# DBKDA 2012

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

ISBN: 978-1-61208-185-4

February 29 - March 5, 2012

Saint Gilles, Reunion Island

**DBKDA 2012 Editors**

Friedrich Laux, Reutlingen University, Germany

Gledson Elias, Federal University of Paraíba, Brazil

Chris Ireland, Open University, UK

# DBKDA 2012

## Foreword

The Fourth International Conference on Advances in Databases, Knowledge, and Data Applications [DBKDA 2012], held between February 29th and March 5th, 2012 in Saint Gilles, Reunion Island, 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, e-health 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 2012 Technical Program Committee, as well as the numerous reviewers. The creation of such a broad and 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 2012. 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 2012 organizing committee for their help in handling the logistics and for their work to make this professional meeting a success.

We hope that DBKDA 2012 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 Saint Gilles, Reunion Island.

**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 2012

# Committee

## DBKDA Advisory 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 2012 Technical Program Committee

Nipun Agarwal, Oracle Corporation, USA
Chris Ireland, Open University, UK
Suad Alagic, University of Southern Maine, USA
Annalisa Appice, Università degli Studi di Bari, Italy
Martine Cadot, LORIA-Nancy, France
Michelangelo Ceci, University of Bari, Italy
Chin-Chen Chang, Feng Chia University Taiwan, Taiwan
Chi-Hua Chen, National Chiao Tung University - Taiwan, R.O.C.
Qiming Chen, HP Labs - Palo Alto, USA
Alfredo Cuzzocrea, ICAR-CNR & University of Calabria, Italy
Maria Del Pilar Angeles, Universidad Nacional Autonoma de Mexico - Del Coyoacan, Mexico
Taoufiq Dkaki, IRIT - Toulouse, France
Cédric du Mouza, CNAM - Paris, France
Jana Dvoráková, Comenius University-Bratislava, Slovakia
Bijan Fadaeenia, Islamic Azad University- Hamedan Branch, Iran
Victor Felea, "A. I. Cuza" University of Iasi, Romania
Daniela Grigori, University of Versailles, France
Martin Grund, Hasso-Plattner-Institute - Potsdam, Germany
Ismail Hababeh, United Arab Emirates University - Al-Ain, UAE
Takahiro Hara, Osaka University, Japan
Bingsheng He, Nanyang Technological University, Singapore
Tobias Hoppe, Ruhr-University of Bochum, Germany
Edward Hung, The Hong Kong Polytechnic University - Hong Kong, PRC
Chris Ireland, Open University, UK
Wassim Jaziri, ISIM Sfax, Tunisia
Mehdi Kargar, York University, Toronto, Canada
Nhien An Le Khac, University College Dublin, Ireland
Sadegh Kharazmi, RMIT University - Melbourne, Australia
Kyoung-Sook Kim, National Institute of Information and Communications Technology, Japan
Christian Kop, University of Klagenfurt, Austria
Jens Krueger, Hasso Plattner Institute / University of Potsdam, Germany
Friedrich Laux, Reutlingen University, Germany

## Table of Contents

# Reasoning about Domain Semantics over Relations, Bags, Partial Relations and Partial Bags

Sebastian Link
*Department of Computer Science*
*The University of Auckland*
*Auckland, New Zealand*
*s.link@auckland.ac.nz*

***Abstract*—Quality database schemata must capture both the structure and semantics of the domain of interest. Classes of data dependencies have been studied extensively to model domain semantics. Traditionally, the theory of data dependencies has been limited to relations. In practice, duplicate and partial information are permitted to occur in database instances. These features are supported to make data processing more efficient. We study the implication problem for an expressive class of data dependencies over all data structures that arise from the two features features. These include bags that permit duplicate tuples, partial relations that permit null marker occurrences, and partial bags that permit duplicate tuples and null marker occurrences. The class of data dependencies studied encompasses uniqueness constraints, functional and multivalued dependencies. We establish axiomatizations and sharp upper bounds for the worst-case time complexity of the implication problem.**

***Keywords*-Data models; Database design; Database semantics; Decision problems; Mathematical logic.**

## I. INTRODUCTION

A database system manages a collection of persistent information in a shared, reliable, effective and efficient way. Most commercial database systems are still founded on *the relational model of data* [1]. Data administrators utilize various classes $\mathcal{C}$ of first-order formulae, called *data dependencies*, to restrict the relations in the database to those considered meaningful to the application domain at hand. A central problem in logic, mathematics and computer science is the *implication problem* of such classes $\mathcal{C}$ [2]. In terms of data dependencies the problem is to decide whether for an arbitrarily given set $\Sigma \cup \{\varphi\}$ of data dependencies in $\mathcal{C}$, $\Sigma$ implies $\varphi$, i.e., whether every database instance that satisfies all the elements of $\Sigma$ also satisfies $\varphi$. For databases specifically, solutions to the implication problem are essential for their modeling and design [3], and can advance many data processing tasks such as updates [4], queries [5], security [6], maintenance [7], cleaning [8], integration [9] and exchange [10]. According to [11] the combined class of uniqueness constraints (UCs) and functional dependencies (FDs) captures around two-thirds, and the class of multivalued dependencies (MVDs) around one-quarter of all uni-relational dependencies (those defined over a single relation schema) that arise in practice. In particular, MVDs are frequently exhibited in database applications [12], e.g., after de-normalization or in views [3]. The next example illustrates how instances of the implication problem arise naturally from table definitions in SQL [13], which has been the industry standard for defining and querying data for the last three decades.

*Example 1:* Consider a table definition SUPPLIES with column headers *A(rticle)*, *S(upplier)*, *L(ocation)* and *C(ost)*. The intention is to collect information about suppliers that deliver articles from a location at a certain cost.

```
CREATE TABLE SUPPLIES (
        Article CHAR[20],
        Supplier VARCHAR NOT NULL,
        Location VARCHAR NOT NULL,
        Cost CHAR[8]);
```

Suppose the database management system enforces the following set $\Sigma$ of constraints: The FD $A \rightarrow S$ says that for every article there is at most one supplier, the FD $AL \rightarrow C$ says that the cost is determined by the article and the location, and the MVD $S \twoheadrightarrow L$ says that the locations are determined by the supplier independently of the articles and costs. Do the following semantically meaningful constraints need to be enforced explicitly, or are they already enforced implicitly by $\Sigma$: i) the UC $u(AL)$, ii) the FD $A \rightarrow C$, and iii) the MVD $A \twoheadrightarrow L$?

SQL table definitions permit occurrences of duplicate tuples and occurrences of a null marker in columns declared `NULL`. While these two features are meant to make data processing more efficient, they do distinguish the 32 billion US dollar market of SQL-based relational database systems from Codd's relational model of data. In this paper, we investigate in detail the impact of these two features on the implication problem of the combined class of UCs, FDs and MVDs. In fact, we use these two features to study the implication problem over the four resulting data structures: relations, bags, partial relations and partial bags. Relations are sets of total tuples. That is, all columns are `NOT NULL` by default and duplicate tuples are not permitted to occur.

*Example 2:* Suppose we use relations to instantiate the SQL table definition of Example 1. Then the constraints i),

ii) and iii) are all enforced implicitly by $\Sigma$. In particular, the FDs $A \to S$ and $AL \to C$ imply the FD $AL \to ALCS$. Since duplicate tuples are identified in sets of tuples, the latter FD is equivalent to the uniqueness constraint $u(AL)$.

Bags of tuples are more general than relations, i.e., sets of tuples. In fact, relations are bags where no duplicate tuples can occur, i.e., no two different tuples can occur that have matching values on all attributes.

*Example 3:* Suppose we use bags of tuples to instantiate the SQL table definition of Example 1. Then the semantically meaningful constraints ii) and iii) are still enforced implicitly by $\Sigma$. However, the uniqueness constraint $u(AL)$ is not enforced implicitly by $\Sigma$. Indeed, the bag

| *Article* | *Supplier* | *Location* | *Cost* |
|---|---|---|---|
| *Kiwi* | *Kiwifruitz* | *Tauranga* | *3* |
| *Kiwi* | *Kiwifruitz* | *Tauranga* | *3* |

satisfies $\Sigma$ but violates $u(AL)$.

Partial relations are sets of partial tuples $t$, i.e., $t(A)$ can carry a null marker occurrence on every attribute $A$. Here, we adapt the most general interpretation of a null marker, denoted by $\texttt{ni}$, i.e., the *no information* interpretation [14], [15], [16]. In general it may happen that two different tuples *subsume* one another. That is, there are two tuples $t$ and $t'$ such that for every attribute $A$ it holds that $t'(A) = \texttt{ni}$ or $t'(A) = t(A)$. We require partial relations to be *subsumption-free*. This requirement is a natural generalization of relations which are duplicate-free, and is in line with previous research [14], [15], [16]. Furthermore, in SQL one can define any attribute $A$ as `NOT NULL`. That is, for every partial tuple $t$ it must hold that $t(A) \neq \texttt{ni}$. We say that the set of attributes declared `NOT NULL` forms the null-free subschema of the underlying schema.

*Example 4:* Suppose we use partial relations to instantiate the SQL table definition of Example 1. The UC $u(AL)$ is implied by $\Sigma$ since duplicate tuples are not allowed in partial relations. For the semantically meaningful constraints ii) and iii) it depends on the null-free subschema whether they are enforced implicitly by $\Sigma$. If the null-free subschema is $\{S, L\}$, then both ii) and iii) are implied by $\Sigma$. However, if it is $\{A, L, C\}$, then the following partial relation

| *Article* | *Supplier* | *Location* | *Cost* |
|---|---|---|---|
| *Kiwi* | $\texttt{ni}$ | *Tauranga* | *3* |
| *Kiwi* | $\texttt{ni}$ | *Gisborne* | *4* |

satisfies $\Sigma$, but violates the FD $A \to C$ and MVD $A \twoheadrightarrow L$.

Finally, partial bags are bags of partial tuples. In particular, partial bags may contain two tuples that subsume one another. This includes the special case of duplicate tuples.

*Example 5:* Suppose we use partial bags to instantiate our SQL table definition. Then the situation is similar to Example 4, but the UC $u(AL)$ is not implied by $\Sigma$.

**Contributions.** We establish finite axiomatizations for the combined class of uniqueness constraints, functional and multivalued dependencies over bags and partial bags. In particular, the presence of duplicate (partial) tuples makes it necessary to include the class of uniqueness constraints into the combined class. That is, in the presence of duplicate tuples, uniqueness constraints are no longer covered by functional dependencies - in contrast to (partial) relations. Our main proof arguments for the case of (partial) bags uses a reduction to the case of (partial) relations, respectively. The benefit of these reductions is to pinpoint exactly which new inference rules are required to gain completeness in each of the cases. Our proof techniques also enable us to establish sharp upper bounds on the worst-case time complexity of the associated decision problems. In particular, the bounds match the currently best known bound for the special case of relations known from the literature. Our findings establish a complete picture of how reasoning about domain semantics in different data structures can be automated effectively and efficiently. Our most general case addresses partial bags, which are used to instantiate SQL table definitions in practice. Our findings close the gap between existing database theory and database practice. The class of data dependencies studied is treated in most introductory textbooks on databases; unfortunately for the case of relations only. Our results provide therefore new insight for students and researchers on the impact of popular data structures on the reasoning about domain semantics. Finally, note that more expressive classes of data dependencies, such as join dependencies, are not finitely axiomatizable [17].

**Organization.** We briefly summarize related work in Section II. The general data model of partial bag schemata is introduced in Section III. The known special cases from the literature are reviewed in Section IV. In Section V we establish axiomatizations for the general case of partial bag schemata. Our proof argument enables us to establish a sharp upper bound on the worst-case time complexity in Section VI. We comment on the applicability of our theories in practice in Section VII. We conclude in Section VIII where we also comment on future work.

## II. Related work

Data dependencies can capture the semantics of the domain of interest in the target database. Therefore, data dependencies are essential to database design, and the maintenance of the database during its lifetime, and all major data processing tasks, cf. [3], [18].

In the relational model, a UC $u(X)$ over relation schema $R$ is satisfied by a relation if and only if the relation satisfies the FD $X \to R$. Hence, in this context it suffices to study the class of FDs and MVDs. Beeri, Fagin and Howard established the first axiomatization for FDs and MVDs [19], [20], [21]. The associated implication problem can be decided in time almost-linear in the input [22].

One of the most important extensions of the relational model [1] is partial information [23]. This is mainly due to the high demand for the correct handling of such information in real-world applications. While there are several possible interpretations of a null marker, many of the previous work on data dependencies is based on Zaniolo's *no information* interpretation [14], [15], [24], [16]. Atzeni and Morfuni established an axiomatization for the class of FDs over partial relations [14]. In particular, they did not permit subsumption between partial tuples and did not consider MVDs. Köhler and Link investigated UCs and FDs over bags, but considered neither null markers nor MVDs [25]. Finally, Hartmann and Link established an axiomatization for the class of FDs and MVDs over partial relations [26].

## III. The SQL data model

In this section we introduce the general SQL data model, which is based on partial bags. We utilize this general model to define the remaining three cases of relations, partial relations and bags as important special cases.

### A. Structures and data structures

Let $\mathfrak{A} = \{A_1, A_2, \ldots\}$ be a (countably) infinite set of distinct symbols, called *attributes*. A *partial bag schema* is a pair $\mathcal{S} = (S, \mathit{nfs}(S))$ consisting of a finite non-empty subset $S$ of $\mathfrak{A}$, and a subset $\mathit{nfs}(S) \subseteq S$. Each attribute $A$ is associated with a countably infinite domain $dom(A)$, which represents the possible values that can occur in the column $A$ represents. To encompass partial information every attribute may have a null marker, denoted by $\mathtt{ni} \in dom(A)$. The intention of $\mathtt{ni}$ is to mean *no information*. This interpretation can therefore model non-existing as well as existing but unknown information [14], [16], but it cannot distinguish between the two - as is the exact case in SQL.

For attribute sets $X$ and $Y$ we may write $XY$ for the set union $X \cup Y$. If $X = \{A_1, \ldots, A_m\}$, then we may write $A_1 \cdots A_m$ for $X$. In particular, we may write simply $A$ to represent the singleton $\{A\}$. A *partial tuple* over $\mathcal{S}$ is a function $t : S \to \bigcup_{A\in S} dom(A)$ with $t(A) \in dom(A)$ for all $A \in S$, and $t(A) \neq \mathtt{ni}$ for all $A \in \mathit{nfs}(S)$. The null marker occurrence $t(A) = \mathtt{ni}$ associated with an attribute $A$ in a partial tuple $t$ means that no information is available about the attribute $A$ for the partial tuple $t$. For $X \subseteq S$ let $t(X)$ denote the restriction of the partial tuple $t$ over $\mathcal{S}$ to $X$. A partial tuple $t$ is said to be $X$-total, if for all $A \in X$ it holds that $t(A) \neq \mathtt{ni}$. Hence, every partial tuple over $\mathcal{S}$ is $\mathit{nfs}(S)$-total. A *partial bag* over $\mathcal{S}$ is a finite multi-set of partial tuples over $\mathcal{S}$.

A *bag schema* is a partial bag schema $(S, \mathit{nfs}(S))$ where $\mathit{nfs}(S) = S$. Here, we may simply write $S$ instead of $(S, S)$. Consequently, all partial tuples $t$ over a bag schema $S$ are $S$-total partial tuples, i.e., for all $A \in S$ it holds that $t(A) \neq \mathtt{ni}$. In this case, we may also speak of total tuples or just tuples.

For two partial tuples $t$ and $t'$ we say that $t$ *subsumes* $t'$, if for every attribute $A$ it holds that $t'(A) = \mathtt{ni}$ or $t'(A) = t(A)$. A partial relation over $(S, \mathit{nfs}(S))$ is a partial bag that is *subsumption-free*, i.e., there are no two different partial tuples in the partial relation that subsume one another. We call a partial bag schema $(S, \mathit{nfs}(S))$ a *partial relation schema* if we restrict all partial bags over $(S, \mathit{nfs}(S))$ to be partial relations.

Finally, a *relation schema* is a partial relation schema $(S, \mathit{nfs}(S))$ where $\mathit{nfs}(S) = S$. Again, we may simply write $S$ instead of $(S, S)$. Consequently, every partial relation over a relation schema is a relation, i.e., a set of tuples.

*Example 6:* The following database instance over (Supplies, *nfs*(Supplies)), where *nfs*(Supplies) $=$ $\{$*Article, Location, Cost*$\}$, is a partial bag that is not a partial relation.

| *Article* | *Supplier* | *Location* | *Cost* |
|---|---|---|---|
| *Kiwi* | *Kiwifruitz* | *Gisborne* | *4* |
| *Kiwi* | ni | *Gisborne* | *4* |

Indeed, the first tuple subsumes the second tuple.

### B. Semantics

In what follows we define the syntax and semantics of uniqueness constraints, functional and multivalued dependencies in the context of partial bags. We will briefly comment on the restrictions to the special cases of relations, partial relations and bags.

Following the SQL standard a *uniqueness constraint* (UC) over a partial bag schema $\mathcal{S} = (S, \mathit{nfs}(S))$ is an expression $u(X)$ where $X \subseteq S$. A partial bag $\mathfrak{b}$ over $\mathcal{S}$ is said to satisfy the uniqueness constraint $u(X)$ over $\mathcal{S}$ ($\models_{\mathfrak{b}} u(X)$) if and only if for all partial tuples $t, t' \in \mathfrak{b}$ the following holds: if $t \neq t'$ and $t$ and $t'$ are both $X$-total, then there is some $A \in X$ such that $t(A) \neq t'(A)$. Note that the notion of a uniqueness constraint over bag and relation schemata matches the well-known notion of a key.

Functional dependencies are important for the relational [1] and other data models [27], [28], [29], [30], [31]. Generalizing notions by Lien [15], a *functional dependency* (FD) over a partial bag schema $\mathcal{S}$ is a statement $X \to Y$ where $X, Y \subseteq S$. The FD $X \to Y$ over $\mathcal{S}$ is satisfied by a partial bag $\mathfrak{b}$ over $\mathcal{S}$ ($\models_{\mathfrak{b}} X \to Y$) if and only if for all $t, t' \in \mathfrak{b}$ the following holds: if $t$ and $t'$ are both $X$-total and $t(X) = t'(X)$, then $t(Y) = t'(Y)$. We call $X \to Y$ *trivial* whenever $Y \subseteq X$, and non-trivial otherwise. The general FD definition is consistent with the *no information* interpretation [14], [15]. For bag and relation schemata the notion of a functional dependency reduces to that of the standard definition of a functional dependency [18], and so is a sound generalization. Note that any partial relation $\mathfrak{b}$ satisfies the FD $X \to S$ over $\mathcal{S}$ if and only if $\mathfrak{b}$ satisfies the UC $u(X)$. This is invalid for bags and partial bags.

Generalizing notions by Lien [15], a *multivalued dependency* (MVD) over $\mathcal{S}$ is a statement $X \twoheadrightarrow Y$ where $X, Y \subseteq S$. The MVD $X \twoheadrightarrow Y$ over $\mathcal{S}$ is satisfied by a partial bag $\mathfrak{b}$ over $\mathcal{S}$, denoted by $\models_{\mathfrak{b}} X \twoheadrightarrow Y$, if and only if for all $t, t' \in \mathfrak{b}$ the following holds: if $t$ and $t'$ are both $X$-total and $t(X) = t'(X)$, then there is some $\bar{t} \in \mathfrak{b}$ such that $\bar{t}(XY) = t(XY)$ and $\bar{t}(X(S-Y)) = t'(X(S-Y))$. We call $X \twoheadrightarrow Y$ *trivial* whenever $Y \subseteq X$ or $XY = S$, and non-trivial otherwise. This MVD definition is consistent with the *no information* interpretation [15]. For bag and relation schemata the notion of an MVD reduces to that of the standard definition of an MVD [32], and so is a sound generalization.

*Example 7:* Consider the partial bag schema $(\text{SUPPLIES}, \textit{nfs}(\text{SUPPLIES}))$ where $\textit{nfs}(\text{SUPPLIES}) = \{\textit{Supplier}, \textit{Location}\}$. The partial relation

| *Article* | *Supplier* | *Location* | *Cost* |
|---|---|---|---|
| *Kiwi* | `ni` | *Tauranga* | *3* |
| *Kiwi* | `ni` | *Gisborne* | *4* |

satisfies the UC $u(AL)$, the FDs $A \to S$ and $AL \to C$, and the MVD $S \twoheadrightarrow L$. It violates the UC $u(A)$, the FD $A \to C$ and the MVD $A \twoheadrightarrow L$.

### C. Semantic implication and syntactic inference

For a set $\Sigma$ of constraints over some partial bag schema $\mathcal{S}$, we say that a partial bag $\mathfrak{b}$ over $\mathcal{S}$ *satisfies* $\Sigma$ ($\models_{\mathfrak{b}} \Sigma$) if $\mathfrak{b}$ satisfies every $\sigma \in \Sigma$. If for some $\sigma \in \Sigma$, $\mathfrak{b}$ does not satisfy $\sigma$ we say that $\mathfrak{b}$ violates $\sigma$ (and violates $\Sigma$) and write $\not\models_{\mathfrak{b}} \sigma$ ($\not\models_{\mathfrak{b}} \Sigma$). In the general case of partial bags we are interested in the combined class $\mathcal{C}$ of UCs, FDs and MVDs.

Constraints interact with one another. Let $\mathcal{S}$ be a partial bag schema, and let $\Sigma \cup \{\varphi\}$ be a set of UCs, FDs and MVDs over $\mathcal{S}$. We say that $\Sigma$ *implies* $\varphi$ ($\Sigma \models \varphi$) if every partial bag $\mathfrak{b}$ over $\mathcal{S}$ that satisfies $\Sigma$ also satisfies $\varphi$. If $\Sigma$ does not imply $\varphi$ we may also write $\Sigma \not\models \varphi$. For $\Sigma$ we let $\Sigma^* = \{\varphi \mid \Sigma \models \varphi\}$ be the *semantic closure* of $\Sigma$, i.e., the set of all UCs, FDs and MVDs implied by $\Sigma$. In order to determine the implied constraints we use a syntactic approach by applying inference rules. These inference rules have the form

$$\frac{\text{premise}}{\text{conclusion}}\,\text{condition,}$$

and inference rules without any premise are called axioms. An inference rule is called *sound*, if whenever the set of constraints in the premise of the rule are satisfied by some partial bag over $\mathcal{S}$ and the constraints satisfy the conditions of the rule, then the partial bag also satisfies the constraint in the conclusion of the rule. We let $\Sigma \vdash_{\mathfrak{S}} \varphi$ denote the *inference* of $\varphi$ from $\Sigma$ by $\mathfrak{S}$. That is, there is some sequence $\gamma = [\sigma_1, \ldots, \sigma_n]$ of constraints such that $\sigma_n = \varphi$ and every $\sigma_i$ is an element of $\Sigma$ or results from an application of an inference rule in $\mathfrak{S}$ to some elements in $\{\sigma_1, \ldots, \sigma_{i-1}\}$.

$$\frac{}{XY \to Y}\ \text{(reflexivity, } \mathcal{R}_{\text{F}})\qquad \frac{X \to Y}{XU \to YV}\,V \subseteq U\ \text{(FD augmentation, } \mathcal{A}_{\text{F}})$$

$$\frac{X \to Y \qquad Y \to Z}{X \to Z}\ \text{(transitivity, } \mathcal{T}'_{\text{F}})$$

$$\frac{X \twoheadrightarrow Y}{X \twoheadrightarrow S - Y}\ (S\text{-complementation, } \mathcal{C}^{S}_{\text{M}})\qquad \frac{X \twoheadrightarrow Y}{XU \twoheadrightarrow YV}\,V \subseteq U\ \text{(MVD augmentation, } \mathcal{A}_{\text{M}})$$

$$\frac{X \twoheadrightarrow Y \qquad Y \twoheadrightarrow Z}{X \twoheadrightarrow Z - Y}\ \text{(pseudo-transitivity, } \mathcal{T}'_{\text{M}})$$

$$\frac{X \to Y}{X \twoheadrightarrow Y}\ \text{(MVD implication, } \mathcal{I}_{\text{FM}})\qquad \frac{X \twoheadrightarrow Y \qquad Y \to Z}{X \to Z - Y}\ \text{(mixed pseudo-transitivity, } \mathcal{T}'_{\text{FM}})$$

Table I
AXIOMATIZATION $\mathfrak{R}$ OVER RELATION SCHEMA $S$

For a finite set $\Sigma$, let $\Sigma^+_{\mathfrak{S}} = \{\varphi \mid \Sigma \vdash_{\mathfrak{S}} \varphi\}$ be its *syntactic closure* under inferences by $\mathfrak{S}$. A set $\mathfrak{S}$ of inference rules is said to be *sound* (*complete*) for the implication of UCs, FDs and MVDs if for every partial bag schema $\mathcal{S}$ and for every set $\Sigma$ of UCs, FDs and MVDs over $\mathcal{S}$ we have $\Sigma^+_{\mathfrak{S}} \subseteq \Sigma^*$ ($\Sigma^* \subseteq \Sigma^+_{\mathfrak{S}}$). The (finite) set $\mathfrak{S}$ is said to be a (finite) *axiomatization* for the implication of UCs, FDs and MVDs if $\mathfrak{S}$ is both sound and complete.

## IV. AXIOMATIZATIONS FOR RELATIONS AND PARTIAL RELATIONS

In this section we briefly review a well-known axiomatization for the class of FDs and MVDs over relations. We then review a recent axiomatization for the same class of data dependencies over partial relations.

### A. Relations

Beeri, Fagin, and Howard [19] established the first axiomatization for the class of FDs and MVDs over relations. The axiomatization $\mathfrak{R}$ of Table I is based on the (mixed) pseudo-transitivity rules by Zaniolo [21].

The following example demonstrates the use of the inference rules to infer some data dependencies implied over relations. Firstly, it highlights how in the absence of partial data, the transitivity rules can be applied soundly. Secondly, it highlights how in the absence of duplicate tuples, FDs can be used to infer uniqueness constraints.

*Example 8:* Consider the relation schema SUPPLIES and the set $\Sigma$ containing the FDs $A \to S$, $AL \to C$ and the MVD $S \twoheadrightarrow L$.

$$\frac{}{XY \to Y} \quad \text{(reflexivity, } \mathcal{R}_{\mathrm{F}}\text{)} \qquad \frac{X \to YZ}{X \to Y} \quad \text{(decomposition, } \mathcal{D}_{\mathrm{F}}\text{)}$$

$$\frac{X \to Y \qquad X \to Z}{X \to YZ} \quad \text{(FD union, } \mathcal{U}_{\mathrm{F}}\text{)}$$

$$\frac{X \twoheadrightarrow Y}{X \twoheadrightarrow S - Y} \quad \text{(}S\text{-complementation, } \mathcal{C}^{S}_{\mathrm{M}}\text{)} \qquad \frac{X \twoheadrightarrow Y \qquad X \twoheadrightarrow Z}{X \twoheadrightarrow YZ} \quad \text{(MVD union, } \mathcal{U}_{\mathrm{M}}\text{)}$$

$$\frac{X \twoheadrightarrow W \qquad Y \twoheadrightarrow Z}{X \twoheadrightarrow Z - W}\ {\scriptstyle Y \subseteq X(W \cap S')} \quad \text{(null pseudo-transitivity, } \mathcal{T}_{\mathrm{M}}\text{)}$$

$$\frac{X \to Y}{X \twoheadrightarrow Y} \quad \text{(MVD implication, } \mathcal{I}_{\mathrm{FM}}\text{)} \qquad \frac{X \twoheadrightarrow W \qquad Y \to Z}{X \to Z - W}\ {\scriptstyle Y \subseteq X(W \cap S')} \quad \text{(null mixed pseudo-transitivity, } \mathcal{T}_{\mathrm{FM}}\text{)}$$

Table II
AXIOMATIZATION $\mathfrak{pR}$ OVER PARTIAL RELATION SCHEMA $(S, S')$

An application of the *MVD implication rule* $\mathcal{I}_{\mathrm{FM}}$ to $A \to S$ results in the MVD $A \twoheadrightarrow S$. We can apply the *pseudo-transitivity rule* $\mathcal{T}'_{\mathrm{M}}$ to the MVDs $A \twoheadrightarrow S$ and $S \twoheadrightarrow L$ to infer the MVD $A \twoheadrightarrow L$.

An application of the *MVD augmentation rule* $\mathcal{A}_{\mathrm{M}}$ to $A \twoheadrightarrow L$ yields the MVD $A \twoheadrightarrow AL$. An application of the *mixed pseudo-transitivity rule* $\mathcal{T}'_{\mathrm{FM}}$ to the MVD $A \twoheadrightarrow AL$ and the FD $AL \to C$ results in the FD $A \to C$.

An application of the *FD augmentation rule* $\mathcal{A}_{\mathrm{F}}$ to $A \to S$ yields the FD $AC \to SC$, and an application of the same rule to the FD $AL \to C$ yields the FD $AL \to AC$. An application of the *transitivity rule* $\mathcal{T}'_{\mathrm{F}}$ to the FDs $AL \to AC$ and $AC \to SC$ results in the FD $AL \to SC$. A final application of the *FD augmentation rule* $\mathcal{A}_{\mathrm{F}}$ to $AL \to SC$ yields the FD $AL \to ACLS$. Note that over a relation schema $R$ the FD $X \to R$ is satisfied by the same relations as the UC $u(AL)$. Thus, the last inference yields the UC $u(AL)$.

## B. Partial relations

Over partial relations, Hartmann and Link recently established the axiomatization $\mathfrak{pR}$ for the class of FDs and MVDs [26]. As Example 4 shows, the choice of a null-free subschema has an impact on the data dependencies implied. In particular, the presence of partial data requires that the applicability of the (mixed) pseudo-transitivity rules are suitably restricted.

The next example illustrates applications of the null (mixed) pseudo-transitivity rules to infer implied data dependencies. Note how changes in the null-free subschema influence the applicability of these rules, cf. Example 4.

*Example 9:* Consider the partial relation schema $(\text{SUPPLIES}, \{SL\})$ and the set $\Sigma$ containing the FDs $A \to S$, $AL \to C$ and the MVD $S \twoheadrightarrow L$.

An application of the *MVD implication rule* $\mathcal{I}_{\mathrm{FM}}$ to $A \to S$ results in the MVD $A \twoheadrightarrow S$. We can apply the *null pseudo-transitivity rule* $\mathcal{T}'_{\mathrm{M}}$ to the MVDs $A \twoheadrightarrow S$ and $S \twoheadrightarrow L$ to infer the MVD $A \twoheadrightarrow L$. Note that $S \in nfs(\text{SUPPLIES})$.

An application of the *reflexivity axiom* $\mathcal{R}_{\mathrm{F}}$ followed by an application of the *MVD implication rule* $\mathcal{I}_{\mathrm{FM}}$ results in the MVD $A \twoheadrightarrow A$. An application of the *MVD union rule* $\mathcal{U}_{\mathrm{M}}$ to $A \twoheadrightarrow A$ and $A \twoheadrightarrow L$ results in the MVD $A \twoheadrightarrow AL$. An application of the *null mixed pseudo-transitivity rule* $\mathcal{T}_{\mathrm{FM}}$ to the MVD $A \twoheadrightarrow AL$ and the FD $AL \to C$ results in the FD $A \to C$. Again, note here that $L \in nfs(\text{SUPPLIES})$.

## V. AXIOMATIZATIONS FOR BAGS AND PARTIAL BAGS

In this section we establish the first main results of this article, i.e., finite axiomatizations for the combined class of UCs, FDs, and MVDs over bags and over partial bags. We prove the general case of partial bags in detail.

### A. Partial bags

Let $\mathfrak{pB}$ denote the set of inference rules in Table IV. We first establish the soundness of the rules in $\mathfrak{pB}$.

*Lemma 1:* The set $\mathfrak{pB}$ of inference rules is sound.

*Proof:* The soundness of the rules in $\mathfrak{pR}$ has been established in previous work [26]. It remains to show the soundness of the *FD implication rule* $\mathcal{I}_{\mathrm{UF}}$ and the *null pullback rule* $\mathcal{P}_{\mathrm{UF}}$.

For the soundness of the *FD implication rule* $\mathcal{I}_{\mathrm{UF}}$ assume there is some partial bag $\mathfrak{b}$ that violates the FD $X \to Y$. Then there must be two different tuples $t, t' \in \mathfrak{b}$ that are $X - total$ and $t(X) = t'(X)$. Consequently, $\mathfrak{b}$ also violates the UC $u(X)$.

For the soundness of the *null pullback rule* $\mathcal{P}_{\mathrm{UF}}$ assume there is some partial bag $\mathfrak{b}$ over $\mathcal{S} = (S, S')$ that violates the uniqueness constraint $u(X)$. Then there must be two different tuples $t, t' \in \mathfrak{b}$ that are $X - total$ and $t(X) = t'(X)$. If $\mathfrak{b}$ violates the FD $X \to Y$, then we are done. Otherwise, it follows that $t(Y) = t'(Y)$. If $Y \not\subseteq XS'$, then we are done. Otherwise, it follows that $t$ and $t'$ are both $Y$-total. Consequently, $\mathfrak{b}$ violates the UC $u(Y)$. ∎

For the completeness of $\mathfrak{pB}$ we use the result that the set $\mathfrak{pR}$ forms an axiomatization for FDs and MVDs over partial relations [26]. In fact, the completeness of $\mathfrak{pB}$ follows from that of $\mathfrak{pR}$ and the following lemma. For a set $\Sigma$ of UCs, FDs, and MVDs over partial bag schema $(S, nfs(S))$ let $\Sigma[\text{FM}] = \{X \to S \mid u(X) \in \Sigma\} \cup \{X \to Y \mid X \to Y \in \Sigma\} \cup \{X \twoheadrightarrow Y \mid X \twoheadrightarrow Y \in \Sigma\}$.

*Lemma 2:* Let $\Sigma$ be a set of UCs, FDs and MVDs over the partial bag schema $(S, S')$. Then the following hold:

1) $\Sigma \models X \to Y$ if and only if $\Sigma[\text{FM}] \models X \to Y$,
2) $\Sigma \models X \twoheadrightarrow Y$ if and only if $\Sigma[\text{FM}] \models X \twoheadrightarrow Y$,

| $XS'$ | $S - XS'$ |
|---|---|
| $0 \cdots 0$ | ni$\cdots$ni |
| $0 \cdots 0$ | ni$\cdots$ni |

Table III
THE PARTIAL BAG $\mathfrak{b}_X$

3) $\Sigma \models u(X)$ if and only if $\Sigma[\text{FM}] \models X \to S$ and there is some $u(Z) \in \Sigma$ such that $Z \subseteq XS'$.

*Proof:* The *if* directions of 1) and 2) follows straight from the soundness of the *FD implication rule* $\mathcal{I}_{\text{UF}}$. For the *only if* direction of 2), for example, assume that there is a partial bag $\mathfrak{b}$ over $\mathcal{S}$ such that $\models_{\mathfrak{b}} \Sigma[\text{FM}]$ and $\not\models_{\mathfrak{b}} X \twoheadrightarrow Y$. Then it follows from a result in [26] that there are two tuples $t, t' \in \mathfrak{b}$ such that $\models_{\{t,t'\}} \Sigma[\text{FM}]$ and $\not\models_{\{t,t'\}} X \twoheadrightarrow Y$. Consequently, $t(X) = t'(X)$, and $t, t'$ are $X$-total, and for some $A \in S - X$, $t(A) \neq t'(A)$. Suppose there is some $u(Z) \in \Sigma$ such that $\not\models_{\{t,t'\}} u(Z)$. Then $t, t'$ are $Z$-total and $t(Z) = t'(Z)$. However, $\models_{\{t,t'\}} Z \to S$ and thus $t(S) = t'(S)$, a contradiction. Consequently, $\models_{\{t,t'\}} \Sigma$ and $\not\models_{\{t,t'\}} X \twoheadrightarrow Y$. The remaining direction of 1) is similar.

It remains to show 3). Suppose that $\Sigma[\text{FM}] \models X \to S$ and there is some $u(Z) \in \Sigma$ such that $Z \subseteq XS'$. Then it follows that $X \to Z$ is implied by $\Sigma[\text{FM}]$ due to the soundness of the *decomposition rule* $\mathcal{D}_{\text{F}}$. The soundness of the *null pullback rule* $\mathcal{P}_{\text{UF}}$ allows us to derive the fact that $u(X)$ is implied by $\Sigma$. It remains to show the *only if* direction of 3).

From $\Sigma \models u(X)$ we conclude $\Sigma \models X \to S$ by soundness of the *FD implication rule* $\mathcal{I}_{\text{UF}}$. According to 1) it follows that $\Sigma[\text{FM}] \models X \to S$. It remains to show that there is some $u(Z) \in \Sigma$ such that $Z \subseteq XS'$. Assume to the contrary that for all $u(Z) \in \Sigma$ we have $Z \not\subseteq XS'$. Under this assumption we will derive the contradiction that $\Sigma \not\models u(X)$ by constructing a two-tuple partial bag $\mathfrak{b}_X$ that satisfies $\Sigma$ and violates $u(X)$. Indeed, $\mathfrak{b}_X$ is the bag in Table III. Clearly, it satisfies $\Sigma[\text{FM}]$. Moreover, under our current assumption it is true that for every $u(Z) \in \Sigma$ we have $Z \cap (S - XS') \neq \emptyset$. Thus, $\models_{\mathfrak{b}_X} u(Z)$ for all $u(Z) \in \Sigma$. It also follows that $\not\models_{\mathfrak{b}_X} u(X)$. Hence, $\Sigma \not\models u(X)$, a contradiction. Consequently, our assumption must have been wrong and there is some $u(Z) \in \Sigma$ such that $Z \subseteq XS'$. ■

*Theorem 1:* The set $\mathfrak{pB}$ of inference rules forms a finite axiomatization for the implication of UCs, FDs and MVDs over partial bags.

*Proof:* The soundness of $\mathfrak{pB}$ was shown in Lemma 1. We establish the completeness of $\mathfrak{pB}$ by showing that for an arbitrary partial bag schema $\mathcal{S} = (S, \mathit{nfs}(S))$, and an arbitrary set $\Sigma \cup \{\varphi\}$ of UCs, FDs and MVDs over $\mathcal{S}$ the following holds: if $\Sigma \models \varphi$, then $\Sigma \vdash_{\mathfrak{pB}} \varphi$. We consider two cases. In case (1) $\varphi$ denotes the FD $X \to Y$ or the MVD $X \twoheadrightarrow Y$. Then we know by Lemma 2 that $\Sigma[\text{FM}] \models \varphi$ holds. From the completeness of $\mathfrak{pR}$ for the implication of FDs and MVDs over partial relations we conclude that $\Sigma[\text{FM}] \vdash_{\mathfrak{B}} \varphi$. Since $\mathfrak{pR} \subseteq \mathfrak{pB}$ holds we know that $\Sigma[\text{FM}] \vdash_{\mathfrak{pB}} \varphi$ holds, too. The *FD implication rule* $\mathcal{I}_{\text{UF}}$ shows for all $\sigma \in \Sigma[\text{FM}]$ that $\Sigma \vdash_{\mathfrak{pB}} \sigma$ holds. Consequently, we have $\Sigma \vdash_{\mathfrak{pB}} \varphi$. This concludes case (1). In case (2) $\varphi$ denotes the UC $u(X)$. From $\Sigma \models u(X)$ we conclude by Lemma 2 that there is some $u(Z) \in \Sigma$ such that $Z \subseteq XS'$ holds. We also conclude from $\Sigma \models u(X)$ that $\Sigma \models X \to Z$ holds by soundness of the *FD implication rule* $\mathcal{I}_{\text{UF}}$. From case (1) it follows that $\Sigma \vdash_{\mathfrak{pB}} X \to Z$ holds. A final application of the *null pullback rule* $\mathcal{P}_{\text{UF}}$ shows that $\Sigma \vdash_{\mathfrak{pB}} \varphi$ holds. ■

$$\frac{}{XY \to Y}\ \text{(reflexivity, } \mathcal{R}_{\text{F}}) \qquad \frac{X \to YZ}{X \to Y}\ \text{(decomposition, } \mathcal{D}_{\text{F}})$$

$$\frac{X \to Y \qquad X \to Z}{X \to YZ}\ \text{(FD union, } \mathcal{U}_{\text{F}})$$

$$\frac{X \twoheadrightarrow Y}{X \twoheadrightarrow S - Y}\ (S\text{-complementation, } \mathcal{C}^S_{\text{M}}) \qquad \frac{X \twoheadrightarrow Y \qquad X \twoheadrightarrow Z}{X \twoheadrightarrow YZ}\ \text{(MVD union, } \mathcal{U}_{\text{M}})$$

$$\frac{X \twoheadrightarrow W \qquad Y \twoheadrightarrow Z}{X \twoheadrightarrow Z - W}\, {\scriptstyle Y \subseteq X(W \cap S')}\ \text{(null pseudo-transitivity, } \mathcal{T}_{\text{M}})$$

$$\frac{u(X)}{X \to Y}\ \text{(FD implication, } \mathcal{I}_{\text{UF}}) \qquad \frac{X \to Y \quad u(Y)}{u(X)}\, {\scriptstyle Y \subseteq XS'}\ \text{(null pullback, } \mathcal{P}_{\text{UF}})$$

$$\frac{X \to Y}{X \twoheadrightarrow Y}\ \text{(MVD implication, } \mathcal{I}_{\text{FM}}) \qquad \frac{X \twoheadrightarrow W \qquad Y \to Z}{X \to Z - W}\, {\scriptstyle Y \subseteq X(W \cap S')}\ \text{(null mixed pseudo-transitivity, } \mathcal{T}_{\text{FM}})$$

Table IV
AXIOMATIZATION $\mathfrak{pB}$ OVER PARTIAL BAG SCHEMA $(S, S')$

## B. Bags

The set $\mathfrak{B}$ of inference rules in Table V forms a finite axiomatization for the combined class of UCs, FDs, and MVDs over bag schemata. The proofs necessary to establish this axiomatization are similar to those we have just described in detail for the case of partial bags. Indeed, the main argument utilizes the fact that $\mathfrak{R}$ forms an axiomatization for FDs and MVDs over relations, and the restriction of Lemma 2 to bag schemata. We omit the details. Finally, we would like to emphasize the uniformity in generalizing the axiomatization from partial relations to partial bags, and relations to bags. It suffices to add the *FD implication rule* $\mathcal{I}_{\text{UF}}$ and, in case of bags, the *pullback rule* $\mathcal{P}'_{\text{UF}}$, and, in case of partial bags the *null pullback rule* $\mathcal{P}_{\text{UF}}$, respectively.

$$\frac{}{XY \to Y}\ \text{(reflexivity, } \mathcal{R}_{\text{F}}) \qquad \frac{X \to Y}{XU \to YV}\, V\subseteq U\ \text{(FD augmentation, } \mathcal{A}_{\text{F}})$$

$$\frac{X \to Y \quad Y \to Z}{X \to Z}\ \text{(transitivity, } \mathcal{T}'_{\text{F}})$$

$$\frac{X \twoheadrightarrow Y}{X \twoheadrightarrow S - Y}\ \text{(}S\text{-complementation, } \mathcal{C}^{S}_{\text{M}}) \qquad \frac{X \twoheadrightarrow Y}{XU \twoheadrightarrow YV}\, V\subseteq U\ \text{(MVD augmentation, } \mathcal{A}_{\text{M}})$$

$$\frac{X \twoheadrightarrow Y \quad Y \twoheadrightarrow Z}{X \twoheadrightarrow Z - Y}\ \text{(pseudo-transitivity, } \mathcal{T}'_{\text{M}})$$

$$\frac{u(X)}{X \to Y}\ \text{(FD implication, } \mathcal{I}_{\text{UF}}) \qquad \frac{X \to Y \quad u(Y)}{u(X)}\ \text{(pullback, } \mathcal{P}'_{\text{UF}})$$

$$\frac{X \to Y}{X \twoheadrightarrow Y}\ \text{(MVD implication, } \mathcal{I}_{\text{FM}}) \qquad \frac{X \twoheadrightarrow Y \quad Y \to Z}{X \to Z - Y}\ \text{(mixed pseudo-transitivity, } \mathcal{T}'_{\text{FM}})$$

Table V
AXIOMATIZATION $\mathfrak{B}$ OVER BAG SCHEMA $S$

## VI. SHARP UPPER BOUNDS FOR THE TIME COMPLEXITY

Lemma 2 also establishes an algorithmic characterization of the associated implication problems. In fact, it suffices to compute the *attribute set closure* $X^*_{\Sigma[\text{FM}]} := \{A \in S \mid \Sigma[\text{FM}] \vdash_{\mathfrak{pR}} X \to A\}$ and the *dependency basis* $\textit{DepB}_{\Sigma[\text{FM}]}(X)$ of $X$ with respect to $\Sigma$[FM] [26]. In particular, $\textit{DepB}_{\Sigma[\text{FM}]}(X)$ is the set of atoms for the Boolean algebra $(\textit{Dep(X)}, \subseteq, \cup, \cap, (\cdot)^{\mathcal{C}}_{S}, \emptyset, S)$ where $\textit{Dep(X)} = \{Y \subseteq S \mid \Sigma[\text{FM}] \vdash_{\mathfrak{pR}} X \twoheadrightarrow Y\}$. The size $||\varphi||$ of $\varphi$ is the total number of attributes occurring in $\varphi$, and the size $||\Sigma||$ of $\Sigma$ is the sum of $||\sigma||$ over all elements $\sigma \in \Sigma$. For a set $\Sigma$ of FDs and MVDs let $k_\Sigma$ denote the number of MVDs in $\Sigma$, $p_\Sigma$ denote the number of sets in the dependency basis $\textit{DepB}_\Sigma(X)$ of $X$ with respect to $\Sigma$, $\bar{p}_\Sigma$ denote the number of sets in $\textit{DepB}_\Sigma(X)$ that have non-empty intersection with the right-hand side of $\varphi$, and $\Sigma[XS']$ denote the set of FDs and MVDs in $\Sigma$ where the left-hand side is a subset of $XS'$. The following result follows from Lemma 2 and the upper time bound established by Galil for relational databases [22].

*Theorem 2:* Let $\varphi$ denote either the UC $u(X)$, the FD $X \to Y$, or the MVD $X \twoheadrightarrow Y$ over the partial bag schema $\mathcal{S} = (S, S')$. The problem whether $\varphi$ is implied by a set $\Sigma$ of UCs, FDs and MVDs over $\mathcal{S}$ can be decided in $\mathcal{O}(||\Sigma|| + \min\{k_{\Sigma[\text{FM}][XS']}, \log \bar{p}_{\Sigma[\text{FM}][XS']}\} \times ||\Sigma[\text{FM}][XS']||)$ time.

The bound from Theorem 2 becomes $\mathcal{O}(||\Sigma|| + \min\{k_{\Sigma[\text{FM}}, \log \bar{p}_{\Sigma[\text{FM}]}\} \times ||\Sigma[\text{FM}]||)$ time for bag schemata.

## VII. APPLICABILITY OF THEORY IN PRACTICE

As mentioned in the introduction data dependencies are essential in the design and maintenance of databases, and useful for many data processing tasks. In database practice, e.g., in SQL database systems, partial and duplicate information are allowed to occur. Partial information allows users of the database system to enter information into the database that is incomplete. The permission of duplicate information is motivated by the costs of duplicate identification and removal. The applicability of the theory of data dependencies in practice has been limited since partial and duplicate information has been largely ignored in theory. The current paper provides a summary of solutions to the implication problem for the expressive class of uniqueness constraints, functional and multivalued dependencies over all data structures arising from the permission or prohibition of partial and duplicate information. While the permission of both features covers the general case of SQL databases, many data engineers decide to specify at least a primary key on their schemata, i.e., a set of attributes that are all specified `NOT NULL`, and enforce uniqueness of tuples on their projection. Thus, duplicate tuples are not allowed to occur in database instances, which results in the usefulness of our theory over partial relations. On the other hand, data engineers may want to prohibit the occurrence of partial information. This can be done by specifying all attributes of the schema as `NOT NULL`. In this case, our theory over bags is useful. Whenever the engineers decide to prohibit partial and duplicate information by the methods above, the theory over relations can be applied.

## VIII. CONCLUSION AND FUTURE WORK

Quality database schemata must capture the structure and semantics of their application domain. Database constraints enforce the domain semantics within database systems. They are therefore invaluable for database design and data processing. Surprisingly, the existing theory of database constraints has only addressed the idealized special case where database instances are relations. In practice, e.g., SQL databases, duplicate tuples and partial information are permitted to occur. In this article, a complete theory has been established for reasoning about an expressive class of database constraints over partial relations, bags and partial bags. This closes the gap between theory and practice.

In future work it would be worthwhile to study many application areas of database constraints, including database normalization, query optimization, and data cleaning to name a few. One may also study the implication problem over different data structures, including trees and graphs, and consider other classes of data dependencies, e.g., join and inclusion dependencies. It is still an open problem whether the implication problem of MVDs can be decided in linear time. Finally, the study of Armstrong instances over different data structures may also be rewarding [33], [34], [35].

## References

[1] E. F. Codd, “A relational model of data for large shared data banks,” *Commun. ACM*, vol. 13, no. 6, pp. 377–387, 1970.

[2] E. Börger, E. Grädel, and Y. Gurevich, *The classical decision problem*. Heidelberg, Germany: Springer, 1997.

[3] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of Databases*. Addison-Wesley, 1995.

[4] M. Arenas and L. Libkin, “An information-theoretic approach to normal forms for relational and XML data,” *J. ACM*, vol. 52, no. 2, pp. 246–283, 2005.

[5] A. Deutsch, L. Popa, and V. Tannen, “Query reformulation with constraints,” *SIGMOD Record*, vol. 35, no. 1, pp. 65–73, 2006.

[6] J. Biskup, *Security in computing systems*. Heidelberg, Germany: Springer, 2009.

[7] A. Klug and R. Price, “Determining view dependencies using tableaux,” *ACM Trans. Database Syst.*, vol. 7, no. 3, pp. 361–380, 1982.

[8] W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis, “Conditional functional dependencies for capturing data inconsistencies,” *ACM Trans. Database Syst.*, vol. 33, no. 2, 2008.

[9] A. Cali, D. Calvanese, G. De Giacomo, and M. Lenzerini, “Data integration under integrity constraints,” *Inf. Syst.*, vol. 29, no. 2, pp. 147–163, 2004.

[10] R. Fagin, P. Kolaitis, R. Miller, and L. Popa, “Data exchange: semantics and query answering,” *Theor. Comput. Sci.*, vol. 336, no. 1, pp. 89–124, 2005.

[11] C. Delobel and M. Adiba, *Relational database systems*. North Holland, 1985.

[12] M. Wu, “The practical need for fourth normal form,” in *ACM SIGCSE Conference*, 1992, pp. 19–23.

[13] C. Date and H. Darwen, *A guide to the SQL standard*. Reading, MA, USA: Addison-Wesley Professional, 1997.

[14] P. Atzeni and N. Morfuni, “Functional dependencies and constraints on null values in database relations,” *Information and Control*, vol. 70, no. 1, pp. 1–31, 1986.

[15] E. Lien, “On the equivalence of database models,” *J. ACM*, vol. 29, no. 2, pp. 333–362, 1982.

[16] C. Zaniolo, “Database relations with null values,” *J. Comput. Syst. Sci.*, vol. 28, no. 1, pp. 142–166, 1984.

[17] S. Y. Petrov, “Finite axiomatization of languages for representation of system properties: Axiomatization of dependencies,” *Information Sciences*, vol. 47, pp. 339–372, 1989.

[18] B. Thalheim, *Dependencies in relational databases*. Teubner, 1991.

[19] C. Beeri, R. Fagin, and J. H. Howard, “A complete axiomatization for fds and mvds in database relations,” in *SIGMOD Conference*. ACM, 1977, pp. 47–61.

[20] S. Hartmann and S. Link, “On a problem of Fagin concerning multivalued dependencies in relational databases,” *Theor. Comput. Sci.*, vol. 353, no. 1-3, pp. 53–62, 2006.

[21] C. Zaniolo, “Mixed transitivity for functional and multivalued dependencies in database relations,” *Inf. Process. Lett.*, vol. 10, no. 1, pp. 32–34, 1980.

[22] Z. Galil, “An almost linear-time algorithm for computing a dependency basis in a relational database,” *J. ACM*, vol. 29, no. 1, pp. 96–102, 1982.

[23] T. Imielinski and W. Lipski Jr., “Incomplete information in relational databases,” *J. ACM*, vol. 31, no. 4, pp. 761–791, 1984.

[24] S. Link, “On the implication of multivalued dependencies in partial database relations,” *Int. J. Found. Comput. Sci.*, vol. 19, no. 3, pp. 691–715, 2008.

[25] H. Köhler and S. Link, “Armstrong axioms and Boyce-Codd-Heath normal form under bag semantics,” *Inf. Process. Lett.*, vol. 110, no. 16, pp. 717–724, 2010.

[26] S. Hartmann and S. Link, “When data dependencies over SQL tables meet the Logics of Paradox and $S$-3,” in *Proceedings of the 29th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PoDS)*, 2010, pp. 317–326.

[27] M. Arenas and L. Libkin, “A normal form for XML documents,” *ACM Trans. Database Syst.*, vol. 29, no. 1, pp. 195–232, 2004.

[28] S. Hartmann, S. Link, and K.-D. Schewe, “Weak functional dependencies in higher-order data models,” in *FoIKS Conference*, 2004, pp. 134–154.

[29] S. Hartmann and S. Link, “Efficient reasoning about a robust XML key fragment,” *ACM Trans. Database Syst.*, vol. 34, no. 2, 2009.

[30] ——, “Numerical constraints on XML data,” *Inf. Comput.*, vol. 208, no. 5, pp. 521–544, 2010.

[31] M. Vincent, J. Liu, and C. Liu, “Strong FDs and their application to normal forms in XML,” *ACM Trans. Database Syst.*, vol. 29, no. 3, pp. 445–462, 2004.

[32] R. Fagin, “Multivalued dependencies and a new normal form for relational databases,” *ACM Trans. Database Syst.*, vol. 2, no. 3, pp. 262–278, 1977.

[33] ——, “Armstrong databases,” IBM Research Laboratory, San Jose, California, USA, Tech. Rep. RJ3440(40926), 1982.

[34] S. Hartmann, M. Kirchberg, and S. Link, “Design by example for SQL table definitions with functional dependencies,” *The VLDB Journal*, doi: `10.1007/ s00778-011-0239-5`, 2012.

[35] S. Hartmann, U. Leck, and S. Link, “On Codd families of keys over incomplete relations,” *The Computer Journal*, vol. 54, no. 7, pp. 1166–1180, 2011.

# Distinguishing Soft-Goals and Quality Requirements in Software Requirements Modeling

Thi-Thuy-Hang HOANG
Louvain School of Management (LSM),
Université catholique de Louvain,
Louvain-la-Neuve, Belgium.
E-mail: thi.hoang@uclouvain.be

Alain PIROTTE
Louvain School of Management (LSM), and
Information Technologies, Electronics and Applied Mathematics (ICTEAM)
Université catholique de Louvain,
Louvain-la-Neuve, Belgium.
E-mail: alain.pirotte@uclouvain.be

*Abstract* - **Requirements engineering plays an essential role in software development. Requirements are prescriptive statements that express situations to be enforced by a system in terms of its effects on its environment. There have been many discussions of functional versus non-functional requirements, of hard-goals versus soft-goals, of non-functional and quality requirements versus soft-goals. Quality requirements have often been treated as special cases of non-functional requirements or of soft-goals, without a clearly convincing distinction. In this paper, we formulate a somewhat unusual definition and metamodel for "quality requirements" and we explore some of its consequences. Quality requirements are not defined as a special kind of soft-goals, but as constraints on goals. We adapt the usual techniques of goal refinement to our definition and we argue that thus distinguishing soft-goals and quality requirements contributes to clarifying the system development process and the management of quality by the resulting software products.**

***Keywords - requirements models, data modeling, soft-goals, quality requirements, non-functional requirements***

## I. REQUIREMENTS, GOALS, AND QUALITIES

Requirements engineering plays an essential role in software development. Requirements are prescriptive statements to be enforced by a new software system or by a revised version of an already existing system, possibly in cooperation with other system components, and formulated in terms of its environment [10]. Requirements contribute to a global model of the system in its early stages of development or revision. Then they help construct a specification for system design.

Requirements are identified and captured from exchanges of information between analysts and stakeholders of the future or revised system. Requirement analysts play the role of architects who progressively transform informal needs of stakeholders into a precise and possibly formal model. Collecting and understanding stakeholder informal descriptions and performing the transformations from informal to precise models are hard creative tasks. Thus, requirements engineering remains a delicate phase of software development as widely documented in the literature (see, e.g., [10,11,13]).

Requirements that concern functions and services of the future or revised system (i.e., questions about what the system should do) are called functional requirements. Requirements that concern how well (e.g., quality aspects of how speedily (performance), how cheaply (costs), how accurately, etc.) the system should provide its functions are typically called non-functional requirements. Non-functional requirements are often also presented as quality requirements in the literature without a clear distinction between them.

A more modern view than the distinction between functional and non-functional requirements presents the system as responsible to bring about desired states of the world specified as goals. Goals have been typically classified into hard-goals and soft-goals depending on the precision of their satisfaction criteria. Hard-goals have satisfaction criteria that are clearly defined. Like functional requirements, they concern functionality. Soft-goals are goals whose satisfaction criteria could not been defined in a clear-cut manner when they were formulated or whose satisfaction can be subjective. They may be judged as satisfied or unsatisfied to different degrees by different people and at different stages of system development. The intuition about soft-goals can be conveyed by the following examples that we further discuss later: "all banking transactions must be handled in a secure manner" or "online banking transactions must be offered with high availability". For some reasons, not enough information was available when they were formulated as requirements to precisely assess their satisfaction criteria.

There have been many analyses of requirements in terms of functional versus non-functional requirements, of hard-goals versus soft-goals, of non-functional and quality requirements versus soft-goals (see, e.g., [5,7], that clearly illustrate the difficulties involved). There are similarities between non-functional requirements and soft-goals in their typical characteristics of defining satisfaction criteria in an imprecise manner. To argue for a difference between soft-goals and non-functional requirements, it has been suggested that non-functional requirements define constraints but not system functions, while soft-goals characterize system states and thus can describe directly or indirectly system functions. But this is more intuition than an exploitable definition. In multi-agent systems, where goals are used extensively for modeling requirements, non-functional requirements are often considered as a subset of soft-goals.

Much thinking has been devoted specifically to software quality. The coexistence of several approaches to defining software quality has been well summarized in a recent book [10]. It is argued that there are three principal ways to view software quality: (1) conformance to requirements; (2) reliability, portability and other -ilities; and (3) absence of defects. The third approach privileges quantitative methods for defect detection and removal using quality metrics. For [10], qualities like the -ilities are not practical because they are too vague (e.g., survivability) and most of them are irrelevant for users (like, e.g., portability). Problems with the first approach can arise if some requirements are badly selected, i.e., "toxic", missing or excess requirements, that unintentionally cause requirement-compliant products to go wrong. Our approach is of course more related to the first definition (quality as conformance to requirements).

This paper does not aim at comparing various definitions. In particular, many approaches to quality attributes regard them as useful only when they can be measured quantitatively. We do not discuss either the nuances between non-functional requirements and quality requirements. Instead, we present our own definition and metamodel of quality requirements by which they are different from soft-goals, and they constrain hard-goals and soft-goals. We explore some interesting consequences of that definition by adapting the usual techniques for goal analysis and refinement so that they can take advantage of this additional dimension. This paper substantially revises the core ideas of [8] after the completion of the first author's thesis [9].

Broadly, model-driven engineering allows software developers to focus on concepts more abstract than those directly supported by implementation platforms. The approach has contributed (1) languages (or metamodels) to describe various aspects of systems and (2) correspondences between metamodels, and the operationalization of such mappings as techniques and tools to help system development tasks. Model-driven system development is thus conducted by building and composing models, and by transforming them progressively into operational systems.

This paper essentially deals with a proposal for a new metamodel for quality requirements and soft-goals. It also discusses the correspondence of goal-refinement graphs expressed in our metamodel with other representations of goals. Section 2 of the paper motivates our approach to quality requirements. Section 3 gives precise definitions. Section 4 presents the integration of our approach into an adapted strategy for goal analysis and refinement. Section 5 expands on fulfilling quality requirements.

## II. SEPARATING QUALITY REQUIREMENTS AND SOFT-GOALS

The intuition of our approach can be grasped through the following examples of soft-goal requirements mentioned in Section 1: "all banking transactions must be handled in a secure manner" and "online banking transactions must be offered with high availability".

Qualifying terms, like "in a secure manner" or "with high availability", represent qualities (of security or of availability) that are embedded inside the soft-goal statements. If those qualifying terms are "taken out" of the soft-goals, then we argue that the same situation can be described by a hard-goal (for example, "all banking transactions must be treated") constrained by what we will call a quality requirement (for example, "in a secure manner", i.e., with some quality of security).

General issues arising from this discussion include the following: (1) to what extent can quality requirements be extracted from soft-goals? (2) how can their relationships be adequately expressed in a precise metamodel? (3) when they can, how does the separation benefit software development, especially how does it improve and simplify the fulfillment of requirements in both functional and quality aspects? (4) which tools could help software developers? Although the examples above suggest that the answer to the first question is affirmative, these issues deserve more analysis. This paper addresses the first and second issues and, briefly, the third one. Some tools are proposed in [9].

## III. OUR DEFINITION OF QUALITY REQUIREMENTS

We first formulate our basic definition of quality requirement (together with definitions of hard-goal and soft-goal), where the novelty essentially lies in the fact that quality requirements are defined as constraints on goals and not as non-functional requirements. Then, after some comments and examples, we complement the basic definition by definitions of the links between goals and quality requirements that express their dependencies.

A **goal** describes a state that the system-to-be should be able to bring about. A **hard-goal** is a goal for which satisfaction criteria can be precisely defined. A **soft-goal** is a goal for which satisfaction criteria are not defined in a clear-cut way. A **quality requirement** describes a constraint whose satisfaction or fulfillment ranges on a scale of possibilities and that can constrain a goal.

Degrees of satisfaction or of fulfillment of quality requirements are typically expressed as abstract levels that range from "not satisfied" to "fully satisfied" (e.g., "high", "low", "average", "cheap", "expensive", "affordable" etc.).

The following examples illustrate the proposed definitions. "Payment sent" is a hard-goal since it describes a well-defined state (of having or not having been sent) leading to defining payment functionalities. "Money transferred with high security" is a soft-goal since the level of security is not precisely stated. "Web pages served with high availability" is also a soft-goal for similar reasons. In those examples, "with high security" and "with high availability" are quality requirements.

Concerning the choice of terms, using "quality requirement" for a constraint of a different nature than a non-functional requirement may not be the best idea. Just saying "quality" would not be a better choice because the term is overloaded. We could maybe have chosen "quality constraint". We stayed with "quality requirement" and we tried to be as clear as possible throughout the text to avoid ambiguities.

A reason why it is difficult to define satisfaction criteria for some soft-goals is the presence of quality requirements tightly integrated "inside the soft-goal", like, e.g., "message confidentially sent" or "web pages served with high availability". Such soft-goals can be called "quality soft-goals". They typically appear in early stages of requirements formulation. Quality requirements can be elicited (i.e., extracted) from them, like "confidentiality" from "message confidentially sent". Such a complex soft-goal can be represented as a quality requirement "high availability" that constrains a hard-goal "web pages served". Similarly, the soft-goal "money transferred with high security" can be understood as the combination of the hard-goal "money transferred" and of the quality requirement "high security".

Other soft-goals are not so explicitly linked with a quality requirement, like, e.g., "Increase Sales" or "Make Customers Happier". The goal refinement process described in Section 4 transforms such soft-goals into other goals (eventually into hard-goals) that will make qualities more visible. For example, a strategy to "Make Customers Happier" could involve sub-soft-goals like "Provide Reliable and High-speed Connections", which could result in the hard-goal "Provide Connections" constrained by quality requirements "Reliability" and "High-speed".

Our full metamodel of quality requirement combines the basic definitions above with the following definition of **links between soft-goals and quality requirements**: some soft-goals can be viewed as the combination of a goal (hard-goal or soft-goal) and a quality requirement; such soft-goals can be alternatively re-expressed as a combination of the derived goal constrained by the derived quality requirement. We call "qualification link" the link from a quality requirement to a constrained goal and "elicit link" the link from a soft-goal and a quality requirement extracted from it.

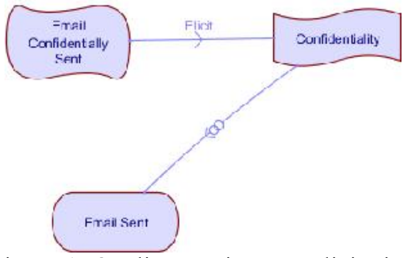


Figure 1: Quality requirement elicitation

Figure 1 shows an example of a soft-goal ("Email Confidentially Sent") re-expressed as a hard-goal ("Email Sent") constrained by a quality requirement ("Confidentiality") elicited from the soft-goal. We follow the familiar notations of [4] for goals: soft-goals are drawn as boxes with round-shaped vertical sides and wavy horizontal sides while hard-goals are drawn as boxes with straight horizontal sides and round-shaped vertical sides. We draw a quality requirement as a box with straight vertical sides and wavy horizontal sides. We draw a qualification link as an arrow joined with two small circles from a quality requirement to the goal that it constrains. We signal an elicit link by an arrow from the soft-goal to the elicited quality.

## IV. GOAL REFINEMENT WITH QUALITY REQUIREMENTS

Section 4.1 recalls the basic ideas of goal refinement as a strategy for software development. Section 4.2 describes a modified strategy taking into account the new node type of quality requirements and the new types of link: elicitation, qualification, and contribution (the latter still to be defined). Section 4.3 summarizes the objectives of goal refinement.

### *4.1 Goal refinement*

The main objective of goal-based requirements engineering, is to iteratively refine higher-level requirements until concrete system requirements are obtained. AND/OR decomposition graphs in KAOS [6,13] have become standard tools for such tasks. AND/OR decomposition is appropriate for hard-goals since they can be defined as logical conditions and their satisfaction can be checked by AND/OR relations between corresponding logical conditions of their sub-goals.

For soft-goals, additional weaker types of goal transformation are needed, as was suggested in the style of i* [1,4,12,14]. Derived soft-goals can contribute, negatively or positively, to fulfilling parent soft-goals. Such a contribution is made explicit by so-called contribution links that can take, in [4], one of five contribution levels: '++' (make), '+' (help), '?' (unknown), '-' (hurt) and '--' (break) with an obvious intuition. The idea is that such contributions links will allow application specialists to determine, for each candidate choice of system functions to implement lower-level goals, the satisfaction level of each top-level soft-goal, given the AND/OR contributions, and the satisfaction level of the derived goals and quality requirements.

### *4.2 Revised goal analysis*

This section suggests an extended goal analysis process to accommodate quality requirements. Our definitions suggest to add quality requirement nodes and three new additional links to the usual AND/OR goal analysis. As introduced in Section 3, "elicitation" links describe how a quality requirement is extracted or inferred from a soft-goal in which it was "packaged" during the analysis of early requirements. Qualification" links relate quality requirements to goals on which they bear. "Contribution" links are used to describe how hard-goals can contribute (negatively or positively) to the satisfaction of quality requirements that constrain them. Quality requirements can themselves be refined into subqualities and they can propagate down in the goal decomposition tree.

#### *4.2.1 OR decomposition*

OR decompositions make explicit some alternatives to fulfill a goal in the goal-decomposition tree. In Figure 2, the goal "Invitation Sent" can be satisfied either by the goal "Invitation Sent By Email", or by the goal "Invitation Sent by Post", or by the goal "Invitation Communicated By Telephone". Since the "Promptness" quality is required of the parent goal, it is also required of all the alternatives.

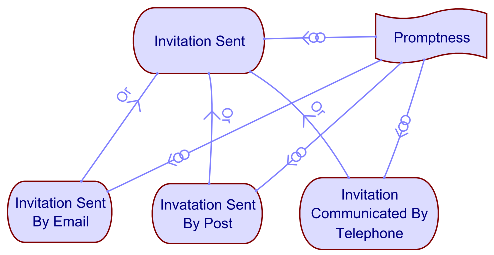
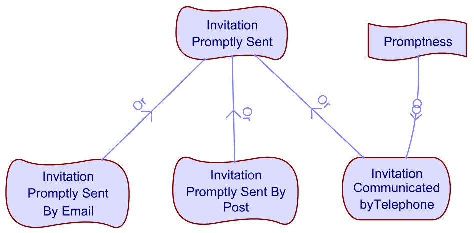


Figure 2: OR decomposition with quality requirement

This example can be analyzed differently with "Invitation Promptly Sent" packaged as a soft-goal. Figure 3 shows a version of the example where the "Promptness" quality requirement has been extracted not from the initial soft-goal "Invitation Promptly Sent", as Figure 2 suggests, but from its OR-subgoals "Invitation Promptly Communicated By Telephone". Of course, these are just illustrative examples and further analysis of the solution in Figure 3 may well evolve into that of Figure 2.


Invitation Promptly Sent
Promptness
Or
Or
Or
Invitation Promptly Sent By Email
Invitation Promptly Sent By Post
Invitation Communicated byTelephone
Figure 3: OR decomposition of Soft-goal

### *4.2.2 AND decomposition*

With AND decomposition, a goal is satisfied if and only if its subnodes are satisfied. In Figure 4, the goal "Music Played" with quality "Legality" is satisfied if a relevant file is found and downloaded legally. Then the downloaded music file is opened to send sound to speakers. Hard-goal "Music File Opened" is not concerned with the "Legality" quality requirement since "Legality" can be completely satisfied by the other two hard-goals.

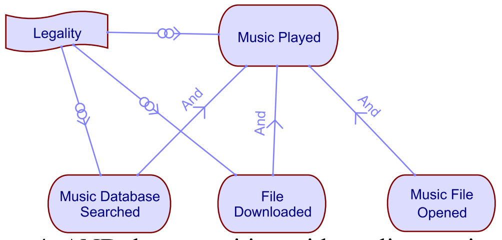


Figure 4: AND decomposition with quality requirement

Qualities can be decomposed into sub-qualities as suggested in [4]. Sub-qualities must be appropriately applied to sub-goals of an AND-decomposition, as illustrated in Figure 5. Quality "Economy" is decomposed into sub-qualities "Efficiency" and "Reusability". To have "Software Developed" with "Economy" (for the sake of the example), it must (i) be designed with "Reusability" in mind, and (ii) be provided with an efficient bug management system, supposing that the coding activities do not affect the overall development cost. This example illustrates the fact that the decomposition of quality requirements can be generic at higher levels of the analysis graph and application-dependent at more concrete lower levels.

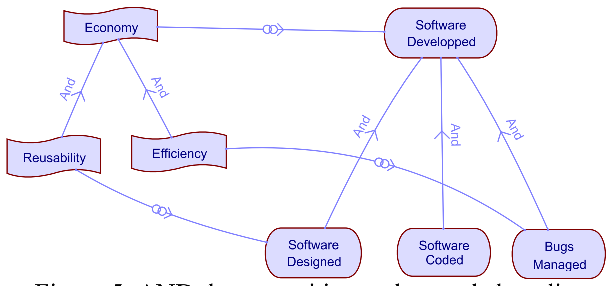


Figure 5: AND decomposition and extended quality requirement

### *4.2.3 Contribution links and quality requirements*

Quality requirements are propagated through goal analysis and refinement. Contribution links can be used in the process to help take quality requirements into account.

Contribution links were introduced to help decompose soft-goals [4]. In i*, they are used only to decompose soft-goals. In our approach, they can also be used with quality requirements. The most common usage is to show how satisfying a hard-goal can contribute to satisfying a quality requirement that the hard-goal is required to fulfill.

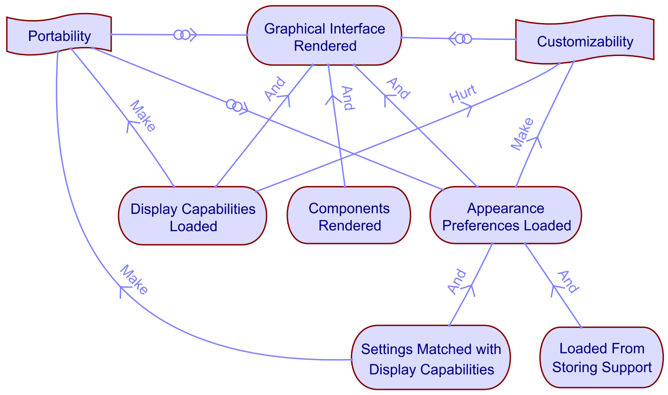

Portability
Graphical Interface Rendered
Customizability
Make
And
And
And
Hurt
Make
Display Capabilities Loaded
Components Rendered
Appearance Preferences Loaded
Make
And
And
Settings Matched with Display Capabilities
Loaded From Storing Support
Figure 6: Contribution links and quality fulfillment

Figure 6 illustrates such a situation: a decomposition of hard-goal "Graphical Interface Rendered" constrained by quality requirements "Portability" and "Customizability". Hard-goal "Components Rendered" expresses that rendering the complete interface can be split into rendering its components. For quality"Portability" (to several interfaces like PC monitors, smartphones, etc.), hard-goal "Display Capabilities Loaded" requires that sufficient capabilities for those interfaces be made available. For quality "Customizability", hard-goal "Appearance Preferences Loaded" requires that the rendering system take into account existing user preferences for display settings (font face, color, font size, etc.). Contribution links of type "Make" (see Section 4.1) from the latter two hard-goals to their respective quality requirement make those contributions explicit.

In Figure 6, too much emphasis on the display capabilities of a specific platform may hamper the

"Customizability" of the overall platform, in the sense of making things more difficult for a device with insufficiently supported settings, a situation made explicit with a "Hurt" contribution link from "Display Capabilities Loaded" to "Customizability". The situation can be improved by further constraining the "Appearance Preferences Loaded" hard-goal with the "Portability" requirement. This new qualification link suggests the decomposition of "Appearance Preferences Loaded" into hard-goals "Settings Matched with Display Capabilities" and "Loaded from Storing Support". The idea is that "Settings Matched with Display Capabilities" will limit user settings to available capabilities and thus positively contribute to the "Portability" quality requirement. The analysis of the example should still be refined to further clarify and somehow sort the conflict between the contributions to quality requirements.

### 4.3 *The objectives of goal analysis and refinement*

The inputs to the qualitative process of goal refinement are a set of hard-goals expressing the important functional requirements for the system and a set of soft-goals in which quality requirements are more or less explicitly embedded.

The main objectives of goal analysis as presented here are: (1) to accompany developers and stakeholders when identifying and clarifying relevant requirements; (2) to organize the exploration and evaluation of alternative detailed requirements for the new or revised system; (3) to progressively make quality requirements explicit and to propagate the responsibility of satisfying them downwards to suitable goals of the analysis graph; (4) to "operationalize" those lower-level qualified hard-goals whenever possible, that is, to relate them to operations to be made available by the future system and that will ensure their best satisfaction.

Such system operations (or services or functions) invoked at the leaves of the goal analysis graph are intuitively similar to UML use cases, that is, specifications of operations that are expected to be provided by the system.

For example, a qualified goal "Message Sent" constrained by the quality requirement "High Confidentiality" could be adequately fulfilled by a system operation like "Send Encrypted Message".

## V. FULFILLMENT OF QUALITY REQUIREMENTS

Figure 7 illustrates our techniques with a more substantial example. Soft-goal "Payment Immediately and Securely Sent" for an online shop is progressively transformed with elicitation links to extract quality requirements, qualification links to constrain derived goals, and contribution links to express contributions to the fulfillment of quality requirements and to the selection of goal decomposition alternatives.

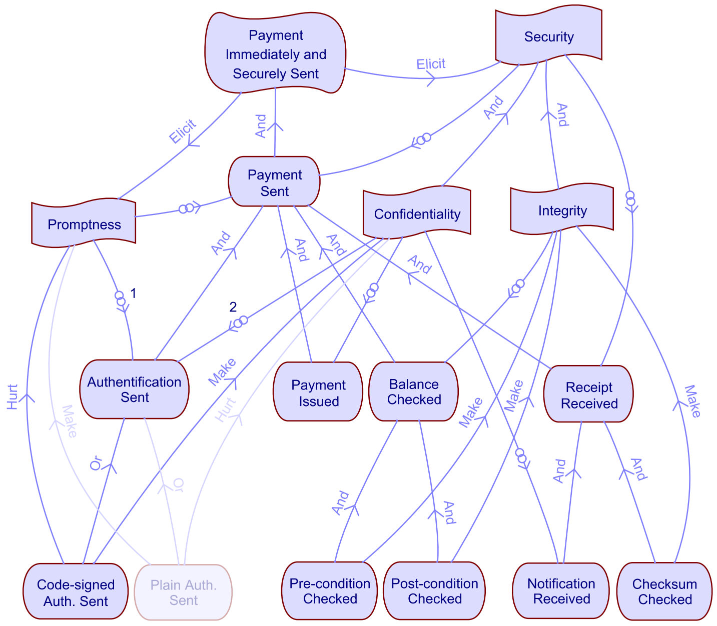


Figure 7: Immediately and Securely Sent soft-goal

Soft-goal "Payment Immediately and Securely Sent" is re-expressed as hard-goal "Payment Sent" constrained by elicited quality requirements "Security" and "Promptness". "Security" is AND-decomposed into "Confidentiality" and "Integrity" quality requirements. Hard-goal "Payment Sent" is AND-decomposed into hard-goals "Authentication Sent", "Payment Issued", "Balance Checked", and "Receipt Received". Hard-goal "Balance Checked" is AND-decomposed into hard-goals "Pre-condition Checked" and "Post-condition Checked". Hard-goal "Receipt Received" is AND-decomposed into hard-goals "Notification Received" and "Checksum Checked".

Hard-goal "Authentication Sent" is given two alternative subgoals: "Code-signed Auth. Sent" and "Plain Auth. Sent". The first one contributes positively ("Make") to "Confidentiality" and negatively ("Hurt") to "Promptness" as it requires user intervention for authentication, while the reverse holds for the second alternative. One way to solve the conflict, as suggested in Figure 7, is to assign a higher priority (say, value 2) to "Confidentiality" than to "Promptness" (value 1) thus privileging one of the subgoals.

It is interesting to compare the bottom part of Figure 7 to similar goal and soft-goal integration in the approach of [4]. There, soft-goals are mostly quality requirements; hard-goals and quality requirements are analyzed separately and correlated late in the analysis process. Our approach allows to analyze hard-goals, soft-goals, and quality requirements in an integrated scheme from the top-most and most abstract goals. If elicit and qualification links are removed from our analysis, then it becomes similar to the goal refinement of [4]. The price to pay for this simplification is the suppression of the traceability of quality requirements, analysis rationale, and late operationalization of quality requirements.

Thus, compared to modeling approaches relying on just hard-goals and soft-goals, our approach is semantically richer, in line with a model-driven emphasis. Quality requirements are decoupled, as analysis and refinement proceed, from the functional part thanks to the additional links (elicitation, qualification, and contribution links).

The possible outcomes for an analysis graph is a list of alternative AND-sequences of leaf nodes together with the quality requirements whose satisfaction is still undecided. In Figure 7, there are two possible sequences of leaf hard-goals, each corresponding to one of the alternatives for handling the "Authentication Sent" hard-goal as described above. If "Confidentiality" is privileged as suggested, the sequence of leaf hard-goals is: "Code-signed Auth. Sent", "Payment Issued", "Pre-condition Checked", "Post-condition Checked", "Notification Received", "Checksum Checked", where two hard-goals ("Payment Issued" and "Notification Received") are still constrained by the "Confidentiality" requirement. Ensuring this quality requirement relies on third-party payment services. If not enough information is available, then handling it can be deferred until enough additional information becomes available.

In some cases, ensuring an acceptable degree of fulfillment must be postponed to run time and programmed with adhoc solutions. Such a late operationalization is typically necessary when the actual load of the system cannot be estimated with sufficient precision at the analysis stage. An example is the estimation by a service provider of an adequate frequency of checking the queue length of waiting customers for addressing the quality requirement of "High Availability". A set of social design patterns to help address such programming tasks is proposed in [9].

## VI. Summary and Conclusions

This paper formulates a new definition and metamodel of quality requirements treated not as special cases of non-functional requirements or of soft-goals, but as constraints on goals, and it explores some of consequences of that definition. Three important aspects of our approach and its effect on goal refinement can be summarized as follows. First, quality requirements are typically embedded (i.e., implicit) within high-level soft-goals at the start of the analysis process. They are made progressively explicit during goal refinement. Second, with elicitation and qualification links, all soft-goals can be, earlier or later, re-expressed as a combination of lower-level hard-goals constrained by quality requirements. Third, quality requirements appear explicitly throughout the goal analysis process. As a result, quality requirements are more decoupled from the functional part, which enables a more flexible treatment of them.

The paper also proposes extensions of the usual process of goal refinement of software development by defining revised transformation of goal graphs in order to accommodate the presence of quality requirements that constrain goals and soft-goals.

Our approach was applied in [9] to enhancing the Tropos methodology [3,14] as QTropos (quality-aware Tropos) where quality requirements are independent notions constraining dependencies. Quality requirements can thus be taken into account during all phases of QTropos (early requirements, late requirements, architectural design, and detailed design). Also described in [9] are QCase, a tool to help apply our metamodel to the development of realistic projects, and a case study to help validation. Subsequent work will strengthen QCase with code generation and broaden its applicability to concepts from other development methodologies.

More perspectives for subsequent work include refining our metamodel at the light of substantial case studies and of further critical comparisons with other languages and methods, e.g., [1,2,12], and the abundant literature on quality in requirement engineering and model-driven development. Clearly, our approach to quality as constraints on goals enhancing conformance to requirements is just one possible approach among many. More work can be done on assessing more finely the locus and strength of its relevance.

## References

[1] D. Amyot, J. Horkoff, D. Gross, and G. Mussbacher. A Lightweight GRL Profile for i* Modeling, in Advances in Conceptual Modeling – Challenging Perspectives, Springer LNCS 5838, pp. 254-264, 2009.

[2] C. Ayala, C. Cares, J. Carvallo, G. Grau, M. Haya, G. Salazar, X; Franch, E. Mayol, C; Quer, A Comparative Analysis of i*-Based Agent-Oriented Modeling Languages, Procs 17th Int. Conf. on Software Engineering and Knowledge Engineering (SEKE'05), pp. 43-50, 2005.

[3] J. Castro, M. Kolp, and J. Mylopoulos. Towards Requirements-Driven Information System Engineering: the Tropos Project, Information Systems 27(6), pp. 365-389, 2002.

[4] L. Chung, A. Nixon, E. Yu, and J. Mylopoulos. Non-Functional Requirements in Software Engineering, Kluwer Academic Publishers, 2000.

[5] L. Chung and J.C.P. Leite, 2009. On Non-Functional Requirements in Software Engineering. Conceptual Modeling: Foundations and Applications – Springer LNCS 5600, pp. 363-379.

[6] R. Darimont and A. Van Lamsweerde. Formal Refinement Patterns for Goal-Driven Requirements Elaboration, ACM Sigsoft Notes, 21(6), pp. 179-190, 1996.

[7] M. Glinz. On Non-Functional Requirements. In: Proceedings of the 15th IEEE International Requirements Engineering Conference. IEEE, pp. 21-26, 2007.

[8] T. T.H. Hoang and M. Kolp. Goal, Soft-goal and Quality Requirement. In: Procs 12th ICEIS (Int. Conf. on Enterprise Information Systems)., pp. 11-22, 2010.

[9] T.T.H. Hoang. Quality-Aware Agent-Oriented Information-System Development. Ph. D. Thesis, Louvain, School of Management, Louvain-la-Neuve, Belgium, 2010. Available at http://www.isys.ucl.ac.be/ThesisHoangTTH.pdf

[10] C. Jones. Software Engineering Best Practices. McGraw-Hill, New York, 2009.

[11] R. Ramsin and R. Paige. Process-Centered Review of Object-Oriented Software Development Methodologies, ACM Computing Surveys, 40(1), pp. 1-89, 2008.

[12] 12. URN (User requirement Notation), Language Definition: Recommendation Z.151, International Telecommunication Union (ITU), Geneva, Switzerland, November 2008.

[13] A. van Lamsweerde. Requirements Engineering: From System Goals to UML Models to Software Specifications, Wiley, 2009.

[14] E. Yu. Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering, Proc. 3rd IEEE Symposium on Requirements Engineering, pp. 226-235, 1997.

# A Case for Inverted Indices in In-Memory Databases

Jens Krueger, Johannes Wust, Martin Faust, Tim Berning, Hasso Plattner
Hasso Plattner Institute for Software Engineering
University of Potsdam
Potsdam, Germany
Email: {jens.krueger@hpi.uni-potsdam.de, johannes.wust@hpi.uni-potsdam.de,
martin.faust@hpi.uni-potsdam.de, tim.berning@hpi.uni-potsdam.de, hasso.plattner@hpi.uni-potsdam.de}

***Abstract*—Recent database research has focused on in-memory databases, which can be used in mixed workload scenarios, enabling OLTP and OLAP queries on the same database engine. These compressed column-oriented database systems use a differential store concept to enable fast inserting and require a merge process to compact the data periodically in a compressed main partition that is changed by the merge process only. This characteristic feature calls for a re-evaluation of the performance of inverted indices. In this paper we present a use case for an inverted index in a column-oriented in-memory database system to reduce the total costs of query processing in a mixed workload environment. We evaluate the benefits and drawbacks of using the index structure to answer queries and the costs of maintaining the index, especially during the merge process. An analytical model is introduced to compute the theoretical cost of index scans. Furthermore a comparison between different index maintenance strategies is presented. The theoretical findings are verified in a prototypic implementation within the HYRISE database system. Our contributions are an analytical framework to evaluate the benefit of an inverted index during query execution, the verification in an in-memory database system, and an evaluation of different index maintenance strategies. From the presented findings we conclude that indexing can be an efficient instrument to meet the performance requirements in a mixed-workload and main memory-based environment.**

***Keywords*-In-Memory Database; Inverted Index; Index Maintenance;**

## I. Introduction

Over the last years the price of main memory has dramatically declined to a level on which it becomes competitive to prices for hard disk storage of only few years ago. This trend allows for servers with several gigabytes or even terabytes of main memory at a relatively low cost. The vast amount of fast accessible memory exceeds by far the space needed by traditional software, which mostly derives from times where main memory was a resource to be handled carefully. Consequently, in-memory databases build on this phenomenon and use the available space to store the application itself as well as the contained data entirely in the main memory, thus eliminating costly lookups on slow hard disk drives when accessing data. Despite all data now being present in the main memory at any time, processing the data still requires it to be loaded into the processor's cache, an operation that - after eliminating expensive reads from hard drives - represents the system's new bottleneck [1]. Due to this fact, reducing memory accesses is crucial to the overall performance of the database. Reading an entire table consisting of several million rows for the sake of a few records' values is an obvious waste, which can be eliminated by a lookup structure called inverted index. While inverted indices have been used for decades in traditional disk-based database systems or text retrieval systems, there are differences when using inverted indices in an in-memory database, such as HYRISE [2], [3], which impact the way an index is leveraged and maintained. In case dictionary compression is applied on a column, the inverted index is used to speed up read performance since the applied dictionary compression facilitates the inverted index structure as the compression can be leveraged to build up a compressed index. The impact is especially high for queries with a low selectivity. As shown in Figure 1 the workload of enterprise applications consist of $\sim$50 percent lookup queries and $\sim$30 percent of range queries. Consequently, in mixed workload environments an index is needed to efficiently support these kinds of read access.

This paper gives an overview over the benefits and tradeoffs, which result from the employment of inverted indices in the context of in-memory databases. The remainder of the paper is structured as follows: First, we will introduce the peculiarities of the used in-memory database HYRISE and provide additional architectural information. Section II gives an overview on the related work on inverted indices and describes the differences regarding an inverted index implemented in an in-memory database. In Section III, the inverted index is explained and theoretically analyzed regarding advantages and disadvantages, as well as other aspects such as the profitability of a delta partition index and a unique approach to index maintenance. The theoretical findings of this section are then compared against the actual HYRISE implementation in Section IV. The paper concludes with a summary and outlook.

### A. Background

There are of course different implementation approaches to in-memory database systems. In this paper, we focus on HYRISE [2], that is an in-memory database, which facilitates a column-oriented data organization and dictionary compression with late materialization strategies during query execution [4].

The key features with impact on structure and performance of an inverted index encompass the following:

*a) Column-oriented Data Storage:* Typically there are two ways of storing the information contained in a table, either row-by-row or column-by-column. Which one to choose primarily depends on the expected workload. Sequentially accessing data on almost any storage medium is substantially faster than random access. It has been found that in many cases there are more analytical-style than transactional-style queries [5]. Typical analytical queries rarely access entire rows of a table but rather focus on the values of a small set of columns. Storing data in a columnar fashion saves the effort of reading unnecessary columns, thus reducing query execution time.

*b) Dictionary Compression:* Even if main memory is not a scarce resource with regards to the available size anymore, it still has to be used efficiently to reduce overall costs besides reducing the memory consumption only. The uncompressed size of a company's productive system may exceed even most-recent terabyte server setups. In HYRISE a column-based dictionary compression technique is used to reduce the memory footprint of each column. All distinct values are captured in a dictionary data structure (vector) that assigns each actual value of the column a unique value id, and provides bilateral translation. Now instead of the actual values in a column, each column consist of a dictionary vector and an attribute vector that stores the respective value ids. The larger those actual values and the smaller the number of distinct values, the more efficient the compression. For instance, applying dictionary compression to a column storing country names as strings: As the number of countries is naturally limited and in practice often narrows down to a handful that are actually used, storing that handful of strings once in the dictionary and using value ids in the column provides good compression rates.

Other lightweight compression techniques are not applied to HYRISE since fast tuple reconstruction is essential for the mixed workload environment HYRISE is build for.

*c) Differential Store Concept:* Due to the applied compression each column is separated into two parts, a primary, read-optimized, read-only part (referred to as "main partition") and a secondary, write-optimized part that serves as buffer for data altering operations (referred to as "delta partition") [3]. The actual state of a column is represented by a union of both partitions with a one-bit validity vector to handle record visibility in case of update and delete queries. Initially being empty, the delta partition gets filled by data modifying queries. Since the main partition is not altered, a number of optimizations regarding read performance can be applied while the delta partition focuses on insertion speed, thus the differential store copes efficiently with OLTP and OLAP style requests at the same time.

