updated: computer programs for epidemiologists, and their teaching potential," *Epidemiologic Perspectives & Innovations*, vol. 8, no. 1, pp. 1-9, February 2011.

[22] "Epi Info™ - Community Edition," http://epiinfo.codeplex.com/ [Dec. 7, 2012].

[23] "HealthMap," http://www.healthmap.org/ [Dec. 7, 2012].

[24] "Outbreak Watch Social Biosurveillance Network," http://www.outbreakwatch.com/ [Dec. 7, 2012].

[25] "Google Flu Trends," http://www.google.org/flutrends/ [Dec. 7, 2012].

Shrinking data balls in metric indexes

Bilegsaikhan Naidan and Magnus Lie Hetland
Department of Computer and Information Science,
Norwegian University of Science and Technology,
Sem Sælands vei 7-9, NO-7491 Trondheim, Norway
{*bileg,mlh*}*@idi.ntnu.no*

***Abstract*—Some of the existing techniques for approximate similarity retrieval in metric spaces are focused on shrinking the query region by user-defined parameter. We modify this approach slightly and present a new approximation technique that shrinks data regions instead. The proposed technique can be applied to any metric indexing structure based on the ball-partitioning principle. Experiments show that our technique performs better than the relative error approximation and region proximity techniques, and that it achieves significant speedup over exact search with a low degree of error. Beyond introducing this new method, we also point out and remedy a problem in the relative error approximation technique, substantially improving its performance.**

Keywords-approximation algorithms, experiments, similarity search, metric space.

I. Introduction

Nowadays, efficient similarity retrieval is becoming more important in various applications such as multimedia repositories (images, audio, video) because of the rapid growth of these data sets and the increasing demand for access to them. In such search applications, the relevance of a data object is often measured by some distance function that provides quantitative information about its similarity to some given sample query. For search techniques that treat the distance as a black-box relevance measure, the main challenge is to quickly retrieve a small set of the most relevant objects (either all within a search radius, or the k nearest neighbors, k-NN) relying on the properties of the distance—usually by exploiting the metric axioms.

Numerous metric indexing structures have been proposed to reduce the computational cost (such as the total number of distance computations at query time) of similarity retrieval [1]. These methods primarily rely on various forms of filtering based on the triangle inequality. Triangular filtering is efficient in low-dimensional spaces. However, as the dimensionality of a space increases, the performance of these indexes degrades because of the so-called curse of dimensionality: distances grow increasingly similar, and eventually one may need to examine more or less all data objects, the equivalent of a linear scan. One promising approach to ameliorating this curse is *approximate* similarity search, where some result quality is sacrificed in order to gain performance. This is acceptable in many applications, as distance-based retrieval is generally approximate to begin with—the distance function is most likely an approximation of the user's perception of similarity, and the user probably wants *similar* objects (e.g., pictures of horses), not necessarily the *most* similar objects (i.e., the most similar horse).

Some important methods used in approximate similarity search are discarding data regions at query time (by shrinking the query ball by a user-defined factor [2–4] or by analyzing the intersection of query and data regions [5]), representing data objects as permutations of a set of pivots [6], and estimating the distance by linear regression [7]. We focus on the first approach, trying to develop a method for discard regions that overlap with the query, but that are likely to contain few relevant objects, if any. Our main contributions are:

- We propose a new approximation technique that shrinks data regions instead of the query region, and show empirically that it is superior to existing methods in many cases.
- We amend a problem in the relative error approximation technique of Zezula et al. [2]. In several experimental studies, this technique was found to be the worst one [1, 2, 5]. We point out a problem with how the method has been used, and show how the amended version has significantly improved performance wrt. the original, making it comparable even to the region proximity method [5].

The rest of the paper is organized as follows. In Section II, we briefly review related work. In Section III, we propose our approximation technique, and describe how to amend the relative error approximation technique. Section IV provides experimental results and some discussion of those results. Finally, Section V contains some concluding remarks.

II. Related Work

In this section, we briefly review two metric indexing structures based on the ball-partitioning principle—the M- and SSS-trees—explain some issues in the construction of indexes and also review some approximate techniques that can be applied to those structures.

The M-tree [8] is a hierarchical dynamic metric ball tree that is designed for secondary memory. The M-tree is built in a bottom-up manner like B-trees. The insertion algorithm starts from the root and moves toward the leaves by selecting nodes that are closer to the new object or that require a minimum enlargement of existing balls. The new object is finally inserted into a leaf node. This may cause the leaf to split (if the node capacity is exceeded), which may trigger splits in some of its ancestor nodes, possibly even the root. For a leaf node split, the covering radius of the split node is set to the distance from the center to the object furthest away, that is, the actual covering radius. For an internal node split, the covering radius is not computed exactly, but overestimated, as follows. For every child node, the covering radius of that node is added to the distance between the center of that child and that of the split node. Then, the covering radius of the split node is then set to the maximum of those sums.

Brisaboa et al. have proposed a static index structure so called the Sparse Spatial Selection (SSS) tree [9], in which the first object in a data set is selected as the first cluster center and then the rest of the objects become new cluster centers if they are far enough away from all current centers (i.e., the minimum distance between the object and current cluster centers is greater than αM, where α is a user-defined parameter and M is the maximum distance between any two objects); otherwise, they are assigned to the cluster associated with the nearest center. The process is recursively applied to those clusters that have not yet fallen below a given size threshold and the diameter M of each such cluster is estimated by using twice the covering radius of the node. Because of the clustering principle used in the construction phase, the internal nodes of SSS-trees will generally have smaller regions than the internal nodes of M-trees. However, there are still sparse regions at higher levels of SSS-trees.

Three approximate techniques for k-NN search were introduced by Zezula et al. [2]. The first, the so-called relative error approximation technique, controls approximation through a user-defined relative distance error $\epsilon \geq 0$. For a given query q and error ϵ, an approximation of a kth nearest neighbor O_A^k is called a $(1+\epsilon)$ kth nearest neighbor, compared to the *true* kth nearest neighbor O_N^k, if and only if $d(q, O_A^k) \leq (1+\epsilon) \cdot d(q, O_N^k)$. Thus, the search algorithm uses the radius $r_q/(1+\epsilon)$ instead of the covering radius of the current k-NN candidate set r_q to check overlap between query and data regions and candidate object qualification as well. An example of this approach is given in Figure 1a. The second, the so-called good fraction approximation technique, uses a distance distribution to provide an early termination criterion which leads to an approximate kNN search. In the third, the so-called small chance of improvement approximation technique, the search algorithm is based on the fact that the dynamic radius of the result set initially decreases rapidly and eventually will slow down. Thus the search stops as soon as the decrease of the radius becomes sufficient.

The PAC method [3] is an extension of $(1+\epsilon)$ nearest neighbor search by a user-specified confidence parameter $\delta \in (0,1)$. The search algorithm stops immediately if the result satisfies the $(1+\epsilon)$ nearest neighbor with a confidence of at least δ.

Probabilistic LAESA [4] is a probabilistic technique for range search that provides user-customizable limits (θ) for the probability of false dismissals. More specifically, the radius r_q is scaled down by a factor $(1+\epsilon)$ $(\epsilon > 0)$ during the filtration of indexed objects. The upper bound for $(1+\epsilon)$ is $r_q\sqrt{1-(1-\theta)^{1/p}}/(\sqrt{2}\sigma)$, where p is the number of pivots and σ^2 is the variance of the distance distribution of the data set. Figure 1b shows an example of this technique.

The region proximity technique [5] estimates the probability of the intersection of the query and data regions containing objects relevant to the query. The data region is discarded if the estimated probability is less than a user-specified threshold.

III. Our approach

First, we explain the basic principles of the so-called *best first* strategy [10] for k-NN search. In essence, we maintain a set of at most k (initially zero) candidates throughout the search. We also maintain a covering radius for this candidate set. This covering radius is infinite as long as we have fewer than k candidates. The algorithm processes the most promising metric regions first by maintaining a priority queue of pair of distances to regions and pointers to those regions. The following actions are repeated until the lower bound of the distance from the query to the region about to be processed is greater than the current dynamic query radius. The most promising region is popped from the queue. The objects of the current region are checked with the current dynamic query radius and, if necessary, the list of candidate k-NN is updated. If we have k candidate, this leads the reduction of the dynamic query radius. Those subregions of the current region that intersect with the query region reinserted into the queue along with the lower bound distances to them from the query.

We now look at our approach. Metric indexes may have highly sparse data regions at higher levels, with the ball radii covering large amounts of empty space—especially for high dimensionalities. During a search, in order to discard those regions that might not contain relevant objects for the query we could use any version of of the $(1+\epsilon)$ NN technique (for instance, the relative error approximation). Our suggested approach is similar to the relative error approximation technique and the key difference is to divide the *data ball radius* by $(1+\epsilon)$, rather than the query radius. See Figure 2 for an example. In the figure, we have shown what happens if we divide both the query radius and the data radius by $(1+\epsilon)$. In the first example (Figure 2a), the query lies close to the data ball, on the outside. In this case, our method lets us eliminate the region simply because it increases the lower bound more. In the second example (Figure 2b), the query is just *inside* the data ball. In this case,

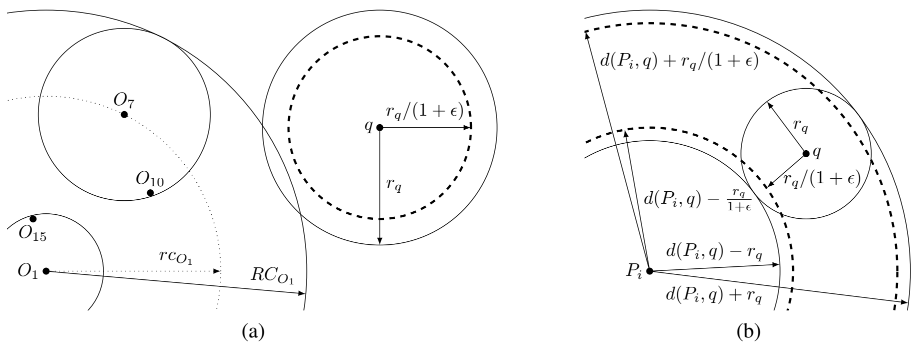


Figure 1: Examples of data and query ball regions in $\mathbb{R}^2$ with $\mathcal{L}_2$. (a) Relative error approximation in the M-tree [8] with two levels (i.e., top level with center O_1 and bottom levels with centers O_1 and O_7). RC_{O_1} represents the covering radius of top region centered at O_1 while rc_{O_1} represents its "true" covering radius. (b) Probabilistic LAESA.

shrinking the query radius will *never* lead to an elimination, whereas our method does.

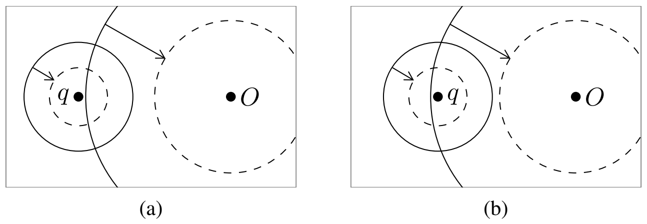


Figure 2: Examples of data and query regions in $\mathbb{R}^2$ with $\mathcal{L}_2$ (a) query q is outside the data region with center O and (b) q is inside the data region.

These examples demonstrate the twofold intuition behind our method: First, ball trees are generally built using some form of clustering. If the data set itself is clustered, and the clustering algorithm is good, this will presumably lead to the center region of a ball being more densely populated than its periphery. Even if this is not the case, by setting the radius to the maximum of all center–object distances, the radius is sensitive to outliers, and the more extreme they are, the fewer there are likely to be. Even if we do not assume a Gaussian distribution, it is not unreasonable to guess that our distance histogram will have the majority of its values clustered roughly around the mean, with fewer occurrences of very high and low distances (ignoring self-distances, $d(x,x)$). Assume that we have a global distance histogram somewhat like that in Figure 3, for example. We also assume that the center–object distances are distributed roughly according to the global distance histogram. Chávez et al. call this behavior as a "reasonable approximation" [11, p. 304]. In this case, it seems that it would be safe to shrink large data balls more than smaller ones, as the number of objects lost would be smaller. The same could be said about the smallest balls, of course; however, we would probably want to examine most of those, if they are close to the query, as they more precisely represent the objects inside them.

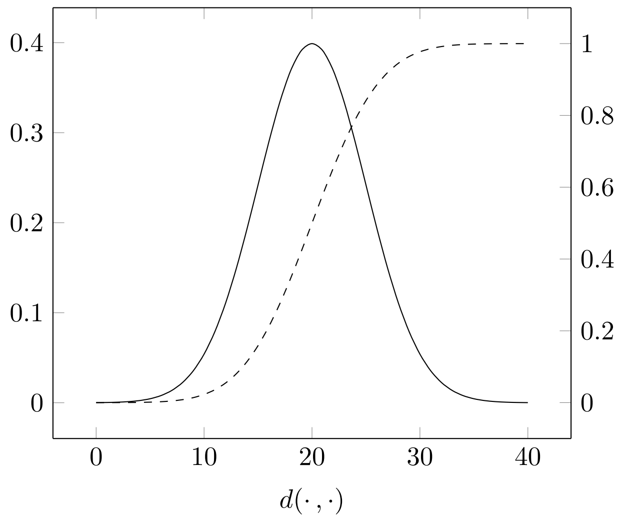


Figure 3: Distance histogram (solid) and cumulative distance histogram (dashed).

Second, shrinking the data balls affords us some elimination possibilities that simply do not exist with the original query-shrinking approach, that is, when the query falls inside the data ball. We have performed some tentative experiments to explore the relative importance of these two factors. At parameter settings that yield similar levels of error, our method generally uses fewer distance computations than the relative error method (see Section IV). We estimate that the proportion of the saved distance computations caused by cases where the query is inside the data ball to vary from about $1\,\%$ to over $50\,\%$ (data not shown).

Another contribution of this paper is that we point out and remedy a problem in the relative error approximation technique. Several experimental studies have showed that the performance of the relative error method (as originally described) is very poor [1, 2, 5]. According to Zezula et al., "the chief reason for the markedly poor performance of the Relative Error Approximation method (with respect to the others) is that precise nearest neighbors algorithms find good candidates for the result sets soon on, and then spend the remainder of their time mostly in refining the current results" [1, p. 157]. We claim, instead, that the main reason for this performance issue is found in the pruning criterion for a candidate given object O, given by Equation 16 on page 280 of the original paper by Zezula et al. [2]:

$$\frac{r_q}{d(q,O)} < 1+\epsilon\,,$$

or, equivalently,

$$\frac{r_q}{1+\epsilon} < d(q,O)\,,$$

where r_q is the covering radius of the current k-NN candidate set.

Now, the radius shrinking is intended to reduce the number of distance computations needed by excluding regions of low relevance. There is no need to use it here, as the distance $d(q,O)$ has already been computed, and we simply wish to know whether the object O is an improvement over the candidates we have found so far. We can determine this by simply comparing $d(q,O)$ directly to r_q. Indeed, if

$$\frac{r_q}{1+\epsilon} < d(q,O) < r_q$$

we will lose an improvement to our candidate set, involving an object whose distance we have *already computed*. This can be particularly important early on, where we wish to add good candidates (thereby reducing the dynamic search radius) as quickly as possible. In our experiments, we use this improved version of the relative error approximation technique, checking each candidate object against the actual covering radius of the candidate set.

IV. Experiments

In this section, we evaluate the performance and result quality of our technique against the amended version of the relative error approximation and the region proximity techniques on synthetic and real-world data sets. For all data sets we use the Euclidean distance.

- Uniform 10: Synthetic. 100 000 uniformly generated 10-dimensional vectors.
- Clusters 10: Synthetic. 100 000 clustered 10-dimensional vectors with 10 cluster centers. The centers were randomly chosen from a uniform distribution and objects in the clusters were generated from the multivariate normal distribution around each of the cluster centers with a variance of 0.1.
- Corel: 60 000 feature vectors with 64 dimensions extracted from the Corel image data set.
- NUS [12]: 269 648 color histograms extracted from Flickr images. Each histogram is 64-dimensional.

Amato et al. claimed that there was no practical difference between the proximity and the PAC-NN technique [5, p. 225]. Also PAC-NN is designed only for approximate NN retrieval ($k = 1$). Therefore, PAC-NN is not considered for our experiments.

We have applied our method, region proximity, and the amended version of the relative error technique on M- and SSS-trees. The maximum arities of the trees were set to 30 for the synthetic and 15 for the real-world data sets. We selected 1000 queries from the respective data set at random and the remaining objects in the data set used for indexing. We compared the search performance and result quality of three techniques by varying the result size threshold (k), using the values 1, 5, 10, 20, 40 and 80. We report only the results with 10 NN because the results with the other result size thresholds were quite similar. For the relative error approximation technique, the relative error α was varied in the interval [0.001, 2.0] with step size 0.1 following the experimental settings of Zezula et al. [2]. For the region proximity technique, the proximity value was varied in the interval [0.003, 0.06] with step size 0.003 following the experimental settings of Amato et al. [5]. For our technique, the data region stretching factor was varied between 0.1 and 2.0 with step size 0.1.

For each query, we counted the number of distance computations needed for the approximate search (the most commonly used criterion for measuring the performance of metric indexing structures), normalized by the number needed for an exact search, and measured a slightly modified version of the error on the position [1, 5]. The original version of the error on the position has some drawbacks. First, it gives an error value that is normalized by the size of data set. The normalized values are harder to interpret. For example, we can not directly see the absolute position difference between exact and approximate results. Second, the error should not be normalized by the approximate result's size and should take missing objects in the result set into account.

Let us have a look at a simple example: Let n be data set size and the result size threshold k be 80 ($k < n$) and the approximate result be only true 3 NNs. For some reason, the approximate result did not retrieve other 77 NNs (for instance, the search algorithm was terminated early). Then if we apply the original formula on this example, the error on the position is $((1-1)+(2-2)+(3-3))/(3n) = 0$. The error value 0 means that the approximate result has no error and as we see that it should not be 0 in this case. The modified version takes these situations into account, and the error is increased by n as a *fine* for every missing object and then the error is only normalized by k. This modified version of the error on the position yields the average absolute position difference between every point of the exact and approximate results.

In general, approximation techniques will produce results that vary both in performance and accuracy. In order to make a fair comparison between different techniques we have to compare their speed-up factors with the same error or vice versa. In some cases, it would be difficult to achieve this goal, as neither performance measure is a deterministic function of the parameter settings. In order to compare the results properly, we plot them as a lines with one point for each parameter setting, with the coordinates for each point given by the mean error and mean normalized distance count for all queries. On the y-axis, the value 10^{-1} means that the indexing structure performed 10 times as fast as an exact search. On the x-axis, the value 10^1 means that the average absolute position difference between exact and approximate result is 10 (for instance, if the result size threshold is 1, then the 11th NN is reported instead of the NN).

In Figure 4, the errors on the position vs normalized distance counts for $10\,\mathrm{NN}$ on M-trees are shown. Note that both axis are logarithmic. For the results of NUS 10 NN in Figure 4, our technique achieved a speed-up by a factor of more than 4 over the exact search, with the position error less than 10, while the region proximity technique achieved almost same speed-up with relatively high position error 10^4 (i.e., reported 10 objects from around $10\,000$ NNs for the query). For the other data sets, our technique achieved about 2.5 speed-up over exact search with the position error less than 10.

Figure 5 shows the results for SSS-trees. The most interesting results are once again obtained with the NUS data set. For NUS 10 NN, the maximum value of the error was 48.39 (with normalized distance count 0.135) for our technique while for the region proximity technique the error was 1751.39 (with normalized distance count 0.136). The results show that our technique is faster than two other competing techniques with a low degree of the error on the position. The relative error approximation technique outperforms the region proximity technique on the real-world data sets while on the synthetic data sets it does not.

In addition to the experiments presented, we have also performed some tentative experiments involving various tradeoffs between query- and data-ball shrinking. So far, this has not yielded substantial improvements.

V. Conclusions

We have proposed an approximate similarity search technique for metric spaces and we have amended a problem in the relative error approximation technique. We have empirically evaluated our technique, showing that it outperforms the amended version of relative error approximation and the region proximity techniques.

Acknowledgements

We wish to thank Øystein Torbjørnsen and Svein Erik Bratsberg for helpful discussions.

References

[1] P. Zezula, G. Amato, V. Dohnal, and M. Batko, *Similarity Search : The Metric Space Approach*. Springer, 2006.

[2] P. Zezula, P. Savino, G. Amato, and F. Rabitti, “Approximate similarity retrieval with M-Trees,” *The VLDB Journal*, vol. 7, no. 4, pp. 275–293, 1998.

[3] P. Ciaccia and M. Patella, “PAC nearest neighbor queries: Approximate and controlled search in high-dimensional and metric spaces,” in *Proceedings of the 16th International Conference on Data Engineering, ICDE* (IEEE Computer Society, ed.), pp. 244–255, 2000.

[4] E. Chávez and G. Navarro, “A probabilistic spell for the curse of dimensionality,” in *Revised Papers from ALENEX’01*, pp. 147–160, 2001.

[5] G. Amato, F. Rabitti, P. Savino, and P. Zezula, “Region proximity in metric spaces and its use for approximate similarity search,” *ACM Transactions on Information Systems, TOIS*, vol. 21, no. 2, pp. 192–227, 2003.

[6] E. Chavez Gonzalez, K. Figueroa, and G. Navarro, “Effective proximity retrieval by ordering permutations,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008.

[7] O. Edsberg and M. L. Hetland, “Indexing inexact proximity search with distance regression in pivot space,” in *Proceedings of the Third International Conference on SImilarity Search and APplications*, SISAP ’10, 2010.

[8] P. Ciaccia, M. Patella, and P. Zezula, “M-tree: An efficient access method for similarity search in metric spaces,” in *Proceedings of the 23rd International Conference on Very Large Data Bases, VLDB*, pp. 426–435, Morgan Kaufmann, 1997.

[9] N. Brisaboa, O. Pedreira, D. Seco, R. Solar, and R. Uribe, “Clustering-based similarity search in metric spaces with sparse spatial centers,” in *Proceedings of SOFSEM’08*, no. 4910 in LNCS, pp. 186–197, 2008.

[10] H. Samet, *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann, 2006.

[11] E. Chávez, G. Navarro, R. Baeza-Yates, and J. L. Marroquín, “Searching in metric spaces,” *ACM Computing Surveys*, vol. 33, pp. 273–321, September 2001.

[12] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y.-T. Zheng, “Nus-wide: A real-world web image database from national university of singapore,” in *Proc. of ACM Conference on Image and Video Retrieval (CIVR’09)*, 2009.