*d) Merge Process:* As stated, data changes occur only in the delta partition. Yet, main and delta partition form a unit and together represent the column, meaning that data retrieval of a column must respect both. Due to the write-optimized nature of the delta partition, it lacks the main partition's read performance and thus should not grow too large in size. Defining a threshold at which the delta partition becomes "too large" is not a trivial task and primarily depends on the individual workload and is discussed in [6]. However, once the delta partition size exceeds certain limits, a merge process is initiated. The merge process combines the main and delta partition to create a new main partition and a new empty delta partition. During this process the respective dictionary vectors are first merged into a new, sorted dictionary, secondly changes in the mapping of values onto value ids are applied to both main and delta partition content and last the contents are concatenated.

## *B. Enterprise Workload Characteristics*

In order to give a background on realistic workloads in enterprise applications, this section presents results on analyzing actual workloads.

The common assumption that enterprise applications work row based and with many updates has driven decades of database research. For example the TPC-C benchmark, which incorporates the characteristics of a transaction processing system, issues around 45% data modification operations and 55% read queries.

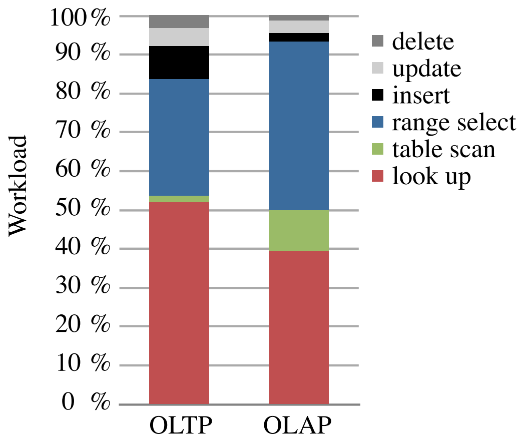


Figure 1. Analyzed Enterprise Workloads

However, our analyses of realistic database statistics derived from 65 customers have shown that more than 80% of all queries are read accesses — in OLAP systems even over 90% are read-only queries. Figure 1 shows the query distribution for transactional and analytical systems of key lookups, tables scans, range selects, inserts, updates and deletes. The analytical system differs in the distribution of inserts and table scans (which become column scans in column-oriented databases) with regards to the overall workload. While this is the expected result for OLAP systems, the high number of read queries on the transactional system is surprising. Consequently, the query distribution leads to the idea of using a read-optimized database for both transactional and analytical systems.

When implementing analytical functionality in a transactional system the support for fast tuple reconstruction based on key lookups still stays essential. At the same time, more

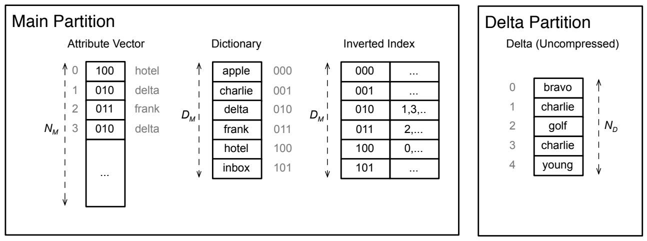


Figure 2. Inverted Index and Dictionary Encoding on a Single Column

complex queries with range requests and especially full column scans will increase. This trend will even further develop in the direction of more complex queries and full column scans as the direct reporting on transactional data enables new ”real-time analytics”. To summarize, highly selective queries are important in all workload variants and need to be supported efficiently. For this use case, indices have been introduced for any database system.

## II. Related Work

Inverted indices have been used for decades and were initially used to reduce the amount of data read from disks for the sake of throughput. Inverted indices in the context of in-memory databases face a similar problem domain which led to examinations of main memory inverted indices before actual in-memory databases were feasible. For instance, Lehman and Carey [7] already compared possible implementation techniques in 1986, at the same time predicting the price decline of main memory.

Most research about main memory inverted indices concentrates on either performance of the index itself or its maintenance. For example Rao and Ross [8], and Raatikka [9] evaluate several index data structures, such as T-trees, B-trees and their descendants, optimizing them for cache-consciousness.

Another approach is taken by Transier and Sanders in [10] who evaluate compression techniques for inverted indices used in text retrieval. The topic of maintaining inverted indices has also been covered, mainly in the context of text retrieval systems by facilitating the nature of certain index implementations [11].

Lester et al. [12] discuss the maintenance strategies in-place, re-build and re-merge, which are close to the deliberations about maintenance in this work.

Several research on inverted files has taken compression into account, such as in [13], [14], [15], and [16].

The purpose of this paper is not to optimize the index in terms of performance or size, but rather to investigate the use inside an in-memory database, such as HYRISE [2], [3].

## III. Inverted Index Structure

As stated before, considering an in-memory database I/O reduction is still the goal on main memory access. To eliminate the access of unnecessary data, the inverted index offers an inverse mapping of a column. As depicted in Figure 2, the inversion results in a map containing a list of record ids for every distinct value. When applying a predicate on a column the requested conditions have to be checked on every value of the column if no inverted index or sorting is in place. While this results in a sequential read access pattern, which is substantially faster than random access [1], still the complete column has to be read, creating a potential bottleneck. Compared to this, a lookup on the inverted index immediately offers the positional information about which records match the criteria and hence reduce the amount of data to be accessed. As described in Section I-A, HYRISE implements a column store without surrogate ids and therefore individual sorting of columns is prohibited. Furthermore, dictionary encoding is used as compression technique in HYRISE. Therefore, the inverted index can likewise leverage dictionary encoding to reduce memory footprint by mapping value ids instead of values to positions (see Figure 2). Since the dictionary for the attribute vector is already in place, there is no additional overhead for compressing the inverted index.

The core of the inverted index is a key-value-container with value ids as keys and record id lists as values. Since each column has presumably a unique dictionary, compound indices have to be implemented as higher-level functionality. Likewise, due to the distinction into a main partition and a delta partition with separate dictionaries on a single column, the inverted index cannot span an entire column but has to be implemented on the main and delta partition of the column respectively. The available options are to index the main partition only or to construct two inverted indices per column (one per partition). The remainder of this section examines benefits and tradeoffs accompanying the inverted index in theory, as well as whether or not to use an inverted index on the delta partition. For the index key-value-container an implementation of a balanced tree is assumed [17], guaranteeing at least $\mathcal{O}(logn)$ for search, insert and delete operations.

### A. Cost Model

The scans with and without an index are examined for their assumed costs in order to provide a common base for comparison. The model is based on the assumption that the bandwidth is the theoretical optimum and that cost can be expressed as number of operations on data. These theoretical operations will not directly map to CPU cycles, because different operations will consume a different amount of CPU cycles, also varying among hardware platforms. Our model is suitable to theoretically define the impact of using an index in a column store. As we show in Section IV, the model and the measurements align in key properties, such as break even points and general cost factors between storage schemes, but also show, that theoretical operations and CPU cycles are not directly interchangeable.

Table I summarizes the symbols that will turn up in calculations later on.

The term “query selectivity” refers to the fraction of a table’s records that will be contained in the query result set. As a shortcut $S_M$ equals $s * N_M$ and $S_M^d$ equals $s * d_M * N_M$

| Description | Unit | Symbol |
|---|---|---|
| Number of rows in table | - | $N$ |
| Table width, i.e. number of columns in table | - | $W$ |
| Number of rows in main partition | - | $N_M$ |
| Number of rows in delta partition | - | $N_D$ |
| Number of entries in main dictionary | - | $D_M$ |
| Number of entries in delta dictionary | - | $D_D$ |
| Fraction of unique values in main partition | % | $d_M$ |
| Fraction of unique values in delta partition | % | $d_D$ |
| Query selectivity | % | $s$ |
| Selected entries in main partition | - | $S_M$ |
| Selected entries in delta partition | - | $S_D$ |
| Selected distinct entries in main partition | - | $S_M^d$ |
| Selected distinct entries in delta partition | - | $S_D^d$ |
| Length of a value id | bytes | $L_{VID}$ |
| Length of a record id | bytes | $L_{RID}$ |
| Memory bandwidth | bytes/cycle | $B$ |
| Costs for operation $X$ | operations | $C_X$ |

Table I
SYMBOLS USED IN CALCULATIONS

for the main partition, while the same applies to $S_D$ and $S_D^d$ for the delta partition. Regarding the lengths of value ids and record ids an implementation as 32 bit integers is assumed, meaning that $L_{VID} = L_{RID} = 4$. Furthermore, all scans are assumed to be non-materializing, i.e. for the result no translation of value ids into values will be done [18]. In order to align the results as close as possible to our actual implementation running on a system as described in Section IV-A, we compute the memory bandwidth. 1066 MHz DDR3 memory in dual channel configuration features a theoretical throughput of 17.066 gigabytes per second. Adding the CPU with a base clock of 2.266 GHz and a turbo boost frequency of 2.666 GHz, averaged to 2.466 GHz or 2.466 million cycles per second, allow for the following calculation:

$$B = \frac{17.066\ GB/s}{2.466\ GHz} = \frac{17{,}066{,}000{,}000\ bytes/s}{2{,}466{,}000{,}000\ cycles/s} \approx 7\ bytes/cycle \quad (1)$$

### B. Performance Benefits

Performance benefits achieved by increased lookup speed are the key argument for inverted indices. Scanning an entire table requires $N_M + N_D$ memory accesses and comparison operations for a non-indexed column, resulting in a complexity of $\mathcal{O}(M + D)$. The actual number of memory accesses might be smaller because the data is read in chunks called cache lines [19] and the sequential processing reduces cache misses to a minimum and facilitates hardware based prefetching. Neither dictionary size nor the query's selectivity influence the performance of scanning the entire table, because each record is processed anyway. The total costs can be summarized as:

$$C'_{TableScan} = (N_M + N_D) * W * \frac{L_{VID}}{B} \quad (2)$$

This equation means that an entire row is loaded to validate a predicate on a single field's value of the row. A slightly improved variant runs a separate check on the column of interest to determine the desired rows and read them afterwards.

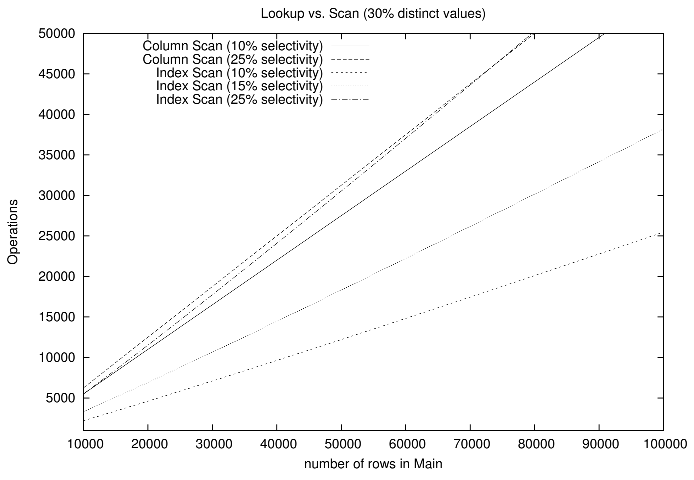


Figure 3. Column scan and index scan with varying main partition size

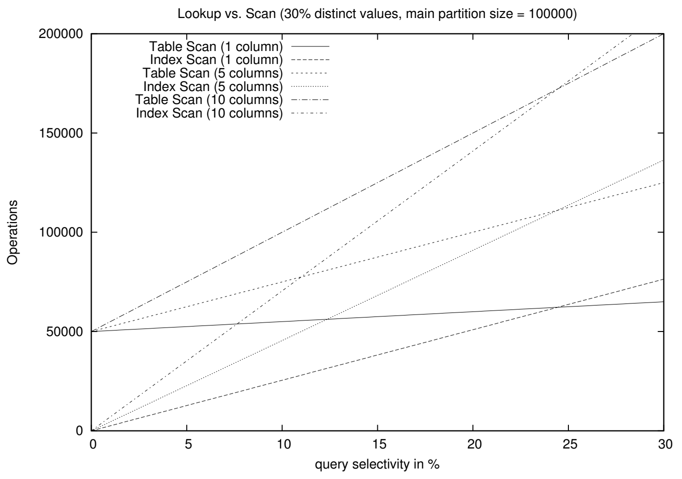


Figure 4. Column scan and index scan with varying selectivity

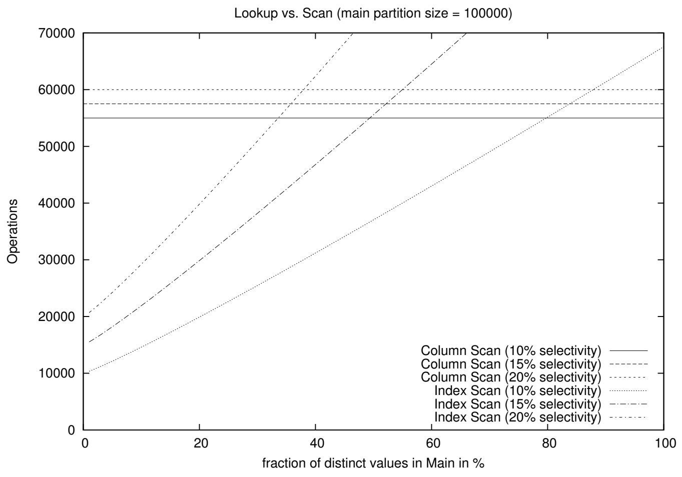


Figure 5. Column scan and index scan with varying fraction of distinct values

Therefore it suffers less from an increasing number of columns and will serve as the standard equation for a table scan in this Section and represents the characteristics of a column-oriented data storage. In this case of course query selectivity does have influence on the total costs:

$$C_{TableScan} = (N_M + N_D + (S_M + S_D) * W) * \frac{L_{VID}}{B} \quad (3)$$

Index-supported queries do not need to process the entire column first. While the delta partition is processed via a table scan as above, the positional information about the main partition is already at hand. Looking up the record id list for a given value id is in $\mathcal{O}(log\ D_M)$ as the number of keys in the index is equivalent to the number of distinct values in the column and therefore the number of dictionary entries. A value id lookup is necessary for each distinct value that meets the query conditions. The returned record id list contains $s * N_M$ entries and has to be read from memory. The actual $s*N_M*W$ can finally be read from the main partition by using the record id list. Data retrieval furthermore requires $N_D$ values to be read from the delta partition to get the positions, followed by reading the $s * N_D * W$ values, thus the overall costs can be calculated by:

$$\begin{aligned} C_{IndexScan} &= (S_M^d * log(D_M) + N_D) * \frac{L_{VID}}{B} \\ &+ S_M * \frac{L_{RID}}{B} + (S_M + S_D) * W * \frac{L_{VID}}{B} \end{aligned} \quad (4)$$

As obvious by this equation, the selectivity has a major impact on the inverted indexes profitability. Figure 3 illustrates this with an one-column table. Neither this nor the other figures referenced in this section feature a delta partition because it influences scan costs for table and index scan alike. At 25% selectivity for a main partition size of about 70,000 rows, the table scan becomes more economic. Since the cost equation for an index scan is logarithmic, the two lines for 15% and 10% cross the table scan line as well, which grows linearly.

A more obvious demonstration is given in Figure 4, including graphs for one, five and ten columns. In contrast to the aforementioned influence of the selectivity, the number of columns does not affect the efficiency, because it again applies to both scan variants. The only other component with impact is the number of distinct values in the column. Its influence is quite obvious since it correlates with the inverted index's size. Less unique values allow for faster lookups in the index as illustrated in Figure 5. An index scan with 10% selectivity on a table with one column and 100,000 rows equals a table scan in terms of cost at about 80% distinct values. The break even point value for $s$ can be calculated by equating $C_{TableScan}$ with $C_{IndexScan}$ (under the assumption that $L_{VID} = L_{RID}$) and reveals the following formula:

$$s_{breakeven} = \frac{1}{d_M * log(d_M * N_M) + 1} \quad (5)$$

Like observed in the graphs, the number of columns does not appear in the equation, neither does the delta partition as it is processed equally in both scan types. The only influencing factors are the number of distinct values and overall number of rows. If either row count or distinct value count increase, $s_{breakeven}$ logarithmically converges to 0, meaning that even when selecting just very few tuples an index scan is more costly than a table scan.

Of course this exact equation is only valid for a table with just one inverted index that can be used to resolve all query conditions. If this is not the case, either another inverted index has to be consulted or the resulting list of records has to be processed by a table scan. A second inverted index means an overhead of another $S_M^d * log(D_M) + S_M$ memory accesses. The returned record id list is intersected with the one from the first index before the respective rows are read from the table, keeping the performance impact low. Alternatively, the resulting rows from the index lookup are checked against all query predicates by using a column scan.

For a table scan the overhead is bigger, requiring another $N_M + N_D$ memory accesses for each column to be checked. If too many columns have to be checked, $C'_{TableScan}$ can become lower than $C_{TableScan}$. In summary the inverted index provides an excellent acceleration of query processing, as long as the query's selectivity does not exceed certain limits depending on table size and number of distinct values.

## C. Tradeoffs

Besides the advantage of increased lookup speed, having an inverted index has some downsides as well. These affect both most important factors in almost any application, time and space.

*1) Construction Costs:* Before it can be used, the inverted index must be built. As there is no positional information besides the column itself, a complete scan of the main partition is required. Processing row by row, the found value id is used to find the corresponding record id list or to create a new one if it did not exist. The find operation is in $\mathcal{O}(log\ D_M)$ and has to be executed $N_M$ times, resulting in a worst case construction cost of:

$$C(IndexConstruction) = N_M * log(d_M) * \frac{L_{VID}}{B} \quad (6)$$

Alongside the initial construction, maintenance costs arise when main and delta partitions are merged. The possibility of a change in the value id to value mapping renders the inverted index invalid and a rebuild is inevitable. Details about the inverted index maintenance including an alternative rebuild strategy are discussed in Section III-E.

*2) Memory Consumption:* The inverted index data structure uses space in memory. As said before, it contains one value id of size $L_{VID}$ for each distinct value in the column, mapping onto a list of record ids (each $L_{RID}$ in size). Consolidating all those lists yields $N_M$ entries for a main partition's index.

Summed up, the inverted index size for one column is:

$$D_M * L_{VID} + N_M * L_{RID} \ bytes \tag{7}$$

If the assumption applies that $L_{VID} = L_{RID}$, the inverted index consumes at least as much space as the underlying attribute vector of the column ($N_M * L_{VID}$ bytes). In case the level of distinct values reaches 100%, the index is effectively double the size of the column.

### *D. Delta Partition Index*

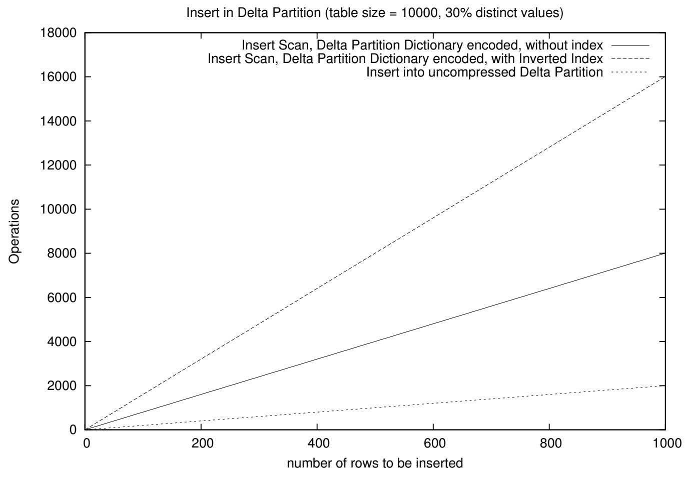


Figure 6. Insertion costs in delta partition.

An additional index on the delta partition can of course speed up lookups since the aforementioned inverted index on the main partition only leads to a retrieval of values from the delta partition with the help of a regular column scan. Unlike the main partition, the delta partition is constantly changed by insert, update and delete operations while the the employed insert only approach implements the logical update and delete queries as technical insert operations with timestamps. An index has to be always valid, therefore insertions into the delta partition entail an index update, which has to be part of the transaction.

In case of applied dictionary compression, an insertion is executed by retrieving the respective value id from the dictionary in $\mathcal{O}(log\ D_D)$, followed by a simple push back into the delta attribute vector in constant time. Having a delta index means another search in the indexes keys to find the appropriate bucket ($\mathcal{O}(log\ D_D)$) and inserting the new record id, which equals a regular insertion in terms of complexity and thus doubles insertion costs as shown in Figure 6. It also becomes obvious, that an uncompressed delta partition naturally offers the best insert performance.

Beneficial to an inverted index on the delta partition is the reduced lookup complexity when processing the delta partition. Using the delta index replaces the column scan with a regular index scan with another lookup. The costs are the same as for two index scans without a delta partition, one regular and the other one with $N_M = N_D$ and $D_M = D_D$. In practice, the delta partition will not grow large in size compared to the main partition, a priori reducing a delta indexes impact on the overall performance in general. Figure 7 demonstrates the theoretical cost reduction in a table containing 100000 rows in the main partition and 10000 rows in the delta partition with varying scan selectivity. The difference in scan costs between index scans with and without delta index constitutes about the ratio of delta to main size, in this case 10%. Recognizing the high insertion costs and the fact that in a productive environment the delta will be orders of magnitude smaller than the main, a delta index becomes fairly unprofitable. We therefore propose to leave the delta partition uncompressed and without an index.

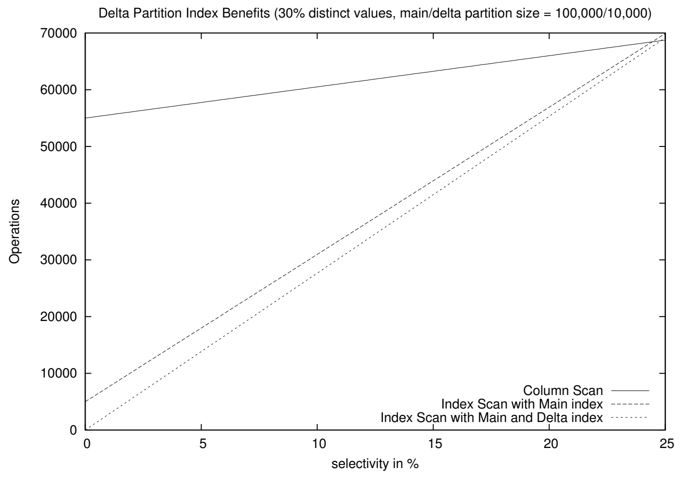


Figure 7. Performance enhancement through Delta index

### *E. Maintenance*

One of the downsides to having an index are the mentioned maintenance costs. When the values of a table change, the index needs to be adapted too. Due to the fact that in in HYRISE a dedicated buffer called delta partition handles all data changes without affecting the main partition the inverted index of the main partition has to be adapted while the merge process compacts the delta and main partition to create a new main partition.

There are different strategies that can be used to retain the validity of the index, which can mostly be associated to one of three basic principles: rebuild, update and merge. A merge could be achieved by combining main and delta partition indexes, but as explained in Section III-D an inverted index on the delta partition is not in favor of the write-optimized characteristics, so we will not go into any more detail about this approach.

The two strategies presented are therefore rebuild and merge.

*1) Rebuild Strategy:* The rebuild strategy constitutes the most naive approach to inverted index maintenance. Basically, after merging the two storage partitions the former index is discarded and a new one is created from scratch. There is no difference to the initial construction apart from the delta partition's rows now being indexed as well and consequently the costs are:

$$C_{IndexRebuild} = (N_M + N_D) * log(D_M + D_D) * \frac{L_{VID}}{B} \quad (8)$$

Although this may seem a less favorable option at first glance, rebuilding after merge process completed has the advantage of independence. First of all, it can be employed outside the merge process, a potentially costly and memory intensive operation. Therefore the merge process is not affected whether there is an index on a column or not.

Secondly, the main partition is already merged and due to its nature of being read-only will undergo no further alterations until the next merge operation. This allows the inverted index to be rebuilt in parallel to regular execution. Queries reading from the main partition will be affected shortly after the completion of the merge, as long as the index is not yet ready, but their requests can be served with table scans. Once the index rebuild is finished, table scans can again be replaced by index scans. Further the index rebuild could be delayed or rather scheduled as, for example, explored in the context materialized view maintenance [20].

*2) Update Strategy:* The update strategy facilitates the special characteristics of the differential store concept that is implemented in the HYRISE database system, the merge process to be exact. During the merge process a new dictionary is created from the combination of the main partition's dictionary and the created dictionary of the delta partition's contents. For the subsequent conversion of table contents a mapping is created that assigns each former value id in both main and delta the respective value id in the newly created dictionary. With this mapping at hand main and delta partition are processed sequentially, applying the mapping whenever the currently read value id has been changed in the new dictionary. As a last step the delta partition's content is appended to the main partition, resulting in a new, empty delta partition, which is ready for new insertions.

Now, the idea behind the update strategy is that a major part of the table, the main partition, is already indexed. Merging main and delta partition also does not change row ids, i.e. the values in the inverted indexes key-value-map, but only the value ids, i.e. the keys since an insert only approach is applied in HYRISE. The value id mapping can be used to update the existing index by changing keys where necessary. The basic balanced tree structure is assumed as basis for the index sorts the keys, thus a key change might result in shifting half or more of the other keys in memory. But since value ids in the main partition are sorted by their respective value, it can be safely assumed that $ValueID_{old} \leq ValueID_{new}$. Parsing the inverted index from the end now will ensure that no values will be shifted or overwritten when applying the mapping, the keys are relocated at constant time, because no re-sorting is necessary. The costs for this update are:

$$C_{OldIndexUpdate} = D_M * \frac{L_{VID}}{B} \quad (9)$$

This update should happen right after creating the new dictionary and mapping. When the delta partition's content is processed in order to apply the mapping, each row has to be extracted, checked and changed if necessary. To avoid a second processing of the delta partition rows, indexing can happen right during this operation. The necessary information (value id and record id) are already present and can be inserted into the inverted index in $\mathcal{O}(logD_M)$. Altogether the additional costs of updating an index during merge are:

$$C_{IndexUpdate} = D_M * \frac{L_{VID}}{B} + N_D * log(D_M + D_D) * \frac{L_{RID}}{B} \quad (10)$$

Despite the fact that the update strategy has to be part of the merge process and may impact the cache friendly sequential processing of table data, the difference as illustrated in Figure 8 is immense in theory.

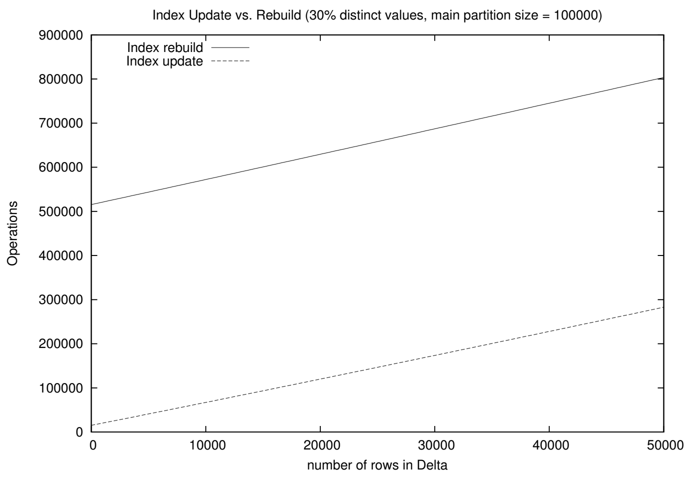


Figure 8. Comparison of index rebuild and update costs with varying delta partition size

## IV. Implementation and Evaluation

### A. Test Environment

The purpose of this section is to compare the theoretical findings with a working implementation. All tests were performed within HYRISE [2], a main memory hybrid storage engine prototype written in C++. While its main feature is the arbitrary partitioning of tables to accommodate mixed workload scenarios, it provides the necessary implementation of a differential store to allow for comparison with the assumed database layout of the previous sections such as a main and delta partition and dictionary compression. For the tests only the full column-oriented storage without horizontal partitioning were used.

All benchmarks were performed on an Intel Xeon X7560 native Octocore with a base clock of 2.26 GHz and a turbo boost frequency of 2.66 GHz accompanied by 256 GB DDR3 1066MHz main memory.

## *B. Inverted Index Implementation*

The implemented inverted indexes core is a Standard Template Library (STL) map. Despite offering better best-case complexity, an implementation as a hash-map was discarded due to its high worst-case complexity of $\mathcal{O}(N)$ [21]. Value ids and record ids are represented by 4 byte unsigned integers, the record id lists are STL vectors of the according type. To answer many types of requests, the inverted index provides methods for inserting single or multiple elements and retrieval of position lists for either one, multiple or a range of value ids.

As already proven, a larger number of columns has the same effect on table and index scan. Consequently the influence of the number of columns is excluded from the measurements, only one column is used.

## *C. Delta Index Insertion Costs*

To verify the calculated double insertion costs into the indexed delta partition, a delta partition index was implemented. The graph shown in Figure 9 indicates, that indeed almost twice as many cycles are used when updating the delta partition's index on insert operations. The actual difference is a bit smaller due to overhead created by the software that adds onto both, but still high enough to confirm the theoretical findings.

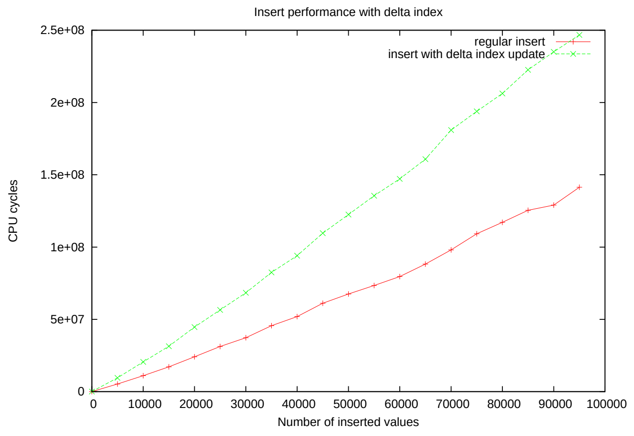


Figure 9. Insertion cost increase with Delta index

## *D. The Inverted Index in Practice*

As done in Section III-B, the influence of variations in main partition size, number of distinct values and selectivity were examined, but this time in practice.

Figure 10, depicts the variations in main partition size with two column scans and two index scans, both fetching once 10% and once 40% of the values. At a lower selectivity level, the index scan outperforms the column scan as expected, at 40% the difference is remarkably smaller. In contrast to Figure 3, the graphs seems not to be logarithmic. An explanation could be the fact that the STL map stores its keys sequentially in memory, allowing for a smaller number of cache misses and in consequence an effective lookup time that is below the theoretically average $log(N)$.

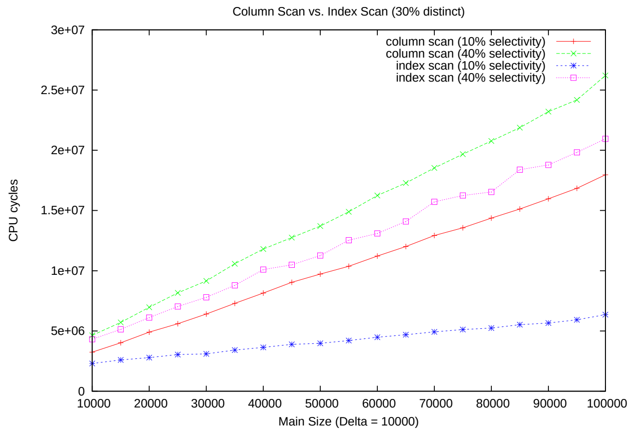


Figure 10. Actual index scan performance with varying main partition size

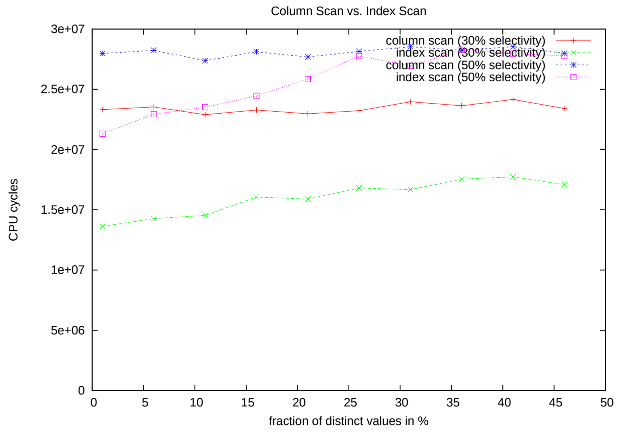


Figure 11. Actual IndexScan performance with varying levels of distinct values

A different perspective is offered in Figure 11. The theoretical analyses have proven that a column scan is not affected by the number of distinct values in a columns, because it scans all values anyways. This theory is justified by the more or less horizontal lines depicting the column scans' costs. The slight deviation can be explained by unpredictable variations in the overall system load, apart from that the scan costs are about constant.

In contrast, the costs for an index scan increase once reaching 20%-25% selectivity. On 50% distinct values it is outrun by the column scan. The graphs for the remaining two scans will meet somewhere around 70%-80%, almost exactly matching the theoretical graph in Figure 5.

Figure 12 underlines the selectivity's impact on table and index scan performance. The costs for an index scan grow almost twice as fast as the column scan costs, on 50% distinct values the break even point is reached at a selectivity of about 45%, and 65% on 20% distinct values respectively. Also here the curves' shapes resemble to the prediction in Figure 4.

In summary, the measurements and observations of the implementation largely align with the expectations that emerged from the theoretical discussion.

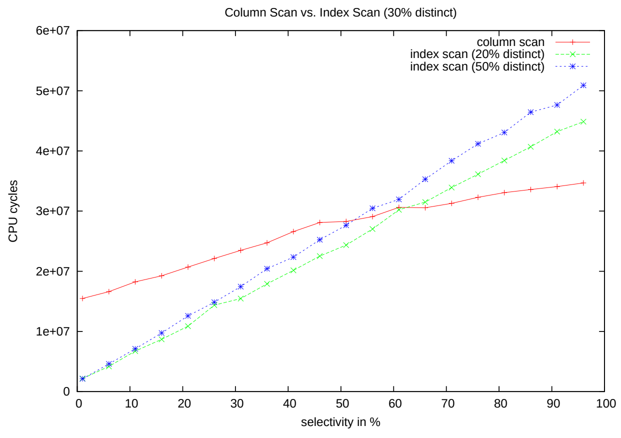


Figure 12. Actual index scan performance with varying selectivity

## E. Maintenance

Both the rebuild and update strategy were implemented in HYRISE to verify the theoretical findings. To make the actual difference more obvious, the regular merge was included in the graphs.

Figure 13 shows a result similar to Figure 8 from the previous chapter. The costs for the mere merge process grow almost equally to the number of rows in the delta partition, whereas a parallel update or subsequent rebuild means a linear growth with a greater factor. The update strategy's advantage of not having to re-index the main is pretty obvious and marks a constant offset from the rebuild costs. Yet, as the delta partition grows, the ratio of indexed to non-indexed rows diminishes and changes the overhead reduction from about 70% in the beginning to about 40% at a main-delta-ratio of 1:2.

Deductively it can be said that the update strategy always has a benefit over discarding and rebuilding, even though the difference decreases when the delta partition grows too large.

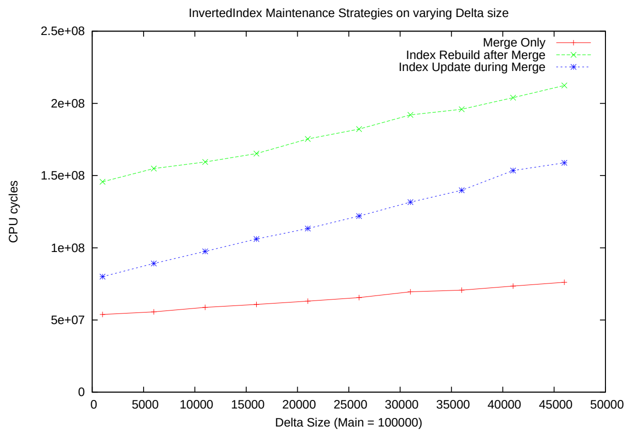


Figure 13. Comparison of implemented index rebuild and update with varying delta partition size

## V. Conclusion

In this paper, we showed that an inverted index can dramatically speed up value lookups in a table with a differential store as long as certain conditions apply. If either selectivity or the number of distinct values in an indexed column exceed limits that depend on the actual implementation, the performance gain is lost. As a consequence, the typical profiteers of such an index are selective queries fetching only small subsets or even just a single tuple from the database, which is the standard case in transactional-style queries. Although this paper concentrates on inverted indices for the sole purpose of data retrieval as a distinct query, many different kinds of typical operators in a database could make use of it. When for instance a merge requires only a handful of value ids to be changed in an indexed column, the respective rows could easily be found using the index, which is the case when choosing the so called in-place merge strategy that implements a sparse dictionary.

Besides, it has been presented that an inverted index used for the delta partition adds benefit to the overall performance. While it provides speedup capabilities similar to the main partition index, even though with less impact on the overall performance, the delta partition inverted index doubles the insertion costs. Due to the fact of the delta partition acting as a buffer and thus predictably being orders of magnitude smaller, the minor reduction in scan costs does not justify the high insertion costs as the delta partition is designed as a write-optimized partition.

The last conclusion regards maintenance and how the merge process's nature can be exploited for it. The two strategies explained both have advantages and disadvantages. While updating the index during merge saves memory accesses, it is encapsulated in the merge which makes parallelization difficult. Being more costly in total, this does not apply to the rebuild strategy. Whether or not an inverted index makes sense is subject to the individual case, i.e. the general pattern of the expected workload. While they cannot outperform full column scans for OLAP style queries, they are beneficial for answering OLTP style queries, which request typically only a few rows. Thereby the inverted index helps to avoid congestion on the main memory interface in a mixed workload environment.

## REFERENCES

[1] P. A. Boncz, S. Manegold, and M. L. Kersten, “Database Architecture Optimized for the New Bottleneck: Memory Access,” in *VLDB*, 1999, pp. 54–65.
[2] M. Grund, J. Krüger, H. Plattner, A. Zeier, P. Cudré-Mauroux, and S. Madden, “HYRISE - A Main Memory Hybrid Storage Engine,” *PVLDB*, vol. 4, no. 2, pp. 105–116, 2010.
[3] J. Krueger, C. Kim, M. Grund, N. Satish, D. Schwalb, J. Chhugani, H. Plattner, P. Dubey, and A. Zeier, “Fast Updates on Read-Optimized Databases Using Multi-Core CPUs,” *PVLDB*, vol. 5, no. 2, 2011/2012.
[4] D. J. Abadi, S. Madden, and M. Ferreira, “Integrating compression and execution in column-oriented database systems,” in *SIGMOD Conference*, 2006, pp. 671–682.
[5] J. Krüger, C. Tinnefeld, M. Grund, A. Zeier, and H. Plattner, “A Case for Online Mixed Workload Processing,” in *DBTest in conjunction with SIGMOD’10*, 2010.
[6] F. Huebner, J.-H. Boese, J. Krueger, F. Renkes, C. Tosun, A. Zeier, and H. Plattner, “A Cost-Aware Strategy for Merging Differential Stores in Column-Oriented In-Memory DBMS,” in *BIRTE - in conjunction with VLDB’11*, 2011.
[7] T. J. Lehman and M. J. Carey, “A Study of Index Structures for Main Memory Database Management Systems,” in *VLDB*, 1986, pp. 294–303.
[8] J. Rao and K. A. Ross, “Making B+- trees cache conscious in main memory,” *SIGMOD Rec.*, vol. 29, pp. 475–486, May 2000. [Online]. Available: http://doi.acm.org/10.1145/335191.335449
[9] V. Raatikka, “Cache-Conscious Index Structures for Main-Memory Databases,” Master’s thesis, University of Helsinki, 2004.
[10] F. Transier and P. Sanders, “Compressed Inverted Indexes for In-Memory Search Engines,” in *ALENEX*, 2008, pp. 3–12.
[11] R. Guo, X. Cheng, H. Xu, and B. Wang, “Efficient on-line index maintenance for dynamic text collections by using dynamic balancing tree,” in *Conference on information and knowledge management*. ACM, 2007, pp. 751–760.
[12] N. Lester, J. Zobel, and H. E. Williams, “In-place versus re-build versus re-merge: index maintenance strategies for text retrieval systems,” in *ACSC*, 2004, pp. 15–23.
[13] I. H. Witten, T. C. Bell, and C. G. Nevill-Manning, “Models for Compression in Full-Text Retrieval Systems,” in *Data Compression Conference*, 1991, pp. 23–32.
[14] T. C. Bell, A. Moffat, C. G. Nevill-Manning, I. H. Witten, and J. Zobel, “Data Compression in Full-Text Retrieval Systems,” *JASIS*, vol. 44, no. 9, pp. 508–531, 1993.
[15] A. Trotman, “Compressing Inverted Files,” *Inf. Retr.*, vol. 6, no. 1, pp. 5–19, 2003.
[16] H. E. Williams and J. Zobel, “Compressing Integers for Fast File Access,” *Comput. J.*, vol. 42, no. 3, pp. 193–201, 1999.
[17] D. Comer, “Ubiquitous B-Tree,” *ACM Comput. Surv.*, vol. 11, pp. 121–137, June 1979.
[18] M. Grund, J. Krüger, M. Kleine, A. Zeier, and H. Plattner, “Optimal Query Operator Materialization Strategy for Hybrid Databases,” in *DBKDA*, 2011.
[19] U. Drepper, “What Every Programmer Should Know About Memory,” 2007.
[20] J. Zhou, P.-Å. Larson, and H. G. Elmongui, “Lazy Maintenance of Materialized Views,” in *VLDB*, 2007, pp. 231–242.
[21] S. A. Crosby and D. S. Wallach, “Denial of Service via Algorithmic Complexity Attacks,” in *Proceedings of the 12th conference on USENIX Security Symposium - Volume 12*. USENIX Association, 2003, pp. 3–3.

# Sparse Dictionaries for In-Memory Column Stores

Johannes Wust, Jens Krueger, Martin Grund, Uwe Hartmann, Hasso Plattner
*Hasso Plattner Institute for Software Engineering*
*University of Potsdam*
*Potsdam, Germany*
*Email: johannes.wust@hpi.uni-potsdam.de, jens.krueger@hpi.uni-potsdam.de*
*martin.grund@hpi.uni-potsdam.de, uwe.hartmann@hpi.uni-potsdam.de*
*hasso.plattner@hpi.uni-potsdam.de*

***Abstract*—A common approach to achieve high read and write performance for column-oriented in-memory databases is to separate the data store into a read-optimized main partition and a write-optimized differential buffer. The differential buffer has to be merged into the main partition periodically to preserve read performance and increase memory efficiency. If data is dictionary-compressed, this merge process becomes time-consuming since it involves rewriting the whole main tables if dictionary mappings are changed. To reduce the merge time, we introduce the concept of sparse dictionaries, which can avoid rewriting the whole table in many cases. The basic idea is to place gaps in the dictionaries of the main partition which allows us to merge the differential buffer into the main partition without expensive recompression of the main tables. We leverage known data characteristics to optimize our algorithms for enterprise applications.**

***Keywords*-Column-store; In-memory databases; Dictionary compression; Write-optimized Store; Read-optimized Store**

## I. Introduction

An In-Memory Database (IMDB) is a database system where the primary persistence resides entirely in the main memory [1]. In recent years, the introduction of a 64 bit address space in commodity operating systems and the constant drop in hardware prices make large capacities of main memory in the order of terabytes technically feasible and economically viable. Together with ever increasing computing power due to multicore CPUs, this change enables the storage and processing of large sets of data in memory and opens the way for general-purpose in-memory data management.

Recently, column-oriented in-memory DBMS were proposed to consolidate transactional and analytical workloads in a single database system, which provides the potential for new enterprise applications and a reduction of the total costs of operating enterprise IT landscapes [2]. Following the data characteristics found in enterprise systems, the proposed architecture relies on dictionary compression per column to utilize memory efficiently. While a column-oriented storage model favors read-mostly analytical workloads, fast write operations on dictionary-compressed column structures are challenging. A common concept in columnar databases is to split the storage in two parts [3], [4]: a read optimized main partition and a write optimized differential buffer or delta store. For dictionary-compressed data, the read optimized store operates on a sorted dictionary, whereas the write-optimized store appends new values to its dictionary.

We call the process of unifying the two parts a merge. The main performance bottleneck of the naive algorithm as described in [5] is that the whole main partition needs to be copied, leading to a time- and memory-consuming operation – experiments with an implementation of this algorithm in our storage engine HYRISE [6] revealed that copying the structures consumes up to 50% of the total execution time of a merge.

Copying the columns to be merged is required as the dictionary mappings change throughout the merge and the value IDs of each record need to be changed. As the dictionary of the main partition is sorted, this happens each time a value is inserted. Our basic idea is to avoid this by placing gaps in the dictionary, leading to so-called sparse dictionaries. That way, we can add new values to the dictionaries of the main partition without having to change all following value IDs and merge differential buffer into the main partition *in-place*. We can perform this intermediate merge several times until the gaps have been filled, before we need to execute the full merge that involves copying the columns.

### A. Contributions

Specifically, our contributions presented in this paper are the following:

1) A new strategy to merge dictionary-compressed read-optimized and write-optimized stores based on a novel data structure called sparse dictionary
2) A cost model for estimating the cost of inserting a value into a sparse dictionary depending on the underlying data characteristics
3) A performance evaluation that compares the runtime of the regular merge process [5] with our optimized sparse merge

### B. *Related work*

Our work is based on the system model of an in-memory database as described in [7]. Besides this architecture specifically targeted for enterprise applications, other in-memory database have been recently developed. From a research point of view, MonetDB [8] and H-Store [9] have been the most influential systems; from a commercial perspective, SAP's In-memory computing engine, IBM's SolidDB and Oracle's Times Ten are best known.

Targeting the challenge of order-preserving dictionaries if the domain size is not known a priori, Binnig et al. [10] describe a data structure specific to string compression.

Concerning the merge algorithm as described in [5], another improvement has been proposed in [11]. This algorithm reduces memory consumption by merging single columns. Our object is to improve run-time and we consider the contribution in this paper as complimentary to this work.

### C. *Structure of this Paper*

This remainder of this paper is structured as follows: Section II gives an overview of the two proposed merge processes, the full sparse merge and the intermediate sparse merge. In Section III, we introduce the sparse dictionary and define the underlying data structure. Section IV describes a cost model for inserting values into a sparse dictionary, based on which we define operations on the sparse dictionary in V. Section VI compares the performance of our proposed sparse merge to the regular merge and we close this paper in Section VII with some concluding remarks.

## II. Full Sparse Merge and Intermediate Sparse Merge

We replace the regular merge process as described in [5] with two merge processes: a full sparse merge and an intermediate sparse merge.

Intermediate merges are faster than full merges, as they perform an *in-place merge* leveraging well-placed gaps of the dictionaries of the main partition in order to insert new values. As no, or only a small number of value IDs change, we do not need to copy the tables of the main partition, but change the values in the columns of the main partition in place using an index. Intermediate merges can still merge the whole differential buffer into the main partition, but the size of the new values in the dictionary of the buffer is limited by the number of the remaining gaps in the dictionary and the value domain (bit width) of the value IDs.

However, we cannot only rely on intermediated merges: at some point in time the number of gaps will be depleted and/or the width of the value IDs has to be increased. In the first case new gaps have to be added and they should be distributed in a way that supports the intermediate merge, once again requiring a rewrite of a large part of the table. In the latter case all value IDs will have to be rewritten. For these cases we propose the full sparse merge. In contrast to the intermediate sparse merge, it does not operate in-place on the main store and always has to rewrite the whole store. While merging the differential and main dictionary, and potentially increasing the resulting new main dictionary, it redistributes the gaps as efficiently as possible to speed up succeeding intermediate sparse merges.

In order to decrease the number of full merges required, the increasing width of the value IDs and the addition and redistribution of the gaps should be synchronized. It turns out that at a high enough filling level of the dictionary, it becomes more expensive to shift the values in a way that uses the last gaps than to do a full merge. Moreover subsequent intermediate merges benefit largely from the full merge.

In the following, we introduce sparse dictionaries, the data structure our proposed merge processes operate on, and describe the operations on this dictionary required for the full and intermediate sparse merge.

## III. Sparse Dictionary

The basic idea of a sparse dictionary is to leave gaps in regular intervals within the dictionary. As a consequence new value IDs can be inserted in the dictionary without moving other values around. In practice, the concept of sparse dictionaries entails a number of implementation specific questions, such as:

- How can gaps be identified?
- Where should the gaps be placed?
- How many gaps should the dictionary have?

In this section, we introduce some definitions to describe our data structures and algorithms throughout the remainder of this paper, and we evaluate data structures used to mark gaps in the dictionary vector.

When it comes to the other two questions, the naive approach would be adding a fixed number of gaps during each full merge and uniformly distribute the gaps over the dictionary vector. However, as some value IDs are used more often than others, moving them around in case no gap is located close to them is more expensive then others. Therefore, we introduce a cost model for changing values in the sparse dictionary in Section IV and discuss optimized operations in Section V.

### A. *Definitions*

To formally describe our data structures and algorithms throughout this paper we use the following notation:

$\mathcal{M}$: Attribute vector of the main partition of the considered column

$\mathcal{U}_{\mathcal{M}}$: Dictionary of the Main partition of the considered column prior to a merge process

$\mathcal{S}$: (new) Sparse Dictionary of the Main partition of the considered column after a merge

$\mathcal{G}_{\mathcal{S}}$: set of value IDs representing sparse elements / gaps within the sparse dictionary

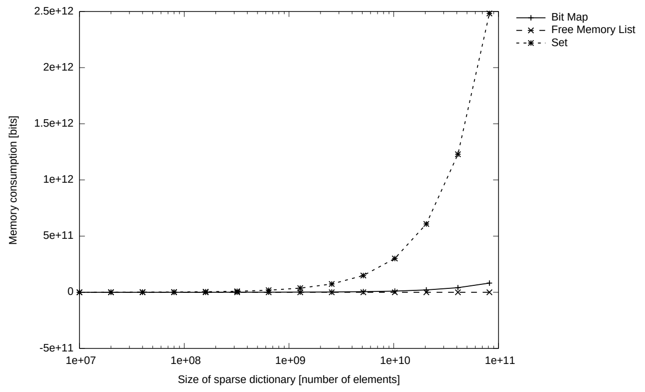


Figure 1. Additional Memory Consumption caused by gap identification with *30%* sparse values running on a 64 bit machine.

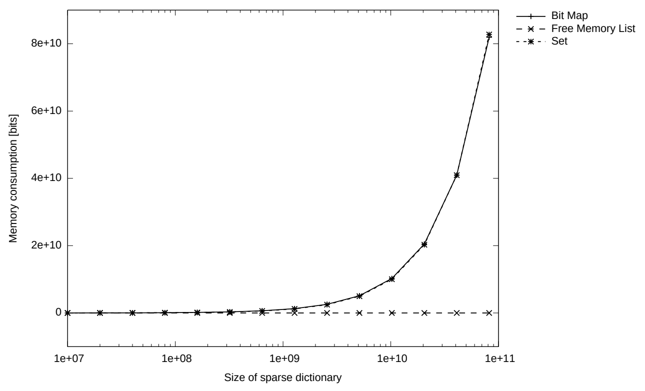


Figure 2. Additional Memory Consumption caused by gap identification with *1%* sparse values running on a 64 bit machine. Memory Consumption of bit map implementation and set are nearly equal.

$l(\mathcal{U})$: denotes the length in bits of a single value in the given dictionary

$\mathcal{D}$: Attribute vector of the differential buffer of the considered column

$\mathcal{U}_{\mathcal{D}}$: Dictionary of the Differential buffer of the considered column prior to a merge

$\mathcal{I}_{\mathcal{S}}$: index to identify which values in a dictionary are sparse

$d(vid) : [0, |\mathcal{S}|[ \mapsto W$: dictionary mapping for a column with domain $W$

Note that we store two vectors for each column: A dictionary vector that holds the mapping from value IDs to values for all distinct values of the considered column, and an attribute vector, that holds the value IDs.

Furthermore we define a metric called *Sparsity* $\Xi$ that represents the relative number of gaps in a sparse dictionary $\mathcal{S}$. It is defined as follows:

$$\Xi(\mathcal{S}) = \frac{|\mathcal{G}_{\mathcal{S}}|}{|\mathcal{S}|} \tag{1}$$

## *B. Evaluation of Data Structures for the Sparse Dictionary*

Our sparse dictionary $\mathcal{S}$ is implemented as a vector holding the values. To quickly identify gaps in the vector, we need an index $\mathcal{I}_{\mathcal{S}}$. The value of a gap can remain undefined – however, we can set it to the value of a preceding entry that is not sparse to assist standard binary search algorithms on the vector. As index structure, we have evaluated three options based on memory consumption and access time: a bit map, free memory lists [12] and a set storing the indexes of gaps.

The bit map stores one bit per entry of the dictionary and therefore leads to a memory consumption of $|Dictionary|$ bits and an access time into the vector of $O(1)$.

The free memory list stores in each gap the index of the next gap. As all values in the dictionary are distinct, the number of bits needed to encode the values is always bigger or equal to the width of the index. So the index always fits into the dictionary slot. In this way the memory overhead of the free memory list is zero (Note that we cannot store the values of preceding entries in the gaps, as mentioned above). The downside of the free memory list is that each check whether a certain value ID is free requires iterating over the whole free memory list. Furthermore each iteration generates a cache miss as the gaps are by purpose distributed over the dictionary. Hence the access time is $O(n)$, with $n$ being the number of gaps.

When using a *set* for storing the indexes of the gaps, memory consumption is heavily implementation dependent, but would scale up with number of gaps and only logarithmically with the whole number of elements in the dictionary as it is the case with the bit map. For our calculation we assume a self-balancing binary search tree, as it is used in the standard c++ library. This leads to a memory overhead $\delta m$ in bits of

$$\begin{aligned} \delta m = (log2(|Dictionary|)) * n \\ +size(Pointer) * (n-1) \end{aligned} \tag{2}$$

and an access time of $O(log(n))$, $n$ being the number of gaps.

Figures 1, 2, and 3 compare the memory consumption of each of the three data structures. Comparing figure 1 and 3 shows that the bit map has a far smaller memory footprint with a larger fraction of gaps than the set. Yet, when using smaller fractions of free spaces in the dictionary, the set outperforms the bitmap. As one can see in Figure 2 the break even point is about 1%. The free memory list does not need any additional memory but has a linear runtime for containment lookups. In conclusion it depends on the scenario to choose the right data structure: If memory consumption is a concern, the free space list is the right choice. If performance is the most important priority, the bit map offers the best characteristics, but the space characteristics of the set are better if only very little gaps (below 1%) are used.

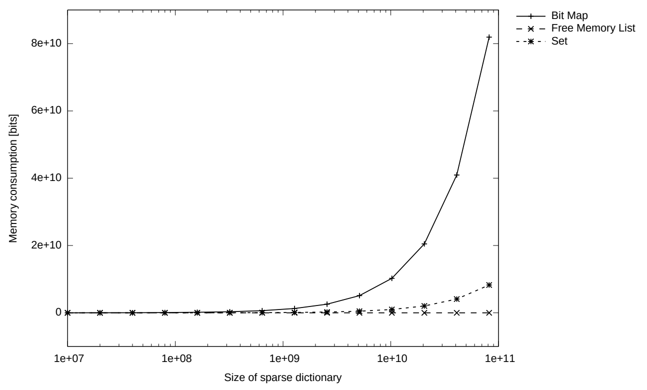


Figure 3. Additional Memory Consumption caused by gap identification with *0.1%* sparse values running on a 64 bit machine.

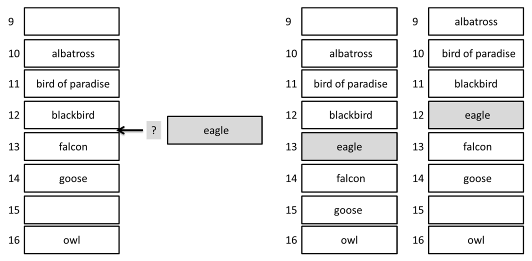


Figure 4. Insertion of a new element into a sparse dictionary and two possible outcomes

## IV. Cost Model for Changes in the Sparse Dictionary

In this section, we address the question of how the right places for the gaps can be found and based on this question, where to insert a new value in the dictionary. For inserting new values, the problem is trivial if we hit a gap at the position we want to insert. But a common case is that we must create an empty space at the desired position by shifting values in an existing gap. Figure 4 illustrates this problem. The two cases have different costs depending on the number of occurrences of each changed value ID. Consequently we define a cost function to approximate the resulting cost for changing a specific value ID in the dictionary. We also apply this for cost function later for placing the gaps in case we create a new sparse dictionary during a full sparse merge (Section V-A). The idea is to leave more gaps around values that implicate high costs when shifted.

*Change Cost Functions:* We define a change cost function $\pi$ that assigns the cost of changing the associated value to each value ID:

$$\pi(vid) : [0, |\mathcal{S}|[ \mapsto \mathbb{N} \tag{3}$$

In the following we discuss potential cost functions.

*1) Simple Cost Function:* The simple cost function returns 1 for every value ID.

$$\pi_{simple}(vid) = 1 \tag{4}$$

Although this function causes no computation overhead, it is only useful, if the occurrence of distinct values in the main store follows a uniform distribution. However, an analysis of the data characteristics of enterprise systems in [13] showed that the value distribution of the columns with more than one distinct values is in more than 50% of the cases better approximated by a zipf distribution.

*2) Quantitative Cost Function:* A more advanced change cost function is the so-called quantitative cost function. It calculates the number of elements in the main storage that have to be rewritten as a result of changing the given value ID, which becomes computationally expensive in case of large tables. In case of a small number of distinct values, a potential solution would be to maintain a list of occurrences of each value; however, the additional memory consumption increases in case of many distinct values.

$$\pi_{quantitative}(vid) = |x : x \in \mathcal{M} \wedge x = vid| \tag{5}$$

*3) Pareto Cost Function:* The problem with the simple cost function introduced in IV-1 is that it assumes an even distribution, whereas the quantitative cost function suffers from a long computation time or high space consumption to save pre-calculated values. As our cost model should support the decision for the placement of gaps, we do not require an optimal result, but rather a heuristic that is fast to calculate. We therefore have defined the pareto cost function, which is based on the observed data characteristics in the enterprise application domain discussed in [13].

Analyzing the characteristics of the data sets discussed in [13] we found out that in most columns at most $20\%$ of the distinct values are sufficient to cover at least $80\%$ of all rows. We use this observation to categorize the values in the dictionary in two categories: The first category, called $FC$, consists of the $20\%$ of the distinct values that cover the most rows in a given column. The second category, called $SC$, contains all remaining values. The pareto cost function is basically the quantitative cost function with a reduced co-domain. Based on the observed value distribution, we give those two categories a relative value. In our analyzed data set we have seen that $20\%$ of values covered between $73.53\%$ up to $94.02\%$ of the of elements in the main store, on average $86.91\%$. Assuming an even distribution within the categories this means that each $1\%$ of the first class distinct values of the first category covers on average $\frac{86.91\%}{20\%} = 4.34\%$ of the rows in a column. In contrast to that, $1\%$ of the second class values cover only $\frac{100\%-86.91\%}{80\%} = 0.16\%$ of the values. Accordingly one distinct value of the first category covers on average *27 times* more values than a value of the second category.

$$\pi_{pareto}(vid) = \begin{cases} 27, & d(vid) \in FC \\ 1, & d(vid) \in SC \end{cases} \tag{6}$$

Note that the assigned values for the pareto cost function depend on the expected value distribution of the data set at hand. The assigned values to each class can be adjusted for different value distributions.

## V. Operations on the sparse dictionary

In this section, we describe operations on the sparse dictionary that are required when performing the proposed full and intermediate sparse merge. These operations are the creation of a new sparse dictionary in a full merge and insertion of values into the dictionary in an intermediate merge.

### A. Creating the Dictionary

Creating a new sparse dictionary during the full sparse merge is considered as a merge of two existing dictionaries. At startup, the initial main dictionary is empty. Before we describe the proposed algorithm in detail, we discuss the rationale for choosing the number of gaps.

*The Size of the Dictionary and the Number of Gaps:* As discussed in Section III, we use a vector as the underlying data structure for a dictionary. Having the sizes of the two input dictionaries as input, we need to determine how much size we want to reserve for our new dictionary. There are several points to consider:

1) The more gaps there are in the dictionary, the faster the intermediate merges are as it reduces the likelihood of shifting values.
2) The more gaps there are in the dictionary, the more intermediate merges can be done before another expensive full merge is required.
3) The more gaps there are in the dictionary, the more space is used by the dictionary.
4) The more gaps there are in the dictionary, the less values fit into a cache line when iterating over the dictionary values, slowing down the iteration.
5) If the total number of elements in the dictionary excesses the size of the next power of 2, a full merge is required to increase the bitmask for a value ID and the marginal cost of additional gaps is increased.
6) The minimum required size of our new dictionary might not be $|\mathcal{U}_\mathcal{M}| + |\mathcal{U}_\mathcal{D}|$ since values might be contained in both dictionaries. This is actually the upper bound of the required minimum size.

*Size Depending on Space Efficiency:* Points 1 - 4 show us that there is basically a trade-off of size and speed. The size of the gaps is determined by $|\mathcal{G}_\mathcal{S}| * l(\mathcal{S})$. Point 5 shows us that there is a sweet spot for this trade-off: total number of elements including gaps is most efficient just before it excesses any power of 2, requiring additional space per value ID. Thus an easy to make estimate for an efficient element number would be:

$$|\mathcal{S}| = 2^{\lceil \log_2(|\mathcal{U}_\mathcal{M}|+|\mathcal{U}_\mathcal{D}|)\rceil} \tag{7}$$

*Distribution of gaps:* Having determined the size of the new sparse dictionary, we need to define the distribution of the gaps based on the cost function $\pi$. A fairly simple but fast algorithm would be to evenly distribute the gaps – however, that way we would not take our cost function into account. The A-Star algorithm always offers a optimal solution for our problem but its runtime and in particular its memory consumption characteristics make it unsuitable for our case [14]. A more viable alternative is the greedy algorithm [15], although it does not offer an optimal solution.

*Algorithm for creating the Dictionary:* Algorithm V.1 describes the merge process of two dictionaries and the creation of the new sparse dictionary.

**Algorithm V.1:** MERGEDICTIONARIES($\mathcal{U}_\mathcal{M}$,$\mathcal{U}_\mathcal{D}$)

```
$\mathcal{S} \leftarrow newSparseDictionary(|\mathcal{U}_\mathcal{M}|, |\mathcal{U}_\mathcal{D}|)$
$\mathcal{I}_\mathcal{S} \leftarrow newSparseDictionaryIndex(|\mathcal{U}_\mathcal{M}|, |\mathcal{U}_\mathcal{D}|)$
$STemp \leftarrow$ vector of size $|\mathcal{U}_\mathcal{M}| + |\mathcal{U}_\mathcal{D}|$
$gaps \leftarrow$ vector of size $|\mathcal{U}_\mathcal{M}| + |\mathcal{U}_\mathcal{D}|$
$mainTemp \leftarrow$ vector of size $|\mathcal{U}_\mathcal{M}|$
$diffTemp \leftarrow$ vector of size $|\mathcal{U}_\mathcal{D}|$
$i, j, k \leftarrow 0$
while $i \neq |\mathcal{U}_\mathcal{M}| \vee j \neq |\mathcal{U}_\mathcal{D}|$
  do
    $value \leftarrow min(\mathcal{U}_\mathcal{M}[i], \mathcal{U}_\mathcal{D}[j])$
    if $value == \mathcal{U}_\mathcal{M}[i]$
      then
        $mainTemp[i] \leftarrow k$
        $i \leftarrow i + 1$
        $value \leftarrow \mathcal{U}_\mathcal{M}[i]$
    if $value == \mathcal{U}_\mathcal{D}[j]$
      then
        $diffTemp[j] \leftarrow k$
        $j \leftarrow j + 1$
        $value \leftarrow \mathcal{U}_\mathcal{D}[j]$
    $STemp[k] \leftarrow value$
    $k \leftarrow k + 1$
$gaps \leftarrow findDistribution(STemp)$
$i, j, k \leftarrow 0$
while $i < |\mathcal{U}_\mathcal{M}| + |\mathcal{U}_\mathcal{D}|$
  do
    $j \leftarrow gaps[i]$
    $value \leftarrow STemp[i]$
    while $j > 0$
      do
        $\mathcal{S}[k] \leftarrow value$
        $\mathcal{I}_\mathcal{S}[k] \leftarrow 0$
        $k \leftarrow k + 1$
        $j \leftarrow j - 1$
    $\mathcal{S}[k] \leftarrow value$
    $\mathcal{I}_\mathcal{S}[k] \leftarrow 1$
    $k \leftarrow k + 1$
    $i \leftarrow i + 1$
```

The merge of the dictionaries is similar to the one described by Krueger et. al. [5] but inserts gaps to the dictionary in a second pass. We require a second pass, as we need to first merge the dictionaries to calculate the distribution of gaps as discussed in the previous subsection.

Having created a temporary dictionary vector with the appropriate size ($STemp$), we iterate over the two ordered input dictionaries from the main partition and the differential buffer in parallel and merge them, while preserving the order of all values. Note that both dictionaries can contain equal values. We enter the old value IDs into the vectors $mainTemp$ and $diffTemp$ to preserve a mapping from old value IDs to new value IDs, required by running transactions. In a next step we calculate the number of gaps that should be inserted into the final dictionary prior to each value ($findDistribution$), applying the cost function $\pi$ and a greedy algorithm [15], as discussed previously. In the second pass, we iterate over the temporary dictionary, as well as the vector including the gaps and add gaps and values to the new sparse dictionary $\mathcal{S}$. To indicate whether an entry of $\mathcal{S}$ is a gap, we set the according position in the bitmap $\mathcal{I}_{\mathcal{S}}$ to 0.

## B. *Inserting into the Sparse Dictionary*

The insertion of an new value is trivial if we have to insert it before or after a gap. However, we might also face the issue that we must create an empty space at the desired position by shifting values in an existing gap (see Figure 4). Note that multiple shifts can happen during an intermediate sparse merge. Therefore we note all changes in so-called change map described in the next section and apply all changes to the main partition afterwards.

If we have to create an empty space to insert the new value, we have to choose whether to shift the succeeding or the preceding values to the next gaps. To decide which values to shift, we calculate the estimated resulting cost using $\pi$ and bit map $\mathcal{I}_{\mathcal{S}}$ we introduced earlier. Afterwards we shift all values between our insert position and the selected gap by one value ID and note this change in a change map. Finally we insert the given value in order.

*Creating the Change Map:* Having added a new entry to the dictionary and having shifted value IDs in it, we need to reflect these changes in the column's main partition. We need to refresh the value IDs of all values that we shifted in the dictionary. However, it would not be practical to apply those changes to the main data store immediately. There are several reasons for this:

- Double updates: Having changed one single dictionary key requires to change all associated keys in the main partition. If the next insertion into the dictionary causes the first value to change again, all values in the main partition have to be changed once more.
- Unusable without index: The performance penalty will further increase if no index is used. The insertion of each value into the dictionary would require a full table scan in order to update the values.

Consequently, we propose a change map data structure that stores a mapping from the old value ID of a shifted value ID to the corresponding new value ID. We store all changes with the dictionary during the whole merge in this data structure. At the end of the merge, we commit these changes to the main store, avoiding the two problems stated above. To record these changes, we use a CSB+ tree that is optimized for read as well as write operations [16].

We only insert position changes of already existing values, meaning that we have to exclude all new values that were shifted in course of the merge. The reason is the following: Given a new value is inserted at position 12. To do this the old value at 12 needs to be shifted to 13. Thus $12 \mapsto 13$ is added to the change set. Now another insertion causes 12 shift to 13 and 13 shift to 14. In this case there would be two values for 12 in the change map: $12 \mapsto 13$ (the new value) and $12 \mapsto 14$. As soon as we replace value IDs in the main storage we would not know whether we should replace the value ID 12 with 13 or 14. Moreover, there is no point in tracking shifts of new values, since none of them are in the main storage and can be replaced. Thus we create another bitmap similar to the one saving whether a value ID is occupied or not. This bitmap saves whether a value ID points to a new Value or not.

Our goal is a tree structure that maps the value ID for a value before the merge to the corresponding value ID after the merge. If a value ID that has to be refreshed was already mapped, we need to know this and have to find the original ID (pre-merge) that maps to this ID. This operation involves a reverse search on the tree which results in a linear search over all values of the change map. This would have to be done at every insert of a change item and thus potentially multiple times per value insertion. In the following, we propose three methods of reducing the search time.

*A Second Mapping:* The simplest approach to this problem would be the construction of a second map, or an index structure for the reverse lookup. But this would involve an increase in memory consumption since the keys and values would have to be stored in memory twice. Moreover the write performance, which is important to us because of the number of updates, would suffer as well.

*Shared-Leave Structure:* Another approach would be the use of a Leaf structure as proposed in [10]. This would avoid storing value IDs twice in memory.

*Mapping with Lazy and Fast Search:* The two proposals above were general solutions – in the following, we propose a specific solution exploiting the characteristics of our specific case

First, we define the Fast Search: The input of our function is a mapping $x \mapsto y$ of the current state of the dictionary which maps the value ID $x$ via the mapping function $f(K)$ to a new ID $y$. In the following we call $x$ the key of the

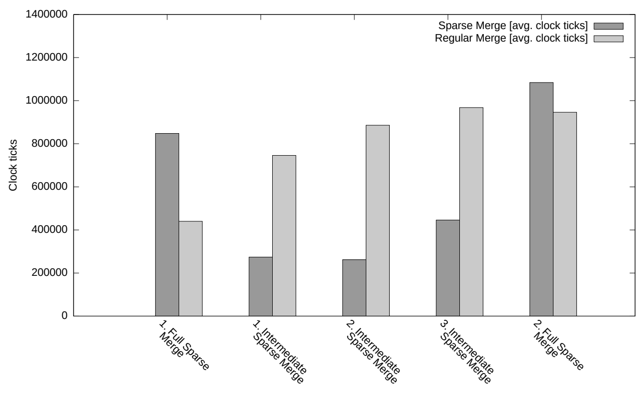


Figure 5. Performance of the Full Sparse Merge and Intermediate Sparse Merge compared to a regular Merge over multiple merge cycles

mapping and $y$ the new value. Hence we need to check the map $m$, whether it already contains a mapping $z \mapsto x$. In this case we have to replace it by $z \mapsto y$. Searching a mapping with new value $x$ in our map would require $\mathcal{O}(n)$ steps ($n$ being the number of elements in $m$), as we need to iterate over all values in $m$ in the worst case.

But we know that we find a mapping with key $z$ in close distance where we would enter $x \mapsto y$ in the map, as each shift increases the distance between $x$ and $z$ by one. Thus we can iterate starting from the closest key to $x$ in both directions, thereby exploiting the cache line by reading chunks of date in each direction. However this assumes that a mapping $z \mapsto x$ is contained in our map which is typically not the case.

In order to deal with this issue we propose the Lazy Search. The idea of Lazy search is to reduce the question: "Which $z$ maps to $x$: $f^-1(x)$?" to the question "Is there a $z$ that maps to $x$: $x \in f(K)$?". If we can answer the latter question quickly, we will not run into worst case complexity of the Fast Search, as we abort the search in case $x \notin f(K)$. For the discussion of deciding whether $x \in f(K)$, we define the following:

| | |
|---|---|
| $K$ | is the set of all value keys of the mappings in the change map |
| $E_{new}$ | is the set of all value IDs which constitute a gap in the current state of the dictionary during merge |
| $E_{old}$ | is the set of all value IDs which constituted a gap before the merge started |
| $N$ | is the set of all new value IDs that belong to a new value from the differential buffer |

Furthermore, we know, that $f(x) \neq x$, as we only record changes of a mapping, and that $x \notin E_{new}$, since we only shift values that are not empty. As discussed earlier, we do not store new values in the changed map, thus $x \notin N$. We can exclude these cases.

If the element at $x$ was not shifted since the beginning of the merge ($x \notin K$) and $x$ was not empty at this point in time, meaning $x \notin E_{old}$, we know that the same element is still at this position as only one element can be contained at the same time and because of $x \notin K$, $x$ was never moved. Consequently, no element could have been moved to $x$ and thus $x \notin f(K)$:

$$x \notin K \land x \notin E_{old} \Rightarrow x \notin f(K) \tag{8}$$

On the other hand, if $x$ has been shifted since the beginning of the merge and $x$ was not empty at this point of time, we know that another value has been mapped to the value ID $x$ and we find a mapping with new value $x$ in our map. Hence:

$$x \in K \land x \notin E_{old} \Rightarrow x \in f(K) \tag{9}$$

If the element at $x$ was empty at the beginning of the merge and we encounter a mapping from value ID $x$ to another ID, we have to find a mapping in $m$ that has assigned a value to $x$. Hence:

$$x \in E_{old} \Rightarrow x \in f(K) \tag{10}$$

We can summarize these observations in the following truth table:

| $x \in K$ | $x \in E_{old}$ | $x \in f(K)$ | Rule |
|---|---|---|---|
| 1 | 0 | ✓ | (9) |
| 0 | 1 | ✓ | (10) |
| 0 | 0 | × | (8) |

As we can see, based on the two conditions $x \in E_{old}$ and $x \in K$ we can quickly decide whether we should perform a fast search. $x \in E_{old}$ can be checked in constant time using the old bitmap and $x \in K$ can be checked in $\mathcal{O}(log(n))$, by searching the CSB+ tree. This way we can efficiently avoid searching the map if the sought-after mapping is not in the map.

Finally, the attribute vector of the main partition must be updated using the change map. This step is quite fast forward. All entries in the main which are equal to a key in the change map must be replaced by the corresponding value. It is best to use an index for this operation because otherwise a full table scan is required. After that, the entries from the attribute vector of the differential buffer must be appended to the main partition using the value IDs of the sparse dictionary of the main storage.

## VI. Performance Evaluation

*Test Setup:* In order to evaluate the performance of the sparse full merge as well as sparse intermediate sparse we set up an experiment that executed five subsequent merges: 2 full merges and 3 intermediate merges and compared the

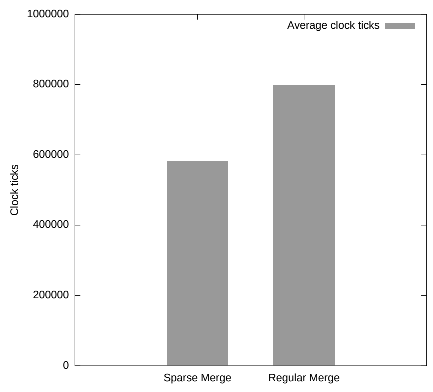


Figure 6. Total time spend on merging in the course of 5 subsequent merges

results against five regular merges. The used table starts with 50000 entries in the main part. At each stage of the experiment, the merge has to add a differential buffer with 20% of the size of the current main partition. Each distinct value is represented on average 10 times in the table. The first full sparse merge creates a sparsity of 50% for the subsequent intermediate merges. The first full merge is required to bring the table and the dictionary in a state that allows the subsequent in-place merges through the gaps in the dictionary that are created in this merge. The last full merge restores the starting state (a dictionary with a lower sparsity) to offer a fair comparison to the regular merge. The performance evaluation was done using the column oriented in-memory database prototype HYRISE [6]. The test machine used a 64 bit Linux (Ubuntu) operating system and was equipped with a 2.4 GHz Core 2 Duo processor with 2 cores and 2 GB of RAM. Figure 5 shows the results of our experiment.

*Results:* As shown in Figure 5 the full sparse merge is slower than a regular merge due to the overhead of reorganizing the dictionary. But it enables the faster intermediate merges. However, it is striking that the runtime of the intermediate merges increases faster than of the regular merges. This is due to the dictionary filling more with each additional sparse merge, leading to more shifting of value IDs in the dictionary. The more often we merge, the closer the runtime approaches the regular merge. Figure 6 shows the run times and deviations of all five merges (consequently including the full sparse merges) added together. We can see, that the *total* time the system uses to merge decrease by roughly 25% when using full sparse merges and intermediate sparse merges compared to a regular merge.

## VII. Conclusion

In this paper we showed implementation techniques for the concept of an intermediate sparse merge and a full sparse merge both using a sparse dictionary. The full sparse merge reorganizes a dictionary and a table so that several in-place merges using the intermediate sparse merge can be executed. The full sparse merge "charges" the table so that it can use a few very fast intermediate sparse merge that do not require the table to be copied. The combination from full sparse merge and intermediate sparse merge can reduce the total time the machine spends merging by around 25%, thus allowing other operations such as queries use the freed resources. Additionally, the intermediate sparse merge avoids the high base cost of the copy operation. Thus it scales with the size of the differential buffer and the state of the dictionary. Therefore it becomes possible to use the merge more often at a lower cost. This improves the general query performance since the differential buffer is smaller and more values reside in the read optimized main store.

## References

[1] H. Garcia Molina and K. Salem, "Main memory database systems: an overview," *IEEE Transactions on Knowledge and Data Engineering*, vol. 4, no. 6, pp. 509–516, 1992.

[2] H. Plattner, "A common database approach for oltp and olap using an in-memory column database," in *Proceedings of the 35th SIGMOD international conference on Management of data*, ser. SIGMOD '09. New York, NY, USA: ACM, 2009, pp. 1–2.

[3] M. Stonebraker, D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. R. Madden, E. J. O'Neil, P. E. O'Neil, A. Rasin, N. Tran, and S. B. Zdonik, "C-store: A column-oriented dbms," in *VLDB*, Trondheim, Norway, 2005, pp. 553–564.

[4] P. A. Boncz, M. Zukowski, and N. Nes, "Monetdb/x100: Hyper-pipelining query execution," in *CIDR*, 2005, pp. 225–237.

[5] J. Krueger, M. Grund, C. Tinnefeld, H. Plattner, A. Zeier, and F. Faerber, Eds., *Optimizing Write Performance for Read Optimized Databases: Database Systems for Advanced Applications*. Springer, 2010.

[6] M. Grund, J. Krüger, H. Plattner, A. Zeier, P. Cudre-Mauroux, and S. Madden, "Hyrise: a main memory hybrid storage engine," *Proc. VLDB Endow.*, vol. 4, pp. 105–116, November 2010.

[7] H. Plattner and A. Zeier, *In-Memory data management: An inflection Point for Enterprise Applications*, 1st ed. Berlin: Springer, 2011.

[8] S. Manegold, M. L. Kersten, and P. Boncz, "Database architecture evolution: mammals flourished long before dinosaurs became extinct," *Proc. VLDB Endow.*, vol. 2, pp. 1648–1653, August 2009.

[9] R. Kallman, H. Kimura, J. Natkins, A. Pavlo, A. Rasin, S. Zdonik, E. Jones, S. Madden, M. Stonebraker, and Y. Zhang, "H-store: a high-performance, distributed main memory transaction processing system," *VLDB*, pp. 1496–1499, 2008.

[10] C. Binnig, S. Hildenbrand, and F. Färber, "Dictionary-based order-preserving string compression for main memory column stores," in *Proceedings of the 35th SIGMOD international conference on Management of data*, ser. SIGMOD '09. New York, NY, USA: ACM, 2009, pp. 283–296.

[11] Jens Krueger, Martin Grund, Johannes Wust, Alexander Zeier, and Hasso Plattner, "Merging differential updates in in-memory column store," *DBKDA 2011*, 2011.

[12] R. Fenichel and J. Yochelson, "A lisp garbage-collector for virtual-memory computer systems," *Communications of the ACM*, vol. 12, no. 11, pp. 611–612, 1969.

[13] F. Huebner, J.-H. Boese, J. Krueger, F. Renkes, C. Tosun, and A. Zeier, "A cost-aware strategy for merging differential stores in column-oriented in-memory dbms," in *BIRTE workshop in conjunction with VLDB, Seattle*, 2011.

[14] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, no. 2, pp. 100–107, 1968.

[15] V. Chvatal, "A greedy heuristic for the set-covering problem," *Mathematics of operations research*, pp. 233–235, 1979.

[16] V. Raatikka, "Cache-Conscious Index Structures for Main-Memory Databases," http://ethesis.helsinki.fi/julkaisut/mat/tieto/pg/raatikka/cachecon.pdf (Accessed February, 2012), Feb. 2004.

# BI: 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 past years the world has witnessed a large number of changes motivated by the financial crisis concerning how the process of management occurs on production on factories. A way to control this situation is through indicators. This article describes not only usual indicators in lean manufacturing that can be used in human resource management but also the best way to design the data warehouse considering the level of granularity and the level of data detail. To begin with, we will analyze the lean manufacturing indicators addressed in the academic literature. Furthermore, we will analyze these indicators with the indicators used by leading research companies in Brazil. Lastly, we will design the data warehouse to comport these indicators considering the level of granularity and the level of details to analyze these data. By the end of this project, we intend to get the list of indicators capable of managing and analyzing the production and performance of the employees in factories. With these indicators, the factory will be able to manage the team correctly, whatever the situation of global finance is. This approach allows us to determine the indicators that will help the company identify problems on the team management for increasing the production and it will also contribute for real growth of the company. The indicators identified in this article will determinate the way to implement successfully BI solutions associated with human resource systems and manufacturing systems.**

***Keywords*-*BI; Lean Manufacturing; Indicators; HR Systems.***

## I. INTRODUCTION

Everything we do has a goal. The goals of generating indicators are many, and the focus of this article is studying the lean production indicators that can be applied to people management. Carreira [1] comments that maybe one of the essential principles of the lean production is the continuous elimination of waste. In order to reduce or eliminate the waste of any activity / process, as lean production preaches, we need first to measure this activity / process. Without the follow-up of the executed processes, it is not possible to indicate if anything needs enhancement. Based on this assumption, this article studies the indicators for monitoring and reducing the waste associated to people management systems. The first question is: why applying the principle of lean production to monitor indicators in people management systems? The answer is simple: waste reduction can be achieved through employee performance or allocation. Waste is not only associated to raw material, but also to the processes that occur inside the company. When we think about waste, the first thing that comes to mind is physical material, but we cannot forget that even employees can be a part of waste (or even wasted).

As Hibbs and Sullivan [2] say, the purpose of lean manufacturing is to deliver the final product to the client as fast as possible, and one way to do that is finding and eliminating waste. However, identifying what is and what isn't waste is not always an obvious task. Shigeo Shingo [3], co-developer of Toyotas production system, identified seven types of waste (and one additional type) that can be easily remembered by the acronym DOTWIMP:

1) Defects: This is probably the most obvious type of waste, lean production focus on preventing defects differently from the usual finding and repairing them.
2) Overproduction: producing more than necessary or producing before it is necessary is regularly seen as stocking.
3) Transportation: Unnecessary movement between processes. When moving material or items between factories, work cells, offices or machines, no value is generated.
4) Waiting: People or items waiting for the next production process.
5) Inventory: All material, work in progress and final product that is not initially processed. Inventory over the minimum will take on space and generate delays on identifying flaws.
6) Motion: People or equipment in motion or moving more than necessary to execute the necessary.
7) Processing: Processing the final product to the client beyond necessary, which only add costs and no value to it.
8) Additional eighth waste: This is regularly noted as an additional waste to the seven original ones, and it refers to under using the creativity and abilities of the employees.

Having identified the waste possibilities related to the employees, it will be necessary to determine a data warehouse that might uphold this data, and that will allow us to focus on the problem.

According to Vercellis [4], a data warehouse depends on the objective that is aimed to achieve. Before designing a data warehouse, we need to bear in mind which questions we would like to answer. For example, which cost center is spending more, or from the most costly center, which is the employee generating more costs? The answer to this question is directly related to the granularity of the data warehouse, and it is extremely important for the success of the project. With all these premises, by the end of the project we will have indicators connected to the lean manufacture that will be able to be applied to the human resources management. Besides that, we will have a data warehouse implementation proposal correspondent to the level of data granularity necessity.

## II. METHODS

According to McClellan [5], for approximately 20 years the Manufacturing Execution Systems (MES) were the focus for manufacture management. Initially, it was developed for providing the first management line, with visibility to manage the job requests and the attributions of the work units. The MES systems were involved on the indispensable connection between the stakeholders and the logistics production processes events. For the fact that the MES system can manage and execute the log of events of the logistics and production processes, they are an important and accurate real-time data source, becoming integrated to the corporation intelligence. In another way, the MES systems are a collection of business processes that provide a real time vision of events that are happening in the factory execution plan, or calculates how much and what must be produced, according to data received from the Enterprise Resource Planning (ERP) system.

Wang [6] comments that the maximum priority of the lean manufacture is to measure the performance of the processes. The measurement of the process performance provides the desirable level of results, which is the objective for taking a decision on production and it helps identifying the desired production level for the programmed activities. Now, be careful with the objective that you are willing to find through the indicators; you might achieve them is what said Levinson and Rerick [7]. Meaning, many times the indicators will reveal information that we are not willing to listen to or recognize.

Levinson and Rerick [7] highlights 3 rules for identifying process performance indicators:

1) Indicators need to be objective. The indicators must be precisely defined and quantifiable (needs to be possible to measure in numbers).
2) The process measured through the indicator needs to be under the control of the team or person that will execute the measurement (the indicator needs self control).
3) The indicator needs to encourage the work environment and needs to help the company to obtain corporate results. Besides the indicators being related to the objectives of the company, it needs to be understood by all as such.

The items above highlighted the indicators characteristics, in order for it to have an expected goal that can be understood by everybody in the company as constructive indicators. This will help the teams to manage these indicators and work positively on obtaining such, or else the indicator might be seen as criticism directly connected to a team or process. The highlighted items showed the objective of the indicators related to the manufacture process to the processes inside a factory. The responsible body of execution of processes inside factories is human resources, or, besides worrying about how the processes are being managed, there is a need to stay alert to the individual performance of the human resources (people).

According to Gawron [8], there are two ways to evaluate the performance of a person:

1) Subjective method: through opinion obtained in interviews and questioning or through the work environment follow-up.
2) Experimental method: through data collection for measuring performance.

The main way of measuring production is though gathering of data for performance measurement. This analysis form is easier, as it can be executed in an automatic way through an information gathering system and can be seen in a synthesized way (making people groups, processes and sector analysis easier) or in an analytical way. For the generation of lean manufacture indicators related to people management, the work must use long-standing references on people management processes, i.e. Before entering on detailing of a few selected indicators, some concepts must be understood:[9]

1) Effort: Level of energy and human creativity put in a task.
2) Performance: Way of using the effort to achieve a final goal.
3) Goal: What is wanted to obtain joining effort and performance on a task.
4) Results: the consequence of using this energy.
5) Productivity: the proportion between the obtained result and the amount of energy necessary to obtain the result.

As some examples of indicators that will be dealt with on the ongoing task, we quote a few indicators below:

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 employees wage that the product retains by its manufacture process.

$$WageCostUnit = \frac{AmountofRemuneration}{TotalofProducedUnits}$$

The generation of the indicators shown above and other indicators that will be worked on, will depend on obtaining data. The methods for obtaining the necessary data for generating the indicators for its turn will depend on the area of execution of the indicators. The indicators generated in this task can be applied to several types of manufacturing companies, considering that in some cases the possibility of investment of the client for producing data will be limited. The gathering of information for the indicator generation may vary from manual data input, the manufacture item read through QR Code or through the manufacturing items through the RFID readers. All these processes can be grouped according to the request related to the process of obtaining data.

## III. Expected Results

Because this article is a work in progress, the expected final result is establishing a list of indicators connected to the manufacturing and applicable to human resources management.

As highlighted on Section II, the process of measurement of the current situation is necessary to execute the production planning of a factory. Using, as focus, sectors that depend on human power to develop its part, there is a need to measure the performance of not only the processes, but also of human resources. Nowadays, there is extensive literature related to measuring the development of processes of manufacture, but our objective is to execute the performance measurement of the human resources connected to the manufacturing processes, being the human resources, in some processes of manufacturing, essential. The proposal of data warehouse that we will have by the end of the project must gather all clients needs, that is, the level of granularity must be from a managing level to an operational level. Considering that the main goal in defining these indicators is waste reduction (employee) and faster delivery of the final product to the client, we must consider these characteristics on the data warehouse definition (taking into account the dimensions). Initially, the following pattern can be used for the definition of the data warehouse (this will suffer alterations as the work evolves):

1) Dimensions: It will be used to define the granularity of data in the moment of the indicators exhibition.
2) Measurements: It will be used to define the granularity of data in the moment of the exhibition of the indicators.

For proving the work efficiency executed, the indicators and the model of data warehouse will be applied to the process of manufacturing of a software house (where the data is collected through a managing system of software development management). Other clients will be used for validation of the projects efficiency. The company responsible for the project development counts with about 20 thousand clients from many different areas. The article finds itself at this moment on the conclusion step of the definition of which indicators will be developed. The technical definitions concerning the data warehouse developed until now are described in this article.

## IV. State of the Art

In order to write this article, a research was made to find other existent articles that talked about the same topics of this work in progress. The more important ones are listed below:

1) Lupu, Bloga, Sabau and Muntean [10]: An article that details the development of real projects of business intelligence in an ERP system. The article defines some success factors for BI projects integrated to ERP systems.
2) Petterson [11]: An article that defines the concepts, methods and results of lean manufacture and also veries the differences between other processes popularly known as factory management.
3) George [12]: It presents concepts related to lean manufacture and shows how companies can use them to reduce waste and increase competitive advantage considering the current global economical situation.

The written works used in this article will help to develop this project in a way that each of the mentioned references is able to connect in a specic way to the work to be produced.

In [12], George details the Six Sigmas of lean manufacture and how to implement them in order to prevent the company from having a financial crisis (as the one that happened in 2008), the work in progress detailed in the article possesses a work line very similar to the one developed in [12], the difference is that this work in progress focus on definition and complementation, where lean manufacture indicators are related to people management. Choosing the topic of this work in progress was inspired by the needs of the market to implement indicators for people management as a solution for the ones already existent in people management. Another motivating factor is the lack of written works in this area. By the end of this article, we will have the definition of a data warehouse ambiance to contemplate lean manufacture indicators related to people management. The main difference between this article and the ones mentioned above are the definition of lean manufacture and how to apply them in a software of people management in the Brazilian market.

## V. Conclusion

This article briefs objectives and activities that are intended to be achieved with this project. This way, partially, we conclude that this paper will help defining the indicators to be used by people management systems. Using manufacture indicators as background, this article identifies manufacturing indicators applicable to people management, with the goal of eliminating waste, remembering that this objective is continuous on implementing lean manufacturing. With the absence of literature with this level of focus on the subject, this article has the objective of gathering academic material to this area of great importance on industries.

## Acknowledgment

We would like to thank all the people who helped directly and indirectly the development of this work so far.

## References

[1] B. Carreira, *Lean manufacturing that works: powerful tools for dramatically reducing waste and maximizing profits*. Amacom Books, 2005.

[2] S. J. Curt Hibbs and M. Sullivan, *The art of lean software development*. O' Reilly, 2009.

[3] S. Shingo, *A Study of the Toyota Production System*. Productivity Press, 1989.

[4] C. Vercellis, *Business Intelligence: Data Mining and Optimization for Decision Making*. John Wiley and Sons, 2009.