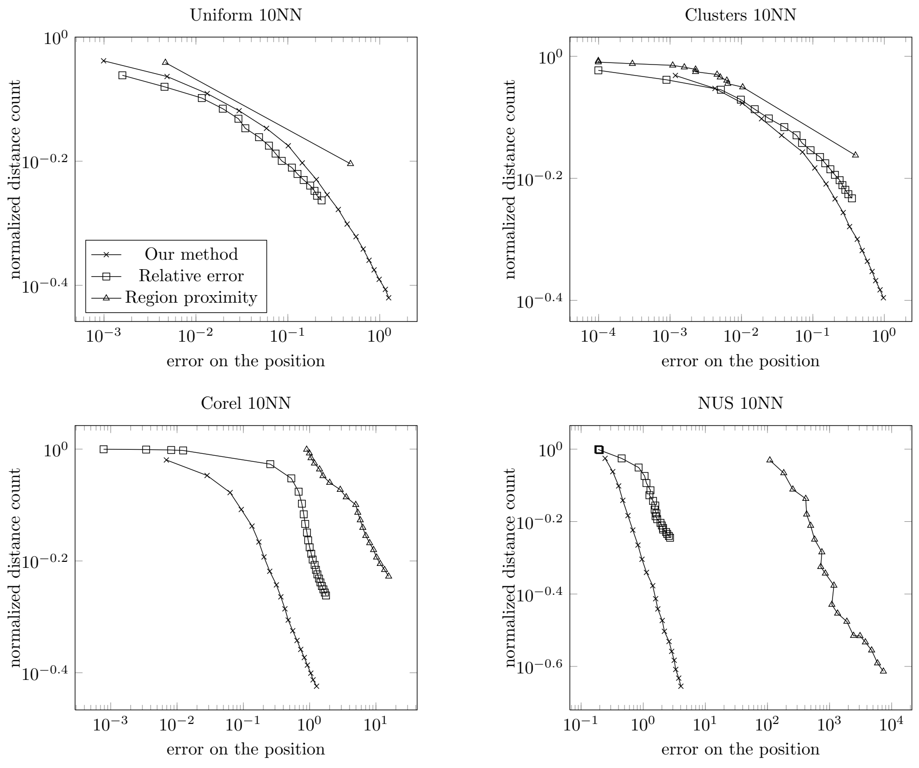


Figure 4: Performance vs. result quality of approximation on the synthetic and real-world data sets with M-trees.

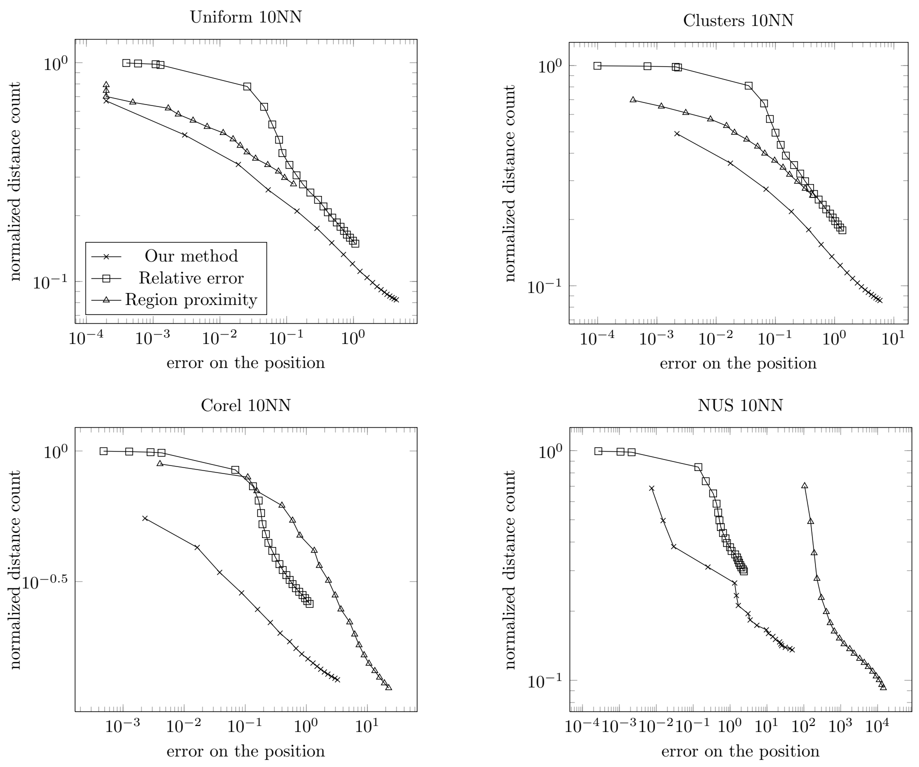


Figure 5: Performance vs. result quality of approximation on the synthetic and real-world data sets with SSS-trees.

An Epidemiological Data Mining Application Based on Census Databases

J. Pérez-Ortega, Alicia Martínez, E. Iturbide-Domínguez, M. Hidalgo-Reyes, A. Mexicano-Santoyo
Department of Computer Science
CENIDET
Cuernavaca, México
jpo_cenidet@yahoo.com.mx, {amartinez, emmanuel.iturbide10c, mh}@cenidet.edu.mx
amexicano@gmail.com

Crispin Zavala-Díaz
Faculty of Accounting, Administration and Informatics
UAEM
Cuernavaca, México
crispin_zavala@uaem.mx

***Abstract*— This paper shows the experience of a specialized data mining process that integrates mortality data collected by the official censuses of 2000 and 2010. The objective is the generation of patterns of interest based on the clustering of districts with high mortality rates for different causes of death. According to the specialized literature, few studies related to data mining applications using census databases have been reported, despite the potential census databases have. Contributions of this work are the implementation of a data preparation subsystem and the integration of a data warehouse that contains records of deaths, occurring in 2000 and 2010 for 2049 different causes of death. In order to validate the results, we analyzed four causes of death related to cancer C16 (stomach) and C34 (lung), and diabetes mellitus E11 (no insulin-dependent), and E14 (unspecified). Experimental results were satisfactory and they show some patterns of interest and an increase in the mortality rate in 2010 compared to 2000.**

Keywords - Data mining application; Mortality data; Census database.

I. INTRODUCTION

Censuses are important because the information collected is used to understand the reality that a country lives; in addition, this information is used by different organizations and institutions of the public, private and social sector [1]. In turn, it allows us to analyze and to understand the problems that occur in several areas of a country, e.g. economy, housing, health. In the public health area, this information is important for the organizations that manage public health services, allowing to identify vulnerable sectors in a population, managing and assigning the resources adequately, and implementing appropriate strategies to combat these diseases.

In Mexico, as in other countries, censuses collect information about different aspects of a given population, e.g. health, educational, economic and social features. Censuses help to support the research performed by academic and educational institutes in the socio-demographic field [2].

There are a few studies related to epidemiological data where data mining is used as a tool to explore them and cartographic visualization systems are used to represent the extracted knowledge. Data analysis performed in [3] shows a clear image of the concentration of cases of breast cancer in New Mexico. The results allow identifying the groups in populations using spatial data. This work is just focused on data related to a single state: New Mexico, USA. This represents an important drawback regarding the amount of available information.

The works described in [4] and [5] are part of a project that uses epidemiological and spatial data in order to identify, establish, and display, cartographically, possible relationships between patients with cancer and their proximity to factories and cell phone antennas. However, this work is closely linked to this hypothesis, and do not contemplate incidence or mortality rates.

The works [6] and [7] have used data mining in order to extract information from epidemiological data, however; the extracted knowledge and its representation follow a different approach to the one presented in this research.

This research takes part of a bigger data mining project in the epidemiological domain. Previous works [8], [9], and [10] were developed specifically for analyzing mortality causes related with cancer; these researches are an important antecedent and are closely related to our data mining objective. In [11] [12], a Data Warehouse was implemented and a data mining tool was developed in order to generate and to display patterns of interest represented as groups of districts with high rates of cancer mortality in maps of Mexico.

This paper shows the experience of a specialized data mining process whose primary objective is the generation of patterns of interest shown as groups of districts with high mortality rates. The process has been developed for several causes of death among which we have selected four causes related to cancer and diabetes mellitus as a practical case of study. As an additional contribution, a subsystem focused on the data preparation step for the epidemiological domain and a data warehouse, that integrates data from the official censuses, were implemented.

This document is organized as follows: Section II shows a brief description of the analysis, data preparation, and data integration performed on the mortality data. A data preparation subsystem is described in Section III. Section IV describes the cartographic visualization subsystem which displays maps of Mexico with groups of districts with high mortality rates for different causes. Section V shows the results obtained and the contributions made by this research. Conclusions and future work are shown in Section VI.

II. CENSUS DATABASES DESCRIPTION AND PREPROCESSING

We have collected databases from official censuses. In the epidemiological domain, data from census carried out by official organisms such as Statistics and Geography National Institute (INEGI, Instituto Nacional de Estadística y Geografía) in Mexico allow us to analyze and to understand the behavior of diseases and how they affect a population, showing information about what diseases have higher recurrence and the number of deaths occurred because of those diseases.

However, it is important to completely exploit this information in order to obtain more significant knowledge that permits health organisms to make decisions to prevent these diseases or to improve the living conditions of those who suffer them. Data mining is a tool that facilitates the task of extracting knowledge from data, which may be useful to understand a phenomenon.

The knowledge extracted represents information of interest to health organisms not only in Mexico, but also in other parts of the world where the censuses are carried out. The above mentioned represents a possibility to apply this project to other countries. In the following sections, the data mining process followed in this research is briefly described:

A. Database description

The data were extracted from different official information sources from Mexico. The information sources and the data description are shown below:

- Mortality database: records of deaths occurring in 2000 and 2010 for different causes of death [13], extracted from: National Health Information System (SINAIS, Sistema Nacional de Informacíón en Salud).
- Geographic database: records of the geographical position of the districts of Mexico [14], extracted from: Database District System (SIMBAD, Sistema Municipal de Bases de Datos).
- Population database: records of the total population by districts in Mexico, for 2000 and 2010 [15], extracted from: INEGI.
- International catalogue of diseases (CIE-10) [16]: code classification and names of diseases. It includes 2049 causes of death (Updated to 2009), extracted from: Collaborating Center for the Family of International Classifiers (CEMECE, Centro Colaborador para la Familia de Clasificadores Internacionales de la OMS en México).

SINAIS and SIMBAD systems have data provided by INEGI. Table I shows the number of attributes and records of the databases for 2000 and 2010.

TABLE I. NUMBER OF ATTRIBUTES AND RECORDS.

Database	**Attributes**		**Records**	
	2000	***2010***	***2000***	***2010***
Mortality	38	40	437,667	592,018
Geographic	7		2475	
Population	3		2475	
CIE-10	24		2049	

B. Data analysis and pre-processing

Databases were analyzed in order to understand the database schema, to select the attributes of interest, and to understand the meaning of these attributes. This analysis was performed using a data description file, which is provided by the information sources from where the databases were extracted.

Additionally, the data analysis allowed us to identify attributes with errors which should be corrected or deleted in order to prevent the knowledge extracted by the data mining process to be wrong. In a first review, attributes with null values or empty attributes were deleted for all tuples; dependent attributes were identified and deleted using a correlation analysis. Then, the attributes of interest, needed to achieve the data mining goal, were selected by an expert domain.

Records that do not represent information of interest to the data mining goal, e.g., records related to districts with population less than 100,000 inhabitants, were eliminated.

Finally, the data construction tasks were performed. In this case, two new attributes, needed to reach the data mining objective, were identified and incorporated. The specific operations to calculate them are: calculation of the mortality Incidence and the calculation of the Mortality Rate; these are relevant indicators for the epidemiological domain experts.

The mortality *Incidence* is the number of cases observed for a particular disease in a specific district, in a particular year. The calculation of the *Mortality Rate* is performed for each district with population over 100,000 inhabitants, by convention in the health area, using the Expression (1):

$$\text{Rate} = \frac{\mathit{Incidence}}{\mathit{Population}} * 100{,}000 \qquad (1)$$

where *Population* is the number of inhabitants in a district for a specific year. This value is extracted from the population data provided by INEGI.

C. Data integration

A data warehouse is a database that integrates data from one or more information systems in an organization and it is oriented to assist in the decision making process [17]. The data were integrated in a data warehouse with a similar structure to that proposed in [18]. A data warehouse stores historical and non-volatile information related with a fact, in this case, the information is related to all deaths occurred in 2000 and 2010, in districts with a population over 100,000 inhabitants. Additionally, the data warehouse structure facilitates the integration of data for different years.

The data warehouse includes three dimensions. The *fact* table stores the results of the operations to calculate the mortality Incidence and the Mortality Rate for a particular disease among 2049 causes contained in the CIE-10.

Figure 1 shows the multidimensional model of the data warehouse. It is represented as a bucket with three views or dimensions that include CAUSE of death related to SPACE (districts) and TIME (year) of occurrence.

III. DATA PREPARATION SUBSYSTEM

In order to automate the data preparation process for the epidemiological domain, a data preparation subsystem was developed. The subsystem was implemented on Java language using Java Database Connectivity (JDBC) and SQL (Structure Query Language).

JDBC allows to establish the connection between the java language and the data warehouse; SQL standard allows to write queries to access data stored in it.

Figure 2 shows the conceptual model of the data preparation subsystem. The subsystem contains two modules that automate some tasks of data construction and data integration. A brief description of these modules is provided below:

A. *Module of data construction tasks*

In this module, the subsystem access to the data stored in the data warehouse and selects all data related to a specific cause of death for a particular year, and then the operations to calculate the mortality Incidence and Mortality Rate are executed. For each operation the subsystem executes:

- Calculation of mortality Incidence. A query sentence that counts all records related to deaths occurred for a specific cause of death in a particular year.
- Calculation of Mortality Rate. The results obtained for the mortality incidence are used to execute arithmetic operations to calculate the Mortality Rate using Expression 1 (showed in Section II.B).

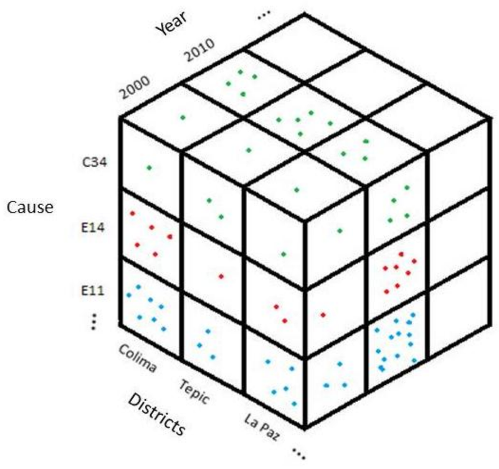


Figure 1. Conceptual view of the data warehouse.

The subsystem executes the previous tasks and the results are stored in the *fact* table described in Section II.C.

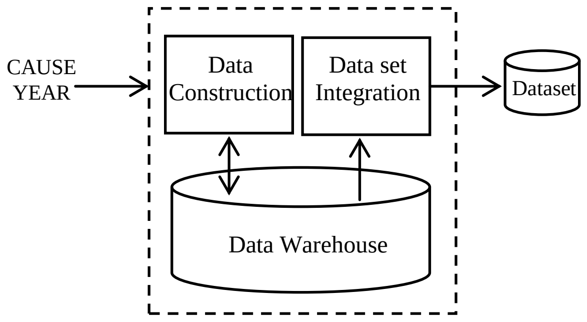


Figure 2. Data preparation subsystem.

B. *Module of final dataset integration*

Additionally, the subsystem automates the creation of a dataset whose structure is described in Table II. The data preparation subsystem takes, from the *fact* table, all the records related to CAUSE and YEAR of death introduced as input.

The dataset structure has four attributes; the first one contains the value for the cause of death to be analyzed, second and third columns contain the values of latitude and longitude of the district used to find its position on the map. The last column corresponds to the calculation of the mortality rate.

The subsystem has the ability to perform these calculations and to build the dataset, receiving as input only the values of CAUSE and YEAR of death; the tests were performed changing these values. For CAUSE, the causes tested were C16, C34, E11, and E14; for YEAR 2000 and 2010.

IV. CARTOGRAPHIC VISUALIZATION SUBSYSTEM

The dataset generated by the data preparation subsystem is used as input by the cartographic visualization subsystem described in [4]; this subsystem contains two modules: a pattern generator and a cartographic display.

The pattern generator takes the dataset (described in Section 4), converts it into an .arff file and, using the K-means clustering algorithm implemented on Weka, classifies the *n*-items into a given number of *k*-groups of districts with similar "Latitude", "Longitude", and "Mortality rate" for a specific cause of death. The results of the clustering algorithm are lists of element groups including the centroid of each group.

TABLE II. DATA SET STRUCTURE.

Cause	Latitude	Longitude	Mor_Rate
Cause of death	Latitude of the district. Position on the map.	Longtude of the district.Position on the map.	Mortality rate.

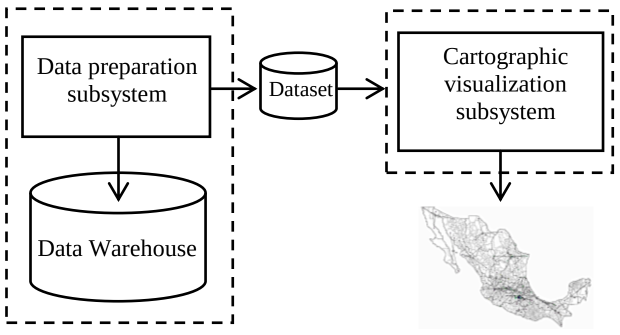

Data preparation subsystem
Dataset
Cartographic visualization subsystem
Data Warehouse

Figure 3. Interaction between the data preparation subsystem and the cartographic visualization subsystem.

The cartographic display permits to select and draw a map of Mexico with all the groups generated after analyzing the data with k-means clustering algorithm. Figure 3 shows the interaction between the data preparation subsystem and the cartographic visualization subsystem.

A set of experiments were conducted with the cartographic visualization subsystem for the same causes tested by the data preparation subsystem. We have used the k-means algorithm and established the value of *k* parameter in 15, 20 and 25 for each cause corresponding to data in 2000. For data in 2010, the *k* parameter was established in 20, 25, and 30 because in 2010, the number of districts was greater than in 2000. In both cases, the best result was obtained with *k* =20.

Figures 4 and 5 show the maps displayed by the cartographic visualization subsystem for the mortality cause E11 related to diabetes mellitus no insulin-dependent. We have highlighted three patterns that represent the groups of districts with the higher mortality rates for this cause in 2000 and 2010.

Figures 6 and 7 show the maps displayed by the cartographic visualization subsystem for the mortality cause E14 related to unspecified diabetes mellitus. We have highlighted three patterns that represent the groups of districts with the higher mortality rates for this cause in 2000 and 2010.

V. RESULTS AND CONTRIBUTIONS

We have implemented a data warehouse that contains information about all deaths occurred in districts of Mexico with populations over 100,000 inhabitants in 2000 and 2010. In addition, we have implemented a data preparation subsystem which has the capacity to generate prepared datasets for 2049 different causes of death registered in the CIE-10.

In order to validate the results obtained by the automated data preparation subsystem, we have performed tests for the causes of death C16 (stomach cancer) and C34 (lung cancer), and we have compared these results with the results obtained by previous works [5], [6], and [7] where the data preparation process were manually performed. The values generated for the mortality *Incidence* and the *Mortality Rate* by the data preparation subsystem correspond exactly to the values obtained in previous works. Table III shows the results obtained for the first group identified in previous works as a group of interest for the cause of death C16 in 2000:

TABLE III. GROUP OF INTEREST FOR THE CAUSE C16 IN 2000.

District	Incidence	Mortality rate
Rio Bravo	14	13,43
Matamoros	54	12,91
Torreon	65	12,27
Monterrey	113	11,97
Piedras Negras	15	11,7
San Nicolas de los G.	53	10,67
Reynosa	42	9,98
Gomez Palacio	27	9,88
Santa Catarina	21	9,25

Table IV shows the results obtained for the first group identified for the cause of death C34 in 2000 in previous works:

TABLE IV. GROUP OF INTEREST FOR THE CAUSE C34 IN 2000.

District	Incidence	Mortality rate
Guaymas	15	11.52
Hermosillo	48	7.87
La Paz	14	7.11
Los Cabos	7	6.64

Another important contribution is the automation of the data preparation tasks that calculate the mortality *Incidence* and the *Mortality Rate*, which represented the largest effort in the data preparation process. We carried out tests for four causes of death (C16, C34, E11 and E14) for 2000 and 2010, Table V shows a comparison between the time required to perform manually and automatically these tasks:

TABLE V. TIME COMPARISON.

Task	Manual average (min)	Automatic average (min)	% Time reduction
Calculation of Incidence	53.255’	.101’	99.81
Calculation of mortality rate	5.5’	.544’	90.11

Additionally, other experiments were performed; datasets with information for the causes of death E11 and E14 for the years 2000 and 2010 and their corresponding maps (Figures 4, 5, 6 and 7) were generated. Group three (Figure 6c) is the one with the largest average mortality rate, but this group is not meaningful because its districts are more dispersed than in other groups.

VI. CONCLUSIONS AND FUTURE WORK

Data collected by official censuses in a country permits to analyze interesting aspects of a population. In the health domain, census databases allow us to have a perspective of how diseases affect some sectors of the population and the evolution of these diseases through time and space.

We have used official census databases from 2000 and 2010 and applied a specialized data mining process in order to analyze different causes of death related to cancer and diabetes.

In the causes of death selected for cancer (C16 and C34) and diabetes (E11 and E14), we observed an increase in the mortality rates in 2000 compared to 2010. For the cause of death E11, we have observed in the maps an increase in the number of districts with high mortality rate and the concentration of these districts in the central region of the country (see Figure 6b).

Other patterns of interest were identified for the cause of death E14. In 2000 a group of districts in the state of Chihuahua were identified (see Figure 6a). Figure 7a shows another group of interest in the northwest region which has the second largest mortality rate; its districts are located between the states of Sonora and Baja California.

As a future work, we propose to make a trend analysis of deaths occurred between 2000 and 2010 for different causes of death. Additionally, we propose the integration of the data preparation subsystem and the cartographic visualization subsystem.

REFERENCES

[1] Consumer's Federal Attorney, PROFECO. Available: http://www.profeco.gob.mx/encuesta/brujula/bruj_2005/b05_censos.asp. Last visited: December, 2012.

[2] Censuses and population counts. Available: http://www.inegi.org.mx/est/contenidos/proyectos/ccpv/presentacion.aspx. Last visited: December, 2012.

[3] B. Zhan "Monitoring geographic concentration of female breast cancer using cluster analysis: the case of New Mexico", Papers and Proceedings of Applied Geography Conference (AGC 02). Vol 25, New York, Oct. 2002, pp 1-8.

[4] V. Bogorny, P. Engel, and L. Alvares "Spatial data preparation for knowledge discovery", IFIP Academy on the state of software theory and practice – PhD Colloquium. Porto Alegre, Brazil, 2005.

[5] V. Bogorny, P. Engel, and L. Alvares "A reuse-based spatial data preparation framework for data mining", International Conference on Sotware Engineering and Knowledge Engineering (SEKE 05), Taiwan, July 2005, pp. 649 - 652.

[6] N. Labib and M. Malek "Data mining for cancer management in Egypt case study: childhood acute lymphoblastic leukemia", Transaction on Engineering Computing Technology, Vol 8,Oct. 2005, pp. 309-314.

[7] M. Izadi, D. Buckeridge, and K. Charland, "Mining epidemiological data sources in H1N1 pandemic using probabilistic graphical models", International Conference on Advances in Information Mining and Management (IMMM 11), Spain, Oct. 2011, pp. 1-6.

[8] J. Salinas, Adaptation of a data mining methodology for its application to a real population-based database of cancer records, 2007. Master thesis, Computer science. Cenidet, Cuernavaca, Mexico.

[9] A. Mexicano, Development of a methodology for feature selection and indicator generation for the application of data mining to a real population-based cancer database, 2007. Master thesis, Computer science. Cenidet, Cuernavaca, Mexico.

[10] M. Barron, Development of a prototype for the application of data mining techniques on a real population-based cancer database, 2008. Master thesis, Computer science. Cenidet, Cuernavaca, Mexico.

[11] J. Pérez et al., "Spatial Data Mining of a Population-Based Data Warehouse of Cancer in Mexico", International Journal of Combinatorial Optimization Problems and Informatics (IJCOPI 10). Vol 1, May 2010, pp. 61 – 67.