[5] M. McClellan, "The heart of intelligent manufacturing," 2004. [Online]. Available: http://www.informationweek.com/news/software/bi/22103211

[6] J. X. Wang, *Lean Manufacturing: Business Bottom-Line Based*. CTC Press, 2010.

[7] W. A. Levinson and R. A. Rerick, *Lean Enterprise: A Synergistic Approach to Minimizing Waste*. ASQ, 2002.

[8] V. J. Gawron, *Human Performance Measures Handbook*. CRC Press, 2000.

[9] J. Bancaleiro, "Indicadores tradicionais de recursos humanos," *Seminar HR Metrics - IIR, Lisboa*, 2006.

[10] A. R. Lupu, R. Bologa, G. Sabau, and M. Muntean, "Influence factors of business intelligence in the context of erp projects," *International Journal of education and Information Technologies*, 2007.

[11] J. Pettersen, "Defining lean production: Some conceptual and practical issues," *International Journal of education and Information Technologies*, 2009.

[12] M. O. George, *The lean six sigma guide to doing more with less : cut costs, reduce waste, and lower your overhead*. John Wiley  Sons, Inc., Hoboken, New Jersey, 2010.

# Improving the Quality of Knowledge Discovery Process by Information Gain Computing

Dumitru Dan Burdescu, Marian Cristian Mihăescu
Software Engineering Department
University of Craiova
Craiova, Romania
{mihaescu, burdescu}@software.ucv.ro

Bogdan Logofatu, Costel Marian Ionaşcu
CREDIS Department /
Analysis and Statistics Department
University of Bucharest/Craiova, Romania
logofatu@credis.ro, icostelm@yahoo.com

***Abstract*— Knowledge discovery process is one of the key activities in improving the quality of a system. This paper presents a custom approach for improving the quality of a knowledge discovery process based on information gain computing. The baseline knowledge discovery process is based of M Trees and is used to cluster learners from an e-Learning environment based on parameters representing performed activities. The baseline process is improved by assigning weights to each parameters according with the information gain computed for each parameter.**

***Keywords- knowledge discovery; M Tree; information gain; e-Learning.***

## I. INTRODUCTION

This paper addresses the problem of building a higher quality knowledge discovery process for an e-Learning platform. The knowledge discovery process is based on data that represent activities performed by learners in an e-Learning environment. The learners are distributed into clusters using an M Tree structure. The items that are involved in the process are represented by learners and each learner is described by a set of parameters. The creation of the hierarchical structure (in this case the M Tree) is based on the notion of distance between items. A trivial perspective is to normalize the parameters and to assign equal importance (weight) for each parameter. From the data analyst point of view this approach simplifies the analysis procedure but the obtained knowledge may not reflect the data from qualitative point of view. This paper introduces the concept of weight for all the features that describe the instances. The paper introduces the concept of weight as an automatically objective computed value. Computing a weight for each parameter may represent an overhead for the analysis process but the obtained knowledge may have more contextual value. Under these circumstances, the main issue becomes the procedure of assigning weights to parameters. Computing the weight is based on entropy and information gain computing for each defined feature. After the information gain is computed for each feature, a proportional weight is assigned for each feature. Therefore, the classical Euclidian distance formula between items becomes a weighted one.

The obtained clusters represent in a more realistic manner the input data since each parameter is weighted according with the amount of knowledge it brings into the data.

The second section presents the related work in the field. The third section presents the employed infrastructure and methods. Section four presents the analysis process and section five presents a sample experiment. The final section presents the conclusions and future works.

## II. RELATED WORK

One domains that is discussed in this paper is clustering as state of the art machine learning methodology. Here, the clustering quality computation as a main tool in assessing the quality of the knowledge discovery process is discussed. The second domain regards information theory and is concerned with entropy and information gain as a mechanism for determining the weight of each parameter that describes data items. These two domains are put together in a framework that aims at improving the quality of knowledge discovery process for an e-Learning application [1].

Clustering is a machine learning technique used to group a set of items into subsets. This technique may be used in educational domain to enhance our understanding of learning process to focus on identifying, extracting and evaluating variables related to the learning process of students [2]. K-means clustering is a widely used method that is easy and quite simple to understand [3]. Cluster analysis describes the similarity between different cases by calculating the distance. These cases are divided into different clusters due to their similarity. There are studies [4] that use students data to analyze their learning behavior to predict the results and to warn students at risk before their final exams. The theoretical background of k-means is presented in [5]. This well-known clustering algorithm tends to uncover relations among variables already presented in dataset and is implemented by tools and libraries [6, 7].

M Trees [8] are spatial data structures that may be used for clustering data that is described by a set of parameters. The main drawback of this approach is that even though it is able to employ more features during the search, these features are compared using a single distance function. Another drawback regards the fact that the distance function does not make any difference between parameters. This drawback is discussed in this paper by computing the information gain for each parameter. An extension of the M tree [9], which goes further, is able to compare different features with arbitrary distance functions. This approach is used in general in situations when a custom query mechanism based on multiple features are required. In general, the output of the procedure is represented by computed centroids for each obtained cluster. An item belongs to one cluster or another according with the minimum distance to a centroid. This approach is used when the goal of the procedure is just to cluster a new test instance. Still, more complex queries (range or k-nearest neighbor) may also be set up. In this scenario, we may be interested in finding all the items that reside in a certain range or the closest k items.

When a clustering algorithm is used, it is necessary to test its performance, and compare it with that of other methods. Such comparisons are difficult to be performed but the effort is necessary because the quality of the clustering process must be assessed. The commonly used approach uses a benchmark that implements a set of quality assessment metrics.

Finally, the information gain is computed for each attribute that describes the items from the input dataset. The computing is based on entropy computing as an average measure of information [10, 11].

## III. Employed Infrastructure and Methods

### A. The e-Learning Environment

E-Learning systems are mainly concerned with delivery and management of content (e.g., courses, quizzes, exams, etc.). Since we are speaking about a web platform the client is represented by the browser, more exactly by the learner that performs the actions.

Defining the e-Learning infrastructure or the presented purpose represents the first and the most important step. In this phase, all the possible actions that may be performed by a learner need to be presented. The resources that are delivered by the e-Learning system are also identified. Finally, there are identified the highly complex business logic components that are used when actions are performed by learners.

Each implemented action needs to have an assigned weight. In the prototyping phase, the assignment of weights is performed manually according with a specific setup. This assumes that we have an e-Learning system that is already set up. The main characteristics regard the number of learners, the number of disciplines, the number of chapters per discipline, the number of test/exam questions per chapter and the dimension of the document that is assigned to a chapter. The data that is obtained from analyzing a certain setup will represent the input data for the simulation procedure.

Another type of activities regarding learners are represented by the communication that takes place among parties. Each sending or reading of a message is assigned a computed average weight.

A sample e-Learning setup infrastructure may consist of 500 students, 5 disciplines, 5 to 10 chapters per discipline, 10 to 20 test/exam questions.

For this infrastructure here may be established a list of costs for all needed actions that may be performed by learners. The weight assigned to an action takes into consideration the complexity of the action and the dimension of the data that is obtained as response after the query is sent.

For obtaining reasonable weight, a pre-assessment procedure is performed. The simulation tool performs this procedure from a computer that resides in the same network as the server such that response times are minimal. Each request that is composed and issued to the e-Learning platform is measured in terms of time and space complexity. A scaling factor will assign each action a certain weight such that the scenarios that will be created when real time testing starts will have a sound basis.

The pre-assessment procedure firstly loads all the data regarding the analyzed e-Learning platform. This means the data about all managed resources (e.g., disciplines, chapters, quizzes, etc.) are loaded such that the simulation tool may build valid requests for the e-Learning environment.

### B. Clustering with k-means algorithm

K-means is the most important flat clustering algorithm. Its objective is to minimize the average squared Euclidean distance of items from their cluster centers where a cluster center is defined as the mean or centroid #μ of the items in a cluster ω: centroid.

$$\bar{\mu}(\omega)=\frac{1}{|\omega|}\sum_{\bar{x}\epsilon\omega}\bar{x} \qquad (1)$$

The algorithm is:

```
procedure k-means ($x_1$, $x_2$, …, $x_N$; K)
 {$c_1$, $c_2$, …, $c_K$} ← Select Random Centroids
 for ( k=1, k<K )
   centroid$_k$ = $c_k$;//these are initial centroids
 while (#centroids are not same){
  for ( k=1, k<K ){
   for ( n=1, n<N ){
     j = index of corresponding cluster
     #put $x_n$ in corresponding cluster Cj
    }//end for
  }//end for
  for (k=1, k<K)
    # compute centroids for all clusters
 }//end while
```

In most cases, K-means quickly reaches either complete convergence or a clustering that is close to convergence. In the latter case, a few items would switch membership if further iterations are computed. This computation has a small effect on the overall quality of the clustering.

### C. M Trees

The classical M Tree algorithm has been adapted such that is the final structure has two levels. The procedure for building the structure takes into consideration both the desired number of clusters and the filling factor of a leaf node.

```
procedure MTree ($x_1, x_2, \ldots, x_N$; K; F)
// K – the number of clusters
// F– filling factor
 for ( i=1, i<N ){
   $C_i$ = FindCentroid(centroids, $x_i$);
   if ( #Leaf[$C_i$] has F instances)
     if (#we have k clusters)
       #put $x_i$ in Leaf[$C_i$]
     else
       #split Leaf[$C_i$]
   else
     #put $x_i$ in Leaf[$C_i$]
   RecomputeCentroids(Leaf[$C_i$])
 }//end for
```

### D. Information Gain

Information gain is calculated using a measure called entropy, which we first define for the case of a binary decision problem and then define for the general case.

Given a binary categorization, **C**, and a set of examples, **S**, for which the proportion of examples categorized as positive by **C** is *p+* and the proportion of examples categorized as negative by **C** is *p-*, then the **entropy** [5] of **S** is:

$$\mathrm{Entropy(S)} = -\mathrm{p}_+ \log_2(\mathrm{p}_+) - \mathrm{p}_- \log_2(\mathrm{p}_-) \quad (2)$$

The reason we defined entropy first for a binary decision problem is because it is easier to get an impression of what it is trying to calculate.

Given an arbitrary categorization, **C** into categories **c1, ..., cn**, and a set of examples, **S**, for which the proportion of examples in $\mathbf{c_i}$ is $\mathbf{p_i}$, then the entropy of **S** is:

$$\mathrm{Entropy(S)} = \sum_{i=1}^{n} -\mathrm{p_i} \log_2(\mathrm{p_i}) \quad (3)$$

We now return to the problem of trying to determine the best attribute to choose for a particular node in a tree. The following measure calculates a numerical value for a given attribute, **A**, with respect to a set of examples, **S**. Note that the values of attribute **A** will range over a set of possibilities which we call **Values(A)**, and that, for a particular value from that set, ***v***, we write ***Sv*** for the set of examples which have value ***v*** for attribute ***A***.

The information gain [5] of attribute ***A***, relative to a collection of examples, ***S***, is calculated as:

$$\mathrm{Gain\ (S, A)} = \mathrm{Entropy(S)} - \sum_{v \in \mathrm{Values(A)}} \frac{|\mathrm{S_v}|}{|\mathrm{S}|} \mathrm{Entropy(S_v)} \quad (4)$$

The information gain of an attribute can be seen as the expected reduction in entropy caused by knowing the value of attribute ***A***.

### E. Assessing Clustering Quality

The indicators [12] that are taken into considerations are:

***Tightness Indicator:***

$$\mathrm{Q} = \sum_{i=1}^{k} \frac{1}{|\mathrm{C_i}|} \sum_{x \epsilon \mathrm{C_i}} \mathrm{d}(\mathrm{x}, \mu_i) \quad (5)$$

where $|C_i|$ is the number of points from cluster i. The value for Q will be small if the data points from the cluster are close. Thus, in the comparison analysis procedure the clusters with smaller computed values of Q have higher quality.

***Homogeneity Indicator:***

If the centroids of clusters are computed with formula: $r_k = \frac{1}{n_k} \sum_{x \epsilon C_k} x$, where x are the instances from cluster $C_k$ than homogeneity indicator is:

$$\mathrm{H(C)} = \sum_{k=1}^{K} \sum_{x \epsilon C_k} \mathrm{d}(\mathrm{x}, \mathrm{r_k})^2 \quad (6)$$

The value for H will be small if a cluster has homogeneous structure. This, in the comparison analysis procedure the clusters with smaller computed values of H, have higher quality.

***Cluster Distance:***

$$CD = \sum_{1 \le j \le k \le K} d(r_j, r_k)^2 \quad (7)$$

where j and k are indexes of clusters whose centroids r are taken into consideration. The value for CD will be big if the similarity among clusters themselves is low. Thus, in the comparison analysis procedure the method with bigger computed values of CD have higher quality.

***Weakest Link between Points***:

The weakest link for a cluster is the maximal value of all pairs of points belonging to the same cluster.

$$\mathrm{WL} = \max(\mathrm{d}(x_i, x_j)) \quad (8)$$

for all $x_i$ and $x_j$ belonging to the same cluster.

## IV. Analysis Process

The analysis process uses a dataset of 150 students represented by seven attributes.

The parameters that characterize each instance are:

*positiveCount* – represents the number of correctly answered questions;

*correctPercent* – represents the percentage of correctly answered questions from the total number of questions;

*totalTries* – represents the total number of tries (answered questions);

*avgTries* – represents the medium number of tries per question;

*avgQuestionTime* – represents, on average, how long (in minutes) it takes for a student to answer a question;

*totalTime* – represents the total time spent on testing;

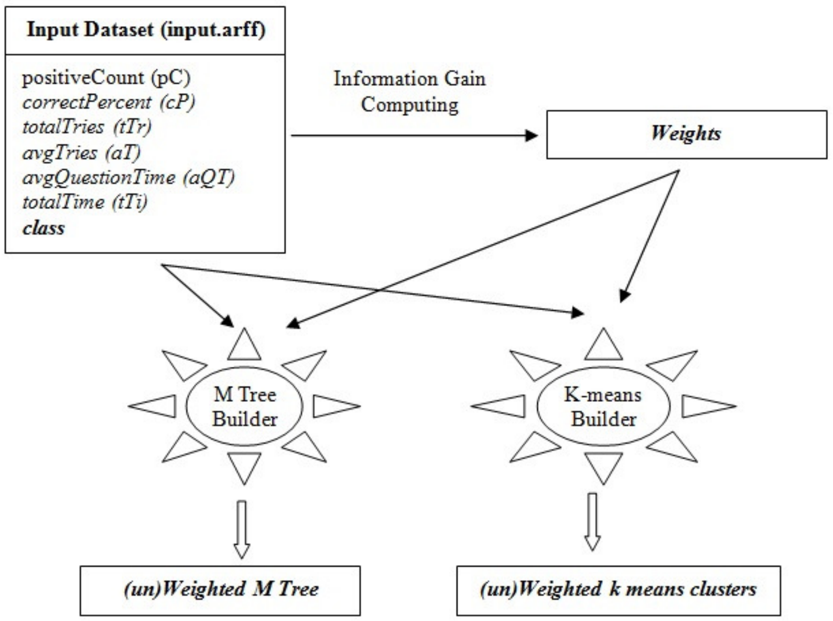


Figure 1. The architecture of the analysis framework

The last attribute represents the class and is not used during the clustering procedures. The class of the student may be *low*, *average* or *high*. The *class* attribute makes possible the usage of input data in an supervised learning context. This is necessary for computing the information gain brought by each attribute. This value is set by the data analyst which has also the domain knowledge. Still, the k-means and M Tree algorithms are unsupervised methods and thus the class of each item is not taken into consideration.

Figure 1 presents the analysis process. It may be observed that the input dataset is represented by *input.arff* file. In this file resides the data regarding the activity performed be learners that is used for computing the information gain for each attribute. After the information gain is determined for each attribute the weight of each attribute may be determined. Once the weights are determined, they may be used to build the weighted M Tree and weighted k-means clustering. These weighted models along with un-weighted models are thereafter analyzed by the clustering quality metrics.

The overall idea of the knowledge discovery process is to assess the quality of the obtained model in each analyzed situation.

This setup uses only normalized and continuous type parameters and a final nominal class attribute. The appearance of the class attribute allows a supervised approach on input data with the possibility of computing the information gain for each attribute. Once the class attribute is removed the learning becomes unsupervised and a clustering procedure (e.g., k-means, M Tree) may be used. The class variable is set by a domain knowledge data analyst for real life examples and thus there is no clear (mathematical) prior dependency between this variable and the rest of variables.

This approach makes the experiment to have real consistency regarding the learning process.

The analysis process is used in an e-Learning application for clustering students. All the students that have followed and finished courses represent the training set for the analysis process. The data coming from a new student is used as test data. The new student is clustered, which means he is assigned to the cluster which has the minimum distance from its centroid to the instance itself. Once the student in clustered the target cluster may be determined, as the cluster with the next increasing knowledge weight. A recommender system may use these data to determine the features in which the current student needs improvements. In a more general approach, each educational resource may represent a feature and thus the educational resources that need more attention may be determined.

## V. SAMPLE EXPERIMENT

The goal of the experiment is to prove the concept and to objectively describe the results. The experiment uses an input dataset which there data for 50 students. The *input.arff* file has the following structure:

```
@RELATION activity

@ATTRIBUTE positivCount NUMERIC
@ATTRIBUTE correctPercent NUMERIC
@ATTRIBUTE totalTries NUMERIC
@ATTRIBUTE avgTries NUMERIC
@ATTRIBUTE avgQuestionTime NUMERIC
@ATTRIBUTE totalTime NUMERIC
@ATTRIBUTE class {low, avg, high}
```

```
@DATA
75,90,80,19,45,87,high
85,65,71,10,25,92,high
41,59,67,75,31,56,avg
…
```

All numeric attributes have real continuous values that are normalized in the range of 0 to 100. The normalization of the numeric values is performed with the following formula:

$$x_i = (\text{value}_i - \text{mean})/\text{range} \tag{9}$$

where:

- *value* is the initial value of the feature;
- *mean* is the average value from all values of the feature in the training set;
- *range* is the difference between maximum value of the feature and minimum value of the feature;
- $x_i$ is the normalized computed value of the feature;

The first step is to build the k-means and M Tree models. At this step there are taken into consideration only the six numeric attributes. The obtained clusters by k-Means clustering have the following centroids and composition:

**C1 (3, 5, 10, 28, 32, 45)** *//Cluster 1's Centroid*
{12 instances}
**C2 (34, 42, 56, 78, 62, 58)** *//Cluster 2's Centroid*
{18 instances}
**C3 (61, 75, 85, 69, 88, 69)** *// Cluster 3's Centroid*
{20 instances}

The obtained clusters by M Tree clustering have the following centroids and composition:

**C1 (4, 6, 9, 31, 28, 41)** *//Cluster 1's Centroid*
{14 instances}
**C2 (37, 40, 52, 80, 63, 62)** *//Cluster 2's Centroid*
{19 instances}
**C3 (65, 77, 82, 83, 89, 72)** *// Cluster 3's Centroid9*
{17 instances}

For each clustering procedure there were computed the evaluation metrics presented in section three. The results are presented in the following table:

TABLE I. TIGHTNESS, HOMOGENEITY AND CLUSTER DISTANCE INDICATORS FOR K-MEANS AND M TREE DISTRIBUTIONS

| Indicator | Clustering Procedure | |
|---|---|---|
| | *k-means* | *M Tree* |
| Tightness | 7.80 | 8.85 |
| Homogeneity | 105.29 | 138.55 |
| Clusters Distance | 220.32 | 203.11 |

The link analysis for both distributions is presented in the following table:

TABLE II. WEAKEST LINK VALUES OBTAINED FOR K-MEANS AND M TREE DISTRIBUTIONS

| Indicator | Clustering Procedure | |
|---|---|---|
| | *k-means* | *M Tree* |
| Weakest Link Cluster 1 | 0.92 | 1.25 |
| Weakest Link Cluster 2 | 0.88 | 1.25 |
| Weakest Link Cluster 3 | 0.89 | 0.57 |

The k-means results are obtained using Weka [6]. Weka is a collection of machine learning algorithms for data mining tasks which has implemented the k-means clustering algorithm.

All results presented so far do not take into consideration any weight of parameters. Thus, information gain is computed for each parameter and a normal distribution of weights is determined.

Firstly, the entropy is computed:

Entropy(S) = $-p_{high} \log_2(p_{high}) - p_{avg} \log_2(p_{avg}) - p_{high} \log_2(p_{high}) = -(14/50) * \log_2(14/50) - (20/50) * \log_2(20/50) - (16/50) * \log_2(16/50) = -(14/50) * -1.83 - (20/50) * -1.32 - (16/50) * -1.64 = 0.51 + 0.52 + 0.52 = 1.55$

For computing the information gain of each parameter all continuous values are transformed into nominal values of *low*, *average* and *high* using a normal distribution.

Gain(S, positiveCount) = $1.55 - (|S_{low}|/50)*\text{Entropy}(S_{low}) - (|S_{avg}|/50)*\text{Entropy}(S_{avg}) - (|S_{high}|/50)*\text{Entropy}(S_{high}) = 1.55 - (0.3)*\text{Entropy}(S_{low}) - (0.4)*\text{Entropy}(S_{avg}) - (0.3)*\text{Entropy}(S_{high}) = 1.55 - (0.3)*(0.91) - (0.4)*(0.81) - (0.3)*(0.92) = 0.677$

In a similar way, there is computed the information gain for all other parameters.

Gain(S, correctPercent) = 0.57
Gain(S, totalTries) = 0.88
Gain(S, avgTries) = 0.56
Gain(S, avgQuestionTime) = 0.38
Gain(S, totalTime) = 0.77

The formula for computing the corresponding weight for a feature takes into account the overall gain brought by all features. In general, if there are defined m features, the overall gain is defined as the following sum:

$$AllGain = \sum_{i=1}^{m} Gain\ (S, f_i) \tag{10}$$

The formula for computing the value of the weight for a certain feature $f_i$ is:

$$W_{f_i} = \frac{Gain(S,f_i)*100}{AllGain} \tag{11}$$

According to the values obtained by computing the information gain the weight for each parameter is:

$W_{positiveCount} = 17.4$
$W_{correctPercent} = 14.8$
$W_{totalTries} = 22.97$
$W_{avgTries} = 14.62$
$W_{avgQuestionTime} = 9.92$
$W_{totalTime} = 20.10$

The obtained weights are used when computing the Euclidian distances between items in the process of building the M Tree structure and the k-means clusters.

Now, it time to rebuild the k-means and M Tree models taking into consideration the above computed weights. The obtained clusters by k-Means clustering have the following centroids and composition:

**C1 (4, 5, 9, 30, 31, 7)** *//Cluster 1's Centroid*
{14 instances}

**C2 (37, 45, 52, 75, 65, 62)** *//Cluster 2's Centroid*
{17 instances}

**C3 (60, 72, 87, 68, 81, 72)** *// Cluster 3's Centroid*
{19 instances}

The obtained clusters by M Tree clustering have the following centroids and composition:

**C1 (5, 7, 10, 17, 26, 37)** *//Cluster 1's Centroid*
{13 instances}

**C2 (34, 40, 52, 77, 63, 59)** *//Cluster 2's Centroid*
{16 instances}

**C3 (62, 78, 69, 87, 81, 79)** *// Cluster 3's Centroid9*
{21 instances}

For each clustering procedure there were computed the evaluation metrics presented in section three. The results are presented in the following table:

TABLE III. TIGHTNESS, HOMOGENEITY AND CLUSTER DISTANCE INDICATORS FOR K-MEANS AND M TREE DISTRIBUTIONS

| **Indicator** | **Clustering Procedure** | |
|---|---|---|
| | ***k-means*** | ***M Tree*** |
| Tightness | 7.85 | 9.25 |
| Homogeneity | 107.35 | 141.55 |
| Clusters Distance | 225.32 | 201.15 |

The link analysis for both distributions is presented in the following table:

TABLE IV. WEAKEST LINK VALUES OBTAINED FOR K-MEANS AND M TREE DISTRIBUTIONS

| **Indicator** | **Clustering Procedure** | |
|---|---|---|
| | ***k-means*** | ***M Tree*** |
| Weakest Link Cluster 1 | 0.95 | 1.22 |
| Weakest Link Cluster 2 | 0.91 | 1.33 |
| Weakest Link Cluster 3 | 0.93 | 0.78 |

The M Tree results are obtained using a custom Java implementation of the algorithm. The main differences of this implementation compared with classical M Tree algorithm regards two aspects. One regards the general structure of the tree that is restricted to two levels. This means there is only one root node where centroids along with covered radius are placed. The second issue regards the way k (the number of clusters) and f (the filling factor) are managed. If the algorithm is required to produce a certain number of clusters, the instances are placed into appropriate clusters until a filling factor is reached. When this happens, a split is performed. Splitting is no longer performed when the desired number of clusters is reached. In this way, the clustering process is directly managed by the values k and s.

The comparison of the two obtained distributions reveals the fact that the M Tree distribution clusters have lower quality than the ones obtained by usage of k-means. Still, the results obtained by M Tree are very different from the ones obtained by k-means. All indicators presented in table 1 have better results for k-means than the ones obtained for M Tree. It can be observed that the tightness and homogeneity are better (because they have smaller values) for k-means than for M Tree.

The results obtained when the attributes are weighted show a similar quality with the ones obtained without weights. Still, the models are quite distinct and we think the one in which attributes are weighted is a more realistic one.

The clustering process belongs to the class of unsupervised learning schemes that tries to obtain patterns in the training dataset. One common approach used to enhance the learning scheme is feature scaling and/or mean normalization. The presented approach of assigning weights features is custom to e-Learning domain but may adapted in any learning process. Intuitively, the main reason for this approach is that features that characterize an instance may not all have the same importance or significance. One approach might be to have a domain expert assign a weight value for each feature. This may be regarded as a manual configuration of the learning scheme. In this paper, we use an automatic approach. This means that the weights are set according with the information gain provided by each feature. That is why the obtained patterns have the chance of being more realistic since they are determined in an objective way in correspondence with the provided dataset. The fact that the quality of the both clustering schemes, weighed and un-weighed, are similar means that both obtained models may be used with confidence. Still, the weighted model is different from the un-weighted model in the way that the obtained values of coordinates for centroids are different. Taking into account that the weights were automatically and objectively determined, the weighted model is a more realistic one.

## VI. CONCLUSION AND FUTURE WORK

This paper presented a procedure that measures the degree in which the effectiveness of on e-learning process has improved. The analysis process is data centered. The data represents experiences provided by learners. In this study, six features (attributes) characterize each learner.

The study is repeated with weighted attributes. The weights are proportional with the information gain produced by each attribute. The information gain is computed as the difference between system's entropy and the entropy of the system when each attribute is taken into consideration.

The goal of the procedure is to produce clusters of users using two different techniques: standard k-means algorithm implemented in weka and a custom flavor of M Tree algorithm with a custom implementation.

The input dataset is restricted to a sample of 50 learners. An automated analysis of the obtained clusters is performed by computing some basic clustering quality metrics: Tightness, Homogeneity, Clusters Distance and link analysis. The obtained results show an acceptable quality of the M

Tree clusters although the computational complexity of the algorithm is much lower than complexity of k-means.

The main goal of the paper is to find a more realistic knowledge discovery process that obtains acceptable results with complexity much smaller than a classical procedure.

The quality of the obtained clusters has a direct influence over the degree in which the e-learning process has been performed. Unsupervised classification (clustering) is one of the main methods for making evidence regarding the knowledge acquisition of learners. Once a high quality distribution has been discovered a learner may by clustered at certain moments and progress may be evaluated. Of course, the process needs to be well defined and needs to be based on a high quality clustering procedure.

The future works regard different aspects. A first issue would be to replicate the procedure with more data. This may be accomplished on hundreds or even thousands of learner, if data is available. The clustering procedure is highly influenced by the initial centroids. In custom initialization is advisable. A good starting point may be obtained by using a k-means clustering on a sample dataset from the entire dataset. The quality of the clustering process is directly influenced by the choices made regarding k and f values. Thus, an initialization step may also refer to prior computation of the optimal number of clusters and optimal filling factor. The computation of these parameters may be delegated to other high quality clustering procedure that works on a data sample.

Finally, there may be defined procedures for assessing progress in time and even recommendations. The progress in time may be computed classifying the learner from time to time. This may yield to a learning path that has been followed by the learner. More than this, there may be obtained recommendations for the learner. The recommendations may regard necessary actions necessary to be taken by the learner in order to improve his learning curve.

## REFERENCES

[1] C. Romero, S. Ventura, A. Zafra, and P. de Bra, "Applying Web Usage Mining for Personalizing Hyperlinks in Web-based Adaptive Educational Systems", Computer&Education, Elsevier, Volume 53, Issue 3, pp. 828-840, 2009.

[2] A. El-Halees, "Mining Students Data to Analyze e-Learning Behavior: A Case Study", Department of Computer Science, Islamic University of Gaza, 2009.

[3] K.S. Qaddoum, "Mining student evaluation using associative classification and clustering", Communications of the IBIMA vol. 11 IISN 1943-7765, 2009.

[4] G. Ben-Zadok, A. Hershkovitz, R. Mintz, and R. Nachmias, "Examining online learning processes based on log files analysis: a case study", Research, Refelection and Innovations in Integrating ICT in Education, 2007.

[5] J. Han and M. Kamber, "Data Mining: Concepts and Techniques", 2nd edition, The Morgan Kaufmann Series in Data Management Systems, Jim Gray, Series Editor, 2006.

[6] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA Data Mining Software: An Update", SIGKDD Explorations, Volume 11, Issue 1, 2009.

[7] http://www.ibm.com/developerworks/java/library/j-mahout/ "Introducing Apache Mahout", ibm.com. 2011 [last update]. Retrieved 13 September 2011.

[8] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: An efficient access method for similarity search in metric spaces", in Jarke, M., Carey, M. J., Dittrich, K. R., Lochovsky, F. H., Loucopoulos, P., and Jeusfeld, M. A., editors, Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB 1997), Morgan Kaufmann, pp. 426-435, 1997.

[9] P. Ciaccia and M. Patella, "The M2-tree: Processing complex multi-feature queries with just one index", In Proceedings of the First DELOS Network of Excellence Workshop on Information Seeking, Searching and Querying in Digital Libraries, 2000.

[10] W. Baobao, M. Jinsheng, and S. Minru, "An enhancement of K-Nearest Neighbor algorithm using information gain and extension relativity", in International Conference on Condition Monitoring and Diagnosis (CMD2008), pp.1314-1317, 2008 .

[11] M. H. Dunham, "Data Mining: Introductory and Advanced Topics", Prentice Hall, 2003.

[12] P. Zezula, G. Amato, V. Dohnal, and M. Batko, "Similarity Search-The Metric Space Approach", Advances in Database Systems, Vol. 32, Springer, 2006.

# Education Cluster for Intelligent Provision of eLearning Services

Stanimir Stoyanov
Department of Computer Systems
University of Plovidiv “Paisii Hilendarski”,
Plovdiv, Bulgaria
stani@uni-plovdiv.bg

Emil Doychev
Department of Computer Systems
University of Plovidiv “Paisii Hilendarski”,
Plovdiv, Bulgaria
e.doychev@isy-dc.com

Veselina Valkanova
Department of Computer Systems
University of Plovidiv “Paisii Hilendarski”,
Plovdiv, Bulgaria
veselinaviva9@gmail.com

Georgi Cholakov
Department of Computer Systems
University of Plovidiv “Paisii Hilendarski”,
Plovdiv, Bulgaria
gcholakov@uni-plovdiv.bg

***Abstract* - The DeLC (Distributed eLearning Center) project aims to develop an interactive, proactive and personalized e-Learning environment. Two nodes of DeLC architecture are presented in the paper. The first node is a service-oriented portal, providing personalized educational services and teaching material. The second node is an agent-oriented server which incorporates three agents supporting electronic testing of students. This paper focuses on the hybrid architecture, known as education cluster, which integrates the two nodes. The main results of the use of the cluster are increased interest and more active inclusion of the students in the education process.**

***Keywords – service-oriented architectures; intelligent agents; e-learning.***

## I. INTRODUCTION

Recently, the interest towards electronic education is growing. As a result, many universities developed and implemented their own systems for electronic and long-distance education. Alternatively, many of the large IT corporations (e.g., Microsoft Class Server [1], IBM [2], and HP [3]) developed commercial systems. On the other hand, there are different open-source systems available on the market (the best known is Moodle [4]). A number of standards for electronic and life-long learning are also emerging. There are many organizations working to develop specifications and standards such as Innovation Information Management System Global Learning Consortium [5], Advance Distributed Learning [6], IEEE, ISO, etc., to provide a framework for e-Learning architectures, to facilitate interoperability, content packaging, content management, Learning Object Meta data, course sequencing, and others. One significant example is played by the Sharable Content Object Reference Model 2004 (SCORM 2004) standard [7].

DeLC (Distributed eLearning Center) [8, 9] is one of the projects aiming to develop an environment that supports electronic and long-distance forms of education. Why, despite the presence of so many systems, do we find it necessary to dwell on this subject? DeLC tries to provide architectural support for more effective eLearning systems which involve the students in a more personalized and creative education process and which stimulate students’ activities and cooperation. An important goal of this project is the development and experimentation with prototypes of such architectures that are service-and-agent-oriented. To achieve it, we developed an environment which provides teaching materials and educational electronic services.

Furthermore, in many of the existing e-Learning systems, the interaction with the teachers is somewhat static – it is achieved mainly through pre-defined templates for choosing information resources. The information resources are the electronic equivalent of the traditional textbooks. Some of the existing systems use visualization and animation for improving the presentation of the teaching materials. In our project, we would like to research how such architectures can promote the development of electronic education environments, which support an interactive, reactive, proactive and personalized process of education and stimulate the students’ creative and innovative thinking and performance.

The DeLC is a network, consisting of separate nodes, called eLearning Nodes. Nodes model real units (laboratories, departments, faculties, colleges, and universities), which offer a complete or partial educational cycle. Each eLearning Node is an autonomous host of a set of electronic services. This network configuration enables access, incorporation, use and integration of electronic services located on the different nodes. The eLearning Nodes can be isolated or integrated in more complex virtual structures, called clusters. Remote eService activation and integration is possible only within a given cluster. In the network model, we can easily create new clusters, reorganize or remove existing ones. The reorganization is virtual and it does not affect the real organization. For example, the reorganization of an existing cluster can be made not by removing a node but only by denying access to the services it offers. The reorganization does not disturb other nodes' function as nodes are autonomous self-sufficient educational units providing one or more integral services. An important feature of the eLearning Nodes is the access to supported services and electronic content. In relation to the access there are two kinds of nodes – mobile access eLearning Node [10] and fixed access eLearning Node [11].

In this publication, we present the architecture of a standardized eLearning Node with fixed access, implemented as an education portal. Furthermore, an education cluster with hybrid (service-and-agent-oriented) architecture is described. Agents, which are resident in the cluster, are presented as well.

## II. EDUCATION PORTAL

The fixed node is implemented as an education portal with a service-oriented and multi-layered architecture, consisting of three logical layers: user interface, e-services and digital libraries (Fig. 1).

The user interface supports the connection between the users and the portal. It allows users to register in the system and create their own personalized educational environment. The user interface provides visualization and access to services, depending on the user's role which is assigned during the registration.

The e-services are classified in two groups: engines and eLearning services. The engines are invisible for the users and their main purpose is to assist in processing eLearning services. The engines support and manage the meta-data in the portal. Using the provided data, they can effectively support the activation, execution and completion of the eLearning services. In the portal architecture, the following engines are incorporated: SCORM Engine, Test Engine, Event Engine, Integration & User Profiling Engine and AV-Call Processor.

SCORM Engine is an interpreter of the electronic content, developed in accordance with the SCORM 2004 standard.

The Test Engine assists in performing electronic testing through the portal. It processes the meta-objects, which describe the questions and patterns of the tests.

The Event Engine supports a model for event management, enabling the users to see and create events and be notified for them in advance. The events in the system reflect important moments for the users, such as a lecture, examination, test, national holiday, birthday, etc. Each event is characterized by attributes, such as a name, start and end date and time, details, and information whether it is a recurring one, as well as rules for its recurrence. The Event Engine supports yearly, monthly and weekly recurrence.

The Integration & User Profiling implements the supported user model.

eLearning services are grouped in three categories:

- Services for training, organizing and planning of the educational process;
- Services for conduction and management of the educational process – here belong services as electronic lectures, electronic testing, online and offline consultations;
- Services for recording and documenting the educational process – they support automated generation of documents recording the educational process in the form of examination protocols, students' books, teachers' personal notebooks and archives).

The third layer contains electronic content in the form of digital repositories (libraries). In the current version, digital libraries are implemented for electronic lecture courses, questions for knowledge testing, electronic tests templates, course projects and diploma theses. The portal services can work directly with the digital libraries. Moreover, a generalized catalog representing the libraries' contents, is provided to the users.

With the implementation of the portal we aim at providing personalized eLearning services and teaching material. A multi-aspect model of personalization is implemented in the architecture. The first aspect is user classification and role organization. The users' profiles can be classified by roles, user groups, communities, and organizations. The standard user profile includes the following attributes:

- Standard attributes – necessary for user identification through username, password, e-mail, and others;
- Extended attributes – addresses, phone numbers, Internet pages, IM, social networks' contacts, and others;
- DeLC custom attributes – other user identifications. For example, for users with role "student" these can be faculty number, subject, faculty, and course.

The portal gives an opportunity for extending the user profile with some additional attributes. The users' profiles contain the complete information needed for personalization of the portal services provided by DeLC, educational content and user interface. The profile is created automatically during the user's first login, through a request to the university's database, after providing the standard and custom attributes. The integration with the university database and other external components is supported by the Integration & User Profiling Engine. Extended attributes are provided by the user. During the following user's logins the information in their profile is synchronized, as newer updates in the university's database are automatically migrated in the user's profile, for example change of course or subject.

The second aspect is structuring of the teaching material, which is saved in the digital libraries. The teachers have the freedom to specify different structural schemes according to the desired teaching approach.

The third aspect is personalization of the provided eLearning services. In the current portal version it is possible to generate individual tests. The student can use personalized schedules and calendars and individual student reports and reviews can be prepared.

## III. EDUCATION CLUSTER

In order to enhance the portal architecture so that it can provide eLearning services in a more intelligent manner, we are going to extend it with new intelligent components implemented as agents. In the same time we aim at a pro-active architecture. The pro-activity improves the system's usefulness and friendliness to the users. Pro-activity means that the software can operate "on behalf of the user" and "activate itself" when it "estimates" that its intervention is necessary. Pro-activity can be ensured through "reinforcement" of the portal architecture with intelligent components, which demonstrate proactive behavior. Two approaches are available:

- Direct integration of intelligent agents in the currently existing education portal;
- Building an education cluster according to the DeLC concept.

For different reasons the latter approach has been chosen where intelligent agents, called "assistants", will "live" in a newly-built agent-oriented server known as "Agent Village" (Fig. 2).

Finding solutions to the following three problems is important for the creation of an education cluster:

- What will the architecture of the Agent Village (AV) be?
- How will the communication between the service-oriented architecture of the portal and the agent-oriented architecture of the Agent Village be achieved?
- What about the particular assistants that will reside at the Agent Village?

### A. *Agent Village Architecture*

The Agent Village has to provide an environment where the assistants can operate in an intelligent manner. "Intelligent" suggests that the agents can expose interactive, reactive, proactive and personalized behavior in respect to users' requests.

Different information resources, which are not saved in the digital libraries, will be located in the agents' environment. In conformity with DeLC's multi-aspect concept three models have to be implemented as ontologies, in order to ensure the next aspect of personalization in the education cluster – student model, pedagogical model an domain (discipline) model.

### B. *Portal-AV Interaction*

The connection between the portal and the AV node is made through the middle layer of the portal architecture, where the electronic services are located. Depending on the direction of the assistance required we distinguish reactive and proactive behavior of the architecture.

In the reactive behavior, the interaction between the two nodes is initiated by the portal. This is necessary when a user request is processed and a service needs "expert" assistance. The service addresses the corresponding agent located in the AV. The problem is that the services are passive and static software modules in nature, intended

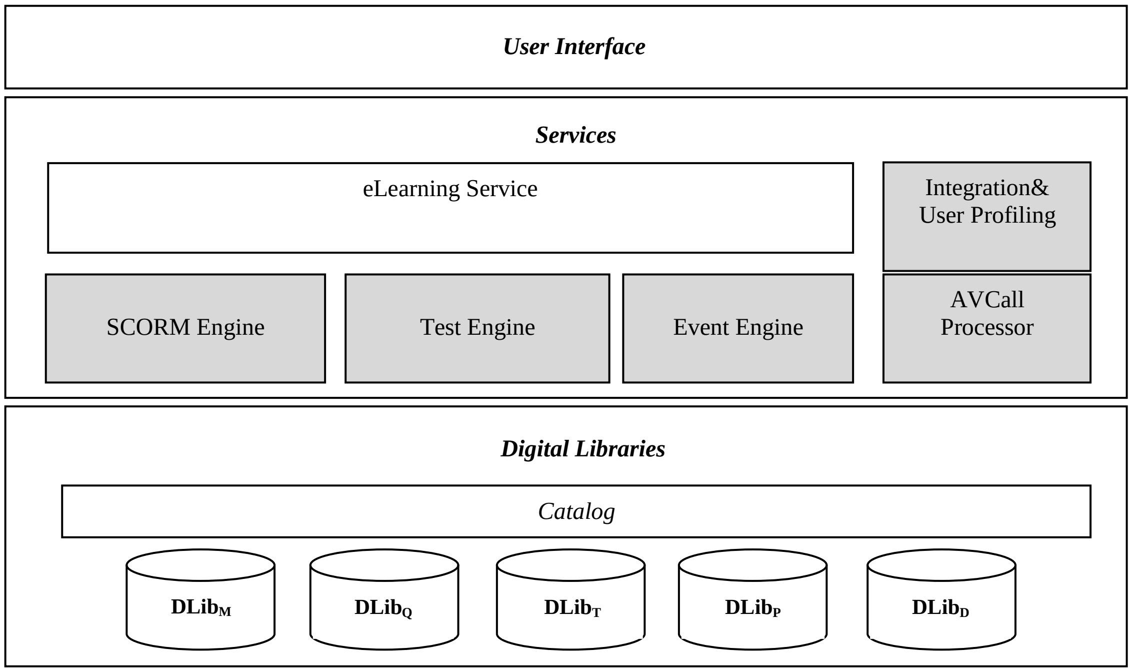


Figure 1. DeLC Education Portal.

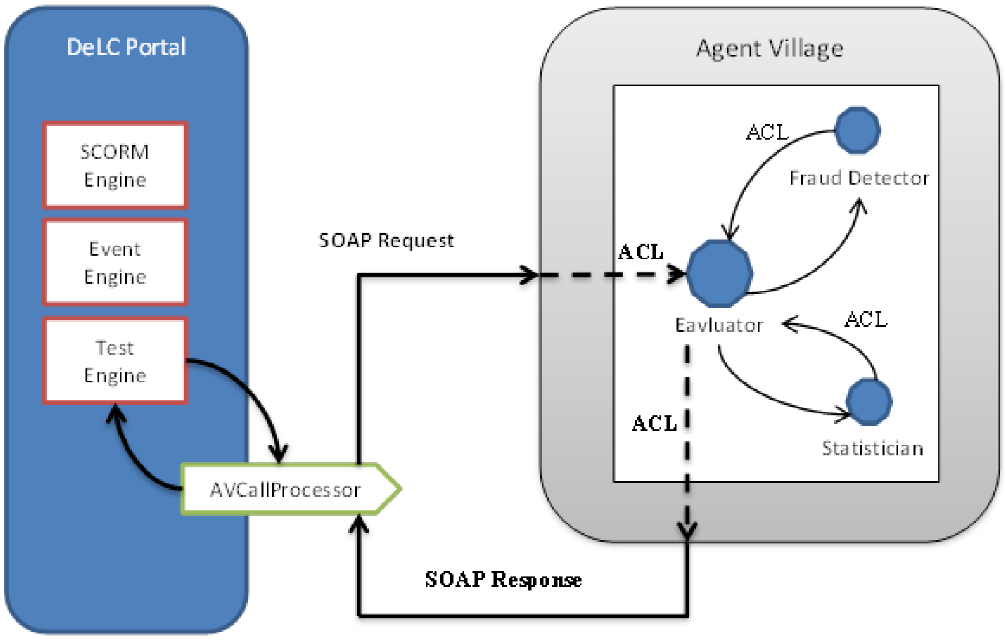


Figure 2. Architecture of the Education Cluster.

mainly for the convenient realization and integration of business functionality. Therefore, they must "transfer" the responsibility for activation and support of the connection to an active component of the architecture, as agents do. To do this, the service sends an explicit message to the agent's environment, which in turn identifies the change of environment and reacts by interpreting the message. Depending on the identified need for assistance, the agent activates the actions required. The reactive behavior of the architecture could be implemented by:

- Synchronous model – it is analogous to calling subroutines in programming languages. In this model, the service sends a message to AV and waits for the result from the corresponding agent before continuing its execution.
- Asynchronous model – in it the interaction is accomplished through a mechanism for sending and receiving messages.

The communication between the portal's services and the Agent Village's agents is achieved via web services (Fig. 2). The intelligent assistants' functions are wrapped in web service functions and thus are accessible for the portal's needs – this represents the AV's reactive behavior. On the other hand, the agent environment is active and could react to impending changes or events. Thus, the agents are able to undertake certain actions if they decide that there are proper conditions to execute them. Then, the agents send the results to the portal and, if necessary, to its users; this represents the AV's proactive behavior.

The internal communication among agents inside the AV uses ACL messages (Agent Communication Language) [12].

When a service is needed, which is functionality of some of the intelligent assistants, the portal generates SOAP Request message, which is sent to the agent's web service, whose functionality has to be exploited. This message is captured by a system component of the agent's environment, called Web Services Integration Gateway (WSIG), which is responsible for transforming the incoming SOAP Request message to an ACL message. Then this ACL message is transmitted to the proper agent, whose functionality serves the called-for web service. When the agent does its job it generates an answer in the form of an ACL message which is sent back to the environment, where a SOAP Response message is generated to be sent to the portal's calling component.

For processing SOAP messages in the portal, there is a system component, called AV-Call Processor. Its purpose is to generate a proper xml SOAP Request message, so that the right web service is called and, on the other hand, extract the result returned from the SOAP Response message and provide it to the portal in a convenient form.

In the proactive behavior (agents work "on behalf of the user"), an agent from the AV can determine that in its environment "something is happening", that would be interesting for the user, who is assisted by that agent. The agent is activated and can perform certain actions to satisfy the preferences (wishes) of the user. The agent can inform the user of its actions through the educational portal.

The difficulties associated with the management of our architecture's pro-activity, result from the fact that the portal is designed for reaction to the user's requests. Therefore, the pro-activity can be managed only asynchronously and, for this purpose, we provide development of a specialized service, which is to check a "mailbox" for incoming messages from the AV periodically.

## IV. ASSISTANTS

A part of the functionality of the Education Cluster is to organize students' examinations – to be more precise, this is a part of the DeLC portal. It consists of automated test generation, including various questions – "choice-like", answer matching and open-ended questions. To help the teacher estimate these answers, an intelligent service is added, which automatically assesses the open-ended questions.

Another function which is also related to student examination is comparing students' answers among themselves and to Internet search engines resources. Since the portal provides an integrated chat system, the students are able to communicate through it. But during the examination it could lead to cheating. That is why an intelligent assistant is built, which tracks the chat system communications and when it detects a suspicious message, it is written in the database, where the teacher can see it at any moment. This assistant will also compare the answer to Internet resources.

To control the effectiveness of the aforementioned intelligent assistants, there is a third assistant which stores statistical data about their operation. When the collected data is sufficient, the assistant could make conclusions and give advices to other assistants.

### A. *Evaluator Assistant*

The Evaluator Assistant provides expert assistance to the teacher in assessing electronic tests. A system service is built in the Test Engine for automated assessment of "choice-like" questions. In the standard version of the architecture, open-ended questions are assessed by the teacher and the mark is entered manually in the service which prepares the final assessment for the test. In the new architecture the Test Engine calls the assistant (an intelligent agent), which makes an "external" assessment of the open-ended questions.

If an "external" assessment is needed, where the Test Engine initiates a "request" for expert assistance, the reactive behavior of the EA is exploited. The EA has a wrapper (the environment of the agent) for identification, which "masks" it as a web service for the portal. The Test Engine relies on an AVCallProcessor for making the request and procession of the answer. When a request for assistance arises, this service generates a SOAP Request message and sends it to the Agent Village. When a SOAP Response is received, it parses the answer and extracts the estimated rating by the EA.

In the surrounding environment of the EA, the received SOAP Request messages are transformed into ACL (Agent Communication Language) [12] messages which are understandable for the agent. Some of the basic parameters of the messages are:

- Text, which is an answer of an open-ended\ question.
- Parameters for the estimation method used.
- Maximum number of points for this answer.

After the calculations, the EA generates an answer in the form of an ACL message, which is then transformed from the environment into a SOAP Response message (a result from a web-service call). In the answer, there is a parameter for the amount of points which is then extracted by the AV-Call Processor.

### B. *FraudDetector*

The role of this assistant is using its behavior to recognize some of the most frequent kinds of fraud during examination. They include:

- Keyword guessing for a particular question;
- Copy/paste from Internet search-engines resources;
- Exchange of information among students using the portal's chat system.

This agent cooperates with the Evaluator assistant during the examination process and if it recognizes a probability for fraud, it informs the Evaluator to take action concerning a specific question, thus demonstrating its proactive behavior. All suspicious information is stored in the database for the teacher to access at any moment.

### C. *Statistician*

This assistant is still in development. Its functionality will be to store information about all processes of the aforementioned two assistants: details about automated answer-assessment and about checking chat communication and suspicious messages between students.

Regarding the automated answer-assessment, this agent will need feedback about the teacher's actual assessment for each answer. Thus, it will compare the assessment of all used algorithms in the Evaluator assistant and make conclusions about their effectiveness.

## V. APPROBATION

The education cluster is used in a real education process. Some results for the "Introduction to Databases" lecture course (IDB) are summarized in this section.

In summary, there were 142 students examined in 2 subjects: 74 studying "Informatics" (3rd-year students) and 68 studying "Business Information Technologies" (2nd-year students). These students answered 453 open-ended questions. 127 of the answers were blank, considered as irrelevant for the system's effectiveness test, so they were excluded from the automated-evaluation statistics, given below. These statistics concern the remaining 326 answers.

Currently, the Evaluator Assistant uses two different algorithms for answer evaluation, called word-matching and optimistic-percentage. Some results of the evaluation results are shown in Fig. 3 in comparison to the points given by the teacher.

The average number of maximum points for the answer is 4.2. That number makes sense when we calculate the number of answers, which are evaluated with a one-point difference between the agent's (automated) and the teacher's evaluation. We consider these evaluations to be very close.

The summarized results of the automated evaluation of the answers are shown in Fig. 4.

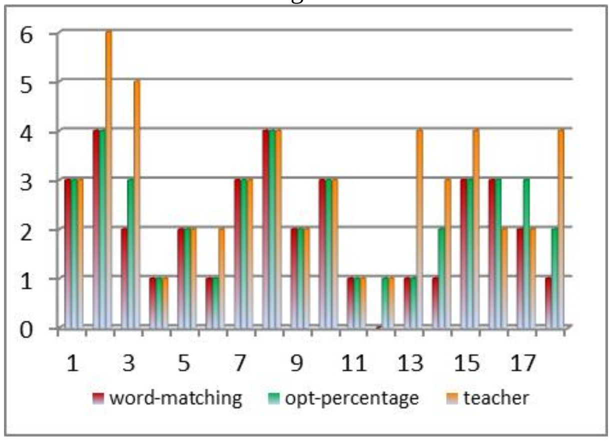


Figure 3. The points given by the two algorithms and the teacher.

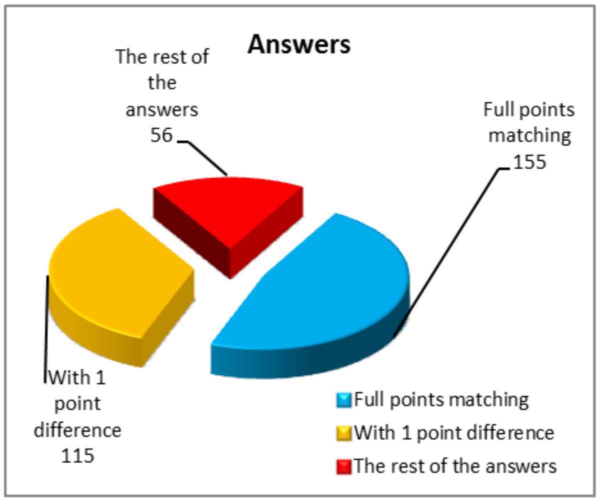


Figure 4. The results of the automated answers evaluation.

Currently, there is a new version of the evaluation algorithm. It was tested with these answers and the results were slightly better. They are shown in the Table 1 below.

TABLE I. THE RESULTS COMPARISON

| Version | old | | new | |
|---|---|---|---|---|
| *Answers* | *326* | *%* | *326* | *%* |
| Full points matching | 155 | 47.55 | 162 | 49.70 |
| With 1 point difference | 115 | 35.27 | 116 | 35.58 |
| The rest of the answers | 56 | 17.18 | 48 | 14.72 |

## VI. CONCLUSION

In this paper, an extension of the DeLC (Distributed eLearning Center) architecture was presented. The extension consists of making the architecture agent- and service-oriented, adding reactivity and proactivity to the existing e-learning environment. The reactive and proactive behavior of the architecture was demonstrated with the Evaluator Assistant, which provides expert assistance to the lecturer in assessment of electronic tests. Two other agents were projected for giving assistance to the educational portal's services. The entire system was tested for a year in the Faculty of Mathematics and Informatics at the University of Plovdiv "Paisii Hilendarski" (Bulgaria) in the bachelor courses of two subjects – Informatics and Business Information Technologies. The main results of the use of the cluster are increased interest and more active inclusion of the students in the education process.

## ACKNOWLEDGMENT

The authors wish to acknowledge the support of the National Science Fund (Research Project Ref. No. DO02-149/2008) and the Science Fund of the University of Plovdiv "Paisii Hilendarski" (Research Project Ref. No. NI11-FMI-004).

## REFERENCES

[1] Micro Soft Class Server, http://www.microsoft.com/education/products/ <retrieved: 12, 2011>.

[2] The IBM Learner Portal, https://www-04.ibm.com/jct03001c/services/learning/ites.wss/us/ <retrieved: 12, 2011> .

[3] HP Learning Center, http://h30187.www3.hp.com/ <retrieved: 12, 2011>.

[4] Moodle, http://moodle.org/ <retrieved: 01, 2012> .

[5] IMS Global Learning Consortium, http://www.imsglobal.org/ <retrieved: 01, 2012> .

[6] Advanced Distributed Learning, http://www.adlnet.org/ <retrieved: 01, 2012> .

[7] SCORM 2004, http://www.adlnet.gov/Technologies/scorm/SCORMSDocuments/ <retrieved: 01, 2012> .

[8] Stoyanov S., I. Ganchev, I. Popchev, M. O'Droma, R. Venkov, DeLC – Distributed eLearning Center, Proc. of the 1st Balkan Conference in Informatics BCI'2003, 21-23 November, Thessaloniki, Greece, 2003, pp. 327-336.

[9] Stoyanov, S., Ganchev, I., Popchev, I., O'Droma, M., From CBT to e-Learning, Journal Information Technologies and Control, No. 4/2005, Year III, pp. 2-10.

[10] S. Stoyanov, I. Ganchev, M. O'Droma, H. Zedan, D. Meere, V. Valkanova, Semantic Multi-Agent mLearning System, A. Elci, M. T. Kone, M. A. Orgun ( Eds.): "Semantic Agent Systems: Foundations and Applications", Book Series: Studies in Computational Intelligence, Vol. 344, Springer Verlag, 2011, pp. 243-272.

[11] S. Stoyanov, I. Popchev, E. Doychev, D. Mitev, V. Valkanov, A. Stoyanova-Doycheva, V. Valkanova, I. Minov, DeLC Educational Portal, Cybernetics and Information Technologies (CIT), Vol.10, No 3., Bulgarian Academy of Sciences, 2010, pp. 49-69.

[12] Agent Communication Language Specifications, http://www.fipa.org/repository/aclspecs.html <retrieved: 01, 2012> .

# Towards Accurate Electricity Load Forecasting in Smart Grids

Zeyar Aung, Mohamed Toukhy
Computing and Information Science Program
Masdar Institute of Science and Technology
Abu Dhabi, PO Box 54224, United Arab Emirates
e-mails: {zaung, mahmed}@masdar.ac.ae

John R. Williams, Abel Sanchez, Sergio Herrero
Engineering Systems Division
Massachusetts Institute of Technology (MIT)
Cambridge, MA 02139, United States of America
e-mails: {jrw, doval, sherrero}@mit.edu

***Abstract*—Smart grids, or intelligent electricity grids that utilize modern IT/communication/control technologies, become a global trend nowadays. Forecasting of future grid load (electricity usage) is an important task to provide intelligence to the smart gird. Accurate forecasting will enable a utility provider to plan the resources and also to take control actions to balance the supply and the demand of electricity. In this paper, our contribution is the proposal of a new data mining scheme to forecast the peak load of a particular consumer entity in the smart grid for a future time unit. We utilize least-squares version of support vector regression with online learning strategy in our approach. Experimental results show that our method is able to provide more accurate results than an existing forecasting method which is reported to be one of the best. Our method can provide 98.4–98.7% of average accuracy whilst the state-of-the-art method by Lv *et al.* is able to provide only 96.7% of average accuracy. Our method is also computationally efficient and can potentially be used for large scale load forecasting applications.**

***Keywords—smart grids; data mining; load forecasting; regression analysis; support vector machines.***

## I. INTRODUCTION

In this section, we will briefly introduce the concept of a smart grid, discuss the problem of electricity load forecasting in the smart grid and the prior arts to solve it, and outline our proposed solution to the problem.

### A. *Smart Grid*

A smart grid is an advanced electricity transmission and distribution network (grid) that utilizes information, communication, and control technologies to improve economy, efficiency, reliability, and security of the grid [23]. Nowadays, it is a priority of many governments worldwide to replace/upgrade their several decades old electricity grids with smart grids. For example, in 2010, the US government spent $7.02B on its smart grid initiative, while the Chinese government used $7.32B for its smart grid program [20].

The smart grid is characterized by several new trends and features: smart meters, demand response mechanisms, online customer interactions though PCs/mobile devices, dynamic electricity tariffs, online billing, incorporation of renewable energy generation (such as solar and wind energy) and electric vehicles, more reliable power transmission and distribution, dynamic load balancing, better power quality, better power security, etc. The architecture and components of the smart grid are illustrated in Figure 1.

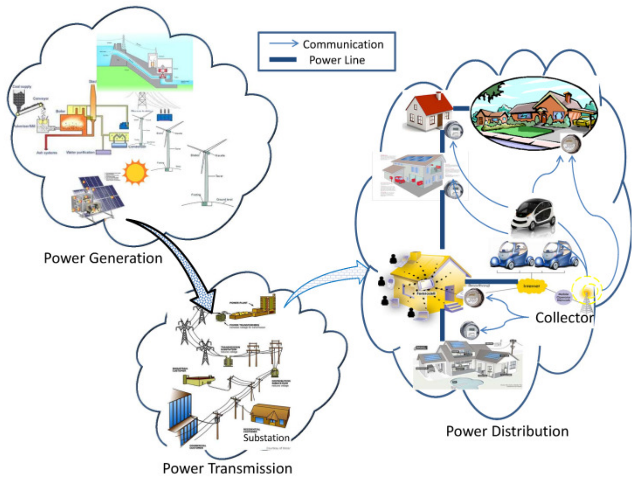


Figure 1. Architecture and components of the smart grid (reproduced with permission from [9]).

Information technology (IT) is one of the major driving forces behind a smart grid, and various IT systems and techniques such as artificial intelligence, high performance computing, simulation and modeling, data network management, database management, data warehousing, and data mining are to be used to facilitate smooth running of the smart grid [2].

### B. *Load Forecasting Problem*

Grid load forecasting is an important task to provide intelligence to the smart grid. Accurate forecasting will enable a utility provider to plan the resources like fuel in advance and also to take control actions like switching on/off demand response appliances and revising electricity tariffs, etc.

In our research, we will focus on a specific problem of forecasting the "peak load" (i.e., the maximum electricity usage) of a particular consumer entity for a future time unit. The consumer entity in question can be of various granularity levels. For example, it can be a smart meter (for a household), a cluster of smart meters (for a neighborhood), a power substation (for a town or city), or a power station (for an entire grid covering a large geographical area). Similarly, the time unit in question can be of different lengths. It can be 5 minutes, 15 minutes, 1 hour, 1 days, 1 week, etc.

In this paper, we will study a system to forecast the daily peak loads of individual smart meters. However, it should be noted that the same principles and techniques used in our studies are generally applicable to any load forecasting problems with any combinations of consumer entities and time granularities.

Researchers have been trying to solve the problem of electricity load forecasting since 1990's [1]. A number of methods based on different techniques such as time series analyses (like autoregressive integrated moving average (ARIMA) method [5]), fuzzy logic [14], neuro-fuzzy method [8], artificial neural network (ANN) [3], and support vector regression (SVR) [11] have been proposed.

Among these various techniques, support vector regression (SVR) is one of the latest developments. In [11], it was demonstrated that SVR could provide better results than the older methods like artificial neural network (ANN) [3] could.

### C. *Outline of Proposed Solution*

As in the previous work [3] and [11], we try to approach the problem of smart grid's load forecasting from the data mining perspective. In particular, we propose a peak load forecasting model based on the data mining technique of support vector regression (SVR) using least squares [15]. More specifically, we use the online greedy least-squares SVR proposed by Engel *et al.* [7].

In order to predict the peak load $P_d$ of a particular day $d$, we can roughly consider $P_d$ as a non-linear combination of a number of attributes from different sources: peak loads of previous $N$ days, average temperatures of previous $N$ days, holiday records of previous $N$ days, forecasted temperature of day $d$, and whether day $d$ is a holiday (weekend or public holiday).

So, for each "target" peak load value in the historical record, we construct a "feature vector" covering the abovementioned attributes associated with the target. Then, we train our least-square SVR system using a set of <*feature vector, target*> pairs for a large enough number of days. The result of this training process is a least-squares regressor model.

We can use the resultant regressor model to forecast the peak load value $P_d$ of a given day $d$. For that, we have to construct a feature vector for the day $d$ in the same manner as in the training step. In constructing the feature vector, we need to know the forecasted temperature of the day $d$ (if it is in the future) and whether it is a holiday (which can be easily known in advance). Then, the feature vector of day $d$ is supplied to the regressor model to generate the forecasted peak load value of that day.

After the day $d$ is already passed and its actual peak load value (the target) already known, the regressor model is updated with the <*feature vector, actual target*> pair for the day $d$, thus resulting in a fresh model which best reflects the latest trend of events.

A schematic representation of our proposed load forecasting scheme is given in Figure 2.

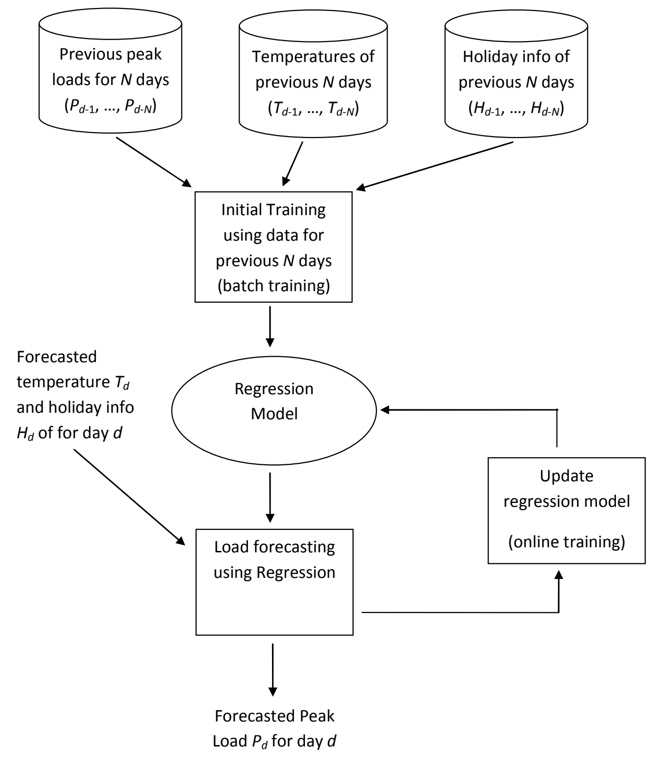

Previous peak loads for $N$ days ($P_{d-1}$, …, $P_{d-N}$)
Temperatures of previous $N$ days ($T_{d-1}$, …, $T_{d-N}$)
Holiday info of previous $N$ days ($H_{d-1}$, …, $H_{d-N}$)
Initial Training using data for previous $N$ days (batch training)
Forecasted temperature $T_d$ and holiday info $H_d$ of for day $d$
Regression Model
Update regression model (online training)
Load forecasting using Regression
Forecasted Peak Load $P_d$ for day $d$

Figure 2. Overview of the proposed load forecasting scheme.

We tested our proposed method using the smart metering data from the two regions in Germany for the year 2009. Experimental results demonstrate that our approach is both accurate and computationally efficient.

The remaining of the sections is organized as follows. Section II describes our proposed load forecasting method using online least-squares support vector regressions in details. Section III presents the experimental results of the proposed method in comparison with a state-of-the-art method. Section IV discusses the future work and concludes the paper.

## II. METHOD DETAILS

Support vector machine (SVM) [16] is a kind of maximum margin classifier which was originally proposed to solve the problem of binary classification. Among a large number of training data vectors, only a few are selected as "support vectors" that define the maximum margin. Only the support vectors are utilized in predicting the classes of the testing data vectors, thus leading to a good generalization.

Later, it was realized that SVM can be adapted to solve the problem of regression [13]. Suykens and Vandewalle proposed a least-squares version of support vector regression (SVR) [15] which is particularly suitable to solve regression problems in time series data. The least-squares SVR tries to find the solution by solving a set of linear equations instead of a convex quadratic programming for classical SVMs.

A brief description of the least-squares SVR is given below in Subsections A, B, and C. This description is adapted and modified from the original ones given in [10], [15], and [19].

## A. *General SVM Formulation*

Suppose we have a training set of $n$ samples $\{x_i, y_i\}$ $(i = 1, \cdots, n)$ with input data vector $x_i \in \mathbb{R}^m$ (where $m$ is the dimensionality of $x_i$) and corresponding binary class labels $y_i \in \{-1, +1\}$. In Vapnik's original formulation [16], the SVM classifier is defined by the conditions:

$$\left.\begin{array}{ll} w.\phi(x_i) + b \geq 1, & \text{if } y_i = +1 \\ w.\phi(x_i) + b \leq -1, & \text{if } y_i = -1 \end{array}\right\} \quad (1)$$

which can be rewritten as a single condition:

$$y_i(w.\phi(x_i) + b) \geq 1, \qquad i = 1, \cdots, n \qquad (2)$$

where $\phi(x)$ is a nonlinear mapping function of a vector from original space to the high (possibly infinite) dimensional space, $w$ is a weight vector which defines the separation hyperplane, and $b$ is an offset of the separation hyperplane from the origin $(0, 0)$.

If the given data set is inseparable (i.e., separating hyperplane does not exist), a slack variable $\xi_i$ is introduced in such a way that:

$$\left.\begin{array}{ll} y_i(w.\phi(x_i) + b) \geq 1 - \xi_i & i = 1, \cdots, n \\ \xi_i \geq 0, & i = 1, \cdots, n \end{array}\right\} \quad (3)$$

By applying the structural risk minimization principle, the risk bound (i.e., learning error) of the classifier can be minimized by solving the following minimizing problem:

$$\min J_1(w, \xi) = \frac{1}{2} \parallel w \parallel^2 + C \sum_{i=1}^{n} \xi_i \qquad (4)$$

subject to the constraints:

$$\begin{array}{ll} y_i[w.\phi(x_i) + b] \geq 1 - \xi_i, & i = 1, \cdots, n \\ \xi_i \geq 0, & i = 1, \cdots, n \end{array}$$

where $C$ is the slack penalty parameter to control the net effect of the slack variables.

In order to remove the complex constraints of the above minimization problem in Equation (4), we introduce Lagrangian multipliers $\alpha_i \geq 0$ $(i = 1, \cdots, n)$ [4]. Thus, the minimization problem becomes:

$$\begin{aligned} &\min L_1(w, \xi, \alpha) \\ &= \frac{1}{2} \parallel w \parallel^2 + C \sum_{i=1}^{N} \alpha_i \left(y_i(w.\phi(x) + b) - 1 + \xi_i\right) \end{aligned} \qquad (5)$$

subject to the constraints:

$$\xi_i \geq 0, \quad i = 1, \cdots, n$$

The optimal point will in the saddle point of the Lagrangian function. Thus, we have:

$$\left.\begin{array}{l} \dfrac{\partial L_1}{\partial w} = 0 \Rightarrow w = \displaystyle\sum_{i=1}^{n} \alpha_i \phi(x_i) \\ \dfrac{\partial L_1}{\partial b} = 0 \Rightarrow \displaystyle\sum_{i=1}^{n} \alpha_i y_i = 0 \\ \dfrac{\partial L_1}{\partial \xi_i} = 0 \Rightarrow 0 \leq \alpha_i \leq C, \qquad i = 1, \cdots, n \end{array}\right\} \quad (6)$$

By substituting $w$ by its expression, we get the following quadratic programming problem:

$$\max Q_1(\alpha) = \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \qquad (7)$$

subject to the constraints:

$$0 \leq \alpha_i \leq C, \qquad i = 1, \cdots, n$$

Here, $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$ is called the kernel function (which will be elaborated below in Section II.C). By solving this quadratic programming problem subject to the constraints, we will get the separating hyperplane in the high dimensional space, that is, the classifier in the original space.

## B. *Least Squares SVM Formulation*

Suykens and Vandewalle derived the least squares version of the SVM classifier by reformulating the minimization problem as below [15]:

$$\min J_2(w, b, e) = \mu \left(\frac{1}{2} \parallel w \parallel^2\right) + \zeta \left(\frac{1}{2} \sum_{i=1}^{n} e_i^2\right) \qquad (8)$$

subject to the equality constraints:

$$y_i(w.\phi(x_i) + b) = 1 - e_i\,, \qquad i = 1,\cdots,n$$

The least-squares SVM classifier formulation above implicitly corresponds to a regression interpretation with binary targets $y_i = \pm 1$.

Both $\mu$ and $\zeta$ are parameters to tune the amount of regularization versus the sum squared error. The solution does only depend on the ratio $\gamma = \mu / \zeta$, therefore the original formulation uses only $\gamma$ as tuning parameter. Therefore, we have:

$$\min J_2(w,b,e) = \frac{1}{2} \| w \|^2 + \gamma \frac{1}{2} \sum_{i=1}^{n} e_i^2 \tag{9}$$

The solution of the least-squares regressor is obtained after the Lagrangian function is constructed as follows:

$$\left.\begin{aligned} &L_2(w,b,e,\alpha) \\ &= J_2(w,b,e) - \sum_{i=1}^{n} \alpha_i \left(y_i(w.\phi(x_i) + b) - 1 + e_i\right) \\ &= \frac{1}{2} \| w \|^2 \\ &\quad + \gamma \frac{1}{2} \sum_{i=1}^{n} e_i^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(w.\phi(x_i) + b) - 1 + e_i\right) \end{aligned}\right\} \tag{10}$$

where $\alpha_i \in \mathbb{R}$ $(i = 1,\cdots,n)$ are the Lagrange multipliers. Again, the conditions for optimality are:

$$\left.\begin{aligned} &\frac{\partial L_2}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{n} \alpha_i y_i \phi(x_i) \\ &\frac{\partial L_2}{\partial b} = 0 \Rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0 \\ &\frac{\partial L_2}{\partial e_i} = 0 \Rightarrow \alpha_i = \gamma e_i\,, \qquad i = 1,\cdots,n \\ &\frac{\partial L_2}{\partial \alpha_i} = 0 \Rightarrow y_i(w.\phi(x_i) + b) - 1 + e_i = 0, \\ &\qquad\qquad\qquad\qquad i = 1,\cdots,n \end{aligned}\right\} \tag{11}$$

By the elimination of $w$ and $e$, we will have a linear programming problem instead of a quadratic programming one:

$$\begin{bmatrix} 0 & y^T \\ y & \Omega + \gamma^{-1} I_n \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ 1_n \end{bmatrix} \tag{12}$$

where $y = [y_1,\cdots,y_n]$, $1_n = [1,\cdots,1]$, $\alpha = [\alpha_1,\cdots,\alpha_n]$, and $I_n$ is an $n \times n$ identity matrix.

Here $\Omega \in \mathbb{R}^{n \times n}$ is the kernel matrix whose individual element $\Omega_{i,j}$ $(i,j = 1,\cdots,n)$ is defined as follows [6], [10].

$$\Omega_{i,j} = \phi(x_i.x_j) = K(x_i,x_j) \tag{13}$$

## C. Kernel Function

For the kernel function $K(\bullet,\bullet)$ one typically has the following choices [10], [19]:

Linear kernel:

$$K(x_i,x_j) = x_i.x_j \tag{14}$$

Polynomial kernel of degree $p$:

$$K(x_i,x_j) = \left(\frac{x_i.x_j}{c}\right)^p \tag{15}$$

Radial basis function (RBF) kernel:

$$K(x_i,x_j) = \exp\left(-\frac{\| x_i - x_j \|^2}{\sigma^2}\right) \tag{16}$$

Multi-layer Perceptron (MLP) kernel:

$$K(x_i,x_j) = \tanh(k\, x_i.x_j + \theta) \tag{17}$$

where $p, c, \sigma, k$ and $\theta$ are constants. Here, Mercer's condition holds for all $c, \sigma \in \mathbb{R}^+$ and $p \in \mathbb{N}$ values in the polynomial and RBF case, but not for all possible choices of $k$ and $\theta$ in the MLP case. The scale parameters $c, \sigma$ and $k$ determine the scaling of the inputs in the polynomial, RBF and MLP kernel functions. This scaling affects the kernel's bandwidth, which is an important factor in generalization of a kernel method [10], [19].

## D. Online Learning

In our proposed method, we employ a version of least-squares SVR, namely the online greedy SVR proposed by Engel *et al.* [7]. In this online learning setup, we first train our least-squares SVR system with a large enough set of data in a batch mode. Then, we deploy the system for regressing (forecasting) an unknown future data.

When the actual value of the forecasted data is came to know, the SVR system is updated using this actual data. In this way, the SVR system is always up-to-date and can truthfully represent the latest trend of the data. Therefore, online learning enables us to reduce the effect of the "concept drift" phenomenon [18] which usually occurs in time-series data, and thus improving the accuracy of forecasting.

## E. Feature Vector Construction

In order to forecast the peak load $P_d$ of a given day $d$, we construct (encode) a feature vector with 32 attributes as listed in Table I. These 32 attributes are empirically chosen.

TABLE I. FEATURE VECTOR USED IN FORECASTING THE PEAK LOAD $P_d$ OF A GIVEN DAY $d$.

| Attrib -ute ID | Feature Description | Formula |
|---|---|---|
| 1 to 28 | Peak load of previous 28 days | $P_{d-1}$ to $P_{d-28}$ |
| 29 | Average peak load of previous 7 days | $1/7\ (P_{d-1} + P_{d-2} + \ldots + P_{d-7})$ |
| 30 | Average temperature of previous 7 days | $1/7\ (T_{d-1} + T_{d-2} + \ldots + T_{d-7})$ |
| 31 | Forecasted average temperature of the day *d* | $T_d$ |
| 32 | Whether the day *d* is a holiday (weekend or public holiday) | $H_d$ |

The individual and the average peak load information can be obtained from the given peak load data set itself. The historical temperature information for different regions of the world can be extracted from the Weather Underground website [24]. Holiday information of countries all over the world is available from the Holidays-Info.com website [22].

We use the scaling facility of the LibSVM software [21] to map the values of each attribute into the range of –1 to +1. This scaling exercise helps us improve the forecasting accuracy by a considerable extent. The experimental results of scaling vs. without scaling are discussed in Section III.D.

## III. EXPERIMENTAL RESULTS

In this section, we will discuss about the datasets that we use in our experiment, how the datasets are of training and testing subsets, the results that our proposed method achieved in comparison with an existing state-of-the-art method, the effects of scaling vs. non-scaling, and finally the computational efficiency of the method.

### A. Datasets

We use two datasets in our experiment. The first one is the electricity usage data for the year 2009 logged by a smart meter deployed in a household in Lower Saxony (LS) region of Germany. We will call this dataset as **LS Dataset**. The second one is the data for 2009 logged by a smart meter installed in a household in North Rhine-Westphalia (NRW) region of Germany. We will name this dataset as **NRW Dataset**.

Each original datasets contains the electricity usage readings of the smart meter at every 15 minutes. From these readings, we extract the peak load (i.e., the maximum reading) for each day. The daily peak load profile of LS Dataset for the whole year of 2009 is illustrated in Figure 3.

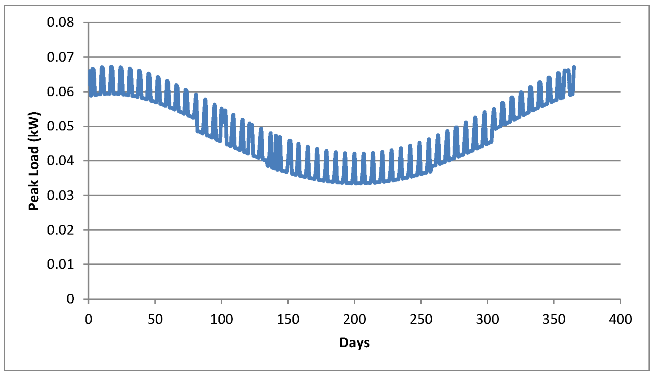


Figure 3. Daily peak load profile of an individual household in Lower Saxony (LS), Germany in 2009. The spikes indicate the increases of peak loads on holidays. Electricity usage is higher in winter and lower in summer.

### B. Training and Testing

For each dataset, we use each daily peak load value and its associated feature vector (as discussed in Section III) from February 01, 2009 to June 30, 2009 (150 days) as the training data for our forecasting system. (Note: we simply cannot start from January 01, 2009 because we need the data for the previous 4 weeks, i.e., 28 days, to construct a feature vector.)

The remaining days of the year from July 01, 2009 to December 31, 2009 (184 days) are used for testing (as well as for model updating in our online learning setup).

We use dlib C++ library [17] for the implementation of online greedy least-squares SVR algorithm by Engel *et al.* [7]. We use a radial basis function (RBF) kernel, which is described in Equation (16), with the kernel scaling parameter σ = 15, which is empirically determined.

### C. Results

We compare the accuracy performance of our proposed method with another least-square SVR-based method by Lv *et al.* [11], which uses a different feature vector encoding. A RBF kernel is also used for it with the parameter σ = 18, which is the optimum for that method. To enable a fair comparison, their regression model is also re-trained after every test instance in order to ensure an up-to-date model.

The forecasted peak load values for 184 test days from July 01, 2009 to December 31, 2009 are computed using both methods and are compared against the actual peak load values.

An example of the forecasted values by the two methods and the actual values for the month of December 2009 for LS Dataset are demonstrated in Figure 4. It can be observed from the figure that our method can predict daily peak loads more accurately than the method by Lv *et al.*

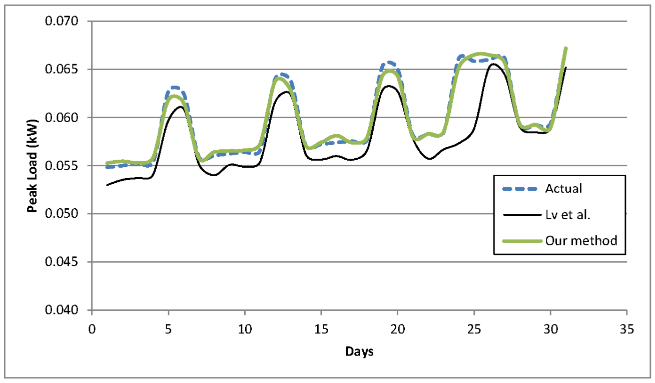


Figure 4. Example of forecasted results. The actual peak loads and the forecasted values by Lv et al. [11] and our method for the period of December 1 to 31, 2009 on LS Dataset.

In order to systematically analyze the performance of the two methods, we use two criteria: *relative error* and *accuracy* in our experiment. For each testing day, the relative error and the accuracy of the forecasted peak load are calculated as follows.

$$relative\ error = \frac{|actual\ peak\ load - forecasted\ peak\ load|}{actual\ peak\ load} \times 100\% \quad (18)$$

$$accuracy = 100 - relative\ error \quad (19)$$

For the testing period of 184 days, the comparisons of relative error values of the two methods are given in Figure 5 for LS Dataset and Figure 6 for NRW Dataset respectively. We can visually observe from the figures that our proposed method provides lower relative errors than the method by Lv *et al.* in a majority of cases.

For LS Dataset, the average relative error of our method is 1.3% (i.e., 98.7% average accuracy) whilst that of Lv *et al.* is 3.3% (i.e., 96.7% average accuracy).

For NRW Dataset, the average relative error of our method is 1.6% (i.e., 98.4% average accuracy) whilst that of Lv *et al.* is still 3.3% (i.e., 96.7% average accuracy).

Load forecasting is quite a mature technology in which many methods are able to provide an accuracy level of ~95% in general [11]. For example, the method by Lv *et al.* provides 96.7% of accuracy in our experiment. Here in our research, we are able to further improve the forecasting accuracy to the level of 98.4–98.7%. Although the 1.7–2.0% increase in accuracy may be small in absolute terms, this achievement is non-trivial. It is a common phenomenon that improving the performance of a particular technology is quite difficult when it becomes mature (i.e., when the higher plateau in its technology S-curve is reached) [12].

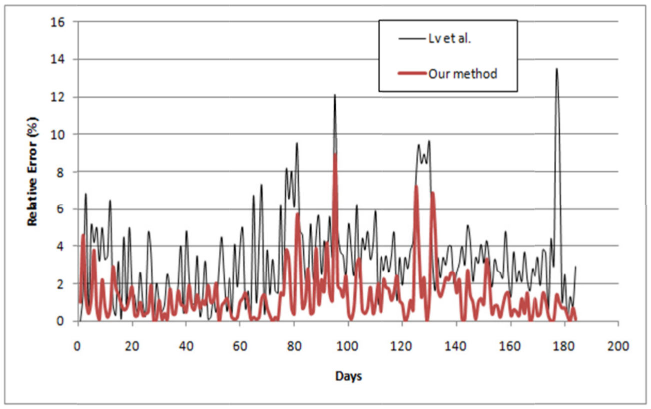


Figure 5. Relative error performances of Lv et al. [11] and our method for the period of June 1, 2009 to December 31, 2009 on LS Dataset.

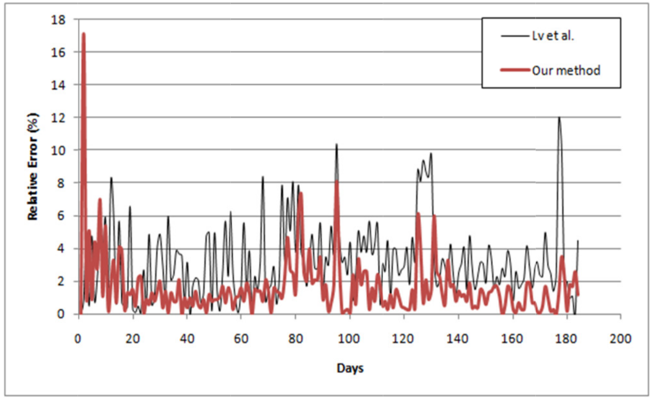


Figure 6. Relative error performances of Lv et al. [11] and our method for the period of June 1, 2009 to December 31, 2009 on NRW Dataset.

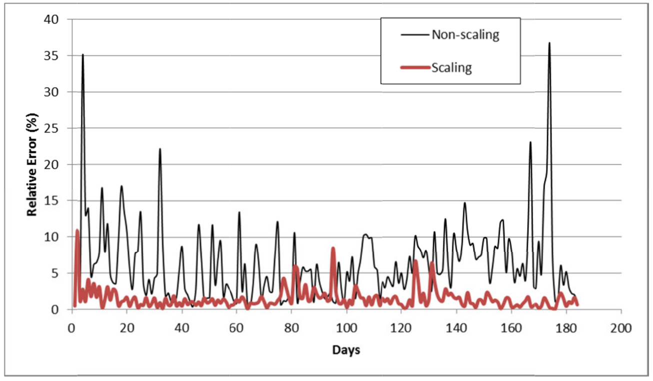


Figure 7. Relative error performances of scaling vs. non-scaling approaches in our method. The results are the averages of those for LS and NRW Datasets for the period of June 1, 2009 to December 31, 2009.

### D. Scaling vs. Non-scaling

As mentioned in Section II.E, we use the scaling facility of LibSVM [21] to map the values of each of our 32 attributes into the range of –1 to +1. Scaling helps significantly reduce the relative error (i.e., improve the accuracy) of our proposed method over directly using the data without any scaling. We can clearly observe this trend in Figure 7. The overall average of the relative errors for both LS and NRW Datasets with scaling is 1.4% while that without scaling is 6.2%. The superiority of scaling is because it prevents the dominance of attributes with larger value ranges over those with smaller ranges in calculating the Lagrange multipliers $\alpha$ in Equation 12.

### E. Computational Efficiency

The proposed method is developed in C++ and tested on a modest laptop PC with Intel Core Duo 1.83 GHz processor and 2GB of main memory running Windows Vista 32-bit Edition. The program is compiled with Microsoft Visual C++ 2008 using O2 optimization.

The method is found to be quite efficient and scalable. The overall running time of the training for 150 days and the testing (and re-training) for 184 days is only 210 milliseconds for LS Dataset and 220 milliseconds for NRW Dataset respectively. Thus, our proposed method can be potentially deployed in a larger scale to forecast the loads of tens of thousands of consumer entities like smart meters on a distributed computing platform.

## IV. Conclusion and Future Work

In this paper, we have presented an accurate load forecasting method which can potentially provide greater intelligence (smartness) to the upcoming smart grids. In our approach, we adopt the least-squares support vector regression technique incorporated with online learning. Experimental results show that our method is able to provide more accurate results than an existing forecasting method by Lv *et al.* [11], which is reported to be one of the best methods, and is also computationally efficient. As the future work, we intend to explore the idea of automatic feature selection for our regression model in order to further improve its accuracy. In addition, we plan to rigorously test our method with multiple smart grid load datasets from different countries/industries and fine tune the method so as to ensure its general usability. Finally, we hope our proposed method with these future improvements can be potentially useful to utility companies in their large-scale load forecasting applications for consumer entities at any granularity levels (such as individual households, neighborhoods, towns, cities, and large geographical regions) by providing results with better precisions.

## Acknowledgement

This research was sponsored by the Government of Abu Dhabi, United Arab Emirates through its funding of the MIT-Masdar Institute Collaborative Research Project on “Data Mining for Smart Grids” (award number 10CAMA1).

## References

[1] G. Adams, P. G. Allen, and B. J. Morzuch, “Probability distributions of short-term electricity peak load forecasts,” *International Journal of Forecasting*, vol. 7, pp. 283-297, 1991.

[2] M. Arenas-Martínez, S. Herrero-Lopez, A. Sanchez, J. Williams, P. Roth, P. Hofmann, and A. Zeier, “A comparative study of data storage and processing architectures for the smart grid,” in *Proceedings of the First IEEE International Conference on Smart Grid Communications*, pp. 285-290, 2010.

[3] M. Beccali, M. Cellura, V. L. Brano, and A. Marvuglia, “Forecasting daily urban electric load profiles using artificial neural networks,” *Energy Conversion and Management*, vol. 45, pp. 2879-2900, 2004.

[4] D. P. Bertsekas, *Nonlinear Programming (Second Edition)*, Athena Scientific, 1999.

[5] P. J. Brockwell and R. A. Davis, *Introduction to Time Series and Forecasting (Second Edition)*, Springer, 2002.

[6] Z. S. H. Chan, H. W. Ngan, A. B. Rad, A. K. David, and N. Kasabov, “Short-term ANN load forecasting from limited data using generalization learning strategies,” *Neurocomputing*, vol. 70, pp. 409-419, 2006.

[7] Y. Engel, S. Mannor, and R. Meir, “Sparse online greedy support vector regression,” in *Proceedings of the 2002 European Conference on Machine Learning*, pp. 84-96, 2002.

[8] A. Khotanzad, E. Zhou, and H. Elragal, “A neuro-fuzzy approach to short-term load forecasting in a price-sensitive environment,” *IEEE Transactions on Power Systems*, vol. 17, pp. 1273-1282, 2002.

[9] D. Li, Z. Aung, J. Williams, and A. Sanchez, “Efficient authentication scheme for data aggregation in smart grid with fault tolerance and fault diagnosis,” accepted for publication at *The 2012 IEEE Power and Energy Society Conference on Innovative Smart Grid Technologies*, Washington D.C., USA, January 2012.

[10] H. Liu, W. Ma, H. Liu, and Y. Hu, “Predicting price of target in acquisition by support vector machine,” in *Proceedings of the 2011 International Conference on E-Business and E-Government*, pp. 1-4, 2011.

[11] G. Lv, X. Wang, and Y. Jin, “Short-Term Load Forecasting in Power System Using Least Squares Support Vector Machine,” in *Proceedings of the 2006 International Conference on Computational Intelligence, Theory and Applications, 9th Fuzzy Days*, pp. 117-126, 2006.

[12] M. A. Schilling and M. Esmundo, “Technology S-curves in renewable energy alternatives: Analysis and implications for industry and government,” *Energy Policy*, vol. 37, pp. 1767-1781, 2009.

[13] A. J. Smola and B. Schölkopf, “A tutorial on support vector regression,” *Statistics and Computing*, vol. 14, pp. 199-222, 2004.

[14] K. B. Song, Y. S. Baek, D. H. Hong, and G. Jang, “Short-term load forecasting for the holidays using fuzzy linear regression method,” *IEEE Transactions on Power Systems*, vol. 20, pp. 96-101, 2005.

[15] J. A. K. Suykens and J. Vandewalle, “Least squares support vector machine classifiers,” *Neural Processing Letters*, vol. 9, pp. 293-300, 1999.

[16] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, 1995.