[12] J. Pérez, O. Fragoso, R. Santaolaya, A. Mexicano, and F. Henriques, "A data mining system for the generation of geographical C16 cancer patterns" International Conference on Software Engineering Advances (ICSEA 10), France, Aug. 2010, pp. 417-421.

[13] National Health Information System, SINAIS. Available: http://www.sinais.salud.gob.mx/basesdedatos/estandar.html. Last visited: December, 2012

[14] Database District System, SIMBAD. Available: http://sc.inegi.org.mx/sistemas/cobdem/contenido-arbol.jsp. Last visited: December, 2012

[15] Statistics and Geography National Institute, INEGI. Available: http://www.inegi.org.mx/. Last visited: December, 2012

[16] Collaborating Center for the Family of International Classifiers, CEMECE. Available: http://www.cemece.salud.gob.mx/fic/cie/index.html. Last visited: December, 2012

[17] J. Moral, "Introduction to data Warehousing", Annals of Mechanical and Electrical. Vol. LXXXVI, Spain, Mar. 2009, pp. 42-47.

[18] R. Boone, Identification of regions with high rate of cancer incidence by integrating and using data mining techniques, data warehouse, clustering and geographic information systems, 2011. Master thesis, Computer science. Cenidet, Cuernavaca, Mexico.

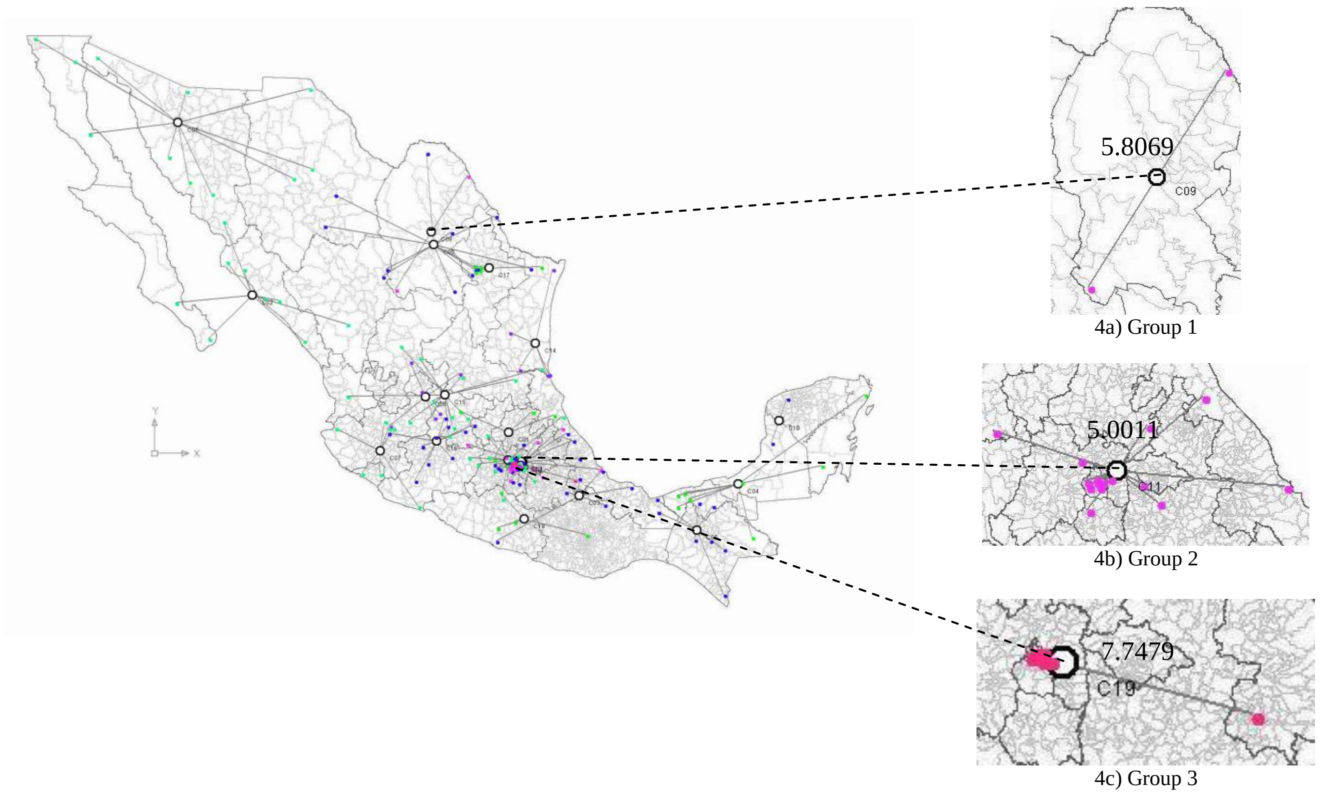


Figure 4. Identification of groups of interest for the cause of death E11 (diabetes mellitus no insulin-dependent) in 2000. The groups 1, 2, and 3 have the highest mortality rates. Figures 4a, 4b, and 4c show the average value for each group.

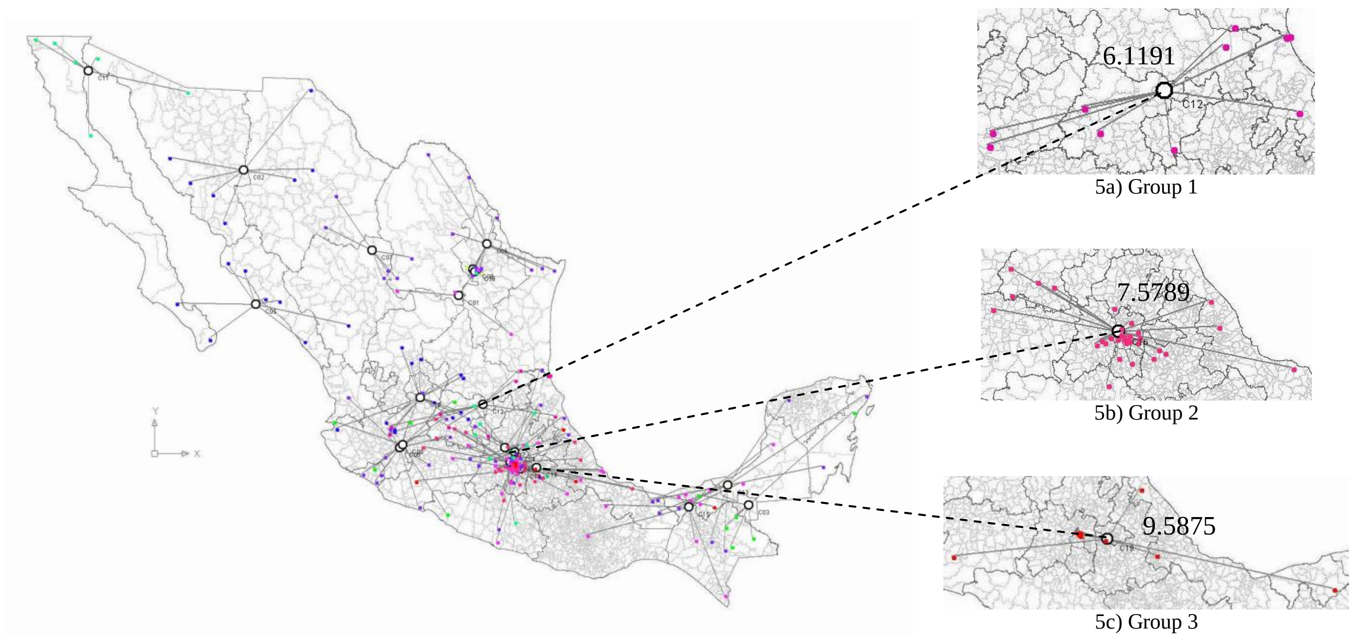


Figure 5. Identification of groups of interest for the cause of death E11 (diabetes mellitus no insulin-dependent) in 2010. The groups 1, 2, and 3 have the highest mortality rates. Figures 5a, 5b, and 5c show the average value for each group.

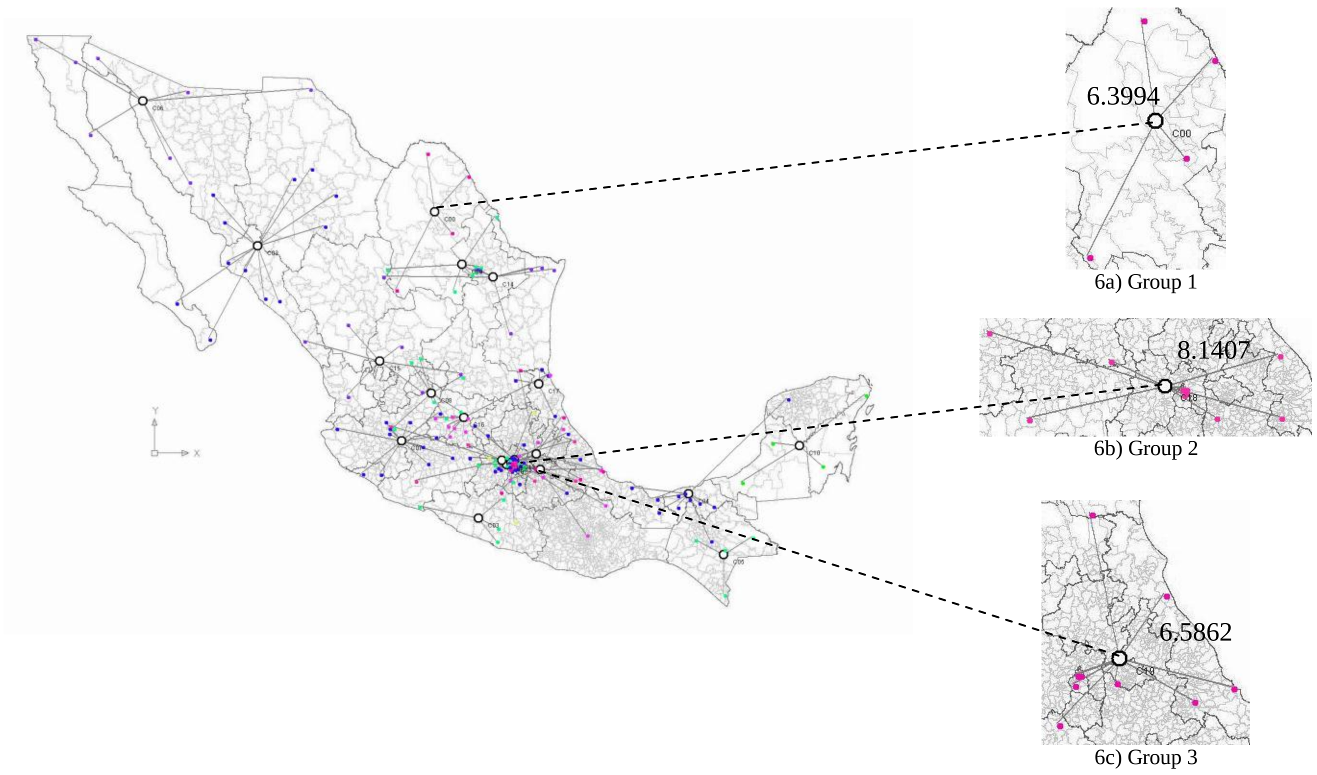


Figure 6. Identification of groups of interest for the cause of death E14 (unspecified diabetes mellitus) in 2000. The groups 1, 2, and 3 have the highest mortality rates. Figures 6a, 6b, and 6c show the average value for each group.

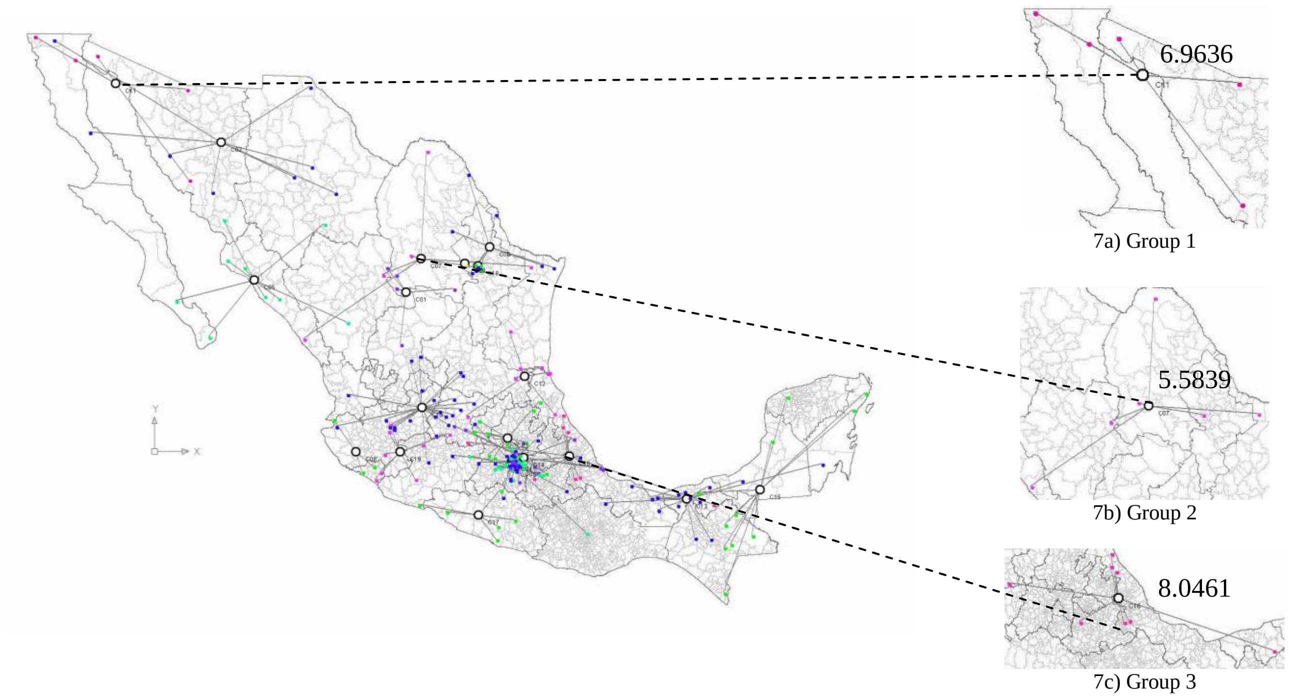


Figure 7. Identification of groups of interest for the cause of death E14 (unspecified diabetes mellitus) in 2010. The groups 1, 2 , and 3 have the highest mortality rates. Figures 7a, 7b, and 7c show the average value for each group.

Clustering XML Data Streams by Structure based on Sliding Windows and Exponential Histograms

Mingxia Gao
College of Computer Science
Beijing University of Technology
Beijing, China
gaomx@bjut.edu.cn

Furong Cheng
R&D Center TravelSky
Technology Limited
Beijing, China
bjcfr@163.com

Abstract—**To group online XML data streams by structure, this paper introduces an algorithm named the CXDSS-SWEH. It is a dynamic clustering algorithm based on sliding windows and exponential histograms. Firstly, the algorithm formalizes an XML document into a structure synopsis named Temporal Cluster Feature for XML Structure (TCFXS). Secondly, it allots the TCFXS to some cluster through measuring similarities between the TCFXS and each existing cluster. At last, updating clusters in sliding windows are real-time modified through criterions of false positive exponential histograms. We have conducted a series of experiments involving real and simulative XML data streams for validating empirical effects on clustering quality, memory and time consumption. Our experimental results have confirmed: (1) clustering quality of the CXDSS-SWEH is close to the methods XCLS and SW-XSCLS; (2) memory and time consumption of the CXDSS-SWEH are efficient and effective, compared to the SW-XSCLS.**

Keywords-**XML data stream; temporal cluster feature**

I. Introduction

The eXtensible Markup Language (XML) [1] is a self-description language used for data exchange and sharing. It has become more prevalent on the Web after recommended by W3C in February 1998 [8]. A lot of applications and services produce huge amounts of online XML data streams. Examples abound from online network monitoring to stock market updates. To analyze this category of data, many researchers focus on clustering XML data and propose a number of algorithms [3]–[12]. However, existing clustering methods mainly focus on static XML data and, generally, need to scan data many times. Though technologies of mining XML data have recently been extended to XML data streams [5], [7], [10], [11]. Most of methods query XML streams through different methods [5], [7], [10].

This paper proposes a dynamic algorithm named the CXDSS-SWEH to cluster online XML data streams by structure. It uses technologies of sliding windows and exponential histograms. Firstly, the algorithm formalizes an XML document into a structure synopsis named Temporal Cluster Feature for XML Structure (TCFXS). Secondly, it allots the TCFXS to some cluster through evaluating similarities between the TCFXS and each existing cluster. In fact, each existing cluster in sliding windows is a team of TCFXSs, which must satisfy criterions of false positive exponential histograms. At last, updating clusters in sliding windows are real-time modified through merging, deleting and so on. To validate empirical effects, we have conducted a series of experiments involving real and simulative XML data and compared with the static clustering method XCLS in the literature [8] and the stream clustering method SW-XSCLS in the literature [11] and accepted some promising results.

The remainder of this paper is organized as follows. Section 2 surveys the related work. Section 3 describes the problems and basic ideas. Section 4 presents a structure synopsis and a method of computing similarity between two synopses. Section 5 discusses the online clusters maintained in sliding windows. Section 6 provides experimental data, design, and results. Section 7 concludes the paper.

II. Related Work

Existing technologies of clustering XML data mainly focus on static data. A great lots of methods for computing similarities between XML documents have been developed, ranging from various tree edit distance methods [3], [9], [12] to direct extracting document feature approaches [4], [6], [8].

The basic idea in all of these tree edit distance algorithms is to find the minimum cost of transforming a tree to another tree by using edit operators. A key differentiator of most of algorithms is the set of edit operations allowed or the structure of trees. For example, Theodore et al. [3] exploited the tree nature of XML documents and provided techniques for tree matching, merging and pruning. Nierman and Jagadish [9] proposed a method of edit distance including five kinds of operators. Zheng et al. [12] described an XED distance to evaluate difference between documents.

Different from edit distance methods, many approaches use other kinds of features to present XML documents, then directly measure similarities between features. Flesca [4] showed structure of XML documents into a time serial, appearance of every tag into an impulse and computed similarities between documents by the frequency domain of Fourier transform. Wang et al. [6] built an S-GRACE based on structure of documents and proposed a method of computing similarities between documents by graph matching. Nayak [8] modelled structure of XML documents into a Level Structure (LF), which can be seen as a simple ordered labelled tree and proposed a method of directly computing similarities between LSs.

The above methods generally need to scan and parse data many times. But, XML data streams only are permitted to scan or parse once. So, these static methods are not directly suitable for XML data streams.

Recently, technologies of mining XML documents has been successfully extended to XML data streams. But, these methods mainly focus on query fields. Koch and Scherzinger [5] introduced the notion of XML Stream Attribute Grammars (XSAGs), which is the first scalable query language for XML data streams. Yang et al. [10] built a SAX-based XML data streams query evaluation system and designed an algorithm that consumes buffers in line with the concurrency lower bound. Mayorga et al. [7] presented a method for building a stream synopsis to approximately query XML streams.

Mao et al. [11] proposed a clustering stream method named the SW-XSCLS. It extended LS features in the literature [8] to present stream synopses, and used sliding windows to maintain features as the CXDSS-SWEH in this paper. However, two methods have important differences as follows. (1) Methods of computing similarities are different. The SW-XSCLS uses the original method in the literature [8]. The CXDSS-SWEH proposes a new feature named Node List and a method of computing similarities between features. (2) Steps of combining two clusters are different. The SW-XSCLS uses same steps as those in the literature [2] and the CXDSS-SWEH presents a new method to save usable memory.

III. Problem Statement

A. XML Data Stream

An XML data stream in query fields often is defined as a massive, continuous sequence of tokens in XML documents. Among, tokens, respectively, denote beginning tokens, ending tokens of elements, and actual values of elements or attributes. The paper focuses on clustering structure between XML documents. So, an XML data stream is defined as a massive, continuous sequence of documents, as shown in definition 1.

Definition 1: Let $S = \{X_1, \ldots, X_i, \ldots\}$ be an XML data stream, and $\{< T_1, E_1 >, \ldots, < T_i, E_i >, \ldots\}$ be time stamps of these documents. T_i and E_i, respectively, denote the timestamps of document beginning and ending, where $i < m$, such that $T_i < T_m$, $E_i < E_m$. X_t is a sequence of tokens produced by parsing an XML document based on SAX.

B. Clustering XML Data Streams by Structure

Important differences between data streams and traditional data sets are that arriving data units are massive, continuous and infinite. To adapt to these characteristics, features to represent and analyze these data general use approximate formats. For semi-structured XML documents, approximate formats can be structure, content, or structure+content. The clustering method in this paper only focuses on structure synopses and omits all content information. It defines a Temporal Cluster Feature for XML Structure (TCFXS) as a structure synopsis. The detailed information about TCFXSs will be addressed in Section 4.

A sliding window mode is a better method of solving massive or infinite units in data streams. It only considers and deals with nearly arriving units at any time. It emphasizes that importance of units in a stream will wear off with time. To real-time maintain units in sliding windows, new added units and overdue units must be managed timely. The technology of exponential histograms is one of methods to manage data synopses in sliding windows. This paper uses the online method in the literature [2] for reference and defines an Exponential Histogram of Temporal Cluster Feature for XML Structure (EHTCFXS) to manage structure features in sliding windows. An EHTCFXS is a team of TCFXSs based on criterions of false positive exponential histograms. The detailed definition and maintaining algorithm will be addressed in Section 5.

Based on the above idea, clustering XML data streams by structure based on sliding windows and exponential histograms can be shown in definition 2. The definition tells us that there are three key problems must be solved to cluster online XML data streams, including of building TCFXSs, measuring similarities between TCFXSs, and maintaining EHTCFXSs in sliding windows timely.

Definition 2: Given an XML data stream in a sliding window at time t, the clustering solution, denoted by $C = \{C_1, C_2, \ldots, C_q\}$, is a partition of n XML documents, where C_i can be represented by an EHTCFXS. Among, $EHTCFXS = \{TCFXS_0, \ldots, TCFXS_i\}$; n is the number of XML documents in the stream; q is the number of clusters. The partition must satisfy two following rules: (1) $TCFXS(C_1 \cup C_2 \cup \ldots \cup C_q) = TCFXS(X_i \cup X_{i+1} \cup \ldots \cup X_{i+n-1})$ and (2) $TCFXS(C_i) \cap TCFXS(C_j) \neq TCFXS(X_i)$.

IV. Temporal Cluster Features

A. Temporal Cluster Features

Heterogeneous XML documents are basic units of XML documents streams. These XML documents with different structure and contents imply complex hierarchy and semantic information. To extract cluster features by structure, many secondary information can be overlooked. This paper defines a structure feature, called Node List, for a set of XML documents. A Node List is a list of pairs (node code and level value). They, respectively, present distinct elements in documents and the hierarchy of each element in own document.

A distinct integer in a Node List replaces with the name of each distinct element in XML data streams. It denotes the first appearance order for the start event of a element in a start events stream (only recording start events) in a sliding window. Figure 1-(a) presents an example of a sequence of tokens parsed through SAX and the order of these start events, where the sequence of tokens is the first document arriving at sliding windows. The coding method can ensure that elements with same name in different documents use a same integer. A global name-index list in sliding windows will be defined and used to code names of distinct elements in XML data streams. It consists of pairs (element name and coding integer), as shown in Figure 1-(b). The coding integer of an element name will be acquired by searching the global name-index list. The corresponding integer is its coding integer when finding the element name from the global name-index list; otherwise existing max value plus one is its coding integer. At same time, a pair (the element name and the coding integer) is inserted in the name-index list as a new node. The level values of Node Lists can be acquired by a runtime-stack technology in the literature [13]. Figure 1-(c) gives an example to illustrate the process of acquiring level values by a runtime-stack. The start event of each document activates a null stack. Start events and end events of elements, respectively, inspire pushing and popping. The level value of an element just is the corresponding pointer value of the element in the stack. Figure 1-(d) presents the Node List formalized by the XML document in Figure 1-(a).

Definition 3: Given an XML data stream $S = \{X_1, \ldots, X_n\}$, let $(NodeList_{1\to n}, n, t)$ be a Temporal Cluster Feature for XML Structure (for short TCFXS(S)), where $NodeList_{1\to n} = \sum_{i=1}^{n} NodeList_i = NodeList(X_1 \cup X_2 \cup \ldots \cup X_n)$; n is the number of XML documents included in S; t is the time stamp T_n of the newest arriving XML document in S.

Two Node Lists can be combined by property 1 on basis of the above definition of Node Lists. The actual operating steps include comparing ordered integers, and inserting a new node. Figure 2 presents a schematic illustration of combining two Node Lists. The combining result only contains one copy of integer 1 and 2 because of they being a same level, and contains two copies of integer 3 and 5 with different level values. These integers in the combining Node List still satisfy partial order. The definition 3 is a Temporal Cluster Feature for XML Structure on basis of Node Lists. Two TCFXSs without time overlap in an XML data stream also are combined by property 2.

Property 1: Given two Node Lists, the result of combining them can be acquired by uniting all elements of every Node List, such that repeating elements of same level value and same code only keeps one copy.

Property 2: Given two TCFXSs without time overlap TS_1 and TS_2, the result of jointing two TCFXSs TS_3 can be acquired by combining two Node Lists, adding two number of documents, and maximizing two time stamps as following:
$TS_3.NodeList = TS_1.NodeList \cup TS_2.NodeList$,
$TS_3.n = TS_1.n + TS_2.n$,
$TS_3.t = Max\{TS_1.t, TS_2.t\}$.

B. Similarities between Two TCFXSs

The equation (1) is defined to compute similarities between two Node Lists. Its range is [0,1], among 1 denotes that two Node Lists are same, otherwise 0 denotes that two Node Lists have not any common element. The comparing progress of two Node Lists accords with the comparing progress of ordered integers. The progress can be described as the following: (1) node values are basic moving units; (2) if a node value in Node List 1 is less or equal to that in Node List 2, the Node List 1 is moved to its next node value and the progress continues, otherwise, the Node List 2 is moved to its next node value and the progress continues. Two Node Lists cannot back up in the comparing progress. So, the time complexity is $O(max\{N_1, N_2\})$ in the worst state, where N_1 and N_2, respectively, represent the total number of all elements in Node List 1 and Node List 2. As definition 3, at time t, structure synopses in a sliding window are TCFXSs including Node Lists. So, the similarity between two TCFXSs is equal to the similarity between corresponding Node Lists in the two TCFXSs.

$$\begin{aligned} NodeSim_{1\leftrightarrow 2} &= \frac{ComWeight_1 + ComWeight_2}{ObjWeight_1 + ObjWeight_2} \\ &= \frac{\sum_{i=1}^{M_1}(1/r)^{L_1^i} + \sum_{j=1}^{M_2}(1/r)^{L_2^j}}{\sum_{k=1}^{N_1}(1/r)^{L_1^k} + \sum_{k=1}^{N_2}(1/r)^{L_2^k}}, \end{aligned} \tag{1}$$

Where $ComWeight_1$ and $ComWeight_2$, respectively, denote the total weight of common elements in Node List 1 and Node List 2. $ObjWeight_1$ and $ObjWeight_2$, respectively, denote the total weight of all elements in Node List 1 and Node List 2. M_1 and M_2, respectively, represent the sum of occurrences of common elements in Node List 1 and Node List 2. N_1 and N_2, respectively, represent the sum of all elements in Node List 1 and Node List 2. L_1^i is level value of the i^{th} common element in Node List 1. L_2^j is the level value of the j^{th} common element in Node List 2. L_1^k and L_2^k, respectively, represent the level value of the k^{th} element in Node List 1 and Node List 2. r is the increasing factor of weight proposed by users.

V. Maintaining TCFXSs in Sliding Windows

A. EHTCFXSs

Definition 4: Let an EHTCFXS be a team of TCFXSs in sliding windows, that is $EHTCFXS =$

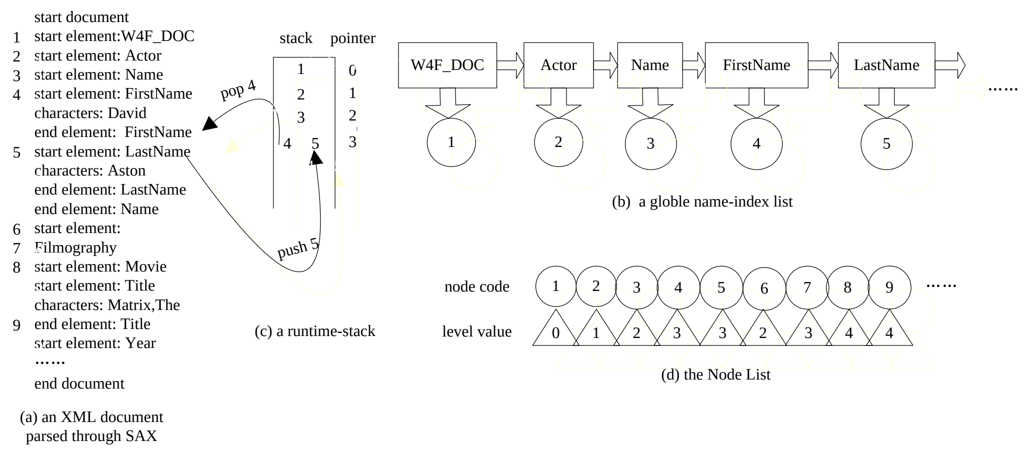


Figure 1. An example of acquiring coding values and level values for an XML document

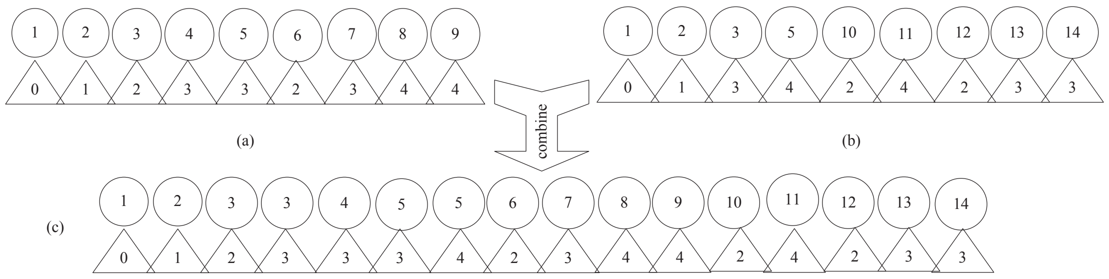


Figure 2. Combining Node Lists

$\{TCFXS(S_0), \ldots, TCFXS(S_i), \ldots\}$. $TCFXS(S_i)$ is the structure synopsis of the i^{th} sub-stream $S_i = \{X_{i_1}, \ldots, X_{i_n}\}$ with time stamp $T_{i_1}, \ldots, T_{i_n}$, such as $T_{i_j} < T_{i_m}$, where $j < m$.

To real-time maintain TCFXSs in sliding windows, an Exponential Histogram of Temporal Cluster Feature for XML Structure (for short EHTCFXS) in definition 4 can be seen as a cluster. The corresponding sub-stream of every TCFXS in an EHTCFXS is appointed to a grade value, which are denoted by superscripts such as $S_1^m, S_2^m, \ldots$. The grade value is related to the total number of documents in every sub-stream. The total number of documents in every sub-stream and the total number of sub-streams in each grade must adhere to four criterions of false positive exponential histograms in the literature [2] as following: (1) the timestamps of all XML documents in S_i are less than those in S_j, where $i < j$; (2) the number of documents in any sub-stream S just is $2^0 = 1, 2^1 = 2, 2^2 = 4 \ldots$, and the new arriving sub-stream S_n only includes one document; (3) the grade value of S^j is j , where the number of documents in S^j is 2^j. The total number of sub-streams in each grade must be $[\frac{1}{\varepsilon}]$ or $[\frac{1}{\varepsilon}+1]$, as ε is error parameter proposed by users; (4) all documents in every sub-stream after S_1 are not overdue, where $TCFXS(S_1).t$ is in effect. The first criterion is used to ensure that TCFXSs in an EHTCFXS have not time overlap. The second criterion is an inherent requirement of an exponential histogram. The third criterion is used to limit the number of documents in every sub-stream. Besides, it uses an optional parameter (defined by users) to limit the total number of sub-streams in each grade. The forth criterion denotes how to identify overdue documents. When the first criterion is in effect, we only need to detect the oldest TCFXS in an EHTCFXS for identifying overdue documents.

B. Grouping XML Data Stream by Structure

At any time t, multi-EHTCFXSs are maintained in a sliding window. When an XML document X_p arrives to the sliding window, three interrelated processes are used to decide which cluster X_p will belong to. Firstly, X_p is parsed and rebuild into a $TCFXS(X_p)$. Secondly, similarities between each existing cluster and $TCFXS(X_p)$ are computed and the largest similarity value sim_{max} and the corresponding $EHTCFXS_{max}$ are selected. At last, one of two managing methods is implemented according to the result of comparing sim_{max} with the least similarity threshold ω. $TCFXS(X_p)$ is merged into the existing cluster $EHTCFXS_{max}$, where $sim_{max} \geq \omega$. Otherwise, a new EHTCFXS only including $TCFXS(X_p)$ is built.

The steps of merging $TCFXS(X_p)$ into

$EHTCFXS_{max}$ are introduced as follows: (1) build a new 0-grade $TCFXS(S^0)$ based on definition 3, where $S^0 = \{X_p\}$; (2) add $TCFXS(S^0)$ into the set of 0-grade sub-streams in $EHTCFXS_{max}$. If the total number of 0-grade sub-streams is $[\frac{1}{\varepsilon}+2]$, two oldest TCFXSs in the set are merged into a new 1-grade by property 2; (3) repeat the step 2 for sets of the different grade sub-streams until the total number of each grade sub-streams meet with the 3th criterion of false positive exponential histograms described in Section 5.1; (4) compare the total number of documents in the sliding window with the size of the sliding window N, and then delete overdue TCFXSs and corresponding EHTCFXS where the total number of documents exceeds N.

If similarities between $TCFXS(X_p)$ and each existing cluster are less than the least similarity threshold, a new EHTCFXS only including X_p will be built. Online maintaining these EHTCFXSs needs memory, so the total number of clusters in a sliding window is limited to usable memory. When the total number of EHTCFXSs in a sliding window exceeds the allowed largest number NC, some rules are used to decrease the total number of EHTCFXSs in the sliding window. This paper proposes a simply and direct method to delete some EHTCFXSs according to the following rules. One rule is that the selected EHTCFXS should include the least number of documents; another is that the selected EHTCFXS must include a special TCFXS which is in 0-grade sub-streams and has the oldest time stamp. The first rule indicates the EHTCFXS is an isolated point in streams. The second rule denotes that the EHTCFXS doesn't be updated lately and be close to overdue clusters. The experiments in this paper use the second rule, reasons are as the following: (1) deleting isolated points only increases few memory; (2) mining isolated points is one of goals of clustering research; (3) the technology of sliding windows focuses on recent or current clusters, which just tallies with the goal of the second rule.

C. Algorithm and Time Complexity

Further to the above discussions, Algorithm 1 (called Clustering XML Data Streams by Structure based on Sliding Windows and Exponential Histograms, for short CXDSS-SWEH) outlines incremental updating progress of clustering results. The Step 1 directly calls sub-algorithm $CreateTCFXS(X_t, NameIndex)$ to build $TCFXS(X_t)$ for a newest arriving XML document X_t. Given the length of the existing global name-index is L_{max} and the total number of distinct elements in X_t is N_t, the time complexity of the sub-algorithm can be approximated as $O(L_{max} \times N_t)$ according to the definition and building method introduced in Section 4.1. The initialization of clustering (Steps 2 to 4) generates an EHTCFXS only including $TCFXS(X_t)$. Therefore, the overall time of Steps 2 to 4 is constant. The first loop of Steps 6 to 8 computes similarities between each

Algorithm 1 CXDSS-SWEH.

```
Input: $X_t$ is a XML document arriving on window at time t;
        $\omega$ is the least similarity threshold;
        NC is the largest number of EHTCFXSs included in window;
        N is the largest number of XML documents contained in window;
        NameIndex is a list of saving pairs of element name and number;
      H[k] is a group of EHTCFXSs of representing clustering results before time t.
Output: H[l] is a group of EHTCFXSs which represent clustering results at time t.

1: get TCFXS($X_t$) using sub-Algorithm CreateTCFXS($X_t$,NameIndex);
2: if (k ==0)
3:     { generate an EHTCFXS H[1] only containing TCFXS($X_t$) ;
4:       return H[1]; }
5: else{ $sim_{max}$=0; count=k;
6:         for(i=1; i<=k; i++)
7:             { computing similarity sim between $H_i$ and TCFXS($X_t$) ;
8:               if (sim> $sim_{max}$) { $sim_{max}$=sim; $H_{max}$=$H_i$;}}
9:           if (NodeSim(TCFXS($X_t$), $H_{max}$) >$\omega$)
10:              { add      ($X_t$) into the set of 0-grade sub-streams in $H_{max}$;
11:                num=the cardinal of the set of 0-grade sub-streams in $H_{max}$;
12:                i=1;
13:                 while (num =1 /$\varepsilon$+2)
14:                     {merge two oldest TCFXSs in the set of (i-1)-grade into $T_i$ ;
15:                      add $T_i$ into the set of i-grade sub-streams in $H_{max}$;
16:                      num=the cardinal of the set of i-grade sub-streams in $H_{max}$;
17:                      i=i++; }}
18:          else {generate an EHTCFXS $H_{k+1}$ only containing TCFXS($X_t$) ;
19:              count=k+1;
20:                 if (count> NC)
21:                     {delete the overdue EHTCFXS ;
22:                    count--; }}
23:        sum number of XML documents contained in H[count] as $D_{sum}$;
24:        if ($D_{sum}$> N)
25:           {get the EHTCFXS $H_{old}$ containing the oldest TCFXS;
26:            delete the oldest TCFXS;
27:            if ($H_{old}$ == null)
28:               {delete the $H_{old}$ from H[count] ; count--; } }
29:          return H[count] ;}
```

existing cluster and $TCFXS(X_t)$ and chooses the largest value Sim_{max} and the corresponding EHTCFXS H_{max}. Its time complexity can be involved in comparing progress of computing similarities in Section 4.2. Given the total number of elements in two Node Lists, respectively, is N_1 and N_2, the time of Steps 6 to 8 is the same as larger value in two values, that is $O(Max(N_1, N_2))$, in the worst state. Conditional statements of Steps 9 to 17 firstly compare Sim_{max} with the least similarity threshold, then decide whether building a new cluster (Steps 18 to 22) or inserting $TCFXS(X_t)$ into H_{max} (Step 10 to 17) according to the comparing result. Among, the work of Steps 20 to 22 adjusts the total number of clusters through comparing the total number of existing clusters with the largest number of clusters allowed in a sliding window. Step 21 takes $O(k)$ time to scan existing clusters and delete the cluster which is not updated newly, where k is the number of existing clusters. Step 25 takes constant time to acquire the total number of XML documents in the sliding window. Overdue TCFXSs and corresponding EHTCFXS are deleted (Steps 27 to 29), where the total number of documents exceeds threshold value N. Give a cluster including M TCFXSs, deleting overdue TCFXSs need to scan all TCFXSs. So the total time complexity of Steps 25 to 29 is $O(M)$. In summary, the total time complexity of Algorithm 1 normally can be approximated as $O(L_{max} \times N_t + max(N_1, N_2) + k + M) \approx O(L_{max} \times N_t)$, where $O(k), O(M), O(max(N_1, N_2)) \ll O(L_{max} \times N_t)$.

VI. Experimental Validations

A. Experimental Data Sets and Design

Our experiments involve two datasets, the XMLFiles real dataset used to evaluate cluster quality, and the XMLSimples simulated dataset used to measure memory and time consumption. The XMLFiles dataset contains 437 XML documents. The documents are from 23 various domains such as Movie (74), University (22), Automobile (189), Bibliography (16), Company (35), Hospitality message (25), Travel (10), Order (10), Auction data (4), Appointment (2), Document page (14), Bookstore (2), Play (20), Club (12), Medical (2), and Nutrition (1). The number of tags varies from 10 to 100 in these sources. The nesting level varies from 2 to 15. The XMLSimples simulated dataset randomly is created by an XML tool named oxygen based on some schemas of mature industries such as civil aviation and web application. There are 10419 documents in the dataset. Their size varies from 1k to hundreds of k.

The experiments are run in a PC with Pentium IV 2.4GHz and Windows XP. The static algorithm XCLS in the literature [8] and the stream-oriented clustering algorithm SW-XSCLS in the literature [11], as comparing methods, are implemented in same conditions and criterions. To eliminate influence of orders of XML documents and acquire more precise clustering results, we re-adjust orders of documents in two datasets through a hash algorithm, and simulate smooth XML data streams.

B. Experimental Results and Analysis

Figure 3 shows intra-cluster and inter-cluster similarities on XMLFiles dataset for three algorithms. Intra-cluster similarities of three methods across different number of documents are over 0.975, and inter-cluster similarities are less 0.07, as shown in Figure 3. In essence, the XCLS and the SW-XSCLS use same methods to compute similarities, so the change of clusters quality is similar. However, existing clusters in the SW-XSCLS can include fewer documents because of removing overdue documents in sliding windows. So the influence of Level Structures in the SW-XSCLS is less than that in the XCLS. Results of the SW-XSCLS are better than that of the XCLS. The intra-cluster similarity of the CXDSS-SWEH is less than others and the inter-cluster similarity of the CXDSS-SWEH is larger than them, as shown in Figure 3. The reason is that the equation of computing similarities in the XCLS and the SW-XSCLS is not symmetrical for some peculiar documents in real dataset XMLFiles. For example, two documents have common number of levels, but elements in their level 3 satisfy inclusion relation instead of equivalence relation. For these documents, similarities computed by the XCLS and the SW-XSCLS are 1, otherwise the similarity computed by the CXDSS-SWEH is less 1. In fact, their structure is not exactly same. So, the computing equation in the CXDSS-SWEH is more logical than that in the XCLS and the SW-XSCLS.

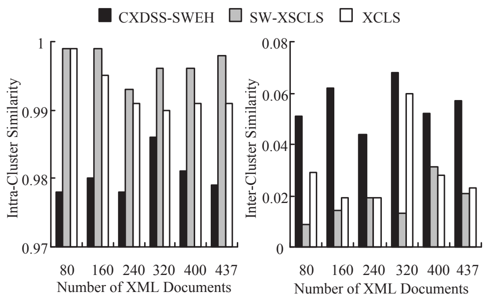


Figure 3. Quality comparison ($r = 2, \varepsilon = 2, \omega = 0.8, N = 100, NC = 50$)

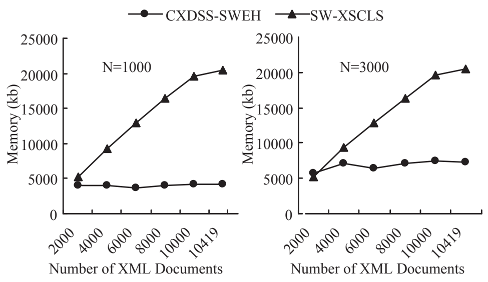


Figure 4. Memory consumed vs. number of XML documents ($r = 2, \varepsilon = 2, \omega = 0.8, NC = 130$)

Though the size of sliding windows is different, as shown in Figure 4-light and right, the memory consumption of the same algorithm has not increased evidently. But, the memory consumption of the CXDSS-SWEH is obviously less than that of the SW-XSCLS. The memory consumption often consists of two part. One part is used to maintaining clusters, and another is used to building structure synopses. For a smooth stream, the number of clusters in a sliding window generally fixed on at any time. A structure cluster with high intra-cluster similarities consumes fixed memory to save the DTD or Schema of all XML documents in the cluster. So, the memory consumption of maintaining clusters is steady. The SW-XSCLS consumed more memory than the CXDSS-SWEH in building structure synopses. As described in Section 4.1, the progress of parsing and building Node Lists uses runtime stacks to save memory. The progress of parsing and building Level Structures in SW-XSCLS is based on pruning DOM trees. When the structure of documents is complex, lots of memory is used to save trees.

Figure 5 shows a comparison of time cost on XMLSimples data with the SW-XSCLS and the CXDSS-SWEH. Though sizes of sliding windows are different, the trends of time change for two methods are similar and increase with the number of XML documents. As described in Section 5.3, the time complexity of the CXDSS-SWEH is proportion to the total number of distinct elements in sliding windows. When the number of documents increases, the total number of distinct elements generally can increase. So time cost also

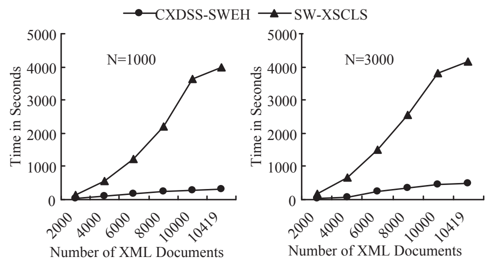


Figure 5. Time vs. number of XML documents ($r = 2, \varepsilon = 2, \omega = 0.8, NC = 130$)

increase with the number of documents. Structure synopses in the SW-XSCLS also are Level Structures. Time cost of maintaining these synopses in sliding windows increases with the number of documents, as described in the literature [8]. The time cost of the CXDSS-SWEH is less than that of the SW-XSCLS with the same number of documents and parameters user-defined, as presented in Figure 5. The equation of computing similarities in the XCLS is not transitive. The asymmetry would lead to twice matching progress for acquiring common elements. In the worst state, the time complexity of each matching progress is $O(m \times n)$: m and n, respectively, are the total number of elements in two Level Structures. And the time complexity of matching progress in the CXDSS-SWEH is $O(max\{N_1, N_2\})$: N_1 and N_2, respectively, are the number of elements in two Node Lists. From angle of time complexity, these steps just are the key steps in whole algorithm, and will take the most time. Hence, the time cost of the SW-XSCLS increases extremely large as the number of XML documents increases, whereas there is no significant difference in time cost of the CXDSS-SWEH.

VII. Conclusion And Future Work

To group an online XML data stream by structure, this paper introduces an algorithm, named CXDSS-SWEH, based on sliding windows and exponential histograms. The method parses XML documents through SAX, and then formalizes their structure into synopses named TCFXS. Each cluster in sliding windows consists of a team of TCFXSs, which satisfy criterions of false positive exponential histograms. To validate empirical effects, we have conducted a series of experiments involving real and simulative XML data. Our experimental results have confirmed: (1) clustering quality of the CXDSS-SWEH is close to the static clustering method XCLS and the stream clustering method SW-XSCLS; (2) memory and time consumption of the CXDSS-SWEH are efficient and effective, compared to the SW-XSCLS.

But, the existing cluster feature only considers structure, and omits all semantic information. In many fields, such as identifying online buyers, context in XML is more important. In future work, we will improve some context features.

Acknowledgment

The work is the National Natural Science Foundation of China (60873145), and the Doctor Start-up Program of Beijing University of Technology (X0007011200901).