[17] http://dlib.net/ <retrieved December, 2011>

[18] http://en.wikipedia.org/wiki/Concept_drift <retrieved December, 2011>

[19] http://en.wikipedia.org/wiki/Least_squares_support_vector_machine <retrieved December, 2011>

[20] http://greenenergyreporter.com/renewables/cleantech/china-to-invest-7-3-bln-in-smart-grid-projects-in-2010/ <retrieved December, 2011>

[21] http://www.csie.ntu.edu.tw/~cjlin/libsvm/ <retrieved December, 2011>

[22] http://www.holidays-info.com/ <retrieved December, 2011>

[23] http://www.nist.gov/smartgrid/ <retrieved December, 2011>

[24] http://www.wunderground.com/ <retrieved December, 2011>

# Operating on Hierarchical Enterprise Data in an In-Memory Column Store

Christian Tinnefeld, Bjoern Wagner, Hasso Plattner
*Hasso Plattner Institute, University of Potsdam*
*Prof.-Dr.-Helmert-Str. 2-3, 14482 Potsdam, Germany*
*christian.tinnefeld@hpi.uni-potsdam.de, bjoern.wagner@hpi.uni-potsdam.de, hasso.plattner@hpi.uni-potsdam.de*

***Abstract*—Enterprise data management is currently separated into online transactional processing (OLTP) and online analytical processing (OLAP). This separation brings disadvantages such as the need for costly data replication, maintaining redundant systems or the inability to run data reports on the latest transactional data. Academia and industry are working on a reunification by storing all data in a single columnar in-memory database which is designed to sustain high transactional workloads while simultaneously handle complex analytical tasks as well. Since enterprise data often includes hierarchical data, this paper focuses on modeling, storing, and operating on hierarchical data in an in-memory columnar database. The paper contributes by describing the implementation of the most frequently used hierarchical data operations on such a database while maintaining the ability to execute performant analytical queries on such data as well. A set of benchmarks demonstrates that hierarchical data operations can be executed up to three times faster on an in-memory column store than on an in-memory row store. The paper closes with a discussion which enterprise applications can benefit from this contributions.**

***Keywords-hierarchical data; enterprise data management; in-memory column store; SanssouciDB.***

## I. Introduction

Hierarchies are very common in every day life. Organizational structures of companies, books, websites and most file systems have a natural hierarchical structure. Moreover hierarchies help to abstract complex things. E.g. software engineers use object-oriented decomposition techniques to encapsulate complex applications into independent modules. The object hierarchy and inheritance are two of the key concepts of object-oriented programming. Even one of the first scientific data model proposals, the IBM Information Management System, was based on a hierarchical data structure, long before the rise of relational database management systems [1]. There have been many proposals to store hierarchical data in relational databases as well as in specialized graph databases. This paper explores the opportunities of in-memory columnar databases for the persistence of hierarchical data. These columnar databases are superior in processing speed of huge datasets and provide high scalability especially in parallel scenarios. In addition to that, columnar databases sustain high transactional and analytic workloads at the same time. Therefore, they are used for reuniting transactional and analytical workloads which are currently separated in the context of enterprise data management [2].

The remainder of this paper is organized as follows. Section II describes several enterprise applications that heavily operate on hierarchical data in order to illustrate the need and the requirements towards operating on hierarchical data in the context of enterprise data management. Section III describes the relevant concepts in terms of encoding, data model, and data operations and their implementation in terms of describing the resulting SQL statements. The subsequent Section IV evaluates the performance by performing measurements on SanssouciDB, which is a prototypical in-memory columnar database [3]. Section V closes the paper with a summary of the most important insights and by discussing the implications for enterprise applications.

## II. Hierarchcial Data in Enterprise Applications

This section introduces enterprise application domains with a strong focus on hierarchical data. The discussion focuses on central characteristics of the applications and the resulting requirements towards data management.

**Supply Chain Management** Supply chain management tries to improve the flow of materials, information and financial resources within a company and between different companies [4]. The foundation of supply chain management is information sharing, coordinated planning, scheduling and execution as well as collaborative monitoring and controlling. Supply chain management enables companies to execute just-in-time production, reduce stock levels, provide better forecasting and a shorter time-to-market for new products. The reduction of vertical integration over the last decade requires a more intensive collaboration between suppliers and customers. E.g. in the automotive and high-tech industry supplier coordination is vital to keep the complex production processes up and running. At the same time, the complexity of products and production processes increased rapidly. The National Electronics Manufacturing Initiative, Inc. (NEMI) outlined in [5] that the Bill-of-Materials plays a major role in such collaboration scenarios. The Bill-of-Materials (BoM) (see Figure 1 for an example of a BoM) defines what and how many components are necessary to build a product. Therefore, the BoM is basically a hierarchy of components

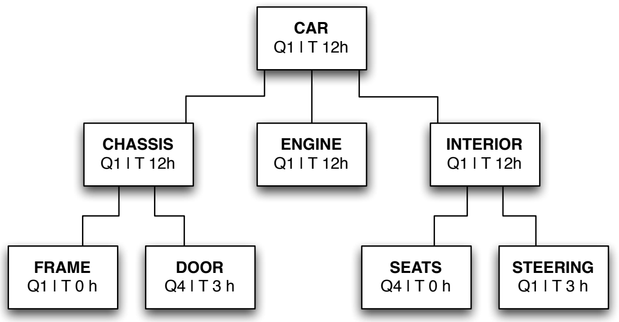


Figure 1. Bill-of-Material for a car containing quantity and assembly time

and sub components of a specific product. Many processes and stakeholders operate on the BoM, such as procurement, planning or production processes. Therefore it is necessary for supply chain management solutions to provide a fast, scalable and reliable BoM implementation. In addition to those primary transactional scenarios, the BoM is also used for analytics. For example, one might wants to know which components cause the most delays in order to improve the procurement process of this component.

**Product Lifecycle Management** Product lifecycle management (PLM) software is designed to manage the whole lifecycle of a product. This includes the conception, design and manufacturing phase as well as services and disposal. PLM integrates processes, data, people and provides interfaces to other business systems. One key requirement of PLM is the effective management of Bills-of-Information (BoI). Although Bills-of-Material from the previous section are also created and maintained by PLM processes, Bills-of-Information play a more important role. BoIs can be considered as a superset of Bills-of-Materials, because a BoM only contains all information necessary to assemble the product and a BoI holds lots of additional data generated during design, testing, manufacturing, sales and support of the product. A major goal of product lifecycle management is to reduce the time-to-market by centralizing the data organization. This drastically removes search time for component reuse, provides more accurate results and allows a better management as well as traceability of intellectual property assets throughout the whole product life cycle. This brings up two significant technical requirements for the implementation of the BoI hierarchy. First, the BoI hierarchy needs a very fast search interface. Second, the hierarchy has to be update-optimized, because in contrast to the BoM, the BoI changes often within the design iterations.

**Project Management** Another important application suite based on hierarchical data is project management. Usually, at the beginning of projects at set of high-level tasks is defined. During the project these tasks are divided in more detailed sub tasks. Beside this task hierarchy, employees and resources in general are organized in a hierarchical manner. Project management software uses those structures e.g. to calculate project costs, estimates, and reporting.

**Workforce Scheduling** The scheduling of a workforce is closely related to project management, but has special requirements concerning the hierarchy implementation. Project management focuses on static calculations and workforce scheduling runs more sophisticated optimizing algorithms. Workforce scheduling is calculated based on a hierarchy of tasks and resources in oder to find an optimal schedule to utilize all available resources. This is critical for utilize the available resources for a project in the most optimal way.

The preceding application domains gave an overview of the broad spectrum of technical requirements for processing hierarchies. It turned out that hierarchy implementations have to tackle read as well as write workloads. In addition to that real-time analytics on the complete hierarchies play an important role e.g. when finding out certain properties of a BoM or to find out how many resources are occupied for a certain task within a project.

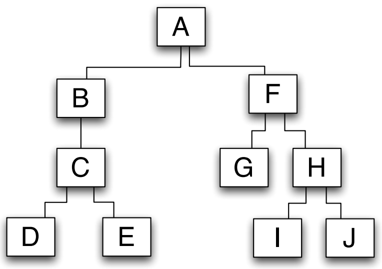
## III. CONCEPTS AND IMPLEMENTATION

This section focuses on the concepts and their implementation for handling hierarchical data in an in-memory column store. The example of the perviously introduced Bill-of-Material is used as requirement driver throughout this section.

### A. Preorder and Postorder Encoding

As mentioned in the previous section, fast execution of queries along the 4 main axes (ancestor, descendant, preceding, following) is crucial for Bill-of-Material processing. According to Grust et al. [6], the 4 main axis of each node in the tree partition the tree in 4 disjunct sets. The union of all those sets with the node itself contains all nodes of the tree exactly once. Grust further defines a mapping of tree nodes to a relational structure that preserves this partitioning and retrieves all the main axes using a simple range query. This mapping is based on so called preorder and postorder tree traversal. A preorder traversal assigns an unique rank to each node in the tree before its children are recursively traversed from left to right. The root node usually has the rank 0. Postorder traversal also assigns an unique rank to each node, but the value is assigned after the children has been traversed recursively.

In [7], Boncz et al. optimize the pre- and postorder encoding by size and level attributes that replace the postorder attribute. The size attribute describes the number of descendants and the level attribute describes the number



| | pre | size | level | *post* |
|---|---|---|---|---|
| *a* | 0 | 9 | 0 | 9 |
| *b* | 1 | 3 | 1 | 3 |
| *c* | 2 | 2 | 2 | 2 |
| *d* | 3 | 0 | 3 | 0 |
| *e* | 4 | 0 | 3 | 1 |
| *f* | 5 | 4 | 1 | 8 |
| *g* | 6 | 0 | 2 | 4 |
| *h* | 7 | 2 | 2 | 7 |
| *i* | 8 | 0 | 3 | 5 |
| *j* | 9 | 0 | 3 | 6 |

Figure 2. Example of a Pre- and Postorder table

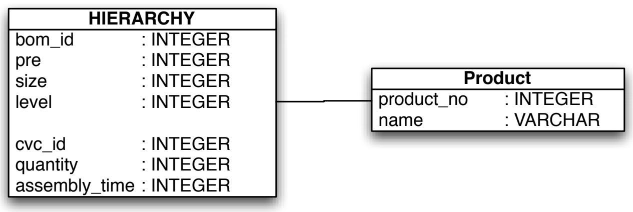


Figure 3. Schema of the pre- and postorder table

of ancestors of a node. The postorder rank is not stored explicitly in this schema anymore, but can be calculated for any node v via post(v) = pre(v) + size(v) - level(v). Figure 2 shows an example of the optimized encoding, Figure 3 depicts the resulting schema.

## B. Operations

This subsection describes the implementation of the needed operations. The description is illustrated with SQL queries that refer to the relational schema introduced above. The most important operations are compared in a benchmark in the next section.

*1) Number of Nodes in the Tree:* The calculation of the number of nodes in a tree can be straightforward or complex depending on the implementation. E.g. if each node is related to one row in a relational database the number of node equals the number of rows that can be retrieved with a simple count() SQL statement as shown in Listing 1, since each node in the tree is represented by a database row.

Listing 1. Query to calculate the number of nodes in the tree

```
SELECT count(pre) FROM TREE
```

*2) Number of Leaves in the Tree:* A leave is node without any child nodes. Hence, the number of leaves complies with the number of nodes without any children in the tree. The number of leaves in the tree can be calculated using the size attribute of the node with the SQL query of Listing 2.

Listing 2. Query to calculate the number of leaves in the tree.

```
SELECT count(pre) FROM TREE
WHERE     size=0
```

*3) Height of the Tree:* The height of the tree expresses the maximum distance of a leave to the root node in nodes. The height of the tree can be derived from the maximum level attribute plus 1, because it is zero based. The query can be found in Listing 3.

Listing 3. Query to retrieve the height of the tree.

```
SELECT max(level) + 1 FROM TREE
```

*4) Finding the Root Node:* The root node is the only node on level 0 and can be found with the following SQL statement as shown in Listing 4

Listing 4. Query to find the root node.

```
SELECT * FROM TREE
WHERE     level=0
```

*5) Main Axes Query:* The main axes traverse four different kind of axis. The ancestor axis describes the parent nodes of an input node excluding the node itself. The result list is ordered and starts with the root node of the tree. Because of that, the result can also be considered as path from the root node to the input node. The descendant axis is the opposite of the the ancestor axis. It describes the child nodes of the input node and recursively the descendants of those child nodes. As already mentioned, the tree structure is similar to a XML data model. Therefore the best way to illustrate preceding axis, to select all XML elements that have been closed before the context node. The following axis contains all elements that begin after the context element has been closed.

Because the 4 main axes partition the tree in 4 disjunct regions the range queries can be derived from the pre-post-plane. The context_node_pre and context_node_post are named parameters representing the pre- and postorder rank of the context node. The ORDER BY statement is used to ensure document order.

Listing 5. Query for the ancestor axis of the context node.

```
SELECT * FROM TREE
WHERE     pre < :context_node_pre
          AND post > :context_node_post
ORDER BY pre
```

Listing 6. Query for the descendant axis of the context node.

```
SELECT * FROM TREE
WHERE     pre > :context_node_pre
          AND post < :context_node_post
ORDER BY pre
```

Listing 7. Query for the preceding axis of the context node

```
SELECT * FROM TREE
WHERE     pre < :context_node_pre
          AND post < :context_node_post
ORDER BY pre
```

Listing 8. Query for the following axis of the context node.

```
SELECT * FROM TREE
WHERE     pre > :context_node_pre
          AND post > :context_node_post
ORDER BY pre
```

*6) Subtree matching:* Subtree matching is more complex than the operations presented above. First of all, according to [8] there are several kinds of subtree matching. For an ordered rooted tree T with vertex set V and edge set E, an rooted ordered Tree $T'$ is a bottom-up subtree if and only if

1) $V' \subseteq V$
2) $E' \subseteq E$
3) the labeling of V' and E' is preserved in T'
4) if vertex $v \in V$ and $v \in V'$ then all descendants of v must be in V'
5) left-to-right ordering among siblings of T must be preserved in T'

The bottom-up subtree matching defined by Zaki [9] is implemented by transferring the depth encoding tree representation to the relational model. The basic idea behind the subtree matching query is, to pick one tree and match it against all the others by joining the hierarchy table with itself. The query in Listing 9 shows one possibility to implement subtree matching in SQL. It takes the bomId and the number of nodes of the subtree as input parameters. It is possible to retrieve the number of nodes of the subtree by another join with the hierarchy table, but this would also introduce more complexity in this example. Since the subtree matching operates on products, the hierarchy table is joined on the cvcId attribute and where-clause ensures, that subtree isn't matched against itself. But the most important part of this query is the group-by-statement. First of all, the bomId attribute separates the result of different trees. Second, the *tree.pre-subtree.pre* and *tree.post-subtree.post* statements group all nodes of the tree together, that have a similar preorder and postorder rank. To be more exactly, the structure is of the subtree is preserved in the tree, but the position of the root node of subtree in the matching tree doesn't matter. Hence all nodes of a tree that match the structure of the subtree are contained in one group. The last having-clause verifies that all nodes of the subtree can be found in the matching tree. Finally, the select-statement returns the id of the matching tree as well as the preorder and postorder rank of the node that matches the root node of the subtree.

Listing 9. Query for subtree matching
```
SELECT    tree.bom_id, min(tree.pre), max(tree.post)
FROM      hierarchy subtree INNER JOIN hierarchy tree
ON        tree.cvc_id=subtree.cvc_id
WHERE     subtree.bom_id = :subtree_bom_id
          AND tree.bom_id <> :subtree_bom_id
GROUP BY  tree.bom_id, tree.pre-subtree.pre,
          tree.post-subtree.post
HAVING    count(tree.pre) = :subtree_node_count
```

*7) Add, Move, Delete Nodes or Subtrees:* The following operations change the structure of the tree. An insert operation inserts new nodes or subtrees below an existing node into the tree. Moving a node or a subtree means to change the parent of the node or the root node of the subtree. Deleting a node from a tree detaches it from the hierarchy. If this node has descendants, all descendants are also removed.

The implementations of the read-only statements aren't complex and can be executed fast on a relational database system as seen above. Unfortunately, this is not completely true for structural update operations of the tree. No matter if a single node or a subtree is inserted, moved, or deleted, all pre values following the insert node and all size values of ancestors of the insert node have to be updated. Especially the first update is critical from a performance perspective, because it has to update the half table on average. A highly optimized implementation of the update operation including transaction management was proposed by [7]. Listing 10 illustrates a much simpler query that inserts a node into a tree.

Listing 10. Query to insert a node or subtree
```
UPDATE   Hierarchy SET pre=pre+:subtree_node_count
WHERE    pre > :insert_node_pre
         AND post > :insert_node_post
UPDATE   Hierarchy SET size=size+:subtree_node_count
WHERE    pre < :insert_node_pre
         AND post > :insert_node_post
```

*8) Analytical Query:* Analytical operations often operate on all nodes of the tree instead of a subset. Path based aggregation traverses the tree or a subtree recursively from root to leaves and applies a arithmetical operation on one attribute of each node on the path.

The analytical query operation flattens the Bill-of-Material hierarchy by processing each component of each product and aggregate the quantity of each node. The result is a simple list of all components necessary to assemble the product from scratch as shown in Listing 11.

Listing 11. Query for flattening data hierarchy
```
SELECT    Hierarchy.bom_id, Cvc.name,
          sum(Hierarchy.quantity)
FROM      Hierarchy INNER JOIN Cvc
ON        Hierarchy.cvc_id=Cvc.id
GROUP BY  Hierarchy.bom_id, Cvc.name
```

## IV. Benchmarking

This section evaluates the performance of the previously discussed tree implementations by benchmarking the major tree operations and compare the results. The benchmarks are executed on two different database systems: one system is a row-oriented MySQL database running on a RAMDisk, the other database is a column-oriented SanssouciDB [3]. The machine they are running on is an Intel Xeon E5450 Quadcore with 3 GHz and 32GB of RAM.

### A. Read Operation

We selected the 4 main axis (ancestor, descendant, preceding, and following) as example query for read operations.

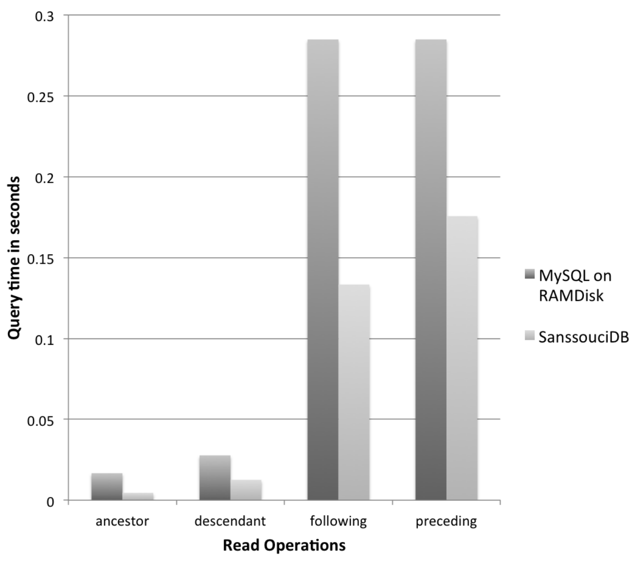


Figure 4. Read operations on a tree with 500.000 nodes

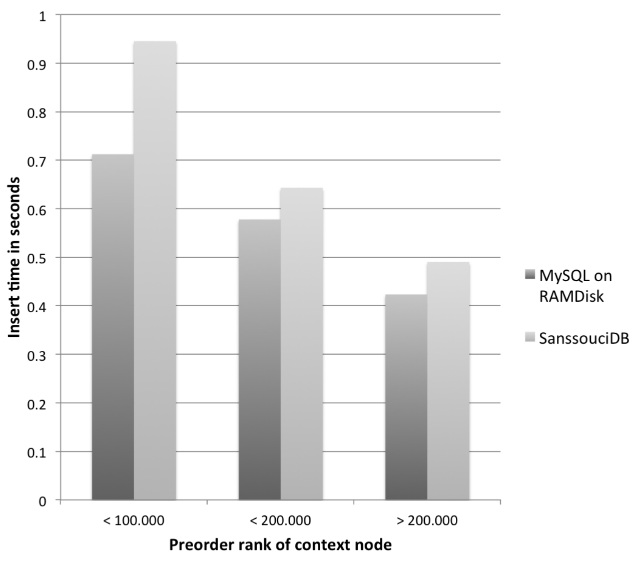


Figure 5. Insert operation on a tree with 500.000 nodes

Figure 4 shows the results for the read operations on a tree with 500.000 nodes. The column store is faster for all 4 read queries. However it occurs that there is a noticeable variance between the different axis on both databases. While the ancestor axis performs very well, the preceding and following axis operation is about ten times slower. This can be explained by the structure of the tree. Because the width of the tree is bigger than the height, the result set of the following axis operation is usually also larger. E.g. the result set of the following axis operation for a node with a small preorder rank can contain almost the complete tree, but the ancestor axis result set is limited to the height of the tree.

### B. Write Operation

The write operation we benchmarked adds a new child node to a leaf of the tree. It turns out, the lower the preorder rank of a leaf is, the slower performs SanssouciDB compared to MySQL. Obviously, SanssouciDB is optimized for read-intensive workloads, but this case is compounded by the fact that for lower preorder ranks almost all nodes in the hierarchy table are updated during an insert.

### C. Analytic Operation

Finally, the analytic operation flattens the Bill-of-Material hierarchy as described in the previous section. Throughout the different measurements, the number of nodes has been increased, starting with 100.000 nodes up to 1.000.000 nodes. Especially in the latter case, the column store can play off its analytic strengths and passes the MySQL up to a factor three. This is remarkable, because it shows that the described data operations on SanssouciDB provide a performant execution of transactional operations (such as reading and inserting nodes) as well as fast analytics on

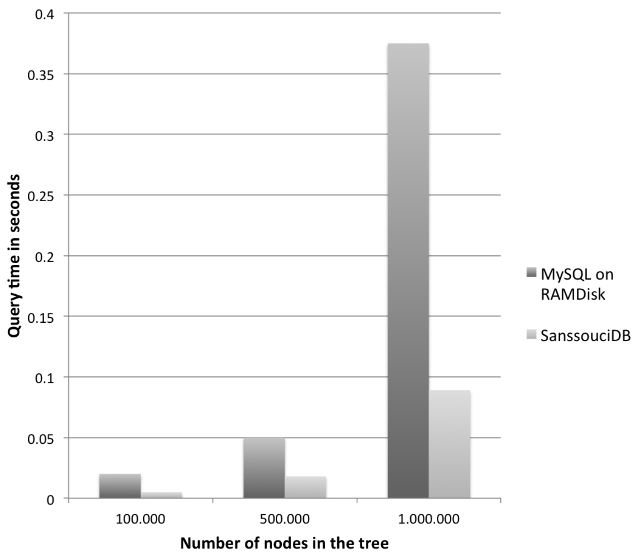


Figure 6. Analytic operation on a tree with varying number of nodes

the same data. Although the non-exponential growth of the response time when adding more nodes indicates that even bigger trees can be analyzed while still maintaining a sub-second response time.

## V. Conclusion

This paper shows that the reunification of transactional and analytical workloads on one single database is possible even when operating on hierarchical data. The described tree operations build the foundation for serving enterprise applications such as supply chain management, product lifecycle management, project management or workforce scheduling. Furthermore, the paper demonstrates that the described operations can be executed in a performant manner on an in-memory column store such as SanssouciDB.

Since this is the technical prerequisite for enabling more analytical functions within applications that work on hierarchical data, the following new application scenarios are within reach: when working on BoMs in the context of supply chain management, the properties of each component of a product can be taken into consideration when doing Supply Network Planning or Production Planning and not only the product as a whole. E.g. when a production plan has to be rescheduled due to a temporary outage of some machines, it can be instantly determined which sub-products can still be produced. Or for example in the field of product lifecycle management, analytics and simulations on every new iteration of a product is possible which leads to quicker validation of new product proposals and allows the product designer to validate that a new product fulfills certain requirements while he is still working on it.

## References

[1] M. Stonebraker and J. M. Hellerstein, “What Goes Around Comes Around,” *Architecture*, 2005. [Online]. Available: http://www.w3.org/TR/xpath/

[2] J. Krueger, C. Tinnefeld, M. Grund, A. Zeier, and H. Plattner, “A case for online mixed workload processing,” in *Proceedings of the Third International Workshop on Testing Database Systems*, ser. DBTest ’10. New York, NY, USA: ACM, 2010, pp. 8:1–8:6. [Online]. Available: http://doi.acm.org/10.1145/1838126.1838134

[3] H. Plattner and A. Zeier, *In-Memory Data Management: An Inflection Point for Enterprise Applications*. Springer, 2011.

[4] G. Knolmeyer, P. Mertens, A. Zeier, and J. Dickersbach, *Supply Chain Management Based on SAP Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[5] S. Your and S. Chain, “In Search of the Perfect Bill of Materials ( BoM ),” *Material Interchnage Journal*, no. March, 2002.

[6] T. Grust, “Accelerating XPath location steps,” *Proceedings of the 2002 ACM SIGMOD international conference on Management of data - SIGMOD ’02*, p. 109, 2002.

[7] P. Boncz, T. Grust, M. V. Keulen, S. Manegold, J. Rittinger, J. Teubner, and C. W. I. Amsterdam, “MonetDB / XQuery : A Fast XQuery Processor Powered by a Relational Engine,” 2006.

[8] Y. Chi, “Frequent Subtree Mining — An Overview,” *Fundamenta Informaticae*, pp. 1001–1038, 2001.

[9] M. J. Zaki, “Efficiently mining frequent trees in a forest,” *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD ’02*, p. 71, 2002.

[10] P. A. Boncz, J. Flokstra, T. Grust, M. van Keulen, S. Manegold, K. S. Mullender, J. Rittinger, and J. Teubner, “Monetdb/xquery-consistent and efficient updates on the pre/post plane,” in *EDBT*, ser. Lecture Notes in Computer Science, Y. E. Ioannidis, M. H. Scholl, J. W. Schmidt, F. Matthes, M. Hatzopoulos, K. Böhm, A. Kemper, T. Grust, and C. Böhm, Eds., vol. 3896. Springer, 2006, pp. 1190–1193.

[11] Y. E. Ioannidis, M. H. Scholl, J. W. Schmidt, F. Matthes, M. Hatzopoulos, K. Böhm, A. Kemper, T. Grust, and C. Böhm, Eds., *Advances in Database Technology - EDBT 2006, 10th International Conference on Extending Database Technology, Munich, Germany, March 26-31, 2006, Proceedings*, ser. Lecture Notes in Computer Science, vol. 3896. Springer, 2006.

[12] T. Grust, S. Sakr, and J. Teubner, “Xquery on sql hosts,” in *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30*, ser. VLDB ’04. VLDB Endowment, 2004, pp. 252–263. [Online]. Available: http://dl.acm.org/citation.cfm?id=1316689.1316713

[13] H. Plattner, “A common database approach for oltp and olap using an in-memory column database,” in *Proceedings of the 35th SIGMOD international conference on Management of data*, ser. SIGMOD ’09. New York, NY, USA: ACM, 2009, pp. 1–2. [Online]. Available: http://doi.acm.org/10.1145/1559845.1559846

[14] G. Valiente, “Algorithms on trees and graphs,” *Security*, 2002.

[15] T. Fiebig, S. Helmer, C.-C. Kanne, G. Moerkotte, J. Neumann, R. Schiele, and T. Westmann, “Anatomy of a native XML base management system,” *The VLDB Journal The International Journal on Very Large Data Bases*, vol. 11, no. 4, pp. 292–314, Dec. 2002.

[16] M. M. Tseng, “Generic Bill-of-Materials-and-Operations for High-Variety Production Management,” *Concurrent Engineering*, vol. 8, no. 4, pp. 297–321, Dec. 2000.

[17] X. Shen, X. Papademetris, and R. T. Constable, “Graph-theory based parcellation of functional subunits in the brain from resting-state fMRI data.” *NeuroImage*, vol. 50, no. 3, pp. 1027–35, Apr. 2010.

[18] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Second Edition*, 2001.

[19] P. Umversay and W. Lafayette, “Pattern Matching in Trees,” *Computing*, vol. 29, no. I, pp. 68–95, 1982.

[20] T. Grust, “Staircase Join : Teach a Relational DBMS to Watch its ( Axis ) Steps,” *Evaluation*.

[21] K. Beyer, N. Seemann, T. Truong, B. Van der Linden, B. Vickery, C. Zhang, R. J. Cochrane, V. Josifovski, J. Kleewein, G. Lapis, G. Lohman, B. Lyle, F. Özcan, and H. Pirahesh, “System RX,” *Proceedings of the 2005 ACM SIGMOD international conference on Management of data - SIGMOD ’05*, p. 347, 2005.

[22] M. H. Xiong, S. B. Tor, and L. P. Khoo, “WebATP: a Web-based flexible available-to-promise computation system,” *Production Planning & Control*, vol. 14, no. 7, pp. 662–672, Jan. 2004.

# About Top-k Flexible Queries in Large Databases

Khaoula Mabrouki
*Informatique*
*FST*
*Tunisia*
*mab.khaoula@gmail.com*

Amel Grissa Touzi
*Technologies of Information and Communications*
*ENIT*
*Tunisia*
*amel.touzi@enit.rnu.tn*

Habib Ounalli
*Informatique*
*FST*
*Tunisia*
*habib.ounelli@fst.rnu.tn*

***Abstract*—The problem of obtaining efficient answers to top-k queries has attracted a lot of research attention. Unfortunately, current top-k query processing techniques focus on Boolean queries, and cannot be applied to the large Data Bases (DB) seen the gigantic number of data. In this paper, we propose a new approach for top-k flexible queries taking into account another degree of granularity in the process of the evaluation of the query. We start by generating a Meta-DB formed by a set of clusters resulting of a preliminary fuzzy classification on the data. This set represents a reduced view of the initial DB and permits to deduct the semantics of the initial DB. We prove that our approach permits an optimal search of the relevant data sources and generate automatically the better k answers while proposing a new operator called stratified operator for taking into account the user's preferences.**

***Keywords-Large databases; flexible query; preference; formal concept analysis; Top k.***

## I. Introduction

An issue in extending databases is to increase the expressiveness of query languages. Based on fuzzy set theory, flexible querying enables users to express preferences inside requirements. Particularly, at the level of the requests addressed to large databases, the integration of the preferences allows to obtain more relevant answers. Besides, the user can be interested to receive a limited number of answers, in mind of ”Top k queries ”, k represents an ideal number of answers that is recommended to reach [1]. Unfortunately, current top-k query processing techniques focus on Boolean queries, and cannot be applied to large DB seen the gigantic number of data. Indeed, the majority of these systems use a function of score which remains difficult to establish seen the voluminous number of data. In addition, the approaches presented in this context present several limits, in particular in the hold in account of the dependencies between the search criteria that permit to detect the unrealizable queries (having an empty answer) with the user and in the generation and the scheduling of the turned over approximate answers.

In [2], a flexible and cooperative database flexible querying approach within the fuzzy theory framework has been proposed. This approach contributes two promising shares compared to the similar approaches. The first, taking into account the semantic dependencies between the query search criteria to determine its realizability or not. The second contribution related to its cooperative aspect in the flexible querying.

To ensure these functionalities, they have proposed to construct mono-attribute Type Abstraction Hierarchy (TAH) and a Multi-attribute Type Abstraction Hierarchy (MTAH) [3]. Problems lie in: 1) The generation of TAH's and MTAH from relieving attributes, 2) The storage and the indexing of such structures, and 3) The update of HATM.

In this paper, we propose a new approach for top-k flexible queries taking into account another degree of granularity in the process of the evaluation of the query. Seen that before, the interrogation of flexible queries was applied to raw data (the tuples of the base), the idea is to change this level of granularity and apply the clustering operation, so the interrogation will focus necessarily on clusters. Thus, we start by generating a Meta-DB formed by a set of clusters resulting of a preliminary fuzzy classification on data. This set represents a reduced view of the initial BD and permits to deduct semantics of the initial DB. The data classification is to divide a data set into subsets, called classes, so that all data in the same class are similar and data from different classes are dissimilar. Thus:

- The number of clusters generated by a classification algorithm is always less than the number of objects starting on which we apply the classification algorithm.
- All objects belonging to the same cluster have the same properties.

In this context, the query is modelled knowing the set of clusters modelling the meta-DB. To generate the meta-DB, we use the concepts of Clustering and Formal Concept Analysis (FCA). The use of these methods is justified by 1) fuzzy clustering has been a very successful data analysis technique as demonstrated in different domains [4]. These techniques allow data to belong to several groups (or clusters) simultaneously, with different membership degrees; 2) FCA is a method for knowledge representation that takes advantage of the features of formal concepts [5].

The rest of the paper is organized as follows. Section II makes a review of flexible querying and Formal Concept Analysis. Section III presents the problems of the other approaches and the contributions of this paper. Section IV describes our database flexible querying approach. Section V presents an example of relieving query. Section VI presents the experimental study we conducted to validate our approach. Section VII studies the complexity of our approach. Finally, Section IX concludes the paper and gives some future works.

## II. Basic concepts

In this section, we present the basic concepts of flexible queries and Formal Concept Analysis (FCA).

### A. Flexible queries

The traditional systems of interrogation distinguish two categories of data: those which satisfy the search criteria and those which do not satisfy them. The principle of the flexible interrogation aims at extending this bipolar behaviour by introducing the concept of approximate pairing. Thus, an element returned by a request will be at least relevant according to its satisfaction degree to the constraints of interrogation. Four principal approaches have been proposed to express and evaluate the flexible queries: 1) Use of the secondary criteria [6][7], 2) Use of the distance and the similarity [8] [9], 3) Expression of the preferences with linguistic terms [10] and 4) Modelling of the inaccuracy by the fuzzy subsets theory [11][12]. A comparative study of the systems of flexible interrogation has been achieved in [13][14].

The problem of the expression of the users' preferences in the flexible queries received much attention these last years [11][15][16][17][18].In general, it is possible to distinguish two families of approaches for the expression of the preferences: implicit and explicit.

**In the implicit approach**, mechanisms of numerical scores, commensurable or not, are used to represent the preferences. In the first case, the values of preferences can be aggregated to deliver a total value and to define a total order on the answers. In the second case, when there is not commensurability, only a partial order of the answers, based on the order of Pareto [19], is possible for the incomparable classes of answers are built. This approach is detailed in [20] and is illustrated by the Skyline operator [21] or in PreferenceSQL [22].

**In the explicit approach**, the preferences are specified by binary relations of preferences and in the majority of the cases, a partial order is obtained on the tuples. In addition, the preferences can be regarded as being constraints (preferences obligatory) and wishes (optional preferences). This reveals that this bipolar vision [23] of the preferences makes it possible to bring a refinement of the set of the answers: to satisfy the constraints, then, if possible, wishes. The preferences of the users can also be expressed by criteria of selection based on fuzzy sets. The predicates are not then any more in *"all or nothing"* but can be more or less satisfied. Other researchers used charts to model the preferences on a great number of alternatives. As an example, we can quote the Conditional Preferences Networks (CP-Nets)[24], which constitute a chart appraisal for modelling the preferences.

Bosc et al. [25] suggest the introduction of the preferences in the form of subsets of n-uplets (stratified divisor). Thus, they used the terms of ***"stratified divisor"*** and ***"stratified division"***. Consequently, an element x of the dividend will be more acceptable as it will be associated with a large number of subsets $(S_i)$ defining the divisor. Three types of requests studied by Bosc et al. are expressed in SQL language, where the dividend can be an intermediate relation and the stratified divisor is given explicitly by the user or is the result from sub queries.

As example of principal systems of interrogation with preferences, we can quote, the systems PreferenceSQL [22] and Preference Queries [16] which are based on a partial order, consequently, they deliver to the user the not dominating tuples. Preference SQL also incorporates a concept of bipolarity in the Preferring clause. The system top-K queries [15][24]. uses an ad-hoc score function f and delivers the k better answers of the total order obtained by f. However, this score function remains difficult to establish. The SQLf language uses the fuzzy set theory to define the preferences and makes the assumption of commensurability. It offers a framework founded to combine obligatory preferences.

### B. Fuzzy Conceptual Scaling and FCA

Conceptual scaling theory is the central part in Formal Concept Analysis (FCA). It allows to embed the given data into a much more general scale than the usual chains and direct products of chains. In the direct products of the concept lattices of these scales, the given data can be embedded. FCA starts with the notion of a formal context specifying which objects have which attributes and thus a formal context may be viewed as a binary relation between the object set and the attribute set with the values $0$ and $1$.

In [26], an ordered lattice extension theory has been proposed: Fuzzy Formal Concept Analysis (FFCA), in which uncertainty information is directly represented by a real number of membership value in the range of $[0, 1]$. This number is equal to similarity defined as follows:

***Definition 1:*** The similarity of a fuzzy formal concept $C_1 = (\varphi(A_1), B_1)$ and its sub-concept $C_2 = (\varphi(A_2), B_2)$ is defined as:

$$S(C1, C2) = \frac{|\varphi(A_1) \cap \varphi(A_2)|}{|\varphi(A_1) \cup \varphi(A_2)|}$$

where $\cap$ and $\cup$ refer intersection and union operators on fuzzy sets, respectively;
$\varphi$ is the relation which associates degrees to the elements of a fuzzy set $I = X \times V$ (X is the set of objects and V is the set of attributes). Each pair $(x_i, v_j) \in I$ has a membership degree $\mu(x_i, v_j) \in [0, 1]$.

In [27][28], we showed how these FFCA are very powerful as well in the interpretation of the results of the fuzzy clustering and in optimization of the flexible query.

**Example:** Let a relational database describing travel, means of transport and hotels, which a passenger can reserve for business trips or for pleasure. The primary key of each relation is underlined:

**Travel** (idV, price, stay, date, typeStay, offer , idT, idH) **V**
**Transport** (idT, means, route, comfort) **T**
**Hotel** (idH, category, name, region, city, restaurant) **H**

Price $\in$ $[100, 10000]$, Stay $\in$ $[1, 12]$, Category $\in$ $[1, 5]$ and Comfort $\in [1, 8]$. The result of fuzzy clustering (using Fuzzy C-Means [29]) and the application of the $\alpha$-*Cut* are shown in Table I and Table II.
We use a Cut of a fuzzy context $U$, noted, $\alpha$-*Cut*, and defined as the inverse of the number of clusters obtained. It is given by the following expression:

$$\alpha\text{-}Cut(U) = (c)^{-1}$$

For Stay attribute (respectively Price, Category and Comfort), fuzzy clustering generates three clusters (C1, C2 and C3) (respectively three clusters (C4, C5 and C6), two clusters (C7 and C8) and three clusters (C9, C10 and C11)). In our example, $\alpha$-*Cut*(Stay) = 0.3, $\alpha$-*Cut*( Price) = 0.3, $\alpha$-*Cut*(Category) = 0.5 and $\alpha$-*Cut*(Comfort) = 0.3.

Table I
FUZZY CONCEPTUAL SCALES FOR STAY AND PRICE ATTRIBUTES WITH $\alpha$-*Cut*

| | Stay | | | Price | | |
|---|---|---|---|---|---|---|
| | C1 | C2 | C3 | C4 | C5 | C6 |
| T1 | - | 0.67 | - | 0.86 | - | - |
| T2 | - | 0.94 | - | 0.94 | - | - |
| T3 | 0.98 | - | - | 0.47 | 0.47 | 0.47 |
| T4 | - | 0.94 | - | 0.35 | - | 0.35 |
| T5 | - | 0.95 | - | 0.49 | - | 0.45 |
| T6 | 0.83 | - | - | 0.67 | 0.27 | 0.05 |

Table II
FUZZY CONCEPTUAL SCALES FOR CATEGORY AND COMFORT ATTRIBUTES WITH $\alpha$-*Cut*

| | Category | | Comfort | | |
|---|---|---|---|---|---|
| | C7 | C8 | C9 | C10 | C11 |
| T1 | 0.98 | - | - | 0.81 | - |
| T2 | 0.92 | - | - | 0.88 | - |
| T3 | 0.62 | - | - | - | 0.85 |
| T4 | - | 0.93 | 0.61 | - | 0.34 |
| T5 | - | 0.9 | - | - | 0.89 |
| T6 | 0.53 | - | 0.51 | 0.43 | - |

## III. PROBLEMS AND CONTRIBUTIONS

The majority of the current approaches presented to support flexible queries have several limits, in particular 1) in the consideration of the dependencies between the search criteria that permit to detect the unrealizable requests (having an empty answer) with the user 2) and in the generation and the scheduling of the turned over approximate answers.

At the level of the requests addressed to large databases, the current top-k query processing techniques focus on Boolean queries, and cannot be applied to the large DB seen the gigantic number of data. The majority of these systems [15][24] uses an ad-hoc score function $f$ and delivers the k better answers of the total order obtained by f. However, this score function remains difficult to establish seen the voluminous number of data.

In this paper, we propose a new approach for top-k flexible queries taking into account another degree of granularity in the process of the evaluation of the query.
The advantage of this approach is that it can be applied to the large DB and that it does not require modifying the SQL language. The contributions of our approach are (1) the extraction of the dependencies between the search criteria to detect the unrealizable query; (2) an optimal search of the relevant data sources for a given query; (3) the automatic generation of the better k answers while proposing a new operator called stratified operator for taking into account the user's preferences.

## IV. THE PROPOSED APPROACH

In this section, we propose a relieving approach within the fuzzy set framework. We consider a relational database containing relieving attributes i.e. attributes which the users can use in a predicate of comparison containing a linguistic term.

In this paper, we limit ourselves to the relieving numerical attributes. Figure 1 shows the proposed approach.
This approach consists on three principal steps:

1) Generate the meta-DB: Apply a fuzzy algorithm of classification (example FCM): this allows generating clusters which overlap. Each cluster represents a set of the data verifying the same properties.
2) Deduct the semantic of the data: Represent the matrix obtained in the first step under the form of fuzzy concept lattice (FCA).
3) Generate the k better answers: Apply a stratified operator.

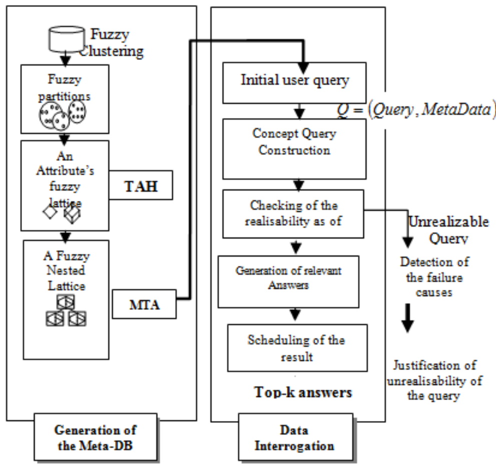

Fuzzy Clustering
Fuzzy partitions
An Attribute's fuzzy lattice
TAH
A Fuzzy Nested Lattice
MTA
Generation of the Meta-DB
Initial user query
Q = (Query, MetaData)
Concept Query Construction
Checking of the realisability as of
Unrealizable Query
Detection of the failure causes
Generation of relevant Answers
Scheduling of the result
Top-k answers
Justification of unrealisability of the query
Data Interrogation

Figure 1. Proposed approach

In this part, we present the theoretical foundations of the proposed approach based on the following properties:

- The number of clusters generated by a classification algorithm is always lower than the number of starting objects to which one applies the classification algorithm.
- All objects belonging to one same cluster have the same proprieties. These characteristics can be deduced easily knowing the center and the distance from the cluster.
- The size of the lattice modelling the properties of the clusters is lower than the size of the lattice modelling the properties of the objects.
- The management of the lattice modelling the properties of the clusters is optimum than the management of the lattice modelling the properties of the objects since the number of clusters is less than to the number of database objects.

To model the expression of the users' preferences, and generate the top-k answers we define **the stratified operator** as follows:

***Definition** 2:* The **stratified operator** r whose schema is R (A, X) by the relation s whose schema is S(B) is expressed according to the mechanism of partitioning and a similar expression is suggested here, as follows [25]:

**select top k X from r [where condition] group by X having set(A) contains** $v_{1,1}, \dots, v_{1,j_1} and - or \dots and - or v_{n,1}, \dots, v_{n,j_n}$.
This expression infers an order on the elements of the divisor namely $S = (S_1 = v_{1,1}, \dots, v_{1,j_1}) \succ \dots \succ (S_n = v_{n,1}, \dots, v_{n,j_n})$ where a $\succ$ b denotes the preference of a over b.

An ordinal scale L with labels $l_i$ (such as $l_1 > \dots > l_n > l_{n+1}$) is associated with this relation and it is used to attribute levels of satisfaction to the elements of the result of a stratified division ($l_1$ corresponds to the maximal satisfaction and $l_{n+1}$ express the refusal); they are the counterpart of 1 and 0 in the unit interval.

The principle of interpretation of these queries is to involve all levels for which the association is completely valid. An element is the more preferred as it is associated with a set $S_i$ so strongly preferred. The degree of satisfaction of x is expressed by a vector $V(x)$ of dimension $n$ where $V(x)[i] \in ]0..1]$ if $x$ is associated with the fuzzy values of $S_i$, 0 otherwise. The classification of elements means comparing these vectors according to the lexicographical order($\succ_{lex}$):

$$x \succ y \Leftrightarrow V(x) \succ_{lex} V(y) \Leftrightarrow \exists k \in [1, n]$$
$$that \ \forall j < k, V_j(x) = V_j(y) and V_k(x) > V_k(y)$$

The scale L is not used then directly, but we notice that the order obtained reflects it in the sense that if $i < j$, $V_i(x)$ is more important than $V_j(x)$ quite as $l_i > l_j$. It would nevertheless be possible to use a symbolic scale to make the comparison of elements within the framework of this request. The scale in question have $2^n$ levels. This would be made by means of a function transforming a vector $V$ into a whole score as follows:

$$SAT(x) = \sum_{i=1}^{n} (V_i(x) * 2^{(n-i)}) \qquad (1)$$

It is easy to show that the preference of x on y so defined is equivalent to $SAT(x) > SAT(y)$.

## V. Example of relieving query

For better explaining this step, we consider a relational database table describing travel, means of transport and hotels. Let's the following query:

```
Select   V.idV, V.price, T.means,
         T.comfort , H.idH, H.name
From     TRAVEL V, TRANSPORT
         T, HOTEL H
Where    V.idH = H.idH
And      V.idV = T.idT
And      T.means = 'plane'              (A1)
And      T.route = 'direct'             (A2)
And      V.offer <> 'circuit'           (A3)
And      V.typeStay = 'full-board'      (A4)
And      V.stay = 7                     (Pref1)
And      V.price = 800                  (Pref2)
And      H.category = 3                 (Pref3)
And      T.comfort = 3                  (Pref4)
```

In this query, the user wishes that his preferences be considered according to the descending order: Stay, Price, Category and Comfort with Top-k=3. In other words, returned data must be ordered and presented to the user according to these preferences. Without this flexibility, the user must refine these search keys until obtaining satisfaction if required since it does not have precise knowledge on the data which it consults.

According to the criteria of the query $Q$ , only the ($Pref_1$, $Pref_2$, $Pref_3$ and $Pref_4$) criteria correspond to relaxable attributes. Initially, we determine, starting from the DB, the tuples satisfying the non relaxable criteria $A_1$,$A_2$,$A_3$ and $A_4$.

```
Select   V.idV, V.price, T.means, T.comfort ,
         H.idH, H.nom
From     TRAVEL V, TRANSPORT T, HOTEL
         H
Where    V.idH = H.idH
And      V.idV = T.idT
And      T.means = 'plane'                    (A1)
And      T.route = 'direct'                   (A2)
And      V.offer <> 'circuit'                 (A3)
And      V.typeStay = 'full-board'            (A4)
```

These tuples, answering this request, are broken up into clusters according to labels of the relaxable attributes Stay, Price, Category and Comfort.

Following this operation of clustering, the expert can assign linguistic terms to the clusters generated for each relaxable attribute. The minimum value (resp. maximum) of each cluster corresponds to the lower (resp. higher) interval terminal of its values (of this cluster). For example, the linguistic terms *"Short, Medium and High"* (respectively *"Low, Medium and High"* , *"Medium and High"* and *"Low, Medium and High"*) will be associated to the relaxable attribute Stay (respectively Price, Category and Comfort).

## A. Construction of the query concept

We define a query concept $Q = (Q_A, Q_B)$ where $Q_A$ is a name to indicate a required extension and $Q_B$is the set of clusters describing the data reached by the query. The set $Q_B$ of clusters is determined by the following procedure:

```
Procedure Construction of the query concept
Input: Vector $V(A) = \{v_j : j = 1, \ldots, C(A)\}$ of cluster
centers of relaxable attribute $A$ and
the value of $Q$ associated to this last.
Output: Query concept $Q = (Q_A, Q_B)$ .
Begin
Step 1: Calculate the membership degrees of the specified
clusters for each value of the criterion of $Q$ associated to the
relaxable attribute $A$.
Step 2: Apply $\alpha - Cut$ to generate the fuzzy context.
Step 3: Form the set $Q_B$ of clusters whose membership is higher than the $\alpha - Cut$ value.
End Procedure
```

Table III and Table IV present the membership degrees associated to the query. These degrees are obtained while basing on memberships matrix obtained by a fuzzy clustering algorithm. Then, we apply the $\alpha - Cut$ for each attribute to minimize the number of concepts.

Table III
QUERY MEMBERSHIPS DEGREES (STAY AND PRICE)

| Stay | | | Price | | |
|---|---|---|---|---|---|
| C1 | C2 | C3 | C4 | C5 | C6 |
| 0.353 | - | 0,331 | 0,333 | 0,333 | 0,333 |

Table IV
QUERY MEMBERSHIPS DEGREES (CATEGORY AND COMFORT)

| Category | | Comfort | | |
|---|---|---|---|---|
| C7 | C8 | C9 | C10 | C11 |
| 0.516 | - | 0,334 | 0,351 | - |

According to our example, the query $Q$ seek the data sources having the metadata $Q_B = \{C1, C3, C4, C5, C6, C7, C9, C10\}$.

## B. Generation of the approximate answers

In our example, the request $Q$ is realizable then we can build his conceptlattice. Figure 2 represents the concept lattice associated with this request. The concepts are either new concepts, or concepts modified following construction of the request $Q$ knowing the HATM. These concepts are the only ones which divide clusters with the request and which can thus contain relevant answers.

Let given a query $Q = (Q_A, Q_B)$, all the relevant data sources are in the extension of $Q$ and of its subsumers in the concepts lattice since the intention of each one of these concepts are included in $Q_B$(the intention of the query concept).

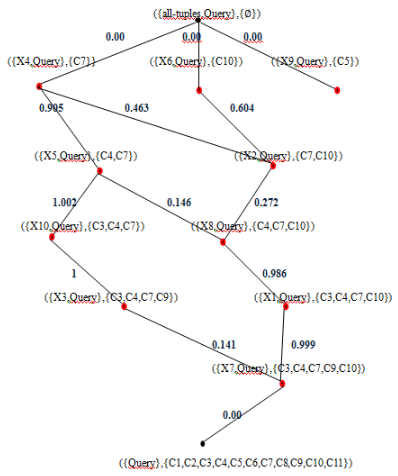


Figure 2. Concept lattice associated to the query

## C. Generation of the Top-K answers

Relevant data sources can be sorted according to the distance separating the concepts in the lattice. This step consists of ordering the n-uplets obtained according to their satisfaction degrees of the initial query.

The satisfaction degree corresponds to the similarity of a fuzzy formal concept and its sub-concept defined in Definition 1. In our example, these degrees are given in Table V. To model the expression of the users' preferences,

Table V
SATISFACTION DEGREE OF THE GENERATED ANSWERS

| Data sources | Meta data |
|---|---|
| E1 | C3,C4,C7,C10 |
| E2 | C7,C10 |
| E3 | C3,C4,C7,C9 |
| E4 | C7 |
| E5 | C4,C7 |
| E6 | C10 |
| E7 | C3,C4,C7,C9,C10 |
| E8 | C4,C7,C10 |
| E9 | C5 |
| E10 | C3,C4,C7 |

and generate the top-k answers, we use the Fuzzy stratified divisor, our query is rewritten in this form:

| | |
|---|---|
| **Select** | V.idV, V.price, T.means, T.comfort , H.idH, H.name |
| **From** | TRAVEL **V**, TRANSPORT **T**, HOTEL **H** |
| **Where** | V.idH = H.idH |
| **And** | V.idV = T.idT |
| **And** | T.means = 'plane' |
| **And** | T.route = 'direct' |
| **And** | V.offer <> 'circuit' |
| **And** | V.typeStay = 'full-board' |
| **having set** | (HOTEL) |
| **contains** | {Short_Stay, Long_Stay} **and-or** {Low_Price, Medium_Price, High_Price} **and-or** { Medium_Category } **and-or** { Low_Comfort , Medium_Comfort}. |

In our example, the scale $L = l_1 > l_2 > l_3 > l_4$ with $l_1 = \{C1, C3\}, l_2 = \{C4, C5, C6\}, l_3 = \{C7\}$ and $l_4 = \{C9, C10\}$. We have in this case four strata. Each line of the table represents a set of answers E1,E2,. . . ,E10. Let S the divisor l1 $\succ$ l2 $\succ$ l3 $\succ$ l4 and the dividend: r = {(E1, {C3,C4,C7,C10}), (E2, {C7,C10}), (E3, {C3,C4,C7,C9}), (E4, {C7}), (E5, {C4,C7}), (E6, {C10}), (E7,{C3,C4,C7,C9,C10}), (E8, {C4,C7,C10}), (E9, {C5}), (E10,{C3,C4,C7})}.

The degree of satisfaction of x is expressed by a vector $V(x)$ of dimension $n$ where V(x)[i] $\in$ $]0 \ldots 1]$, if x is associated with all the values of $S_i$, it represent the Query Memberships degrees of the associate cluster in the query (Table III and Table IV), 0 otherwise.

$V(X1) = (0.331, 0.333, 0.516, 0.351)$
$V(X2) = (0, 0, 0.516, 0.351)$
$V(X3) = (0.331, 0.333, 0.516, 0.334)$
$V(X4) = (0, 0.516, 0, 0)$
$V(X5) = (0, 0.333, 0.516, 0)$
$V(X6) = (0, 0, 0, 0.351)$
$V(X7) = (0.331, 0.333, 0.516, 0.685)$
$V(X8) = (0, 0.333, 0.516, 0.351)$
$V(X9) = (0, 0.333, 0, 0)$
$V(X10) = (0.331, 0.333, 0.516, 0)$

Using Expression 1, Table III, Table IV and Table V, we obtain:

$SAT(E1) = 0.331 * 2^3 + 0.333 * 2^2 + 0.516 * 2^1 + 0.351 * 2^0 = 5.363$
$SAT(E2) = 0 * 2^3 + 0 * 2^2 + 0.516 * 2^1 + 0.351 * 2^0 = 1.383$
$SAT(E3) = 0.331 * 2^3 + 0.333 * 2^2 + 0.516 * 2^1 + 0.334 * 2^0 = 5.346$
$SAT(E4) = 0 * 2^3 + 0 * 2^2 + 0.516 * 2^1 + 0 * 2^0 = 1.032$

$SAT(E5) = 0*2^3 + 0.333*2^2 + 0.516*2^1 + 0*2^0 = 2.346$
$SAT(E6) = 0*2^3 + 0*2^2 + 0*2^1 + 0.351*2^0 = 0.351$
$SAT(E7) = 0.331*2^3 + 0.333*2^2 + 0.516*2^1 + 0.658*2^0 = 5.67$
$SAT(E8) = 0*2^3 + 0.333*2^2 + 0.516*2^1 + 0.351*2^0 = 2.715$
$SAT(E9) = 0*2^3 + 0.333*2^2 + 0*2^1 + 0*2^0 = 1.332$
$SAT(E10) = 0.331*2^3 + 0.333*2^2 + 0.516*2^1 + 0*2^0 = 5.012$
Thus, E7 $\succ$ E1 $\succ$ E3 $\succ$ E10 $\succ$ E8 $\succ$ E5 $\succ$ E2 $\succ$ E9 $\succ$ E4 $\succ$ E6;
Now, suppose that the user wishes to have the 30 Top answers (30 best answers). In that case, the tuples returned are the first 30 tuples starting with all $E7$ and so on.

## VI. Experimentations

### A. Context

The general principle used to implement the previous queries is based on the use of SQL queries to access data encapsulated and calculate the satisfaction degree (denoted by SAT) assigned to each element of the result. The proposed method has the following characteristics:

1) It is based on the usual way of expressing a division with the counting function and
2) It benefited from the stratification of the divisor to access primarily to user preferences with the highest priority. The strata are traversed in decreasing order of importance ($S_1$ to $S_n$), which has a real impact for those requests.

### B. Experimental Results

The aim of the experiments is to evaluate the additional cost due to the inclusion of preferences in the division queries. Queries are evaluated with dividend relations of different size (300, 1000, 3000, 5000, 10000 and 15000 tuples), with a divisor composed of four strata (i.e four preferences). The results obtained are reported in the tables below, where:

- Each instance was executed 10 times to avoid the variable load of the machine used,
- The size of the result (K) is 10 (respectively 30, 100) for a dividend of 300 (respectively 1000, 3000, 5000, 10000 and 15000) tuples,

Table VI
The experimental results for the operation of the Division

| | Processing time(s) | | | Used Memory(MB) |
|---|---|---|---|---|
| Number of tuples | K=10 | K=30 | K=100 | |
| 300 | 3.236 | 3.487 | 4.210 | 0.168 |
| 1000 | 4.052 | 4.356 | 4.923 | 0.215 |
| 3000 | 4.573 | 4.789 | 5.002 | 0.275 |
| 5000 | 4. 590 | 4.706 | 5.403 | 2.808 |
| 10000 | 4. 661 | 4.776 | 5.624 | 3.636 |
| 15000 | 4.789 | 5.129 | 5.942 | 3.779 |

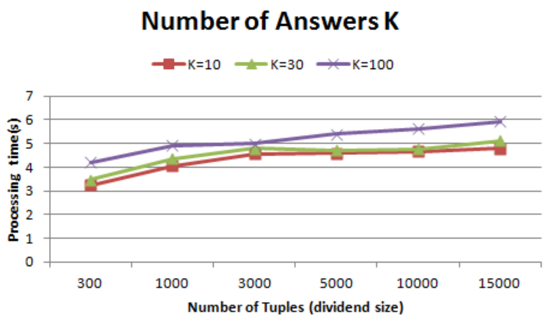


Figure 3. Variation of the processing time according to the variation of K

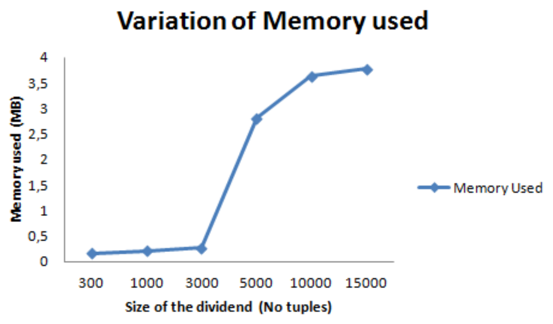


Figure 4. Variation of the memory used for the Division Operation

In Table VI and as shown in both figures (3 and 4), we varied the number of answers requested by the user (Top K answers) to assess its impact on the processing time and the memory used. Thus, we note that the number K does not have a major effect on the processing time since the division operation retrieves tuples (answering the query) that are already listed and highly ranked in a text file.

1) The cost of the division operation depends linearly, as expected, on the size of the dividend,
2) The number of answers (K) required by the user does not have a major influence on the execution time,

These first results are quite encouraging, even if they must be completed to achieve more definitive conclusions about the best way to develop tools over an existing DBMS or otherwise intervening in the kernel of the DBMS to run these queries.

## VII. Study of Complexity

A study of spacial and temporal complexities of the proposed approach is presented in this section.

- Space complexity: In the field of space complexity, we store only XML files. The clusters of the relaxable attributes are not stored any more in the KBAR( Knowledge Base of Relaxable Attributes). What constitutes an asset for the practical application of this approach.

- Temporal complexity: It includes the following costs: a) construction of the clusters of the relaxable attributes, b) construction of lattice (with the algorithm of Ganter) and c) scheduling of the approximate answers.

For the construction of the clusters of the relaxable attributes , we calculated the theoretical complexity of the approaches of clustering suggested. It is equal to $O(Nc^2)$, where $N$ corresponds to the number of data and $c$ is the maximum number of clusters. For the construction of the lattice, temporal complexity depends on the method of adopted construction. In our approach, we were interested in the method of Ganter [30] where its complexity is $O((max(|N|, |n|)).(|N|.|n|))$.

Thus, the total complexity is equal to $O(Nc^2) + O((max(|N|, |M|)).(|N|.|M|)) + O(n * level)$, where $N$ corresponds to the number of data (the objects of the DB), $M$ is the number of attributes, $c$ is the number of clusters, $n$ is the number of concepts from the lattice and $level$ corresponds to the number of levels present in the lattice.
We present in Table VII a study of the complexity of some algorithms of construction of the lattices.

Table VII
STUDY OF TEMPORAL COMPLEXITY OF LATTICE CONSTRUCTION ALGORITHMS

| Algorithm | Temporal Complexity |
|---|---|
| Bordat [31] | $O(n.|N|.(|N| + |M|))$, $n$ is the number of concepts |
| Nourine et Raynaud [32] | $O(n.|N|.(|N| + |M|))$, $n$ is the number of concepts |
| Ganter [30] | $O((max(|N|, |M|)).(|N|.|M|))$ |
| Godin [33] | Quadratic compared to the number of elements in the lattice of concepts. |

## VIII. COMPARISON WITH OTHER APPROACHES

In this section, we present the essential idea of the principal existing flexible querying approaches closest to our approach. Those differ primarily by the manner used to find the values closest to those required by the user and the used formalism to model the uncertainty and the imperfection of the real world.

The literature on the flexible querying and the co-operative systems abounds. We can distinguish three principal categories. The first category, indicated by C1, includes ”ad hoc” approaches specific to particular systems. The objective of such approaches is the introduction of flexibility by the use of linguistic terms and the specification of the preferences of the users between the various search keys from the desired data. Among the approaches of C1, we can quote the systems ARES [9], MULTOS [11], SEAVE [34], FLEX [35]. Second category approaches indicated by C2, use the formalism of sets and fuzzy logic to model in addition to the imperfection and the uncertainty of the real world, the evaluation of the query known as vague or fuzzy.

The principal common point between these approaches is the modification of the query language, generally. This modification consists in introducing vague linguistic terms, like ”accessible price” or ”large budget”, and of the operators of approached comparison like ”Near-to” and ”similar-to” of the system CoBase [4]. To not modify the DBMS system, these systems add an additional layer charged to transform a fuzzy query into a traditional one known as ”wraps query”. This one is subjected to the target DBMS for evaluation. Its result is then filtered according to preferences of the user before being presented to him. This process of transformation and filtering is based on established properties of the sets and fuzzy logic.

The third category, indicated by C3 comprises approaches which lie within the scope of the artificial intelligence techniques and aims at determining tacit knowledge starting from the explicit data. Several systems like DBLEARN [36], DB-Discover [37] and GBDR [38], belong to this category. Generated knowledge is in the form of rules or of hierarchy of concepts.
The results obtained by these approaches, in particular within clustering, are of a great utility for this work.

The contributions of approaches of C2, such as for example those of CoBase [4], are significant, in particular the concepts of TAH and MTAH to model generalization and specialization by hierarchies of concepts. However, we estimate these systems remain demanding with respect to the end-users. For example, in CoBase, the operators used require a precise knowledge of the contents of the database,. It does not detect the realizability of a query only after its execution. CoBase can also generate false answers. The users must also know the organization of the database since they must specify the attributes which they must release or not as well as the level of relieving of each attribute.

In [2], no modification of SQL is necessary, which constitutes an asset for the practical application of this approach. The user is not solicited to make choices during relieving, which can be hazardous, as it is the case in several systems such as Flex, Vague [10] and CoBase, to quote only those.

In this approach, the relieving attributes are fixed by the administrator of the database. This is the more significant, since the approach suggested is addressed to end-users not having the knowledge precise and detailed on the organization and the data which they consult. It is easier to an expert to specify that a price attribute of a table of the database is relaxable and than it can be used with the terms "weak" or "accessible". This is easier than to use the operator "Within" $(100, 120, 150, 300)$ of CoBase. However, this approach present limits at the level of the structures which it uses. We quote:

a. The incremental maintenance of the base of knowledge of the relaxable attributes (KBAR),
b. The clustering of the relaxable attributes without fixing a priori the number of clusters; and
c. The problem of storage of the clusters and indexing of the MTAH.

In the proposed approach, the clusters generated for each relaxable attribute are not stored any more in the catalogue of the DBMS. So, the maintenance of this meta-base does not pose any problem. Indeed, to be able to trace the lattices, it is quite simply necessary to charge an XML file which makes it possible to recover all information necessary to the tracing of these lattices. XML parsers recover information and recall the lattice starting from the methods of constructions of these structures. In this file are backed up:

- The title of the lattice.
- Identifiers of the concepts, their positions with the styles of the labels of the objects and attributes of the concept.
- The set of data and attributes of each concept.
- The set of the arcs and the concepts which they bind.

This parser also allows curing the problem of storage of the clusters and indexing of the MTAH.

The problem of clustering does not arise with this approach since the approaches of clustering suggested allow, in addition to the optimization of the number of clusters, the evaluation of the quality of the latter. Finally, at the level of the requests addressed to large databases, the current top-k query processing techniques focus on Boolean queries, and cannot be applied to the large DB seen the gigantic number of data. The majority of these systems uses an ad-hoc score function f and delivers the k better answers of the total order obtained by f. However, this score function remains difficult to establish seen the voluminous number of data.

## IX. Conclusion

Several algorithms of the top-k retrieval problem have been introduced in recent years. Unfortunately, these techniques cannot be applied to the large DB seen the gigantic number of data. The majority of these systems uses a score function which remains difficult to establish seen the voluminous number of data.

In this paper, we have proposed a new approach for top-k flexible queries taking into account another degree of granularity in the process of the evaluation of the query. The proposed approach consists of the following steps: 1) Generation the meta-DB, for this we apply a fuzzy algorithm of clustering. Each cluster represents a set of data verifying the same properties; 2) Deduction of the semantic of the data, we represent the matrix obtained in the first step under the form of fuzzy concept lattice (FCA); and 3) Generation of the k better answers while proposing a new operator called stratified operator for taking into account the user's preferences.

The contributions of this approach are (1) the extraction of the dependencies between the search criteria to detect the unrealizable query; (2) the optimal search of the relevant data sources for a given query; (3) the automatic generation of the k better answers. This work can be spread while proposing to adapt our method to the large fuzzy DB.

## References

[1] R. Fagin and E. Wimmers, *Incorporating User Preferences in Multimedia Queries*,ICDT '97 Proceedings of the 6th International Conference on Database Theory Delphi, Greece, pp. 247-261, January 1997.

[2] H. Ounalli and R. Belhadj, *Interrogation flexible et coopérative d'une BD par abstraction conceptuelle hiérarchique*, INFORSID 2004, pp. 41-56.

[3] W.W. Chu, K. Chiang, C. Hsu, and H. Yau, *An Error-based Conceptual Clustering Method for Providing Approximate Query Answers*, Communications of ACM, 1996.

[4] W.W. Chu, H. Yang, K. Chiang, M. Minock, G. Chow, and C. Larson, *CoBase: A Scalable and Extensible Cooperative Information System*, Journal of Intelligence Information Systems, Boston, 1996.

[5] K. Uri and Z. Jianjun, *Fuzzy Clustering Principles, Methods and Examples*, Technical Report ,Technical University of Denmark,. Department of Control and Engineering Design (IKS), 13p, December 1998.

[6] R. Wille, *Lattices in data analysis: how to draw them with a computer*, In: I.Rival (ed.): Algorithms and order, Kluwer, Dordrecht-Boston, pp. 33-58, 1989.

[7] M. Lacroix and P. Lavency, *Preferences : Putting more knowledge into queries*.In Proceedings of the 13th International Conference on Very Large Data Bases, Morgan Kaufmann Publishers Inc., pp. 217-225, 1987.

[8] CL. Chang, *Decision support in an imperfect world*. Trends and applications on automating intelligent behavior: applications and frontiers, 25, 1983.

[9] T. Ichikawa and M. Hirakawa, *ARES: A relational database with the capability of performing flexible interpretation of queries*. IEEE Transactions on Software Engineering, 12(5), pp. 624-634, 1986.

[10] A. Motro, *VAGUE : A User Interface to Relational Databases that Permits Vague Queries*. ACM Transactions on Office Information Systems, 6(3), pp. 187-214, 1988.

[11] F. Rabitti and P. Savino, *Retrieval of multimedia documents by imprecise query specification*. Advances in Database Technology-EDBT'90, pp. 203-218, 1990.

[12] P. Bosc and O. Pivert, *SQLf : a relational database language for fuzzy querying*. IEEE Transactions on Fuzzy Systems, 3(1), pp. 1-17, 1995.

[13] V. Tahani, *A conceptual framework for fuzzy query processing-A step toward very intelligent database systems*. Information Processing and Management, 13(5), pp. 289-303, 1977.

[14] O. Pivert, *Contribution à l'interrogation flexible de bases de données : expression et évaluation de requêtes floues*. PhD thesis, Université de Rennes 1, 1991.

[15] P. Bosc, L. Liétard, and O. Pivert, *Databases and flexibility : gradual queries*. TSI. Technique et science informatiques, 17(3), pp. 355-378, 1998.

[16] P. Bosc and L. Liétard, *Aggregates computed over fuzzy sets and their integration into sqlf*. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 16(6), pp. 761-792, 2008.

[17] J. Chomicki, *Preference formulas in relational queries*. ACM Trans. Database Syst., 28(4), pp. 427-466, 2003.

[18] W. Kießling, *Foundations of preferences in database systems*. In Very Large Data Base (VLDB) Endowment Inc, pp. 311-322, 2002.

[19] L. Liétard, D. Rocacher, and S.E. Tbahriti, *Preferences and bipolarity in query languages*. Fuzzy Information Processing Society, New York City, NY, pp. 1-6, 2008.

[20] L. Liétard and D. Rocacher, *On the definition of extended norms and co-norms to aggregate fuzzy bipolar conditions*. In the European Society of Fuzzy Logic and Technology Conference, pp. 513-518, 2009.

[21] S. Börzsönyi, D. Kossmann, and K. Stocker, *The skyline operator*. In International Conference on Data Engineering (ICDE), pp. 421-430, 2001.

[22] W. Kießling and G. Köstler, *Preference sql - design, implementation, experiences*. In Very Large Data Base (VLDB) Endowment Inc, pp. 990-1001, 2002.

[23] D. Dubois and H. Prade, *Bipolarity in Flexible Querying*. Proceedings of the 5th International Conference on Flexible Query Answering Systems, FQAS '02, London, UK, pp. 174–182, 2002.

[24] C. Domshlak, H. Hoos, C. Boutilier, R. Brafman, and D. Poole, *Cp-nets : A tool for representing and reasoning with conditional ceteris paribus preference statements*. Journal of Artificial Intelligence Research, 21 , pp. 135-191, 2004.

[25] P. Bosc and O. Pivert, *A propos de requêtes à préférences et diviseur stratifié*, In INFORSID, pp. 311-326, 2010.

[26] C. Li, MA. Soliman, CK. Chang, and IF. Ilyas, *Ranksql : Supporting ranking queries in relational database management systems*. In Very Large Data Base (VLDB) Endowment Inc, pp. 1342-1345, 2005.

[27] T. Thanh, H.S. Cheung, and C.Tru Hoang, *A Fuzzy FCA-based Approach to Conceptual Clustering for Automatic Generation of Concept Hierarchy on Uncertainty Data*. Proceedings of the CLA 2004 International Workshop on Concept Lattices and their Applications, Ostrava, Czech Republic, pp. 1-12, 2004.

[28] A. Grissa Touzi, M. Sassi, and H. Ounelli, *An innovative contribution to flexible query through the fusion of conceptual clustering, fuzzy logic, and formal concept analysis*; International Journal of Computers and Their Applications. Vol. 16, N 4, pp. 220-233, December 2009.

[29] M. Sassi, A. Grissa Touzi, and H. Ounelli, *Clustering Quality Evaluation based on Fuzzy FCA*, 18th International Conference on Database and Expert Systems Applications, (DEXA'07), Regensburg, Germany, pp. 62-72, LNCS, Springer 2007.

[30] B. Ganter, *Two Basic Algorithms in Concept Analysis*. Technical report, Darmstadt, 1984.

[31] J. Bordat, *Calcul pratique du treillis de Galois dune correspondance* , Mathmatique, Informatique et Sciences Humaines, pp. 31-47, 1986.

[32] L. Nourine and O. Raynaud, *A fast algorithm for building lattice*, Information Processing Letters, vol. n 71, pp. 199-204, 1999.

[33] R. Godin and R. Missaoui, *An incremental concept formation approach for learning from databases*, Theoretical Computer Science, vol. n 133, pp. 387-419, 1994.

[34] A. Motro, *SEAVE: A Mechanism for verifying User Presuppositions in Query Systems*, ACM Transactions on Information Systems, pp. 312-330 ,1986.

[35] A. Motro, *FLEX: Tolerant and Cooperative User Interface to Database*, IEEE Transactions on Knowledge and Data Engineering, 4(4), pp. 231-246 , 1990.

[36] H.J. Hamilton and D.F. Fudger, *Estimating DBLEARNs Potential for knowledge Discovery in Databases*, Computational Intelligence, 11(2), 1995.

[37] C.B. Rivera and C.L. Carter, *A tutorial Guide to DB-Discovrer*, Version 2.0, Technical Report CS-95-05, University of Regina, pp. 280-296, 1995.

[38] C.L. Carter, H.J. Hamilton, W.B. Hase, and C. Rivera, *GDBR: An Optimal Relation Generalization Algorithm for Knowledge Discovery from Databases*, Department of Computer Science, University of Regina, 1998.

# Association Rule Mining from Large and Heterogeneous Databases with Uncertain Data using Genetic Network Programming

Eloy Gonzales* and Koji Zettsu*
* *Information Services Platform Laboratory*
*Universal Communication Research Institute*
*National Institute of Information and Communications Technology*
*3-5 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0289, Japan*
*Tel: +81-774-98-6866, Fax: +81-774-98-6960*
*e-mail: {egonzales, zettsu}@nict.go.jp*

***Abstract*—Association Rule Mining is one of the most important tasks in data mining and it has been deeply studied during last years. Recently several rule mining algorithms have been developed due to many real-world applications. Most of these studies have generally considered only precise data, which means that items within each datum or transaction are definitely known and precise. However, there are also many real life situations where the data is uncertain, which means that items are expressed in terms of existential probabilities. In this paper, a method for association rule mining from large, heterogeneous and uncertain databases is proposed using an evolutionary method named Genetic Network Programming (GNP). Some other association rule mining methods can not handle uncertain data directly, they are inapplicable or computational inefficient under such a model. GNP uses direct graph structure and is able to extract rules without generating frequent itemsets to improve mining efficiency. The performance of the method is evaluated through extensive experiments using real scientific large-scale heterogeneous databases that show its effectiveness and efficiency.**

***Keywords-Association rule mining; heterogeneous databases; uncertain data; evolutionary computation.***

## I. Introduction

The continuously growing in the size and number of databases in a variety of domains has boosted the develop of several data mining methods during the last decade. There is an increasing need to discover associations and relations among large and heterogeneous databases, which may be tackled by association rule mining. Actually, several association rule mining algorithms have been proposed. Most of them assume a data model, which *transactions* capture the doubtless facts about items contained in each transaction, that is, they handle *precise* data, such as databases of market basket transactions, web logs and click streams where the user definitely knows whether an item is present in, or is absent from, a transaction in the databases. However, in many other applications, the existence of an item in a transaction is best captured by a likelihood measure or probability, for example, a medical dataset may contain a list of patients as records (tuples) and for each record a set of symptoms or illnesses that a patient suffers as the items. Applying association analysis on such dataset allows to discover any potential correlation among the symptoms and illnesses, a physician may highly suspect (but cannot guarantee) that a patient suffers some disease. The uncertainty of such suspicion can be expressed in terms of *existential probability*. Other examples of uncertain datasets are pattern recognition databases where image processing techniques are applied on a picture to extract features that indicate the presence or absence of certain objects in an area, but due to noises and limited resolution, the presence of an object is ofter uncertain and expressed as probability.

Many of the developed algorithms for uncertain mining have been focused on data mining tasks like clustering and classification of uncertain data [1]. With respect to association rule mining of uncertain data, Chui et al.[2] proposed an Apriori-based algorithm called *U-Apriori* and introduced a trimming strategy to reduce the number of candidates that need to be counted by the algorithm. To speed up the mining process, they also proposed a decremental pruning technique, however as an Apriori-based algorithm, U-Apriori relies on the candidate generate-and-test paradigm. Kai-San Leung et al. [3] have tried to reduce the searching space by adding constraints given by users, but the scalability issues have not been described.

In this paper, a method for mining association rules from uncertain data is proposed using an evolutionary optimization algorithm named Genetic Network Programming (GNP). There have been some proposals of association rule mining using GNP [4][5], however all of them use certain data. The advantages of the proposed method are as follows: (1). It is widely known that the search space of frequent patterns from precise data is very huge, and from uncertain data is even much bigger. Thus, the proposed method extracts rules directly without generating the frequent patterns. (2). The support and confidence are the most used framework to evaluate the association rules. However, this measurements are not longer valid for probabilistic datasets. GNP provides the flexibility to use any correlation measure either independently or combined. Thus, in this paper *hyper-lift* and *hyper-confidence* proposed in [6] are used. (3). The scalability issue is not an important concern in other algorithms since most of them deal with mining frequent patterns, which is computationally expensive and therefore use relatively small datasets. In this paper, large-scale and heterogeneous databases are mainly focused.

The following sections of this paper are organized as follows: In Section II, the uncertain data model is presented, the concepts and explanations of general association rules are introduced in Section III, the outline of GNP is briefly reviewed in Section IV where also the method for rule extraction from uncertain data is presented. Simulation results are described in Section V, and finally, conclusion and future work are given in Section VI.

## II. Uncertain Data Model

Because of the uncertainty in various real-life situations, users may not be certain about the presence or absence of an item $x$ in a transaction $t_i$. They may suspect the presence of $x$ in $t_i$, but cannot guarantee. The uncertainty of such suspicion can be expressed in terms of *existential probability* $P(x, t_i)$, which indicates the likelihood of $x$ being present in $t_i$ in a probabilistic database $D$ of uncertain data. The existential probability $P(x, t_i)$ ranges from a positive value close to 0, which indicates that $x$ has an insignificantly low chance to be present in $D$, to a value of 1, which indicates that $x$ is definitely present. Applying this notion to the traditional databases of precise data, each item of any transaction can be viewed as an item with a 100% likelihood of being present in such a transaction.

## III. Association Rules

A transaction database consist of a series of transactions, each of them containing a subset of available items[7]. Let $I = \{A_1, A_2, \ldots A_l\}$ be a set of attributes. Let G be a set of transactions,where each transaction $T$ is a set of attributes such that $T \subseteq I$. Associated with each transaction is a unique identifier whose set is called $TID$. A transaction $T$ contains $X$, a set of some attributes in $I$, if $X \subseteq I$. An association rule is an implication of the form of $X \Rightarrow Y$, where $X \subset I, Y \subset I$, and $X \cap Y = \emptyset$. $X$ is called antecedent and $Y$ is called consequent of the rule. Both are called **itemsets**. In general, an itemset is a non-empty subset of $I$. There are some assumptions in our model, 1) transactions occur randomly following a homogeneous Poisson process. 2) all items occur independently of each other and for each item exist a probability of being contained in a transaction.

Looking at the observed co-occurrences counts for all pairs of two items $A_i$ and $A_j$, in a dataset with $N$ transactions, it is possible to form an $n \times n$ contingency table. Each cell can be modeled by a random variable $C_{ij}$, which given fixed marginal counts $c_i$ and $c_j$, follows a *hyper-geometric distribution*. In the case of two itemsets $X$ and $Y$, the random variable $C_{XY}$ follows a hyper-geometric distribution with the counts of the itemsets as its parameter [6], that is, the probability of counting exactly $k$ transactions, which contain the two independent itemsets $X$ and $Y$ is given by:

$$P(C_{XY} = k) = \frac{\binom{C_X}{k}\binom{N-C_X}{C_Y-k}}{\binom{N}{C_Y}} \tag{1}$$

The probability of counting more than $k$ transactions is:

$$P(C_{XY} > k) = 1 - \sum_{i=0}^{k} P(C_{XY} = i) \tag{2}$$

The contingency table is shown in Table I.

Table I
The contingency table of $X$ and $Y$.

| | $Y$ | $\neg Y$ | $\sum_{row}$ |
|---|---|---|---|
| $X$ | $C_{XY}$ | $C_X - C_{XY}$ | $C_X$ |
| $\neg X$ | $C_Y - C_{XY}$ | $(N - C_Y) - (C_X - C_{XY})$ | $N - C_X$ |
| $\sum_{col}$ | $C_Y$ | $N - C_Y$ | $N$ |

( $N$: the number of transactions $(= |TID|)$ )

### A. Hyper-Lift

The expected value of a random variable $C_{XY}$ for the co-occurrence counts for two itemsets $X$ and $Y$ is:

$$E(C_{XY}) = \frac{C_X C_Y}{N} \tag{3}$$

Therefore, lift can be written as:

$$\textit{lift}(X \Rightarrow Y) = \frac{C_{XY}}{E(C_{XY})} \tag{4}$$

However, it works well for items with a relatively high occurrence frequency. Thus, for relatively infrequent itemsets the hyper-lift is defined as:

$$\textit{hyper-lift}_{\delta}(X \Rightarrow Y) = \frac{C_{XY}}{Q_{\delta}(C_{XY})} \tag{5}$$

where, $Q_{\delta}(C_{XY})$ is the quantile distribution. The minimal value of the $\delta$ quantile of the distribution of $C_{XY}$ is defined by the following inequalities:

$$\begin{aligned} &P(C_{XY} < Q_{\delta}(C_{XY})) \leq \delta \ \text{, and} \\ &P(C_{XY} > Q_{\delta}(C_{XY})) \leq 1 - \delta \end{aligned} \tag{6}$$

### B. Hyper-confidence

The hyper-confidence is defined directly by the probability of realizing a count smaller that the observed co-occurrence count $c_{XY}$ given the marginal counts $c_X$ and $c_Y$ as follows:

$$\begin{aligned} \textit{hyper-confidence}(X \Rightarrow Y) &= P(C_{XY} < c_{XY}) \\ &= \sum_{i=0}^{c_{XY}-1} P(C_{XY} = i) \end{aligned} \tag{7}$$

where $P(C_{XY} = i)$ is calculated using Equation 1.

Note that hyper-confidence is equivalent to a special case of Fisher's exact test, the one-sided test on $2 \times 2$ contingency tables. In this case, the p-value is directly obtained from the hyper-geometric distribution, which is computationally negligible compared to the effort of counting support and finding frequent itemsets. Furthermore, for the application of mining association rules where only rules with positively correlated elements are of interest, a one-side test as used here is much more appropriate.

Therefore, the problem of mining probabilistic association rules from uncertain data is to find all rules that are highly likely to be interesting, that is, satisfying the minimum hyper-confidence threshold.

$$\begin{aligned} &\textit{hyper-confidence}(X \Rightarrow Y) \geq min_{hyper-conf} \text{ , and} \\ &\textit{hyper-lift}(X \Rightarrow Y) \geq 1 \end{aligned} \quad (8)$$

## IV. Genetic Network Programming

Genetic Network Programming (GNP) is one of the evolutionary optimization algorithms, which evolves directed graph structures as solutions instead of strings (Genetic Algorithms) or trees (Genetic Programming) [8], [9], [10]. The main aim of developing GNP was to deal with dynamic environments efficiently by using the higher expression ability of graph structures.

The basic structure of GNP is shown in Fig. 1. The graph structure is composed of three types of nodes that are connected on a network structure: a start node, judgment nodes (diamonds), and processing nodes (circles). Judgment nodes are the set of $J_1$, $J_2$, ..., $J_p$, which work as *if-then* conditional decision functions and they return judgment results for assigned inputs and determine the next node to be executed. Processing nodes are the set of $P_1$, $P_2$, ..., $P_q$, which work as action/processing functions. The start node determines the first node to be executed. The nodes transition begins from the start node, however there are no terminal nodes. After the start node is executed, the next node is determined according to the node's connections and judgment results.

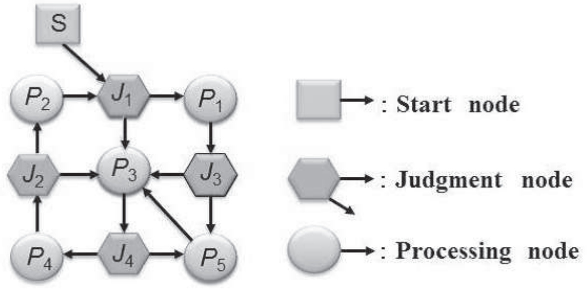


Figure 1. Basic structure of GNP

The gene structure of GNP (node $i$) is shown in Fig. 2. The set of these genes represents the genotype of GNP-individuals. $NT_i$ describes the node type, $NT_i = 0$ when node $i$ is the start node, $NT_i = 1$ when node $i$ is a judgment node and $NT_i = 2$ when node $i$ is a processing node. $ID_i$ is an identification number, for example, $NT_i = 1$ and $ID_i = 1$ mean node $i$ is $J_1$. $C_{i1}$, $C_{i2}$, ..., denote the nodes,

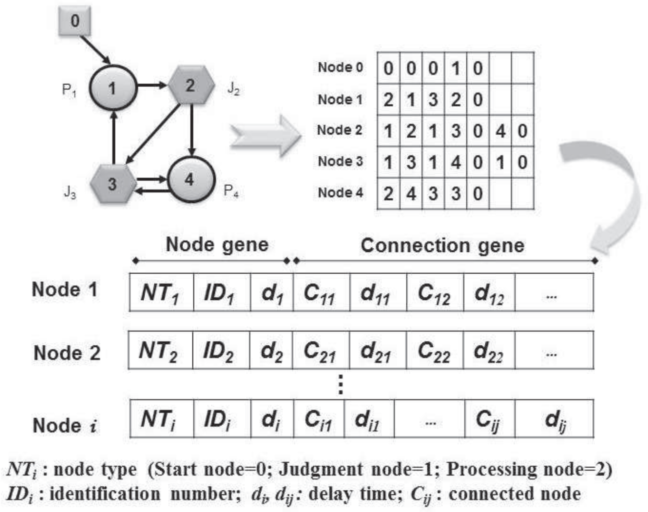

0
1
2
3
4
P1
J2
J3
P4
Node 0 0 0 0 1 0
Node 1 2 1 3 2 0
Node 2 1 2 1 3 0 4 0
Node 3 1 3 1 4 0 1 0
Node 4 2 4 3 3 0
Node gene
Connection gene
Node 1 NT1 ID1 d1 C11 d11 C12 d12 ...
Node 2 NT2 ID2 d2 C21 d21 C22 d22 ...
Node i NTi IDi di Ci1 di1 ... Cij dij
NTi : node type (Start node=0; Judgment node=1; Processing node=2)
IDi : identification number; di, dij: delay time; Cij : connected node

Figure 2. Gene structure of GNP (node $i$)

which are connected from node $i$ firstly, secondly, ..., and so on depending on the arguments of node $i$. $d_i$ and $d_{ij}$ are the delay time, which are the time required to execute the judgment or processing of node $i$ and the delay time of transition from node $i$ to node $j$, respectively.

In this paper, the execution time delay $d_i$ and the transition time delay $d_{ij}$ are not considered. All GNP-individuals in a population have the same number of nodes.

The characteristics of GNP are described as follows. (1) The judgment and processing nodes are repeatedly used in GNP, therefore the structure becomes compact and an efficient evolution of GNP is obtained. (2) Since the number of nodes is defined in advance, GNP can find the solutions of the problems without bloating, which can be sometimes found in Genetic Programming (GP). (3) Nodes that are not used at the current program execution will be used for future evolution. (4) GNP is able to cope with partially observable Markov processes. (5) The node transition in GNP individual is executed according to its node connections without any terminal nodes.

In the conventional GNP-based mining method, the attributes of the database correspond to the judgment nodes in GNP. Association rules are represented by the connections of nodes. Candidate rules are obtained by genetic operations. Rule extraction using GNP is done without identifying frequent itemsets used in Apriori-like methods [11]. Therefore, this method extracts important rules sufficient enough for user's purpose in a short time. The association rules extracted are stored in a pool through generations. The fundamental difference with other evolutionary methods is that GNP evolves in order to store new interesting rules in the pool, not to obtain the individual with the highest fitness value. GNP method has also advantages over other evolutionary methods such as Genetic Algorithms (GA) and Genetic Programming (GP). For GA-based methods [12], there are limitations in the number of association rules extracted because they are represented in individuals. In GP-base methods [13], an individual is usually represented by a tree with attribute values in the functions (e.g., logical, relational or mathematical operators) of the internal nodes. An individual's tree can grow in size and shape in a very dynamical way making it very difficult to understand for real applications.

## *A. GNP for rule extraction in a uncertain database*

In this section, a general association rule mining method for uncertain databases is proposed using GNP. Let $A_i$ be an attribute in an uncertain database and its value an existential probability. Each attribute $A_i$ is associated with $a_i$, which is a threshold value. One of the features of the proposed method is to evolve the threshold $a_i$ along with the evolution of GNP in order to obtain as many rules as possible [14]. The initial threshold $a_i$ is determined as follows: (1). The mean $\mu_i$ and standard deviation $\sigma_i$ of every attribute $A_i$ is calculated. (2). The initial threshold is selected randomly from the interval $[\mu_i - \alpha_i\sigma_i, \mu_i + \alpha_i\sigma_i]$, where $\alpha_i$ is a parameter that determines the size of the interval. Then, the initial threshold is evolved by mutation in every generation of GNP. Once the threshold $a_i$ is selected for all attributes, each value of the attribute $A_i$ is checked to verify whether it is greater than the threshold $a_i$ in the judgment nodes of the GNP individuals. The evolution of the thresholds is carried out by introducing an additional parameter that determines the mutation rate $t_r$. In this paper, the mutation rate $t_r$ is gradually adjusted as it is described in [14].

*1) Rule Representation:* Attributes and its values correspond to the functions of judgment nodes in GNP. Association rules are represented as the connections of nodes .

Fig. 3 shows a sample of the connection of nodes in GNP for probabilistic association rule mining. $P_1$ is a processing node and is a starting point of association rules. "$A_1 \geq a_1$", "$A_2 \geq a_2$", "$A_3 \geq a_3$" and "$A_4 \geq a_4$" in Fig. 3 denote the functions of judgment nodes. Association rules are represented by the connections of these nodes, for example, $(A_1 \geq a_1) \Rightarrow (A_2 \geq a_2)$, $(A_1 \geq a_1) \wedge (A_2 \geq a_2) \Rightarrow (A_3 \geq a_3)$, $(A_1 \geq a_1) \wedge (A_2 \geq a_2) \wedge (A_3 \geq a_3) \Rightarrow (A_4 \geq a_4)$ and $(A_1 \geq a_1) \wedge (A_2 \geq a_2) \Rightarrow (A_3 \geq a_3) \wedge (A_4 \geq a_4)$.

Judgment nodes in GNP are used to examine the attribute values of database tuples and processing nodes calculate the measurements of association rules. Judgment nodes determine the next node by a judgment result. Each judgment node has two connections Continue-side and Skip-side. The Continue-side of the judgment node is connected to another judgment node. Skip-side of the judgment node is connected to the next numbered processing node. If the attribute value is greater or equal to $a_i$, then move to the Continue-side. If the attribute value is less than $a_i$, then the transition goes for the Skip-side.

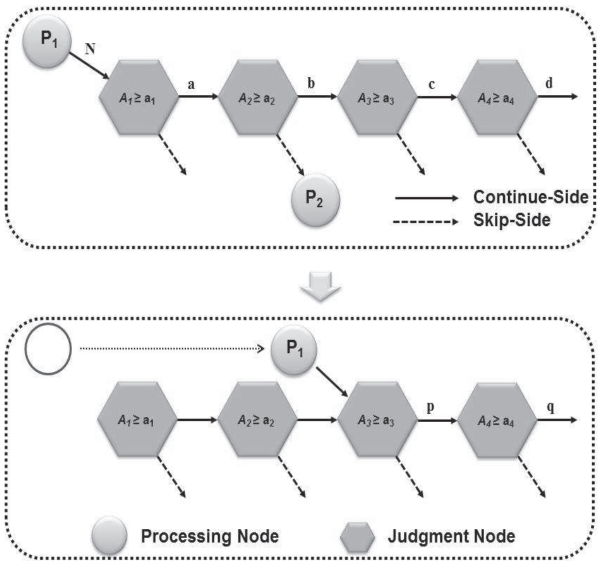


Figure 3. A connection of nodes in GNP for probabilistic association rule mining

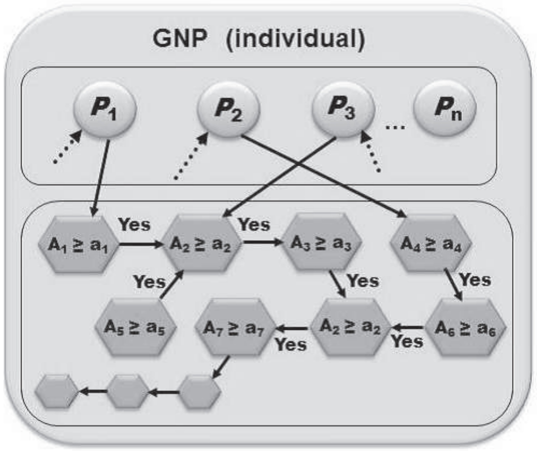


Figure 4. Basic structure of GNP for uncertain association rule mining

A basic structure of GNP-individual for association rule mining is shown in Fig. 4. In Fig. 4, the Skip-side of judgment nodes is abbreviated. Each processing node has an inherent numeric order ($P_1$, $P_2$, ..., $P_s$) and is connected to a judgment node. Start node connects to $P_1$. For each judgment node, the examinations of attribute values start and in case to move to the Continue-side continuously, the connection is obligatorily transfered to the next processing node using the Skip-node when the maximum number of attributes ($MaxLength$) in the rule is reached. When the examination of the attribute values of tuple $TID = 1$ from the starting point $P_s$ ends, then GNP examines the next tuple $TID = 2$ from $P_1$ likewise. Therefore, all tuples in the database are examined.

*2) Rule Measurements:* In GNP the number of tuples moving to the Continue-side are counted up and they are used for calculation of the measurements In upper side of Fig. 3, $a$, $b$, $c$ and $d$ are the number of tuples moving to the Continue-side at each judgment node when the attribute values are greater or equal to $a_1$, $a_2$, $a_3$ and $a_4$, respectively.

In the proposed method the significance of the associations are measured via the hyper-geometric distribution used in classical statistics. For example in lower side of Fig. 3 it is possible to calculate the number of tuples going to the Continue-Side of $A_3$ and $A_3 \wedge A_4$ if the connection of node $P_1$ is changed from node $A_1 \geq a_1$ to node $A_3 \geq a_3$. This procedure is repeated like a chain operation in each generation. The extracted important association rules are stored in a local pool all together through generations. When an important rule is extracted by GNP, the redundancy of the attributes is checked and it is also checked whether the important rule is new or not, that is, whether the rule is already in the local pool or not.

*3) Genetic Operations:* Changing an attribute to another one or adding some attributes in the rules would be considered as candidates of important rules. These rules can be obtained effectively by GNP genetic operations, because mutation and crossover will change the connections or contents of the nodes.

Three kinds of genetic operators are used for judgment nodes: GNP-crossover, GNP-mutation-1 (change the connections) and GNP-mutation-2 (change the function of nodes).

- GNP-Crossover: uniform crossover is used. Judgment nodes are selected as the crossover nodes with the probability of $P_c$. Two parents exchange the gene of the corresponding crossover nodes.
- GNP-Mutation-1: Mutation-1 operator affects one individual. The connection of the judgment nodes is changed randomly by mutation rate of $P_{m1}$.
- GNP-Mutation-2: Mutation-2 operator also affects one individual. This operator changes the functions of the judgment nodes by a given mutation rate $P_{m2}$.

On the other hand, all the connections of the processing nodes are changed randomly. At each generation, all GNP-individuals are replaced with the new ones by the following criteria: The GNP-individuals are ranked by their fitness values and the best one-third GNP-individuals are selected. After that, these GNP-individuals are reproduced three times for the next generation using the genetic operators described before.

If the probabilities of crossover ($P_c$) and mutation ($P_{m1}, P_{m2}$) are set at small values, then the same rules in the pool may be extracted repeatedly and GNP tends to converge prematurely at an early stage. If the probability of mutation is set at high values, then some genetic characteristics of the individuals might be lost. These parameter values are chosen experimentally avoiding these issues.

*4) Heterogeneity Level:* The heterogeneity level of rule $r$, $hl(r)$, is defined as follows:

$$hl(r) = \frac{\prod_{k}^{T}[na_k(r)/NA_k]}{T}; \; k = 1, 2, \ldots, T \tag{9}$$

where,

$na_k(r)$ is the number of attributes in rule $r$ (antecedent and consequent), which belong to database $k$.

$NA_k$ is the number of attributes of database $k$.

$T$ is the number of heterogeneous databases.

The heterogeneity level of rule $r$ measures the ratio of attributes that exist in the rules, which belong to one or another database. $hl(r) \geq \gamma$, where $\gamma$ is a threshold value for the heterogeneity level. It ensures that every rule contains at least one attribute per every heterogeneous database. $\gamma$ is defined experimentally and its value decreases when the number of databases increases.

*5) Fitness of GNP:* The number of processing nodes and judgment nodes in each GNP-individual is determined based on experimentation depending on the number of attributes processed. All GNP-individuals in a population have the same number of nodes. The connections of the nodes and the functions of the judgment nodes at an initial generation are determined randomly for each GNP-individual.

Fitness of GNP is defined by:

$$\begin{aligned} F = \sum_{r \in Q} \{ & \textit{hc(r)} + \alpha_{new}(r) + hl(r)(NA_A(r) - 1) + \\ & hl(r)(NA_C(r) - 1)\} \end{aligned} \tag{10}$$

The terms in Eq. (10) are as follows:

$Q$: set of suffixes of extracted important association rules satisfying (8)

*hc(r)*: value of *hyper-confidence(r)* of rule $r$

$\alpha_{new}(r)$: additional constant defined by

$$\alpha_{new}(r) = \begin{cases} \alpha_{new} & \text{(rule } r \text{ is new)} \\ 0 & \text{(rule } r \text{ has been already extracted)} \end{cases} \tag{11}$$

$hl(r)$: heterogeneity level of rule $r$.

$NA_A(r)$: the number of attributes in the antecedent of rule $r$.

$NA_C(r)$: the number of attributes in the consequent of rule $r$.

Constant $\alpha_{new}(r)$ in Eq. 10 is defined empirically based on the values of *hyper-confidence(r)*. Thus, $\alpha_{new}(r) = 0.3$.

$NA_A(r) \leq MaxLength$ and $NA_C(r) \leq MaxLength$. $MaxLength = 2T + 1$, where $T$ is the number of heterogeneous databases.

*hc(r)*, $NA_A(r)$ and $NA_C(r)$, and $\alpha_{new}(r)$ are concerned with the importance, complexity and novelty of rule $r$, respectively. The fitness represents the potential to extract new rules.

## V. Simulation Results

In order to test and validate the effectiveness of the proposed method, two real-time scientific databases from UCI ML Repository [15] and World Data System (WDS) [16] were taken to conduct the experiments, which are frequently used in data mining community. Both of them contains heterogeneous spatial-temporal data and they are suitable for mining general association rules. The first one ("A" dataset) is El Nino dataset and contains oceanographic and surface meteorological readings taken from a series of buoys positioned throughout the equatorial Pacific. The second one ("B" dataset) correspond to the weather information of the Pacific Ocean taken by sensors of World Ocean Circulation Experiment (WOCE).

### *A. Experiment Setting*

Both datasets are combined taken into account the date and each attribute is separated into two corresponding attributes according to their values. For instance, if $Latitude \leq 0$ correspond to the $Latitude = South$. In this experiment, data only from one year (1993) is considered. Thus, one large database is generated (36 attributes $\times$ 20609 records), then the data is normalized between the interval [0, 1] and these values are used as existential probabilities.

*1) Parameters of GNP:* The population size of GNP is 120. The number of processing nodes and judgment nodes in each GNP individual are 10 and 75, respectively. The maximum number of changing the connections of the processing nodes (*MaxLenght*) in each generation is $2(2) + 1 = 5$. The conditions of crossover and mutation are $P_c = 1/5$, $P_{m1} = 1/3$ and $P_{m2} = 1/5$. The termination condition is 100 generations. 10 runs were executed and the results are given as an average. These parameters were selected through experimentation. All algorithms were coded in Java language. Experiments were performed on a 3.2GHz Intel Xeon PC with 12G of main memory, running Windows 7 Ultimate 64bits.

Table II shows some examples of the rules extracted by GNP. The termination "A" or "B" of each attribute means the correspondence to its dataset. From Table II, the rules extracted by GNP are simple due to the small number of itemsets, which contribute to their understandability. Although the GNP-based data mining method extracts significant number of rules in a short period of time, it does not extract all the possible rules. Instead, it extracts rules with higher quality as it is shown in Table II.

Fig. 5 shows the number of extracted rules according to the number of generations using the complete database and $min_{hyper-conf} \geq 0.9$ . It can be seen that most of the association rules are extracted at earlier generations becoming stable at later generations, which improves the performance of the method.

Fig. 6 shows the number of extracted rules for different values of minimum hyper-confidence. Fig. 6 shows that when the minimum hyper-confidence increases, the number of association rules decreases because the conditions become more strict and fewer rules are able to satisfy them.

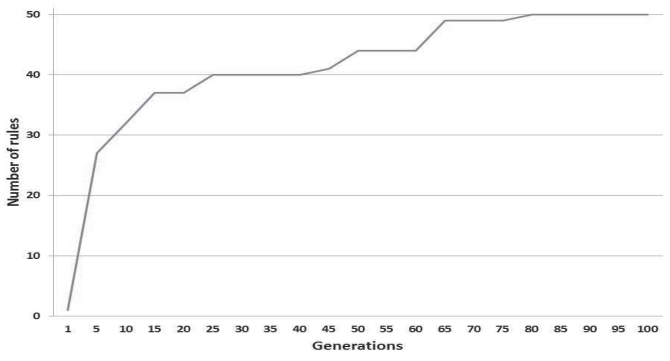


Figure 5. Number of extracted rules vs. number of generations

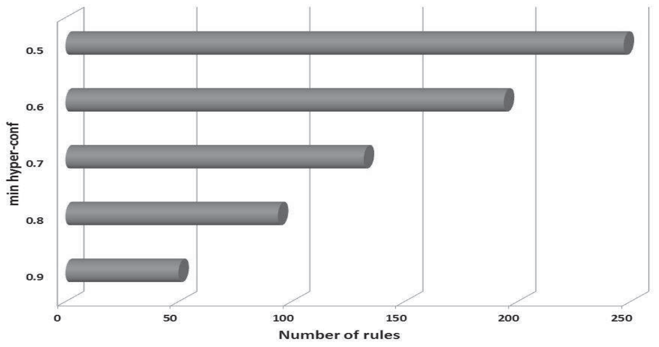


Figure 6. Number of extracted rules vs. min hyper-confidence

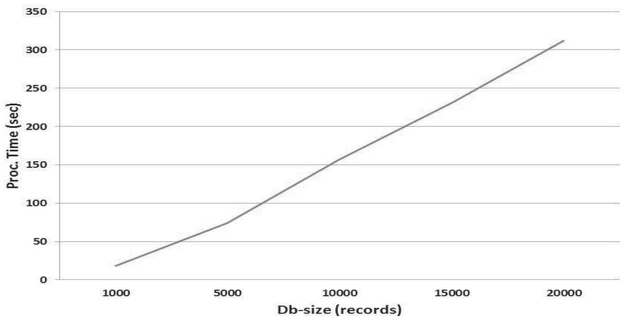


Figure 7. Processing Time vs. database size

Fig. 7 shows the processing time for extraction of association rules when the database size varies. Fig. 7 shows that the processing time increases linearly when the database size increases.

Fig. 8 shows the processing time for extraction of association for different values of hyper-confidence. Fig. 8 shows that the processing time does not vary significantly when hyper-confidence changes.

Table II
EXAMPLES OF RULES EXTRACTED BY GNP. GENERATIONS=100, $min_{hyper-conf} \geq 0.9$

| *Association Rules* | *Hyper-Conf.* |
|---|---|
| IF Sea_Surf_Temp = High_A ∧ Latitude = South_B, THEN Longitude = West_B ∧ Rel_Hum = High_B ∧ Precip = High_B | 1.0 |
| IF Longitude=West_A ∧ Zon_Winds=West_A ∧ Humidity=Low_A, THEN Precip = High_B ∧ Temp_Water = Low_B | 0.9871 |
| IF Temp_Air=High_A ∧ Speed=High_B, THEN Meridional_Winds= South_A ∧ Rel_Hum = High_B | 0.9962 |
| IF Pressure_Atm=High_B ∧ Temp_Air=Low_A ∧ Sea_Surf_Temp = High_A, THEN Longitude = West_B ∧ Temp_Water = High_B | 1.0 |
| IF Temp_Air=Low_B ∧ Zon_Winds=West_A ∧ Latitude=South_B, THEN Rel_Hum = High_B ∧ Precip = High_B | 1.0 |

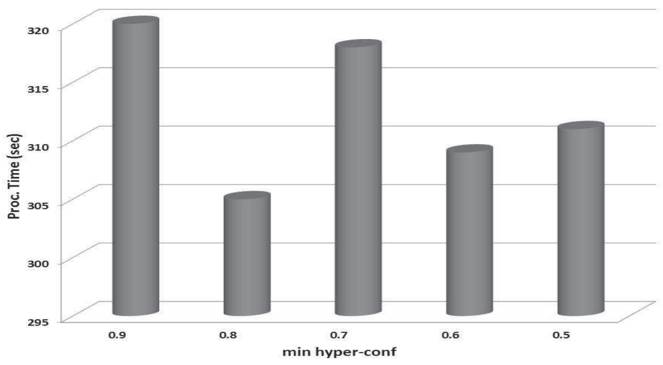


Figure 8. Processing Time vs. min hyper-confidence

## VI. CONCLUSION AND FUTURE WORK

A method for association rule mining from uncertain databases has been proposed using GNP. An uncertain database includes the existential probability of every item in a transaction. Traditional approaches count the frequency of the itemsets. The proposed method can extract directly important rules without calculating the frequency and the conditions of important association rules can be flexibly defined by users. The performance of the rule extraction has been evaluated using real data sets. The results shows that the proposed method has the potential to realize associations considering heterogeneous databases and may be applied for rule discovery from probabilistic databases in several other fields. For future work, the method will be extended to deal with large and heterogeneous scientific databases combined with web data.

## REFERENCES

[1] R. Cheng et al., "Probabilistic verifiers: evaluating constrained nearest-neighbor queries over uncertain data". In *Proc. of the IEEE ICDE 2008*, pp. 973-982, 2008.

[2] C.K. Chui, B. Kao, and E. Hung, "Mining Frequent Itemsets from Uncertain Data". In *Proc. of the 11th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 2007*, pp. 47-58, 2007.

[3] C.K.S. Leung and D. A. Brajczuk, "Efficient Algorithms for the Mining of Constrained Frequent Patterns from Uncertain Data", *SIGKDD Explorations*, Vol.11, Issue 2, pp. 123-130, 2009.

[4] K. Shimada, K. Hirasawa, and T. Furuzuki, "Genetic Network Programming with Acquisition Mechanisms of Association Rules", *Journal of Advanced Computational Intelligence and Intelligent Informatics*, Vol. 10, No. 1, pp. 102-111, 2006.

[5] E. Gonzales, K. Taboada, K. Shimada, S. Mabu, and K. Hirasawa, "Combination of Two Evolutionary Methods for Mining Association Rules in Large and Dense Databases", *Journal of Advanced Computational Intelligence and Intelligent Informatics*, Vol.13, No.5, pp. 561-572, 2009.

[6] M. Hahsler and K. Hornik. "New probabilistic interest measures for association rules" in *Journal of Intelligent Data Analysis*, Vol. 11, No. 5, pp.437-455, 2007.

[7] C. Zhang and S. Zhang, *Association Rule Mining: models and algorithms*, Springer, 2002.

[8] S. Mabu, K. Hirasawa, and J. Hu, "A Graph-Based Evolutionary Algorithm: Genetic Network Programming (GNP) and Its Extension Using Reinforcement Learning", Evolutionary Computation, *MIT Press* , Vol 15, No. 3, pp. 369-398, 2007.

[9] K. Hirasawa, T. Eguchi, J. Zhou, L. Yu, J. Hu, and S. Markon, "A Double-deck Elevator Group Supervisory Control System using Genetic Network Programming",*IEEE Trans. on System, Man and Cybernetics, Part C*, Vol.38, No.4, pp. 535-550, 2008.

[10] T. Eguchi, K. Hirasawa, J. Hu, and N. Ota, "A study of Evolutionary Multiagent Models Based on Symbiosis",*IEEE Trans. on System, Man and Cybernetics, Part B*, Vol.36, No.1, pp. 179-193, 2006.

[11] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules", in *Proc. of the 20th VLDB Conf.*, pp. 487-499, 1994.

[12] C.Z. Janikow, "A knowledge-intensive genetic algorithm for supervised learning", *Machine Learning 13*, pp. 189-228, 1993.

[13] C.C. Bojarczuk, H.S. Lopes, and A.A. Freitas, "Genetic programming for knowledge discovery in chest pain diagnosis", *IEEE Trans. on Engineering in Medicine and Biology Magazine*, Vol. 19, No.4, pp. 38-44, 2000.

[14] K. Taboada, E. Gonzales, K. Shimada, S. Mabu, K. Hirasawa, and J. Hu, "Association Rule Mining for Continuous Attributes using Genetic Network Programming", *IEEJ Transactions on Electrical and Electronic Engineering*, Vol. 3, No. 2, pp. 199-211 March 2008.

[15] Frank, A. Asuncion, A. (2010). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science. [Last Access: Jun 14th, 2011]

[16] Walden, B; WOCE Surface Meteorology Data, WOCEMET (2006): Continuous meteorological surface measurement during KNORR cruise 316N138_12. Woods Hole Oceanographic Institution, Physical Oceanography Department.

# A Multidimensional Data Modeling of the SEER Database from the USA National Cancer Institute

Heidy M. Marin-Castro, Jose Torres-Jimenez and Diana I. Escalona-Vargas
*Center of Research and Advanced Studies of the National Polytechnic Institute*
*Information Technology Laboratory*
*Scientific and Technological Park of Tamaulipas TECNOTAM, km 5.5*
*Cd. Victoria, Tamaulipas*
{*hmarin,jtj,descalona*}*@tamps.cinvestav.mx*

***Abstract*—Nowadays, one of the main challenges in computer science is to process the large amount of data available in diverse data sources, such as databases or files, in order to find useful information. For this purpose, it is required specialized tools that process raw data in a smart way to discover knowledge. In this paper, we present the design of a data warehouse and a tool called TDR (Tool Drill-Roll) that allow to discover knowledge from the database SEER (Surveillance, Epidemiology, and End Results) from the Cancer Institute in the United States of America, which has more than five million of records. The data warehouse is designed using a multidimensional approach and the TDR tool allows to exploit interesting information from SEER using drill-down and roll-up (two operators of On line Analytical Processing (OLAP)). The data warehouse can be seen at many levels of granularity. Our developed TDR tool allows knowing the statistics of the incidence, mortality and survival of patients with cancer along of years and extract useful information related to this disease that could be used to establish a relation between certain characteristics of patients that has an specific type of cancer. The knowledge discovered by our TDR tool could be of interest for government, health care institutes or research community for decision making. The main contribution of this paper is the discovery of new knowledge from the SEER database. The methodology used to design the data warehouse and the TDR tool could be applied to others domains with minimal changes.**

***Keywords-data warehouse; OLAP; drill-down; roll-up; cancer-database.***

## I. Introduction

Cancer is a disease fundamentally characterized by uncontrolled growth of malignant cells (known as carcinogenic or cancerous). The associated terminology with the issue of cancer is between benign and malignant tumors. A tumor or neoplasm is any abnormal growth of cell, which may be either benign or malignant. A benign tumor remains confined to its original location. It neither invades surrounding normal tissue nor spreads to distant body sites. In contrast, a malignant tumor is capable both of invading adjacent normal tissue and of spreading to others tissues and organs [1]. There are more than 100 different types of cancer. Most cancers are named for the organ or type of cell in which they start [2]. According to a report presented in 2010 of the U.S Population [3] approximately 789,620 men and 739,940 women will be diagnosed with cancer. Also, it is estimated that 569,490 men and women will die due to cancer in 2010 considering all cities in the United States of America.

In this paper, we present the construction of a data warehouse and a software tool called TDR to exploit interesting information related with records of patients with cancer in the United Stated of America using OLAP. We use information provided by the SEER (Surveillance, Epidemiology, and End Results) database which contain records of patients with different types of cancer from 1978 to 2007. Our design is based on a multidimensional approach for building the data warehouse. We created a six dimensional model that allows to uncover and grouping information not yet discovered with different levels of granularity. The multidimensional model was built in SQL language. To query the data warehouse we implemented a tool called TDR (Tool Drill-Roll) that incorporates the operators drill-down, roll-up and slice-dice. This operators acts on the defined hierarchies for exploring and visualizing the information at different levels of granularity. This ability makes our tool very useful to discover new knowledge from the SEER database that could be used by National Cancer Institute for decision making. TDR is a friendly tool that present information in tabular and graphical form making its use more easy to the end user. TDR has a browser of information which displays 3D graphs of the results of OLAP operators. Our design methodology for the data warehouse could be applied to different domains or databases with minimal changes.

Unlike well known tools for multidimensional analysis as Business Object [4] and Talend [5], the TDR tool presented in this work allows greater flexibility to add ad-hoc analysis on the data, and the possibility of calling proper functions at the time the data is being manipulated visually. In addition, our tool allows visualizing the discovered information displaying the recovered data in a 3D bar chart.

The rest of this document is organized as follows: next section presents an overview of the literature, describing general and related concepts and techniques. Section 3 describes the design process of the proposed data warehouse, explaining the dimensional approach and how hierarchies were defined. Section 4 describes the six-dimensional data cube designed in this work. Section 5 presents the architecture of the TDR tool. In Section 6 are presented the results achieved in this work and a discussion of them. Finally, Section 7 presents the conclusions of this work.

## II. Literature Review

The concept of data warehouse was defined by W. H Inmon [6] as a subject-oriented, integrated, time variant and non-volatile collection of data in support of managements decision making process. Another definition states that a data warehouse is a collection of large amount of data or a repository (collection of resources that can be accessed to retrieve information) of organizations electronically stored data, useful to generate reports [7]. The data warehouse paradigm provides architectures and tools for business executives to systematically organize, understand, and use their data to make strategic decisions. Data warehouse systems are valuable tools in today competitive, fast-evolving world. The data warehouse management is a relational database that contains the data that is collected from a server that is a data collection target. This data is used to generate the reports for the System Data collection sets, and can also be used to create custom reports.

The drill-down, roll-up and slice-dice operators are included in the design of a data warehouse manager. These operators allow the user to view the data at differing degrees of summarization. The first operator, drill-down builds a view to go from a general level of granularity to an specific level. The roll-up operator works in inverse sense to drill-down, roll-up builds a view to go from an specific level to a general level.

The data warehouse paradigm has been used in many business contexts. Recently, it is being used in the biomedical field such as biological science on clinical and genomic data [8]. The objective is to improve the capabilities of the On-line Analytical Processing (OLAP) to make effective medical decisions [9]. In the literature, there are few works related to a multidimensional approach for processing data in the domain of cancer disease. In [10] Wah*et al.* developed a complete multidimensional model design of a data warehousing for a Lymphoma cancer used the relevant information from the Internet. They proposed a six dimensional model. However, in this work the design of hierarchies is not considered as we do in our work.

## III. Data warehouse Architecture

### A. Database description

The study presented in this work uses the new version of SEER [2] database during the years from 1973 to 2007 with 5,178,804 records covering to eight cancer types (breast, colon and rectum, respiratory, female genital, urinary, male genital, lymphoma in all sites, leukemia, and digestive). We

| # | Item name | Positions | Length |
|---|---|---|---|
| 1 | Patient ID number | 01-08 | 8 |
| 2 | Registry ID | 09-18 | 10 |
| 3 | Marital Status at DX | 19-19 | 1 |
| 4 | Race/Ethnicity | 20-21 | 2 |
| 7 | Sex | 24-24 | 1 |
| 8 | Age at diagnosis | 25-27 | 3 |
| 12 | Month of diagnosis | 37-38 | 2 |
| 13 | Year of diagnosis | 39-42 | 4 |
| 20 | Grade | 58-58 | 1 |
| 21 | Diagnostic Confirmation | 59-59 | 1 |
| 87 | Age Recode $<$ 1 Year olds | 183-184 | 2 |
| 107 | SEER Summary Stage 1977 | 231-231 | 1 |
| 108 | SEER Summary Stage 2000 | 232-232 | 1 |
| 115 | Vital Status recode | 255-255 | 1 |

TABLE I
Attributes selected from multidimensional model.

organized the information of this database in dimensional tables to exploit information of patients with any type of cancer. Our study uses the attributes show in Table I of Computer Record Format (CRF) of SEER data fields.

### B. Facts and Dimensional Hierarchies

The number of patients with cancer has been increasing in recent years. It could be interesting to know the statistics of the incidence, mortality or survival along of years and extract useful information related of the disease and establish a relation between certain attributes of patients that has a type of cancer. Therefore, the most important characteristics of a data warehouse is the fact table that allows to quantify the number of incidences of cancer recorded. Our study consists in the quantification of medical cancer events that happened on certain dates, we can view how many patients are alive or dead with an specific type of cancer and with certain characteristics related to sex, group of age at diagnosis (child, young, adult, senior), marital status. The design of the data warehouse is based on a dimensional approach.

In this approach, the design is guided by the definition of hierarchies that allows to exploit data at different granularity levels. These hierarchies are designed from the tables in SEER database according to the hierarchical relation among the fields in such tables. In this work, these hierarchies are expressed as a direct acyclic graph. One example of these hierarchies is shown in Figure 1. In this figure, to the left there is the hierarchy of the fields in the corresponding dimension. Each node in the hierarchy is referred as a *category*. To the right of the figure is the same hierarchy but in terms of the possible *values* for each category. The possible values for category *group study type* are *microscopical and non-microscopical confirmed*. In the next level of this hierarchy is the category *name* with values *positive histology, positive cytology, etc.* Hierarchies, categories and values play an important role in the design and implementation of the proposed data warehouse in this work.

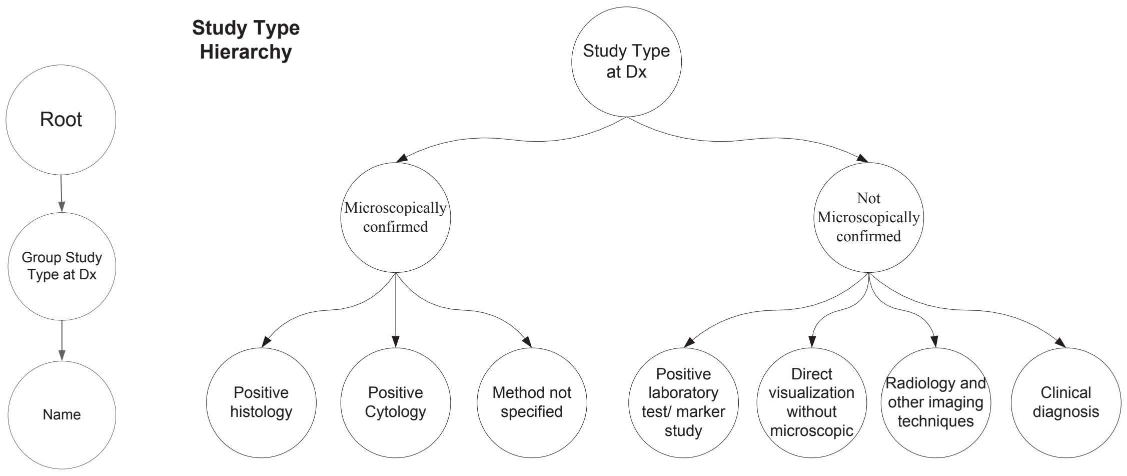

Study Type Hierarchy
Root
Group Study Type at Dx
Name
Study Type at Dx
Microscopically confirmed
Not Microscopically confirmed
Positive histology
Positive Cytology
Method not specified
Positive laboratory test/ marker study
Direct visualization without microscopic
Radiology and other imaging techniques
Clinical diagnosis

Figure 1. Hierarchy model from dimension *type of study*

All information in tables of SEER database are related by means of a table of facts called *medical_facts*. This table stores facts about patients such as if they are alive or dead with due to specific type of cancer and with certain characteristics related of sex, age group of diagnosis (child, young, adult, senior), marital status among others characteristics.

This characteristics are called *variables*, and they are variables of interest tracked by the TDR tool. The diagram of the table of facts allows to build reports that answer questions such as:

- The number of patients of black race diagnosed with breast cancer from 1990 to 2000.
- The number of women died due to mama cancer.
- The predominant marital status of patients having colon rectal cancer.

## IV. MULTIDIMENSIONAL DATA CUBE

We created OLAP cubes from a data model in order to obtain answers to questions such as the ones listed at the end of section III-B. An OLAP cube is not strictly a cuboid [11], it is the name given to the process of linking data from the different dimensions. In this work the data cube is created using four out of seven available dimensions.

Figure 2 shows a cuboid using three dimensions. To the right of this figure it is shown the four ways of data can be explored from this cube by fixing categories and values at each dimension. In this work we use a data cube defined by four dimensions ($X$, $Y$, $Z$, $W$) instead three. This data cube is commonly known as a *teserac*. For dimension four ($Z$), we fix one from all possible values for this dimension. Then the associated cuboid given by ($X$, $Y$, $Z$) is processed by OLAP operators in order to get the four possible view of data.

### A. Refinement OLAP operators or querying manipulation

The main advantage when building a data warehouse is the use of operators for facilitating the aggregation (consolidation) or the disaggregation (division) of data. The operator *roll-up* (aggregation) allows to eliminate a grouping criterion. The *drill-down* operator (disaggregation) allows to introduce a new grouping criterion. These operators allow querying the data source that can not be done using traditional operations like selections, projections, concatenations or groupings.

Drill-down and roll-up OLAP operators make extensive processing of hierarchies described in Section III-B. When a dimension is selected in the *teserac*, all records of patients in the data source are considered for querying (no grouping restrictions are given). The hierarchies associated to the selected dimensions are reset to the root node. From this point, an specialized query can be done by selecting one of the reached categories from the root node in one of the hierarchies. The result is the selection of an *slice*, a subset of data such as the ones shown in Figure 2. Several selected slices form a *dice*. For example, when a category is selected, all values for this category define several slices from the data cube, that is, they define a *dice*.

If we want to descend in a hierarchy, we need to fix a value for the current category, which means to reduce the data set only to that records meeting the condition *category = value*. Now, for the new data subset we can repeat the same process in case more descendants exist in the hierarchy. Each time we perform a drill down operation a new condition given by the expression *category=value* must

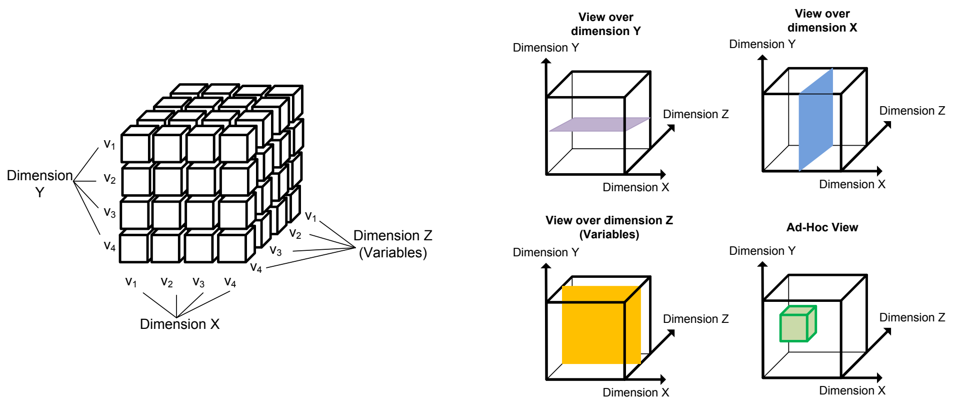


Figure 2. Cuboid and four ways to explore data

be satisfied, together with the previous conditions. So, the drill-down operator defines a *path* that specifies a set of conditions that group data for specialized querying. Drill down operator can be applied over any dimension at any time. The data grouped by the drill down operator is dis grouped by the roll up operator. This means that we need to go back one condition in the current built path. Doing this causes the immediately previous data set be recovered and used for querying.

Both drill down and roll up operators allow to obtain a measure of facts, with restrictions given by the conditions imposed over dimensions. The number of possible data subsets that can be explored is given by the numbers of paths that can be formed from the hierarchies, from the root node to each leaf in the graph.

## V. A GRAPHICAL USER INTERFACE FOR THE MULTIDIMENSIONAL MODEL

We built a software tool called TDR to get data out for users by applying the drill down and roll operators up over the SEER database. However, this tool allows operate over any other database with minimal changes.

All information about the dimensions, facts and hierarchies is stored in a database, so, this tool is scalable to support new dimensions or hierarchies. In the reports we focus on a single variable from the table of facts, like “survival” or “mortality”. This is also possible for the user to select the tracked variable.

Users can decide how to explore data in the SEER database by selecting any of the available dimensions presented in Section 2.3. For each dimension, the user can go down across the hierarchy associated to that dimension for a more specialized query. If required, the user can go back the hierarchy or select a different dimension at any time. Always a dimension is selected, the associated hierarchy is recovered and the navigation starts form the root node. Drill down is the main operator for doing specialized queries to the database. At the beginning, suppose that one of the selected dimensions is “age”. The hierarchy associated to this dimension indicates that the first criterion is “year of diagnosis”. Then, the tool presents all possible values in the facts for this given current criterion. If the user wants to descend in this hierarchy, he performs a drill down operator by selecting one of the possibles values of this current criterion. The hierarchy associated to dimension “age” indicates that the next criterion is “month of diagnosis”, so now we see in the output all possible values for this now current criterion. The user can proceed in the same way until no more descendants exist in the hierarchy. A roll up operation on a hierarchy causes the grouping done by the previous drill down operation be dissolved. Slice and dice operations are implemented by selecting specific values for current categories selected in dimension $Z$ and dimension four. Dimension $Z$ is assigned to the tracked variables in the fact table. The fourth dimension is traversed by its corresponding hierarchy. In this case all possibles values for the current criterion in the hierarchy is not shown in the table of results due to this table will be very large in the browser. Instead, just a single value is considered at a time, which ca be selected from a combo box.

The software tool uses a plotting library for 3D graphics that can be downloaded from [12]. All tables needed for operating like dimensions and hierarchies are stored in a database in MySQL. The tool uses the dimensions: *age-dimension, cancer-dimension, marital-status-dimension, race-ethnicity-dimension, sex-dimension, study-dimen-sion*.

Our developed tool uses four dimensions that can be selected among the ones previously registered. From these dimensions the user can determine the search criterion for answering a given question for example *“How many women died because of a not solid cancer type in the year 1990?”*. In this case, the user must select the dimensions *sex, cancer-type and age-diagnosis*. If user wanted specialize the above query, he only needs to do a drill down operation on any of the available dimensions, for example, specializing the above query to the month September in the same year of

1990.

## VI. Results

In this study, we used the TDR tool to obtain new and relevant knowledge about cases of cancer in patients that live or die in the United States between the year 1973 to 2007. In the first analysis we obtained the rate of mortality and survival of women with mama cancer. The results shown 104677 widowed women died due to mama cancer and 83448 survived. For this case, the rate of mortality and survival is shown in the equations 1, 2 respectively. For the case of single women, the rate of mortality of single women is 31.67% and the rate of survival is 68.33%. These results show that the mortality in widowed women is higher that in single women. However, the survival rate for married women is 71%, being the highest. It seem to be that the marital status has an important effect asociated with the mortality and the risk of mortality of widowed women over married women.

$$mortality\_rate = \frac{104677 * 100}{188,125} = 55.64 \qquad (1)$$

$$survival\_rate = \frac{83448 * 100}{188,125} = 44.36 \qquad (2)$$

In US, rectal colon cancer is the fourth cancer with more incidence. It is more common in old people (around 50s) and risk increasing with age. In our study we observe that this cancer type affects in equal number to women and men. This keeps true for patients that survive to this cancer.

Another interesting uncovered fact is that most of the realized studies for detection of cancer were confirmed microscopically. It seems to be that patients rely on this technique. We observe that the number of died patients due to mama cancer diagnosed microscopically is higher than the number of survived patients. In this case, the number of patients died due to not solid cancer (leukemia) diagnosed using any kind of study (not a microscopically study) is around 10287. The number of cases confirmed microscopically were 272913 and the case confirmed not microscopically were 9673.

## VII. Conclusion

This work presented the development of a data warehouse based on the SEER database to find relevant and useful knowledge in data. We developed a tool called TDR that makes extensively use of operators drill-down, roll-up, and slice-dice to explore data at different granularity levels. We found interesting information on the SEER database such as the mortality and survival rate of patients for different types of cancer. The development of data warehouse required of some considerations. First, data cleaning played the most critical role in the data warehouse development. It is a time consuming task that must be handled thoroughly to avoid no useful data. Second, the structure of the data warehouse is designed in a flexible way allowing further extensions and adaptation to other domains databases. Finally, the dimensional approach used for the construction of the data warehouse allowed defined the granularity of queries at different levels.

The results presented in this work can be extended or new knowledge could be explored using our TDR tool. The end user has fully freedom to form questions and getting the answers using our tool. For example, *"How many women were diagnosed with breast cancer in 1980?"*.

## Acknowledgments

This research was partially funded by the following projects: CONACyT 58554-Cálculo de Covering Arrays, 51623-Fondo Mixto CONACyT y Gobierno del Estado de Tamaulipas.

## References

[1] G. M. Cooper, *The Cancer Book: A Guide to Understanding the Causes, Prevention, and Treatment of Cancer*. London,UK: Printed in the United States of America, 1993.

[2] National Cancer Institute, "Surveillance Eoidemiology and End Results," http://seer.cancer.gov.

[3] National Cancer Institute SEER, "NCI Fact Sheets," http://seer.cancer.gov/cancertopics/factsheet.

[4] SAP, Company, "Soluciones SAP para el almacenamiento de datos ," http://www.sap.com/mexico/solutions/sapbusinessobjects/data-warehousing/index.epx.

[5] www.datawarehouse4u.info, "Talend Open Studio ," http://datawarehouse4u.info/Talend-Open-Studio.html.

[6] W. H. Inmon and R. D. Hackathorn, *Using the Data Warehouse*. New York: Wiley–QED, 1994.

[7] P. Vassiliadis, C. Quix, Y. Vassiliou, and M. Jarke, "Data Warehouse Process Management," *Information Systems*, vol. 26, pp. 205–236, 2001.

[8] L. Wang, A. Zhang, and M. Ramanathan, "Biostar models of clinical and genomic data for biomedical data warehouse design," *Int. J. Bioinformatics Res. Appl.*, vol. 1, pp. 63–80, April 2005. [Online]. Available: http://portal.acm.org/citation.cfm?id=1356096.1356101

[9] S. Palaniappan and C. S. Ling, "Clinical Decision Support Using OLAP With Data Mining," *International Journal of Computer Science and Network Security*, vol. 8, no. 9, 2008.

[10] T. Y. Wah and O. S. Sim, "Development of a data warehouse for lymphoma cancer diagnosis and treatment decision support," *WSEAS Trans. Info. Sci. and App.*, vol. 6, pp. 530–543, March 2009. [Online]. Available: http://portal.acm.org/citation.cfm?id=1553642.1553661

[11] R. Kimball and J. Caserta, *The Data WarehouseETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data e*. Canada: Wiley, 2004.

[12] L. Tautenhahn, "SVG-VML-3D 1.3 Graphic Library," available at http://www.lutanho.net/svgvml3d/index.html.

# Constructing a Synthetic Longitudinal Health Dataset for Data Mining

Shima Ghassem Pour*, Anthony Maeder* and Louisa Jorm†
*School of Computing, Engineering and Mathematics
University of Western Sydney, Campbelltown, Australia
Email: A.maeder@uws.edu.au
†School of Medicine, University of Western Sydney, Campbelltown, Australia
Email: L.jorm@uws.edu.au

***Abstract*—The traditional approach to epidemiological research is to analyse data in an explicit statistical fashion, attempting to answer a question or test a hypothesis. However, increasing experience in the application of data mining and exploratory data analysis methods suggests that valuable information can be obtained from large datasets using these less constrained approaches. Available data mining techniques, such as clustering, have mainly been applied to cross-sectional point-in-time data. However, health datasets often include repeated observations for individuals and so researchers are interested in following their health trajectories. This requires methods for analysis of multiple-points-over-time or longitudinal data. Here, we describe an approach to construct a synthetic longitudinal version of a major population health dataset in which clusters merge and split over time, to investigate the utility of clustering for discovering time sequence based patterns.**

***Index Terms*—cluster analysis; synthetic data**

## I. Introduction

The rapid growth of digital information and the related capability of collecting and storing huge amounts of data offer two new opportunities. On one hand, these data represent the potential for discovering useful information and knowledge which has not been uncovered before because of the sheer volume of data which is now available. On the other hand, the limited ability of conventional processing of large amounts of data to discover useful information and new knowledge can be overcome by applying a new generation of mathematical techniques to extract patterns from the data [1]. Data availability is increasing exponentially, while the human level of processing ability is almost constant. As this gap increases, there is a growing necessity for knowledge discovery in databases and data mining [1], [2].

Data mining refers to discovering insight from data, which is reliable statistically and not known as priori [3]. This data must be available, relevant, adequate, and clean. Also, the data mining problem must be well-defined, yet cannot be solved simply by query and reporting tools, and should be guided by a data mining process model [4]. Overall, data mining is the process of discovering and interpreting previously unknown patterns in databases [5]. There are many methods of data mining used for different purposes and goals.

We can use data mining for discovering structure in large volumes of data. Most data mining techniques aim to analyse only one single set of data elements which are static in time. But, often in health data, we have many sets of observations at different times and we want to follow their trajectories over time. Longitudinal data is essentiality data observed sequentially over time [6]. It may be collected from a designed experiment or an observed study, where the outcome variables are related to a sequence of event or responses recorded at certain time point during study period. In particular, large scale longitudinal data is frequently encountered in research in biology, medicine, and public health.

The organization of the paper is as follows: in Section II we describe the problem of analysing a longitudinal dataset, in Section III we present the proposed method for creating a synthetic dataset, and in Section IV we discuss experimental result using the synthetic dataset.

## II. Problem

Our overall research addresses the area of cluster analysis and attempts to reformulate cluster analysis for application to longitudinal health data problems, with a resulting improvement in conceptual simplicity and statistical capability. In clustering methodology, one is generally given a sample of N objects, each of which is measured on M variables. From this information alone, one must devise a classification scheme for grouping the objects into K classes. The number of classes and the characteristics of those classes are to be determined to solve some underlying problem.

If all the objects in a given class were identical to one another, the problem would be simple. However, in the usual situation the objects in each class differ on most or all of the measures [2]. Most cluster analysis procedures try to measure the similarity between any two objects, and then try to group the objects so as to maximize within-class similarity. Unfortunately, the appropriate measure of similarity is a subject of some controversy. It would be desirable to derive a cluster analysis system without arbitrary assumptions about similarity [7], [8].

Since the objects within a class differ from one another, it is reasonable to assume the existence of probability distributions of characteristics for a population belonging to this class. Lazarsfeld and Henry introduced Latent Structure Analysis

[9], which is closely related to the mixture analysis problem. In "Latent Class Analysis" [10] (which is a special case of Latent Structure Analysis) the associations among variables are explained by assuming the population is a mixture of potential or 'latent'classes, within each of which the variables are independently distributed.

Latent Class Analysis (LCA) methods classify subjects into one of K unobserved classes based on the observed data, where K is a constant and known parameter. These latent or potential classes are then refined based upon their statistical relationships with the observed variables. LCA is a probabilistic clustering approach: although each object is assumed to belong to one cluster, there is uncertainty about an object's membership of a cluster. This type of approach offers some advantages in dealing with noisy data or data with complex relationships between variables, although as an iterative method there is always some chance that it will fail to converge.

An important related issue in how to make use of clustering methods such as LCA is to decide on the ideal number of clusters. Researchers use a range of information criteria to determine the number of clusters which best fit the data, such as Akaike Information Criterion (AIC) [11] and Bayesian Information Criterion (BIC) [12]. BIC is known as a good indicator for making a decision about the number of clusters for large datasets [13]–[15]. If L is the maximized value of the likelihood function for the estimated model, n represents the sample size and P indicates the number of parameters for the estimated model, BIC is represented by [12], [16]:

$$BIC = -2logL + Plog(n) \tag{1}$$

Longitudinal datasets are created when the same characteristics are measured repeatedly over time and can take a long time to build up if the time steps are numerous and of long duration. Complex health datasets contain different types of variables such as demographics, lifestyle and health measures, which can be difficult to analyse without long time sequences.

To understand the workability and stability of data mining methods on such data, there is a need to have an alternative way to create a longitudinal dataset. In this paper, we describe an approach to construct a longitudinal dataset synthetically. Use of a synthetic dataset would not be as accurate as using existing datasets, however a synthetic dataset would share some of properties of interest that one would expect to see in actual longitudinal datasets.

## III. Method

The 45 and Up study [17] is a population-based cohort study with participants aged over 45 resident in the Australian State of New South Wales, randomly selected from the Australian Medicare database [18]. The study is organized by the Sax Institute and State Government of New South Wales as core partners and also has some partners from public health and health service research centres and universities across New South Wales [19], [20]. Recruitment into the 45 and Up study began in February 2006: by July 2008 the first 103,042 participants had joined the study [19] and since then it has more than doubled. The 45 and Up study combines socio-economic and demographic factors, as well as health and lifestyle. It is especially useful for studying effects of slow and chaotic emergence of diseases such as cancers.

The 45 and Up database serves as a good foundation dataset for testing data mining methods because of the richness of these different kinds of variables contained in it. Also it will in the future serve as a benchmark time based dataset, recording trajectories of individual health histories over a long period of time (potentially from age 45 until death). At the present time it is entering the second cycle of data collection providing a time step of approximately 2 years for most participants who have remained part of the study. Consideration of how longitudinal data mining approaches could be developed for use on this database in the longer term would be a useful contribution to extracting further value from it. For example, we are investigating a subset of the data concerning prostate cancer prevalence and risk factors and requiring longitudinal analysis.

We will construct a synthetic time version of data to test the ability of LCA to discover time sequence based patterns in this data. The synthetic time data will be formed by identifying a group of variables which could reasonably lead to splitting and merging of clusters over successive time steps. A simple example in the area of lifestyle related data is changes in body mass index (BMI), and exercise or Physical Activity (PA). These two variables were chosen as they are deemed most influential of those available.

Data from the first (current) time step will be clustered according to these variables, and then a revised set of data for the next successive time steps will be calculated using the above identified variables to produce converging and diverging sets of values. Addition of some controlled noise or perturbation of these calculated values according to an anticipated variation pattern could also be incorporated. As the imposed structure will be known in order to carry out these operations, this synthetic data can serve as a kind of gold standard for validating the clustering methods we will be applying. In order to construct a synthetic time version of 45 and Up dataset, we are interested to group the data based on variations of BMI and PA over time.
To create a synthetic dataset we follow these 5 steps:

*Step 1*: The first step is to choose a sample dataset from the actual dataset (in this case the 45 and Up dataset). We randomly select 23,000 elements to create a reasonably large dataset. For this sample dataset we choose the two variables, computed body mass index (BMI) and how many times per week subjects do some physical activity (PA). We have a BMI range from 16-44 and PA range from 0-100 times per week. To simplify the situation we divide the BMI variable into 4 different categories: the first category is people with BMI<18.5 (underweight), the second category people with BMI between 18.5 to 25 (normal), the third group people with BMI between 25 to 30 (overweight) and finally the last category with BMI>30 (obese). We divide the PA variable into 10 categories, people who have PA<10 times a week belong to the first category, PA between 10-20 in the second category and so forth. As a conclusion at this step we have 23,000 cases with 2 different variables: BMI variable with four categories and PA variable with 10 categories.

*Step 2*: In the second step, we apply LCA to cluster our data based on BMI and PA, and the result shows that two clusters fit our baseline data best (under the BIC criterion). People who are underweight, normal weight and overweight belong to Cluster1_B (52.6%) while Cluster2_B consists mostly of obese people (47.4%).

*Step 3*: Next, we change some of the values for the above two variables to vary how elements from the two clusters that we have from the first time step will appear in the second time step. To achieve this, we decide to choose people who were in Cluster1_B and randomly choose about one third of them to remain as before (3,797 cases), and divide half of the rest of them in two groups. For the majority (2,249) of them we change the BMI to overweight and the rest of them (1,519) we move to the normal BMI group. The other half of the BMI range is unchanged, however the PA is changed for this group: we randomly choose about one third of them (1,265) and increase the amount PA and decrease PA for the rest of them (2,531).

*Step 4*: The next step is to change Cluster2_B of the first time step. We split Cluster2_B in two groups equally. The first group remains without any change while the other is divided in two new groups. The first new group includes 4,212 cases which are 1,054 cases of decreased PA and 3,160 increased PA. In the second new group which includes 2,106 cases we change the BMI in 1,264 cases to the underweight category and the rest to the obese category.

*Step 5*: The final step is to apply LCA to the second time step to see how two clusters in the first time step actually split under the same clustering method as was applied to the first time step data.

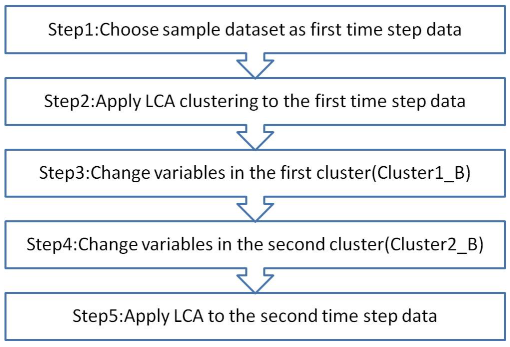

Step1:Choose sample dataset as first time step data
Step2:Apply LCA clustering to the first time step data
Step3:Change variables in the first cluster(Cluster1_B)
Step4:Change variables in the second cluster(Cluster2_B)
Step5:Apply LCA to the second time step data

Fig. 1. Five step creation of synthetic dataset

This sequence of steps is summarized in Fig. 1. At each step in the synthesizing we also have the opportunity to add some controlled noise to each group, but in the interests of simplicity we have not included that aspect in the work reported here. We can also repeat the above steps to construct more synthetic data for another time step. Our synthetic dataset will give us a good understanding of splitting and merging clusters over time as some of groups are designed differently but with some similarity to force the cluster to merge in the next time step. Additionally, when future time step data from 45 and Up becomes available, the methods we developed can be applied to that and a comparison can be made as to the similarity with our simulated results. For this aspect of the work, a minimum of 3 time steps will be needed in the dataset.

## IV. Results

For this research project we used the LatentGOLD software package to perform LCA clustering due to the attractive range of additional features, and its widespread user base as noted from our readings in the literature. LatentGold is a commercial product [21] that uses SPSS to automatically provide a variety of output, graphics and diagnostics to help the user interpret the resulting clusters and to refine their analysis.

Applying Latent Class Analysis to the above synthetic dataset shows us that at the baseline our data fitted well into a two cluster model, with the majority of people who are obese but with different amounts of physical activity belonging to Cluster1_B and the rest of people in Cluster2_B. Table 1 shows how the clustering using LCA can be compared over four different models with respectively K = 1 to 4. The parameter BIC is an estimate of the overall tightness or stability of the clusters, while Npar is the number of free parameters to be estimated, LL is the log-likelihood and L2 is the square likelihood value for the estimated model. As the BIC values show, the model with 2 clusters is optimal for explaining the data. Fig. 2 shows the range of baseline data elements at the first time step assigned to Cluster1_B (circles) which includes mostly people who are underweight, normal and overweight, and Cluster2_B (crosses) which is exclusively people who are obese.

TABLE I
Four different models for baseline data

| K | LL | BIC(LL) | Npar | L2 |
|---|---|---|---|---|
| 1 | -56492.24 | 113105.58 | 12 | 498.05 |
| 2 | -56261.70 | 112775.70 | 25 | 36.97 |
| 3 | -56250.99 | 112885.47 | 38 | 15.55 |
| 4 | -56243.54 | 113001.78 | 51 | 0.66 |

Table 2 shows that for the synthetic data for the second time step, there are 4 clusters which best fit the data according to the BIC measure, Fig. 3 shows how these clusters cover the data space, with some elements of Cluster1_B from the first time step now moved in to the expanded Cluster2_B which becomes Cluster4_S in the second time step. The other elements from Cluster1_B in the first time step are split across 3 new clusters which roughly map to underweight (Cluster1_S), normal (Cluster2_S) and overweight (Cluster3_S) people. It is noted from Fig. 2 that the split of the first time step Cluster1_B is not entirely dependent on the BMI variable.

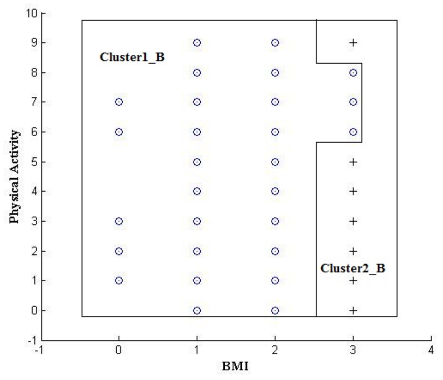


Fig. 2. Two cluster explanation of baseline data

The underweight cluster (Cluster1_S) extends to include some normal people with low exercise values, and the overweight cluster (Cluster3_S) extends to include some normal and underweight people with high exercise values (144 cases in all).

TABLE II
SIX DIFFERENT MODELS FOR SYNTHETIC DATA

| K | LL | BIC(LL) | Npar | L2 |
|---|---|---|---|---|
| 1 | -57679.5 | 115470.0 | 11 | 35529.41 |
| 2 | -43022.1 | 86276.10 | 23 | 6214.50 |
| 3 | -40551.0 | 81454.98 | 35 | 1272.36 |
| 4 | -39916.0 | 80306.04 | 47 | 2.40 |
| 5 | -39915.7 | 80426.31 | 59 | 1.66 |
| 6 | -39915.7 | 80547.34 | 71 | 1.67 |

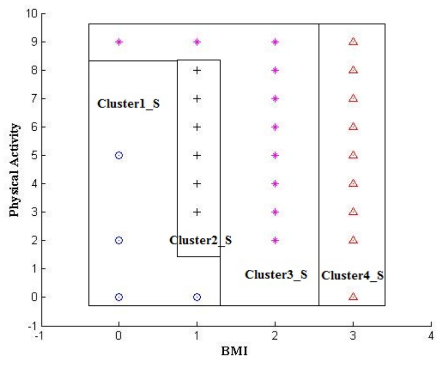


Fig. 3. Four cluster explanation of synthetic data

## V. Conclusion

Analysis of longitudinal data is becoming popular as researchers are more interested to describe how people change during time. However, collecting an appropriate longitudinal dataset takes a long time to build up. On the other hand, to validate the workability and stability of data mining methods, there is need to have such a dataset available. The work here was done to explain how we can construct a longitudinal dataset synthetically. Such synthetic datasets will give us better understanding of methods to extract more useful and valuable information from our data, to explain how characteristics of people's health will change during time. In order to create a synthetic dataset, we followed a systematic approach which was based on the use of LCA to understand structural characteristics of the variables. Based on the initial clustering, we decided which variables should be changed and how this change should be applied, in order to simulate a desired change in the characteristics. This approach was tested using the 45 and Up dataset. These results shown here confirm that the desired effect was observed and the approach did not adversely affect clustering stability for the chosen example. We are now proceeding with creation of further large scale synthetic datasets for prostate cancer pattern analysis for multiple variables over multiple time steps.

## References

[1] J. Han, M. Kamber, and J. Pei, *Data mining: concepts and techniques*. Morgan Kaufmann, 2011.

[2] H. Mannila, “Data mining: machine learning, statistics, and databases,” in *Scientific and Statistical Database Systems, Proceedings*. IEEE, 1996, pp. 2–9.

[3] C. Elkan, “The foundations of cost-sensitive learning,” in *International Joint Conference on Artificial Intelligence*, vol. 17, no. 1, 2001, pp. 973–978.

[4] N. Lavrač, B. Kavšek, P. Flach, and L. Todorovski, “Subgroup discovery with cn2-sd,” *The Journal of Machine Learning Research*, vol. 5, pp. 153–188, 2004.

[5] D. Hand, H. Mannila, and P. Smyth, *Principles of data mining*. The MIT Press, 2001.

[6] D. Hand, “Statistics and data mining: intersecting disciplines,” *ACM SIGKDD Explorations Newsletter*, vol. 1, no. 1, pp. 16–19, 1999.

[7] C. Fraley and A. Raftery, “How many clusters? which clustering method? answers via model-based cluster analysis,” *The Computer Journal*, vol. 41, no. 8, pp. 578–588, 1998.

[8] ——, “Model-based clustering, discriminant analysis, and density estimation,” *Journal of the American Statistical Association*, vol. 97, no. 458, pp. 611–631, 2002.

[9] P. Lazarsfeld and N. Henry, *Latent structure analysis*. Houghton, Mifflin, 1968.

[10] L. Collins and S. Lanza, *Latent class and latent transition analysis: With applications in the social, behavioral, and health sciences*. John Wiley & Sons Inc, 2009, vol. 718.

[11] H. Akaike, “Factor analysis and aic,” *Psychometrika*, vol. 52, no. 3, pp. 317–332, 1987.

[12] G. Schwarz, “Estimating the dimension of a model,” *The Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[13] L. Collins, P. Fidler, S. Wugalter, and J. Long, “Goodness-of-fit testing for latent class models,” *Multivariate Behavioral Research*, vol. 28, no. 3, pp. 375–389, 1993.

[14] J. Hagenaars and A. McCutcheon, *Applied latent class analysis*. Cambridge Univ Pr, 2002.

[15] J. Magidson and J. Vermunt, “Latent class models,” *The Sage handbook of quantitative methodology for the social sciences*, pp. 175–198, 2004.

[16] K. Nylund, T. Asparouhov, and B. Muthén, “Deciding on the number of classes in latent class analysis and growth mixture modeling: A monte carlo simulation study,” *Structural Equation Modeling*, vol. 14, no. 4, pp. 535–569, 2007.

[17] “Study overview,” May 2010. [Online]. Available: http://www.45andup.org.au/[accessedOct2011]

[18] N. Mealing, E. Banks, L. Jorm, D. Steel, M. Clements, and K. Rogers, “Investigation of relative risk estimates from studies of the same population with contrasting response rates and designs,” *BMC Medical Research Methodology*, vol. 10, no. 1, pp. 10–26, 2010.

[19] E. Banks, L. Jorm, S. Lujic, and K. Rogers, “Health, ageing and private health insurance: baseline results from the 45 and up study cohort,” *Australia and New Zealand health policy*, vol. 6, no. 1, pp. 6–16, 2009.

[20] E. Banks, S. Redman, L. Jorm, B. Armstrong, A. Bauman, J. Beard, V. Beral, J. Byles, S. Corbett, R. Cumming *et al.*, “Cohort profile: the 45 and up study.” *International Journal of Epidemiology*, vol. 37, no. 5, pp. 941–947, 2008.

[21] “Latent gold,” May 2011. [Online]. Available: http://www.statisticalinnovations.com/products/latentgold.html[accessedMarch2011]

# Business Lead Generation for Online Real Estate Services: A Case Study

Md. Abdur Rahman, Xinghui Zhao, Maria Gabriella Mosquera, Qigang Gao and Vlado Keselj
Faculty Of Computer Science
Dalhousie University
Halifax, Nova Scotia, Canada
{mrahman, xzhao, mosquera, qggao, vlado}@cs.dal.ca

Hai Wang
Sobey School of Business
Saint Mary's University
Halifax, Nova Scotia, Canada
hwang@smu.ca

***Abstract*—Business leads generation is a crucial and challenging task for online business to attract customers and improve their services. This paper presents a case study of an online real estate service company which provides potential home buyers useful neighborhood information, and accordingly offers them business leads to real estate companies. The company's current business lead generation is based on a "brute-force" method which requires tedious manual efforts. We developed an automatic business lead generation system which includes data integration as well as data mining tasks of classification and association rule mining. We demonstrated through experiments that this system can empower online real estate service companies to quickly and effectively generate targeted business leads.**

***Keywords- Business lead generation; Data modeling; Data integration; Data mining; Business Intelligence.***

## I. INTRODUCTION

Over the past three decades, there has been a significant change in which information was collected, disseminated, and used. Form the customers' point of view, the information overload that they experience on a daily basis can hinder their decisions when selecting products and services to purchase. From the business organizations' point of view, the vast array of products and services being offered has made data mining a critical competitive advantage. Data mining techniques allow business organizations to improve their services and attract more customers. This paper presents a case study of business lead generation for an on-line real estate service company through data integration and data mining. A business lead, also referred to as a lead for simplicity, is defined as a potential sales contact, i.e., a potential customer who expresses an interest in the company's products or services [2].

In the real estate research literature, a recent study shows that almost 80% of home buyers start a home search online [8]. There are many online real estate service companies that provide potential home buyers useful neighborhood information, and accordingly provide them business leads to real estate companies. We conducted a study on a Canadian based online real estate service company. Currently, the lead products being offered by the company are based on a "brute-force" method which requires tedious manual efforts. There is no automatic lead generation system available on the market that could empower the company to quickly and effectively generate leads.

This paper proposes a business lead generation system which is capable of identifying effective landing pages on the websites of online real estate service companies. A landing page, also referred to as a lead capture page, is a web page to be displayed in response to clicking on an online advertisement which is posted on the Internet, either on the website of an online real estate service company or on a third-party website. For an online real estate service company, these landing pages may generate leads as potential home buyers may leave contact information on them. Data mining techniques such as classification and association rule mining are employed for lead generation.

The proposed automatic lead generation system can help the online real estate service company to increase revenue by selling more leads to real estate companies. It will also equip the real estate service company with a platform to reduce the cost of generating leads. Fig. 1 shows the business model of a typical online real estate service company:

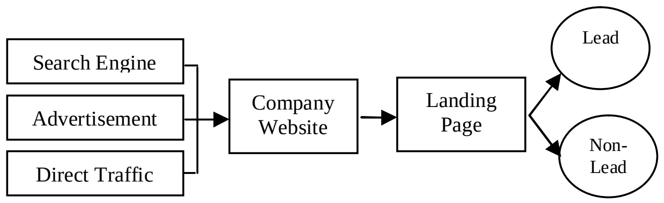


Figure 1. Business model for a typical online real estate service company.

Most of the users visit the online real estate service company's site through search engines or third-party websites where the company's advertisements are posted. While visiting the company's website, the users are exposed to a landing page. If a user visits the landing page and leaves his/her contact information then, the user is considered as a lead. On the other hand, if a user leaves the site at any point during their visit, the user is considered as non-lead.

To attract users, landing pages are designed carefully by considering the text size, style, color and contain various business offers and rewards. Generally, more than one landing pages are used to attract various user groups. The biggest challenge is how to provide the right landing page to the right user so that the user can become a lead.

The rest of the paper is organized as follows. Section 2 shows related work. Section 3 shows the system architecture of the lead generation engine; Section 4 describes the data model design and integration based on activity flow of users who use online services for home buying/selling purpose; Section 5 shows the data mining tasks, results, as well as interpretation of the results. Finally, Section 6 concludes the paper.

## II. RELATED WORK

Currently, there are no lead generation tools available on the market. One of the main difficulties is due to large volume of data, which are distributed into several sources. This large volume of distributed data needs to be integrated into a single source before conducting any analysis. The data also needs to be transformed into a suitable format for data mining tasks to discover leads and non-leads.

In the research literature, automatic lead generation has been studied in a few cases for typical B2B and B2C types of e-commerce. [7] studied automatic sales lead generation based on online customers' survey, and [5][11] studied lead generation based on online customers' purchasing patterns. An online real estate service company differs from typical B2B or B2C e-commerce companies because it acts like a broker between the real estate companies and potential home buyers. To our best knowledge, there is no previous research on business lead generation for online real estate service companies.

## III. SYSTEM ARCHITECTURE

For lead generation two major data mining tasks are involved in this system: Association rules mining and Classification. Association rule generation is a method for discovering interesting relationships among various item sets in a dataset and classification is the task of discovering a model first and applying it to predicate a new data object into one of the several classes of a predetermined target.

The system architecture is illustrated in Fig. 2. At first, the proposed system will gather all available data from the application. The primary data sources are the server log files, which include web server access logs and application server logs. Some additional data are also essential for both data preparation and pattern discovery, including the site files and meta-data, operational databases, application templates, and domain knowledge.

After gathering data from all the required sources, the web data will go through a pre-processing phase to clean the data by removing irrelevant or redundant entries from the data sets. Following the pre-processing phase, the data will be formatted appropriately according to the application business model. After formatting, the log data will be converted into a form suitable for data-mining task. The transformation will be accomplished according to the transaction model for that particular task (i.e., classification or association rule mining). Finally, after discovering the hidden statistics patterns from the data, the new discovered knowledge will be evaluated and filtered to only keep those which are statistically sound and novel to the business.

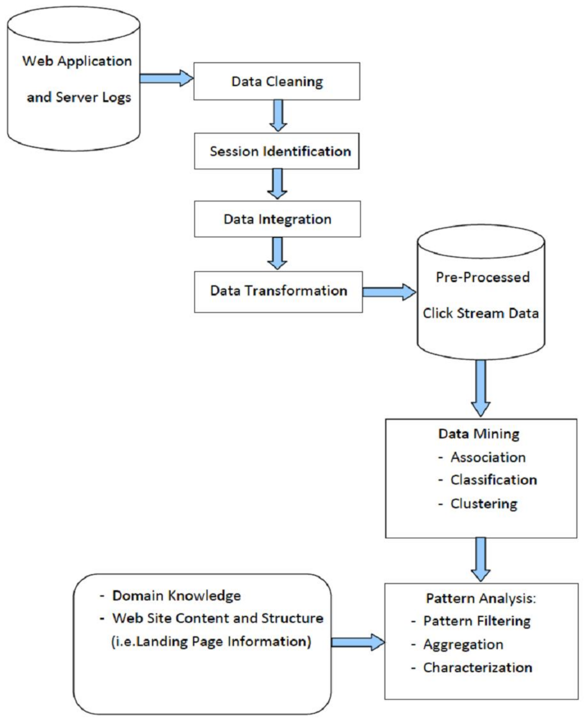


Figure 2. System Architecture.

## IV. DATA MODEL DESIGN AND DATA INTEGRATION

Data model design is a critical step for data warehousing and data mining. It mainly involves four steps: identifying relevant datasets, feature selection, data integration and data transformation.

The current data repository setup, for our study of the online real estate service company, contains 3 datasets, which hold information for the period of January to June 2011. The "Dataset 1" contains the data pertaining to what users' actions were while browsing the website. This dataset also contains information about landing page visits where users' preferences about home buying/selling were recorded. The "Dataset 2" and "Dataset 3" contain search type information such as whether a user searched for schools, banks, and other retailers while searching for a home or neighborhood, as well as which ads were served. The "Dataset 2" also contains more detailed information about each search (e.g., search strings, referrer pages, user agent or browser used). A summary of these datasets is shown in Table 1.

TABLE I. SUMMARY OF DATASETS

| Name | Size | #Cases | #Fields |
|---|---|---|---|
| Dataset 1 | 668.3 MB | 2,197,320 | 10 |

| | | | |
|---|---|---|---|
| Dataset 2 | 81.8 MB | 1,015,332 | 7 |
| Dataset 3 | 109.1MB | 349,850 | 18 |

"Dataset 1" is dynamically generated in such a way that a new data instance is created for every different action performed by an user and the properties of a particular instance depends on the action of the user (i.e., submitting form, searching etc). In other words, the dataset contains many data instances for one user if the user performs more than one action. Therefore, this dataset is a semi-structured table in which one field may contain different types of information, and for one user there might be various numbers of rows associated with them. For data mining purposes, we normalized this table, combined all information related to one user into one single row, and then selected a fixed number of representative fields, which contain useful information for data mining.

Based on our analysis on user activity flows, we have divided the users into two groups and created a separate data model for each group. The two groups are "General User Data Model" and "Landing Page User Data Model". Landing page users are important and treated specially since they are more likely to become leads and they have more information recorded in the dataset.

## A. *General User Data Model*

The general user data model covers all users including both the leads users and the non-leads users. The purpose of this data model is to form a single integrated dataset, which contains all activities of all users, i.e., including who did not reach any landing pages, as well as the activities of the users prior to reaching any landing pages. Table 2 shows the features for the general user data model.

Furthermore, to facilitate data mining tasks such as classification, we artificially created a new field "landing page related" to indicate whether the user eventually reached landing page and whether the user eventually became a lead. This new field is for the purpose of differentiating landing-page-users from non-landing-page-users, and differentiating leads from non-leads. Specifically, we use an integer value for the "landing page related" field. We assign the value "-1" for users who did not reach the landing page, "0" for non-leads users who reached the landing page, and "1" for leads.

TABLE II. GENERAL USER DATA MODEL.

| Category | # | Features | Description |
|---|---|---|---|
| Before reaching website | 1 | referrer to website | referrer page to website |
| | 2 | ad served | ad served on referrer page |
| | 3 | ad name | ad name |
| Activity on website | 4 | page view on website | Website's page(s) being visited |
| | 5 | search address | address being searched |
| User info | 6 | IP location | IP location of user |
| | 7 | landing page visit | whether the user visit landing page or not |
| Additional | 8 | error | whether there is any error while performing any action. |
| | 9 | time stamp | activity time stamp |

## B. *Landing Page User Data Model*

For all users who reached landing page (leads or non-leads), we designed a separate data model, in order to accommodate the extra landing page related activities, as shown in Table 3.

TABLE III. LANDING PAGE USER DATA MODEL

| Category | # | Features | Description |
|---|---|---|---|
| Before reaching website | 1 | referrer to website | referrer page to website |
| | 2 | ad served | ad served on referrer page |
| | 3 | ad name | ad name |
| On website, before landing page | 4 | page view on website | website page(s) being visited |
| | 5 | search address | address being searched (Prov.) |
| | 6 | referrer to LP | referrer page to landing page |
| Landing page info (LP) | 7 | LP selection method | method of selecting the LP |
| | 8 | LP version | version of LP |
| User info | 9 | Total 8 features in this category. | User's preferences about buying/ selling home. |
| Leads or not | 17 | submit | whether user submitted form to be contacted |
| Additional | 18 | error | whether there is an error |
| | 19 | time stamp | activity time stamp |

We selected 19 features for the landing page user data model, which belong to 6 categories: activities before reaching website, activities on website (before landing page), landing page information, user information, lead or not, and additional information.

As shown in Table 3, most of the features are landing page related features, which do not exist for users who did not reach a landing page. Having separate data models can facilitate data mining tasks such as classification and association rule mining.

## C. *Data Integration Design and Implementation*

We integrated the data from different sources into a single dataset for data mining. For each of the data models, we need to generate the integrated dataset. The data integration consists of two steps:

1. Integrate features from different tables into one table.
2. Integrate multiple rows which are related to one user into one row.

For Step 1, because the three original datasets (i.e., Dataset 1, Dataset 2 and Dataset 3) share a common id, we used that common id to join all tables. In Step 2, users' sessions were identified and all the activities during a single session were integrated into a single instance. The detailed algorithm is as follows and the corresponding flow chart is depicted in Fig. 3.

**Algorithm:**
**Step 1:** All Landing Page visits are identified.
**Step 2:** For each Landing Page visit, attributes are checked to identify whether this user has activities on the website's search page or not. If the user has search activities on the website's search page, all search related attributes are extracted for this user, if there is no search activity performed by the user, the program move to next step.
**Step 3:** User session is identified from "Dataset 1".
**Step 4:** All required attributes, for data model, are extracted from each row within the session.
**Step 5:** All extracted attributes, from a session, are stored in a new table within a single row where each single row hold data for only one session.
**At the end,** if all sessions are identified and data extraction is completed then the integrated file is generated for data mining purposes.

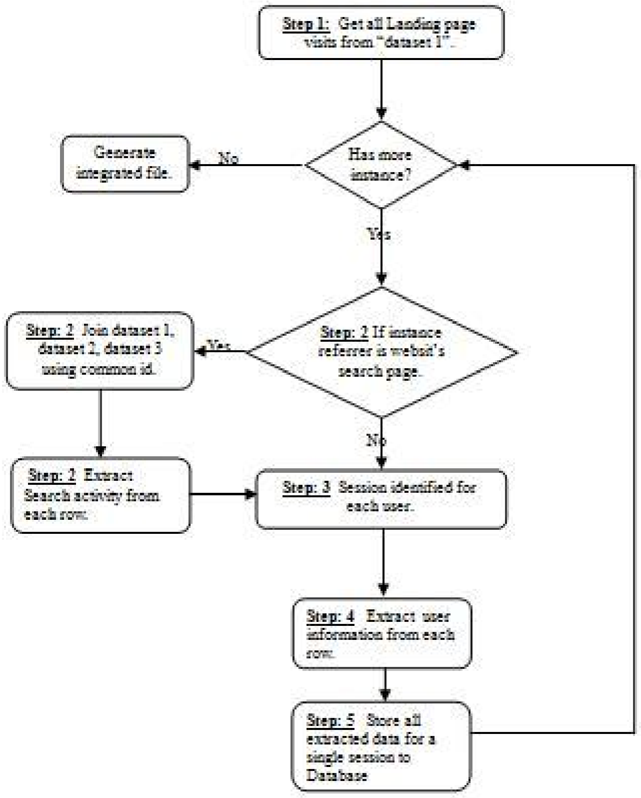

Step 1: Get all Landing page visits from "dataset 1".
Has more instance?
No
Generate integrated file.
Yes
Step: 2 If instance referrer is websit's search page.
Yes
Step: 2 Join dataset 1, dataset 2, dataset 3 using common id.
Step: 2 Extract Search activity from each row.
No
Step: 3 Session identified for each user.
Step: 4 Extract user information from each row.
Step: 5 Store all extracted data for a single session to Database

Figure 3. Flow Chart of Data Integration

## V. DATA MINING AND EXPERIMENTS

We carried out data mining tasks of classification and association rule mining on the integrated datasets to identify effective landing pages and generate rules for leads.

### A. *Classification*

The goal of classification is to find out whether a user profile belongs to lead user group, or non-lead user group. The output of our classification rule mining algorithm is a decision tree, in which the user models are represented as sets of rules for lead users and non-lead users. These rules can then be used to analyze the behavior patterns of the two groups of users. In this research, the classification algorithm C4.5 is applied to generate decision trees [9][6].

Fig. 4 shows an example of the decision tree generated by the C4.5 algorithm. We use the landing page version to show how decision tree can help to select the right landing page for right user. For example, landing page "lp11" has 100% lead turn out for the users who visit the site using advertisement named "X" and contain home buying price range for 200-400k. On the other hand, 90% of users who visit landing page "lp11", using advertisement for "Z" with the same price range, become non-leads. Clearly, for this category of users, "lp11" is not working well in terms of generating lead.

Using the decision tree, the company can analyze which landing page is working well for a specific category of users instead of choosing landing page randomly. In addition, the landing page can also be preselected based on the performance rate in converting users to leads.

```
pricerange = 200-400k
|       pageview = lp5 : Non Lead (0.96)
|       pageview = lp6 : Lead   (0.80)
|       pageview = lp7 : Lead   (0.50)
|       pageview = lp11
|       |       ad_name = Y : Non Lead (1.00)
|       |       ad_name = Z : Non Lead   (0.90)
|       |       ad_name = X : Lead   (1.00)
|       |       ad_name = M : Non Lead   (1.00)
|       pageview = lp10
|       |       intent = buy-sell: Non Lead  (0.75)
|       |       intent = buy : Lead   (0.70)
|       |       intent = sell : Non Lead  (1.00)
|       pageview = lp0 : Lead   (0.61)
pricerange = 400-600k
|       iplocation = ON : Non Lead  (1.00)
|       iplocation = NS : Lead   (1.00)
pricerange = 600-800k : Lead   (1.00)
pricerange = 800-1000k : Non Lead  (0.66)
pricerange = 1000k
|       error = no_error : Non Lead  (1.00)
|       error = yes : Lead   (1.00)
```

Figure 4. Decision tree from classification

### B. *Association rule mining*

In data mining, association rule generation [10] is used for discovering interesting relationships among various item sets of a dataset. We used association rule mining to find useful relationships among the user activities, landing pages and advertisements information of the company.

Association rules can be expressed in the form X => Y where X and Y are two disjoint subsets of all available items. We used the Apriori algorithm [1] to generate association rules. The algorithm can generate large quantities of patterns from the data, however most of which are of no interest. To remove the uninteresting rules, interestingness measurements techniques are applied to the result sets. Two commonly used measures are support rate (*supp*) and confidence rate (*conf*), which are defined as the following:

• Support rate of X => Y: The percentage of transactions in dataset containing both X and Y. Support(X,Y) = $P(X \cup Y)$.

• Confidence rate of X => Y: The percentage of transactions containing X and also containing Y. Confidence(X,Y) = $P(Y|X)$.

However, the combination of support and confidence measurements are insufficient at filtering out the uninteresting association rules [4]. As a result, another measurement technique, "Lift" [3], is applied. Lift value is equivalent to the ratio of the confidence of the rule and the expected confidence of the rule. The formula for measuring the lift can be expressed as : Lift (X, Y) = $P(Y) / P(X)*P(Y)$.

The value of Lift can be from 0 to infinity and can be interpret in the following way:

- If the value is greater than 1 then the rule (X=>Y) occurs more often than expected, which means that the occurrence of the rule body(Y) has a positive impact on the occurrence of the rule head(X).
- If the value is smaller than 1 then the rule (X=>Y) occurs less often together than expected, which means that the occurrence of the rule body(Y) has a negative impact on the occurrence of the rule head(X).
- If the value is near 1 then the rule (X=>Y) occurs almost as often together as expected, which means that the occurrence of the rule body(Y) has almost no effect on the occurrence of the rule head(X).

In our experiment, we set the support threshold to be 0.10 and Lift threshold to be greater than 1 to filter all positively correlated rules. To compare the effectiveness of the rules in terms of interestingness measures, both Lift and Confidence values are calculated for each rule. Some examples of the association rules are shown in Fig. 5.

The first rule shows that the users from "ON" (ip location, ON means Ontario, which is a province in Canada), who visited the site through advertisement "X" are more attracted to landing page version 10 and website page "page3". Therefore, the landing page "lp10" proved to be a high lead conversion page for users with iplocation "ON" and who visit the site through advertisement "X".

1. iplocation="ON", ad_name= "X" ==> site_pageview="page3", landing_page= "lp10" [conf:(0.73) , lift:(3.88)]
2. ad_name="Y" ==> landing_page="lp10", site_pageview = "page3" [conf:(0.69), lift:(3.67)]
3. pricerange="200-400k", intent="buy" ==> landing_page ="lp10", site_pageview ="page3" [conf:(0.67) , lift:(3.56)]
4. ad_name= "Y" ==> pricerange="200k" [conf:(0.5), lift:(1.6)]
5. landing_page ="lp11" ==> pricerange= "200k" [conf:(0.6), lift:(1.92)]
6. lp_referrer= "search engine x", choice= "NA" ==> landing_page= "lp0", address= "Other", iplocation= "AB" [conf:(0.75) , lift:(3.33)]
7. landing_page = "lp0" ==> lp_referrer="search engine x", site_pageview= "page1" [conf:(0.5), lift:(3.33)]
8. landing_page = "lp6", reward_choice= "yes" ==> ad_name= "Z" [conf:(1) , lift:(4)]
9. ad_name="X" ==> landing_page="lp10", iplocation="ON" [conf:(0.56) , lift:(3)]

Figure 5. Association rules generated from landing page data set

Rule number 7 shows that users whose referrer is "search engine x" and visit website's "page 3" are more attracted to landing page "lp0".

This knowledge can be used by the online real estate service company to increase the sales lead from different interest of users.

## VI. Conclusion and Future Work

This paper presented a framework for business lead generation in the area of online real estate services via a business case study. The framework includes data model design, data integration from multiple sources, data mining, and lead pattern evaluation. The results of classification rules can tell which groups of users are more likely to become leads, so that the corresponding pages/ads on the website may be emphasized for attracting those users. The association rules provide useful information for web designers, in that the togetherness of certain association patterns are important and the correlations among attributes in terms of desired target analysis. The knowledge base of lead generation should be updated periodically in order to best support business practices. In future research, the web page content data may be combined with web click stream data together form improving the lead generation results. This system may also be extended to build a real time lead generation system, which can select the most effective landing page by matching the user's request and the existing knowledge base. In addition, more rigorous experiments can be conducted to compare different data mining approaches for lead generation, including the validity testing and performance evaluation of the data mining methods.

## REFERENCES

[1] R. Agrawal, T. Imielienski, and A. Swami. “Mining association rules between sets of items in large databases”. Proceeding of the SIGMOD Conference, pp. 207–216. ACM Press, New York, NY, USA 1993.

[2] S. J. Bigelow. "Lead", August 2007. <http://searchitchannel.techtarget.com/definition/lead> [accessed October 1, 2011].

[3] D. S. Coppock. "Data Modelling and Mining: Why Lift?", June 2002. <http://www.information-management.com/news/5329-1.html> [accessed October 4, 2011].

[4] J. Han and M. Kamber, “Data mining Concepts and Technique”, Morgan Kaufmann Publishers, San Francisco, CA, 2006.

[5] D. V. D. Poel and W. Buckinx. “Predicting online-purchasing behaviour”, European Journal of Operational Research, Volume 166, Issue 2, 16 October 2005, pp. 557-575.

[6] J. R. Quinlan. “C4.5: Programs for Machine Learning. Morgan”, Kaufmann Publishers, 1993.

[7] G. Ramakrishnan, S. Joshi, S. Negi, R. Krishnapuram, and S. Balakrishnan. "Automatic Sales Lead Generation from Web Data”, Proceedings of the ICDE Conference, pp. 101-101, 2006.

[8] E. Weintraub. "Writing Purchase Offers in a Buyer's Market", <http://homebuying.about.com/od/offersnegotiations/tp/BuyersMKTOffers.htm> [accessed October 2, 2011].

[9] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. McLachlan, A. Ng, B. Liu, P. Yu, Z. Zhou, M. Steinbach, D. Hand and D. Steinberg, "Top 10 algorithms in data mining", Knowledge and Information Systems, Volume: 14, Issue: 1, 1 January 2008. pp. 1-37.

[10] Y. Woon, W. Ng and E. Lim, Association Rule Mining, Infomration System, 77-82, (2009).

[11] X. Zhang, W. Gong, and Y. Kawamura. “Customer behavior pattern discovering with web mining”, Lecture Notes in Computer Science, vol. 3007, pp. 844–853. Springer-Verlag Berlin, Heidelberg, 2004.

# On Parallel Evaluation of SPARQL Queries

Zbyněk Falt, David Bednárek, Miroslav Čermák and Filip Zavoral
*Department of Software Engineering*
*Charles University in Prague, Czech Republic*
*{falt, bednarek, cermak, zavoral}@ksi.mff.cuni.cz*

***Abstract*—The Semantic Web databases are growing bigger and much effort is given to introduce new approaches to querying them effectively. Most of the current approaches are based on query optimization or their parallel or distributed run. However, they do not fully benefit from potential of the modern, multicore computers in parallel processing. Although the parallel relational database systems have been well examined, parallel query computing in Semantic Web databases has not been extensively studied. This paper follows our previous research in parallelization of evaluation of SPARQL queries by outlining the possibility of further parallelization of query operators themselves. As a result, we developed a parallel SPARQL query execution engine built over Bobox framework. We show that intraoperation parallelism yields to better performance.**

***Keywords*-parallel, SPARQL, Bobox**

## I. Introduction

As prevalence of semantic data on the web is getting bigger, the Semantic Web databases are growing in size. There are two main approaches to storing and accessing these data efficiently: using traditional relational means or using sematic tools, such as different RDF triplestores [1] and SPARQL [2] language. Since semantic tools are still in development, a lot of effort is given to research of effective storing of RDF data and their querying [3]. One way of improving performance is the use of modern, multicore CPUs in parallel processing. Nowadays, there are several database engines which are capable of evaluating SPARQL queries, such as SESAME [4], OWLIM [5] or RDF-3X [6], which is currently considered to be one of the fastest single node RDF-store [7]. These stores support parallel computation of multiple queries; however, they do not use the potential of parallel computation of query itself, in contrary to the work [8]. It introduces implementation of RDF-store based on RDF-3X with parallelized join operators, but not full set of operators.

In our previous work, we chose the semantic approach and presented parallel SPARQL engine [9], that provides streamed parallel query execution on basic operation level. While the implementation of operations is based on sequential algorithms, there is considerable research on parallel versions, such as parallel joins [10], [11], [12]. Moreover, new architectures have been explored to improve performance. Gedik et al. [13] adapts join operation for the Cell processors. He et al. [14] uses GPUs for the processing of joins. Both papers try to exploit parallel nature of these architectures and show performance benefits over optimized CPU-based counterparts.

The SPARQL algebra is similar to the relational algebra; however, there are several important differences, such as absence of NULL values. As a result of these differences, the application of relational knowledge into semantic is not straightforward and the algorithms have to be adapted so it is possible to use them.

In this paper, we explore the effects of further decomposition of basic operations in our SPARQL engine in order to increase parallelism during the query evaluation. In pilot implementation, we focused on following basic operations: nested loops join, index scan and filter. This enables evaluation of some queries completely in parallel and reaching better scalability.

The rest of the text is organized as follows: In Section II, we present background information about Bobox framework we used. It also contains more detailed description of some of its parts, since it is important for understanding of the rest of the paper. Section III introduces two main concepts of parallelization of operation in Bobox. In Section IV, we describe parallelization of selected operations. The results of our experiments are introduced in Section V. Section VI concludes the paper and presents our future work.

## II. Background

The platform we used to build our execution engine is Bobox [15]. Bobox is a parallel framework, which was designed to support development of data-intensive parallel computations. The main idea behind Bobox is connecting a large number of relatively simple computational components into a nonlinear pipeline. This pipeline is then executed in parallel, but the interface used by the computational components is designed in such way that they do not need to be concerned with the parallel execution issues such as scheduling, synchronization and race conditions.

This system may be easily used for database query evaluation. Since Bobox supports only a custom low-level interface for the definition of the structure of the pipeline, a separate query compiler and optimizer has to be created for required query language. Firstly, we developed SPARQL compiler for Bobox [16]. Its main task is to perform lexical, syntactic and semantic validation of the given query, to perform static optimizations and to build optimal query execution plan using heuristics to reduce search space and statistics collected over stored data.

Traditionally (e.g. in relational databases), query execution plans have the form of directed rooted trees in which the edges indicate the flow of the data and all of them are directed to the root. The nodes of the tree are the basic operations used by the evaluation engine, such as full table scan, indexed access, merge join, filter etc. This plan corresponds to Bobox architecture, since the tree is a special case of the nonlinear pipeline supported by the system.

### A. Evaluation of the plan in the Bobox

After the Bobox receives a plan for evaluation, it replaces the operations in the plan by the boxes, which are elementary execution units in Bobox, and connects them according to edges in the execution plan.

Each box is able to receive data, process them and send resulting data out to the boxes which are connected to its output. Data are processed by small parts, which, in Bobox, are called envelopes. The envelope contains a list of columns which contain the data. The columns may have arbitrary data types, but all columns have always the same number of data elements. Therefore, from other point of view, the envelopes are also lists of rows where each element of the row may have arbitrary data type. Additionally, each envelope may contain also some scalar data (for example integer or boolean value) which may be used for additional communication between the boxes.

The processing of envelopes inside the box is always single-threaded, because boxes are not allowed to create another threads. There are two reasons for this limitation:

- The development of the boxes is easier, since a developer does not have to take any parallelization into account.
- If the boxes were allowed to create its own threads, the total number of threads in the system would be out of control and may easily exceed the number of physical threads in the computer by several order of magnitude. This may cause significant slowdown of the execution.

The evaluation of the execution plan works as follows: when the box receives an envelope, the envelope is stored into its (limited) input buffer and the box is scheduled. When the scheduler, which is an important part of Bobox, decides to run the box, the box is executed and one envelope of its input buffer is processed. If the buffer is not empty yet, it is scheduled again. When the input buffer of the box is full, the preceding box which causes this state is paused as long as the buffer is full.

Since boxes are independent on each other and their state is determined only by the data they received, it is possible to run all boxes which have at least one envelope in the input buffer in parallel. This enables automatic parallelization of the plan evaluation.