References

[1] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, S. Microsystems, and F. Yergeau. *Extensible Markup Language (XML) 1.0 (Fourth Edition).* W3C Recommendation 16 August 2006, 2006.

[2] J. Chang, F. Cao, and A. Zhou. *Clustering evolving data streams over sliding windows.* Journal of Software, 18(4):905-918, 2007.

[3] T. Dalamagas, T. Cheng, K.-J. Winkel, and T. Sellis. *A methodology for clustering xml documents by structure.* Information Systems Journal, 31(3):187-228, 2006.

[4] S. Flesca1, G. Manco, E. Masciari, L. Pontieri, and A. Pugliese. *Detecting structural similarities between xml documents.* In Proceedings of the 5th International Workshop on the Web and Databases, WebDB, pages 55-60, June 2002.

[5] C. Koch and S. Scherzinger. *Attribute grammars for scalable query processing on xml streams.* The VLDB Journal, 16:317-342, 2007.

[6] W. Lian, D. W. lok Cheung, N. Mamoulis, and S.-M. Yiu. *An efficient and scalable algorithm for clustering xml documents by structure.* IEEE Transactions on Knowledge and Data Engineering, 16(1):82-96, 2004.

[7] V. Mayorga and N. Polyzotis. *Sketch-based summarization of ordered xml streams.* In IEEE 25th International Conference on Data Engineering, 2009, pages 541-552, April 2009.

[8] R. Nayak. *Fast and effiective clustering of xml data using structural information.* Knowl Inf Syst, 14:197-215, 2008.

[9] A. Nierman and H. V. Jagadish. *Evaluating structural similarity in xml documents.*In Proceedings of the 5th International Workshop on the Web and Databases, WebDB 2002, pages 61-66, June 2002.

[10] C. Yang, C. Liu, J. Li, J. X. Yu, and J. Wang. *Semantics based buffier reduction for queries over xml data streams.* In Nineteenth Australasian Database Conference,ADC2008, pages 145-153, January 2008.

[11] G. Mao, M. Gao, W. Yao. *An algorithm for clustering XML data stream using sliding window.* The Third International Conference on Advances in Databases, Knowledge, and Data Applications, pages 96-101, January 2011.

[12] S. Zheng, A. Zhou, and L. Zhang. *Similarity measure and structural index of xml documents.* Chinese Journal of Computers, 26(9):1116-1122, 2003.

[13] N. Bruno, L. Gravano, N. Koudas, and D. Srivastava. *Navigation- vs. index-based XML multi-query processing.* In Proceedings of ICDE, pages 139-150, 2003.

Feasibility of the Implementation of a UML to XML Evolution Architecture Using Eclipse as Technological Ecosystem

Beatriz Pérez, Ángel Luis Rubio, Gloria Yanguas
Departamento de Matemáticas y Computación
Universidad de La Rioja
La Rioja, Spain
beatriz.perez@unirioja.es, arubio@unirioja.es, gloria.yanguas@alum.unirioja.es

***Abstract*—Unified Modeling Language (UML) and eXtensible Markup Language (XML) are two of the most commonly used languages in software engineering processes. One of the most critical of these processes is that of model evolution and maintenance. More specifically, when an XML schema is modified, the changes should be propagated to the corresponding XML documents, which must conform to the new, modified schema. A current trend in this context consists of propagating the changes from the conceptual level (UML in our case) to the other levels (XML Schemas and documents). This paper is devoted to the study of the feasibility of the implementation of a UML to XML evolution architecture using the Eclipse framework, by means of UML2 and XSD plug-ins. A conclusion drawn from our study is that the chosen plug-ins lack of technological capabilities to implement this architecture.**

Keywords-evolution architecture; UML class model; XML; Eclipse plug-ins

I. INTRODUCTION

The modification of existing systems and models, in order to be adapted to requirement changes or technical advances, while maintaining the consistency between the generated artifacts, is one of the most important challenges in model-based software engineering processes nowadays [1][2].

From its origins, eXtensible Markup Language (XML) has constituted one of the most commonly used forms of representation of information covering data and metadata processing, management and retrieval. Additionally, in this context, Unified Modeling Language (UML) is widely used in the early phases (analysis, design) of development process [3][4], while the design of XML schemas are a consequence of the decisions made in those stages [5]. Different works have highlighted the importance of minimizing the effort of updating an XML document conforming to modified XML schemas [1][6]. For this reason, several authors propose to propagate the changes from the conceptual level (UML in our case) to the others levels (XML Schemas and documents) [2][5][7][8]. This approach freed analyst to make low-level implementation decisions.

In order to provide with a solution in this context, our research group undertook the search for an automated tool out of specific technological requirements. Particularly, this solution has been tackled in two different steps. Firstly, a generic architecture, named Generic Evolution Architecture (GEA), has been defined for managing those tasks when a model-driven development is followed [5]. Particularly, we focused on the transformation of UML class models to XML schema (due to the great majority of the papers that deal with this kind of transformation [9]) and provided an evolution framework by means of which the XML schema and documents are updated conforming to the changes in the UML class model.

Secondly, we need a specific implementation of such architecture in order to obtain, as a long-term goal for our approach, the development of a software tool which implements GEA. In this line, a laboratory prototype as a proof-of-concept was already developed; implementing a subset of the evolution transformations with a textual user interface, but such prototype is far away from being a complete solution. Due to the complexity of our architecture, it makes no sense to consider developing it from scratch. Particularly, we need to carry out a first task exploring partial solutions currently available in order to find one which allows us to implement our architecture. One of these possible solutions consists of using Eclipse as technological space, since it is considered to be the most versatile, plural and configurable open source Integrated Development Environment (IDE) tool.

Taking this into account, the paper aims at presenting the results obtained from the study and analysis of several existing Eclipse plug-ins used within the model-based development context, for the implementation of GEA. More specifically, a conclusion drawn from such study is that the chosen plug-ins lack of technological capabilities to implement our architecture. This fact has made us to consider new lines of research to follow-up, considering more complex tools within the own Eclipse ecosystem, which are explored in this paper.

The rest of the paper is structured as follows. The main features of GEA are presented in the following section. Section 3 is devoted to describe briefly the Eclipse plug-ins

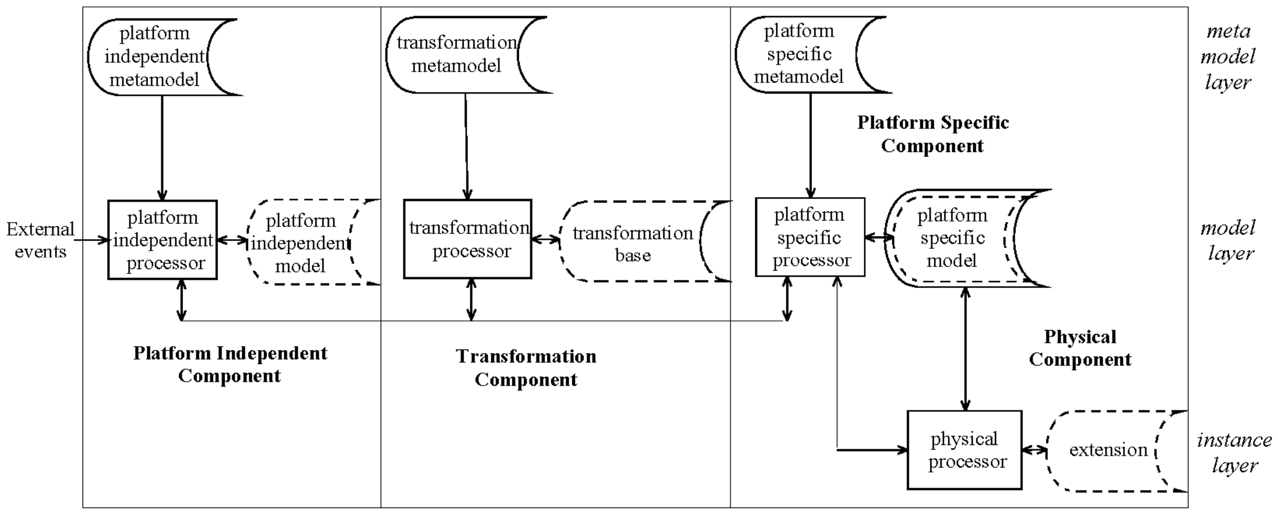


Figure 1. GEA: Generic Evolution Architecture (taken from [5]).

that have been used (UML2 and XSD). In Section 4, the proofs performed using the plug-ins, geared towards the development of GEA are explained in detail. Finally, conclusions and further work are presented in Section 5.

II. Generic Evolution Architecture

GEA, standing for Generic Evolution Architecture, is a generalization of a metamodel-based database evolution architecture called MeDEA and presented in [10]. As reflected in [5] "GEA keeps the characteristics of MeDEA stated in [10] and at the same time fits into a wider application context".

The main features of GEA, all of them deeply explained in [5], are enumerated in the remainder of this section.

(1) It follows the *Model Driven Architecture* (MDA) approach.
As it can be seen in Figure 1, GEA is structured around two dimensions. Vertically the different artifacts are divided into three abstraction levels which correspond with the M0, M1 and M2 layers of the 4-layer metamodel architecture pattern. Horizontally, the artifacts are identified with the developments phases established by MDA [11]. For the case of UML-XML, three specific metamodels (the Stereotyped UML Class metamodel, the UML-to-XML Transformation metamodel and the XML Schema metamodel) were proposed in [5] showing its graphical representation.

(2) The *transformation component* stores the links between the different elements of the platform independent component and the related elements of the platform specific component. It ensures the traceability of the transformation process.

(3) The *extension* to the Physical Component, propagating the evolution process from the platform specific model to the instances is another feature. Within the XML context, the XML documents are modified conforming to the evolved XML schema.

(4) *Evolution* is supported by the previous three features. Transformation and evolution process always start at the Platform Independent Component.

III. Eclipse

Eclipse [12] is an open source software project, which provides a highly integrated tool platform. One of the main characteristics of Eclipse is its extensibility, since it allows the user to develop plug–ins which are integrated into the core, defining a particular IDE. Eclipse has been described as "an IDE for anything, and nothing in particular [13]."

As described previously, the goal of this paper is to study the feasibility of using different plug-ins to implement the architecture explained in the previous section. Taking this into account, we have based on one of the top level projects, the Modeling Project [14], which mainly focuses on the evolution and promotion of model-based development technologies within the Eclipse community by providing a unified set of modeling frameworks, tooling, and standards implementations. More specifically, we have based on two of its subprojects: Eclipse Modeling Framework (EMF) and Model Development Tools (MDT).

On the one hand, the *EMF project* [15][16] is presented as a modeling framework and code generation facility for building tools and other applications based on a structured data model. EMF allows the user to define a model in any of three forms, Java Interfaces, UML diagrams or XML Schema, and later, generate the other forms from it, including even the corresponding implementation classes.

The purpose of the *MDT project* [17], on the other hand, is to provide an implementation of industry standard metamodels as well as tools for developing models based on those metamodels. For its interest in our work, two subprojects within this project have been considered: UML2 and XSD. *UML2* [18] is an EMF-based implementation of the UML 2.x metamodel for the Eclipse platform. Besides providing a usable implementation of the UML metamodel, it also includes a common XMI schema to facilitate interchange of models, test cases and validation rules. *XSD* [19] is a library that provides an Application Programming

Interface (API) for creating and manipulating W3C XML Schema and XML documents as well as an API for keeping documents conforming to their schemas as these are modified.

IV. TOWARDS AN IMPLEMENTATION OF GENERIC EVOLUTION ARCHITECTURE

This section is devoted to describe the most relevant aspects of the technological approach that have been undertaken to implement GEA. First, Eclipse Indigo (v 3.7.1) and the plug-ins UML2 Extender SDK (v 3.2.1) and XSD - XML Schema Definition SDK (v 2.7.1) have been installed in order to perform this implementation. Besides, the examples followed in [16] and commonly used in papers that deal with UML and XML [7][20] have been used as a benchmark (for instance, Simple Purchase Order, the Primer Purchase Order –PPO- and Extended Purchase Order –ExtendedPO-).

Before going into detail on the parallelism between these proofs and the architecture shown in Section 2, let us define some concepts related to EMF. "An *EMF model* is essentially the Class Diagram subset of UML [16]." "The model used to represent models in EMF is called Ecore. Ecore is itself an EMF model, and thus is its own metamodel [16]." Due to lack of space the graphical representation of the metamodels used in this section are not showed. Anyway, a comprehensive explanation on the Ecore metamodel can be found in [16], and it is stored in the file Ecore.ecore (in turn, contained in the file org.eclipse.emf.ecore_2.7.0.v20120127-1122.jar).
Essentially, the Ecore metamodel defines four types of objects:

EClass is used to represent a modeled class. It is identified by name and can have a number of attributes and references. A class can refer to a number of other classes as its supertypes.

EAttribute models attributes. It is identified by name and has a type.

EDataType models the type of an attribute. It is used to represent simple types whose details are not modeled as classes.

EReference is used to represent one end of an association between classes. It has a name, a boolean flag to indicate if it represents containment, a lower and upper bounds to specify multiplicity and a reference (target) type, which is an EClass. Besides, related classes and data types are grouped in *EPackage*, which is the root element of a serialized Ecore model.

In our implementation, both *the platform independent metamodel* and *the platform specific metamodel* are Ecore models. The first (uml.ecore) is included in the UML2 plug-in (org.eclipse.uml2.uml_3.2.100.v201108110105.jar). It consists of an `EPackage`, 247 `EClass`, 13 `EEnum` (which is a subclass of EDataType and it is used to model enumerated types) and 4 primitive types. On the other hand, the models generated into the Platform Specific Component will be conformed to the metamodel xsd.ecore provided in the XSD plug-in, within org.eclipse.xsd_2.7.1.v20120130-0943.jar. It is simpler than the previous metamodel, and it consists of an `EPackage`, 57 `EClass`, 20 `EEnum` and 5 primitive types.

The UML class model is transformed to an EMF model and afterwards the EMF model is transformed to an XML schema. The Eclipse framework defines and has total control of these *transformations*. In particular, the UML2 project defines a mapping from UML 2.0 to Ecore. A similar mapping, except subtle details, is described for UML version 1.4 in [16]. This mapping only concerns with the constructs of UML classes. Broadly speaking, a Package maps to an `EPackage`; a class is mapped to an `EClass`, `EEnum`, or an `EDataType`, depending on the class's stereotype; an attribute maps to an `EAttribute` and an operation maps to an `EOperation`, which models the behavioral features of an EClass. It is worth noting that an UML association maps to two `EReferences` and each of them has the other as its `eOpposite`. Taking these results into account, we have demonstrated that is not possible to map an association class. Furthermore, we have also proved that the UML model and its EMF counterpart are not automatically synchronized.

Detailed information about how the second transformation, from EMF model to XML schema, is performed can be obtained from [16]. At a high level, the mapping is as follows: an `EPackage` maps to a schema, an `Eclass` maps to a complex type, an `EDataType` maps to a simple type, an `EAttribute` and an `EReference` map to an attribute or element declaration.

With respect to the *instance layer*, the XSD plug-in provides the tools for creating XML documents from an XML schema and for validating them if the XML schema changes.

Taking this into account, we can conclude that the analysis and tests carried out on these tools confirm us that their expected capacities seem to be far away from those really supported, at least regarding models synchronization. Regarding *evolution*, although intuition points out to get similar results, we plan to follow a source-like approach, that is, propagating the changes from the UML class model (so this work must be considered 'in progress').

V. CONCLUSION AND FUTURE WORK

In this paper, the possibility of implementing a UML to XML Evolution Architecture by means of three of the plug-ins that seem to be the most appropriate ones (EMF, UML2 and XSD) have been studied.

These plug-ins have been tested founding several difficulties described below. To transform a UML class model to XML Schema is required an intermediate transformation to an EMF model. Besides, to update the EMF model conforming to the changes in the UML model is a very complex task for which it is necessary to be a specialized expert in adapters and notifiers.

It is worth noting that EMF only concerns itself with a small subset of UML. For instance, a UML class model that contains a class association cannot be mapped to an EMF model. Therefore, this process is valid only for specific UML class models. Finally, we want to note that all the transformations are automatically carried out; to create and manage our own transformation rules is not possible.

For all these reasons, in despite of these plug-ins provide us with a lot of structures and procedures, we conclude that the development of our evolution architecture using only and directly these plug-ins is an exceedingly complex task, and it may not even be feasible.

There exist several possibilities for follow-up this implementation:

(1) To use the Ecore metamodel as the Platform Independent Metamodel. In this case, we give up the richness of UML since Ecore is a small subset of UML. We would like to advance that we have already obtained some preliminary results in this line, which lead us to think that our impressions about using this metamodel to implement our GEA architecture are justified.

(2) To create our own transformations by means of Atlas Transformation Language (ATL) and MofScript, both of them subprojects of the Eclipse Modeling Project. Thereby we would have under control the transformations of each element. Regarding this line of work, we have experience in using both tools (ATL and Mofscript) for the particular case of implementing a framework that automatically generates decision support systems for clinical guidelines [21]. This experience makes us to think that they are feasible solutions for our implementation problem, but we are aware that it is required a conceptual task to align our transformation approach with these tools.

(3) To explore the Hypermodel plug-in [22], in order to know if it could be used to implement our architecture. Hypermodel was designed and implemented by David Carlson and it is stated that generates XML schemas from any UML model. This fact leads us to think that this plug-in could be a good reference among other existing tools.

ACKNOWLEDGMENT

This work has been partially supported by Spanish Ministry of Science and Innovation project TIN2009-13584 and University of La Rioja, project API12/12.

The authors would like to thank the others authors of [5] for their contributions: Eladio Domínguez, Jorge Lloret, Áurea Rodríguez, María A. Zapata.

REFERENCES

[1] G. Guerrini, M. Mesiti and D. Rossi, "Impact of xml schema evolution on valid documents", in: WIDM '05: Proceedings of the 7th annual ACM international workshop on Web information and data management, ACM, NY, USA, 2005, pp. 39-44. http://doi.acm.org/10.1145/1097047.1097056.

[2] M. Klettke, "Conceptual XML schema evolution - The CoDEX approach for design and redesign", in M. Jarke, T. Seidl, C. Quix, D. Kensche, S. Conrad, E. Rahm, R. Klamma, H. Kosch, M. Granitzer, S. Apel, M. Rosenmüller, G. Saake, O. Spinczyk (Eds.), Workshop Proceedings Datenbanksysteme in Business, Technologie und Web (BTW 2007), Aachen, Germany, 2007, pp. 53-63.

[3] T. Krumbein and T. Kudrass, "Rule-based generation of XML schemas from UML class diagrams", in: Proceedings of the XML Days at Berlin, Workshop on Web Databases (WebDB), 2003, pp. 213-227.

[4] I. Kurtev, K. V. Berg and M. Aksit, "UML to XML-schema transformation: a case study in managing alternative model transformations in MDA", in: Proceedings of the Forum on specification and Design Languages (FDL'03), European Electronic Chips & Systems Design Initiative, Frankfurt, Germany, 2003.

[5] E. Domínguez, J. Lloret, B. Pérez, Á. Rodríguez, Á. L. Rubio and M. A. Zapata, "Evolution of XML schemas and documents from stereotyped UML class models: A traceable approach", Information and Software Technology. Vol. 53, no. 1, pp. 34-50, 2011.

[6] D. K. Kramer, "XEM: XML evolution management", Ph.D. thesis, Worcester Polytechnic Institute (2001).

[7] N. Routledge, L. Bird and A. Goodchild, "UML and XML schema", in: X. Zhou (Ed.), Thirteenth Australasian Database Conference (ADC2002), ACS, Melbourne, Australia, 2002.

[8] R. Elmasri, Q. Li, J. Fu, Y.-C. Wu, B. Hojabri and S. Ande, "Conceptual modeling for customized XML schemas", Data Knowl. Eng. 54 (1) (2005) 57-76.

[9] E. Domínguez, J. Lloret, B. Pérez, A. Rodríguez, A. L. Rubio and M. A. Zapata, "A survey of UML models to XML schemas transformations", in: Proceedings of the Web Information Systems Engineering (WISE) Conference, Vol. 4831 of Lecture Notes in Computer Science, Springer, 2007, pp. 184-195.

[10] E. Domínguez, J. Lloret, A. L. Rubio and M. A. Zapata, "MEDEA: A database evolution architecture with traceability", Data and Knowledge Engineering 65 (3) (2008) 419-441. doi:10.1016/j.datak.2007.12.001.

[11] J. Mukerji and J. Miller, "MDA guide version 1.0.1", available at http://www.omg.org/cgi-bin/doc?omg/03-06-01/ 19.11.2012 (June 2003).

[12] The Eclipse Foundation, "Eclipse - The Eclipse foundation open source community website", available at http://www.eclipse.org 19.11.2012

[13] Object Technology International, Inc., "Eclipse platform technical overview", available at http://www.eclipse.org/whitepapers/eclipse-overview.pdf 19.11.2012 (February 2003).

[14] The Eclipse Foundation, "Eclipse modeling project", available at http://www.eclipse.org/modeling/ 19.11.2012

[15] The Eclipse Foundation, "EMF", available at http://www.eclipse.org/projects/project.php?id=modeling.emf.emf 19.11.2012

[16] D. Steinberg, F. Budinsky, M. Paternostro and E. Merks, EMF: Eclipse modeling framework, Addison-Wesley, 2008.

[17] The Eclipse Foundation, "Eclipse modeling - MDT", available at http://www.eclipse.org/modeling/mdt/ 19.11.2012

[18] The Eclipse Foundation, "MDT-UML2", available at http://www.eclipse.org/modeling/mdt/?project=uml2 19.11.2012

[19] The Eclipse Foundation, "MDT-XSD", available at http://www.eclipse.org/projects/project.php?id=modeling.mdt.xsd 19.11.2012

[20] W3C, "W3C XML schema definition language XSD", available at http://www.w3.org/TR/2012/REC-xmlschema11-1-20120405/ 19.11.2012 (April 2012).

[21] E. Domínguez, B. Perez and MA Zapata. Towards a traceable clinical guidelines application. Methods Inf Med 2010; 49: 571–580.

[22] D. A. Carlson, "hyperModel|XMLmodeling.com", Available at http://xmlmodeling.com/hypermodel 19.11.2012 (2012).

Capturing the History and Change Structure of Evolving Data

George Papastefanatos, Yannis Stavrakas, Theodora Galani
IMIS, RC ATHENA
Athens, Greece
{gpapas,yannis,theodora}@imis.athena-innovation.gr

Abstract—**Evo-graph is a model for data evolution that encompasses multiple versions of data and treats changes as first-class citizens. A change in evo-graph can be compound, comprising disparate changes, and is associated with the data items it affects. In previous papers, we have shown that recording data evolution with evo-graph is very useful in cases where the provenance of the data needs to be traced, and past states of data need to be re-assessed. We have specified how an evo-graph can be reduced to the snapshot holding under a specified time instance, we have given an XML representation of evo-graph called evoXML, and we have presented how interesting queries can be answered. In this paper, we explain how evo-graph is used to record the history of data and the structure of changes step by step, as the current snapshot evolves. We present C2D, a novel framework that implements the concepts in the paper using XML technologies. Finally, we experimentally evaluate C2D for space and time efficiency and discuss the results.**

Keywords-data evolution; change modeling

I. Introduction and Preliminaries

Data published on the Web undergo frequent changes due to advancements in knowledge and due to the cooperative manner of their curation. Users of scientific data, in particular, would like to go beyond revisiting past data snapshots, and review how and why the recorded data have evolved, in order to re-evaluate and compare previous and current conclusions. Such an activity may require a search that moves backwards and forwards in time, spread across disparate parts of a database, and perform complex queries on the semantics of the changes that modified the data. The need for accounting for past changes and tracing data lineage is evident not only in scientific data, but also in a wide range of web information management domains.