Bobox also determines the size of envelopes according to hardware parameters of the computer. The optimal size of one envelope is chosen to be $\frac{1}{2}L2/N$ bytes, where L2 denotes the total size of L2 cache in the system and N the number of physical threads [17]. According to our experiments, this value gets the best results, since all envelopes which are being processed at a moment may be completely stored in L2 cache and there is still space for auxiliary data needed for the execution.

## III. Basic concepts of parallelization of operations

The straightforward approach to the implementation of boxes is that each operation in the execution plan is represented by one box. Despite the fact that execution of one box is strictly single-threaded, there is space for parallelization. The main reasons are:

- The execution plan has typically a form of rooted tree, therefore, its branches may be executed in parallel.
- Parallel evaluation may be reached through pipeline processing, i.e. while one box is processing envelope number $i$, the consecutive box may be processing envelope number $i + 1$.

Because of these facts, even straightforward implementation of boxes yields to parallel evaluation and causes indispensable speedup of the plan evaluation. Unfortunately, it is still usually insufficient to utilize all physical threads in the computer.

In order to enhance parallelization, the operations should be implemented by more boxes than one. In this case, the number of threads which perform the operation is limited only by the number of boxes corresponding to the operation and by the number of physical threads in the system. Unfortunately, the decomposition of operation to multiple boxes may be difficult.

Generally, there are two types of operations:

- Stateless operations – their state does not depend on envelopes received so far, i.e. processing of one envelope is totally independent on processing of any other envelope. For example filter operation meets this condition.
- Stateful operations – processing of one envelope is influenced by the content of envelopes received before, therefore, their state depends on all data received so far.

### A. Decomposition of stateless operations

Decomposition of stateless operations is quite simple. The box which performs the operation may be duplicated and each instance of the box may process only the proportional part of the data. Predecessors of these boxes should be some kind of dispatch box, which resends incoming envelopes to them. Their successor must be a box, which aggregates the resulting data together and passes them to another operation. This scheme is shown in Figure 1.

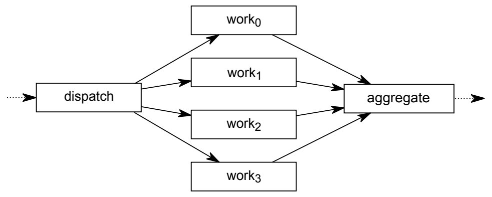


Figure 1. Decomposition of stateless operation

The `dispatch` box just forwards envelopes to the worker boxes in round-robin manner. This is very efficient, since we keep the size of envelope to be a constant. Therefore, the work boxes receive approximately the same amount of data for processing. In this case, the `aggregate` box is also simple, since it receives envelopes from work boxes in round-robin manner as well and resends them to the output.

It may seem, that `dispatch` and `aggregate` are bottlenecks of the algorithm, but they only resends the incoming envelopes, which is by several order of magnitude faster than accessing or processing their data.

## B. Decomposition of stateful operations

Stateful operations are much harder to decompose, since in order to generate envelope number $i$, the box needs to know the state after generation the envelope number $i-1$. One possible way of dealing with this is to have two algorithms: the first algorithm P (*Processing algorithm*) performs the real operation and the second algorithm S (*Skipping algorithm*) computes only the state of the box after the performance of the operation. If the second algorithm exists and is much faster than the first, then the decomposition may follow the scheme depicted in Figure 2.

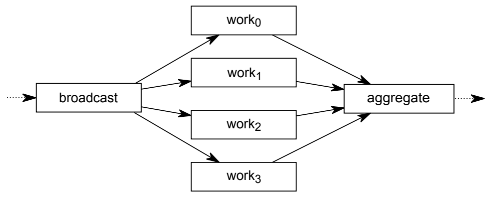


Figure 2. Decomposition of stateful operation

The `broadcast` box forwards all incoming envelopes to all work boxes. The work boxes keep a phase counter except their state. This counter is continually increasing during the evaluation and all work boxes must have these number synchronized. This may be easily done, since all work boxes receive the same input data. Boxes increase the phase counter after some event which might be reception of envelope, sending of output envelope etc. The work box number $i$ performs algorithm P if phase counter mod $N$, where $N$ is number of work boxes, is equal to $i$. Otherwise, it performs algorithm S which updates the internal state and also phase counter if needed.

The `aggregate` box might be more complicated, since it must know which work box is sending the next envelope. However, if the phase counter is increased on the sending of an envelope, the `aggregate` box works in the same way as for a stateless operation, i.e. it receives the data in round-robin manner. One example of this kind of decomposition is in Subsection IV-A.

## IV. Implementation of SPARQL operations

Operations needed for SPARQL queries evaluation have typically one or two inputs and one output. The format of envelopes used for communication between two consecutive operations is as follows: columns correspond to variables and rows contain all allowed mappings of these variables before or after the operation depending on whether the envelope is incoming or outcoming.

In the rest of this section, we describe the decomposition of the operations, which are minium for performance of several basic experiments.

### A. Decomposition of scan operation

The main objective of scan operation is to fetch from RDF database all triples that matches the input pattern. We keep six indexes to the database, which are simply the list of indexes of all triples sorted in all possible order (SPO, SOP, OPS, OSP, POS and PSO). Therefore, it is easy for any input pattern to find this range in corresponding index where all triples which match the pattern are. To find this range, we use binary search. Each work box may find this range independently. After that, the boxes start to send envelopes with requested data in round-robin manner. This is an example of a very simple stateful operation, since the only state of the box is position of the triple which was sent for the last time. Algorithm S is very simple, because it only adds value $C*(N-1)$ to the position where C is number of triples in one envelope and $N$ is number of work boxes.

### B. Decomposition of filter operation

Filter operation is a typical stateless operation, therefore, we may use the scheme for decomposition of stateless operation. Unfortunately, this straightforward approach has one drawback – the output envelope contains typically less rows than the input envelope. However, it is ineffective to work with half-empty envelopes since the manipulation with them brings some overhead such as box scheduling etc. If the envelopes are too small, this overhead may be higher than the useful work and it may yield to significantly slow down of evaluation. Therefore, we need to defragment the envelopes in order that the output envelopes of the filter operation have the optimal size.

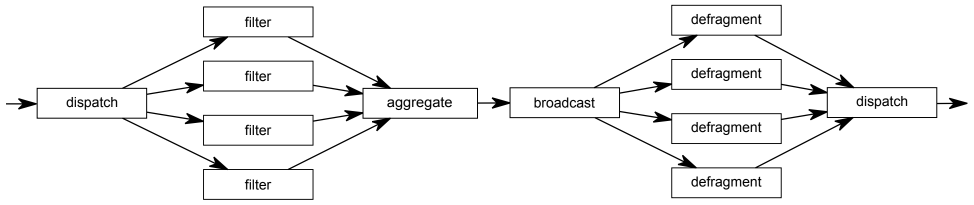


Figure 3. Decomposition of filter operation

The solution we chose is to create boxes which join the output envelopes of the filter boxes and copy their data into the envelopes with optimal size. The output envelopes from filter boxes are aggregated and broadcasted to the stateful defragment boxes. These boxes follow the scheme for stateful operation. The phase counter is increased when an output envelope is sent. Algorithm P is easy, because it only copies some subset of incoming data into the output envelope. Algorithm S is also simple, because the box knows the number of rows in incoming envelopes. According to this number, the algorithm decides to either skip the incoming envelope or process it.

Output envelopes from the defragment boxes are simply aggregated and sent to consecutive operation. The complete scheme is shown in Figure 3. On the other hand, the defragmentation may bring some slow down when the selectivity of the filter is very high. In this case, the defragmentation may become a blocking operation, which receives data, but does not send them into the output. The rest of the execution plan must wait at the worst case until the all data are filtered. The optimal solution of this problem would probably require some hints from the compilator, which we plan in future.

## C. Decomposition of nested loops join

Nested loops join operation has two inputs – we denote them as left and right. The join tries all combination of one row from the left input and one row from the right input. If the combination is compatible, i.e. selected variables in the left row are compatible with selected variables in the right row, it evaluates given filter condition on the combination. If the condition is true, then this combination should be send to the output.

Because the input data are received in envelopes, we may modify the algorithm as follows: The operation tries all combination of left and right envelopes and for each such combination it tries all rows from left and right envelope. Since the output of the nested loops join may be in arbitrarily order, both these algorithms are equivalent.

A simple approach to implementation of the algorithm is an use of $N$ work boxes. Work box number $i$ is responsible for processing all, for example left, envelopes of which sequence number mod $N$ is equal to $i$, i.e. the box generates all combinations of left envelopes it is responsible for and all right envelopes.

This approach, however, may lead to the situation, in which one box processes all data and the others do nothing. If the left input consists of very few envelopes and the opposite input of a large number of envelopes, then redundant work boxes do nothing and the employed boxes become bottleneck of the system.

To avoid this situation, the combinations of envelopes must be processed uniformly by the work boxes. Therefore, we numbered all combinations of left and right envelopes by ordinal numbers as shown in Table I. This numbering does not prefer neither left nor right input. The corresponding formula is $(L + R) * (L + R + 1)/2 + R$ where $L$ is the number of the left envelope and $R$ is the number of the right envelope. The work box number $i$ processes combination only if its number mod $N$ equals to $i$ which causes that boxes are utilized uniformly.

Table I
NUMBERING OF PAIRS OF ENVELOPES

| left \ right | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 2 | 5 | 9 |
| 1 | 1 | 4 | 8 | |
| 2 | 3 | 7 | | |
| 3 | 6 | | | |

The nested loops utilize memory bus in a great deal because all pairs of rows must be tested and therefore fetched from the main memory. This slows down the evaluation, especially when the nested loops join is computed on multiple thread since the memory becomes the bottleneck of the system. Fortunately, the modified algorithm prevents this situation. The size of envelopes is optimal according to cache memory. When the box generates all combinations of rows from one envelope with another, the data of one envelope are kept hot in cache while the data of the other are read sequentially from the main memory, and sequential access to memory is cache optimal.

## D. Optional nested loops join

Optional nested loops join works in two phases. In the first phase, it does the same operation as nested loops join. In the second phase, it sends to the output all rows from the left input which were not send to the output so far.

In order to implement the second phase, boxes need to know which rows from the left input were already sent and which not. To fulfil this requirement, we used the fact that the data in one envelope are shared among all boxes which received the envelope. Therefore, if we add one column of boolean variables to envelopes from the left input before their are broadcasted to work boxes, they can easily mark all left rows which were sent to the output. When the boxes finishes the first phase, they know which rows from the left should be sent to the output in the second phase.

## V. Experiments

To measure the influence of the decomposition of selected operations, we selected three queries from SP$^2$Bench benchmark [18] since they may be evaluated completely with the new decomposed operation. We chose this benchmark, since this is considered to be standard in the area of semantic processing. We used a computer with two Intel Xeon E5310, which run at 1,60GHz with 6MB shared L2 cache. Each processor has 4 cores, thus the system has 8 physical threads. The size of operating memory is 8 GB. The operating system is Red Hat Enterprise Linux Server 6.1 and we used g++ 4.4.5 with -O2 switch. The database was in-memory.

For each query, we performed four different tests to measure scalability of our solution. The tests differ in the number of parallel work boxes we used in decomposition of operations. We used 1,2,4 and 8 parallel boxes. Important fact is, that the first test (with 1 work box) is the same as if no decomposition was performed. Additionally, each test was evaluated twice – single-threaded and multithreaded, i.e. we used each physical thread in the system for the query evaluation.

We performed each experiment 5 times and we selected the median of all measurement. Since the database loading and query compilation time is not relevant for the experiments the results do not include them. On ther other hand, they include time spent by the operation decomposition, since such overhead is an integral part of the query execution.

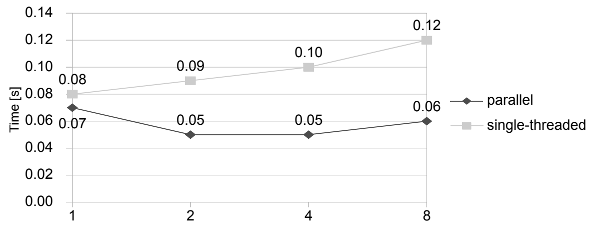


Figure 4. Results of query q1

The first query we used is the query q1 on a database with 5M triples. This query uses only index scan and nested loops join operation. The results are show in Figure 4. The second query is q3a on a database with 250k triples which utilizes both nested loops join and filter operation (Figure 5) and the last is query q6 on the same database as the query q3a which additionally uses optional nested loops joins (Figure 6).

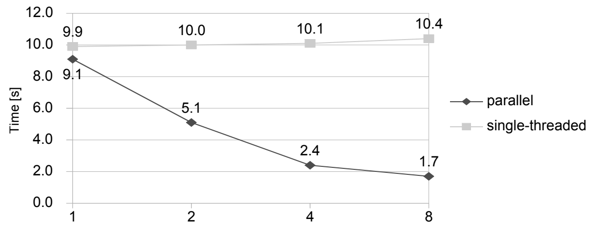


Figure 5. Results of query q3a

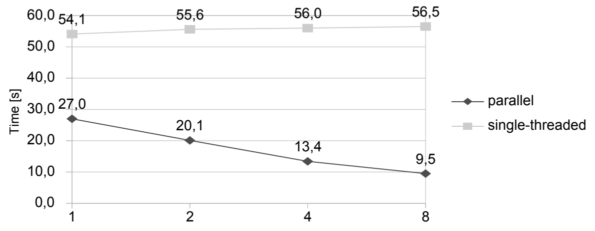


Figure 6. Results of query q6

The query q6 benefits from multithreading even in the case when no decomposition is performed. This is caused due to the pipeline processing and the fact that there are independent braches in the plan. Additional experiments show that increasing the number of work boxes in the decomposition increases the performance.

The query q3a benefits only a little from parallel environment when there is no decomposition. The execution plan contains one filter operation followed by the nested loops join and the small speed up is caused by the fact that filter operation is much faster operation then the nested loops join. Therefore, the evaluation of the filter operation in parallel with the join operation is not so profitable. However, the decomposition of the operations, mainly of the nested loops join, increases the performance noticeably.

The query q1 is a little bit atypical. The graph contains two consecutive nested loops joins, but the input data are very small. In fact, they fit in one or two envelopes. Therefore, the decomposition of problem does not cause significant speed up, since the whole operation is performed by only a subset of work boxes.

In each test, the single-threaded evaluation shows that with the increasing number of work boxes, the overhead caused by the manipulation of envelopes is slightly increasing.

## VI. Conclusion and Future Work

In this paper, we presented possibility of the decomposition of basic SPARQL operations to increase parallelism during the evaluation of execution plans when evaluated by Bobox. In our pilot implementation, we focused on subset of basic operations - nested loops join, filter and index scan. These operations were sufficient to measure the influence of the decomposition on a query evaluation performance.

The tests were performed using SP$^2$Bench test queries that used implemented operations only. As follows from presented results, further decomposition of basic SPARQL operations can provide additional performance gain – for 8 physical threads in the system, the evaluation of time consuming queries is almost 6 times faster than single-threaded execution, therefore, our solution scales well.

We do not compare these results with other RDF stores, since the execution plans were not optimal because we forced the compiler to use only nested loops join instead of merge joins, with which the evaluation would be much faster. The comparison between version with optimal but not decomposed plans and the SESAME database may be found in our previous work [9]. On the other hand, for the database sizes, for which we performed the tests, the evaluation of the queries with decomposed operations is faster than the evaluation without the decomposition but with optimal operations.

In the future, we want to focus on paralellizing extended set of basic SPARQL operations such as sorting, union or faster join operations such as hash join, or merge join. After that, we will be able to perform more extensive and accurate performance tests and comparisons of parallel version with similar semantic engines.

## Acknowledgment

The authors would like to thank the GAUK project no. 28910, 277911 and SVV-2011-263312, and GACR project no. 202/10/0761, which supported this paper.

## References

[1] J. J. Carroll and G. Klyne, *Resource Description Framework: Concepts and Abstract Syntax*, W3C, 2004. [Online]. Available: http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

[2] E. Prud'hommeaux and A. Seaborne, *SPARQL Query Language for RDF*, W3C, 2008. [Online]. Available: http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/

[3] Y. Yan, C. Wang, A. Zhou, W. Qian, L. Ma, and Y. Pan, "Efficiently querying rdf data in triple stores," in *Proceeding of the 17th international conference on World Wide Web*, ser. WWW '08. New York, NY, USA: ACM, 2008, pp. 1053–1054.

[4] J. Broekstra, A. Kampman, and F. v. Harmelen, "Sesame: A generic architecture for storing and querying RDF and RDF schema," in *ISWC '02: Proceedings of the First International Semantic Web Conference on The Semantic Web*. London, UK: Springer-Verlag, 2002, pp. 54–68.

[5] A. Kiryakov, D. Ognyanov, and D. Manov, "Owlim a pragmatic semantic repository for owl," 2005, pp. 182–192.

[6] T. Neumann and G. Weikum, "The rdf-3x engine for scalable management of rdf data," *The VLDB Journal*, vol. 19, pp. 91–113, February 2010.

[7] J. Huang, D. Abadi, and K. Ren, "Scalable sparql querying of large rdf graphs," *Proceedings of the VLDB Endowment*, vol. 4, no. 11, 2011.

[8] J. Groppe and S. Groppe, "Parallelizing join computations of sparql queries for large semantic web databases," in *Proceedings of the 2011 ACM Symposium on Applied Computing*. ACM, 2011, pp. 1681–1686.

[9] M. Cermak, J. Dokulil, Z. Falt, and F. Zavoral, "SPARQL Query Processing Using Bobox Framework," in *SEMAPRO 2011, The Fifth International Conference on Advances in Semantic Processing*. IARIA, 2011, pp. 104–109.

[10] A. Brown and C. Kozyrakis, "Parallelizing the index-nested-loops database join primitive on a shared-nothing cluster," 1998.

[11] C. Kim, T. Kaldewey, V. W. Lee, E. Sedlar, A. D. Nguyen, N. Satish, J. Chhugani, A. Di Blas, and P. Dubey, "Sort vs. hash revisited: fast join implementation on modern multi-core cpus," *Proc. VLDB Endow.*, vol. 2, pp. 1378–1389, August 2009.

[12] H. Lu, K.-L. Tan, and M.-C. Shan, "Hash-based join algorithms for multiprocessor computers," in *Proceedings of the 16th International Conference on Very Large Data Bases*, ser. VLDB '90. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990, pp. 198–209.

[13] B. Gedik, P. S. Yu, and R. R. Bordawekar, "Executing stream joins on the cell processor," in *Proceedings of the 33rd international conference on Very large data bases*, ser. VLDB '07. VLDB Endowment, 2007, pp. 363–374.

[14] B. He, K. Yang, R. Fang, M. Lu, N. Govindaraju, Q. Luo, and P. Sander, "Relational joins on graphics processors," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '08. New York, NY, USA: ACM, 2008, pp. 511–524.

[15] D. Bednarek, J. Dokulil, J. Yaghob, and F. Zavoral, "Using Methods of Parallel Semi-structured Data Processing for Semantic Web," in *3rd International Conference on Advances in Semantic Processing, SEMAPRO*. IEEE Computer Society Press, 2009, pp. 44–49.

[16] M. Cermak, J. Dokulil, and F. Zavoral, "SPARQL Compiler for Bobox," in *SEMAPRO 2010, The Fourth International Conference on Advances in Semantic Processing*. IARIA, 2010, pp. 100–105.

[17] Z. Falt and J. Yaghob, "Task Scheduling in Data Stream Processing," in *Proceedings of the Dateso 2011 Workshop*. Citeseer, 2011, pp. 85–96.

[18] M. Schmidt, T. Hornung, G. Lausen, and C. Pinkel, "SP2Bench: A SPARQL performance benchmark," *CoRR*, vol. abs/0806.4627, 2008.

# A New Approach For Top-k Flexible Queries In Large Database Using The Knowledge Discovered

Amel Grissa Touzi
Technologies of Information and Communications
ENIT
Tunisia
amel.touzi@enit.rnu.tn

Habib Ounalli
Informatique
FST
Tunisia
Habib.ounelli@fst.rnu.tn

***Abstract*—In this paper, we propose our contribution to support top-k flexible query in large DB. Generally, the current top-k query processing techniques focus on Boolean queries, and cannot be applied to the large DB seen the gigantic number of data. Our approach proposes to uses the generated knowledge result of an algorithm for Knowledge Discovery in Database (KDD). It consists of two steps: 1) Extraction of Knowledge by applying a new approach for KDD through the fusion of conceptual clustering, fuzzy logic and formal concept analysis, and 2) generation efficient answers to top-k flexible queries using the generated knowledge in the first step. We prove that this approach is optimum sight that the evaluation of the query is not done on the set of starting data which are enormous but rather by using the set of knowledge on these data; what is to our opinion one of the principal's goal of KDD approaches.**

***Keywords-Top-k queries; KDD; Data minig; FCA; Fuzzy logic.***

## I. INTRODUCTION

Top-k queries have attracted much interest in many different areas such as network and system monitoring [1, 2], information retrieval [3], sensor networks [4], multimedia databases [5], spatial data analysis [6], P2P systems [7] , data stream management systems [8], etc. The main reason for such interest is that they avoid overwhelming the user with large numbers of uninteresting answers which are resource-consuming.

The problem of answering top-k queries can be modeled as follows [9]. Suppose we have m lists of n data items such that each data item has a local score in each list and the lists are sorted according to the local scores of their data items. And each data item has an overall score which is computed based on its local scores in all lists using a given scoring function. Then the problem is to find the k data items whose overall scores are the highest. The most efficient algorithm for answering top-k queries over sorted lists is the Threshold Algorithm (TA) [10]. Based on TA, many algorithms have been proposed in the literature [9, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21].

Unfortunately, current top-k query processing techniques focus on Boolean queries, and cannot be applied to the large DB seen the gigantic number of data.

In this paper, we propose to use the set of rules generated by an algorithm of KDD for the evaluation of the top-k flexible query in large DB. Indeed, in our opinion these rules are very beneficial in the optimization of the evaluation of the flexible query. Our approach consists of two steps: 1) Extraction of Knowledge and 2) generation efficient answers to top-k flexible queries using the generated knowledge in the first step.

In literature, several algorithms for KDD were proposed [22]. Generally, generated rules by these algorithms, exceeding some times of thousand rules, are not easily exploitable [23, 24]. In our opinion, this constitutes a real handicap to use them in the evaluation of the flexible query. To cure these problems, we propose a new KDD approach having the following characteristics Extract knowledge taking in consideration another degree of granularity into the process of knowledge extraction. Indeed, we propose to define rules (Meta-Rules) between classes resulting from a preliminary fuzzy clustering on the data.

The rest of the paper is organized as follows: Section 2 presents the basic concepts of top-*k* queries, discovering association rules and Formal Concept Analysis (FCA). Section 3 presents problems and limits of the existing approaches. Section 4 gives notations related to our new proposed approach. Section 5 describes our KDD model. Sections 6 evaluate the proposed approach. We finish this paper with a conclusion and a presentation of some future works.

## II. BASIC CONCEPTS

In this section, we present the basic concepts of top-*k* queries, discovering association rules and Formal Concept Analysis (FCA).

### *A. Top-k queries*

Originally, top-*k* (ranking) queries have been proposed in a multimedia context [25, 26, 27], where the aim is to produce a number of highest ranking results from a set of ordered lists, according to monotone ranking functions defined on the elements of the lists. Each list consists of a tuple identifier and an attribute value and is arranged in non increasing order of that value. Each tuple identifier is assigned a *score* according to the ranking function computed on the attribute values of the associated list and the objective is to identify the *k* tuples with the highest scores.

The threshold algorithm (TA) constitutes the state of the art for top-*k* query answering [9, 10, 15]. The TA algorithm accesses list items in lock-step, traversing each list in a sequential fashion.

Several variants of the basic ideas of the TA algorithm have been proposed in the literature. In one variant (TA-Sorted) [9, 15] lists are always accessed sequentially. No random accesses are performed and thus at any point the score of a tuple identifier is partially known. Variants of the basic top-*k* problem have been considered in a web context [11], in a relational database context [14, 19] as well as on join scenarios [17, 18, 20]. Others considered nearest neighbor type of approaches for this problem [12, 13, 21].

## B. *Discovering Association Rules*

Association rules mining have been developed in order to analyze basket data in a marketing environment. Input data are composed of transactions: each transaction consists of items purchased by a consumer during a single visit. Output data is composed of rules. An example of an association rule is "90% of transactions that involve the purchase of bread and butter also include milk" [28]. Even if this method was introduced in the context of Market Business Analysis, it can also be used to search for frequent co-occurrences in every large data set.

The first efficient algorithm to mine association rules is APriori [29]. The first step of this algorithm is the research of frequent itemsets. The user gives a minimum threshold for the support and the algorithm searches all itemsets that appear with a support greater than this threshold. The second step is to build rules from the itemsets found in the first step. The algorithm computes confidence of each rule and keeps only those where confidence is greater than a threshold defined by the user. One of the main problems is to define support and confidence thresholds. Other algorithms were proposed to improve computational efficiency. Among them, we mention CLOSED [30], CHARM [31] and TITANIC [32].

## C. *Fuzzy Conceptual Scaling and FCA*

Conceptual scaling theory is the central part in Formal Concept Analysis (FCA). It allows introduce for the embedding of the given data much more general scales than the usual chains and direct products of chains. In the direct products of the concept lattices of these scales the given data can be embedded. FCA starts with the notion of a formal context specifying which objects have what attributes and thus a formal context may be viewed as a binary relation between the object set and the attribute set with the values 0 and 1. In [33], an ordered lattice extension theory has been proposed: Fuzzy Formal Concept Analysis (FFCA), in which uncertainty information is directly represented by a real number of membership value in the range of [0,1]. This number is equal to similarity defined as follow:

**Definition.** The similarity of a fuzzy formal concept $C_1 = (\varphi(A_1), B_1)$ and its sub-concept $C_2 = (\varphi(A_2), B_2)$ is defined as:

$$S(C_1, C_2) = \frac{|\varphi(A_1) \cap \varphi(A_2)|}{|\varphi(A_1) \cup \varphi(A_2)|}$$

where $\cap$ and $\cup$ refer intersection and union operators on fuzzy sets, respectively; φ is the relation which associates degrees to the elements of a fuzzy set I = X × V (X is the set of objects and V is the set of attributes). Each pair $(x_i, v_j) \in$ I has a membership degree $\mu(x_i, v_j) \in [0,1]$.

In [34, 35], we showed that these FFCA are very powerful as well in the interpretation of the results of the fuzzy clustering and in optimization of the flexible query.

**Example:** Let a RDB describing apartment announces, where the primary key of each relation is underlined:

**Announce (**<u>réfAn</u>, date_an, codPr, codAp**)**
**Apartment (**<u>codAp</u>, price, state, site, surface, n_ room, city**)**
**Owner (**<u>codPr</u>, name,surname, num_phone, address**)**

Price ∈{100, 110, 120, 160, 180, 200, 350, 400, 450, 500, 640, 700, 720, 2000, 2100, 2900, 3000} and Surfaces ∈{30, 50, 52, 60, 68, 70, 140, 150, 200, 220, 250, 300, 400, 500}. Let us suppose that: The degrees of importance of the relieving attributes price and surface are respectively 0.6 and 0.4. Table I presents the results of fuzzy clustering (using Fuzzy C-Means [36, 37]) applied to *Price* and *Surface* attributes.

For *Price* attribute, fuzzy clustering generates three clusters (C1, C2 and C3). For *Surface* attribute, two clusters have been generated (C4 and C5). In our example, $\alpha-Cut$ ( *Price*) = 0.3 and $\alpha-Cut$( *Surface*) = 0.5, so, the Table I can be rewriting as show in Table II. The corresponding fuzzy concept lattices of fuzzy context presented in Table II, noted as TAH's are given by the line diagrams presented in the Figure 1.

TABLE I. FUZZY CONCEPTUAL SCALES FOR PRICE AND SURFACE ATTRIBUTES.

| | **Price** | | | **Surface** | |
|---|---|---|---|---|---|
| | **C1** | **C2** | **C3** | **C4** | **C5** |
| **A1** | 0.1 | 0.5 | 0.4 | 0.5 | 0.5 |
| **A2** | 0.3 | 0.6 | 0.1 | 0.4 | 0.6 |
| **A3** | 0.7 | 0.1 | 0.2 | 0.7 | 0.3 |
| **A4** | 0.1 | 0.4 | 0.5 | 0.2 | 0.8 |
| **A5** | 0.2 | 0.4 | 0.4 | 0.6 | 0.4 |
| **A6** | 0.5 | 0.3 | 0.2 | 0.5 | 0.5 |

TABLE II. FUZZY CONCEPTUAL SCALES FOR PRICE AND SURFACE ATTRIBUTES WITH $\alpha-Cut$.

| | Price | | | Surface | |
|---|---|---|---|---|---|
| | C1 | C2 | C3 | C4 | C5 |
| A1 | - | 0.5 | 0.4 | 0.5 | 0.5 |
| A2 | 0.3 | 0.6 | - | - | 0.6 |
| A3 | 0.7 | - | - | 0.7 | - |
| A4 | - | 0.4 | 0.5 | - | 0.8 |
| A5 | - | 0.4 | 0.4 | 0.6 | - |
| A6 | 0.5 | 0.3 | - | 0.5 | 0.5 |

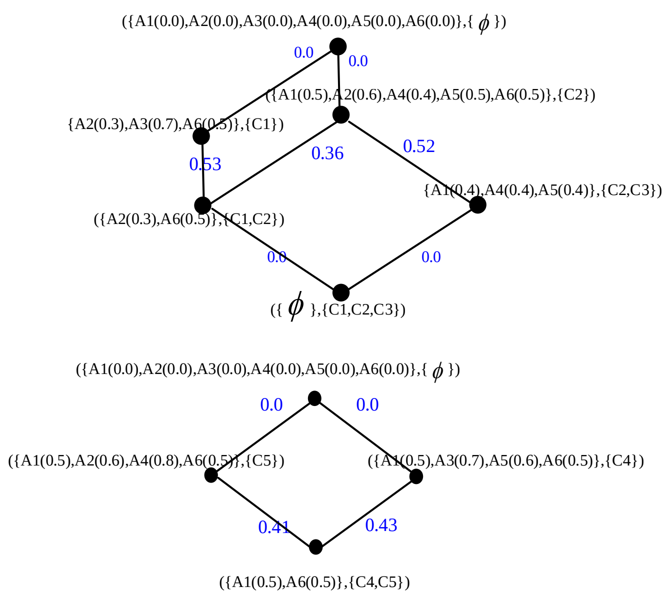

({A1(0.0),A2(0.0),A3(0.0),A4(0.0),A5(0.0),A6(0.0)},{ ϕ })
0.0
0.0
({A1(0.5),A2(0.6),A4(0.4),A5(0.5),A6(0.5)},{C2})
{A2(0.3),A3(0.7),A6(0.5)},{C1})
0.53
0.36
0.52
{A1(0.4),A4(0.4),A5(0.4)},{C2,C3})
({A2(0.3),A6(0.5)},{C1,C2})
0.0
0.0
({ ϕ },{C1,C2,C3})
({A1(0.0),A2(0.0),A3(0.0),A4(0.0),A5(0.0),A6(0.0)},{ ϕ })
0.0
0.0
({A1(0.5),A2(0.6),A4(0.8),A6(0.5)},{C5})
({A1(0.5),A3(0.7),A5(0.6),A6(0.5)},{C4})
0.41
0.43
({A1(0.5),A6(0.5)},{C4,C5})

Figure 1. Price TAH and Surface TAH.

## III. PROBLEMS AND CONTRIBUTIONS

We are confronted to two types of problems:

- At the level of the requests addressed to large databases, the current top-k query processing techniques focus on Boolean queries, and cannot be applied to the large DB seen the gigantic number of data. The majority of the proposed systems uses a score function f ad-hoc and delivers the k better answers of the total order obtained by f. However, this score function remains difficult to establish seen the voluminous number of data.

- At the level of KDD approaches: several solutions have been used but, authors of these approaches don't propose any solutions for the evaluation of the queries knowing knowledge generated by their approaches. Thus, the goal to exploit these data is often neglected.

In our opinion, this problem was not really neglected but it was not sufficiently treated since the generated rules by these approaches, exceeding some times of thousand rules, are not easily exploitable [23, 24]. Indeed, this big number of rules is due to the fact that these approaches try to determine rules starting from the data or a data variety like the frequent item-sets or the frequent closed item-sets, which may be huge. To cure all these problems, we propose:

1) A new approach for knowledge extraction taking in consideration another degree of granularity into the process of knowledge extraction. We propose to define rules (Meta-Rules) between classes resulting from a preliminary classification on the data. Indeed while classifying data, we construct homogeneous groups of data having the same properties, so defining rules between clusters implies that all the data elements belonging to those clusters will be necessarily dependent on these same rules. Thus, the number of generated rules is smaller since one processes the extraction of the knowledge on the clusters which number is relatively lower compared to the initial data elements.
2) A new algorithm to support database flexible querying using the generated knowledge in the first step. This approach allows the end-user to easily exploit all knowledge generated.

## IV. NOTATIONS RELATED TO OUR KDD MODEL

In this section, we present the notations related fuzzy conceptual scaling and some news concepts for our new approach.

**Definition.** A *fuzzy Clusters Lattice* (FCL) of a Fuzzy Formal Concept Lattice, is consist on a Fuzzy concept lattice such as each equivalence class (i.e. a node of the lattice) contains only the intentional description (intent) of the associated fuzzy formal concept.

We make in this case a certain abstraction on the list of the objects with their degrees of membership in the clusters. The nodes of FCL are clusters ordered by the inclusion relation.

**Definitions.** A level L of a FCL is the set of nodes of FCL having cardinality equal to L.

A Knowledge level is an abstraction level is regarded as a level in the FCL generated.

**Definition.** Let *I= {C1, C2, ..., Cp, Cq , ..., Cn} n* Clusters generated by a fuzzy clustering algorithm. A *fuzzy association meta-rule* (called *meta-rule)* is an implication of the form **R: I1 => I2, (CF***)* where

*I1 = { C1, C2, ..., Cp* } and *I2={ Cq , ..., Cn* }.

*I1* and *I2* are called, respectively, the *premise part* and *conclusion part* of the meta-rule *R*. The value CF is in ]0..1] and called *Confidence Factor* of this rule. This value indicates the relative degree of importance of this meta-rule.

R is interpreted as follows: if an object belongs to a cluster $C1 \cap C2 \cap ... \cap Cp$ then this object can also belongs to the cluster $Cq \cap ... \cap Cn$ with a probability equal to *CF*.

Note that classical (or crisp) association meta-rules can be defined as a special case of fuzzy association meta-rules. Indeed, when *CF=1*, then a fuzzy association meta-rule is equivalent to a classical one.

**Example.** Let R: C1 => C2 (60%). This means that any object belongs to a cluster *C1* can also belongs to the cluster *C2* with a probability equal to *60%.*

**Definition.** Let *A1,A2...,Ap,Aq,...An* n attributes having respectively {*l11,l12...,l*1m },**{***l21,l22...* ,*l*2m}..., **{***lp1 ,lp2...,* lpm }, **{***lq1,lq2...,l*qm}…., ,**{***ln1,ln2...,l*nm} as linguistic labels. A *fuzzy association rule* (or *rule*) is an implication of the form

**r : I1 => I2, (CF)**;

where *I1 = { A1(l1), A2(l2), ..., Ap*(lp) } and *I2= {Aq(lq), ..., An*(ln) }. *Ai(li) models the attribute Ai having a linguistic label li. I1* and *I2* are called, respectively, the *premise part* and *conclusion part* of the fuzzy rule *r*. The value CF is in ]0..1] and called *Confidence Factor* of this rule.

**Definition.** We define *Meta Knowledge (resp. Knowledge),* as a set of fuzzy association meta-rule (resp. rule). We define *the level i of Meta Knowledge (resp. knowledge)* as the set of fuzzy association meta-rule (resp. rule) on all objects verifying *i properties*.

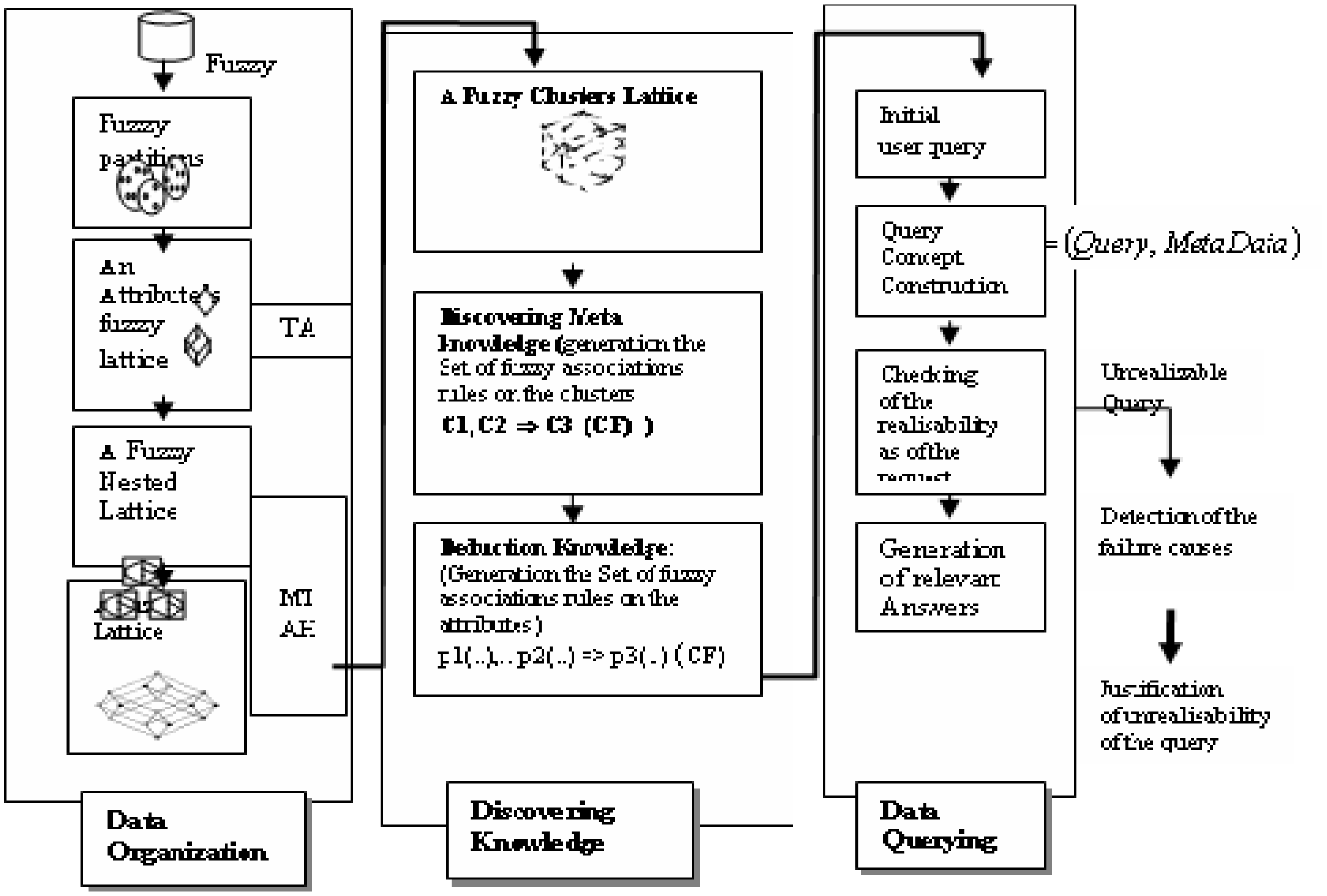

Fuzzy
Fuzzy partitions
An Attribute's fuzzy lattice
TA
A Fuzzy Nested Lattice
Lattice
MT
AH
Data Organization
A Fuzzy Clusters Lattice
Discovering Meta Knowledge (generation the Set of fuzzy associations rules on the clusters
C1, C2 ⇒ C3 (CF) )
Deduction Knowledge: (Generation the Set of fuzzy associations rules on the attributes)
p1(..),..p2(..) => p3(..) (CF)
Discovering Knowledge
Initial user query
Query Concept Construction
= (Query, MetaData)
Checking of the realisability as of the request
Generation of relevant Answers
Unrealizable Query
Detection of the failure causes
Justification of unrealisability of the query
Data Querying

Figure 2. Proposed Approach

**Proposition.** *Rewriting meta- rule*

Let *C1= {A1, A2, ..., An} and C2={B1 , ..., Bm*} two set of Clusters. The fuzzy association meta-rule

R : A1,..,An => B1,..,Bm (CF)

is equivalent to R1 defined as follow:

R1: A1,..,An => D1,..,Dq (CF) such that

{D1,…,Dq} = C2\C1

## V. KDD MODEL DESCRIPTION

In this section, we present the architecture of the KDD model and the process for discovering and exploiting knowledge.

The architecture of the KDD model is presented in Figure 2. It consists of three steps: the first step consists in data organization the second aims at Extraction of Knowledge and the third step consists to define a new method for support database flexible querying using the generated knowledge in the second step. In the following, we detail these different steps.

### *A. Data Organization Step*

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 attributed on attribute's domain. This step consists of TAH's and MTAH generation of relieving attributes. This step is very important in KDD Process because it allows to define and interpreter the distribution of objects in the various clusters.

**Example**: Let a relational database describing apartment announces. Table I presents the results of fuzzy clustering applied to *Price* and *Surface* attributes.

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 labels*** provided by the user or a domain expert.

For example, the fuzzy labels *Small* and *Large* could belong to a partition built over the domain of the attribute *Surface*. Also, the fuzzy labels *Low*, *Medium* and *High*, could belong to a partition built over the domain of the attribute *Price*. The Table III presents the correspondence of the linguistic labels and their designations for the attributes *Price* and *Surface*. The corresponding fuzzy concept lattices of fuzzy context is presented in Table IV; noted as TAH's are given by the line diagrams presented in Figure 1.

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. Figure 3 shows the fuzzy nested lattice constructed from Figure 1.

TABLE III. CORRESPONDENCE OF THE LINGUISTIC LABELS AND THEIR DESIGNATIONS

| Attribute | Linguistic labels | Designation |
|---|---|---|
| Price | Low | C1 |
| Price | Medium | C2 |
| Price | High | C3 |
| Surface | Small | C4 |
| Surface | Large | C5 |

TABLE IV. FUZZY CONCEPTUAL SCALES FOR PRICE AND SURFACE ATTRIBUTES WITH $\alpha - Cut$

| | Price | | | Surface | |
|---|---|---|---|---|---|
| | Low | Medium | High | Small | Large |
| A1 | - | 0.5 | 0.4 | 0.5 | 0.5 |
| A2 | 0.3 | 0.6 | - | - | 0.6 |
| A3 | 0.7 | - | - | 0.7 | - |
| A4 | - | 0.4 | 0.5 | - | 0.8 |
| A5 | - | 0.5 | 0.5 | 0.6 | - |
| A6 | 0.5 | 0.5 | - | 0.5 | 0.5 |

## B. *Discovering Knowledge Step*

This step consists on Extraction of Knowledge. It consists to deduce the Fuzzy Cluster Lattice corresponding to MTAH lattice generated in the first step, then traverse this lattice to extract the Meta Knowledge ( Set of fuzzy associations meta-rules on the clusters ), and in end deduce the rules modeling the Knowledge (Set of fuzzy associations rules on the attributes). This set is denoted by SFR.

**Example:** From the fuzzy lattice, obtained in the first step (Figure 3), we can draw the correspondent FCL. As shown from the Figure 4, we obtain a lattice more reduced, simpler to traverse and stored.

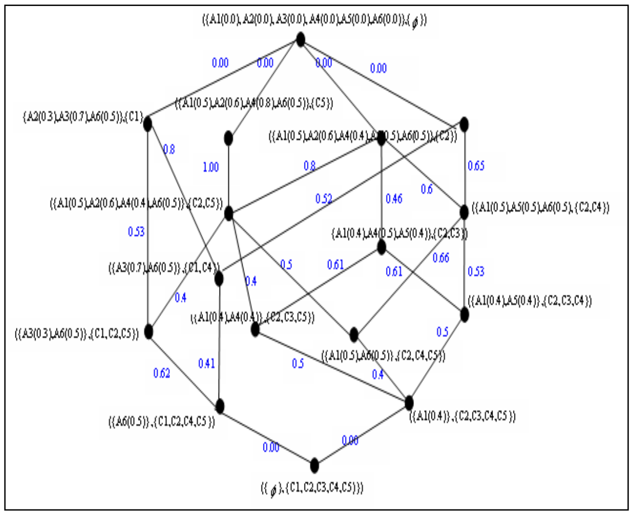

({A1(0.0), A2(0.0), A3(0.0), A4(0.0),A5(0.0),A6(0.0)},{ ϕ })
0.00
0.00
0.00
0.00
{A2(0.3),A3(0.7),A6(0.5)},{C1}
({A1(0.5),A2(0.6),A4(0.8),A6(0.5)},{C5})
({A1(0.5),A2(0.6),A4(0.4),A5(0.5),A6(0.5)},{C2})
0.8
1.00
0.8
0.46
0.6
0.65
({A1(0.5),A2(0.6),A4(0.4),A6(0.5)},{C2,C5})
0.52
({A1(0.5),A5(0.5),A6(0.5),{C2,C4})
0.53
{A1(0.4),A4(0.5),A5(0.4)},{C2,C3})
({A3(0.7),A6(0.5)},{C1,C4})
0.5
0.61
0.61
0.66
0.53
0.4
0.4
({A1(0.4),A4(0.4)},{C2,C3,C5})
({A1(0.4),A5(0.4)},{C2,C3,C4})
({A3(0.3),A6(0.5)},{C1,C2,C5})
0.5
({A1(0.5),A6(0.5)},{C2,C4,C5})
0.62
0.41
0.5
0.4
({A6(0.5)},{C1,C2,C4,C5})
({A1(0.4)},{C2,C3,C4,C5})
0.00
0.00
({ ϕ },{C1,C2,C3,C4,C5})

Figure 3. Fuzzy Lattice: MTAH

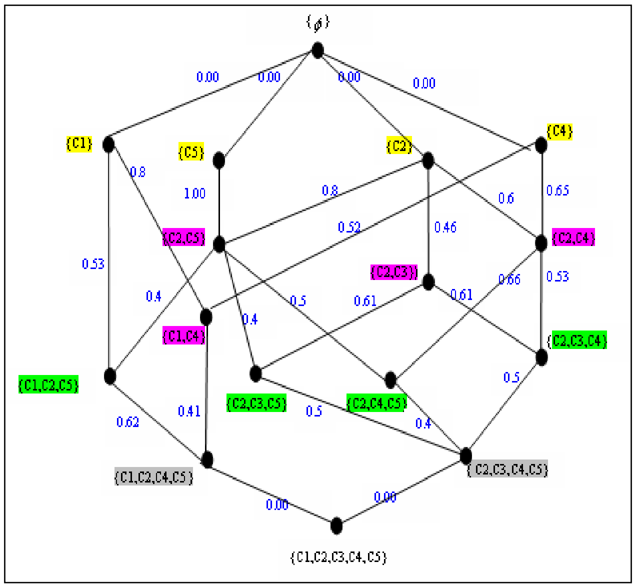


Figure 4. The FCL

Considering the FCL in Figure 4, we can generate the following levels with the corresponding FCL. The Level 0 and Level 5 are both the root and leaves of FCL. The Level 1 corresponds to the nodes {C1}, {C5},{C2},{C4}. Generally Level i corresponds to the nodes having i clusters. This permits to identify all the existing of overlapping between i clusters. It allows the knowledge discovery on all objects belonging to the intersection of these i clusters.

Thus, the derivation of fuzzy association meta-rules can be performed straightforwardly. Indeed, the meta-rule represent "inter-node" implications, assorted with the CF, between two adjacent comparable equivalence classes, i.e., from a set of clusters to another set of clusters immediately covering it. The confidence Factor will be equal to the weight of the arc binding the two nodes. Such an implication brings into participate two comparable equivalence classes, i.e. of a set of clusters towards another set of cluster including it in the partial order structure.

**Example** The meta-rule C2 $\Rightarrow$ C2,C5 (80%), is generated starting from the two equivalence classes, whose their respective nodes are Clusters {C2}, {C2,C5} having as distance d=0.8. This meta-rule can rewrite as C2 $\Rightarrow$ C5 (80%).

The Algorithm for Discovering Fuzzy Association Meta-rules traverses the search space (FCL) by level to determine the Fuzzy Meta Rules Set (FMRS). As input it takes the lattice of Clusters FCL and returns, as output, the list of all Fuzzy Meta Rules Set (FMRS) generated. It works as follows: For each non empty node $\in$ FCL in descending, it generates all meta-rules with one cluster in conclusion (level 1). Then, it generates the set of all meta-rules with two Clusters in conclusion. The same process is applied to generate conclusions with four clusters, and so on until conclusions with *n* clusters have been generated.

Let's note that the FMRS set doesn't contain any redundant rule. This is due that of a level to another of the lattice the nodes are obligatorily distinct (by definition even of a level of lattice).

**Example.**

We present in Table V the Meta-Knowledge generated from Table I. The list of rules is order by level of Knowledge (every *level i of knowledge* fact to intervene *i properties*) .

From the FMRS set we can easily deduce the rules modeling the Knowledge SFR. It's sufficient to use the Table III presents the correspondence of the linguistic labels and their designations for the attributes Price and Surface.

**Example.** The meta-rule C2 => C5 80% is transformed in Price (Medium) => Surface (Large) 83%

TABLE V. META-KNOWLEDGE GENERATION FROM TABLE I

| |
|---|
| **Level 1: List of clusters that permits to generate other properties** |
| R1: => C1 R2: => C2 R3: => C4 R4: => C5 |
| **Level 2: Definition of the objects belonging to two clusters** |
| R5: C1 => C4 80% R6: C5=> C2 100% R7: C2 => C5 80% |
| R9: C4 => C2 65% R8: C2 => C3 46% R10: C4 => C1 52% |
| R11: C2 => C4 60% |
| **Level 3: Definition of the objects belonging to three clusters** |
| R12: C1 => C2, C5 53% R13: C2,C5 =>C1 40% |
| R14: C2,C5 => C3 40% R15: C2, C5 => C4 50% |
| R16: C2,C3 => C5 61% R17: C2,C3 => C4 61% |
| R18 C2, C4 => C5 66% R19 :C2,C4 => C3 53% |
| **Level 4 definition of the objects belonging to four clusters** |
| R20 : C1, C4 => C2, C5 41% R24: C2, C3,C4 => C5 50% |
| R21: C2, C3, C5 => C4 50% R22: C1, C2, C5 => C4 62% |
| R23: C2, C4, C5 => C3 40% |

## C. Data Querying Step

This step presents our flexible interrogation algorithm using the generated knowledge in the second step. Let R the user Query. The pseudo-code for the algorithm is given in the following:

```
Evaluation of Flexible Query
Input: The user Query R
Output : List of answers
Begin
    - Concept_Query ( R, $Q_B$ )
    - let i = Cardinality ($Q_B$)
    - examine the rules in level i.
    - if there is a rule which utilizes all the elements
      of $Q_B$ then
            R is realizable with CF = 100%
        Extract (R, i);
        else if there is a rule which utilizes at least elements
            of the $Q_B$ then
                R is realizable with CF < 100%
                Extract (R ,i);
            else R is not realizable.
End
```

Note that ***Concept_Query*** ( R, $Q_B$ ) : is a procedure that determine the concept $Q_B$ of R. ***Extract***(R,i) : is a procedure that determines answers of the request while using the *Backward chaining*. This procedure calls upon all the rules closely related to the request of level < = i.

For better explaining this step, we consider a relational database table describing apartment announces and the following query:

```
    Select refAn, price, surface
    From   Announce, Apartment
    Where  price = 105                                   (A1)
Q   and    surface = 75                                  (A2)
    and    city = 'Paris'                                (A3)
    and    place = '16ème arrondissement'                (A4)
    and    Apartment.codAp= Announce.codAp               (A5)
```

In this query, the user wishes that its preferences be considered according to the descending order: Price and Surface and **Top-k=2.** In other words, returned data must be ordered and presented at the user according to these preferences. Without this flexibility, the user must refine these search keys until obtaining satisfaction if required since it does not have precise knowledge on the data which it consults.

According to the criteria of the query $Q$ , only the **A1 and A2 criteria** correspond to relievable attributes.

Initially, we determine starting from the DB the tuples satisfying the non relievable criteria ($A_3$,$A_4$,$A_5$), result of the following query:

```
    Select refAn, price, surface
    From   Announce, Apartment
Q   and    city = 'Paris'                                (A3)
    and    place = '16ème arrondissement'                (A4)
    and    Apartment.codAp= Announce.codAp               (A5)
```

These tuples is broken up into clusters according to labels of the relievable attributes *Price* and *Surface*

*1) Construction of the query concept*

We define a query concept $Q=(Q_A,Q_B)$ where $Q_A$ is a name to indicate a required extension and $Q_B$ is the set of clusters describing the data reached by the query. The set $Q_B$ of clusters is determined by the following procedure:

```
Procedure Construction of the query concept
Input : Vector $V(A)=\{v_j : j=1,...,C(A)\}$ of cluster centres of relievable attribute $A$ and the value of $Q$ associated to this last.
Output : Query concept $Q=(Q_A,Q_B)$.
Begin
Step 1 : Calculate the membership degrees of the specified clusters for each value of the criterion of $Q$ associated to the relievable attribute $A$.
Step 2 : Apply $\alpha-Cut$ to generate the fuzzy context.
Step 3 : Form the set $Q_B$ of clusters whose membership is higher than the $\alpha-Cut$ value.
End Procedure
```

These metadata are given with part of the fuzzy clustering operation to determine the objects membership's degrees in the various clusters. Table VI present the membership degrees associated to the query. These degrees are obtained while basing on memberships matrix obtained by a fuzzy clustering algorithm. Then, we apply the $\alpha-Cut$ for each attribute to minimize the number of concepts.

TABLE VI. QUERY MEMBERSHIPS DEGREES.

| Price | | | Surface | |
|---|---|---|---|---|
| C1 | C2 | C3 | C4 | C5 |
| 0.1 | 0.3 | 0.6 | 0.2 | 0.8 |

According to our example, the query $Q$ seek the data sources having the metadata $Q_B$ ={C2,C3,C5}.

**Proposition:** A data source $S$ is relevant for a given query $Q=(Q_A,Q_B)$ if and only if $S$ is characterized by at least one of the meta-data given from $Q_B$. The relevance degree of $S$ is given by the number of meta-data that $S$ divide with $Q_B$.

This proposition of relevance is at the base of the research process which detailed in the rest of this section and illustrated by an example. It is different from the vicinity concept used in [38], which can lead to obtaining the data divide no metadata with the query, what does not correspond to our needs.

*2) Checking of the Query Realisability*

If the query criteria are in contradiction with their dependences extracted the database, it is known as unrealisable.

**Proposition:** Let a query $Q$ having the concept $Q=(Q_A,Q_B)$. A query $Q$ is unrealisable if and only if $\exists$ data source in $Q_A$ which dividing any metadata of the set $Q_A$.

*3) Generation of Top-K answers*

**Example**

$Q_B$ = {C2,C3,C5}: t**his Query** made intervene three clusters. **Cardinality (**$Q_B$) =3. Then, we must use the Table V. and we examine the rules describe in level 3;

***Extrac****t*(R,i) : is a procedure that determines answers of the request while using the *Backward chaining*. This procedure calls upon all the rules closely related to the request of level < = i. This is illustrated by Figure 5.

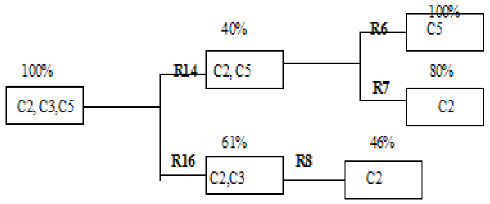


Figure 5. Backward chaining

In our example, we can calculate the satisfaction degrees of the various generated answers. These degrees are given in Table VII.

TABLE VII. SATISFACTION DEGREE OF THE GENERATED ANSWERS

| Data sources | Meta data | Satisfaction degree |
|---|---|---|
| {A1,A4} | {C2,C3,C5} | 100% |
| $\{A2, A6\}$ | {C2,C5} | 40% |
| $\{A5\}$ | {C2,C3} | 61% |

As show in Table VII, the result of the query is given to several levels according to a satisfaction degree measured compared to that initial one.

**A simple course of this table by order descending of the satisfaction degrees makes it possible to generate K better answers. Example for k=2 the K answers are {A1, A4}.**

## VI. EVALUATION OF THE PROPOSED APPROACH

Different advantages are granted by the proposed approach: (1) The definition of the Meta knowledge concept: This definition is in our opinion very important, since the number of rules generated is smaller. Besides, the concept of Meta knowledge is important to have a global view on the data set which is very voluminous. This models a certain abstraction of the data that is fundamental in the case of an enormous number of data. In this case, we define the set of meta-rules between the clusters. That can generate automatically the association rules between the data, if we want more details. (2) The definition of new approach to support top-k flexible querying using the generated knowledge in the first step. This approach allows the end-user to easily exploit all knowledge generated. (3) Extensibility of the proposed approach: Our approach can be applied with any fuzzy clustering algorithm to classify the initial data.

## VII. Conclusion and futures works

Knowing the essential goal of the extraction of knowledge is to help the user to seek information in this data set; in this paper, we propose a new approach to dealing with top-k flexible queries using Knowledge Discovery in large Databases (KDD). For this, we propose 1) an approach for KDD through the fusion of conceptual clustering, fuzzy logic and formal concept analysis, and 2) defining a new method to support top-k flexible querying using the generated knowledge in the first step. We prove that this approach is optimum sight that the evaluation of the query is not done on the set of starting data which are enormous but rather by using the set of knowledge on these data; what is to our opinion one of the principal's goal of KDD approaches.

As futures perspectives of this work, we mention 1) to test our approach on several the large data set, and 2) to define an incremental method that permits to deduct the Knowledge Base generated by our model knowing the modifications carried out in the initial data base.

## References

[1] B. Babcock and C. Olston, "Distributed top-k monitoring," SIGMOD Conf., 2003.

[2] P. Cao and Z. Wang, "Efficient top-k query calculation in distributed networks," PODC Conf., 2004.

[3] B. Kimelfeld and Y. Sagiv, " Finding and approximating top-k answers in keyword proximity search," PODS Conf., 2006.

[4] M. Wu, J. Xu, X. Tang and W-C Lee, "Monitoring top-k query in wireless sensor networks," ICDE Conf., 2006.

[5] S. Chaudhuri, L. Gravano and A. Marian, "Optimizing top-k selection queries over multimedia repositories," IEEE Trans. on Knowledge and Data Engineering 16(8), 2004.

[6] G.R. Hjaltason and H. Samet, "Index-driven similarity search in metric spaces," ACM Transactions on Database Systems (TODS), 28(4), 2003.

[7] R. Akbarinia, E. Pacitti and P. Valduriez, "Reducing network traffic in unstructured P2P systems using Top-k queries," Distributed and Parallel Databases 19(2), 2006.

[8] A. Metwally, D. Agrawal, A. El Abbadi, "An integrated efficient solution for computing frequent and top-k elements in data streams," J. ACM Transactions on Database Systems (TODS) 31(3), 2006.

[9] R. Fagin, J. Lotem and M. Naor, "Optimal aggregation algorithms for middleware," J. of Computer and System Sciences 66(4), 2003.

[10] S. Nepal and M.V. Ramakrishna, "Query processing issues in image (multimedia) databases," ICDE Conf., 1999.

[11] L. G. A. Marian and N. Bruno, "Evaluating Top-k Queries Over Web Accesible Sources," TODS 29(2), 2004.

[12] Y. chi Chang, L. Bergman, V. Castelli, C. Li, M. L. Lo, and J. Smith, "The Onion Technique: Indexing for Linear Optimization Queries," Proceedings of ACM SIGMOD, pp. 391–402, June 2000.

[13] P. Ciaccia, M. Patella, and P. Zezula. M-tree, "An Efficient Access Method for Similarity Search Metric Spaces," Proceedings of VLDB, pp. 426–435, Aug. 1997.

[14] L. Gravano and S. Chaudhuri, "Evaluating Top-k Selection Queries," Proceedings of VLDB, Aug. 1999.

[15] U. Guntzer, "Optimizing Multifeature Queries in Image Databases," VLDB, 2003.

[16] V. Hristidis, N. Koudas, and Y. Papakonstantinou, "Efficient Execution of Multiparametric Ranked Queries," Proceedings of SIGMOD, June 2001.

[17] A. E. I. Ilyas and W. Aref, "Supporting Top-k Queries in Relational Databases," VLDB J. 13(3), 2004.

[18] S. H. K. Chang, "Minimal Probing: Supporting Expensive Predicates for Top-k Queries," SIGMOD, 2002.

[19] L. G. N. Bruno and S. Chaudhuri, "Top-k Selection Queries Over Relational Databases: Mapping Strategies and Performance Evaluation," TODS 27(2), 2002.

[20] A. Natsev, Y.-C. Chang, J. Smith, C.-S. Li, and J. S. Vitter, "Supporting Incremental Join Queries on Ranked Inputs,"Proceedings of VLDB, Aug. 2001.

[21] P. Tsaparas, T. Palpanas, N. Koudas, and D. Srivastava, "Ranked Join Indicies," IEEE ICDE, Mar. 2003.

[22] M. Goebel and L. Gruenwald, "A Survey of Data Mining and Knowledge Discovery Software Tools," SIGKDD, ACM SIGKDD, Vol. 1, Issue 1 – June (1999) pp. 20-33.

[23] M. Zaki, "Mining Non-Redundant Association Rules, " Data Mining and Knowledge Discovery, No 9, (2004) pp. 223–248.

[24] G. Stumme, R.Taouil, Y. Bastide, N. Pasquier, L. Lakhal, "Intelligent structuring and reducing of association rules with formal concept analysis," Proceedings of KI'2001 Conference, Vienna, Austria, Lecture Notes in Artificial Intelligence 2174, Springer-Verlag, September (2001) pp. 335–350.

[25] R. Fagin, "Combining Fuzzy Information from Multiple Systems," PODS, pp. 216–226, June 1996.

[26] R. Fagin, "Fuzzy Queries In Multimedia Database Systems," PODS, pp. 1–10, June 1998.

[27] R.Fagin and E. Wimmers, "Incorporating User Preferences in Multimedia Queries," ICDT, pp. 247–261, Jan. 1997.

[28] R. Agrawal, T. Imielinski, A. Swami, "Mining Association Rules between sets of items in large Databases," Proceedings of the ACM SIGMOD Intl. Conference on Management of Data, Washington, USA, June (1993) pp. 207-216.

[29] R. Agrawal, R. Skirant, "Fast algoritms for mining association rules," In Proceedings of the 20th Int'l Conference on Very Large Databases, June (1994) pp. 478-499.

[30] N. Pasquier, Y. Bastide, R.Taouil, and L. Lakhal, " Efficient Mining of Association Rules Using Closed Itemset Lattices", Information Systems Journal, vol. 24, no 1, 1999, pp. 25-46.

[31] M. J. Zaki, and C. J. Hsiao, " CHARM : An Efficient Algorithm for Closed Itemset Mining ", Proceedings of the 2nd SIAM International Conference on Data Mining, Arlington, April 2002, pp. 34-43.

[32] G. Stumme, R.Taouil, Y. Bastide, N. Pasquier, and L. Lakhal, " Computing Iceberg Concept Lattices with TITANIC", J. on Knowledge and Data Engineering (KDE), vol. 2, no 42, 2002, pp. 189-222.

[33] T. Thanh, H.S. Cheung, C.Tru Hoang, "A Fuzzy FCA-based Approach to Conceptual Clustering for Automatic Generation of Concept Hierarchy on Uncertainty Data," CLA (2004) 1–12.

[34] A. Grissa Touzi, M. Sassi, H. Ounelli, "An innovative contribution to flexible query through the fusion of conceptual clustering, fuzzy logic, and formal concept analysis," International Journal of Computers and Their Applications. Vol. 16, N 4, December (2009) pp. 220-233.

[35] M. Sassi, A. Grissa Touzi, H. Ounelli, "Clustering Quality Evaluation based on Fuzzy FCA," 18th International Conference on Database and Expert Systems Applications, (DEXA'07), Regensburg, Germany, pp. 62-72, LNCS, Springer, 2007

[36] H. Sun, S.Wanga, Q. Jiangb, “FCM-Based Model Selection Algorithms for Determining the Number of Clusters,” Pattern Recognition 37, (2004) pp. 2027-2037.

[37] K. Chen, L. Liu, “Best K: critical clustering structures in categorical datasets,” Knowl. Inf. Syst. (KAIS) 20(1): pp. 1-33 (2009)

[38] M.H. Jamil and S. Fereidoon, “Recognizing Credible Experts in Inaccurate Databases,” Springer Berlin/Heidelberg, 869/1994: pp. 46-55, 2006.

# A Knowledge Management Methodology for Studying Health Science Students' Development of Misconceptions

Meaghen Regts, Arturo Fernandez Espinosa, Miguel Vargas Martin, Jayshiro Tashiro
University of Ontario Institute of Technology
Oshawa, Ontario, Canada
{meaghen.regts, arturo.fernandez, miguel.vargasmartin, jay.tashiro}@uoit.ca

***Abstract*- Development of misconceptions by health care students and providers can be deadly to patients during health care planning and delivery. However, ongoing research of methodologies for knowledge management and knowledge discovery has provided ways to study and then remediate misconceptions. Effective use of data warehousing coupled to automated data mining has been successfully integrated to around complex learning objects. Such integrations allow for monitoring and analysis of students' and practicing healthcare providers' processes of learning and decision making. In this paper, we describe how such coupling enables new types of methodologies which can lead to a better understanding of misconception development.**

***Keywords-complex correlations; misconception development; navigational pathways; pattern recognition process mining.***

## I. INTRODUCTION

The research literature does not provide a definitive understanding of how students and professional practitioners in healthcare develop misconceptions [1][2]. This deficiency in the literature is found across all academic and training levels. Yet, development of misconceptions are dangerous both to the individual who has a misconception as well as to others who may depend on that individual to use their knowledge and skills when making life and death decisions. For example, many health sciences students train to become nurses, doctors, and other allied health professionals. In their professional lives, these practitioners are frequently in situations requiring accurate pattern recognition of emerging signs and symptoms of a patient's disease-injury state and then make decisions about appropriate treatment for the patient.

Importantly, few evidence-based frameworks for education have been broadly deployed during the ongoing transformation of education in which there has been increased use of digital media and e-learning modalities [1-3]. Of course few evidence-based frameworks were deployed for education prior to this transformation. However, the enormous pressure on faculty to deploy e-learning solutions for courses and curricula, as well as increased use of digital media and simulations as teaching-learning resources, has resulted in a plethora of educational methods and materials that have little empirical foundation supporting their actual effectiveness in improving learning and knowledge transfer. We found no broadly-based, generalizable studies probing how and why misconceptions are formed. Consequently, we have both an opportunity and an obligation to study how misconceptions are developed and how to mitigate or at least minimize development of misconceptions that impair clinical judgment during health care planning and delivery.

Tashiro, *et al*. [1] studied the critical issues outlined by the US National Research Council. A more recent literature review identified critical gaps in our knowledge, some identified in the National Research Council analysis, but others not identified. We found the development of misconceptions was one of 10 gaps in knowledge about how educational materials "really work" to change an individual's learning outcomes and willingness as well as ability to sustain behaviours related to learning [1][2][4][5]: (1) How does an educational environment impact disposition to engage in a learning process? (2) What are the relationships between the level of realism in an educational environment and learning outcomes? (3) How do you define the threshold of experience within an educational environment that leads to measurable learning outcomes? (4) What are the knowledge domains being developed during learning? (5) In which knowledge domain is learning being retained and how stable is the retention? (6) What is the disposition to act on the knowledge gained during work within an educational environment? (7) How well can the knowledge be transferred? (8) What learning outcomes (conceptual and performance competencies) are developed during the learning process while working within an educational environment? (9) How are misconceptions developed during and sustained after working within an educational environment? (10) How do teacher-student and student-student social networks or e-communities impact learning. Interestingly, initial work indicated that development of misconceptions was the gap that was most difficult to study. However, as we conducted research on how to study misconception development, we began to test a methodology that allowed study of all 10 gaps. We report this newly developed methodology in this paper, using the study of misconceptions as a detailed example.

As a starting point, a study of diverse research literature bases and interviews with educators led us to pose the following definition for the dynamic condition of a "misconception" [1][5-7]: The state of being unaware that your own knowledge domains and cognitive processing are incomplete or incorrect." In this context, we are using knowledge domains and cognitive processing in the sense of Bloom's Revised Cognitive Taxonomy [1][5][6][8][9], wherein knowledge domains include facts, concepts, procedures, and metacognition, while cognitive processing includes remembering, understanding, applying, utilizing, evaluating, and creating. While some researchers might choose a different taxonomy, the key point is that whatever taxonomy might be chosen it can be used to build a knowledge management system to study how people learn. Furthermore, a knowledge management system based on a cognitive taxonomy can be coupled to a second knowledge management system that contains multiple frameworks, each framework based a theory of cognition and theory of behaviour change [11][15]. Our research team worked through the integration of these two knowledge management systems and coupling them to a knowledge discovery system to study how misconceptions are developed during a learning process. Together, these systems can be used to create new types of research platforms that could be used to study misconceptions in concept development and how such misconceptions could be expressed during knowledge transfer into behaviors expressed through knowledge-skills transfers based on prior learning [19].

The research platform can be conceptually described as a multi-tiered array of knowledge management and knowledge discovery systems.  A researcher engages at on tier and selects a theory of cognition and a theory of behavior change. At this tier, the knowledge management system algorithms select templates for instructional design consistent with the theories of cognition and behavior that were chosen by the researcher. The template is then mapped to learning activities, learning resources, learning assessments for each activity, and diagnostic feedback for each assessment.

The next tier of the research platform then loads the learning activities, resources, assessments, and diagnostic feedback into the template so that the template becomes a suite of learning objects that together comprise an educational environment. The educational environment can then be loaded into a Web-based stand-alone simulation or into a module nested within a learning management system. The educational environment then is integrated with a knowledge discovery system. The knowledge discovery system has two subsystems. First there is a subsystem comprised of monitoring and reporting algorithms that can track all of a student's' choices within the educational environments and how much time is spent in each choice of learning activity, resources, assessment, and diagnostic area loaded into that environment. The tracking-timing data are loaded to arrays collated by a set of reporting algorithms. The second subsystem then receives data arrays from the reporting algorithms and loads these arrays into data analysis options that can be selected by the researcher. Such data analysis options include a variety of data mining techniques.

These types of multi-tiered educational environments designed as research platforms can then be opened to access and be used by students in classes or healthcare providers as part of professional development training. Such research platforms are critical to development of empirical frameworks for studying what really works in education, a challenge first posed in 1997 by Tashiro and Rowland [20][21]. However, even as digital media uses within all levels of academia and training has increased dramatically [22], there still are few evidence-based practice frameworks for education.

Importantly, diverse learning environments based on digital media are becoming embedded in health care education and training at multiple levels ranging from all levels of academia, job training, and professional development and enhancement [1][2]. Do such educational digital media really work to improve learning and to reduce misconception development? In 2005, Aldrich stated that use of web-based learning as pedagogy has erupted so quickly it is comparable to the delivery of fast food. Just as fast food chains produce food that compromises nutritional value and increases health risks, educational interventions without substantive evidence for effectiveness may increase learners' development of misconception. Since Aldrich's paper was published, software companies and academic publishers have been flooding the educational and training markets with a variety of digital media and e-learning environments designed to serve diverse audiences in many disciplines. However, lack of a substantive empirical foundation for how effective such environments are in improving educational outcomes may actually lead to ineffective educational practices and dangerous misconceptions in health care education [23].

The digital media transformation of education still has too few well-developed frameworks for evidence-based learning that might provide guidance for development of educational methods and materials that ***really work*** to improve educational outcomes in health care education [1][2]. We now turn to describing how misconception development can be analyzed by the multi-tiered research platform we have described.

The remainder of this paper is organized as follows; Section II discusses the significance of this study while Section III describes our methodology. Results and discussion are provided in Section IV.

## II. SIGNIFICANCE OF STUDY

The significance and primary outcome of the research has been a methodology of building systems for knowledge discovery that could be applied to analyzing the cognitive and behavioural paths leading to misconception development in health science students and practitioners. In addition, we were able to evaluate specific types of research platforms in which we had embedded digital media and e-learning environments. Studies of these platforms and their capacity for collecting and analyzing data within teaching-learning environments led to sensible and logistically feasible capacities for improved data analysis of large educational data sets. Such data sets from our educational research platform allow automated data collection, management, database construction, and data analysis with appropriate data mining methods that had to be applied and evaluated.

Data mining and knowledge discovery have emerged as an important field of research with growing usage in educational research [24]. Data mining in education is an extension of the broader data mining research field. Common educational data mining applications include, but are not limited to, student models [29], educational software [23], collaborative learning environments [7], web logging [28], and factors associated with student development of misconceptions [1][7]. We have chosen to focus on the study of processes for collecting, managing, analyzing and interpreting data for investigating health science students and practitioner's development of misconceptions. Reduction of misconception development in healthcare education and professional development will improve patient care and safety.

## III. METHODOLOGY

The methodology developed to study processes of misconception formation evolved through four phases of research.

*Phase 1.—Teaching-Learning Simulative Environment*

The first tier of the research platform was simplified so that research tested a constructivist model consistent with one cognitive theory (Situative Learning Theory) and one theory of behavioural change (Planned Behaviour). This choice of a constructivist model was studied in the context of ways to improve healthcare education in the area of interprofessional care, which resulted in a template for interprofessional education instructional design. The template was then mapped to learning activities, learning resources for each activity, learning assessments, and diagnostic feedback for each assessment item.

The second tier of the research platform was then built as a virtual world with multiple healthcare settings, each containing a complex patient (funding from HealthForceOntario in Canada). Using the instructional design template and learning objet mapping, we then developed a simulative teaching-learning environment that had instructional design attributes based on prior evidence-based educational research [1-3][25]. This environment was called IPSims, an abbreviation for Interprofessional Simulations. As mentioned above, IPSims was developed to improve interprofessional care among healthcare students and providers and designed as a Web-delivered Flex-based set of patient simulations. The initial target audience was undergraduate health sciences students and the educational goal was to improve students' development and demonstration of skills in interprofessional care of complex patients.

IPSims was substantially derived from earlier work building and studying virtual hospital simulations for nursing students, a virtual medical office for medical assistant students, and virtual patient encounters simulations for paramedic training (funded by the United National Science Foundation and the United Sates National Institute of Nursing Research). As described by Tashiro and colleagues [25], the interprofessional care simulations were developed from a learning map of interprofessional core competencies that all health care providers should be able to demonstrate. The simulations were developed by teams of clinical experts, educational researchers, and software architects, with iterative review by expert panels. These experts provided evidence of construct validity in the patient cases and realism of the simulative teaching-learning environment of IPSims.

The usability of components of the simulations was studied within focus groups of Canadian health care providers. This research was part of a second grant funded by HealthForceOntario that supported a project for improving interprofessional care planning and delivery by health care providers in Southern Ontario, Canada. Components of the simulations were demonstrated in the situated context of how they might be used for interprofessional care education of health care providers. Project researchers recorded comments within a qualitative phenomenological approach. Information from these studies was passed to software architects as the simulations were built.

Phase 2.—Monitoring-Timing System

After three iterations of qualitative probes of usability and construct validity studies, a software architecture and engineering team created the third tier of the research platform. We integrated a set of algorithms that identified each place within the IPSims a student visited, and started a timer for that place. Consequently, we had place-time data that were extracted and saved to data arrays as a student navigated through IPSims and engaged in any or the "places" within the simulation, which were basically learning activities, learning resources, assessments, or feedback systems. This monitoring-timing set of algorithms was derived from a patent by Vargas Martin, Hung, and Tashiro, providing a systematic method for studying learning processes and outcomes of health care students and practicing professionals [19].

This set of algorithms was named PathFinder and was designed to expand our understanding of a student's navigational decisions and time spent in simulation compartments while they were immersed in the online learning environment. The coupling of the second tier knowledge management system of IPSims with the third tier monitoring-timing system provided an automated, systematic teaching-learning environment that could be monitored and timed a student's choices and engagements within the simulation, streaming data to a data warehouse to create a database of choices made by students and times of engagement with each choice. Such databases could then be called into data mining frameworks, e.g., to analyze possible associations with student learning outcomes. In particular, these types of data were tracks of students learning process and allowed us now study relationships between learning processes and learning outcomes indicative of misconceptions in knowledge or skills domains.

The model for this type of research platform evolved from analyses of earlier work on misconception development. These earlier studies led us to conclude that sequences of choices by students during learning and expression of skills based on learning were critically important to exploring the complex associations among decisions made while learning, learning outcomes, and expression of learning as conceptual and performance competencies. These earlier studies led to research focused on navigational pathways which we termed decisional sequelae. We believed the research literature supported the idea that decisional sequelae are the key traces of cognitive and behavioral choices leading to learning, including learning in ways that lead to the development of misconceptions.

The three tiers of the research platform provided us with the model system for studying health care students' development of misconceptions. This is possible due to the PathFinder tracking mechanism within IPSims of student navigational pathways or decisional sequelae. We feel these navigational patterns or choices are an expression of cognitive processes that lead to effective metacognitive pattern recognition that is imperative for knowledge transfer within the health care domain [1]. Student misconceptions can occur prior to engaging in the learning environment, while immersed in the learning environment, and after the student has completed the learning activities. Consequently, misconceptions can lead to erroneous concepts in conceptual and performance competencies [26]. We believe misconceptions are defined as a state of being unaware of cognitive and behavioural deficits. As previously stated, misconceptions developed by health care students can lead to negative patient outcomes and we felt strongly that there should be a sense of urgency and responsibility to implement research on the development of health care students' misconception development within teaching-learning environments like IPSims and other educational digital media learning environment.

*Phase 3.—Testing in Health Sciences Courses*

When the IPSims environment was completed, the patient simulations were introduced into courses within the Health Sciences Faculty of the University of Ontario Institute of Technology. Student usage patterns were evaluated by reviewing discussion board threaded comments as well as emails to the faculty member teaching the courses. While being principally a qualitative phenomenological study, the results indicated the simulations were easily accessible from the BlackBoard learning management system used for the courses and that the navigational schema and content were usable by diverse students. These students were part of the Bachelor of Applied Health Sciences program that provided a baccalaureate curriculum to practicing heath care providers who had been initially trained in two-year undergraduate college programs.

Phase 4.—Pre-processing Data for the Data Mining Analytical Tool

Although we could easily record decisional sequelae and time spent in any part of an educational environment, we still had to solve issues related to preparing the data for a database to apply data mining methods to our data from the IPSims. We simplify a bit here, but the basic model conceptualizes the learning environment as a multidimensional array of "places" in which a student can engage in learning activities, using learning resources, and take learning assessments that are coupled to diagnostic feedback arrays related to each assessment activity. Within the multidimensional array, navigation from a graphic user interface allows the student to "move" from place to place and in each place engage in some type of learning. The PathFinder engine follows students as they move from place to place and starts a time clock at the entry to each place. The clock is stopped when the student exits that place. Through time, PathFinder records the places visited, which becomes the decisional sequela for each student. That is, the decisional sequela is the sequence of places visited within the educational environment. With the clock function always coupled to the place, the decisional sequela of a place within the multidimensional educational environment has concomitant time spent in each place visited. These data are collected and stored in a database that captures multidimensional data arrays for each student that reflects that student's journey within an educational environment, such as IPSims. The data from the database is then configured, normalized, and transformed as necessary for the appropriate data mining methodology to which it has been coupled.

Consequently, we have been able to track the development of health science students' misconceptions by tracking their cognitive processes and behaviours within IPSims learning environment. With such data, we can ask fundamental questions related to misconception formation, such as, "What are the complex relationships between an individual students learning and skills outcomes and the processes they expressed while learning?" Such questions are suitable for analysis and interpretation by data mining techniques. In this current work, we look at Association Rule mining for our data analysis method. This choice is based on the types of data we have, the assumptions of the Association Rule model, and our need to delineate processes of cognitive processing and behaviours in the context of what students can demonstrate they have learned and how they transfer that knowledge.

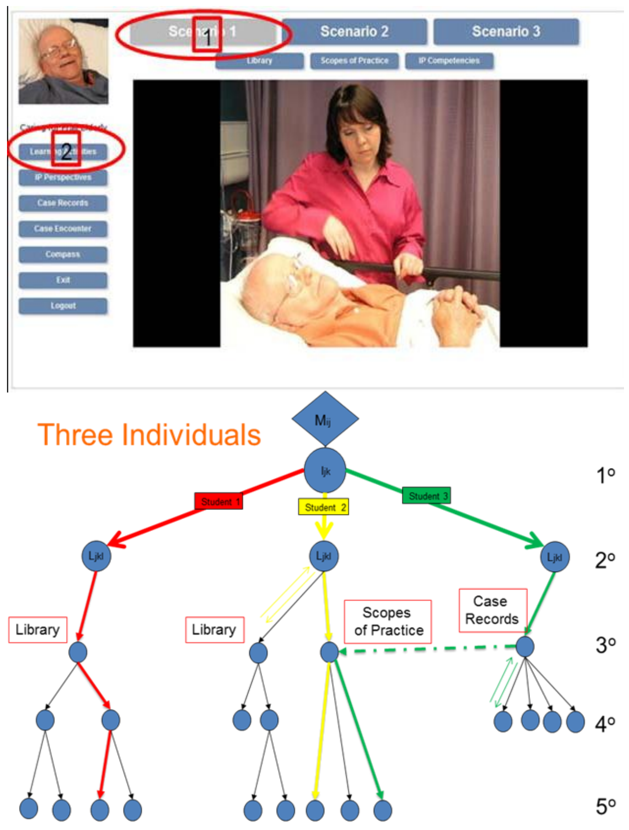
## IV. RESULTS AND DISCUSSION

We have now completed the first major study using our research platform,, with the first tier allowing choices of cognitive and behavioral theories to express instructional design templates, with the second tier being a virtual world for active engagement by the student, and the third tier being the monitoring-timing system we call PathFinder. The research platform was used to conduct a study at the University of Ontario Institute of Technology (UOIT). In this paper, we have focused on the methodology for building knowledge management systems into research platforms to study misconception development.

The IPSims-PathFinder research platform was built by an interdisciplinary collaboration of researchers. The first tier emerged from researchers conducting studies of theories of cognition and behavior change that looked at how instructional templates could be generated from grounded theory. The second tier emerged from a research team who built simulations based on instructional templates. This team included content experts who assembled case studies of complex patients and the realistic scenarios in which such patients would be found within heath care settings. Additionally, this team included educational researcher who took these scenarios and mapped them to the instructional design template for a particular combination of theories of cognition and behavioral change. Such mapping was based on a target audience of undergraduate Health Sciences students who would need the kind of interprofessional training for which IPSims was being designed. Together, the content experts and educational researchers created a learning map for the simulations that delineated the places within the virtual world that patients would exist at each moment in time as their simulation unfolded. Within each place, educators and researchers selected a set of learning activities, learning resources, learning assessments, and diagnostic feedback for students for each learning assessment. The Learning Map was then provided to a software design-engineering team, which had the responsibility of translating the Learning Maps into the designs of each immersive learning experience in the IPSims teaching-learning-assessment system. The designs were translated into immersive environments so that each environment becomes a functioning portion of Flex-based IPSims simulations. The design was prototyped and passed through usability tests. An important intermediate step was creating a prototype that allowed iterative use case analyses. Finally, the software design-engineering team integrated the PathFinder monitoring-timing system.

In Figure 1, we show an interface from the IPSims system. Again, IPSims was developed as the prototype for an immersive experience in which undergraduate health sciences students could develop and demonstrate competencies in interprofessional planning and delivery of health care services. The IPSims system shown in Figure 1 allowed researchers to express every pathway that a student might take in an immersive environment. In the case of IPSims, there were three scenarios developed by clinical experts to portray different facets for the care of each patient, with six patients in total. Figure 1 shows a graphic user interface (GUI) for a simulation of an elderly male patient. On the Figure 1 GUI, you can see three buttons on a top navigational bar Scenario1-Scenario3, which allow a researcher to quickly shift scenarios. In this case of elder care, the GUI shows Scenario1 selected, in which we can explore possible interprofessional health team interactions in a hospital emergency room after the man fell and injured himself. Scenario 2 portrays the elder man's in his home and the interactions during a home health visit as a follow-up to hospital discharge. Scenario 3 portrays a meeting in the elder man's home involving him, his daughter, and a mental health team. The GUI also shows a top navigational bar for Library, Scopes of Practice, and IP Competencies. These are Learning Resources that would be available within the immersive environment for every scenario of every immersive environment for improving competencies related to elder care in this simulation.

The left-side navigation bar in the GUI of Figure 1 provides interactions specific to a particular scenario. So, when the Scenario1 button is clicked on the top navigation bar, data are loaded so that the left-hand navigational bar buttons access data relevant only to the emergency room visit that was conceived as the immersive environment of Scenario1. The IPSims environment allows for iterations of use case analyses to probe how and why a student might make choices within an immersive teaching-learning-assessment environment. One example of a use case iteration is provided below in Figure 1. The use case of Figure 3 unfolds in the following manner.



1. Three students enter the teaching-learning-assessment environment. They have been assigned to complete the same learning activity in Scenario 1. Each selects Scenario 1, and selects the Learning Activity button on the left side of the GUI.

2. The Learning Activity functionality opens an instruction page. A sample page is provided below, along with an interprofessional care core competency.

Scenario 1: Emergency Room

**Competency 3.1.1 –** Respects complementary nature of health team members' scopes of practice.

Learning Activity 1 — You are providing care for an elder man who has fallen in his home and has been transported to the Emergency Department. Your colleagues include an Emergency Department triage nurse and physician, as well as a Geriatric Emergency Management RN. All of you are working together in the Emergency Room to care for this elder man. What should your colleagues' roles be in providing care for this patient? How are these roles complementary? In what ways do these practitioners fulfill their designated roles and in what ways do they not work within their scopes of practice?

3. Each student begins to work on this Learning Activity, but imagine that each of the three students chose a different pathway through the simulation. You can see in Figure 1 that we show a representation of how the three students could have taken different pathways within the IPSims teaching-learning assessment environment.

Indeed, you can see in Figure 1 that Student1 went to the library to review articles on interprofessional care. Student2 also went to the Library but then came back to the Learning Activity, and finally decides to go to the Scopes of Practice resource. Student3 decided first to view the Case Records, but then went back to the Learning Activity and subsequently decided to examine the Scopes of Practice Resource, but went to a different area of that resource than Student2.

The key point of this methodology is that at each of the "places" visited by the student, PathFinder records the location and starts a clock. The clock times the student's presence within the location and when the student navigates out of the location, the time is recorded for that moment of the student's journey through the IPSims learning environment. Figure 1 shows the decisional sequela of each student, that is, the place to place journey within the IPSims environment, and along that decisional sequela is the time spent in each location of the journey. However, the usage of term location needs to be reined here. Some locations are just like a real hospital room in the sense that a student can engage in many interactions with a patient, equipment, electronic health records, and so on.

Figure 1 is a representation of three students' possible excursions into the IPSims learning environment. In this figure, the paths of three different students are described, showing the choices at each level of navigation (1st, 2nd, and so on).

In our usage of "location", we are evoking the multidimensional sense of the database structure mentioned earlier. That is, a student within IPSims can go to locations that are analogous to a room. Within that room, there are locations that allow access too many types of interactions, such as accessing and using as-needed teaching and learning resources. So, Student1 went to the library to review articles on interprofessional care. Student2 also went to the Library but returned to the Learning Activity location before navigating to the Scopes of Practice resource. In contrast, Student3 decided viewed the Case Records, next went back to the Learning Activity, then navigated Scopes of Practice resource, but went into a different area of that resource than Student2. The term "location" has a variety of different meaning. Some locations are analogous to a physical place (e.g., Library, Patient's Room for a Case Encounter, a Web-link resource within the Scopes of Practice functionality. Other locations are analogous to an activity, such as reading a paper in the Library, exploring a Web-site for Scopes of Practice of different types of clinicians, reading one of the different types of patient records within the Case Records functionality.

Consequently, we had to explore how the decisional sequela and time data for each student could be contextualized in semantic and workflow meaning. For example, if a student navigated to the Scopes of Practice location, they would find Web-links to over a hundred different types of health care providers and for each type of provider a description of their respective scope of practice. So PathFinder would record accessing the Scopes of Practice portal and start a clock. If the student accessed a Web-link that location would be recorded and another clock would start. This would be true for any Web-link accessed with the Scopes of Practice location, with each Web-link accessed becoming a new location with a new clock. These types of analyses of students' journeys and work convinced us that the locations within the teaching-learning-assessment environment could be conceptualized as locations within a cognitive process. So, the term "location" was not particularly problematic, despite its usual implication for physical space or something analogous to physical space.

We continued studying the methodology for building and using the IPSims-Pathfinder combined education and research system. Although the delineation of a user's decisional sequelae and associated time coding were robust, the research literature suggests there are other types of data that may be critical to understanding the complexities of cognitive processing and behavioural choices. Therefore, we began building into the IPSims environment a variety of data collection pathways that could complement the data representation of a student's journey through the learning environment of IPSims.

One of our studies built in a data collection process that would provide a student user profile. This profile was a set of variables, with variable values collected when a student entered the IPSims environment for the first time. These variables were taken from the research literature as possible covariates shaping learning processes. Our choice of variables included, age, gender, prior academic knowledge, perceived academic success, discipline major, academic class, and a cluster of variables assessing digital literacy.

A second study examined choices of variables that research literature suggests are important in shaping cognitive processes and behavioural choices. These included variable clusters for measuring disposition to engage in higher order reasoning, formal operational reasoning, and value placed on a learning experience, expectations for success in a learning experience, perceptions of and preferences for digital learning objects and online learning modalities, and perceptions of usability of learning objects and simulations with the different situated experiences of a virtual world.

These additional data are now being integrated into the database for automated data collection systems of the coupled IPSims-PathFinder teaching-learning-assessment system. The methodology described in this paper provides a means to study misconception development. We recently completed a study of 67 students in the Health Sciences Faculty of UOIT. A subsequent paper will describe the methodology in action and the detailed analysis of how misconceptions development can be studied and remedied.

## V. Conclusions and future work

In conclusion, we have developed a methodology that allows us to study learning processes as a sequence of decisions an time spent within learning activities, learning resources, leaning outcomes assessments, and diagnostic feedback for any assessment item. Such activities, resources, and so on, are “places” within an educational environment. We have called these earning processes decisional sequelae in the context of how data related to the processes are collected and analyzed as decisions made by a student to engage and spend time at certain “places” within a learning environment. Importantly, such decisional sequelae now can provide an opportunity to study students’ development of misconceptions. In a subsequent paper we will provide the data analyses of the first set of student research subjects who were undergraduates in the Health Sciences Faculty at UOIT. This next paper will show the efficacy of the methodology for identifying learning processes as decisional sequelae that lead to misconception development.

However, our ongoing work using knowledge management systems to study misconceptions is part of a larger research program to build and evaluate an inclusive and adaptive teaching-learning-assessment-diagnostic system in which we can embed a diverse array of educational environments. Certainly, we believe in the importance of identifying decisions and patterns of learning engagement that lead to misconception development. Nevertheless, difficulties studying misconception development is only part of a larger set of barriers to transforming education into a more evidence-based praxis. Our ongoing research has become increasingly focused on creating human-computer interfaces that are truly inclusive in the way these interfaces engage learners, probe and adjust to their accessibility needs, and identify their preferences for learning. Our model of these interfaces will create a learner profile that can then be used to adapt a computer-based learning environment dynamically in order to meet the learning attributes of the respective learner’s profile. Knowledge management systems can be used to configure a teaching-learning-assessment-diagnostic educational environment based on the learner profile. The methodology for studying student outcomes, including misconception development, can then be integrated into this educational environment.

Furthermore, we already have developed meta-level knowledge management systems that can be used to select theories of cognition and behavioral change, along with their respective associated instructional design templates. These templates can be integrated with the inclusive-adaptive interface to allow instructors or researchers to select combinations of theories of cognition and behavior change. Consequently, a learner’s profile can then be mapped to instructional templates based on a particular combination of theories and the teaching-learning-assessment-diagnostic environment configured to be consistent with that theory combination. The result is a truly inclusive-adaptive educational environment based on grounded theory and nested within a research platform. Such research platforms will allow cross-theory studies that will advance our understanding of how to build evidence-based frameworks for educational methods and materials.

## Acknowledgment

This work was supported in part by the Social Sciences and Humanities Research Council of Canada.

## References

[1] Tashiro, J., Hung, P., and Vargas Martin, M.: Evidence-Based Educational Practices and a Theoretical Framework for Hybrid Learning. In R. Kwan et al. (eds.): ICHL 2011, LNCS 6837, pp. 51–72, 2011.Springer-Verlag Berlin Heidelberg (2011)

[2] Tashiro, J., Hung, P. C. K., and Vargas Martin, M.: ROAD-MAP for Educational Simulations and Serious Games. In Tsang, P., Cheung, S. K. S, Lee, C. S. K., Huang, R. (eds.) Hybrid Learning – Proceedings of The Third International Conference ICHL, pp. 186-204. Beijing China (August 2010)

[3] Garcia-Ruiz, M.A., Tashiro, J., Kapralos, B., and Vargas Martin, M.: Crouching Tangents, Hidden Danger: Assessing Development of Dangerous Misconceptions Within Serious Games For Health care Education. In Hai-Jew, S. (ed.). Virtual Immersive and 3D Learning Spaces: Emerging Technologies and Trends, pp. 269-306. IGI Global, Hershey (2011)

[4] Federation of American Scientists: Summit on Educational Games – Harnessing the power of Video Games for Learning. Federation of American Scientists, Washington, DC (2006)

[5] Patel, V.L., Yoskowitz, N. A., Arocha, J. F., and Shortliffe, E. H.: Cognitive and learning sciences in biomedical and health instructional design: A Review with Lessons for Biomedical Informatics Education. Journal of Biomedical Informatics, 42, 176-197 (2009)

[6] Anderson, L. and Krathwohl, D.: A Taxonomy For Learning, Teaching, and Assessing. Longman, New York (2001)

[7] Baker, R. and Yacef, K.: The State of Educational Data Mining in 2009: A Review and Future Visions (2009)

[8] Michael, J.A., Richardson, D., Rovick, A., Modell, H., Bruce, D., Horwitz, B., Hudson, M., Silverthorn, D., Whitescarver, S., and Williams, S. Undergraduate students' misconceptions about respiratory physiology. *Advances in Physiology Education*, 22(1), 127-135. (1999).

[9] Balkissoon, R., Blossfield, K., Salud, L., Ford, D., and Pugh, C. Lost in translation: Unfolding medical students' misconceptions of how to perform a clinical digital rectal examination. *The American Journal of Surgery*, 197, 525-532. (2009).

[10] Sorden, S.D.: A Cognitive Approach to Instructional Design for Multimedia Learning. Informing Science Journal, 8, 263-279 (2005)

[11] Mayer, R.E., Fennell, S., Farmer, L., and Campbell, J.: A Personalization Effect in Multimedia Learning: Students Learn Better When Words Are in Conversational Style Rather Than Formal Style. Journal of Educational Psychology, 96(2), 389-395 (2004)

[12] Martin, L., Haskard-Zolnierek, K., and DiMatteo, M.R.: Health Behavior Change and Treatment Adherence: Evidence-based Guidelines for Improving Health Care. Oxford University Press, New York (2010)

[13] Prochaska, J.O., Redding, C.A., and Evers, K.E.: The Transtheoretical Model and stages of change. In Glanz K., Rimer B., Viswanath K. (eds.) Health Behavior and health Education, pp. 97-121. Jossey-Bass, San Francisco (2008)

[14] DiMatteo, M.R., & DiNicola, D.: Achieving Patient Compliance. Pergamon Press, New York (1982)

[15] Fishbein, M., Hennessey, M., Kamb, M., Bolan, G., Hoxworth, T., Latesta, M., et al.: Using Intervention Therapy to Model Factors Influencing Behavior Change: Project RESPECT. *Evaluation and Health Professions*, 24(4), 363-384 (2001).

[16] Bensley, R., Mercer, N., Brusk, J., Underhile, R., Rivas, J., Anderson, J., Kelleher, D., and Lupella, M, de Jager, A.C..: The e-Health Behavior Management Model: A Stage-based Approach to Behavior Change and Management. *Preventing Chronic Disease*, 1(4), A14. (2004)

[17] Baranowski, T., Cullen, K., Nicklas, T., Thompson, D., and Baranowski, J.: Are Current Health Behavioral Change Models Helpful in Guiding Prevention of Weight Gain Efforts? *Obesity Research (Suppl.)*, 23S-43S (2003)

[18] Leventhal, H., Halm, E., Horowitz, C., Leventhal, E., and Ozakinci, G.: Living with chronic Illness: A Contextualized, Self-Regulation Approach, In Sutton, S, Baum, A. and Johnston, M. (eds.) Handbook of Health Psychology, pp. 159-194. Sage Publications, London, (2004)

[19] Vargas Martin, M., Hung, P.C.K., and Tashiro, J.: Method for Competency Assessment of Health care Students and Practitioners. U.S. Patent (pending) No. 61/162,597. U.S. Patent and Trademark Office, Washington, DC: (2009)

[20] Tashiro, J.S. and Rowland, P.M..: What Works: Empirical approaches to restructuring courses in biology and environmental sciences. In McNeal A, D'Avanzo, C. (eds.), Student Active Science – Models of Innovation in College Science Teaching, pp. 163--187. Harcourt, Brace, & Company, New York (1997)

[21] Tashiro, J.: Ethical Analysis of Publisher and Faculty Roles in Building and Using Electronic Educational Products. Journal of Electronic Commerce in Organizations, 7(1), 1-17 (2009)

[22] Rudak, L. and Sidor, D. Taxonomy of E-courses. In Islander, M., Kapila, V., Karim, M. A. (eds.) Technological Developments in Education and Automation, pp. 275-280. Springer Science and Budiness Media, New York (2010).

[23] Aldrich, C. *Learning by doing.* San Francisco, CA: Pfeiffer. (2005).

[24] Cios, K. and Kurgan, L.: Trends in Data Mining and Knowledge Discovery (2009)

[25] Tashiro, J., Byrne, C., Kitchen, L., Vogel, E., and Bianco, C.: The Development of Competencies in Interprofessional Health care for Use in Health Sciences Educational Programs. Journal of Research in Interprofessional Practice and Education, 2.1, 1-20. (2011)

[26] Ozmen, H. Some student misconceptions in chemistry: A literature review of chemical bonding. *Journal of Science Education and Technology*, 13(2), 147-159. (2004).

[27] Berthold, M.R., Borgelt, C., Hoppner, F., and Klawonn, F. Guide to Intelligent Data Analysis: How to Intelligently Make Sense of Real Data. Springer-Verlag, London. (2010). DOI: 10.1007/978-1-84882-260-3_4

[28] Romero, C. and Ventura, S.  ( 2010).  Educational Data Mining: A Review of the State of the Art.  IEEE transactions on Systems, Man, and cybernetics-Part C:  Applications and Reviews.  40(6) November.  DOI: 10.1109?TSMCC.2010.2053532

[29] Scheuer, O. and McLaren, B. 2011. Educational Data Mining: Encyclopaedia of the Sciences of Learning, Springer.

# Prediction Model Based on User Profile and Partial Course Progress for a Digital Media Learning Environment

Arturo Fernandez Espinosa Meaghen Regts Jayshiro Tashiro Miguel Vargas Martin
University of Ontario Institute of Technology
{arturo.fernandez, meaghen.regts, jay.tashiro, miguel.vargasmartin}@uoit.ca
Oshawa, Canada

*Abstract*—**This work in progress reports on the implementation of a data mining system aimed at predicting the final GPA of healthcare students within the context of our learning environment called IPSims. The predictions are based on the user profile, and their choices and preferences within IPSims. The intended predictions will use various well-known and widely used IT technologies and data mining techniques combined in such a way to overcome the complexity of IPSims and its associated disaggregation variables. The present work shows the application of well-known data mining techniques within the environment called IPSims.**

***Keywords—Databases; Data Mining; Education, Knowledge Discovery; Digital Media Learning Environment.***

## I. INTRODUCTION

Knowledge discovery is an iterative process that involves different stages of information processing. Fu [1] and Luo [2] divide the knowledge discovery process into five stages: selection, preprocessing, transformation, data mining, and evaluation. The final result of applying these stages should be the acquisition of knowledge.

The purpose of this work is to test the following hypotheses:

1) There is an identifiable set of variables in the student profiles which have the highest impact in students' success, 2) It is possible to predict, with high level of accuracy, the performance of a student at the beginning of a course, based on their profile, and 3) It is possible to predict student performance based on their profile and their history of choices and preferences in the system.

The prediction of the student performances (usually reflected by their grades) may help teaching environments of all levels. Since elementary school to postsecondary education, the detection of students with poor performance is not possible until the final grade is known and it is already too late to take a preventative action.

The main problem of this research is to apply well-known data mining techniques in an efficient way in order to obtain useful information from IPSims [3], which is a digital media-learning environment for health sciences students.

In order to apply data mining techniques, there are different issues to address such as the normalization of a non-normalized database, inconsistency in some student records (registers of the database), the transformation of multiple qualitative variables into quantitative values, lost information, data integration from multiple data sources, outliers for some variable values, the efficiency of the prediction algorithms and cluster techniques, and a reduced number of real cases to train and test the system.

The data integration, data reduction and data transformation steps have been already accomplished. The statistical analysis of the variables in order to detect outliers and understand the behaviour of the information is the next step on this work as well the implementation of the neural network for prediction purposes and cluster techniques for pattern recognition that will surface the variables with greatest impact in the final grade of the students.

The rest of the paper is organized as follows. Section II is the related literature. A description of IPSims is given in Section III. In Section IV, the techniques applied in order to retrieve knowledge from IPSims are reviewed. Section V presents the concluding remarks, final notes and future work.

## II. RELATED WORK

The idea of using neural networks as predictive models has been proposed by multiple authors. Bargallo [4] makes a study of comparison for different neural network architectures for prediction of radio frequency propagation loss this methodology. The architectures proposed are: Multilayer perceptron networks and radial basis networks. In Wang and Yu [5], once again, neural networks are being use to predict. On this specific case the prediction is over software quality. Vilovic and Burum [6] present a comparison study of different neural network models and the accuracy of them on the prediction of indoor field strength.

The previous works lacks a description of one characteristic that is being analyzed in this paper; this is the measure of accuracy and model optimization over time periods. The previous group of papers applied the model that fits best in general for a given phenomena.

The current work describes the specific study case in function of discretized units of time. The phenomena are divided and the model is adequate accordingly in function of this discretize time units. A more detailed description will be given later in the paper.

The related literature shows a common denominator: Neural networks are black boxes. This makes it difficult to know if the results are going to be good enough to support our hypotheses.

The way these papers evaluate the efficiency of each neural network is not known until the testing in real cases is finished.

## III. IPSIMS DESCRIPTION

### A. General Description

IPSims system is designed to help students enhance their learning experience through the effective use of a simulation environment supported by multimedia materials.

IPSims is able to track the activity of users. The traces include time spent on each module or learning activity as well as the path followed within IPSims.

First-time users get registered in the system as they enter a number of disaggregation variables, which include: gender, course, faculty, term, age, undergraduate academic year, undergraduate major, number of hours/week surfing the web, number of hours/week playing video games, how they rate their computer literacy, likelihood to choose a career in computer sciences, interest in course material, experience with computer-based simulations, their perceived educational value of computer-based simulations, expected grade in the course, and current GPA.

These variables are stored in a secure database that is accessed at a later time and used in the knowledge discovery process.

Returning users logon with a user name and password. Upon successfully logging in, IPSims presents a menu of six simulation environments, each consisting of a different healthcare scenario and involving multiple learning activities.

Each of these six simulations contains three scenarios, a library with web links and scholarly journals, scopes of practice, interprofessional competencies, interprofessional perspectives, case records, case encounter (video), main menu, logout, and bookmark links.

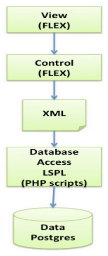


Figure 1. General architecture of IPSims and data recording into the database.

### B. Architecture

The user interface of IPSims is constructed using FLEX technology [7]. FLEX offers an elegant and attractive design environment for user interfaces.

In general, the structure of the IPSims system is as follows, LSPL is a set of PHP scripts that are responsible for the interaction between the XML files and the database as these scripts record activities and perform data retrieval.

Figure 1 describes IPSims architecture in a simplified form. LSPL counts with a query module where we can consult and extract the information related to time stamps and path tracking (but by now and for our research purposes this paper will focus just in the recording functionality of the system).

## IV. KNOWLEDGE DISCOVERY IN THE IPSIMS DATABASE

To fully exploit the potential of the IPSims database it is necessary to create appropriate mechanisms that allow for information retrieval.

The database contains data from three sources: 1) the user profile, 2) users' choices and preferences in IPSims, and 3) student surveys collected after completion of all the learning activities performed in IPSims. The student surveys include nine different sets of questions, namely 1) learning assessment based on assigned learning activities, 2) demographic information, 3) preferences for learning resources and educational scaffolding, 4) rating of web-based course work, 5) rating of IPSims learning environment, 6) satisfaction with educational simulations and serious games, 7) disposition to engage in effortful cognitive endeavor (need for cognition and ambiguity tolerance), 8) expectancy-value questionnaire, and 9) performance evaluation in interprofessional learning activities.

Before the system can exploit these three sources of data, algorithms that perform preprocessing and transformation steps are required; as well as a neural network model will be applied in order to predict the final GPA of the students.

### A. Data Preprocessing and Transformation

It is convenient to firstly analyze the user profile. The user profile is automatically recorded in our database, and the validation of the web registration form is carefully designed to avoid inconsistencies in the data provided by the students.

The qualitative variables in the user profiles require a transformation into quantitative values. For our approach it is desirable to normalize our input set of values into the range [0-1].

Referring back to the question of how to treat the qualitative variables, Driscol et al. [8] propose multiple methods that could be used. One of the most common methods is to count the number of times a qualitative code occurs for each user. It will be interesting to see how the two-three different techniques mentioned in [8] like substitute the values for descendent or ascendant values, or frequency of each value repetition will impact the model accuracy.

The next step to achieve a high-quality input set is the treatment of the timestamps and path tracking for the users. The timestamp for each section has to have a special treatment so that every time the user starts a session the system starts recording time stamps for each web document visited for the user. For example, in a situation where a given user $u_x$ that visits the web document $d_a$ $n$ times, then we will have $n$ time stamps for that given section of the system. The obvious solution in this situation is to merge these times into one total time $T$ for the document $d_a$ given a user $u_x$ , denoted by $T(u_x, d_a)$, the relative times to each session for a given user $u_x$ at a document $d_a$ is given by the expression $t_{\downarrow}i \mid i \in Z^{\uparrow}+$ ,where $i$ denotes the number of session.

$$T(u_x, d_a) = t_0(u_x, d_a) + t_1(u_x, d_a) + t_2(u_x, d_a) + \cdots + t_n(u_x, d_a); \quad (1)$$

Therefore, at the end, there will be just one variable for the time stamp for the web document $d_a$ that will prevent the fast growing of information and will fix the input vector in a specific dimension.

Imagine a situation where a user X visits seven web documents and user Y visits just five, then there will be a dimensionality inconsistency on the vectors, even worse if the documents they visited are partially or completely different. The first solution that comes to mind is to set up the input vector to a defined dimensionality, and fill these n-elements with zeros, and for every available variable fill the element in the vector with the corresponding variable.

Supposing that $u_x$ and $u_y$ define different users, and $d_i; i = 0,1, \dots n;$ define a visited document. Then the vector for each user will be defined by V($u_x$) and V($u_y$) and will be composed as follows:

$$V(u_x) = \begin{bmatrix} User\ profile\ var1 \\ User\ profile\ var\ 2 \\ \vdots \\ User\ profile\ varn \\ T(u_x, d_0) \\ T(u_x, d_1) \\ T(u_x, d_2) \\ \vdots \\ T(u_x, d_n) \end{bmatrix}$$

$$V(u_y) = \begin{bmatrix} User\ profile\ var1 \\ User\ profile\ var2 \\ \vdots \\ User\ profile\ varn \\ T(u_y, d_0) = 0; \\ T(u_y, d_1) \\ T(u_y, d_2) \\ \vdots \\ T(u_y, d_n) \end{bmatrix}$$

$T(u_y, d_0) = 0$ indicates that the user $u_y$ has not visited that document yet.

The path tracking functionality of IPSims is being wasted in this approach, but by the moment and for prediction purposes the model will be based just in the time stamps, user profile and paper survey information.

The last step in our input data preprocessing requires the capture of all the variables in the paper survey; the algorithms applied for preprocessing will be the same algorithms mentioned before for the qualitative, and the quantitative variables. This step will require the addition of new columns or tables to our database.

Figure 2 generalizes the process described before. We can observe how the integration of the three different data sources is accomplished into a one single data source.

## B. *Neural Network Model*

The next step in this framework is the model construction. Different neural network architectures will be used in order to compare different results and determine which architecture is most suitable for the data set.

An optimistic approach is being adopted when the idea that a neural network with *n* inputs and *m* outputs will be enough to solve the linear separability problem for the different groups of students.

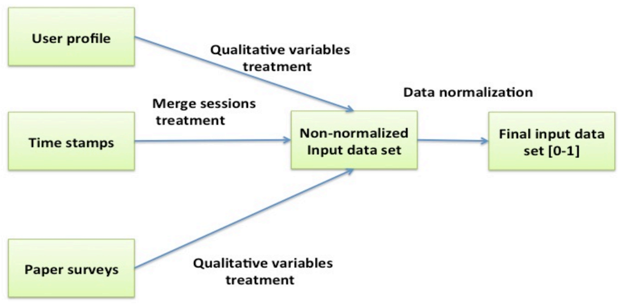

User profile
Qualitative variables treatment
Data normalization
Merge sessions treatment
Time stamps
Non-normalized Input data set
Final input data set [0-1]
Paper surveys
Qualitative variables treatment

Figure 2. Data preprocessing and transformation processes before reaching the final input data set for the predictive model.

The justification for the selection of the model will be determined by the observations in the cluster analysis, the statistical analysis and distribution of the variables as well as the testing results. However, if the linear separability problem cannot be solved for the dataset with neural networks then the use of a support vector machine (SVM) will be required. This has been left for future work.

If the results with the neural network model are acceptable, then a comparison with a SVM model will be interesting in order to determine the tool with the best performance for the given study case.

Our IPSims already has around 70 real student registers with the variables and the final learning outcome. This set of cases will populate the training and testing set with data that can validate the accuracy of the results.

From 70 real cases available, around 75% are going to be used for training purposes and 25% for the testing stage of the model. This is usually the proportion for training and testing proposed in most of the literature.

The last stage in the framework is the refinement of the model as presented in Figure 3. This step will be accomplished with the comparison of results of the different proposed architectures (feed-forward propagation and backtracking) with different numbers of neurons in the input layer, and different activation functions between the neurons. In the experiments, a table with the results will be the one that allow determining, which one is the model with the highest accuracy.

TABLE I. A SAMPLE OF THE MODEL REFINEMENT.

| No. of Neurons | Activation Function | Feed-forward propagation | Backtracking |
|---|---|---|---|
| W | Sigmoid | 67.5% | 78% |
| X | Sigmoid | 70% | 82% |
| Y | Step | 45% | 29% |
| Z | Signum | 38% | 34% |

### C. *Model application*

Once the selection of the final model is done, the system will attempt to predict the final learning outcome of IPSims users at any time during the course. The specific time in which the model is applied to the dataset will show a prediction for that specific time; therefore, still there is the question how accurate the system is when is applied at different given times? This is probably the most challenging question to solve in this work. Right now the idea to approach this problem is to divide the course into time periods, and according to these periods modify the weighted edges that go from the input vector to the first hidden layer in the neural network.

In the density function given by Eq. (2), an example of how to generate a dynamic weighted model for the neural network model is observable. This density function will depend on the time the model is applied; this time will be determined for discrete divisions of the given course, in the example shown the course is divided in three sections. Depending on which period of time the model is applied is the weight for every specific edge.

$$W_{xy}(t) = \begin{cases} 0 \leq W_{xy} \leq A \\ A < W_{xy} \leq B; \\ B < W_{xy} \leq 1 \end{cases} \qquad (2)$$
$$\forall x, y \ \ni W and\ 0 \leq A, B \leq 1$$

## V. CONCLUDING REMARKS

This work aimed at exploiting all the capabilities of the IPSims system. This continuing work is part of a larger project using the IPSims system for healthcare education. High impact variables identification (cluster analysis), predictive models (neural networks and support vector machine), behaviour prediction (neural networks and support vector machines), and uncertain data are part of the projects that will involve the use of IPSims. We expect to get acceptable results when measuring the accuracy of the model against real cases; however, we have as future work the use of a support vector machine. An ambitious approach on this work will be the generation of a framework that could be applied to virtual learning environments of similar nature to that of IPSims. In the future, we expect not only be able to predict outcomes, but to provide feedback and advice to users as to how they can improve their learning performance to maximize their results in a particular course. We are currently redesigning the IPSims database architecture to integrate the three data sources discussed in this paper, as well as the scripts required for the experiments related to data preprocessing and transformation.

## AKNOWLEDGEMENTS

This work was supported in part by the Social Sciences and Humanities Research Council of Canada.

## REFERENCES

[1] Y. Fu, "Data mining Tasks, techniques and applications," 1997 19th International Conference on Data Engineering. IEEE Potentials, 18-20. Doi:10.1109/ICDE.2003.1260873

[2] Q. Luo, "Advancing Knowledge Discovery and Data Mining," *2008* International Conference on Computer Science and Software Engineering, 7-9. doi:10.1109/WKDD.2008.153

[3] IPSims:http://199.212.33.78/LPSL_V2_040610/Main.html [Last accesed: December 17 2011].

[4] J. Bargallo, "A comparison of neural network models for prediction of RF propagation loss," 48th IEEE Vehicular technology conference. Pathway to global wireless revolution, 445-449.

[5] Q. Wang and B. Yu, "Extract rules from software quality prediction model based on neural network. 16th IEEE International conference on tools with artificial intelligence. 191-195

[6] I. Vilovic and N. Burum, A comparison of neural network models for indoor field strength prediction. Elmar 2007. 251-254.

[7] Adobe Developer Connection. Flex Developer Center. Available: http://www.adobe.com/devnet/flex.html [Last accesed: December 12 2011 ].

[8] D. L. Driscoll, A. Appiah-Yeboah, P. Salib, and D. J. Rupert, "Merging qualitative and quantitative data in mixed methods research: How to and why not. Ecological and Environmental Anthropology," (University of Georgia*)*, *3*(1), 18.

# Clustering Large-Scale, Distributed Software Component Repositories

Marcos Paulo Paixão, Leila Silva
Computing Department
Federal University of Sergipe
São Cristóvão, SE, Brazil
marcospsp@dcomp.ufs.br, leila@ufs.br

Gledson Elias
Informatics Department
Federal University of Paraíba
João Pessoa, PB, Brazil
gledson@di.ufpb.br

***Abstract* — In software component repositories, search engines have to deal with challenges related to storage space requirements for indexing semi-structured data models, which are adopted for representing syntactic and semantic features of software assets. In such a context, clustering techniques seem to be attractive for reducing the number of assets in a repository, and so, the size of index files. Accordingly, this paper proposes and evaluates a distributed clustering approach for large-scale, distributed software component repositories. Based on experiments, outcomes indicate relevant gains in storage space requirements for index files.**

***Keywords-clustering techniques, indexing techniques, search engines, software component repositories;***

## I. INTRODUCTION

Software component repositories have to handle metadata for describing stored software assets, providing information employed by search engines for indexing them [1]. As endorsed by Vitharana [2], component description models can adopt high level concepts for describing component metadata, making possible to express syntactic and semantic features, and so, facilitating developers to search, select and retrieve assets. In practice, currently available component description models [3][4], have adopted approaches based on semi-structured data, more specifically XML, allowing structural relationships among elements to aggregate semantic to textual values.

Several proposals exist for indexing semi-structured data [5][6][7]. Despite their contributions, existing techniques still suffer from problems related to storage space requirements, processing time and precision level of queries. For instance, Brito et al. [7] proposes an indexing technique based on semi-structured data, which is precise and efficient in terms of query processing time, but suffer from problems related to storage space requirements for index files. Thus, in the context of large-scale software component repositories, it is a challenge to design indexing techniques that minimize storage space requirements, but without excessively impacting on query processing time and precision level.

In such a context, an interesting insight for optimizing the storage space required by index files is to construct a clustered repository, in which clusters (groups) of similar software assets are identified by applying a clustering heuristic, according to defined similarity criteria. Then, representative assets of the clusters can be generated for defining the clustered repository. Thus, only the reduced set of representative assets in the clustered repository needs to be indexed, instead of all assets in the original repository. As a result, storage space requirements can be notably reduced.

Motivated by such an insight, in [8], we have already proposed a clustering approach for X-ARM based software component repositories, which constructs a clustered repository, reducing the number of assets, and so, optimizing the storage space requirements. Despite the excellent gains in storage space requirements, the clustering approach proposed in [8] operates in local and centralized repositories only.

Based on Seacord [9], local and centralized repositories lead to limited accessibility and scalability. Besides, according to Frakes [10], a large quantity and variety of components can increase software reuse. Thus, scalability issues associated with centralized repositories obviously also imply in low reusability levels. As a consequence, large-scale, distributed software component repositories have been proposed as an infrastructure for minimizing problems related to limited accessibility, scalability and also reusability provided by centralized repositories [11][12].

Accordingly, herein, we have evolved the centralized clustering approach, already proposed in [8], to a distributed clustering approach for large-scale, distributed software component repositories, defined as a collection of connected, integrated storage units that reside throughout independent nodes of the repository in a network like the internet, for instance. The proposed approach is structured in two stages. The first stage is responsible for clustering each storage unit of the distributed repository, generating a set of clustered units. Then, the second stage puts together all clustered units and once again applies the clustering algorithm for constructing the whole clustered repository.

This paper is organized as follows. Section II describes related techniques, evincing the contribution of evolving the centralized cluttering approach. Section II reviews the adopted component description model, called X-ARM. Then, Section IV presents the proposed distributed clustering approach. After that, outcomes observed in experiments are presented in Section V. In conclusion, Section VI presents final remarks and delineates future work.

## II. RELATED TECHNIQUES

Data clustering is a NP-hard problem and several heuristics have been proposed to group similar data. Xu and Wunsch [13] present a relevant survey of the research field. Among existing techniques, the hierarchical and the

K-means algorithms are applied in several domains [14]. In software engineering, clustering techniques have been used in several problems, such as: software maintenance [15], modularization [16], reverse engineering [17], software evolution [18] and software requirements [19].

Although clustering techniques are applied in several problems of software engineering, for the best knowledge of the authors, we are the first research group that adopts such techniques in the context of indexing software component repositories, proposing a centralized clustering approach for reducing the storage space requirements in local and centralized software component repositories [8].

As an evolved version of the previous centralized approach, the differential and so the contribution of the evolved approach proposed herein is twofold. First, the proposal of a new heuristic, also based on the hierarchical algorithm, but adopting a divide and conquer strategy that can be adopted for large-scale, distributed software component repositories. The second contribution is the consolidation of the similarity metric, initially proposed in the centralized version of the clustering approach.

The proposed heuristic has been validated by considering several thresholds for guiding the composition of component groups and also different repository sizes. Moreover, the heuristic was integrated as part of the indexing system adopted in [7], allowing quality evaluation for queries.

## III. THE X-ARM MODEL

The proposed approach explores the X-ARM description model, which is a XML based semi-structured data model for describing several types of software assets [4], which can be produced in CBD (Component-Based Development) processes, proving the required semantic for representing their relationships. As illustrated in Fig. 1, X-ARM describes component and interface specifications, as well as component implementations.

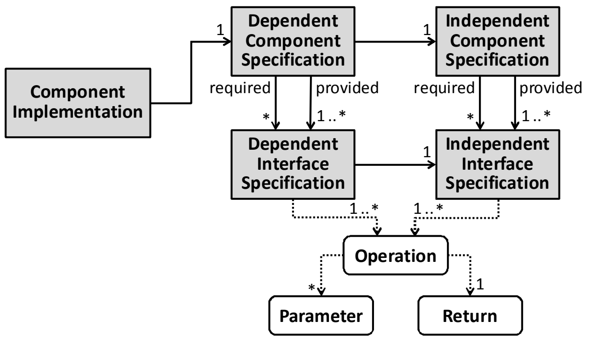

Dependent Component Specification
Independent Component Specification
Component Implementation
Dependent Interface Specification
Independent Interface Specification
required
provided
Operation
Parameter
Return
1
*
1..*

Figure 1. Relationships between assets.

Component and interface specifications can be described as independent or dependent of component models. Independent specifications abstract features and properties of component models, such as CCM, JavaBeans, EJB and Web Services. In turn, dependent specifications ought to consider features and properties related to component models.

In X-ARM, both dependent and independent interface specifications are described as a set of operations. Each operation has a name, a set of input or output parameters and a return value. In CBD processes, dependent interfaces must be in conformance with their independent counterparts. So, in Fig. 1, dependent interfaces refer to their respective independent interfaces.

Dependent and independent component specifications have a set of provided and required interface specifications. However, note that dependent and independent component specifications refer to dependent and independent interface specifications, respectively. In CBD processes, dependent components must be in conformance with their respective independent counterparts. Thus, dependent components refer to their respective independent components.

Similarly, in CBD processes, component implementations must be in conformance with their respective dependent component specifications. So, in Fig. 1, note that component implementations must refer to their correspondent dependent component specifications. Besides, for each dependent component specification, several component implementations can be realized.

As an X-ARM example, Fig. 2 illustrates a fragment of a dependent component specification. Line 1 is the asset header, which has the asset identifier *(id)*. Lines 2 to 4 refer to its respective independent component specification. Then, lines 5 to 14 refer to all provided dependent interface specifications. Although not illustrated in Fig. 2, required interfaces can also be similarly specified.

```
01 <asset name="dependentCompSpec-X"
          id="compose.depenpentCompSpec-X-1.0-beta">
02   <model-dependency>
03     <related-asset name="independentCompSpec-Z"
                      id="compose.independentCompSpec-Z-1.0-stable"
                      relationship-type="independentComponentSpec"/>
04   </model-dependency>
05   <component-specification>
06     <interface>
07       <provided>
08         <related-asset name="dependentInterface-A"
                          id="compose.dependentIntSpec-A-2.0-stable"
                          relationship-type="dependentInterfaceSpec"/>
09       </provided>
10       <provided>
11         <related-asset name="dependentInterface-B"
                          id="compose.dependentIntSpec-B-3.0-stable"
                          relationship-type="dependentInterfaceSpec"/>
12       </provided>
13     </interface>
14   </component-specification>
15 </asset>
```

Figure 2. Dependent component specification in X-ARM.

## IV. A DISTRIBUTED CLUSTERING APPROACH

The task of indexing repositories based on semi-structured data is a well-known relevant issue [5][6][7]. One of the major challenges is to provide an indexing mechanism that reduces storage space requirements, but without excessively impacting on query processing time and precision level. As mentioned, we have been investigating clustering techniques as a mean to handle such a challenge.

As a result of our investigations, we propose herein a distributed clustering approach for optimizing storage space requirements in large-scale, distributed software component repositories. To do that, the approach constructs a clustered repository in which each asset of the original repository belongs to a group (cluster), formed by applying a clustering heuristic, according to similarity criteria detailed afterwards. The clustered repository is composed only by representative assets of the identified clusters. Thus, instead of indexing the original repository, the search engine needs only to index the reduced set of representative assets in the clustered repository, each one referring to its respective original assets.

Clustering techniques [14] consist of three basic phases: *(i)* extraction of relevant features that express the behavior of the elements to be clustered; *(ii)* definition of a similarity metric, used to compare elements; and *(iii)* application of a clustering algorithm in order to construct the clusters. For extracting features, it is identified which information is relevant to capture the behavior of the elements to cluster and how this information is quantified. The result is a vector of relevant features, called *attribute vector*. The similarity metric expresses, in quantitative terms, the similarity between elements. In general, a function is defined and the Euclidean distance [14] between two elements is one of the more common adopted metrics. Finally, the clustering algorithm is a heuristic that generates groups of similar elements, according to the adopted similarity metric.

It must be emphasized that both the centralized approach proposed in [8] and the distributed approach proposed herein adopt the same relevant features and similarity metrics. Thus, such features and metrics are briefly reviewed herein.

## A. *Relevant Features and Similarity Metrics*

The proposed approach considers five types of X-ARM assets and the clustering technique is applied separately for each one, which has a distinct attribute vector for representing its relevant features. Thus, similarity metrics are different for each type of asset.

The similarity between two assets $a$ and $b$ is quantified by an integer number, called *distance*. To avoid negative distances, the initial default distance $d_i$ is 300. The similarity criterion is applied and this value may decrease, in such a way that assets are more similar when the final distance $d_f(a,b)$ approximates to zero. So, for each type of asset, the similarity between two assets $a$ and $b$ is defined by (1), in which the term $s(a,b)$ represents a factor derived based on the similarity criterion for the respective type of asset.

$$d_f(a,b) = d_i - s(a,b) \qquad (1)$$

The relevant features of an independent interface specification are its defined operations, considering their names, input and output parameters and return values. Thus, independent interfaces are similar when they have in common a considerable subset of operations. In this case, the term $s(a,b)$ is defined by $op(a,b) \times 300$, where $op(a,b)$ represents the percentage of common operations provided by both interfaces. Note that the similarity criterion takes into account syntactic features only. Thus, when operations are semantically similar but syntactically different, the proposed approach cannot detect similarity, reducing its effectiveness.

Taking into account dependent interface specifications, the relevant features are the referenced independent interface specification together with their operations. Hence, dependent interface specifications are similar when they refer to the same independent interface specification or have in common a considerable subset of defined operations. Accordingly, $s(a,b)$ is expressed as the sum of two terms: $l(a,b)$ and $op(a,b) \times 100$. The term $l(a,b) = 200$ if both assets refer to the same independent interface specification; otherwise it is 0. In turn, the term $op(a,b)$ is the ratio of common operations provided by both interfaces.

In relation to independent component specifications, the relevant features are the set of provided independent interface specifications. So, independent component specifications are similar when they have in common a considerable subset of provided independent interface specifications. Due that, the term $s(a,b)$ is given by $p(a,b) \times 300$, where $p(a,b)$ is the percentage of common independent interfaces provided by both assets.

For a dependent component specification, the relevant features are its referenced independent component specification, as well as its set of provided dependent interface specifications. Therefore, dependent component specifications are similar when they refer to the same independent component specification or have in common a subset of provided dependent interfaces. As a result, $s(a,b)$ is the sum of two terms: $k(a,b)$ and $p(a,b) \times 100$. The term $k(a,b) = 200$ if both assets refer to the same independent component specification; otherwise it is 0. In turn, the term $p(a,b)$ is the ratio of common dependent interface specifications provided by both assets.

Finally, for a component implementation, the relevant feature is its referenced dependent component specification. Hence, different implementations of the same dependent component specification are considered similar. Due that, the term $s(a,b) = 300$ if both assets refer to the same dependent component specification; otherwise it is 0.

## B. *Clustering Algorithm*

The proposed clustering algorithm has been designed to be applied in distributed software component repositories, like X-CORE [11], defined as a collection of connected, integrated storage units, each one located in different independent nodes, which are dispersed throughout a network like the internet, for instance.

The clustering algorithm has two stages. The first stage adopts a divide and conquer strategy, which is responsible for separately clustering each storage unit of the distributed repository, generating a set of independent clustered units. Subsequently, the second stage joins together all clustered units and generates the whole clustered repository by adopting a pair-wised clustering scan. In order to reduce processing time, the first stage must be concurrently performed in all storage units, and the second stage ought to be concurrently performed in a single node using a multi-threaded, pair-wised clustering scan. In the following both stages are described in more details.

In the first stage, which is executed in each storage unit, initially, assets are randomly chosen from the storage unit and kept in primary memory. It is suggested to exhaust memory capacity with this operation. Next, but still in the first stage, the classical hierarchical clustering algorithm [14] is applied to these assets. Initially, each asset is considered a cluster. Then, the algorithm groups successively the two nearest clusters, until the distance between clusters is greater than an established threshold, specified by the user. The algorithm considers the similarity metric described previously to compute the distance. The combined cluster is considered a representative asset of the joined clusters. For each type of asset, the representative asset includes the relevant features for the similarity metric and also references to the joined assets. At the end of the iteration, a directory containing all formed representative assets (clusters) is saved in secondary memory. Then, in a new iteration, another set of assets, remaining in the original storage unit, is arbitrarily chosen and the process repeats. At the end of the first stage, several directories of representative assets have been constructed, one for each iteration of the algorithm. Fig. 3 illustrates the main steps of the first stage: *(a)* assets are randomly selected from the original storage unit; *(b)* clusters composed of similar assets are constructed by applying the hierarchical clustering algorithm; *(c)* representative assets are created in the respective directory for representing each cluster; and *(d)* a clustered unit is defined based on directories created in each iteration.

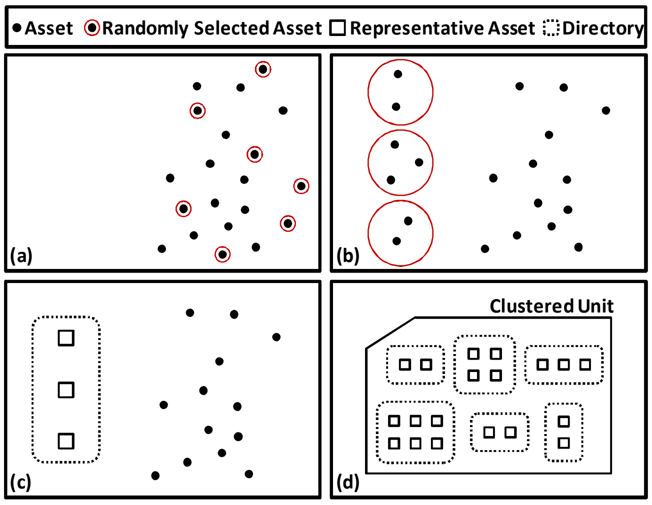


Figure 3. The first stage of the clustering algorithm in each storage unit.

To conclude the description of the first stage, it remains to explain how representative assets are generated. Note that the generation of representative assets is different for each type of asset. A representative asset, resulted from the combination of two clusters composed by dependent component specifications, includes all provided dependent interface specifications of the joined assets and the independent component specification they refer. This specification is the one that mostly occurs in the assets of the combined cluster. In turn, for clusters composed by independent component specifications, the representative asset includes all provided independent interface specifications of the joined assets.

A representative asset, resulted from the combination of two clusters composed by dependent interface specifications, includes all operations of the joined assets, as well as the independent interface they refer. This interface is the one mostly referred by the joined clusters. Taking into account a representative asset, resulted from the combination of two clusters composed by independent interface specifications, it includes all provided operations of the joined assets.

Finally, a representative asset, resulted from the combination of two clusters of component implementations, includes its referenced dependent component specification, which is also the one mostly frequent in the joined assets.

After generating all representative assets, the second stage takes place and joins together all clustered units, generating the whole clustered repository based on a pair-wised clustering scan. To do that, first, all clustered units are transferred to a single node that constitutes the distributed repository, in which the second stage takes place. Since clustered units are composed of representative assets only, the transmission of them to the selected node is faster than the transmission of the original storage units.

Thus, the second stage takes as input a set of directories, which have direct correlation with directories that belong to all clustered units. Once clustered units are stored in the selected node, the second stage adopts a multi-threaded, pair-wised clustering scan, in which pairs of directories are recursively processed and their representative assets are combined by applying the hierarchical clustering algorithm. At the end of the second stage, only one directory remains.

In order to join representative assets from directories $A$ and $B$, the following procedure is applied. For each representative asset $a_i \in A$ and $b_j \in B$, the distance between them is computed, say $d_f(a_i, b_j)$. Let $(a_k, b_l)$ be the pair for which the distance is minimum. If $d_f(a_k, b_l)$ is less than the defined threshold, then $a_k$ and $b_l$ are joined and the combined asset is added to $B$ (also, $a_k$ and $b_l$ are removed from $A$ and $B$); otherwise $a_k$ is moved from $A$ to $B$. The combinations of directories are performed in parallel, according to a divide and conquer strategy. Thus, pairs of directories are combined independently and successively until only one remains.

## V. Results and Discussion

In order to evaluate the proposed approach, a set of experiments has been carried out for identifying the gains in terms of number of assets and storage space requirements between the clustered repository and the original repository. In experiments, the gains were evaluated taking into account a repository composed by 14000 X-ARM assets of different types, created by a customizable script that automatically generates them. After creating the repository, the proposed approach has been applied for grouping the stored assets in clusters, generating their respective representative assets.

Table I presents the number of each type of asset in the original repository and the clustered repositories after the application of the proposed approach using different thresholds, which vary from 100 to 200 in steps of 25. Note that the proposed approach significantly reduces the number of assets. As expected, the number of representative assets decreases as the threshold increases. When the threshold is increased, two assets have more chance of being considered similar, and so, more chance of being grouped together. Thus, when the threshold increases from 100 to 200, the number of original assets reduces to 11176 and 7204 representative assets, respectively.

TABLE I. NUMBER OF ASSETS FOR DIFFERENT THRESHOLDS.

| | Indep Int Spec | Dep Int Spec | Indep Comp Spec | Dep Comp Spec | Comp Impl | Total |
|---|---|---|---|---|---|---|
| Original | 5000 | 3200 | 3200 | 1800 | 800 | 14000 |
| Cluster 100 | 4104 | 2070 | 3023 | 1367 | 612 | 11176 |
| Cluster 125 | 4053 | 2043 | 3012 | 1363 | 609 | 11080 |
| Cluster 150 | 3386 | 1830 | 2634 | 1278 | 598 | 9726 |
| Cluster 175 | 3315 | 1809 | 2628 | 1272 | 594 | 9618 |
| Cluster 200 | 2168 | 1346 | 2045 | 1085 | 560 | 7204 |

For each considered threshold, the gain in number of assets has been identified and evaluated. Fig. 4 illustrates the gain in terms of the number of assets. For example, when the threshold is 150, the number of stored assets in the original repository is reduced around 30.5%, dropping from 14000 original assets to 9726 representative assets. As can be noticed in Fig. 4, the proposed approach performs a significant reduction in the number of stored assets, achieving relevant gains between 48.5% and 20.2%.

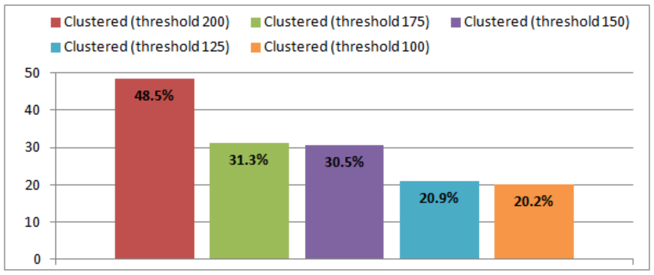


Figure 4. Total gain in number of assets.

However, as shown in Fig. 5, the gains are different for each type of asset. In general, the better gains are achieved for independent and dependent interfaces. For independent interfaces, the gains are between 56.6% and 17.9%. Considering dependent interfaces, the gains become a little bit more expressive, varying between 57.9% and 35.3%. Such higher gains can be explained by the considerable amount of assets of those types. As can be seen in Table I, the original repository has 5000 and 3200 independent and dependent interfaces, respectively. Thus, independent interfaces are the prevalent type of asset in the repository, increasing the likelihood of identifying similar assets.

In the case of dependent interface specifications, the gains become better due to three reasons. First, the number of assets of that type is also relevant. Second, in software projects, it is not rare to implement different versions of software systems for different target platforms. So, in component-based software projects, different versions imply on several dependent interface specifications for each independent interface specification. Considering that dependent interface specifications are considered similar when they refer to the same independent interface specification, it is easy to see that multiple implementations impacts on the likelihood of identifying similar dependent interface specifications. The third reason is a consequence of high gains in independent interfaces, which directly impact in the similarity metric of dependent interfaces.

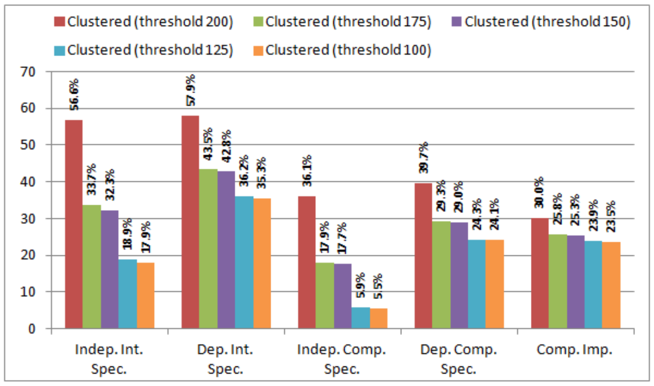


Figure 5. Gains in number of assets for different types of assets.

In relation to independent component specifications, despite the relevant number of assets (3200 assets according to Table I), the gains are notably low, varying from 36.1% to 5.5%. Besides, as can be noticed in Fig. 5, the gain of 36.1% occurs for the higher threshold only. When the threshold is 175 and 125, the respective gains decrease to 17.9% and 5.9%. Such gains are relatively low and indeed not expected. As mentioned before, independent component specifications are similar when they have in common a considerable subset of provided independent interfaces. Thus, it can be inferred that such low gains are a consequence of the difficulty of finding two or more independent component specifications that share a reasonable subset of independent interfaces.

In terms of dependent component specifications, surprisingly, the gains become more expressive, increasing to the range between 39.7% and 24.1%. It is possible to mention two reasons for such a surprising gain and both reasons are similar to effects pointed out before for dependent interface specifications. First, the existence of different versions of software systems for different target platforms implies on several dependent component specifications for each independent component specification. Considering that dependent component specifications are considered similar when they refer to the same independent component specification, it is clear to notice that multiple implementations impacts on the likelihood of identifying similar dependent component specifications. The second reason is a consequence of the gains in independent component specifications, which directly impact in the similarity metric of dependent component specifications.

Finally, considering component implementations, the gains are once more surprisingly good, varying between 30.0% and 23.5%. Taking into account that component implementations are considered similar when they refer to the same dependent component implementation, it is also possible to correlate such a good gain with the existence of different implementations of the same component specification, not only for different target platforms but also for meeting a variety of non-functional requirements, like performance, security and cost. Therefore, considering the various methods, techniques and algorithms that can be employed to meet non-functional requirements, it is obvious that such multiple implementations impact on the likelihood of identifying similar component implementations.

Despite the mentioned gains, it is not enough to be efficient in reducing the number of assets, but also in downgrading storage space requirements for index files. So, after generating the clustered repositories, they were indexed using the indexing technique proposed in [7]. Table II presents the storage space required by all repository versions, after indexing them. As can be noticed, the proposed approach significantly reduces the required storage space, and, as expected, it decreases as the threshold increases. Thus, when the threshold increases from 100 to 200, the required storage space reduces from 12.50 to 9.22 MB.

TABLE II. STORAGE REQUIREMENTS FOR DIFFERENT THRESHOLDS.

| **Original** | **Clust 100** | **Clust 125** | **Clust 150** | **Clust 175** | **Clust 200** |
|---|---|---|---|---|---|
| 20.00 MB | 12.50 MB | 12.42 MB | 11.35 MB | 11.22 MB | 9.22 MB |

For each threshold, Fig. 6 illustrates the gains in terms of storage space requirements. For example, when the threshold is 150, the storage space required by index files reduces around 45.2%, dropping from 20.70 to 11.35 MB. Note that, the proposed approach significantly reduces storage space requirements, achieving gains between 55.5% and 39.6%.

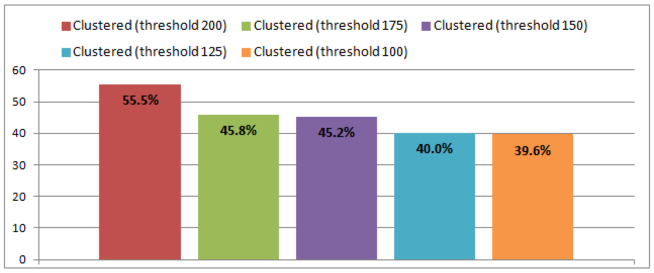


Figure 6. Total gain in storage requirements.

## VI. CONCLUSION AND FUTURE WORK

Based on evaluation outcomes, it can be clearly observed the potential of the proposed approach in significantly clustering a large-scale, distributed repository and consequently reducing storage space required by index files. It must be highlighted that, the bigger the original repository in terms of the number of stored assets, the more expressive the likelihood of clustering assets, and so the better the gain in terms of storage space requirements.

Taking into account that the indexing technique proposed in [7] has an excellent performance in query processing time, even in large-scale index files, it is expected a reasonable gain in terms of query processing time due to the expressive reduction in the size of index files. Thus, the proposed approach clearly reduces storage space requirements, without introducing query processing time costs.

However, such expressive gains in terms of storage space requirements, almost certainly have an impact on the query precision level, since the process of clustering assets introduces some degree of information loss in representative assets. It must be stressed that the tradeoff between the best threshold and the query precision level has been initiated and preliminary results are encouraging, showing high values of precision and recall, which are two well-known metrics adopted in evaluation of information retrieval systems. Despite that, the impact of the proposed approach in terms of information loss still needs to be evaluated in more details. Besides, it is also under investigation a comparative analysis contrasting the distributed approach proposed herein and the centralized approach proposed in [8], both in terms of storage requirements gains and time complexity.

## ACKNOWLEDGMENT

This work was supported by the National Institute of Science and Technology for Software Engineering (INES – www.ines.org.br), funded by CNPq, grants 573964/2008-4.

## REFERENCES

[1] A. Orso, M. J. Harrold, and D. S. Rosenblum, “Component Metadata for Software Engineering Tasks”, Proc. 2nd Int. Workshop on Engineering Distributed Objects, 2000, pp. 126-140.
[2] P. Vitharana, F. Zahedi, and H. Jain, "Knowledge-Based Repository Scheme for Storing and Retrieving Business Components: A Theoretical Design and an Empirical Analysis", IEEE Trans. on Soft. Engineering., vol. 29, issue 7, July 2003, pp. 649-664.
[3] OMG, Reusable Asset Specification – Version 2.2, 2005.
[4] G. Elias, M. Schuenck, Y. Negócio, J. Dias, and S. Miranda, “X-ARM: An Asset Representation Model for Component Repository”, 21st ACM Symposium on Applied Computing, 2006, pp. 1690-1694.
[5] W. Meier, “eXist: An Open Source Native XML Database”, NODe 2002 Web and Database-Related Workshops on Web, Web-Services, and Database Systems, 2002.
[6] R. Goldman and J. Widom, “DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases”, 23rd Int. Conf. on Very Large Data Bases, 1997, pp. 436-445.
[7] T. Brito, T. Ribeiro, and G. Elias, “Indexing Semi-Structured Data for Efficient Handling of Branching Path Expressions”, 2nd Int. Conf. on Advances in Databases, Knowledge, and Data Applications, 2010, pp. 197-203.
[8] M. P. Paixão, L. Silva, T. Brito, and G. Elias, “Large Software Component Repositories into Small Index Files”, 3rd Int. Conf. on Advances in Databases, Knowledge, and Data Applications, 2011, pp. 122-127.
[9] R. C. Seacord, “Software Engineering Component Repositories”, Technical Report, Software Engineering Institute (SEI), 1999.
[10] W. Frakes and K. Kang. “Software Reuse Research: Status and Future”, IEEE Trans. on Soft. Engineering, vol. 31, no. 7, July, 2005.
[11] J. P. Oliveira, T. Brito, A.E. Oliverira, S. E. Rabelo, and G. Elias, “X-CORE: A Shared, Disributed Component Repository Service”, 24th Tools Session / 21st Brazilian Symp. on Soft. Engineering, 2007, pp.100-106 (in portuguese).
[12] C. Boldyreff, D. Nutter, and S. Rank, (2002) “Open-Source Artifact Management”. Workshop Open Source Sof. Engineering, USA, 2002.
[13] R. Xu and D. Wunsch, “Survey of Clustering Algorithms”, IEEE Trans. on Networks, vol.16, issue 3, pp. 645-678, May 2005.
[14] A. K. Jain and R. C. Dubes, Algorithms for Clustering Data, Prentice Hall, 1984.
[15] S. Mancoridis, et al. “Using Automatic Clustering to Produce High Level System Organizations of Source Code”. Proc. IEEE Int. Workshop on Program Comprehension, pp. 45–53, 1998.
[16] B. S. Mitchel and S. Mancoridis. “On the Automatic Modularization of Software Systems Using the Bunch Tool”, IEEE Trans. on Soft. Engineering, vol. 32, issue 3, pp. 1-16, March 2006.
[17] Y. Chiricota, F. Jourdan, and G. Melançon, “Software Component Capture using Graph Clustering”, IEEE Int. Workshop on Program Comprehension, 2003.
[18] J. Wu, A. E. Hassan, and R. C. Holt, "Comparison of Clustering Algorithms in the Context of Software Evolution", 21st Int. Conf. on Soft. Maintenance, 2005, pp. 525-535.
[19] Z. Li, Q. A. Rahman, and N. H. Madhavji, “An Approach to Requirements Encapsulation with Clustering”, 10th Workshop on Requirement Engineering, 2007, pp. 92-96.

# OLAP Authentication and Authorization via Query Re-writing

Todd Eavis
*Department of Computer Science*
*Concordia University*
*Montreal, Canada*
*Email: eavis@cs.concordia.ca*

Ahmad Altamimi
*Department of Computer Science*
*Concordia University*
*Montreal, Canada*
*Email: a_alta@cs.concordia.ca*

***Abstract*—Online Analytical Processing (OLAP) has become an increasingly important and prevalent component of Decision Support Systems. OLAP is associated with a data model known as a cube, a multi-dimensional representation of the core measures and relationships within the associated organization. While numerous cube generation and processing algorithms have been presented in the literature, little effort has been made to address the unique security and authentication requirements of the model. In particular, the hierarchical nature of the cube allows users to bypass - either intentionally or unintentionally - partial constraints defined at alternate aggregation levels. In this paper, we present an authentication framework that builds upon an algebra designed specifically for OLAP domains. It is Object-Oriented in nature and uses query re-writing rules to ensure consistent data access across all levels of the conceptual model. The process is largely transparent to the user, though notification is provided in cases in which a subset of the original request is returned. We demonstrate the scope of our framework with a series of common OLAP queries. The end result is an intuitive but powerful approach to database authentication that is uniquely tailored to the OLAP domain.**

***Keywords*-Data warehouses; Data security; Query processing**

## I. Introduction

Data warehousing (DW) and On-Line Analytical Processing (OLAP) play a pivotal role in modern organizations. Designed to facilitate the reporting and analysis required in decision making environments [1], OLAP builds upon a multi-dimensional data model that intuitively integrates the vast quantities of transactional level data collected by contemporary organizations. Ultimately, this data is used by managers and decision makers in order to extract and visualize broad patterns and trends that would otherwise not be obvious to the user.

One must note that while the data warehouse serves as a repository for all collected data, not all of its records should be universally accessible. Specifically, DW/OLAP systems almost always house confidential and sensitive data that must, by definition, be restricted to authorized users. The administrator of the warehouse is ultimately responsible for defining roles and privileges for each of the possible end users. In fact, a number of general warehouse security models have been proposed in the literature [2]–[5]. Several authors define frameworks that are likely too restrictive for production warehouses. For instance, Rosenthal et al. [4] discuss a security model based on *authorization views*. The views are created for specific users so as to allow access only to selected data. However, the administration of these views becomes quite complex when a security policy is added, changed, or removed. Moreover, complex roles can be difficult to implement in practice, and models of this type tend not to scale well with a large number of users. Conversely, other researchers have focused on the design process itself. For example, Fernndez-Medina et al. [2] propose a Unified Modeling Language (UML) profile for the definition of security constraints. Here, however, the physical implementation of the underlying authentication system remains undefined.

In this paper, we present an authentication model for OLAP environments that is based on a query rewriting technique. The model enforces distinct data security policies that, in turn, may be associated with user populations of arbitrary size. In short, our framework rewrites queries containing unauthorized data access to ensure that the user only receives the data that he/she is authorized to see. Rewriting is accomplished by adding or changing specific conditions within the query according to a set of concise but robust transformation rules. Because our methods specifically target the OLAP domain, the query rules are directly associated with the conceptual properties and elements of the OLAP data model itself. A primary advantage of this approach is that by manipulating the conceptual data model, we are able to apply query restrictions not only on direct access to OLAP elements, but also on certain forms of indirect access.

The remainder of this paper is organized as follows. In Section II, we present an overview of related work. Section III describes the core OLAP data model and associated algebra, and includes a discussion of the object-oriented query structure for which the proposed security model has been designed. The OLAP query rewriting model and its associated transformation rules are then presented in detail in Section IV. Final conclusions are offered in Section V.

## II. Related work

As noted above, a number of security models that restrict warehouse access have been proposed in the literature, including those that focus strictly on the design process.

Extensions to the Unified Modeling Language to allow for the specification of multi-dimensional security constraints has been one approach that has been suggested [2]. In fact, a number of researchers have looked at similar techniques for setting access constraints at an early stage in the OLAP design process [6], [7]. Such models have great value of course, particularly if one has the option to create the warehouse from scratch. That being said, their focus is not on authentication algorithms per se, but rather on design methodologies that would most effectively use existing technologies.

In terms of true authentication models, several researchers have attempted to augment the core Database Management System (DBMS) with authorizations views [4], [8]. Typically, alternate views of data are defined for each distinct user or user group. The end result is often the generation of a large number of such views, all of which must be maintained manually by the system administrator. Clearly, this approach does not scale terribly well, and would be impractical in a huge, complex DW environment.

Query rewriting has also been explored in DBMS environments in a variety of ways, with search and optimization being common targets [9]. Beyond that, however, rewriting has also been utilized to provide fine grained access control in Relational databases [10]. Oracle's Virtual Private Database (VPD) [11], for example, limits access to row level data by appending a predicate clause to the user's SQL statement. Here, the security policy is encoded as policy functions defined for each table. These functions are used to return the predicate, which is then appended to the query. This process is done in a manner that is entirely transparent to the user. That is, whenever a user accesses a table that has a security policy, the policy function returns a predicate, which is appended to the user's query before it is executed.

In the Truman model [10], on the other hand, the database administrator defines a *parameterized* authorization view for each relation in the database. Note that parameterized views are normal views augmented with session-specific information, such as the user-id, location, or time. The query is modified transparently by substituting each relation in the query by the corresponding parameterized view to make sure that the user does not get to see anything more than his/her own view of the database. In this model, the user can also write queries on base relations by plugging in the values of session parameters such as user-id or time before the modified query is executed.

We note, however, that the mechanisms discussed above (e.g., Oracle's VPD) are not tailored specifically to the OLAP domain and, as such, either have limited ability to provide fine grained control of the elements in the conceptual OLAP data model or, at the very least, would make such constraints exceedingly tedious to define. Some commercial tools, such as Microsoft's Analysis Services [12], do in fact provide such mechanisms. However, even here, authentication controls are quite direct in that they must be explicitly associated with any and all affected elements of the model. This is in contrast to the work discussed in this paper, where the primary contribution is a query rewriting technique that transparently supports *indirect* authentication.

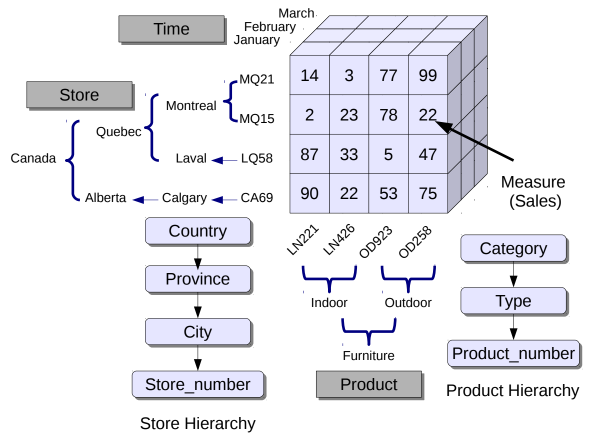


Figure 1. A simple three dimensional data cube

## III. The Conceptual Data Model

We consider analytical environments to consist of one or more *data cubes*. Each cube is composed of a series of $d$ dimensions — sometimes called *feature* attributes — and one or more *measures* [13]. The dimensions can be visualized as delimiting a $d$-dimensional hyper-cube, with each axis identifying the members of the parent dimension (e.g., the days of the year). Cell values, in turn, represent the aggregated measure (e.g., sum) of the associated members. Figure 1 provides an illustration of a very simple three dimensional cube on `Store`, `Time` and `Product`. Here, each unique combination of dimension members represents a unique aggregation on the measure. For example, we can see that Product OD923 was purchased 78 times at Store MQ15 in January (assuming a `Count` measure).

Note, as well, that each dimension is associated with a distinct aggregation hierarchy. Stores, for instance, are organized in `Country` $\rightarrow$ `Province` $\rightarrow$ `City` groupings. Referring again to Figure 1, we see that `Product Number` is the lowest or *base* level in the Product dimension. In practice, data is physically stored at the base level so as to support run-time aggregation to coarser hierarchy levels. Moreover, the attributes of each dimension are partially ordered by the dependency relation $\preceq$ into a dependency lattice [14]. For example, `Product Number` $\preceq$ `Type` $\preceq$ `Category` within the Product dimension. More formally, the dependency lattice is expressed in Definition 1.

*Definition 1:* A dimension hierarchy $H_i$ of a dimension $D_i$, can be defined as $H_i = (L_0, L_1, \ldots, L_j)$ where $L_0$ is the lowest level and $L_j$ is the highest. There is a functional dependency between $L_{h-1}$ and $L_h$ such that $L_{h-1} \preceq L_h$ where (0 ≤ h ≤ j).

Finally, we note that there are in fact many variations on the form of OLAP hierarchies [15] (e.g., symmetric, ragged, non-strict). Regardless of the form, however, traversal of these aggregation paths — typically referred to as *rollup and drill down* — is perhaps the single most common query form. It is also central to the techniques discussed in this paper.

### A. Native Language Object Oriented OLAP Queries

The cube representation, as described above, is common to most OLAP query environments and represents the user's conceptual view of the data repository. That being said, it can be difficult to implement the data cube using standard relational tables alone and, even when this is possible, performance is usually sub-par as relational DBMSs have been optimized for transactional processing. As a result, most OLAP server products either extend conventional relational DBMSs or build on novel, domain specific indexes and algorithms.

In our own case, the authentication methods described in this paper are part of a larger project whose focus is to design, implement and optimize an OLAP-specific DBMS server. A key design target of this project is the integration of the conceptual cube model into the DBMS itself. This objective is accomplished, in part, by the introduction of an OLAP-specific algebra that identifies the core operations associated with the cube (SELECT, PROJECT, DRILL DOWN, ROLL UP, etc). In turn, these operations are accessible to the client side programmer by virtue of an Object Oriented API in which the elements of the cube (e.g., cells, dimensions, hierarchies) are represented in the *native* client language as simple OOP constructs. (We note that our prototype API uses Java but any contemporary OO language could be used). To the programmer, the cube and all of its data — which is physically stored on a remote server and may be Gigabytes or Terabytes in size — appears to be nothing more than a local in-memory object. At compile time, a fully compliant Java pre-parser examines the source code, creates a parse tree, identifies the relevant OLAP objects, and re-writes the original source code to include a native DBMS representation of the query. At run-time, the pre-compiled queries are transparently delivered to the back end analytics server for processing. Results are returned and encapsulated within a proxy object that is exposed to the client programmer.

As a concrete example, Listing 1 illustrates a simple SQL query that summarizes the total sales of Quebec's stores in 2011 for the data cube depicted in Figure 1. Typically, this query would be embedded within the application source code (e.g., wrapped in a JDBC call). Conversely, Listing 2 shows how this same query could be written in an Object-Oriented manner by a client-side Java programmer. Note

```
Select Store.province, SUM(sales)
From  Store, Time, Sales
Where Store.store_ID = Sales.store_ID AND
 Time.time_ID = Sales.time_ID AND
 Time.year = 2011 AND
 Store.province = 'Quebec'
Group by  Store.province
```

Listing 1. Simple SQL OLAP Query

```
Class SimpleQuery extends OLAPQuery{
 Public boolean select(){
  Store store = new Store();
  DateDimension time = new TimeDimension();
  return (time.getYear() == 2011 &&
      store.getProvince() == 'Quebec');
 }
 Public Object[] project(){
  Store store = new Store();
  Measure measure = new Measure();
  Object[] projections = {
   store.getProvince(),
   measure.getSales()};
  return projections;
 }}
```

Listing 2. An Object Oriented OLAP Query

that each algebraic operation is encapsulated within its own method (in this case, SELECT and PROJECT), while the logic of the operation is consolidated within the `return` statement. It is the job of the pre-parser to identify the relevant query methods and then extract and re-write the logic of the `return` statement(s). Again, it is important to understand that the original source code will never be executed directly. Instead, it is translated into the native operations of the OLAP algebra and sent to the server at run-time.

While it is outside the scope of this paper to discuss the motivation for native language OLAP programming (a detailed presentation can be found in a recent submission [16]), we note that such an approach not only simplifies the programming model, but adds compile time type checking, robust re-factoring, and OOP functionality such as query inheritance and polymorphism. Moreover, query optimization is considerably easier on the backed as the DBMS natively understands the OLAP operations sent from the client side. In the context of the current paper, however, the significance of the query transformation process is that the authentication elements (e.g. roles and permissions) will be directly associated with the operations of the algebra. In fact, it is this algebraic representation that forms the input to the authentication module presented in the remainder of the paper.

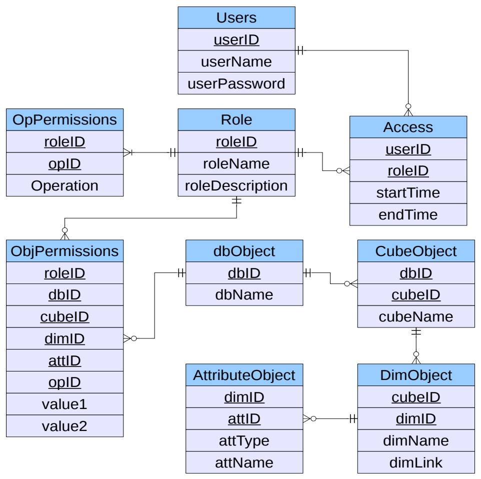


Figure 2. The Authentication DB.

## IV. Authentication and Authorization

Without sufficient security countermeasures, open access to the OLAP repository becomes a powerful tool in the hands of malicious or unethical users. *Access Control* is the process that restricts unauthorized users from compromising protected data. This process can be thought of as occurring in two basic phases: **Authentication** and **Authorization**. Authentication is a form of *identity verification* that attempts to determine whether or not a user has valid credentials to access the system. In contrast, Authorization refers to the process of determining if the user has permission to access a specific data resource. In this section, we will describe our general framework, giving a detailed description of its two primary components and the relationship between them.

### A. The Authentication Module

The authentication component is responsible for verifying user credentials against a list of valid accounts. These accounts are provided by the system administrator and are kept — along with their constituent permissions — in a backend database (i.e., the Authentication DB). The Authentication DB consists of a set of tables (`users`, `permissions`, and `objects`) that collectively represent the meta data required to authenticate and authorize the current user. For example, the `users` table stores basic user credentials (e.g., name, password), while the `permissions` table records the fact that a given user(s) may or may not access certain controlled objects. Figure 2 illustrates a slightly simplified version of the Authentication DB schema. In the current prototype, storage and access to the Authentication DB is provided by the SQLite toolkit [17]. SQLite is a small, open source C language library that is ideally suited to tasks that require basic relational query facilities to be embedded within a larger software stack.

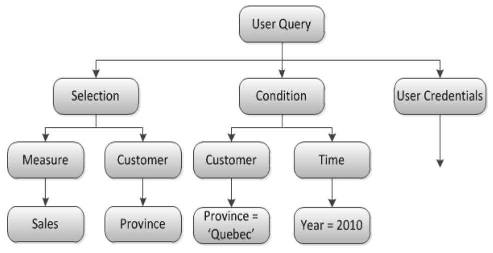


Figure 3. A small Parse Tree fragment.

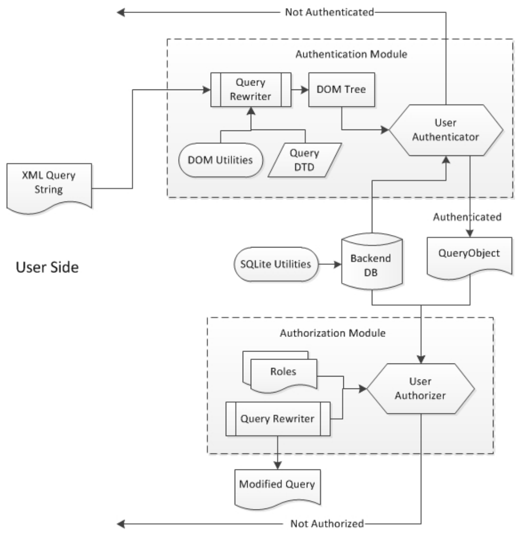


Figure 4. Authentication and Authorization.

Internally, the user's transformed OLAP query is represented in XML format (embedded within the re-written source code). Of course, in order to properly authenticate the query, it must first be decomposed into its algebraic components. The DBMS backend parser uses standard DOM utilities to parse the received query and extract the query elements (user credentials, dimensions, hierarchies, etc). Figure 3 shows the XML tree corresponding to the query depicted in Listing 2. Once the parsing is completed, the Authentication module extracts the user credentials to verify them against the Authentication DB. If the verification is successful, the DBMS proceeds with the authorization process. Otherwise, the query is rejected and the user/programmer is notified. In the case of successful authentication, the query elements are loaded into a memory-resident `QueryObject` structure for Authorization checking. The upper part of Figure 4 depicts the processing logic of the Authentication module.

### B. The Authorization Module

The second — and more significant — phase is authorization, the process of determining if the user has permission to access specific data elements. Specifically, when a user requests access to a particular resource, the request is validated against the *permitted resource list* assigned to that user in the backend database. If the requested resource produces a valid match, the user request is allowed to execute as originally written. Otherwise, the query will either be rejected outright or modified according to a set of flexible transformation rules. To decide if the query will be modified or not, we rely on a set of *authorization objects* against which the rules will be applied. The rules themselves will be discussed in Section IV-D. The lower portion of Figure 4 graphically illustrates the Authorization module and indicates its interaction with the Authentication component.

### C. Specifying Authorization Objects

Authorization is the granting of a right or privilege that enables a subject (e.g., users or user groups) to execute an action on an object. In order to make authorization decisions, we must first define the authorization objects. Note that the *objects* in the OLAP domain are different from those in the relational context. In a relational model, objects include logical elements such as tables, records within those tables, and fields within each record. In contrast, OLAP objects are elements of the more abstract conceptual model and include the dimensions of the multi-dimensional cube, the hierarchies within each dimension, and the aggregated cells (or facts). In practice, this changes the logic or focus of the authentication algorithm. For instance, a user in a relational environment may be allowed direct access to a specific record (or field in that record), while an OLAP user may be given permission to dynamically aggregate measure values at/to a certain level of detail in one or dimension hierarchies. Anything below this level of granularity would be considered too *sensitive*, and hence should be protected. In fact, the existence of aggregation hierarchies is perhaps the most important single distinction between the authentication logic of the OLAP domain versus that of the relational world.

We note that in the discussion that follows, we assume an *open world* policy, where only prohibitions are specified. In other words, permissions are implied by the absence of any explicit prohibition. We use the open world policy mainly for practical reasons, as the sheer number of possible prohibitions in an enterprise OLAP environment would be overwhelming.

Before discussing the authorization rules themselves, we first look at a pair of examples that illustrate the importance of proper authorization services in the OLAP domain. We begin with the definition of a policy for accessing a specific aggregation level in a data cube dimension hierarchy.

*Example 1:* An employee, Alice, is working in the Montreal store associated with the cube of Figure 1. The policy is simple: Alice should not know the sales totals of the individual provinces.

Clearly, Alice is prohibited from reading or aggregating data at the provincial level in the Store dimension hierarchy.

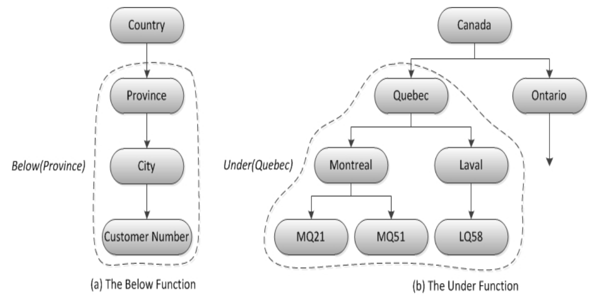


Figure 5. The Below and Under functions.

However, in the absence of any further restrictions, it would still be possible for her to compute the restricted values from the lower hierarchies levels (e.g., City or Store_Number). Ideally, the warehouse administrator should not be responsible for identifying and manually ensuring that all *implied* levels be included in the policy. Instead, our model assumes this responsibility and can, if necessary, restrict access to all *child* levels through the use of the *Below* function. As the name implies, this function returns a list consisting of the specified level $L_i$ and all the lower levels of the associated dimension hierarchy. Figure 5(a) illustrates an example using a *Below(Province)* instantiation. Here, all levels surrounded by the dashed line are considered to be Authorization Objects, and thus should be protected. The formalization of the *Below* function is given by Definition 2.

*Definition 2:* In any dimension $D_i$ with hierarchy $H_i$, the function Below($L_i$) is defined as Below($L_i$) = $\{L_j$ : such that $L_j \preceq L_i$ holds$\}$, where $L_i$ is the prohibited dimension level.

As shown in Example 1, a policy may restrict the user from accessing *any* of the values of a given level or levels. However, there are times when this approach is too coarse. Instead, we would like to also have a less restrictive mechanism that would only prevent the user from accessing a specific value *within* a level(s). For instance, suppose we want to alter the policy in Example 1 to make it more specific. The new policy might look like the following:

*Example 2:* Alice should not know the sales total for the province of Quebec.

In Example 2, we see that Alice may view sales totals for all provinces other than Quebec. However, Alice can still compute the Quebec sales by summing the sales of individual Quebec cities, or by summing the sales of Quebec's many stores. In other words, she can use the values of the lower levels to compute the prohibited value. Hence, all these values should also be protected. To determine the list of restricted member values, our model adds the *Under* function. Figure 5 (b) provides an example using *Under(Quebec)*. Here, all the values surrounded by the dashed line should be protected.

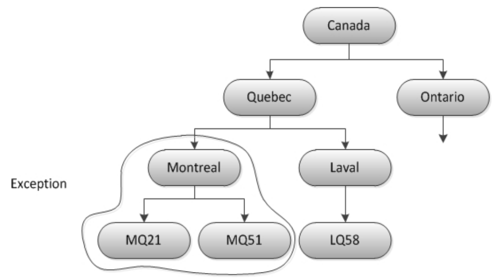


Figure 6. An Authorization Exception.

Finally, it is also possible that *exceptions* to the general authorization rule are required. For instance, Alice should not know the sales of stores in the province of Quebec except for the stores in the city/region she manages (e.g., Montreal). Figure 6 graphically illustrates this policy. In this case, the circled members represent the values associated with the exception that would, in turn, be contained within a larger encapsulating restriction. Note that a user may have one or more exceptions on a given hierarchy. The formalization of the exception object is given in in Definition 3.

*Definition 3:* For any prohibited level $L_i$, there may be an Exception E such that E contains a set Ev of values belonging to Under($L_i$). That is, Ev $\in$ values of Under($L_i$).

To summarize, authorization objects consist of the values of the prohibited level and all the levels below it, excluding zero or more exception value(s). We formalize the concept of the *Authorization Object* in Definition 4.

*Definition 4:* An Authorization Object O = $\{$v : v $\in$ Under($L_i$) - Ev$\}$, where $L_i$ is the prohibited level, and Ev is the exception value.

## D. Authorization Rules

We now turn to the query authorization process itself. As noted above, pre-compiled queries are encoded internally in XML format. For the sake of simplicity (and space constraints), we will depict the received queries in a more compact form in this section. For example, Listing 3 represents the same query shown in Listing 2. Note that the query is divided into three elements: the SELECTION element, the CONDITION element, and the FROM element. The SELECTION element lists all attributes and measures the user wants to retrieve. The CONDITION element, in turn, limits or filters the data we fetch from the cube. Finally, the FROM element indicates the cube from which data is to be retrieved.

In the discussion that follows, we will assume the existence of a cube corresponding to Figure 1. That is, the cube has three dimensions (Product, Store, and Time). Dimension hierarchies include Product_Number $\preceq$ Type $\preceq$ Category for Product, Store_Number $\preceq$ City $\preceq$ Province $\preceq$ Country for Store, and Month $\preceq$ Year for Time. Selection

```
Selection:
   Store.Province, SUM(sales)
Condition:
   Time.year = 2011 AND
   Store.Province = 'Quebec'
From:
   Sales
```

Listing 3. A Query in Simple Form

operations correspond to the identification of one or more cells associated with some combination of hierarchy levels.

One of the advantages of building directly upon the OLAP conceptual model and its associated algebra is that it becomes much easier to represent, and subsequently assess, authorization policies. Specifically, we may think of policy analysis in terms of Restrictions, Exceptions, and Level Values that form a bridge between the algebra and the Authentication DB. There are in fact four primary *policy classes*, as indicated in the following list:

1) $L_i$ Restriction + No Exception
2) $L_i$ Restriction + Exception
3) Restriction on a specific value P of level $L_i$ + no Exception
4) Restriction on a specific value P of level $L_i$ + Exception

As mentioned, the query must be validated before execution. If validation is successful, then it can be executed as originally specified. Otherwise, the query is either rejected or rewritten according to a set of *transformation rules*. In the remainder of this section, we describe the four policy classes and the processing logic relevant to each.

*1) **Policy Class 1**: $L_i$ Restriction + No Exception:* If a user is prohibited from accessing level $L_i$ and the user has no exception(s), then the authorization objects consist of the values of level $L_i$ and all the levels below it. In short, this means that if the user query specifies level $L_i$ or any of its children in the SELECTION element, then the query should simply be rejected. Moreover, if any *value* belonging to the $L_i$ level or any of its children is specified in the CONDITION element of the query, the query should also be rejected. The formalization of the rule and an illustrative example is given below.

***Rule 1.*** *If a user is prohibited from accessing the values of level $L_i$, and there is no exception, then the Authorization Objects (O) = $\{$v : v $\in$ Below($L_i$) $\}$.*

*Example 3:* If Alice sends the query depicted in Listing 4, which summarizes the total sales of Canada's stores in 2011, the query should be rejected.

Why is this query rejected? Recall that Alice is restricted from accessing provincial sales. Consequently, we see that an implicitly prohibited child level (i.e., City) is a component of the SELECTION element. So, if we allow this query, Alice

```
Selection:
    Store.City, SUM(sales)
Condition:
    Time.year = 2011 AND
    Store.Country = 'Canada'
From:
    Sales
```

Listing 4. Rule 1 example

```
Selection:
    Store.province, SUM(sales)
Condition:
    Time.year = 2011 AND
    Store.City = 'Montreal'
From:
    Sales
```

Listing 5. Rule 4 example

```
Selection:
  Store.City, SUM(sales)
Condition:
  Time.Year = 2011
From:
  Sales
```

Listing 6. Simple OLAP Query 2

```
Selection:
  Store.City, SUM(sales)
Condition:
  Time.Year = 2011 AND
  Store.Provice = 'Quebec'
From:
  Sales
```

Listing 7. Rule 5 example

can in fact compute the provincial sales by summing the associated city sales.

2) ***Policy Class 2**: $L_i$ Restriction + Exception:* In this case, the authorization objects that should be protected consist of the prohibited level value and all values below it, *except* of course for the value of the exception or any value under it. Let us first formalize this case, before proceeding with a detailed description.

***Rule 2.*** *If a user is restricted from accessing the values of level $L_i$, and the user has an exception E, then the Authorization Objects (O) = $\{v : v \in Below(L_i)$ - Under(Ev) $\}$.*

As such, when a user is prohibited from accessing the $L_i$ level — excluding the *exception* values — then the query can be (i) allowed to execute, or (ii) modified before its execution. Let's look at these two cases now.

***Rule 3.*** *The query will be allowed to execute without modification* if *the prohibited level value $Lv$ or any of its more granular level values in (Below($L_i$)) exists in the* CONDITION *element AND is equal to the exception value (Ev) or any of its implied values in (Under(Ev)).*

*Example 4:* Suppose that we have the following policy: Alice is restricted from accessing provincial sales *except* the sales for Canadian provinces. If Alice resubmits the query in Listing 3, it will now be executed without modification because the prohibited value (e.g., Quebec) is *under* the exception value (e.g., Under(Canada)).

But what if Alice has an exception value only for a more detailed child level of $L_i$ (e.g., the city of Montreal)? In this case, if Alice submits the previous query, it should now be modified by replacing the restricted value (e.g., Quebec) in the CONDITION element with the exception value (e.g., Montreal). In this example, Alice gets only the values that she is allowed to see. The modified query is depicted in Listing 5. Rule 4 gives the formalization of this case.

***Rule 4.*** *If the prohibited level value $Lv$ or any of its more granular level values (Under($Lv$)) exists in the* CONDITION *element, and the exception value belongs to this set of values, then the query should be modified by replacing the prohibited value with the exception value.*

In addition to the scenario just described, the query can also be modified by adding a new predicate to the CONDITION element when the prohibited level or any of its child levels exists in the SELECTION element only.

***Rule 5.*** *If the prohibited level $Lv$ or any of its more granular levels (Below($L_i$)) exists in the* SELECTION *element only, then the query should be modified by adding the exception $E$ as a new predicate to the query.*

*Example 5:* Suppose that Alice sends the query depicted in Listing 6. In this case, the query will be modified by adding a new predicate (i.e., Store.Province = 'Quebec'), because the prohibited level (i.e., City) exists in the SELECTION element. After the modification, Alice will see only the cities of Quebec. The modified query is depicted in Listing 7.

The complete processing logic for Policy Class 2 (i.e., Rule 3, Rule 4, and Rule 5) is encapsulated in Algorithm 1. Essentially, the algorithm takes the prohibited level $L_i$ and the exception $E$ as input and produces as output an authorization decision to execute or modify the query. The process is divided into two main parts or conditions. In the first case, we are looking at situations whereby the prohibited level $L_j$ exists in the query CONDITION element. Here, the query can either be allowed to execute directly or further modified. It executes directly if the prohibited value *Lv* is equal to the exception value *Ev* or any value under *Ev*. However, if the exception value *Ev* is equivalent to any value under *Lv*, then the query is modified by replacing the prohibited level with the exception level AND the prohibited level value with the exception value.

In the second case, we target the scenario whereby the prohibited level $L_j$ exists in the SELECTION element only. Here, we modify the original query by adding the exception *E* as a new condition.

```
input  : The prohibited level $L_i$ and the exception $E$
output: Decision to directly execute or modify
Let Ev = $E$ value;
foreach level $L_j \in$ Below($L_i$) do
    if $L_j$ exists in the query CONDITION element then
        Let Lv = $L_j$ value;
        if Lv == Ev OR Lv $\in$ Under(Ev) then
            Allow the query to execute without modification;
        end
        else if Ev $\in$ Under(Lv) then
            Replace $E$ by $L_j$, and Ev by Lv, then inform the user, and allow the query to execute;
        end
    end
    else if $L_j$ exists only in the query SELECTION element then
        Add $E$ as new condition to the user query, inform the user, and allow the query to execute;
    end
end
```

**Algorithm 1**:

3) ***Policy Class*** *3: Restriction on a specific value P of level $L_i$ + no Exception:* We now turn to the classes in which specific values at a given level are restricted, as opposed to all members at a given level. We begin with the simplest scenario.

***Rule 6.*** *If a user is prohibited from accessing a specific value P of level $L_i$, and the user has no exceptions, then the Authorization Objects(O)= $\{v : v \in P \cup$ Under(P) where P is the prohibited value$\}$.*

Here, the prohibited value *P*, or some value under *P*, exists in the query CONDITION element. As per Rule 6, the query should simply be rejected. But what if $L_i$ exists in the SELECTION element only? In this case, the query should be modified by adding the prohibited value as a new predicate to the query CONDITION element. Let's look at the following example.

*Example 6:* Suppose that Alice is restricted from accessing Quebec's sales. If Alice sends the query depicted in Listing 8, the query should be modified as shown in Listing 9.

```
Selection:
    Store.Province, SUM(sales)
Condition:
    Time.year = 2011
From:
    Sales
```

Listing 8. Simple OLAP Query 3

```
Selection:
    Store.Province, SUM(sales)
Condition:
    Time.year = 2011 AND
    Store.Province != 'Quebec'
From:
    Sales
```

Listing 9. Rule 7 example

The associated query summarizes the sales of provinces in 2011. As noted, the SELECTION element contains the prohibited level (Province), so instead of rejecting the query we modify it by adding a new predicate to the condition. The modified query returns only the sales that Alice is allowed to see. The logic is formalized in Rule 7 below.

***Rule 7.*** *If the prohibited level $L_i$ exists in the* SELECTION *element only, then the query should be modified by adding a new predicate to the query* CONDITION *element.*

4) ***Policy Class 4****: Restriction on a specific value P of level $L_i$ + Exception:* Finally, we add an exception to the queries described by Class 3. Here, the relevant authorization objects consist of the prohibited value *(P)*, *minus* the exception values.

***Rule 8.*** *If a user is restricted from accessing a value P of level $L_i$, and the user has an exception E, then the Authorization Objects(O)= $\{v : v \in$ (P $\cup$ Under(P)) - (Ev $\cup$ Under(Ev))$\}$ where P is the prohibited value and E is the exception.*

In this scenario, the query can either be allowed to execute or modified according to the following associated rules.

***Rule 9.*** *The query will be allowed to execute, if the prohibited value Lv exists in the* CONDITION *element AND is equal to the exception value Ev or any value Under(Ev).*

```
Selection:
    Store.City, SUM(sales)
Condition:
    Store.City = 'Montreal'
From:
    Sales
```

Listing 10. Rule 9 example

*Example 7:* Suppose that Alice is restricted from accessing the sales of Canadian provinces, *except* for the sales of Quebec. If Alice sends the Query depicted in Listing 10, the query will be allowed to execute since the prohibited value (i.e., Montreal) is under the exception value (i.e., Quebec).

***Rule 10.*** *If the prohibited level $L_i$ exists in the query* SELECTION *element only, the query will be modified by adding the exception E as a new predicate. In principle, this rule is similar to Rule 4.*

***Rule 11.*** *When Lv exists in the query* CONDITION *element AND Lv is under Ev, the query is modified by replacing the prohibited level $L_i$ by the exception level E AND the prohibited level value Lv by the exception value Ev.*

Algorithm 2 illustrates the full processing logic for Policy Class 4 (Rule 8, Rule 9, Rule 10, and Rule 11). In short, the authentication module takes the prohibited level value *P* and the exception *E* as input and gives as output an authorization decision to execute or modify the query. The algorithm is again divided into two main parts. The first component targets the case whereby the prohibited value *P* exists in the query CONDITION element. Here, the query can be modified or executed directly. If the prohibited value belongs to the set of values under *E* , the query is modified by replacing the condition that contains the prohibited value by a new one containing the exception. Conversely, the query is allowed to execute directly if the prohibited level value *Lv* belongs to the values *Under(P)* AND *Lv* is equal to the exception value *Ev* OR *Ev* belongs to the values *Under(Lv)*.

In the second case, a new condition (exception *E*) is added to the query CONDITION element when the prohibited level *Lv* or any level below it *Below(Lv)* exists in the SELECTION element only.

```
input  : The prohibited value P of level $L_i$ and the
         exception E
output: Decision to directly execute or modify
Let Ev = E value;
foreach level $L_j \in$ Below($L_i$) do
    if $L_j$ exists in the query CONDITION element
    then
        Let Lv = $L_j$ value;
        if (Lv == P) AND (P $\in$ Under(Ev)) then
            Add E as a new condition instead of the
            condition that contains $L_j$, inform the
            user, and allow the query to execute;
        end
        else if (Lv $\in$ Under(P)) AND (Lv == Ev
        OR Ev $\in$ Under(Lv)) then
            Allow the query to execute without
            modification;
        end
    end
    else if $L_j$ exists only in the query SELECTION
    element then
        Add E as new condition to the user query,
        inform the user, and allow the query to
        execute;
    end
end
```

**Algorithm 2**:

*E. Authorization Rule Summary*

The preceding sections have formalized the authentication framework in terms of four policy classes and their associated transformation rules. Below, we summarize the authorization decision in terms of its three possible outcomes — **Execute**, **Modify**, **Reject**:

1) The query is allowed to execute without modification in two situations:
   - Level $L_i$ is restricted and there is an exception *E*:
     a) If any upper level exists in the SELECTION or PROJECTION query element, OR
     b) If the $L_i$ value or any value from the levels below it exists in the CONDITION element AND this value is equal to the exception value *Ev* or any value under it.
   - A specific value of $L_i$ is restricted and there is an exception *E*:
     a) If the prohibited value *Lv* or any value under it exists in the CONDITION element AND it is equal to the exception value *Ev* OR any value under it.
2) The query is modified in one situation:
   - A level $L_i$ is restricted and there is an exception *E*:
     a) If level $L_i$ or any value from the levels below it exists in the query SELECTION element only, then we add the exception *E* as a new condition, OR
     b) If the exception value *Ev* belongs to the values under *Lv*, then we replace the prohibited level in the CONDITION element by the exception *E*.
3) The query is rejected in two situations:
   - A level $L_i$ is restricted, and there is no exception:
     a) If level $L_i$ or any value from a lower level exists in the SELECTION element only, OR
     b) If level $L_i$ or any value from the levels below it exists in the CONDITION element.
   - A specific value *P* is restricted, and there is no exception:
     a) If *P* or any value under it exists in the CONDITION element.