Motivating Example. We will use an example taken from Biology: the revision in the classification of diabetes, which was caused by a better understanding of insulin [12]. Initially, diabetes was classified according to the age of the patient, as *juvenile* or *adult onset*. As the role of insulin became clearer two more subcategories were added: *insulin dependent* and *non-insulin dependent*. All *juvenile* cases of diabetes are *insulin dependent*, while *adult onset* may be either *insulin dependent* or *non-insulin dependent*. In Fig. 1, the leftmost image depicts a tree representation of the initial diabetes classification, while the rightmost the revised classification. These two representations, however, do not provide any information about which parts of the data evolved and how, which changes led from one version to another, or what changes were applied on which parts of the data. Recording change operations in a log or discovering deltas out of successive versions, like many systems do, do not solve the problem; in most cases isolated operations are impossible to interpret a posteriori. This is because they usually form more complex, semantically coherent changes, each comprising many small changes on disparate parts of the data.

We argue that in systems where evolution issues are paramount, changes should not be treated solely as transformation operations on the data, but rather as *first class citizens* retaining structural, semantic, and temporal characteristics. In previous work, we proposed a graph model, *evo-graph* [16], and its XML representation, *evoXML* [17], capturing the relationship between evolving data and changes applied on them. A key characteristic is that it explicitly models changes as first class citizens and thus, enables querying data and changes in a uniform way. In what follows, we discuss some preliminary concepts on evo-graph and then present the contribution and structure of this paper.

Snap-graph. We assume that data is represented by a rooted, node-labeled, leaf-valued graph called *snap-graph*. A snap-graph S (V, E) consists of a set of nodes V, divided into complex and atomic, with atomic nodes being the leaves of the graph, and a set of directed edges E. At any time instance, the snap-graph undergoes arbitrary changes.

Evo-graph. An *evo-graph* G is a graph-based model that captures all the instances of an evolving snap-graph across time, together with the actual change operations responsible for the transitions. It consists of the following components:

- *Data nodes*, divided into *complex* and *atomic*: $V_D = V_D^c \cup V_D^a$.
- *Data edges* depart from every complex data node, $E_D \subseteq (V_D^c \times V_D)$.
- *Change nodes* are nodes that represent change events. Change nodes are depicted as triangles, to distinguish from circular data nodes. They are divided into *complex* and *atomic* (denoting basic change operations): $V_C = V_C^c \cup V_C^a$.
- *Change edges* connect every complex change node to the (complex or atomic) change nodes it encompasses: $E_C \subseteq (V_C^c \times V_C)$.
- *Evolution edges* are edges that connect each change node with two data nodes, specifically the version before and after the change: $E_E \subseteq (V_D \times V_C \times V_D)$.

Intuitively, the evo-graph consists of two interconnected graphs: a data graph comprising the different versions of

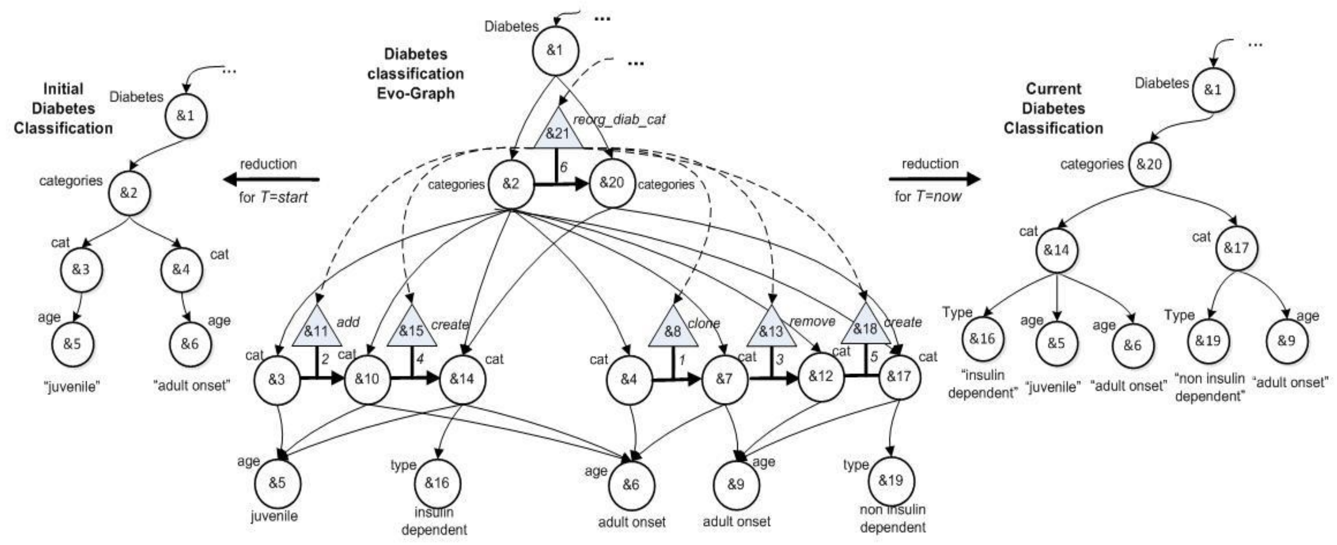


Figure 1. Snap Graphs of diabetes classification before (left) and after (right) revision and the corresponding evo-graph (middle).

data, and a tree of changes. The data graph defines the structure of data, while the change graph defines the structure of changes. These two graphs interconnect via evolution edges. Consequently, there are two roots: the data root, r_D, and the change root, r_C. Moreover, we annotate change nodes with a timestamp denoting the time instance that the specific change occurred. These timestamps are used for determining the validity timespan of all data nodes and data edges in the evo-graph. Evo-graph can be reduced to a snap-graph holding under a specified time instance through the *reduction* process [16]. A snap-graph is actually a trivial case of an evo-graph, consisting of a set of data nodes $V \subseteq V_D$ and a set of data edges $E \subseteq E_D$.

As an evo-graph example consider the middle image in Fig. 1, which represents the revision in the diabetes classification from the graph of Fig. 1 left to the graph of Fig. 1 right. The revision process is denoted by the complex change *reorg_diab_cat*, (node &21) composed by 5 basic snap changes (in the order they occurred): *clone* (node &8), *add* (node &11), *remove* (node &13), *create* (node &15), and *create* (node &18). Note the use of evolution edges; in the case of *add* the evolution edge is annotated with the timestamp 2 and connects node &3 (initial version) with node &10 (version after adding the child node &6). Node &10 is still a child of node &2, but for simplicity the relevant edge is omitted. The reduction of the evo-graph for T=*start* results in the snap-graph of the leftmost image of Fig. 1, while for T=*now* in the snap-graph of the rightmost image of Fig. 1. For the complete definitions of basic snap changes see section 2.1.

EvoXML. In [17] we have shown how evo-graph can be represented in an XML format, called *evoXML*. TABLE I. presents an evoXML example. Due to space limitations, the evoXML example covers up to time instance 1 of the evo-graph in Fig. 1; specifically it includes only the *clone* operation (node &8) in lines 12-15, 20. Notice that the edge from node &7 to node &6 (which actually denotes that &6 remains a child of the next version of node &4) is represented through the evoXML reference *evo:ref* in line 13, which points to the element in line 10. Also notice how the change node &8 is represented in line 20.

Querying Evolution. Finally, in [16],[17] we have outlined *evo-path*, an XPath extension that help us posing regular queries over data snapshots as well as time- and change-aware queries on evo-graph. We have also shown how evo-path expressions can be evaluated on evoXML via equivalent XQuery expressions. Evo-path takes advantage of the complex change information in order to retrieve and relate data that are otherwise distant and irrelevant to each other. Queries expressed on evo-graph include:

- Temporal queries on the history of data nodes, like "which is the structure of categories before the time instance 6"?
  `Evo-path: //Diabetes/categories [ts() not covers {now}]`
- Evolution queries on changes applied to data nodes, like "which changes are associated with the change responsible for the reorganization of diabetes categories" (node &21)?
  `Evo-path: <//reorg_diab_cat/*>`
- Causality queries on relationships between change nodes and data nodes, like "what are the previous versions of all data nodes that changed due to the reorganization of diabetes categories"?
  `Evo-path: //* [evo-before() <//reorg_diab_cat>]`

Contribution and Structure. In this paper, we first define a set of *basic changes* on the snap-graph, and how these can be combined to construct *complex changes* (section 2). We then define a *set of basic operations on the evo-graph, and a translation from snap-graph changes to evo-graph operations*, such that as changes occur on the snap-graph, the evo-graph grows to represent those changes together with all the successive snap-graph versions (section 2). Furthermore, we introduce the *C2D framework* (section 3), a prototype system that implements the concepts introduced in this paper, and progressively builds the evo-graph as changes take place on the current snap-graph. We present

TABLE I. EVOXML FOR TIME INSTANCE 1.

```
<evo:evoXML xmlns=""
 xmlns:evo="http://web.imis.athena-innovation.gr/projects/c2d">
  <evo:DataRoot evo:id="dataroot">
    <Diabetes evo:id="1">
      <categories evo:id="2">
        <cat evo:id="3">
          <age evo:id="5">juvenile</age>
        </cat>
        <cat evo:id="4">
          <age evo:id="6">adult onset</age>
        </cat>
        <cat evo:id="7" evo:ts="1" evo:previous="4">
          <age evo:ref="6"/>
          <age evo:id="9">adult onset</age>
        </cat>
      </categories>
    </Diabetes>
  </evo:DataRoot>
  <evo:ChangeRoot evo:id="changeroot">
    <clone evo:id="8" evo:tt="1" evo:before="4" evo:after="7"/>
  </evo:ChangeRoot>
</evo:evoXML >
```

and discuss a detailed *experimental evaluation* of C2D (section 3). Finally, we review the related work (section 4) and we conclude the paper (section 5).

II. ACCOMMODATING BASIC AND COMPLEX CHANGES IN EVO-GRAPH

A. Snap Basic and Complex Change Operations

In this section, we define the basic change operations applied on a snap-graph S(V,E) (*snap changes* for short) and present how they can be employed to define complex changes. We consider the following snap changes:

- *create*(v^P, v, *label*, *value*). Creates a new atomic node v with a given *label* and *value* and connects it with its parent node v^P. If v^P is an atomic node, it becomes complex.
- *add*(v^P, v). Adds the edge (v^P, v) to E, effectively adding v as a child node of v^P. The nodes v^P, v must already exist in V. If v^P is an atomic node, it becomes complex.
- *remove*(v^P, v). Removes the edge (v^P, v) from E. If v has no other incoming edges, it is removed from V. If v^P has no other children, it becomes an atomic node with the default value (empty string).
- *update*(v, *newValue*). Updates the value of an atomic node v to *newValue*.
- *clone*(v^P, v^{source}, v^{clone}). Creates a new data node v^{clone} with the same label/value as v^{source}, and a deep copy of the subtree under v^{source} as a subtree under the node v^{clone}. The node v^P must be a parent of v^{source}. The edge (v^P, v^{clone}) is added to E, making v^{clone} a sibling of v^{source}.

The above definitions describe the effect of each snap change to the current snap-graph. These changes leave the snap-graph in any possible consistent state. Note that the effect of the *clone* snap-change is to create a deep copy of a subtree under the same parent node. Although *clone* can be expressed as a sequence of other snap changes, we chose to consider it as a basic operation. The reason is that deep copy is difficult to express using successive *create* operations, while at the same time it is an essential operation for expressing complex changes like *move-to*, and *copy-to*.

A *complex change* applied on a node of the snap-graph is a sequence of basic and other complex change operations that are applied on the node itself or/and the node's descendants, and allows us to group operations in semantically coherent sequences. Applying a complex change on a snap-graph involves the application of each constituent change in the order they appear. Consider the complex change *reorg_diab_cat* applied on *categories* node of the leftmost image of Fig. 1. This change is expressed as a sequence of five basic snap changes, as follows:

```
reorg-diab-cat (&2) {
   clone (&4, &6, &9)
   add (&3, &6)
   remove (&4, &6)
   create (&3, &16, "type", "insulin dependent")
   create (&4, &19, "type", "non insulin dependent") }
```

B. Capturing Versions and Changes with Evo-graph

In our approach, snap changes are not actually applied on the snap-graph, but on the evo-graph. This is shown in Fig. 2, which illustrates the effects of snap changes to the evo-graph. Fig. 2 depicts three images for each snap change; the leftmost image shows the initial snap-graph before the change, the rightmost image shows the current snap-graph after the snap change, and the middle image shows the evo-graph fragment encompassing both snapshots, together with the change. Notice that these snap-graph fragments are actually *reductions* [16] of the respective evo-graph under different time instances. Thus, the *create* operation in Fig. 2 actually causes node &4 to be added under the parent node &5, and not under &2, as would be the case if *create* was applied directly on the snap graph. This is a technical issue tackled with at the implementation level, and does not introduce any ambiguities.

In order to implement snap changes on an evo-graph G we introduce the following evo-graph operations:

- *addDataNode* (v_D^P, v_D, *label*, *value*). Creates a new atomic data node v_D as a child of v_D^P with a given *label* and a *value*. If v_D^P is an atomic node, it turns into complex.
- *addDataEdge* (v_D^P, v_D). Creates a new data edge from node v_D^P (parent) towards node v_D (child). The two nodes must already exist in V_D. If v_D^P is an atomic node, it turns into complex.
- *applyAtomicChange*(v_D^1, v_D^2, *value*, v_C, v_C^P, *label*, *timestamp*). This operation "evolves" node v_D^1 to node v_D^2, as the result of applying a snap change. First, a new atomic data node v_D^2 with the same label as v_D^1 and a given value is created, and is connected as a child of all the current parents of v_D^1. Then, a new atomic change node v_C with the *label* and *timestamp* is created, and is connected as a child of node $v_C^P \epsilon V_C^c$. The *label* denotes one of the snap changes defined previously. Finally, a new evolution edge e = (v_D^1, v_C, v_D^2) is created between the data nodes v_D^1, v_D^2 and the change node v_C.

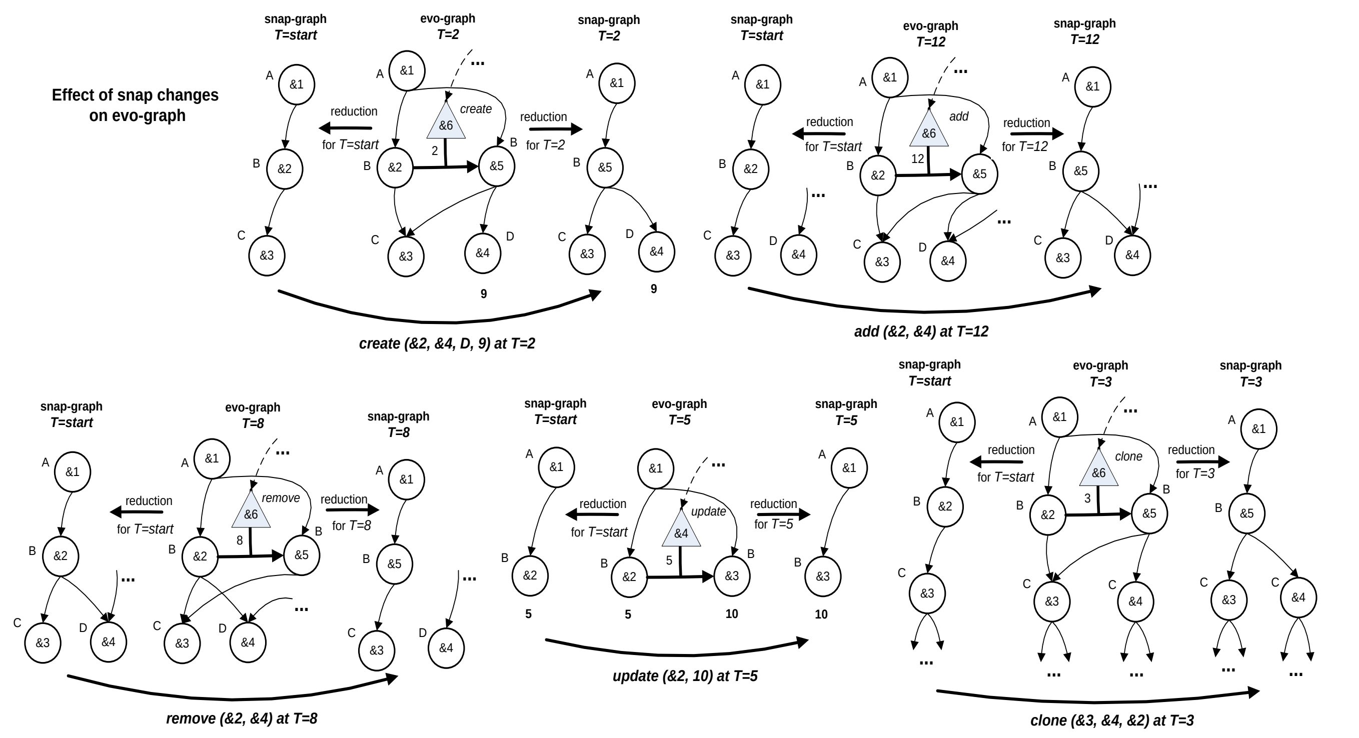


Figure 2. Effect of snap change operations on the evo-graph.

- *applyComplexChange*(v_D^1, v_D^2, v_C, v_C^p, *label*, *timestamp*, {v_C^1, v_C^2, ..., v_C^n}). This operation "evolves" node v_D^1 to node v_D^2, as the result of applying a complex change operation on the snap-graph. First, a new atomic data node v_D^2 with the same label as $v_D^1$and the default value (empty string) is created, and is connected as a child of all the current parents of v_D^1. A new *complex* change node v_C with the *label* and *timestamp* is created, and is connected as a child of the complex change node $v_C^p \epsilon V_C^c$. The *label* is the name of the complex change and can be any string. After that, v_C is connected as a parent of the change nodes {v_C^1, v_C^2, ..., v_C^n}. Finally, a new evolution edge $e=(v_D^1, v_C, v_D^2)$ is created between the data nodes v_D^1, v_D^2 and the change node v_C.

Note that we employ two separate evo-graph operations for applying snap-graph basic and complex changes. For complex changes, the *applyComplexChange* is used, which creates a new *complex* change node, a new version for the affected data node, a new evolution edge connecting the change node and the two data node versions and finally connects the complex change node as the parent of its constituent change nodes. For basic changes, the *applyAtomicChange* is used, which creates a new *atomic* change node, a new version of the data node that is affected by the change, and a new evolution edge. The exact implementation of each snap change in terms of evo-graph operations is given in TABLE II. .

For each snap change in TABLE II. , a *timestamp* is given (appears as *t*) and, if this change is part of a complex change, the parent complex change (v_C^P) is also specified. If no parent complex change is specified, we assume the parent is the change root r_C. Note, that all snap change implementations in TABLE II. start with *applyAtomicChange*, which creates the corresponding change node and the associated data node in evo-graph.

TABLE II. ACCOMMODATING SNAP CHANGES IN EVO-GRAPH.

	***create** (v_D^P, v_D, label, value), t, v_C^P*
1	{ applyAtomicChange(v_D^P, v'^P_D, '',v_C, v_C^P, 'create', t);
2	for $v_i \in$getCurrentChildren(v_D^P)
3	addDataEdge (v'^P_D,v_i);
4	*// create the new data node* and *connect it to the new parent node*
5	addDataNode *(v'^P_D, v_D, label, value);* }
	***add** (v_D^P, v_D), t, v_C^P*
1	{ applyAtomicChange(v_D^P, v'^P_D, '',v_C, v_C^P, 'add', t);
2	*//connect the new parent node to all current children plus v_D*
3	for $v_i \in$(getCurrentChildren(v_D^P)$\cup v_D$)
4	addDataEdge (v'^P_D,v_i) ; }
	***remove** (v_D^P, v_D), t, v_C^P*
1	{ applyAtomicChange(v_D^P, v'^P_D, '',v_C, v_C^P, 'remove', t);
2	*//connect the new parent node to all current children except for v_D*
3	for $v_i \in$(getCurrentChildren (v_D^P)-v_D)
4	addDataEdge (v'^P_D,v_i); }
	***update** (v_D, newValue), t, v_C^P*
1	{ applyAtomicChange(v_D, v'_D, newValue,v_C, v_C^P, 'update', t) }
	***clone** (v_D^P, v_D^{source}, v_D^{clone}), t, v_C^P*
1	{ applyAtomicChange(v_D^P, v'^P_D, '',v_C, v_C^P, 'clone', t);
2	for $v_i \in$(getCurrentChildren (v_D^P)
3	addDataEdge (v'^P_D,v_i);
4	*//clone the source data node*
5	addDataNode *(v'^P_D, v_D^{clone}, v_D^{source}.label, v_D^{source}.value);*
6	*//create a deep copy of the cloned node*
7	for $v_i \in$getCurrentChildren (v_D^{source})
8	addDataNode(v_D^{clone}, v'_i, , v_i.label, v_i.value);
9	*repeat step 7 for v_D^{source} = v_i and v_D^{clone}*=v'_i }

III. Implementation and Evaluation

A. The C2D Framework

We have implemented all above concepts into the C2D (standing for Complex Changes in Data evolution) framework. C2D has been developed in Java, on top of Berkeley DB XML [3], an embedded XML database used to manage the evoXML representation of evo-graphs. In C2D, changes applied on the snap-graph are fed into a process that populates the evo-graph. A snap change is always applied on the current snap-graph (represented in XML in C2D), which is actually produced as a reduction [16] of the evo-graph for the time instance T=*now*. This flow is depicted in Fig. 3. The top layer in Fig. 3 is the *view layer*, where changes are launched. The purpose of the *logical model* layer is to guide the translation processes between the view layer and the *storage representation layer*, where changes actually take place.

Change operations on the evo-graph are implemented as XML update operations on the corresponding evoXML. Expressing evo-graph operations with the XQuery Update language is straightforward. For example the *addDataNode (&17, &19, "type", "non insulin dependent")* operation is expressed with the following XQuery Update *insert* expression on the evoXML.

```
insert node <type evo:id="19">non insulin  dependent </type>
into
/evo:evoXML/evo:DataRoot/Diabetes/categories/cat[evo:id="17"]
```

B. Experimental setting

Our goal was to examine how our approach depends on a number of factors that characterize the data. We first examined the space efficiency of evoXML for various configurations, regarding: the structure of the initial XML tree, the type of snap changes, and the selectivity of the elements. We also examined the performance of the reduction process with respect to the size of the evoXML file. Note that the comparison with other versioning approaches [4], [6], [7] was not pursued, as these works do not consider the role of changes as first class citizens in storing and querying evolving data.

Experiments were performed over synthetic XML data, on a PC with Intel Core 2 CPU 2.26 GHz, and 4.00 GB of RAM. The initial XML file was generated with [19] and contained about 10^5 elements, over which 10^4 snap changes were sequentially applied as XQuery Update statements. A new version was assumed after every 1000 changes; therefore 10 successive versions have been created for each setting. We recorded the size (in terms of the number of XML elements) of each "snap" version, and the size of the evoXML file at the same instance. Furthermore, we examined the performance of the reduction process for the current snapshot (T=*now*), and the initial snapshot (T=*start*).

Regarding the structure of the initial data, we used two XML files with the same number of elements: (a) one corresponding to a snap-graph with a "deep" tree structure (denoted s_1) with five levels and elements having a fan-out of 10, and (b) a file with a "broad" tree structure (denoted s_2) with only two levels and elements with a fan-out of about 330 elements. We have applied three sets of snap changes: (a) equal percentage for all changes except *clone* (denoted t_1), (b) 80% *update* and 20% *create* and *remove* (denoted t_2), and (c) equal percentage for all changes including *clone* (denoted t_3). Finally, concerning elements selectivity, changes have been applied either on all elements (denoted n_1) or on a fixed set of pre-selected elements so that each element is affected by 5 changes on average per version (denoted n_2).

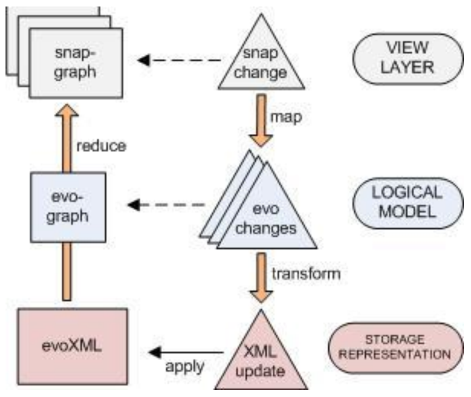


Figure 3. C2D framework overview.

We have examined the following combinations of the above parameters: (t_1n_1), (t_3n_1), (t_2n_1), and (t_2n_2) for each of s_1, s_2. t_1n_1 captures the typical case when random changes are uniformly applied on all elements. t_3n_1 is similar to t_1n_1, but it also includes *clone*. We have separately examined the *clone* operation, as it may arbitrarily result in the addition of a large amount of data. t_2n_1 captures the case where most (80%) change operations are *update* on random leaf elements, and only 20% are create or *remove*. Finally, t_2n_2 is like the previous case except that changes are concentrated on a pre-selected fixed set of elements.

Intuitively, we expect that the size of the evoXML depends on the number of snap changes performed. We also expect that it depends on the average fan-out of the snap-graph, while it remains insensitive to its average height. This is due to the way that each snap change operation is implemented on the evo-graph. Next, we present and discuss the results.

C. Results and Discussion

In Fig. 4 (a) and (b) we present the evoXML sizes per version. Subsequently, we discuss how this size is affected by the aforementioned configurations parameters.

File structure. For all configurations, better space efficiency is achieved for s_1. For smaller fan-outs (s_1), the evoXML has a smoother increase in size than for large fan-outs (s_2). A snap change occurring on an element adds *evo:ref* elements for all of its children (i.e. fan-out) that are still valid in the new version. Hence, the increase in the evoXML size is relative to the average fan-out.

Type of changes. t_2 outperforms t_1 and t_3. The majority of changes in t_2 are *update*, which have a smaller impact on the evoXML size. Again, the key point is the number of new elements that each change adds. Observe from TABLE II. that all changes add at least two new elements;

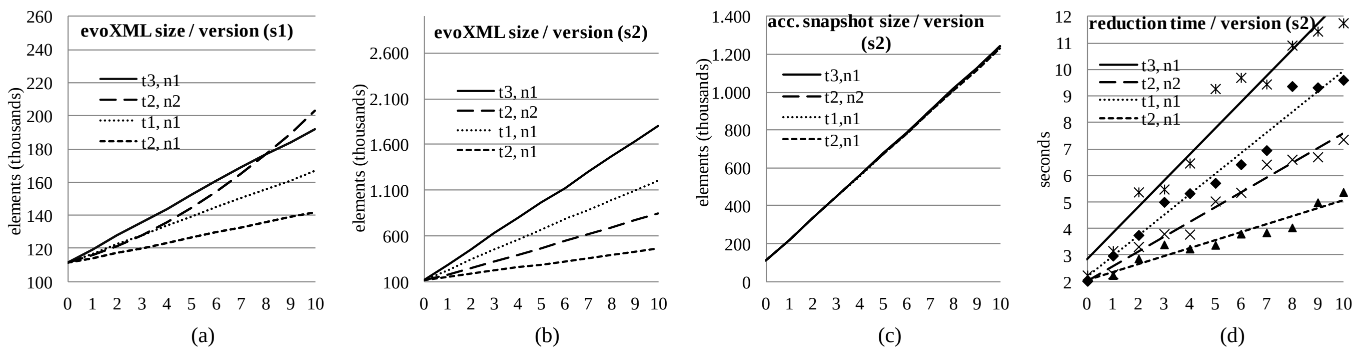


Figure 4. evoXML size (a), (b), accumulative snapshot size (c) and current snapshot reduction time (d) per version for various configurations.

one evolved data element and one change element. *update* adds only these two elements, whereas *create* and *add* insert one additional element for the new child, plus *evo:ref* elements for its siblings. *remove* results in inserting *evo:ref* elements in the evoXML for all the siblings of the removed element. Finally, *clone* adds a variable number of elements according to the height and average fan-out of the subtree that is cloned. On the other hand, the percentage of *create* and *remove* in t_1 is higher. In t_3, the use of *clone* further increases the file size by creating a deep copy of the subtree of the elements on which it is applied.

*Selectivity of elements.*Applying changes randomly on all elements (n_1) seems to have a smoother impact on the increase of the file size (e.g., compare t_2n_1 and t_2n_2 for each of s_1, s_2). This is due to the fact that changes are uniformly distributed over all the elements. On the other hand, the increase is higher when changes are targeting a fixed set of elements (n_2). Changes in t_2n_2 are sequentially applied on the same elements, i.e., *create* is applied on the same elements, increasing the number of their children and thus the number of *evo:ref* elements to be inserted when a subsequent *create* occurs on the same element.

Overall, the evoXML size depends almost linearly on the number of the snap changes applied, given that the average fan-out is constant. Moreover, the increase rate of the evoXML size is proportional to the average fan-out of its elements. This is more evident in t_2n_2 for s_1, where the average fan-out of the elements sustaining changes increases significantly per version, resulting in a boost in the evoXML size, whereas in s_2 the fan out increase rate is much smoother.

In Fig. 4 (c) we present the accumulative size of the snapshots produced per version. This approach can be considered as an alternative to evoXML. For space reasons, we only depict the series for s_2, as s_1 shows a similar trend. The accumulative size of all snapshots per version is significantly bigger than the evoXML size, for all runs over s_1. The same holds for all configurations of s_2, except for t_3n_1 where many *evo:ref* elements are added in the evoXML file. Note that the overlap of the series is due to the small variance in the accumulative snapshot size between configurations.

Regarding the performance of our reduction algorithm, we have measured the time the reduction process takes for producing the current and the initial snapshots. The results for the current snapshot for s_2 are shown in Fig. 4 (d), where the mark signs are the recorded time values, and the series are the trends for each configuration. A safe conclusion is that the reduction time depends mostly on the evoXML size. For small file sizes, the reduction performs the same for all versions. In addition, the increase rates in time are similar for both the current and the initial snapshot, for both s_1 and s_2. Therefore, the time instance parameter of the reduction process does not affect the reduction performance.

Concluding, both space and time efficiency are mostly affected by the average fan-out, which deteriorates as more changes are applied. That is mainly because of the evo:ref elements that are added for all children of an element that “evolves”. Still, our approach is much more efficient than retaining separately every different version. Future optimizations will take into consideration the above and will aim to encode evo:ref elements and to the overall compression of the file.

IV. Related Work

Numerous approaches have been proposed for the management of evolving semistructured data. One of the early works [6] proposes DOEM, an extension of OEM capable of representing changes, such as *Create Node*, *Add Arc*, *Remove Arc* and *Update Node*, as annotations on the nodes and the edges of the OEM graph. In [10], the authors employ a *diff* algorithm for detecting changes between two versions of an XML document and storing them either as edit scripts or deltas. For each new version, they calculate the deltas with the previous and retain only the last version and the sequence of deltas. A similar approach is introduced in [7], where instead of deltas calculation, a referenced-based identification of each object is used across different versions. New versions hold only the elements that are different from the previous version whereas a reference is used for pointing to the unchanged elements of past versions. In [9] the authors propose MXML, an extension of XML that uses context information to express time and models multifaceted documents. Recently, there are works that deal with change modeling [15] and detection [13] in semantic data, in which the aforementioned problems are applied to ontologies and RDF.

Most approaches employ temporal extensions for the lifespan of different versions of documents. In [1], [6], the authors enrich data elements with temporal attributes and extend query syntax with conditions on the time validity of the data. In [14], the authors model an XML document as a directed graph, and attach transaction time information at the edges of the graph. Techniques for evaluating temporal queries on semistructured data are presented in [8], [18]. In [8] the authors propose a temporal query language for adding valid time support in XQuery. In [18] the notion of a temporally grouped data model is employed for uniformly representing and querying successive versions of a document. In [11], the authors extend this technique for publishing the history of a relational database in XML and employ a set of schema modification operators (SMOs) for representing the mappings between successive schema versions. In [1] the problem of archiving curated databases is addressed. The authors develop an archiving technique for scientific data that uses timestamps for each version, whereas all versions are merged into one hierarchy. This is in contrast with approaches that store a sequence of deltas and apply a large number of deltas for retrieving backwards the history of an element. Lastly, [5] deals with provenance in curated databases. All user actions for constructing a target database are recorded as sequences of insert, delete, copy and paste operations stored as provenance links from current data towards previous versions of the target database or external source databases.

Compared to the above approaches, our model introduces a change-based perspective for evolving data, in which changes are not derived by data versions but are modeled as first class citizens together with data. In our view, changes are not described through diffs or transformations with edit scripts between document versions, but are complex objects operating on data, and exhibit structural, semantic, and temporal properties. Change-centric modeling of evolving semistructured data can provide additional information about *what*, *why*, and *how* data has evolved over time.

V. CONCLUSIONS

In this paper, we showed how a data model called evo-graph can be used to progressively capture the structure of changes and the history of data. We believe that capturing structured changes within a data model enables a range of very useful queries on the provenance of data, and on the semantics of data evolution. We defined basic and complex changes over snap-graph, and described the process of building evo-graph step by step, as changes occur on the current snap-graph. We outlined C2D, a framework based on XML technologies that implements the ideas presented in this paper. We evaluated C2D using synthetic XML data for its space and time efficiency, and discussed the results.

ACKNOWLEDGMENT

This research has been co-financed by the European Union (European Social Fund - ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Thales. Investing in knowledge society through the European Social Fund.

REFERENCES

[1] T. Amagasa, M. Yoshikawa, S. Uemura, “A Data Model for Temporal XML Documents”, In DEXA 2000.

[2] P. Amornsinlaphachai , N. Rossiter and M. A. Ali, “Translating XML Update Language into SQL”, Journal of Computing and Information Technology, 2006, 2, 91–110.

[3] Berkeley DB XML. http://www.oracle.com/technetwork/database/berkeleydb/overview/index.html. 19 June 2012.

[4] P. Buneman, S. Khanna, K. Tajima, W.C. Tan, ”Archiving Scientific Data”, ACM Transactions on Database Systems, Vol. 20, pp 1-39, 2004.

[5] P. Buneman, A. P. Chapman, J. Cheney, “Provenance Management in Curated Databases”, In SIGMOD’06.

[6] S. Chawathe, S. Abiteboul, J. Widom, “Managing Historical Semistructured Data”, Journal of Theory and Practice of Object Systems, Vol. 24(4), pp.1-20, 1999.

[7] S-Y. Chien, V. J. Tsotras, C. Zaniolo, “Efficient Management of Multiversion Documents by Object Referencing”, In VLDB 2001.

[8] D. Gao, R. T. Snodgrass, “Temporal Slicing in the Evaluation of XML Queries”, In VLDB 2003.

[9] M. Gergatsoulis, Y. Stavrakas, “Representing Changes in XML Documents using Dimensions”, In 1st International XML Database Symposium, (XSym 2003).

[10] A. Marian, S. Abiteboul, G. Cobena, L. Mignet, “Change-Centric Management of Versions in an XML Warehouse”, In VLDB 2001.

[11] H.J. Moon, C. Curino, A. Deutsch, C.Y. Hou, C. Zaniolo, “Managing and querying transaction-time databases under schema evolution”, In VLDB 2008.

[12] National research council - Committee on Frontiers at the Interface of Computing and Biology. Catalyzing Inquiry at the Interface of Computing and Biology. Edited by J. C. Wooley, H. S. Lin., National Academies Press, 2005.

[13] V. Papavassiliou, G. Flouris, I. Fundulaki, D. Kotzinos, V. Christophides, “On Detecting High-Level Changes in RDF/S KBs”, In ISWC 2009.

[14] F. Rizzolo, A. A. Vaisman, “Temporal XML: modeling, indexing, and query processing”, In VLDB J. 17(5): 1179-1212 (2008).

[15] F. Rizzolo, Y. Velegrakis, J. Mylopoulos, S. Bykau, “Modeling Concept Evolution: a Historical Perspective”, In ER 2009.

[16] Y. Stavrakas, G. Papastefanatos, “Supporting Complex Changes in Evolving Interrelated Web Databanks”, In In CoopIS 2010.

[17] Y. Stavrakas, G. Papastefanatos, “Using Structured Changes for Elucidating Data Evolution”, In DaLi’11 (with ICDE 2011).

[18] F. Wang, C. Zaniolo, “Temporal Queries in XML Document Archives and Web Warehouses”, In TIME 2003.

[19] Xmlgener: Synthetic XML data generator. http://code.google.com/p/xmlgener/.

[20] XQuery Update Facility 1.0. http://www.w3.org/TR/xquery-update-10/, W3C Recommendation, 17 March 2011.

Business Intelligence in an Educational Landscape

Marcio Rodrigo Teixeira
Instituto Superior Tupy
Educational Society of Santa Catarina (SOCIESC)
Joinville, Brazil
e-mail: marcio.teixeira@sociesc.org.br

Mehran Misaghi
Instituto Superior Tupy
Educational Society of Santa Catarina (SOCIESC)
Joinville, Brazil
e-mail: mehran@sociesc.org.br

***Abstract*—This paper reports a work in progress about the effectiveness of Business Intelligence in the academic environment. Business management, competitiveness, excess of information and dispersion of data in the organizational environment have proven to be important barriers that need to be overcome by managers today, both in process improvement and decision making. This paper presents an initial study on this topic and shows the benefits of Business Intelligence application to assist private educational institutions to achieve better academic results in their whole process of management. Our preliminary results indicate that BI is a great differential to promote a successful management.**

Keywords-academic environment; business intelligence; data warehouse; knowledge

I. Introduction

With the development of management models and society, information and knowledge have to be considered the main assets of an organization, contributing to its success and differentiation. However, the over-dispersion of data and information end up becoming barriers that need to be overcome in order to improve processes and decision making.

We can say that control and management of information play an important role in strategic planning and decision making in all sectors of society, including education.

The fast transformation of the contemporary world has given organizations, including educational ones, a great challenge to overcome. How to get passed them and monitor changes effectively, seeking to maintain competitive advantages are the issues of management today.

According to Tachizawa and Andrade [1], a new era in terms of competition is emerging, not only from known competitors in traditional markets (or other organizations that enter in certain economic sectors), but also through the disintegration of access barriers to markets previously isolated and protected.

Educational institutions cannot feel excessively confident by market slices and competitive positions already conquered, instead, they should seek to reduce operational costs, increase the profit margin and improve the quality of services provided.

Many industries felt the shock of competition and survived; now, the time has come to teach these institutions. They can either adhere to a more contemporary and responsive business model or accept that in the next years, many private higher education institutions will be sold or close doors [2].

In this context, it is necessary to use technological information in order to accelerate the competitive intelligence in private educational institutions.

Data analysis is very important for an efficient management. Through Business Intelligence [13] it is possible to take action and make more assertive decisions. While someone deals with management information, others can detect changes in the market. This way, it is possible not only to obtain advantages but also to achieve goals successfully.

This paper relates directly to this topic: knowledge discovery and intelligent knowledge querying. It reports a work in progress on business intelligence, its architecture and components, and finally, the benefits of its application applied in an educational landscape.

II. Analytical Intelligence and Business Intelligence

Analytical intelligence consists in the use of data and systemic reasoning in the process of decision making [3].

According to Pinheiro [4], analytical intelligence constitutes the use of knowledge through practical applications and markets that can generate competitiveness to a company.

With the capacity expansion of data storage and computerization of processes, the volume of data available in organizations is increasing; however, these data contribute very little to decision-making. It is necessary to transform them to be used strategically, in a way that they can interact in the process of making decisions according to the needs of the institution. In this context, analytical intelligence is essential.

This information allows a company to recognize their strengths and weaknesses, making actions more substantial and efficient [4].

A. *History and Concept of BI*

In the early 1980s, the concept of executive information systems (EIS) appeared, increasing the computerized support to managers and top-level executives. Some of the introduced features were dynamic multidimensional reporting (ad hoc or on demand), predictions and forecasts, trend analysis, details, status and access to critical successful factors. These features appeared in dozens of commercial products in the middle of the 1990s. The same features and some new ones appeared under the name BI.

According to Power [5], in 1989, Howard Dresner proposed "business intelligence" as an umbrella term to describe concepts and methods to improve business decision making by using fact-based support systems.

According to Atre and Moss [6], BI is neither a product, nor a system. It is an architecture and a collection of integrated operational programs as well as decision-support applications and databases that provide to the business community an easy access to business data.

According to Kimball and Ross [7], Business Intelligence is a term that has emerged and evolved over the past few years and is now often used to describe all systems and processes an enterprise uses to gather, process, provide access to, and analyze business information. The term data warehouse is now used to mean the platform for all forms of business intelligence.

Business refers to a commercial activity for profit. Intelligence is intelligence, refers to the ability of learning and understanding.

The process of BI is based on transforming data into information, decisions, and then, finally, into action.

B. *Results of Surveys about IT and BI*

2.335 chief information officers (CIO) responded to the Gartner 2012 CIO Survey, the annual appraisal of CIO priorities, including all major industries and geographies [8]. The survey result has identified that analytics and BI will be the top technology priorities for CIOs this year.

According to Goasduff and Pettey [9], analytics/business intelligence was the top-ranked technology for 2012, as CIOs are combining analytics with other technologies to create new capabilities. For example, analytics plus supply chain for process management and improvement.

The research results reveal that Business Intelligence is an essential tool for businesses because it allows making the best choices while minimizing the risks of a wrong decision.

C. *Architecture and Components of the BI*

Figure 1 shows the flow of data, their transformation into information, and finally, knowledge that serves as basis for better decision making. Although this figure shows an architecture with very high level (because it has all the components of a BI tool), it can be used either as a model for a new initiative (according to need of the institution), or as a mean of assessing the current status of an existing architecture in relation to all the components shown in the figure, thus enabling those new components to be added over time.

According to Dresner [10], BI tools and technologies include query and reporting, OLAP (online analytical processing), data mining and advanced analytics, end-user tools for ad hoc query and analysis, enterprise-class query, analysis and reporting, including dashboards for performance monitoring, and production reporting against enterprise data sources.

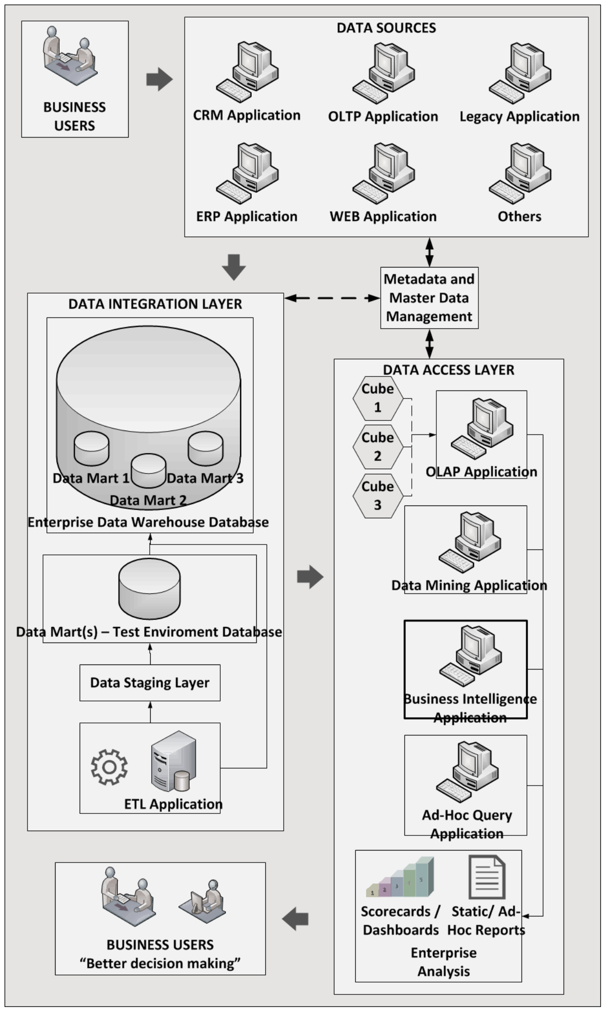


Figure 1. BI Architecture

A BI architecture typically contains the following components:

- People (Business Users, Administrators, Developers, Technical Support): People are fundamental in Business Intelligence architecture, as well as the processes and technologies involved. The community of BI users is diverse, but the vast majority focuses on the tactical and strategic levels and their relation to each other.
- Metadata and Master Data Management: Metadata can be defined as "data about data." The metadata is stored and managed in a database and is used to describe the definition of the structure, policies and data management of a company. The objective of Master Data Management is to provide processes to collect, aggregate, combine and consolidate data, ensuring quality to distribute these data across the organization to ensure consistency and control maintenance and use of this information.
- Data Sources: The data may have several shapes and be obtained from various sources, such as CRM and

ERP. These many data sources make up the landscape of business intelligence company.

- Data Integration Layer: This is where the data gathered by the applications is refined into a corporate structure [11]. Through components such as Data Warehouse, Data Mart, ETL, Ad-Hoc Query Application, it is possible to transform data into quality information that can be useful for any company, especially in decision-making process.
- Data Access Layer: The components as Cubes, Data Mining, OLAP application, BI application allow performing business analysis more efficiently and effectively and are the basis for better decision making.

III. Business Intelligence Applied in an Private Academic Enviroment Management

Private educational institutions are making extensive use of business intelligence and predictive modeling in marketing, recruitment and retention [12]. To achieve these goals, they develop the IT infrastructure with new software and process, taking into account "operational efficiency" and "customer intimacy". They understand the needs of students and employers and adapt the curriculum and course offerings in order to promote an alignment with the requirements of the job. Many educational institutions are also deploying academic analysis to create predictive models to improve the retention of students in order to develop an individual learning plan [12]. The assertiveness level in decision making, planning and resource allocation and process improvement have been observed. Some institutions have developed specific analysis to accompany new students in courses, mainly targeting disadvantaged students, and the financial aspect of performance; therefore, making possible to intervene early if a negative factor is identified, thereby aiming to act as a responsible institution that cares about its students. Analysis focused on issues related to access, learning and students performance at all stages of their academic life, allows them to take greater responsibility for their success, in collaboration with parents, teachers and employers. It is noteworthy that the application of Business Intelligence effectively requires a cultural change where decision making and action are based on evidence. This cultural change in an educational institution requires the support of managers in activities, emphasizing performance, creating incentives to support innovation, promoting a change in the traditional academic culture at all levels of the institution [12].

IV. Our Research

Aiming to verify the effectiveness of a Business Intelligence tool deployed in an educational landscape, we started a research project in 2012 with completion scheduled for 2013 in an educational institution in Brazil. The scope of the research and a summary of the information collected to date are described below.

A. *The Educational Institution*

The Educational Society of Santa Catarina - SOCIESC is an educational, cultural and technological institution in Brazil that has existed for 53 years. It possesses 7 campus in two states, and has more than 100 partnerships in E-learning's Centers. It currently has about 20,000 students enrolled in classroom, 4,000 students enrolled in e-learning mode and 1,200 employees that support this whole structure. It operates in various levels as undergraduate, graduate, business training. In addition, a differential of SOCIESC over other educational institutions is that it offers engineering services, consultancy and management for the development of new technologies for national and international companies. Services are offered from modern infrastructure with laboratories in the areas of metrology, chemical and mechanical, through the areas of Technology Management & Research and Development, Tooling, Foundry and Heat Treatment. SOCIESC is certified NBR ISO9001.

B. *Research Question*

What are the benefits of using a BI tool in an educational institution?

C. *Proposal of Research*

Analysis the contribution of the use of Business Intelligence Tool in management processes and decision making: A Case study in an educational institution (SOCIESC).

D. *Methodology*

- Interview with employees of IT department in SOCIESC who are responsible for the project and support of BI in the institution.
- Analysis of the data collected.

E. *Study Sample*

3 interviews were conducted in the IT department, including one manager and two system analysts who use the BI tool.

F. *The BI Tool*

The SOCIESC has currently the TOTVS BI Version: 1.11.30. Some features of this tool are:

- Integration with LOGIX ERP [14], currently used in SOCIESC.
- Online information access, using different browsers.
- Enables integration with various data sources, such as Oracle database [15] and Microsoft Excel spreadsheets [16].
- Developed in Java [17] and runs through a server Apache Jakarta [18].
- Operates on Linux [19] and Microsoft OS [20].
- Simple and quick search of information and export data to spreadsheets, analysis customization and graphical views, schedule and automatic send of information via e-mail and security access through system access profiles and analysis access.

G. The BI Architecture

The current BI architecture comprises a data warehouse created in Oracle 10G, which is loaded with data extracted from tables of academic ERP (WAE) and business ERP (Logix) that are also on Oracle 10G database.

H. User Profiles and Departments that use BI

Currently 84 employees distributed in 46 different departments of SOCIESC make use of BI. The audience of BI is: 14 directors representing 17% of audience, 23 managers representing 27% of audience, 38 analysts representing 45% of audience and 9 technicians representing 11% of audience.

I. History of BI in SOCIESC

According to the interviews, the initiative to use a BI tool in SOCIESC started in 2007 by the IT department that developed some analysis in order to leverage and maximize their management decision making. Once the results were positive, the tool was adopted by other departments of the institution contemplating the administrative area and also teaching. Over time, new versions were released by TOTVS, making it more agile and with new features which contributed to a rapid deployment and user training.

J. Analysis Developed in BI

The BI tool of SOCIESC currently contains about 22 analysis developed over the years which help users make better decisions about trade issues, costs, purchasing, human resources, information technology and education. When it comes to numbers, just education alone possesses 10 different types of analysis that allow monitoring of entries in selection processes, enrollment, financial monitoring organization, institutional research conducted by students and teachers.

V. Next Steps of Research

The next steps of the research are (i) conduct more interviews and apply questionnaires with users of other departments that use the BI tool, (ii) analyze the data collected, (iii) describe the analysis of BI, (iv) and describe a case study pointing contributions of BI tool in the management of processes and decision making in an educational institution.

VI. Conclusion

Aiming to verify the effectiveness of a BI tool deployed in an educational landscape, we started a research project in 2012 with completion scheduled for 2013 in an educational institution (SOCIESC) in Brazil. So far, we have collected various information through interviews about the profile of institution, the BI tool, the architecture of BI, the profile of users and departments that use the tool, history of BI in the institution, and analysis developed.

Our work in progress has demonstrated so far that the current global competitive environment has also impacted the educational sector and that educational institutions are seeking to adjust its strategic. Several educational institutions are looking for the potential of BI, and, therefore, are adopting BI architectures, applying mainly in predictive modeling in marketing, recruitment and retention of students. The architecture is accessible through the Internet, which provides a greater analytical power, allowing users (parents, students, teachers and analysts) to access and analyze data, information, contextualize and interpret the results and then make the best decisions.

References

[1] T. Tachizawa and R. Andrade, “Gestão de instituições de ensino,” Brazil, Editora FGV, pp. 21-23, 2006.

[2] L. Machado, “Gestão estratégica para instituições de ensino superior privadas,” Brazil , Editora FGV, pp. 15-17, 2008.

[3] T. Davenport, et al., “Inteligência analítica nos negócios: Como usar a análise de informações para obter resultados superiores,” Brazil, Elsevier, pp. 1-3, 2010.

[4] C. Pinheiro, “Inteligência Analítica – Mineração de Dados e Descoberta de Conhecimento,” Brazil, Editora Ciência Moderna ltd., Brazil, pp. 3-10, 2008.

[5] D.J Power, “A Brief History of Decision Support Systems,” 2007. http://dssresources.com/history/dsshistory.html [retrieved: October, 2012]

[6] S. Atre and L. T. Moss, “Business Intelligence Roadmap: the complete project lifecycle for decision-support applications.” Pearson Education, Boston, pp. 4-5, 2008.

[7] R. Kimball and M. Ross, “The Kimball Group Reader; Relentlessly Practical Tools for Data Warehousing and Business Intelligence.” Wiley Publishing, Indiana, pp. 103-105, 2010.

[8] Gartner, “CIO Agenda Report - Key Insights,” 2012. http://imagesrv.gartner.com/cio/pdf/cio_agenda_insights.pdf [retrieved: October, 2012]

[9] L. Goasduff and C. Pettey, “Gartner Executive Programs Worldwide Survey of More Than 2,300 CIOs Shows Flat IT Budgets in 2012, but IT Organizations Must Deliver on Multiple Priorities,” 2012. http://www.gartner.com/it/page.jsp?id=1897514 [retrieved: October, 2012]

[10] H. Dresner, “The Performance Management Revolution: business results through insight and action. John Wiley & Sons,” New Jersey, pp. 12-14, 2008.

[11] W. H. Inmon, C. Imhoff and R. Sousa, “Corporate Information Factory.” John Wiley & Sons, Canada, pp. 7-9, 2001.

[12] D. Norris, L. Baer, J. Leonard, L. Pugliese and P. Lefrere, “Action Analytics: Measuring and Improving Performance That Matters,” 2008. http://net.educause.edu/ir/library/pdf/ERM0813.pdf [retrieved: October, 2012]

[13] M. R. Teixeira and M. Misaghi, “Business Intelligence Aplicado a Gestão de um Ambiente Acadêmico Privado,” 2012. http://www.sociesc.org.br/congressos/SearchCONEPRO/artigos/064_2012_Publicacao.pdf [retrieved: October, 2012]

[14] http://www.totvs.com/software/erp [retrieved: December, 2012]

[15] http://www.oracle.com [retrieved: December, 2012]

[16] http://office.microsoft.com/en-us/excel/ [retrieved: December, 2012]

[17] http://www.oracle.com/us/technologies/java/overview/index.html [retrieved: December, 2012]

[18] http://jakarta.apache.org [retrieved: December, 2012]

[19] http://www.linux.org [retrieved: December, 2012]

[20] http://windows.microsoft.com [retrieved: December, 2012]

BA: Lean Manufacturing Indicators Applied to HR Applications - An Implementation Study

Robson Thanael Poffo
Innovation
TOTVS
Joinville, Brazil
robson.poffo@totvs.com.br

Mehran Misaghi
Researcher
SOCIESC
Joinville, Brazil
mehran@sociesc.org.br

***Abstract*—In the last few years, the world witnessed a large number of changes on the way to manage the production on factories. This changes were motivated by a financial crisis. With each passing year the companies feel the need to be faster when making decisions and one way to do this is through indicators (KPIs and KRIs). Remember that the difference between a company that grows its market share and the other companies is how fast they make decisions. Based on the mission, vision and values of the company, this article describes indicators that are used on lean manufacturing and that can be used in human resource management. Furthermore, it describes the best way to design the data warehouse considering the level of granularity and the level of data detail. First we will analyze the lean manufacturing indicators addressed on the academic literature. Then, we will analyze these indicators with the other indicators used by leading research companies in Brazil. After that, we will analyze the mission, vision and values of the company to determine the KRIs and KPIs that we need to monitor. Thereafter, we will design the data warehouse to comport these indicators considering the level of granularity and the level of data to analyze them. Finally, we will apply these indicators and data warehouse in a software development company to see the results. At the end of this project, we would like to get the list of indicators that would be able to manage and analyze the production and performance of the employees in the factory. With these indicators, the factory will be able to manage the team correctly despite the weather of global finance. This approach allows us to determine the indicators that will make a difference on the management of the company analyzing the performance of human resources associated with the performance of the manufacture. These indicators will determine the way to implement BI solutions associated with human resources systems and manufacturing systems successfully.**

***Keywords*-*BA; Lean Manufacturing; Indicators; HR Systems*.**

I. Introduction

The study of lean manufacturing indicators applied to people management with the purpose of reducing waste involves many crucial concepts for the success of the project. For this study, it is important to know the main concepts of lean manufacturing, the main characteristics of people management, the main characteristics of the indicators and how to apply the management of indicators to people management. Finally, it is also important to acknowledge the main characteristics of how to model a data warehouse in order to use these indicator correctly.

Aiming to determine the main characteristics of lean manufacturing, Pettersen [1] developed a research that contained a total of 38 bibliographical references. After finishing his research, Pettersen [1] identified only two concepts of lean manufacturing equally described in all references:

1) Reduction of the preparation time;
2) Continuous improvement.

With the result obtained by Pettersen it is possible to see not only how vast is the understanding of the concept of lean manufacturing is but also which concepts are more important. Carreira [2] highlighted that one of the main concepts of lean manufacturing is the continuous elimination of waste. For Hibbs, Jewett and Sullivan [3], one of the most important proposals of lean manufacturing is to deliver the final product to the client as fast as possible.

Jekiel [4] points out that the application of the concepts of lean manufacturing for people management had a limited outcome in many companies because they were not ready to provide the necessary support; in other words, applying the concepts of lean manufacturing to people management is not such a simple process and it needs monitoring to achieve success.

Jekiel [4] recommends using the concepts of lean manufacturing on people management in order to implement the culture of continuous improvement with the purpose of waste reduction. Jekiel also highlights the five main causes that make it difficult for companies to implement these concepts:

1) Job positions limit people in their projects: the definition of very specific roles and the attribution of many processes end up limiting knowledge and people's experiences.
2) Power is limited to a select group of people: the problem of power limitation is related to leadership, in many cases, leaders believe to have more knowledge than the rest of the group.

3) People do what they are asked of: Many people at work do only what they are paid for.
4) To channel abilities creates a new job: the problem related to channeling abilities relies in the absence of an administration to manage the result of these abilities.
5) There are no costs for people that work in inferior capacities: The majority of assets inside a company are measured according to their capacity, such as constructions or equipments, however, when it comes to the percentage of production of an employee, abilities that will be able to optimize the process are rarely considered.

The reasons identified by Jekiel [4] have something in common: they can all be monitored through indicators. However, the question now is, how to correctly choose the right indicators for this task.

There are three ways to measure development, according to Parmenter [5]:

1) Key Result Indicators (KRIs): Inform the results obtained in one perspective;
2) Performance Indicators (PIs): Inform what to do;
3) Key Performance Indicators (KPIs): Inform what to do in order to drastically increase the performance.

Usually, the KRIs are reviewed in monthly or six-monthly cycles, not daily or weekly, like it happens with KPIs indicators.

The KPIs represent a series of measures to demonstrate the most critical organizational development to the actual and future success of the organization.

Besides understanding the concepts of KPIs and KRIs, it is also important to understand the basic concepts related to the indicators of development, according to Bancaleiro [6]:

1) Effort: Energy level and human creativity invested on a task;
2) Performance: It is how to use the effort in order to obtain a final goal;
3) Objective: What you wish to obtain by conciliating effort and development in a task;
4) Result: It is the consequence of using this energy;
5) Productivity: It is the proportion between the result obtained and the amount of energy necessary to obtain the result.

Fitz-enz [7] commented on a recently published report, entitled The Conference Board, that only 12% of the interviewees informed to use the management of human capital to help achieving the strategic goals of the company. However, 84% of the same interviewees said that the use of human capital related to the strategies of the company will be bigger on the next 3 years.

Related to indicators of human capital management, Fitz-enz [7] highlights that first it is important to know what is not working properly, the author calls this process system failure measurement (or to measure what is important).

Two questions, according to Fitz-enz [7] help defining what must be measured inside companies:

1) What is the most important thing people management must do?
2) How can it be measured?

We can only measure something that is indeed important, for this reason, the second question is strongly related to the first.

According to Vercellis [8], a data warehouse depends on the objective one desires to achieve. Before drawing a data warehouse, it is necessary to bear in mind which questions must be answered. At this moment, the definitions made by Fitz-enz [7] relate themselves to the main purpose of a data warehouse.

A different approach used by Rainardi [9] is to work with data warehouse with two levels of granularity. With this data warehouse it is possible to navigate with greater flexibility among the data from the data warehouse and their source of formation.

In the next section, the way the concepts approached so far were applied inside the project in execution will be shown.

II. State of the Art

This work in progress aims mostly to update the developed and detailed activities in the article published by Poffo and Misaghi [10].

In this article, new concepts and new directions were considered, taking into account all knowledge obtained and discussed during the HR Metrics Brazil [11], in which many Brazilian companies and multinationals showed how they use the management of indicators inside their companies, supporting the strategy of the company, increasing the involvement of the employees and integrating the indicators of people management to the indicators of the organizations.

III. Methods

According to McClellan [12], for about 20 years, the MES (Manufacturing Execution Systems) systems were the focus of manufacture management. Initially developed to provide the first line of management, with visibility to coordinate service orders and work unites attributions, the MES systems were involved in the essential bond between stakeholders and the events for process of production and logistics. Because MES systems manage and store the events of process of production and logistics they are very important sources of accurate data in real time, integrating themselves to the intelligence of the corporation.

Levinson [13] points out three rules to help identifying development indicators in processes:

1) Indicators need to be objective: Indicators need to be clearly defined and need to be quantifiable (it must be possible to measure through numbers);

2) The process measured through the indicator must be under the subordination of the team or person responsible for the measure (which means the indicator needs its own control);
3) The indicator must encourage the work environment and needs to help the company to obtain corporate results. In addition to being related to the companys wealth or goal it must be understood to all as such.

Besides the data identified by Levinson [13], it is important to consider that employees are measured by their results and not by their work hours anymore, as Sabatini [14] highlights on his research. In other words, an employee that works many long hours but produces as much or less than those who work the regular amount of time, is no good.

Parmenter [5] suggests that indicators of result (KRIs) and indicators of development (KPIs) should be based on the mission, vision and values of the company. This way, the indicators will be consistent with the goal the company pursuits, and also, it will be clearer for the partners of the company to understand the rules they are being charged by. One of the expectations, according to Parmenter [5], is to motivate the partners, because the rules are clear to everyone involved in the process.

Figure 1 shows an example of how to make the deployment of the mission, vision and values of the company to define KPIs and KRIs indicators.

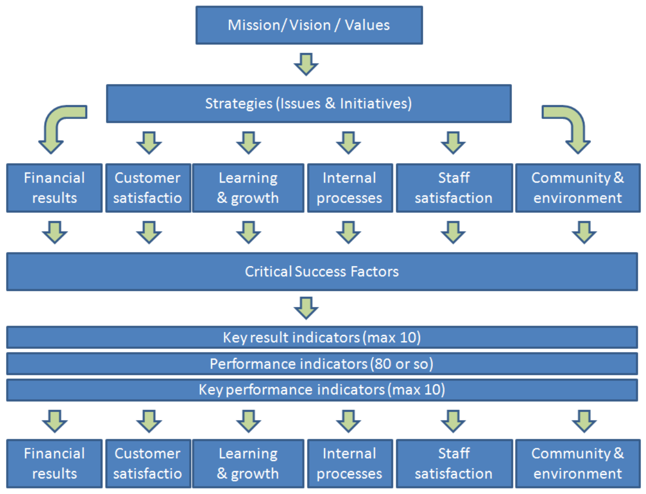


Figure 1. KPIs and KRIs

Among the indicators considered on this work in progress, it is possible to highlight the indicator below (the full paper contains more indicators):

1) Indicator to show the quantity of items produced by hour by an employee.

$$Productivity = \frac{TotalofProducedUnits}{TotalofProductionHours}$$

2) Indicator to show the wage cost of each unit produced, that is, the percentage of an employee's wage that the product retains by its manufacture process.

$$WageCostUnit = \frac{AmountofRemuneration}{TotalofProducedUnits}$$

3) Indicator to show the billing per capita (the value invoiced by each partner, in case this value decreases along time, it indicates an increase of unproductiveness).

$$Billingpercapita = \frac{TotalNetRevenues}{TotalNumberEmployees}$$

4) Indicator to show productivity related to the value paid to the employee for hour.

$$PaidHoursProductivity = \frac{\frac{Netprofitperiod}{totalhoursofwork}}{\frac{Renumberingtotalgross}{totalhoursofwork}}$$

To design the data warehouse the star Model was used. Laberge [15] describes the star model as the model that contains, indeed, a central entity, in which it possesses all the measures as attributes, and a relation with the entity of dimensions.

The goal of this work in progress is to use the approach presented by Rainardi [9] that defines the data warehouse with multiple dimensions, allowing the visualization of the data in a managerial level and once at a granular level.

Figure 2 is the simple implementation of the model of data warehouse for the indicator Paid Hours Productivity, shown on the list above.

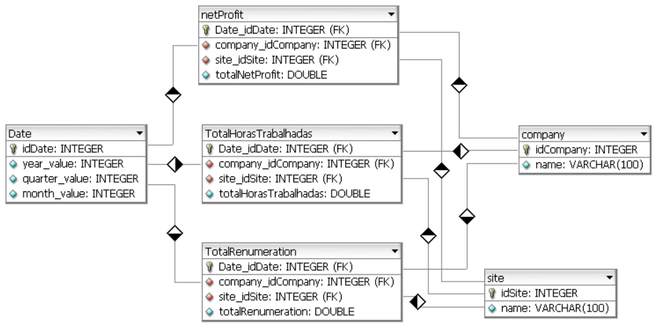

netProfit
Date_idDate: INTEGER (FK)
company_idCompany: INTEGER (FK)
site_idSite: INTEGER (FK)
totalNetProfit: DOUBLE
Date
idDate: INTEGER
year_value: INTEGER
quarter_value: INTEGER
month_value: INTEGER
TotalHorasTrabalhadas
Date_idDate: INTEGER (FK)
company_idCompany: INTEGER (FK)
site_idSite: INTEGER (FK)
totalHorasTrabalhadas: DOUBLE
company
idCompany: INTEGER
name: VARCHAR(100)
TotalRenumeration
Date_idDate: INTEGER (FK)
company_idCompany: INTEGER (FK)
site_idSite: INTEGER (FK)
totalRenumeration: DOUBLE
site
idSite: INTEGER
name: VARCHAR(100)

Figure 2. Data Warehouse Model.

This model of data warehouse considered the date dimension, company and establishment. In case it becomes necessary to visualize any detail in a larger way, it is possible to use the company dimension. In case it is necessary to visualize any data in a smaller dimension (in a more focal way) it is possible to visualize at establishment level. It is possible to use other levels for this treatment, in order to be brief, the other levels were omitted.

One of the proposals of this work in progress is to make it possible to access the level OLTP of data in progress, allowing managers to have access to the initial source of the indicated indicators.

Based on this proposal, the data warehouse model shown on Figure 2 has the objective of allowing managers to see a macro view or detailed as needed. On this figure, through the entities 'empresa' (company) and 'estabelecimento' (establishment) It is possible to create a macro view (company) or a view in detail. The final version of the work will contain the details of all levels that are needed to achieve the goals proposed by the indicators.

IV. Expected Results

Due to the fact that this article is about a work in progress, the final objective expected is the validation of the efficiency of the proposed indicators for people management, with the purpose of reducing waste. In this article, only some indicators that will be fully developed were shown.

Nowadays, literature possesses many works related to measuring the development of manufacturing processes, however, our goal is to measure the development of human resources connected to manufacture processes, considering that in some manufacture processes, human resources are essential.

DW modeling uses the following concepts to work with the level of granularity of the DW model:

1) Dimension: Used to make the dimensions available, in some cases, it will be the level of granularity of the data to be shown;
2) Measures: They are the summarized data, according to the dimension used to visualize the data.

At the present moment, the project finds itself at the final phase of mapping the indicators and elaborations of the DW model that will be used to apply them.

The results obtained so far are related to the identification and categorization of which indicators are important to the company's core business.

By the end of this project, the full proposal of the data warehouse used to implement the indicators will be validated, verifying the advantages and disadvantages obtained by using them.

The DW model defined in this work progress will be validated through its application on a software development project. The various cycles defined in the project will be measured separately, aiming to create ways that will enable people to see the progress of the project.

V. Conclusion

This article reported the developed activities used to list the indicators used in people management that have a connection with the concepts of lean manufacturing.

Besides providing a study of concepts of lean manufacturing, this article also provided a review of the main concepts related to people management, indicators, data warehouse and how to use people management though management of indicators.

At this stage of the project, it is possible to conclude that the indicators created so far will positively aggregate on the process of people management, aiming to eliminate the waste (reminding that this process must be continuous).

Finally, this article adds to a lack of material related about creation of indicators to people management based on the concepts of lean manufacturing.

Acknowledgment

We would like to thank all the people who have directly and indirectly helped the development of this work so far.

References

[1] J. Pettersen, "Defining lean production: Some conceptual and practical issues," *International Journal of education and Information Technologies*, 2009.

[2] B. Carreira, *Lean manufacturing that works: powerful tools for dramatically reducing waste and maximizing profits*. Amacom Books, 2005.

[3] C. Hibbs, S. Jewett, and M. Sullivan, *The art of lean software development*. O' Reilly, 2009.

[4] C. M. Jekiel, *Lean human resources: redesigning HR processes for a culture of continuous improvement*. Productivity Press, 2011.

[5] D. Parmenter, *Key Performance Indicators: Developing, Implementing, and Using Winning KPIs*. John Wiley & Sons, 2009.

[6] J. Bancaleiro, "Indicadores tradicionais de recursos humanos," *Seminar HR Metrics - IIR, Lisboa*, 2006.

[7] J. Fitz-enz, *The new HR analytics: predicting the economic value of your company's human capital investiments*. AMACOM, 2010.

[8] C. Vercellis, *Business Intelligence: Data Mining and Optimization for Decision Making*. John Wiley and Sons, 2009.

[9] V. Rainardi, *Building a Data Warehouse: With Examples in SQL Server*. APress, 2008.

[10] R. Poffo and M. Misaghi, "Bi: Lean manufacturing indicators applied to hr applications - an implementation study," in *Fourth International Conference on Advances in Databases, Knowledge, and Data Applications 2012 (DBKDA 2012)*. Curran Associates, Inc. (Apr 2012), 2012, pp. 34–37.

[11] H. Metrics, "Hr metrics 2012," in *HR Metrics Brasil*. Sao Paulo: IQPC International Quality Productivity Center, Jan 2012.

[12] M. McClellan, "The heart of intelligent manufacturing," set.

[13] W. A. Levinson and R. A. Rerick, *Lean Enterprise: A Synergistic Approach to Minimizing Waste*. ASQ, 2002.

[14] T. Sabatini, "Vagas em ti: quatro exigncias das entrevistas de emprego," jun.

[15] R. Laberge, *The Data Warehouse Mentor: Practical Data Warehouse and Business Intelligence Insights*. McGraw-Hill Companies,Inc., 2011.