### F. A note on Performance

As noted above, the authorization framework has been incorporated into a DBMS prototype specifically designed for OLAP storage and analysis. In practice, the authorization logic has a negligible impact on performance (less than a few milliseconds for the queries presented in this paper). Specifically, the decomposition of the user query into its algebraic components (and conditions) is performed by the underlying query engine; the framework simply *borrows* the result as input to the authorization process. Moreover, the analysis of policy classes is based upon a fixed set of IF/ELSE cases that, in turn, manipulate a small in-memory Authentication Database. The run-time impact of this analysis is completely dominated by the cost of answering the (validated) query.

## V. Conclusions and Future Work

In this paper, we have discussed a query re-writing model to provide access control in multi-dimensional OLAP environments. We began by defining a conceptual model that focused on the data cube and its constituent dimension hierarchies. From there we introduced the notion of authorization objects designed to identify and constrain the relationships between parent/child aggregation levels. We then presented a series of rules that exploited the authorization objects to decide whether user queries should be rejected, executed directly, or dynamically and transparently transformed. In the latter case, we identified a set of minimal changes that would allow queries to proceed against a subset of the requested data.

We note that while the current authentication and authorization framework has been integrated into a prototype DBMS that provides OLAP-specific indexing and storage, we believe that the general principles are broadly applicable to any DBMS product that understands the fundamental data cube model. Exploiting the proposed framework would allow such systems to significantly simplify the process of designing and enforcing OLAP security policies by associating authorization decisions with an intuitive conceptual model rather than the low level logical model of relational DBMSs.

Finally, it is important to point out that the framework presented in this paper cannot block all attempts to access restricted data. In particular, it is possible for a user possessing some degree of external knowledge to combine the results of multiple *valid* queries to obtain data that is itself meant to be protected. We refer to such exploits as *inference* attacks. We are currently working on inference detection mechanisms that will piggy back on top of the core authentication and authorization framework to provide an even greater level of security for OLAP data.

## References

[1] T. H. Davenport and J. G. Harris, “Competing on analytics: The new science of winning,” in *Harvard Business School Press*, 2007.

[2] E. Fernández-Medina, J. Trujillo, R. Villarroel, and M. Piattini, “Developing secure data warehouses with a UML extension,” *Information Systems*, vol. 32, pp. 826–856, 2007.

[3] J. Trujillo, E. Soler, E. Fernandez-Medina, and M. Piattini, “A UML 2.0 profile to define security requirements for data warehouses,” *Computer Standards & Interfaces*, vol. 31, pp. 969–983, 2009.

[4] A. Rosenthal and E. Sciore, “View security as the basic for data warehouse security,” in *International Workshop on Design and Management of Data Warehouse*, 2000, pp. 8.1–8.8.

[5] C. Blanco, E. Fernandez-Medina, J. Trujillo, and M. Piattini, “How to implement multidimensional security into OLAP tools,” *International Journal of Business Intelligence and Data Mining*, vol. 3, pp. 255–276, 2008.

[6] C. Blanco, I. G.-R. de Guzman, D. Rosado, E. Fernandez-Medina, and J. Trujillo, “Applying QVT in order to implement secure data warehouses in SQL Server Analysis Services,” *Journal of Research and Practice in Information Technology*, vol. 41, pp. 135–154, 2009.

[7] J. Trujillo, E. Soler, E. Fernández-Medina, and M. Piattini, “An engineering process for developing secure data warehouses,” *Information and Software Technology*, vol. 51, pp. 1033–1051, 2009.

[8] N. Katic, G. Quirchmay, J. Schiefer, M. Stolba, and A. Tjoa, “A prototype model for data warehouse security based on metadata,” in *DEXA*, 1998, pp. 300–308.

[9] A. Deshpande, Z. Ives, and V. Raman, “Adaptive query processing,” *Foundations and Trends in Databases*, vol. 1, pp. 1–140, 2007.

[10] S. Rizvi, A. O. Mendelzon, S. Sudarshan, and P. Roy, “Extending query rewriting techniques for fine-grained access control,” in *ACM SIGMOD*, 2004, pp. 551–562.

[11] “The Virtual Private Database,” June 2011, http://www.oracle.com/technetwork/database/security/index-088277.html.

[12] “Microsoft Analysis Services,” June 2011, http://www.microsoft.com/sqlserver/2008/en/us/analysis-services.aspx.

[13] J. Gray, A. Bosworth, A. Layman, D. Reichart, and H. Pirahesh, “Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals,” *Data Mining and Knowledge Discovery*, vol. 1, pp. 29–53, 1997.

[14] V. Harinarayan, A. Rajaraman, and J. Ullman, “Implementing data cubes efficiently,” in *ACM SIGMOD*, 1996, pp. 205–227.

[15] E. Malinowski and E. Zimányi, “Hierarchies in a multidimensional model: from conceptual modeling to logical representation,” *Data and Knowledge Engineering*, vol. 59, pp. 348–377, 2006.

[16] T. Eavis, H. Tabbara, and A. Taleb, “The NOX framework: native language queries for business intelligence applications,” in *DaWak*, 2010, pp. 172–189.

[17] “SQL database engine,” June 2011, http://www.sqlite.org.

# Basic Components for Building Column Store-based Applications

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*

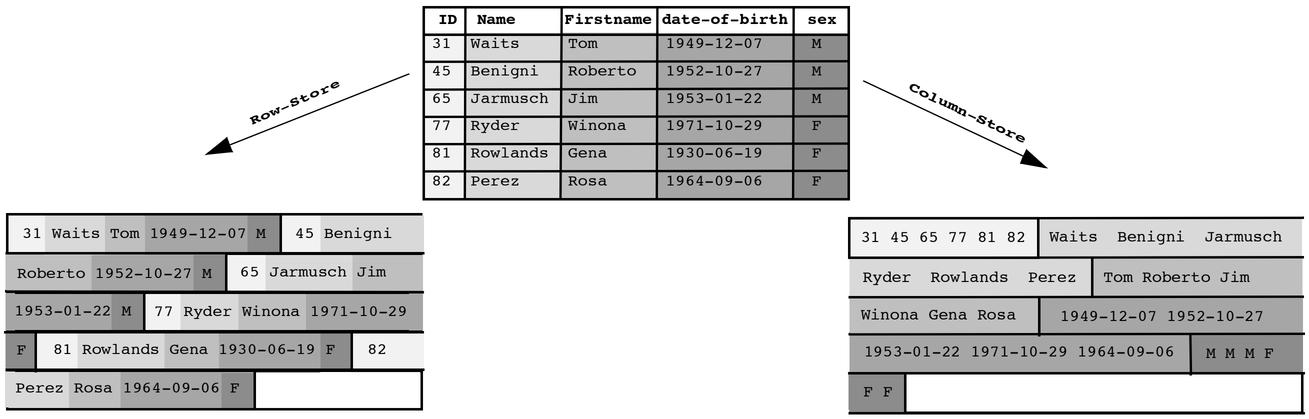
***Abstract*—A constantly increasing CPU-memory gap as well as steady growth of main memory capacities has increased interest in column store systems due to potential performance gains within the realm of database solutions. In the past several monolithic systems have reached maturity in the commercial and academic space. However a framework of low-level and modular components for rapidly building column store based applications has yet to emerge. A possible field of application is the rapid development of high-performance components in various data-intensive areas such as text-retrieval systems. The main contribution of this paper is column-store-kit, a basic building block of low-level components for constructing applications based on column store principles. We present a minimal amount of necessary structural elements and associated operations required for building applications based on our column-store-kit.**

***Keywords*-Column store; basic components; framework; rapid prototyping.**

## I. Introduction

Within database systems, values of a dataset are usually stored in a physically connected manner. A *row store* stores all column values of each single row consecutively (see Figure 1, bottom left). In contrast to that, within a *column store*, all values of each single column are stored one after another (see Figure 1, bottom right). In column stores, the relationship between individual column values and their originating datasets are established via Tuple IDentifiers (TID). The main advantage of column stores during query processing is the fact that only data from columns which are of relevance to a query have be loaded To answer the same query in a row store, all columns of a dataset have to be loaded, despite the fact, that only a small portion of them are actually of interest to the processing. On the other side, the column store architecture is disadvantageous for frequent changes (in particular insertions) to datasets. As the values are stored by column, they are distributed at various locations, which leads to a higher number of required write operations exceeding those within a row store to perform the same changes. This characteristic makes this type of storage interesting especially for applications with very high data volume and few sporadic changes only (preferably as bulk upload), as it is the case in, e.g., data warehouses, business intelligence systems or text retrieval systems. Interest in column store systems has recently been reinforced by steady growth of main memory capacities that meanwhile allow for main memory-based database solutions and additionally by the constantly increasing CPU-memory gap [1]: Today's processors can process data much quicker than it can be loaded from main memory into the processor cache. Consequently, modern processors for database applications spend a major part of their time waiting for the required data. Column stores and special cache-conscious [2] algorithms are attempts to avoid this "waste of time". A number of commercial and academic column store systems have been developed in the past. In the research area, MonetDB [3] and C-Store [4] are widely known. Open Source and commercial systems include Sybase IQ, Infobright, Vertica, LucidDB, and Ingres. All these systems are more or less complete database systems with an SQL interface and a query optimizer.

As column stores are a young field of research, numerous aspects remain to be examined. For example, separation of datasets into individual columns result in a series of additional degrees of freedom when processing a query. Abadi et al. [5] developed several strategies as to when a result is to be "materialized", i.e., at which point in time result tuples shall be composed. Depending on the type of query and selectivity of predicates, an early or late materialization may be reasonable. Interesting studies were published about compression methods [6], various index types as well as the execution of join operations, e.g., Radix-Join [1], Invisible Join [7] or LZ-Join [8]. In addition to that, there are attempts at creating hybrid approaches that try to combine the advantages of column and row stores. The main objective of this paper is to present a number of low-level building blocks for constructing applications based on column store systems. Instead of copying the low-level constructs of existing sophisticated column stores, our research work is focused on identifying components



| 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 |

Row-Store
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
Column-Store
31 45 65 77 81 82 Waits Benigni Jarmusch
Ryder Rowlands Perez Tom Roberto Jim
Winona Gena Rosa 1949-12-07 1952-10-27
1953-01-22 1971-10-29 1964-09-06 M M M F
F F

Figure 1. Comparison of the layouts of a row store and a column store

and operations that allow for building specialized column store based applications in rapid prototyping fashion. As our components can be composed in a “plug and compute”-style, our contribution is column-store-kit, which is a building block for experimental and prototypical setup of applications within the field of column stores. A possible field of application is the rapid development of high-performance components in various data-intensive areas such as text-retrieval systems.

The paper is structured as follows: In the next section, requirements for our column-store-kit will be outlined. Then, the identified components and corresponding operations will be explained on a logical level. On this basis, various implementations of logical components and operations will be presented. Finally, results will be summarized and an outlook will be given on future research activities.

## II. Column Store Principles

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 main memory, it is first checked, whether it already resides within 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 than further 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, that requests to subsequent items are much faster, because they already reside within the cache. Meanwhile, the speed gain when accessing a dataset in the first-level cache is up to two orders of magnitude compared to regular main memory access [9]. Column stores take advantage of this prefetching behaviour, because values of individual columns are physically connected together and therefore often already reside in the cache when requested, as the execution of complex queries is processed column by column rather than dataset by dataset. This also means that the decision whether a dataset fulfills a complex condition is generally delayed until the last column is processed. Consequently, additional data structures are required to administrate the status of a dataset in a query. These data structures are referred to as *Position List*s. A *PositionList* stores the TIDs of matching datasets. Execution of a complex query generates a *PositionList* with entries of the qualified datasets for every simple predicate. Then, the *PositionList*s are linked by *and*/*or* semantics. As an example, Figure 2 shows a possible execution plan for the following query:

```
select name
  from person
 where birthdate < '1960-01-01'
   and sex='F'
```

First, the predicates $birthdate$ <’1960-01-01’ and $sex$ =’F’ must be evaluated against the correponding columns (birthdate and sex) which results in the *PositionList*s *PL1* and *PL2*. These two evaluations could also be done in parallel. Next an *and*-operation must be performed on these two *PositionList*s, resulting in the *PositionList PL3*. As we are interested in the names of the persons that fulfil the query conditions, we have to perform another operation (extract), which finally returns the entries for a TID, specified by the *PositionList PL3*.

## III. Concept

The main focus of our components is modeling the individual columns, which can occur both in the secondary store as well as 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

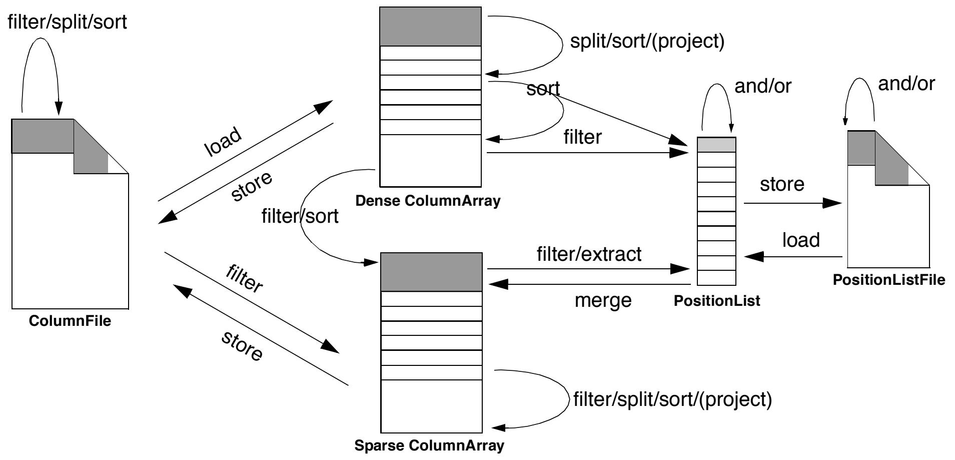


Figure 3. Components and Operations

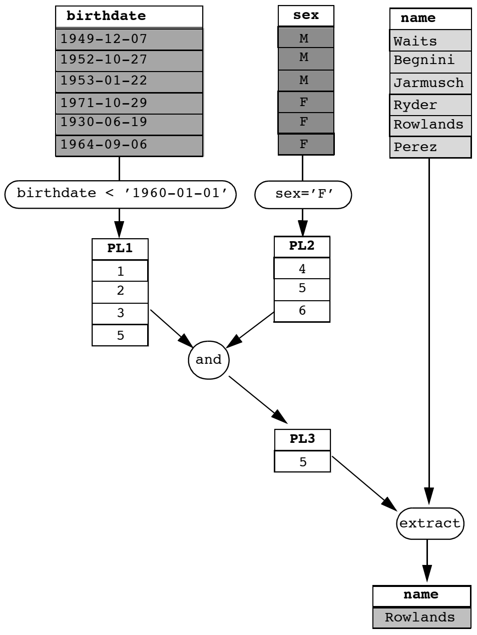


Figure 2. Processing of a query with PositionLists

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 *PositionList* presented in the previous Section II. Just like columns, two different representation forms are available for main and secondary storage. 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 *PositionLists* are applied as filtering arguments. Figure 3 illustrates a high level overview of the most important operations and transformations between the components. In Section IV they will be described in detail. Moreover, the components are to be developed for use on both secondary store and main memory as well as designed for maximum performance. This particularly implies the use of cache-conscious algorithms and structures.

## IV. Presentation of Logical Components

In the following sections, the aforementioned components will be presented together with their structure and their corresponding operations. Section V will then outline potential implementations to reach highest possible performance.

### A. Structure

*1) ColumnFile:* The *ColumnFile* serves to represent a column on the secondary storage. Supported primitive data types are: uint, int, char, date und float. Moreover, the composite type *SimpleStruct* (see below) is supported, which may consist of a runtime definable list of the previously mentioned primitive data types. As a standard, the TID of a value in the *ColumnFile* is given implicitly by the position of the value in the file. If this is not the case, a *SimpleStruct* is used, which explicitly contains the TID in the first column.

*2) SimpleStruct: SimpleStruct* is a dynamic, runtime definable data structure. It is used within *ColumnFile* as well as within *ColumnArrays* (see below). The *SimpleStruct* plays a role in the following cases:

- Result of a filter query, in which the TIDs of the original datasets are also given.
- Combination of results consisting of several columns.
- Setup of hybrid systems having characteristics of both column and row stores. For example, it may be advan-

tageous to store several attributes in a *SimpleStruct* that are frequently requested together.

- Representation of sorted columns, where TIDs are required. This is particularly reasonable for Join operators or a run-length-encoded compression on their basis.

*3) ColumnArray and MultiColumnArray:* A *ColumnArray* represents a column in main memory, which consists of a flexible number of lines. The data types correspond to those of the previously defined *ColumnFile*. If the data type is a *SimpleStruct*, it is referred to as *MultiColumnArray*. In addition to the actual column values, the TIDs of the first and last dataset and the number of datasets stored are given in the header of the *(Multi)ColumnArray*. Two types of representations are distinguished:

- **Dense:** The type of representation is dense, if no gaps can be found in the datasets, i.e., if the TIDs are consecutive. In this case, the TID is given virtually by the TID of the first data set and the position in the array and does not have to be stored explicitly (Figure 4, left side). This type of representation is particularly suited for main memory-based applications, in which all datasets (or a continuous section of them) are located in main memory.
- **Sparse:** This type of representation explicitly stores the TIDs of the datasets (Figure 4, right). The primary purpose of a sparse *ColumnArray* is the storage of (intermediate) results. As will be outlined in more detail in Section V, it may be chosen between two physical implementations depending on the concrete purpose.

*4) ColumnBlocks and MultiColumnBlocks:* Apart from the *(Multi)ColumnArrays* of flexible size, *(Multi)ColumnBlocks* exist, which possess a random, but fixed size. They are mainly used to implement ColumnArrays with their flexible size. In addition, they may be applied in the implementation of an custom buffer management as a transfer unit between secondary and main memory and as a unit that can be indexed.

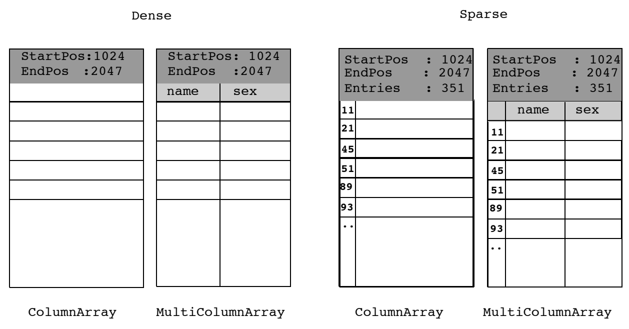


Figure 4. Types of ColumnArrays

*5) PositionList:* A *PositionList* is nothing else than a *ColumnArray* with the data type *uint(4)* as far as structure is concerned. However, it has a different semantics. The *PositionList* stores TIDs. A position list is the result of a query via predicate(s) on a *ColumnFile* or a *(Multi)ColumnArray*, where the actual values are of no interest, but rather the information about the qualified data sets. *Position Lists* store the TIDs in ascending order without duplicates. This makes the typical and/or operations very fast, as the cost for both operations is $O(|Pl1| + |Pl_2|)$. As will be outlined in Section V, various types of implementations may be applied. Analogously to the *(Multi)ColumnArray*, there is a representation of the *PositionList* for the secondary store, which is called *PositionFile*.

## B. Operations

*1) Transformations on ColumnFiles:* Several operations are defined on *ColumnFiles*. A filter operation (via predicate and/or PositionList) can be performed on a *ColumnFile* and the result can be written to another *ColumnFile* (with or without explicit TIDs). Other operations are the splitting of a *ColumnFile* as well as sorting among different criterias (see Section IV-B6) with and without explicitly storing the TID.

*2) Transformations between ColumnFile and (Multi)-Column-Array: ColumnFiles* and *(Multi)ColumnArrays* are different types of representation of one or more logical columns. Physically, ColumnFiles are located in the secondary storage, while *ColumnArrays* are located in main memory. Consequently, both types of representations can also be transformed into each other using the corresponding operators.

A *ColumnFile* can be transformed completely or partially into a dense *(Multi)ColumnArray*. If not all, but only certain datasets that match special predicates or *PositionLists* are to be loaded into a *(Multi)ColumnArray*, this can be achieved using filter operations that generate a sparse *(Multi)ColumnArray*. A sparse *(Multi)ColumnArray* may also be transformed into a *ColumnFile*. In this case, the TIDs are stored explicitly in combination with the values. Other operations refer to the insertion of new values and the deletion of values. An outline of the most important operations of *ColumnFiles* is given in Table I.

*3) Operations on ColumnArrays:* Filter operations can be executed on *(Multi)ColumnArrays* using predicates and/or *PositionLists*. This may result in a sparse *(Multi)ColumnArray* or a *PositionList*. Furthermore, *ColumnArrays* may also be linked with each other by and/or semantics. If the *(Multi)ColumnArrays* have the same structure, the result also possesses this structure. The results correspond to the intersection or union of the original datasets. The result is a sparse *(Multi)ColumnArray*. If *(Multi)ColumnArrays* of differing structure are to be combined, only the and operation is defined. The result is a *(Multi)ColumnArray* that contains a union of all columns of the involved *(Multi)ColumnArrays* and returns the values for the datasets

Table I
OUTLINE OF OPERATIONS ON COLUMNFILES

| Operation | Result type |
|---|---|
| read(ColumnFile) | ColumnArray (dense) |
| read(ColumnFile, start, length) | ColumnArray (dense) |
| filter(ColumnFile, predicate) | ColumnArray (sparse) |
| filter(ColumnFile, predicate-list) | ColumnArray (sparse) |
| filter(ColumnFile, positionlist) | ColumnArray (sparse) |
| filter(ColumnFile, positionlist-list) | ColumnArray (sparse) |
| filter(ColumnFile, predicate-list, positionlist-list) | ColumnArray (sparse) |
| fileFilter(ColumnFile, predicate) | ColumnFile (explicit TIDs) |
| fileFilter(ColumnFile, predicate-list) | ColumnFile (explicit TIDs) |
| fileFilter(ColumnFile, positionlist) | ColumnFile (explicit TIDs) |
| fileFilter(ColumnFile, positionlist-list) | ColumnFile (explicit TIDs) |
| fileFilter(ColumnFile, predicate-list, positionlist-list) | ColumnFile (explicit TIDs) |
| split(ColumnFile, predicate) | ColumnFile, ColumnFile |
| split(ColumnFile, position) | ColumnFile, ColumnFile |
| sort(ColumnFile, column(s), direction) | ColumnFile |
| sort(ColumnFile, Orderlist) | ColumnFile |
| insert(ColumnFile, value) | Tupel-ID |
| delete(ColumnFile, Tupel-ID) | boolean |
| delete(ColumnFile, Positionlist) | integer |
| delete(ColumnFile, predicate) | integer |
| delete(ColumnFile, predicate-list) | integer |

having identical TIDs. If the *(Multi)ColumnArray*s used as input are dense and if they have the same TID interval, the resulting *MultiColumnArray* is also dense. An outline of the most important operations of *ColumnArray*s is given in Table II. *ColumnArray* may also refer to a *MultiColumnArray*. A *MultiColumnArray*, however, only refers to the version having several columns.

*4) Transformation from PositionList to ColumnArray:* If the column values of the stored TIDs inside a *PositionList* are needed, an *extract* operation must be performed. Input to this operation is a *PositionList* as well as a *dense (multi) ColumnArray*. The result is a *sparse (Multi) ColumnArray*.

*5) Operations between PositionLists:* Several *PositionList*s may be combined by *and*, *or* semantics, with the result being a *PositionList*. The result list is sorted in ascending order corresponding to the TIDs. In addition, operations exist to load and store *PositionList*s. An outline of operations of *PositionList*s can be found in Table III.

*6) Sorting:* One basic operation on *(Multi) ColumnArray*s as well as *ColumnFiles* is sorting. Beside the obvious task to bring the result of a query in a specific order, sorting also plays an important role regarding performance considerations. For the elimination of duplicates, for join operations and for compression using run-length encoding, previous sorting can dramatically improve performance. As a consequence of sorting, the natural order is lost. This is critical for dense columns with implicit TIDs, because the relation to the other column values is lost. The problem can be solved by an additional data structure, similar to a *PositionList* which contains the mapping information to the orginal order of the datasets. Figure 5 gives an example of

Table II
OUTLINE OF OPERATIONS ON *ColumnArrays*

| Operation | Result type |
|---|---|
| filter(ColumnArray, predicate) | ColumnArray (sparse) |
| filter(ColumnArray, predicate-list) | ColumnArray (sparse) |
| filter(ColumnArray, positionlist) | ColumnArray (sparse) |
| filter(ColumnArray, positionlist-list) | ColumnArray (sparse) |
| filter(ColumnArray, predicate-list, positionlist-list) | ColumnArray (sparse) |
| filter(ColumnArray, predicate) | PositionList |
| filter(ColumnArray, predicate-list) | PositionList |
| filter(ColumnArray, positionlist) | PositionList |
| filter(ColumnArray, positionlist-list) | PositionList |
| filter(ColumnArray, predicate-list, positionlist-list) | PositionList |
| and(ColumnArray, ColumnArray) | ColumnArray |
| or(ColumnArray, ColumnArray) | ColumnArray |
| and(ColumnArray, ColumnArray) | PositionList |
| or(ColumnArray, ColumnArray) | PositionList |
| project(MultiColumnArray, columns) | (Multi)ColumnArray |
| asPositionList(ColumnArray, column) | PositionList |
| split(ColumnArray, predicate) | ColumnArray (sparse), ColumnArray (sparse) |
| split(ColumnArray(dense), position) ColumnArray (dense) | ColumnArray (dense), |
| split(ColumnArray (sparse), position) ColumnArray (sparse) | ColumnArray (sparse), |
| store(ColumnArray (dense)) | ColumnFile |
| store(ColumnArray (sparse)) | ColumnFile (explicit TIDs) |

Table III
OUTLINE OF OPERATIONS ON *PositionList*S

| Operation | Result type |
|---|---|
| load(ColumnFile) | PositionList |
| store(PositionList) | ColumnFile |
| and(PositionList, PositionList) | PositionList |
| or(PositionList, PositionList) | PositionList |
| read(PositionListFile) | PositionList |
| store(PositionList) | PositionListFile |

this situation. The *Multi ColumnArray* on the left side is to be sorted according to the column “name”. Additionally to the sorting of the *MultiColumn* (top right), a list is generated which holds the information about the original positions (down right). The list can then be reused by applying it as a sorting criterion to other columns later, as shown in Figure 6.

*7) Compression:* Compression plays an important role in column stores [6], as it reduces the data volume that needs to be loaded. Nevertheless, we decided not to include compression in the first prototype. To a certain extent, this constraint can be compensated by the use of dictionary-based compression [10], which will be implemented above the basic components. In later versions, various compression methods will be integrated.

## V. IMPLEMENTATION-SPECIFIC CONSIDERATIONS

After presenting the logical structure and the required operations, this section shall now focus on considerations for achieving a performance-oriented implementation.

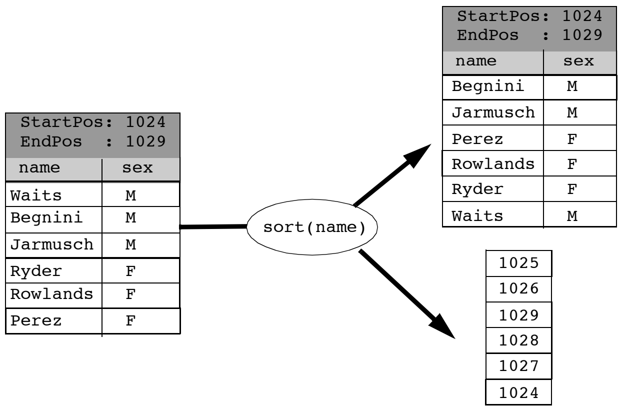


Figure 5. Sorting with explicit generation of an additional mapping list

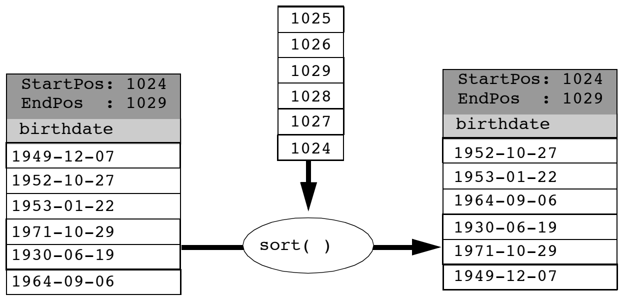


Figure 6. Sorting with explicit given sort-order

Due to the constantly increasing CPU-memory gap, cache-conscious programming is indispensable. For this reason, the implementation was made in C/C++. All time-critical parts were implemented in pure C using pointer arithmetics. The uncritical parts were implemented using C++ classes. The *ColumnBlock* was established as a basic component of the implementation. It is the basic unit for data storage. Its size is defined at creation time and it contains the actual data as well as information on its structure and the number of datasets. The structurization options correspond to those of the *(Multi)ColumnArray*. The *ColumnBlock* also handles all queries by predicates and/or *PositionList*s. A *(Multi)ColumnArray* consists of $1 - n$ *ColumnBlock* instances. All operations on a *(Multi)ColumnArray* are transferred to the underlying *ColumnBlock*s.

*PositionList*s play a central role in column store applications. If the *PositionList*s are short (i.e. if they contain a few TIDs only), representation as *ColumnArray* is ideal. Four bytes are required per selected entry. If the lists are very large, however, memory of 40 MB is required for ten million entries, for instance. In this case, a bit vector is recommended for representation. This bit vector indicates using a fixed bit whether every dataset belongs to the set of results or not. If, for example, 100 million data sets exist for a table, only 12.5 MB are required to represent the *PositionList* for certain selectivities. Moreover, the two important operations *and* and *or* can be mapped on the respective primitive processor commands, which makes the operations extremely fast. If *PositionList*s are sparse, bit vectors can be compressed very well using run-length encoding (RLE) (e.g. to a few KB in case of 0.1% selectivity). The necessary operations can be performed very efficiently on the compressed lists, which further increases the performance. An implementation based on the word-aligned hybrid algorithm [11] with satisfactory compression for medium-sparse representations was developed within the framework of the activities reported here [12], [13]. *MultiColumnArray*s may exist in two different physical layouts. In the first version, the $n$ values are written in a physically successive manner and correspond to the classical n-ary storage model (NSM). This type of representation is particularly suited, if further queries are to be performed on this *MultiColumnArray* with predicates on the respective attributes. The individual values of a dataset are stored together in the cache and all attribute values are checked simultaneously rather than successively with the help of additional *PositionList*s (Figure 7, left). The second type of representation corresponds to the PAX format [14]. Here, every column is stored in a separate *ColumnArray*. In addition, a *PositionList* is stored, which identifies the datasets (Figure 7, left). This type of representation is recommended, for instance, for collecting values for subsequent aggregation functions. Several *(Multi)ColumnArray*s may share a single *PositionList*.

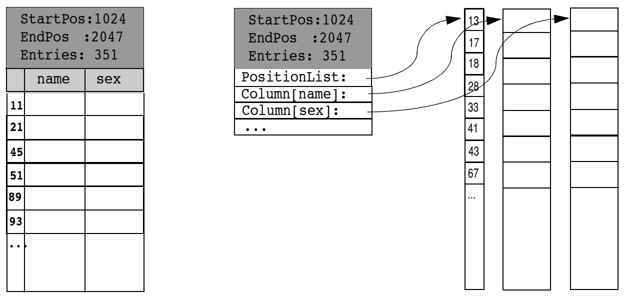


Figure 7. Comparison of storage formats for ColumnArrays

## VI. Conclusion

This paper presented a collection of basic components to build column store applications. The components are semantically located below those of the existing column store database implementations and are suited for building experimental (distributed) systems in the field of column store databases. It is planned to use these components to obtain further scientific findings in the area of column stores and to develop data-intensive applications.

## VII. Future Work

A first version of the column-store-kit is available without support for compression. The next steps planned are the integration of compression and the use in concrete areas, such as text retrieval systems. A future activity will be the implementation of a scripting language interface for the components. With the help of this interface, it will be possible to assemble the developed components more easily without losing the performance of the underlying C/C++ implementation. In this case, the scripting language act as glue between the components, allowing the developer to build up complex high performance applications with very little effort [15]. As an alternative, a custom domain-specific language (DSL) [16] may be used for building column store applications. A bachelor's thesis [17] focused on the extent to which various degrees of flexibility regarding the structure of *MultiColumnArray*s and expression of the predicates affect the performance. According to the thesis, structural definition at the time of compilation is of significant advantage compared to the runtime behavior. If the implemented flexibility of the *SimpleStruct* is not required at runtime, an alternative implementation may be used. It may be realized by defining a language extension for C/C++, for example. Thus, the respective structures and operations can be defined using a simple syntax. With a number of macros of the C++ preprocessor or a separate inline code expander [18], these could then be transformed into valid C/C++ code.

## References

[1] 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, 2000, pp. 231–246.

[2] 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.

[3] P. A. Boncz, M. L. Kersten, and S. Manegold, "Breaking the memory wall in monetdb," Commun. ACM, vol. 51, no. 12, 2008, pp. 77–85.

[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, D. S. Myers, D. J. Dewitt, and S. R. Madden, "Materialization strategies in a column-oriented dbms," in In Proc. of ICDE, 2007, pp. 466–475.

[6] 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.

[7] D. J. Abadi, S. R. Madden, and N. Hachem, "Column-stores vs. row-stores: How different are they really," in In SIGMOD, 2008, pp. 967–980.

[8] L. Gan, R. Li, Y. Jia, and X. Jin, "Join directly on heavy-weight compressed data in column-oriented database," in WAIM, 2010, pp. 357–362.

[9] P. A. Boncz, M. Zukowski, and N. Nes, "Monetdb/x100: Hyper-pipelining query execution," in CIDR, 2005, pp. 225–237.

[10] C. Binnig, S. Hildenbrand, and F. Färber, "Dictionary-based order-preserving string compression for main memory column stores," in SIGMOD '09: Proceedings of the 35th SIGMOD international conference on Management of data. New York, NY, USA: ACM, 2009, pp. 283–296.

[11] K. Wu, E. J. Otoo, and A. Shoshani, "Optimizing bitmap indices with efficient compression," ACM Trans. Database Syst., vol. 31, no. 1, 2006, pp. 1–38.

[12] A. Schmidt and M. Beine, "A concept for a compression scheme of medium-sparse bitmaps," in DBKDA'11: Proceedings of the The Third International Conference on Advances in Databases, Knowledge, and Data Applications. iaria, 2011, pp. 192–195.

[13] M. Beine, "Implementation and Evaluation of an Efficient Compression Method for Medium-Sparse Bitmap Indexes", Bachelor Thesis, Department of Informatics and Business Information Systems, Karlsruhe University of Applied Sciences, Karlsruhe, Germany, 2011.

[14] A. Ailamaki, D. J. DeWitt, and M. D. Hill, "Data page layouts for relational databases on deep memory hierarchies," The VLDB Journal, vol. 11, no. 3, 2002, pp. 198–215.

[15] J. K. Ousterhout, "Scripting: Higher-Level Programming for the 21st Century," IEEE Computer, vol. 31, no. 3, 1998, pp. 23–30.

[16] M. Mernik, J. Heering, and A. M. Sloane, "When and how to develop domain-specific languages," ACM Comput. Surv., vol. 37, no. 4, 2005, pp. 316–344.

[17] M. Herda, "Entwicklung eines Baukastens zur Erstellung von Column-Store basierten Anwendungen" Bachelor's thesis, Department of Informatics, Heilbronn University of Applied Sciences, Germany, Jun. 2011.

[18] J. Herrington, Code Generation in Action. Greenwich, CT, USA: Manning Publications Co., 2003.

# Leveraging Compression in In-Memory Databases

Jens Krueger, Johannes Wust, Martin Linkhorst, Hasso Plattner
Hasso Plattner Institute for Software Engineering
University of Potsdam
Potsdam, Germany
Email: {jens.krueger@hpi.uni-potsdam.de, johannes.wust@hpi.uni-potsdam.de,
martin.linkhorst@hpi.uni-potsdam.de, hasso.plattner@hpi.uni-potsdam.de}

***Abstract*—Recently, there has been a trend towards column-oriented databases, which in most cases apply lightweight compression techniques to improve read access. At the same time, in-memory databases become reality due to availability of huge amounts of main memory. In-memory databases achieve their optimal performance by building up cache-aware algorithms based on cost models for memory hierarchies. In this paper, we use a generic cost model for main memory access and show how lightweight compression schemes improve the cache behavior, which directly correlates with the performance of in-memory databases.**

***Keywords-in-memory databases; database compression; dictionary compression.***

## I. Introduction

Nowadays, most database management systems are hard disk based and - since I/O-operations are expensive - therefore, limited by both the throughput and latency of those hard disks. Increasing capacities of main memory that reach up to several terabytes today offer the opportunity to store an entire database completely in main memory. Besides, the much higher throughput of main memory compared to disk access significant performance improvements are also achieved by the much faster random access capability of main memory and at the same time much lower latency. A database management system that stores all of its data completely in main memory - using hard disks only for persistency and recovery – is called an in-memory database (IMDB).

In earlier work, we have shown that in-memory databases perform especially well in enterprise application scenarios [12], [14]. As shown in [12], enterprise workloads are mostly reads rather than data modification operations; this has lead to the conclusion to leverage read-optimized databases with a differential buffer for this workloads [11]. Furthermore, enterprise data is typically sparse data with a well known value domain and a relatively low number of distinct values. Therefore, enterprise data qualifies particularly well for data compression as these techniques exploit redundancy within data and knowledge about the data domain for optimal results. We apply compression for two reasons:

- Reducing the overall size of the database to fit the entire database into main memory, and
- Increasing database performance by reducing the amount of data transferred from and to main memory.

In this paper, we focus on the second aspect. We analyze different lightweight compression schemes regarding cache behavior, based on a cost model that estimates expected cache misses.

### A. The Memory Bottleneck

During the last two decades, processor speed increased faster than memory speed did [6]. The effect of this development is that processors nowadays have to wait more cycles to get a response from memory than they needed to 20 years ago. Since processors need to access data from memory for any computation, performance improvements are limited by memory latency time. As seen from a processor's perspective, main memory access becomes more and more expensive compared to earlier days – the Memory Gap widens. Nevertheless, it would be possible to manufacture memory that is as fast as a processor is but there is a direct trade-off between memory size and latency. The more capacity memory has, the longer is its latency time or - important as well - the faster memory is, the more expensive it gets. Since manufacturers concentrated on increasing capacity of main memory there wasn't much focus on improving latency times.

A solution to the problem found in modern processors is the use of a cache hierarchy to hide the latency of the main memory. Between the processors registers and main memory, a faster but smaller memory layer is placed that holds copies of a subset of data found in main memory. When a processor finds the needed data in the cache it will copy it from there waiting less processor cycles. The whole cache is usually much smaller and much faster than main memory. Since the Memory Gap widens with every new processor generation one layer of cache is not enough to fulfill both capacity and latency time demands. Therefore, modern CPUs have up to three layers of cache, each of which with more capacity but worse latency times than the one closer to the processor [8].

Since programs usually do not need to access the whole address space of main memory randomly there is the concept of locality. When a processor fetches a processor word from memory, it is very likely that it needs to fetch another word close by, so-called data locality. Leveraging that fact, processors do not only copy the requested data to its registers but also copy subsequent bytes to the cache. The amount of bytes that are copied at once to the cache is called a cache line or a cache block and usually is about four to 16 processor words long depending on the specific CPU architecture. On a current 64 bit machine, it is between 32 and 128 bytes.

Consequently, memory access is not truly random since always a complete cache line is fetched regardless the actual requested value. In the worst case, only one value out of the cache line are needed while the rest of the transferred date is polluting both the limited memory bandwidth and limited capacity on each cache. Data that are not found in the cache needs to be fetched from main memory, a so-called cache miss. There is a direct dependency between the performance of in-memory database algorithms and the number of issued cache misses as for instance described in [4], [15]. To gain significant performance improvements or to avoid performance loss, algorithms have to be cache conscious, which means that they have to efficiently use the cache and cache lines issuing as few cache misses as possible. This means data should be read sequentially from main memory instead of randomly.

Multicore processors that are supposed to work in parallel have to wait for other cores to finish their shared memory access before starting its own. Additionally, the physical distance between a processor and its cache also influences the latency time. Multicore processors' shared cache is normally placed in equal distance to each core resulting in less performance than possible on a single core chip. Intel has a solution called Non-Uniform Memory Access (NUMA), where the shared memory is logically the same but physically splitted on the chip. For example the first half of the address space is local to core one and the second half is local to core two resulting in better performance for core one when accessing addresses in the first half but worse performance for the other addresses. When core one requests data from main memory it will be fetched into an address in the first half of the address space if possible [16].

### B. In-memory databases in Enterprise Application scenarios

Today's disk-based database management systems are either optimized for transactional record-oriented or analytical attribute-oriented workload, also called Online Transaction Processing (OLTP) and Online Analytical Processing (OLAP). The distinguishment arises from enterprises that have transactional systems to support their daily business and need to answer analytical queries on top of that data. OLAP style queries are typically slow on OLTP system; therefore, enterprises usually have a separate OLAP system, e.g., a data warehouse, that stores the same information in a different way and precomputes certain values up-front to improve query performance of analytical queries. The main reason for the performance loss is that OLAP queries are attribute-focused rather than entity-focused, usually reading only a few attributes but more records, e.g., read a whole column or apply a predicate on a complete column. Most OLTP systems store their data row-oriented: a record is stored sequentially on disk and then another record follows maintained by a page layout. Since OLAP queries read only a part of many records, e.g., one attribute of each record, the needed data is not stored sequentially on disk resulting in less read performance. Furthermore, the page layout determines the access pattern that read complete pages from disk as this is the finest granularity to read a record. Due to this fact lots of unnecessary data is transferred in case a few attributes of a relation are requested. Therefore, modern OLAP systems organize the data column-oriented to improve performance of accessing whole attributes [21].

With up to several terabytes of main memory available to applications as well as the increase of computing power with new multi core hardware architectures holding entire databases in main memory becomes feasible [17]; the application of these in-memory databases is especially promising in the field of enterprise applications.

In [14], we could show that Enterprise Applications typically reveal a mix of OLAP and OLTP characteristics. In order to combine both requirements for mixed workload scenarios, the introduction of a write optimized differential buffer together with a read-optimized main storage has been proposed [7], [11], [21]. The differential buffer stores all write operations in an uncompressed manner to allow fast appends. At regular intervals, the differential buffer is merged with the main database to maintain compression and query performance. During this process the buffered values are merge into the read-optimized store as described in [11].

The merge process essentially does two things: it merges the main dictionary with the delta dictionary and keeps track of value ids that may have changed along with their new value. Then, it merges the main attribute vector of the compressed read-optimized store and the attribute vector of the differential buffer while applying the old-value-id/new-value-id mapping from the step before. That second step is not needed if value ids cannot change like in the basic dictionary or hash map approach. However, in an order-preserving dictionary approach that mapping needs to be applied taking a significant amount of clock cycles of the overall merge process. The same happens if the value id are bit compressed and a new dictionary entry make an additional bit necessary in order to represent the values.

## II. Compression

### A. Motivation

As described in the previous section, main memory latency is a bottleneck for the execution time of computations: processors are wasting cycles while waiting for data to arrive. This is especially true for databases as described in [4]. While cache conscious algorithms are one way to improve performance significantly [3], [19], [20] another option is to reduce the amount of data transferred from and to main memory, which can be achieved by compressing data [22]. On the one hand, compression reduces I/O-operations between main memory and processor registers, on the other hand it leverages the cache hierarchy more effectively, because more data fits in each cache line.

The needed processor cycles to compress and decompress data and the less wasted cycles while waiting for memory result in increased processor utilization. This increases overall performance as long as memory access time is the bottleneck.

Once compression and decompression become so processor-intensive that the processor is limiting the performance instead of the memory, compression has a negative effect on the overall execution time. Therefore, most in-memory databases use light-weight compression techniques that have low CPU overhead [1].

In addition, some operators can operate directly on compressed data - saving decompression time. Abadi et al. illustrate [2] this concept of late materialization to further improve execution speed of queries.

In order to estimate the performance improvements achieved by different compression techniques, the cost model to estimate cache misses presented in [15] is extended by taking compression into account. The basic formula of the model for an uncompressed column scan is:

$$M(s\_trav(R)) = \left\lceil \frac{R.w \cdot R.n}{B} \right\rceil \tag{1}$$

where $M(s\_trav(R))$ are the estimated cache misses on one cache level while traversing a region $R$ in memory sequentially for the first time. The parameters are $R.w$ being the width of one data item in bytes, $R.n$ which is the number of data items to traverse, as well as $B$ which is the number of data items that fit into the cache. In case of an in-memory database $R.w$ is the width of a tuple while $R.n$ is the number of tuples.

As in [15], an inclusive Level 1 cache is assumed meaning that all data that is present in the Level 1 cache is also present in the Level 2 cache. This condition may only be violated temporarily when data in the L1 is changed and marked as dirty before being written back to L2. This assumption holds for Intel CPUs but not for AMD CPUs which have an exclusive cache, meaning that data can be either in L1 or in L2 bot not in both. Modeling exclusive caches is left for future work.

In the following, we provide the cost model for various light-weight compression techniques and compare their performance for a typical analytical query size. Other atomic data patterns as the conditional traversal read [12] can be extended the same way.

## B. Run-Length Encoding

When using run-length encoding (RLE), subsequent equal values in memory are stored as a RLE-tuple of $(value, runLength)$, thus encoding a sequence of $(1, 1, 3, 3, 3, 4, 4, 4)$ as $((1, 2), (3, 3), (4, 3))$ reducing the size in memory the larger runs in the data exist. Whether a good amount of runs exist depends on two parameters: First, equal values need to be stored subsequently - this is usually the case if the column is stored in sort order of the values. Second, if the column is ordered, the number of distinct values in the data defines the number of tuples needed to be stored. However, having sorted data is much more important because a randomly ordered column with a few distinct values can contain no runs in worst case if the data is distributed equally. On the other hand, if the number of unique values is close to the number of data items, sorting the items has limited impact on compression, as there are only few runs in this case.

In a sorted run-length encoded column the main indicator of the size of the column is the cardinality of distinct values. Hence, the performance on aggregate operations in an in-memory database is mainly based on the amount of distinct values. Assuming a column's values are in sorted order and the number of distinct values of that column is given by $|D|$, the column can be encoded with $|D|$ RLE-tuples, each holding the value and the run-length. A defensive approach to determine the space needed for saving the run-length is to take the maximum run-length one value can span. Then, the maximum run-length is $R.n$, and therefore, can be encoded with $\lceil \log_2 R.n \rceil$ bits (for simplicity reasons. Actually, it is $runLength_{max} = R.n - |D| + 1$), while bit-compressing the run-length value.

The basic cost model can then be extended to take a run-length encoded column into account:

$$M(s\_trav(R)) = \left\lceil \frac{(R.w + \lceil \log_2 R.n \rceil) \cdot |D|}{B} \right\rceil \tag{2}$$

Since each tuple has the overhead of storing the run-length, run-length encoding becomes less effective as $|D|$ comes close to $R.n$. Hence, the number of distinct values is important. The break even point can be estimated with:

$$|D| = \left\lceil \frac{R.w \cdot R.n}{R.w + \lceil \log_2 R.n \rceil} \right\rceil \tag{3}$$

A generalized formula for unsorted columns encoded with run-length encoding depends on the average run-length of values in the collection which can be answered by examining the topology of the data. For example, imagine a customer table with a column of the customer's address' city name that is ordered in the sort order of another column with zip codes. Clearly the city names aren't in their sort order but they will contain a good amount of runs since equal and similar zip-codes map to the same city name. Given that average run-length $|r|$, one can estimate the cache misses with:

$$M(s\_trav(R)) = \left\lceil \frac{(R.w + \lceil \log_2 R.n \rceil) \cdot \left\lceil \frac{R.n}{|r|} \right\rceil}{B} \right\rceil \tag{4}$$

The break even point, i.e., the minimal number of the average run-length can be computed with:

$$r = \frac{R.w + \lceil \log_2 R.n \rceil}{R.w} \tag{5}$$

## C. Bit-Vector Encoding

Bit-vector encoding stores a bitmap for each distinct value. Each bitmap has the length of the number of data items to encode in bits. The value $1$ in a bitmap for a distinct value indicates that the data item with the same index has this particular value. As each data item can only have one value assigned, only one bitmap has the value $1$ at a given index.

The compression size is therefore dependent on the number of data items and the number of distinct values to encode.

$$M(s\_trav(R)) = \left\lceil \frac{(R.w + R.n) \cdot |D|}{B} \right\rceil \tag{6}$$

For each distinct value, the value itself needs to be stored once plus a bitmap of the number of tuples in bits. The break even point can be estimated with:

$$|D| = \left\lceil \frac{R.w \cdot R.n}{R.w + R.n} \right\rceil \tag{7}$$

## *D. Null Suppression*

Null Suppression stores data by omitting leading 0s of each value. The main indicator of how good the compression will be is the average number of 0s that can be suppressed. Think of an integer column that stores the number of products sold per month. Since only the less significant bits would equal to 1 and no negative values would appear one could get a good compression ratio with Null Suppression. Since each value has a variable length Null Suppression needs to store the length of each value. A good way to do that is to suppress only byte-wise, so a value can be stored with one to four bytes. To encode that length one needs two bits so the length-metadata for four values fits on one byte. Given an average number of 0s to suppress $|z|$ and the suppressable bits $|z_b| = 8 \cdot \left\lfloor \frac{|z|}{8} \right\rfloor$ an estimation of the cache misses is possible:

$$M(s\_trav(R)) = \left\lceil \frac{R.n \cdot (R.w - |z_b|) + 2 \cdot R.n}{B} \right\rceil \tag{8}$$

## *E. Dictionary Encoding*

Dictionary Encoding is a widely used compression technique in column-store environments. A dictionary is created by associating each distinct value with a generated unique key - a value id - and replacing the original values in the attribute vector with their value id replacements. By combining the attribute vector with the dictionary entries the original values can be reconstructed. Each distinct value is stored only once while the smaller value ids are used as their references which saves space in memory as well as allowing compatible operators to directly work on the dictionary only, e.g find all distinct values, or vice versa operate on the attribute vector without accessing the dictionary. Usually, storing the value ids instead of the actual values take much less space in memory and their length is fixed allowing variable length values to be treated as fixed length values in the document vector, which leads to increased performance [9]. Given the dictionary fits into the cache the cache misses for a single column scan can be estimated with the following formula:

$$M(s\_trav(R)) = \left\lceil \frac{R.id \cdot R.n}{B} \right\rceil + \left\lceil \frac{|D| \cdot R.w}{B} \right\rceil \tag{9}$$

The more distinct values the dictionary needs to hold the more likely it is that a lookup leads to a cache miss. Since the access is random previously unloaded cache lines need to be fetched again. Hence, the size of the dictionary matters. The cache misses can be estimated with the formula for a repetitive random access pattern $rr\_acc$ presented in [15], since the accessed position in the dictionary is random and can be the same multiple times.

$$M(s\_trav(R)) = \left\lceil \frac{R.id \cdot R.n}{B} \right\rceil + M(rr\_acc(|D| \cdot R.w)) \tag{10}$$

with:

$$M(rr\_acc(R)) = \left\lceil C + (\frac{r}{I} - 1) \cdot (C - \frac{\#}{C} \cdot \#) \right\rceil \tag{11}$$

and $C = \left\lceil \frac{I \cdot R.w}{B} \right\rceil$ where $I$ is an approximation of the number of accessed tuples. Since the whole data is read, each value in the dictionary is read at least once and $I = R.n$. $r$ is the number of access operations which is equal to $R.n$, too. $\#$ is the number of slots in the cache and therefore equals to $cacheSize/B$ in a fully associative cache.

## *F. Comparison*

We compare the expected cash misses for a table with a size typical in Enterprise Data: Given one million 48 byte string values of which 50,000 are distinct an uncompressed column scan would issue $\left\lceil \frac{48 \cdot 10^6}{64} \right\rceil = 750,000$ cache misses with a 64 byte cache line. We calculate the expected cash misses for each algorithm and provide a sensitivity analysis on the number of distinct values.

*1) Run-Length Encoding:* A run-length encoded column when stored in sort order would issue only $\left\lceil \frac{(8 \cdot 48 + \log_2 10^6) \cdot 50,000}{8 \cdot 64} \right\rceil = 39,454$ cache misses.

The break even point would be at $\left\lceil \frac{8 \cdot 48 \cdot 10^6}{8 \cdot 48 + \log_2 10^6} \right\rceil = 950,496$ distinct values.

If the column was stored unsorted the number of expected cash misses would be in the worst case $\left\lceil \frac{(8 \cdot 48 + \log_2 10^6) \cdot 10^6}{8 \cdot 64} \right\rceil = 788,929$. This worst case would occur in the scenario of an average run-length of 1. The average run-length for each value has to be at least $\left\lceil \frac{8 \cdot 48 + \log_2 10^6}{8 \cdot 48} \right\rceil = 1,05$ to issue less cache misses compared to no compression.

*2) Bit-Vector Encoding:* For the example above a full scan would issue $\left\lceil \frac{(8 \cdot 48 + 10^6) \cdot 50,000}{8 \cdot 64} \right\rceil = 97,693,750$ cache misses. Bit-vector encoding clearly is not suitable for lots of distinct values or a small amount of bytes to compress. Since each column has a good number of 0-runs, run-length encoding on top of bit-vector encoding might help. The break even point is at $\left\lceil \frac{8 \cdot 48 \cdot 10^6}{8 \cdot 48 + 10^6} \right\rceil = 384$ distinct values.

On the other hand, the formulas show that the amount of bytes per value play a little role in the overall amount of cache misses. Thus, bit-vector encoding should be used when the values to compress have a certain size.

*3) Null Suppression:* Given that the average amount of bytes to suppress is 3 then the estimated cache misses are $\left\lceil \frac{10^6\cdot(8\cdot48-8\cdot3)+2\cdot10^6}{8\cdot64} \right\rceil = 707,032$.

*4) Dictionary Encoding:* Given a 4 byte value id the number of cache misses for a single column scan in a dictionary encoded column are $\left\lceil \frac{4\cdot10^6}{64} \right\rceil + \left\lceil \frac{50,000\cdot48}{64} \right\rceil = 100,000$

The following table shows the estimated number of cache misses for the example above with the 5% distinct cardinality.

| | Cache Misses |
|---|---:|
| No compression | 750,000 |
| RLE (sorted) | 39,454 |
| RLE (unsorted, worst case) | 788,929 |
| Bit-Vector Encoding | 97,693,750 |
| Null Suppression | 707,032 |
| Dictionary Encoding | 100,000 |

Figure 1 shows a sensitivity analysis with regards to the number of distinct values in order to investigate the influence of a changing cardinality of those.

### G. Evaluation

The usefulness of compression algorithms depends on the data profile of a column. The following table describes the applicability of each compression technique for several data profiles.

| | few distinct | many distinct |
|---|---|---|
| No compression | - - | - - |
| RLE (sorted) | + + | + |
| RLE (unsorted) | - - | - - |
| Bit-Vector Encoding | + | - - |
| Null Suppression | - | - |
| Dictionary Encoding | + | + |

Our goal is to find a compression technique that performs best under OLTP as well as OLAP workloads in an enterprise environment. Based on our findings of enterprise data characteristics in [12], we focus on a sparse data set with a vast amount of columns but most of them storing a small number of distinct values. We use a column-oriented store since it performs better under an OLAP workload than a row store does [21]. However, in an OLTP scenario most queries fetch only few complete records. The column store finds each respective entry in all columns separately and then reconstructs the record. Since there are lots of columns finding those entries has to be fast.

Run-length encoding on a sorted column issues by far the fewest cache misses of all presented compression techniques. However, applying run-length encoding requires sorting each column before storing it. In order to reconstruct records correctly we would have to store the original row number as well, called the surrogate id. When reconstructing records each column needs to be searched for that id resulting in a complexity of $O(R.n)$ per column. As Enterprise Applications typically operate on tables with up to millions records we cannot use surrogate ids and prefer direct or implicit offsetting instead ($O(1)$).

Basic dictionary encoding allows for direct offsetting into each column and also benefits from a sparse data set as enterprises have it. In addition the compaction process' performance increases when using dictionary encoding compared to run-length encoding. Therefore, dictionary encoding fits our needs best and is our compression technique of choice for a mixed workload. Furthermore, it still can be optimized as described in the following section.

## III. Dictionary Compression Techniques

In this section, we discuss various optimization of the basic dictionary approach introduced in section II-E. We evaluate their applicability for different data access profiles.

### A. Order-Indifferent Dictionary

The basic dictionary approach described in the last chapter did not care about how the values in the dictionary are ordered. That makes finding a value in the dictionary, e.g., for an insert - one needs to find out whether the value is already in the dictionary - an expensive operation ($O(|D|)$). A possible solution is to store the values based on their hash value. Given a good hash function one can find a value in the dictionary in $O(1)$ as well as finding a value for a specific value id in $O(1)$. This clearly depends on a good hash function and may increase compression size.

### B. Order-Preserving Dictionary

In comparison to the basic dictionary a hash value supported dictionary approach could speed up finding a value in the dictionary but still does not enforce ordered data items in memory. That becomes a disadvantage when executing range queries like finding all records that begin with the letter $K$, e.g., in a column that stores names. An order-indifferent dictionary needs to traverse the whole dictionary filtering all values that begin with $K$ and returning their associated value ids. This has a complexity of $O(|D|)$. In a sorted dictionary, one could find the first occurrence of a value starting with $K$ and $L$ with binary search and then return the lower and upper bound for all value ids that are associated with values beginning with $K$ without actually checking their values. The complexity is $O(\log_2 |D|)$. The downside of that approach is that if new values need to be added to the dictionary they can destroy the sort order invalidating possibly all value-id/value associations and resulting in a complete rewrite of the attribute vector $O(R.n)$.

### C. Bit-Compressed Attribute Vector

The size of the attribute vector, hence the read performance, is also affected by the compression ratio between the original values and the value id replacements. However, the number of distinct values in the uncompressed values collection is important as well. Firstly, because the entries in the dictionary increase with every unique value - for every value-id/value compression ratio, there is a number of distinct values hat dictionary encoding becomes useless - secondly, the more unique values need to be encoded, the more unique value

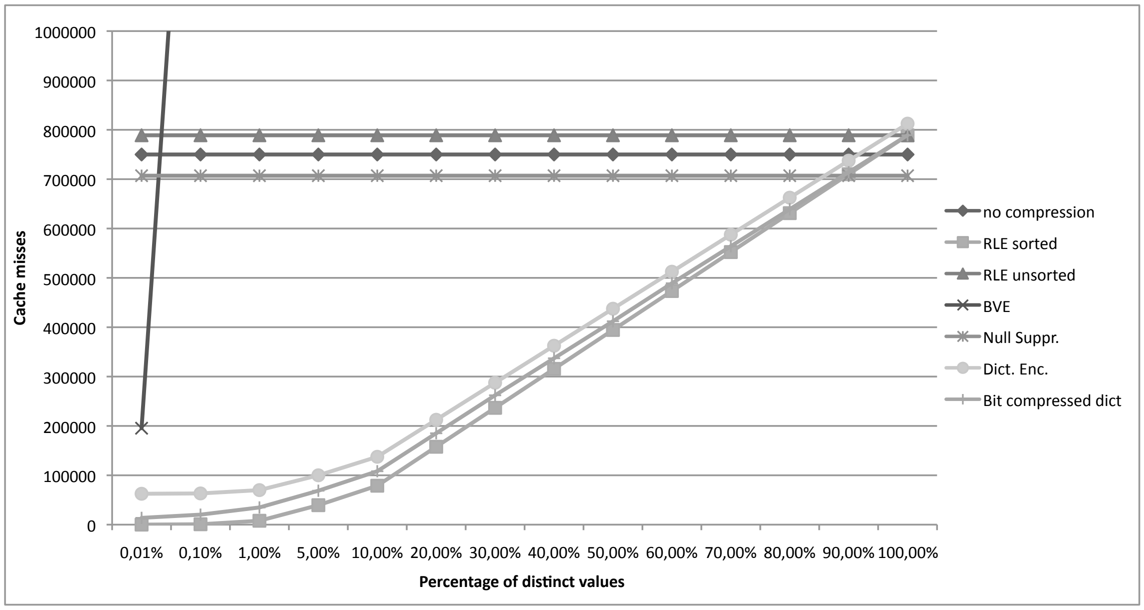


Fig. 1. Compression techniques with regards to cache misses and distinct values.

ids are needed. In the basic dictionary example above a lot of compression opportunities are wasted by reserving 32 bits to define the value id space. Dictionary encoding with bit-compressed value ids varies the length of the value ids and reserves only the needed number of bits to encode all distinct values but still guarantees fixed-length value ids. Given 200 values in the dictionary, the attribute vector that needs to be compressed needs only one byte to store each value id, allowing 64 value ids to fit on a 64 byte cache line. Similar to the order-preserving dictionary the disadvantage of this approach is that the bit-length of the value ids needs to be increased and all values in the attribute vector need to be rewritten when the number of distinct values in the dictionary exceeds the amount of values that can be encoded with the current number of bits - $O(R.n)$.

The cost model can then be extended with:

$$M(s\_trav(R)) = \left\lceil \frac{\lceil \log_2 |D| \rceil \cdot R.n}{B} \right\rceil + \left\lceil \frac{|D| \cdot R.w}{B} \right\rceil \quad (12)$$

Taking the same parameters from the previous section the issued cache misses then are $\left\lceil \frac{\lceil \log_2 50{,}000 \rceil \cdot 10^6}{8 \cdot 64} \right\rceil + \left\lceil \frac{50{,}000 \cdot 48}{64} \right\rceil = 68{,}750$ which is almost half the amount of the basic dictionary.

## D. Bit-Compressed Order-Preserving Dictionary Encoding

The last two described dictionary compression techniques have the same problem of rewriting the whole attribute vector for different reasons. However, the two problems are connected. A reordering of the dictionary can only happen if new distinct values are added to the collection or when deleting values. Furthermore, an extension of the value id space can only be a result of adding new distinct values to the dictionary. Using both approaches together can lower the cost of inserts and updates. When new values are added to the dictionary and the amount of values exceeds the value id space then the rewriting of the attribute vector can do both, updating to new value ids with the new bit-length, in one step.

## E. Comparison

The following table shows the different dictionary encoding variants under different workloads.

| | Basic | Hash Map | Bit-Compr. | Order-Pres. |
|---|---|---|---|---|
| few inserts, many equal queries | + | + | + + | - |
| few inserts, many range queries | - | - - | + | + + |
| many inserts, many equal queries | + | + + | - | - |
| many inserts, many range queries | - | - | - - | + |

The following table lists the advantages and disadvantages of the different dictionary encoding variants.

| | ADVANTAGES | DISADVANTAGES |
|---|---|---|
| Basic Dict. | fixed length, compression time | compression size |
| Hash Map | compression time | execution time |
| Bit-Compr. | compression size | compression time |
| Order-Pres. | execution time | compression time |
| Order-Pres. & Bit-Compr. | execution time, compression size | compression time |

## IV. RELATED WORK

In the area of database management systems, compression is also used to improve query speed as described in [22] as work focused on reducing the amount of data only. That becomes especially useful when data is stored column-wise such as in C-Store, a disk-based column-oriented database, see [21]. The work presented in [1] describes how compression can be integrated into C-Store and shows the impact on read performance. In a real world scenario one has to consider the negative impact on write performance when using compression. [10] comes to the conclusion that column stores perform better than row stores in most cases.

However, data compression can limit the applicability to scenarios with frequent updates leading to dedicated delta structures to improve the performance of inserts, updates and deletes. The authors of [7] and [18] describe a concept of treating compressed fragments as immutable objects, using a separate list for deleted tuples and uncompressed delta columns for appended data while using a combination of both for updates. In contrast, the work of [11] maintains all data modification of a table in one differential buffer that is write-optimized and keeps track of invalidation with a valid bit-vector. Later work of the same authors shows how to enable fast updates on read-optimized databases by leveraging multi-core CPUs [13].

The work of [5] depicts a technique of maintaining a dictionary in a order-preserving way while still allowing inserts in sort order without rebuilding the attribute vector due to changed value id's.

In the area of in-memory databases with the focus on OLTP and real-time OLAP, the customer study presented in [12] show a very high amount of read queries compared to write queries supporting the fact that a compressed read-optimized store is useful.

## V. Conclusion

In this paper, we showed and explained the positive impact on read performance for an in-memory database when using data compression. In order to compare different kinds of lightweight compression techniques under different data distributions we extended the generic cost model to take compression into account. We also presented several lightweight compression techniques and different optimizations regarding dictionary compression as well as trade-offs that have to be made in favor of late materialization and write performance. The paper also described why focussing on read performance is necessary and how a sufficient write performance can be achieved as well. It concludes that under most circumstances - especially for column stores - dictionary compression is the best choice when it comes to optimizing read and write performance under a mixed workload.

## References

[1] D. J. Abadi, S. Madden, and M. Ferreira. Integrating compression and execution in column-oriented database systems. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Chicago, Illinois, USA, June 27-29, 2006*, pages 671–682, 2006.

[2] D. J. Abadi, D. S. Myers, D. J. DeWitt, and S. Madden. Materialization Strategies in a Column-Oriented DBMS. In *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007*, pages 466–475, 2007.

[3] A. Ailamaki, D. J. DeWitt, and M. D. Hill. Data page layouts for relational databases on deep memory hierarchies. *VLDB J.*, 11(3):198–215, 2002.

[4] A. Ailamaki, D. J. DeWitt, M. D. Hill, and D. A. Wood. DBMSs on a Modern Processor: Where Does Time Go? In *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*, pages 266–277, 1999.

[5] C. Binnig, S. Hildenbrand, and F. Färber. Dictionary-based order-preserving string compression for main memory column stores. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2009, Providence, Rhode Island, USA, June 29 - July 2, 2009*, pages 283–296, 2009.

[6] P. A. Boncz, S. Manegold, and M. L. Kersten. Database Architecture Optimized for the New Bottleneck: Memory Access. In *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*, pages 54–65, 1999.

[7] P. A. Boncz, M. Zukowski, and N. Nes. MonetDB/X100: Hyper-Pipelining Query Execution. In *CIDR*, pages 225–237, 2005.

[8] U. Drepper. What every programmer should know about memory, 2007.

[9] S. Harizopoulos, V. Liang, D. J. Abadi, and S. Madden. Performance tradeoffs in read-optimized databases. In *VLDB '06: Proceedings of the 32nd international conference on Very large data bases*, pages 487–498. VLDB Endowment, 2006.

[10] S. Harizopoulos, V. Liang, D. J. Abadi, and S. Madden. Performance Tradeoffs in Read-Optimized Databases. In *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, September 12-15, 2006*, pages 487–498, 2006.

[11] J. Krüger, M. Grund, C. Tinnefeld, H. Plattner, A. Zeier, and F. Faerber. Optimizing Write Performance for Read Optimized Databases. In *Database Systems for Advanced Applications, 15th International Conference, DASFAA 2010, Tsukuba, Japan, April 1-4, 2010, Proceedings, Part II*, pages 291–305, 2010.

[12] J. Krüger, M. Grund, A. Zeier, and H. Plattner. Enterprise Application-Specific Data Management. In *Proceedings of the 14th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2010, Vitoria, Brazil, 25-29 October 2010*.

[13] J. Krüger, C. Kim, M. Grund, N. Satish, D. Schwalb, J. Chhugani, H. Plattner, P. Dubey, and A. Zeier. Fast Updates on Read-Optimized Databases Using Multi-Core CPUs. *PVLDB*, 5(1):61–72, 2011.

[14] J. Krüger, C. Tinnefeld, M. Grund, A. Zeier, and H. Plattner. A case for online mixed workload processing. In *Proceedings of the Third International Workshop on Testing Database Systems, DBTest 2010, Indianapolis, Indiana, USA, June 7, 2010*, 2010.

[15] S. Manegold, P. A. Boncz, and M. L. Kersten. Generic Database Cost Models for Hierarchical Memory Systems. In *VLDB 2002, Proceedings of 28th International Conference on Very Large Data Bases, August 20-23, 2002, Hong Kong, China*, pages 191–202, 2002.

[16] D. E. Ott. Optimizing software applications for numa.

[17] H. Plattner. A common database approach for OLTP and OLAP using an in-memory column database. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2009, Providence, Rhode Island, USA, June 29 - July 2, 2009*, pages 1–2, 2009.

[18] R. Ramamurthy, D. J. DeWitt, and Q. Su. A case for fractured mirrors. *VLDB J.*, 12(2):89–101, 2003.

[19] J. Rao and K. A. Ross. Cache Conscious Indexing for Decision-Support in Main Memory. In *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*, pages 78–89, 1999.

[20] J. Rao and K. A. Ross. Making $B^+$-Trees Cache Conscious in Main Memory. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, pages 475–486, 2000.

[21] M. Stonebraker, D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madden, E. J. O'Neil, P. E. O'Neil, A. Rasin, N. Tran, and S. B. Zdonik. C-Store: A Column-oriented DBMS. In *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005*, pages 553–564, 2005.

[22] T. Westmann, D. Kossmann, S. Helmer, and G. Moerkotte. The Implementation and Performance of Compressed Databases. *SIGMOD Record*, 29(3):55–67, 2000.

# A System Overview for netCDF-FastBit Integration

David Marks, Elias Ioup, John Sample, Kevin Shaw
Geospatial Computing
Naval Research Laboratory
Stennis Space Center, Mississippi
{dmarks,eioup,jsample,kshaw}@nrlssc.navy.mil

Mahdi Abdelguerfi
Computer Science Department
University of New Orleans
New Orleans, Louisiana
mahdi@cs.uno.edu

***Abstract*—This paper discusses the creation of a FastBit bitmap index from the contents of a netCDF file. Using a two-step transformation, netCDF is loaded into a FastBit bitmap index which can then be used to perform quick and highly efficient data queries. Metadata from the original netCDF is utilized along with the bitmap indexes and the output is extracted and visualized in several formats. The performance of the netCDF-FastBit indexes is shown to be far greater than accessing the netCDF file without.**

***Keywords-netCDF; geospatial processing; bitmap indexing***

## I. INTRODUCTION

As highly advanced sensors began to penetrate into the awareness of the disparate scientific fields, many researchers found themselves grappling with the need to store and usably interact with vast amounts of data. Unfortunately, the real world scientific data that was being collected violated several of the classical assumptions about data in database design. The largest hurdle of all for traditional database structures was the high multidimensionality of the scientific data, lacking any field or even reasonable combination of fields that could serve as a unique identifier for a gathered data.

In contrast to traditional database design, bitmap indexes do not require a unique key identifier, and indeed work quite well with very multidimensional data. Bitmap indexes are usually overlooked in lieu of traditional databases because of their inefficiency in handling update and delete operations, but in the case of scientific data that is written once this inefficiency does not matter. The other detriment normally levied against bitmap indexes, the “Curse of Cardinality”, has largely been solved using the process of order-preserving bin-based clustering [1]. Indeed, bitmap indexing serves as an excellent form of indexing for write-once highly multidimensional data, which real world scientific data is a prime example of.

NetCDF is a popular scientific data storage format, and it provides a very compact encoding of large multidimensional data sets. It is not, however, optimized for the retrieval, analysis, and mining of the data stored within its format. Performing efficient queries over data stored within netCDF files is difficult, especially if a large number of netCDF files are involved.

FastBit is an open source bitmap indexing toolkit. FastBit creates highly efficient bitmap indexes, and provides an interactive SQL interface into its generated indexes. Its WAH based compression has been proven to provide optimal query response time with the compressed bitmaps smaller than comparable B-trees [2].

Using the system outlined within this paper, a netCDF file can be loaded into a FastBit bitmap index, providing efficient and interactive query access to the data stored within. The process requires only a one-time cost of bitmap index creation time and room to store the generated indexes. Further, a number of different ways to envision the produced query results are demonstrated.

In the next section, the method used to create bitmap indexes from netCDF data is discussed, with experimental results generated from testing provided. In System Architecture, we present the structure and functions of the proposed system. Screenshots of our proposed graphical user interface are provided, as well as a schematic for the system architecture. In Query Processing, the benefits of such a system are described, both in terms of speedup of generalized data access and by a number of advanced spatio-temporal queries. Finally, in Conclusions we reiterate the goals and benefits of this system and stress the benefits its use could provide.

## II. METHODS

Because FastBit currently only accepts comma separated values as input for bitmap generation, the first step in netCDF-FastBit integration involves a transformation of the netCDF data into the comma separated value format employed by FastBit. A number of tools are already existent for the purpose of netCDF to comma separated values, but as the concept of comma separated values is only loosely defined, different tools produce outputs in different formats. One tool, ncks (short for netCDF Kitchen Sink), offers a correctly formatted comma separated value transformation however, and with the correct configuration of options our source netCDF files can be converted into a comma separated value format readable by FastBit. After FastBit ingests the transformed netCDF data, the process is complete. In testing, a 1.4 GB netCDF file required slightly over 30 minutes to transition from netCDF to FastBit bitmap index. The resulting index used up 6.8 GB of space.

## III. SYSTEM ARCHITECTURE

The main purpose of the netCDF-FastBit integration is to provide an efficient and interactive SQL interface into the data stored within. To maximize the search capability of a user's queries, however, requires more than just the netCDF extracted indexes. Further, retrieved data may require different formats for different questions. Thus, the below system in Fig. 3 is proposed.

### A. *Internal Components*

The *Bitmap Index Manager* is the component charged with the building and maintenance of the bitmap indexes as described in Methods above. The *Manager* allows for both single and multiple file indexing for netCDF scientific data. Further, the netCDF files may be accessed either locally or across a network.

The *Metadata Repository* holds any relevant metadata found within the netCDF file's header. Often these headers contain comments and notes about the data that would not be immediately obvious in an examination of the data itself, such as the data's provenance. This metadata will aid in maintaining organization between the bitmap indexes of several netCDF files, as well as providing a boost to query speed by screening out queries from which no point (or perhaps all points) meets the desired criteria.

The *Query Processor*'s role is to transform the user submitted high level queries into sequences of bitwise logical operations executed in the reduced search space generated using the *Metadata Repository*. Further, the *Query Processor* will optimize the processing of a number

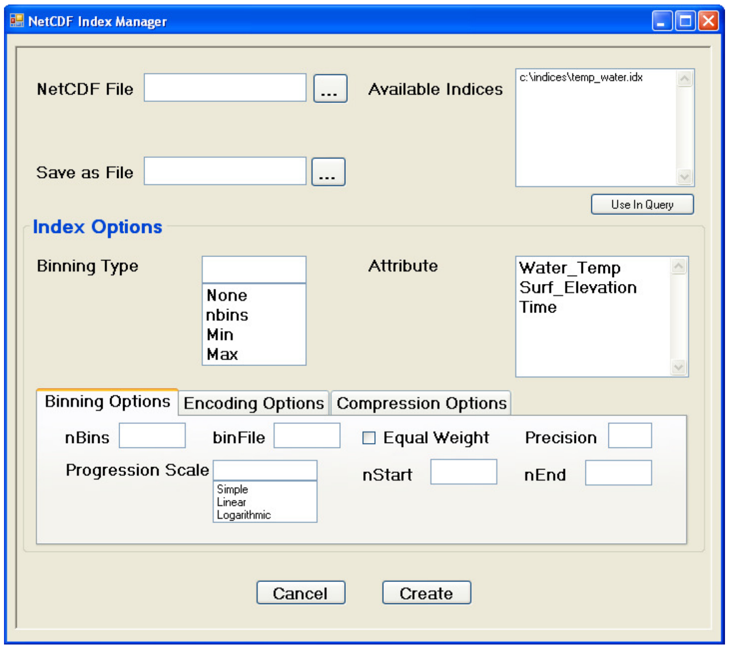


Figure 1. Index Manager Screen

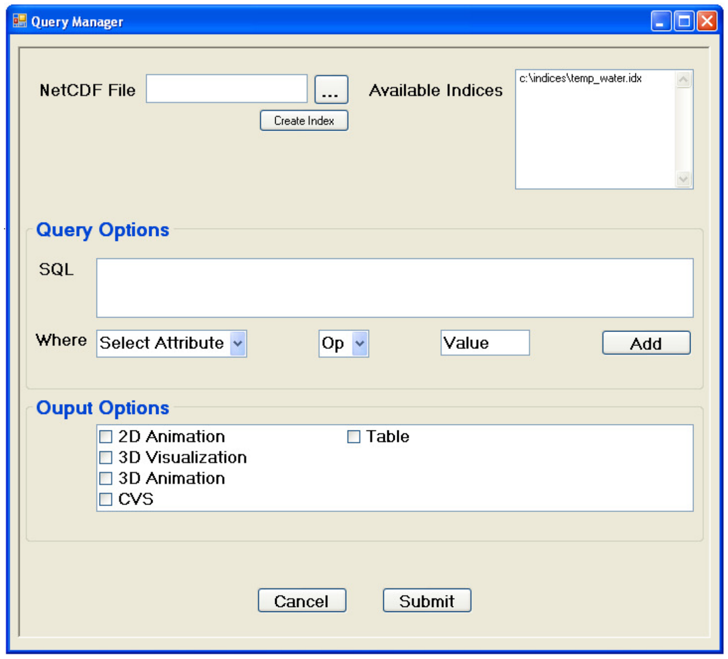


Figure 2. Query Manager Screen

of primitive operations found in a recent study on query processing of mesh data in forming the basis of higher level queries [3].

### B. *Graphical User Interface*

To utilize the outlined system, a graphical user interface is provided, as seen in Fig. 1 and Fig. 2. A user either selects a locally available netCDF file(s) or points to one available over the network. A list of already processed indexes saved with the *Metadata Repository* are then displayed for the selected netCDF file, and can be instantly used via the "Use In Query" button.

Alternatively, a new index can be created from a selected netCDF file, with a number of different bitmap indexing options made available via the index options tabs. These options will alter the generated bitmap index's underlying structure, and can be used to fine tune the created index for specific tasks and queries. The default values will serve for most general purposes, however, and the most common usage will simply consist of selecting which attributes to include in the generated index.

Once an index to query have been selected, the user can submit a query either freehand with the SQL textbox or by using the dropdown Where boxes to programmatically generate one. When submitting their query, a user will also select which kind of output is desired, either in the form of text or as a visualization of the results. Note that more than one form can, of course, be chosen, although some forms are not appropriate for all results. Animations, for example, are useless unless the results include some concept of passing time.

## IV. QUERY PROCESSING

The system outlined above allows for an unprecedented level of access into the contents of a netCDF file. Absent the created bitmap indexes, any kind of search on the netCDF could only be accomplished via a linear search of the file itself. Using our test data, this linear search took approximately 20 minutes to complete, and all values of speedup given below are based on this figure.

The simplest types of queries examined were simple equality queries, such as "return all points where the temperature was X". These queries returned in only 1 second, a 1200x speedup. Retrieving all temperatures for a single point in time returned in 4 seconds, a 300x speedup. The slowest execution time was found in inequality queries, such as "return all points where the temperature was above Y", which took 4 minutes to compute. That still represented a 5x speedup, however.

But this system is not limited to only simple queries. More complex queries are possible given the powerful SQL interface afforded by FastBit. A range query was created and implemented to return all points within a specified distance of a chosen point. Using a two-step bounding box filter and the haversine function, this range query often returned results within a second or less, resulting in a 1200+x speedup. Further, by ordering the returned results based on distance to the chosen point and returning only a specified k number of results, a pseudo-KNN search was built, with equivalent performance results [4].

## V. Conclusion

This system provides a way to efficiently process data stored within netCDF files and to produce data output in a number of formats. The graphical user interface allows for an easy to use interface into the system and allows a user to leverage a number of different bitmap index customization options in the creation of their index. Beyond providing an otherwise unavailable amount of access into the contents of a netCDF file, our query results demonstrate the massive gains in performance inherent in this approach. Using this system, querying the contents of a netCDF file not only becomes possible, but easy and efficient.

## References

[1] Kesheng Wu, Kurt Stockinger, and Arie Shoshani. "Breaking the curse of cardinality on bitmap indexes." Scientific and Statistical Database Management. Ed. Bertram Ludäscher & Nikos Mamoulis. Springer, 2008, pp. 348–365.

[2] Kesheng Wu, Ekow J. Otoo, and Arie Shoshani. *An efficient compression scheme for bitmap indices.* ACM Transactions on Database Systems 31:1 (2006), pp. 1-38

[3] B.S. Lee and R. Musick. *MeshSQL: the query language for simulation mesh data*. Information Sciences, volume 159, issue 3-4, 2004, pp. 177-202.

[4] David Marks, "An Analysis of netCDF-FastBit Integration and Primitive Spatial-Temporal Operations" (2009). *University of New Orleans Theses and Dissertations.* Paper 984. http://scholarworks.uno.edu/td/984 Retrieved 12/2011

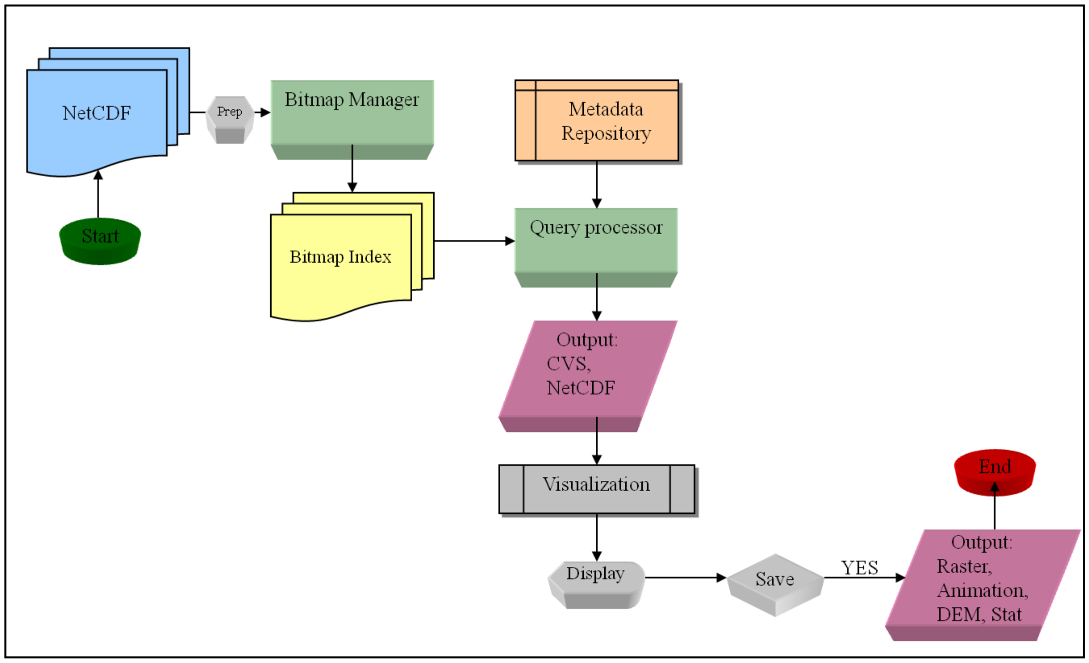


Figure 3. System Architecture

# The Multi-Tenant Data Placement Problem

Jan Schaffner*, Dean Jacobs†, Tim Kraska‡, and Hasso Plattner*
*Hasso-Plattner-Institute
University of Potsdam
August-Bebel-Str. 88
14482 Potsdam, Germany
Email: firstname.lastname@hpi.uni-potsdam.de
†SAP AG
Dietmar-Hopp-Allee 16
69190 Walldorf, Germany
Email: dean.jacobs@sap.com
‡UC Berkeley
465 Soda Hall MS-1776
Berkeley, CA 94720, USA
Email: kraska@berkeley.edu

***Abstract*—With the advent of the Software-as-a-Service (SaaS) deployment model, managing operational costs becomes more and more important for providers of hosted software. The cost for hosting and providing a service is directly proportional to the operational margin that can be achieved when running a SaaS business. Possible avenues for reducing operational costs are consolidation (i.e. co-locating multiple customers onto the same server) and automation of cluster management (i.e. migration of customers between servers, automatic replication for performance or high availability). In this paper, we propose the formalization for the problem of assigning "tenants" (i.e. the customers) to servers of an on-demand database cluster. We will pose this problem in the form of an optimization problem, omitting database specifics and thus presenting the problem in an abstract fashion, using the metaphor of assigning tokens to baskets (i.e., tenants to servers). This formalization is both a first step towards solving the placement problem in an efficient way and a helpful basis for comparing multiple solutions to the problem to each other.**

***Index Terms*—multi-tenancy; software-as-a-service; databases; enterprise applications; column-store**

## I. Introduction

Implementations of Enterprise SaaS commonly maintain data in a farm of conventional databases. To reduce total cost of ownership, multiple tenants are consolidated into each database instance [1]. As an example, in October 2007, the SaaS CRM vendor RightNow had 3,000 tenants distributed across 200 MySQL database instances with 1 to 100 tenants per instance [2]. Table I shows the cost of hosting such a system with and without multi-tenancy on Amazon's Elastic Compute Cloud (EC2) for a year. This calculation assumes each database is run on one small EC2 instance, which may not be sufficient in some cases. The standard on-demand pricing for one such unit is $0.085 per hour. The one-year reserved pricing is $227.50 up front plus $0.03 per hour [3]. These figures do not include the costs to administer the databases, which would further skew the results in favor of multi-tenancy by a wide margin. The moral of the story is clear: the more consolidation the better.

| | **Single-tenant** | **Multi-tenant** |
|---|---|---|
| Standard on-demand pricing | $2,233,800 | $148,920 |
| One-year reserved pricing | $1,470,900 | $98,060 |

TABLE I
Yearly Cost to Host RightNow Databases on Amazon EC2

SaaS implementations commonly use pre-existing database replication mechanisms for availability and performance. This practice treats each group of tenants as a single unit for the purposes of replication. As a result, the excess capacity that is reserved to handle failures and workload surges is pooled across only a small number of servers and significant over-provisioning is required. In particular, common practice is to maintain master/slave pairs in which each slave gets no traffic so that it can handle the full load in case its master fails. This technique is often referred to as *mirroring* (cf. Figure 1). The downside of mirroring is that in case of a failure all excess workload is re-directed to the other mirror. In doing so, the mirror server is a local hot-spot in the cluster until the failed server is back online. A technique for avoiding such hot-spots is to use *interleaving*, which was first introduced in Teradata [4]. Interleaving entails performing replication on the granularity of the individual tenants rather than all tenants inside a database process. This allows for spreading out the excess workload in case of a server failure across multiple machines in the cluster. As a result, excess capacity is pooled across a larger number of servers, the work on any one server is smoothed out with high probability, and less over-provisioning is required. As an example, consider a scenario in which the layout is "perfect" in the sense that, for any two servers, there is at most one tenant with a replica on both servers. If there are 10 tenants per server, then over-provisioning by 50% would allow a failure of up to 5 servers that share a tenant with any given server. The benefits of this scheme are proportional to the amount of consolidation: with 100 tenants per database, allowing a failure of 5 servers would require over-provisioning by only 5%.

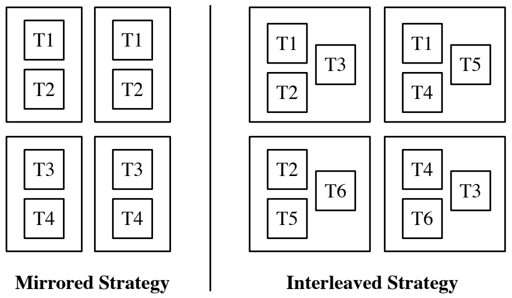


Fig. 1. Example Layouts of Tenant Data

Replica placement is an on-line problem: the layout must be modified as tenants join and leave the system and their resource consumption varies with seasons and success. Placement algorithms must be judged not only on the quality of the layout, but also on the amount of data that must be migrated to achieve it. Migration should be minimized because it consumes resources and it can complicate failure recovery. Web-scale databases generally sacrifice some layout quality in order to avoid excessive migration. For traditional data warehouses, mostly large and expensive server and storage systems are used. For small- and medium-sized companies, it is often too expensive to implement and run such systems. Given this situation, the SaaS model comes in handy, since these companies might opt to run their OLAP at an external service provider. The challenge is then for the analytics service provider to minimize total cost of ownership by consolidating as many tenants onto as few computing resources as possible, a technique often referred to as multi-tenancy [1].

The *Rock* project at the HPI [5], [6], [7], [8] seeks to maximize throughput in a cluster of main memory column databases. The goal is to support the highest possible number of concurrently active users while guaranteeing hard service level objectives on end-user response times (e.g. “99 percent of all queries have to complete in less than 1 second”). We explore different data placement strategies for deciding which tenants are co-located on which servers in order to minimize the number of servers when running a given number of users. This problem is at the heart of large-scale Internet services trying to minimize the cost of their data centers.

This paper is structured as follows: Section II will formulate the placement problem as an optimization problem. We will omit database specifics and pose the problem in an abstract fashion, using the metaphor of assigning tokens to baskets (i.e., tenants to servers). Section III discusses the computational complexity of the placement problem. In Section IV, we discuss related work. Section V concludes this paper.

## II. Problem Statement

Let $N = \{i_1, \ldots, i_{|N|}\}$ be a set of baskets and $T = \{t_1, \ldots, t_{|T|}\}$ a set of tokens. An assignment of tokens to baskets is given, as shown in Figure 2. All tokens have a radius $r(t)$ and a color $c(t)$, which are also known for all tokens. Each color occurs exactly twice (or, in other words, each token occurs exactly twice, which means that there are $2|T|$ tokens in an assignment). The problem to be solved is to find a new assignment of tokens to baskets with the following properties:

- No color occurs twice in the same basket.
- The sum of all token radiuses in each basket does not exceed a fixed upper bound $cap(i)$.
- The sum of all token radiuses should have a similar value for all baskets.
- Any two colors appearing together in one basket should preferably not appear together in a second basket.

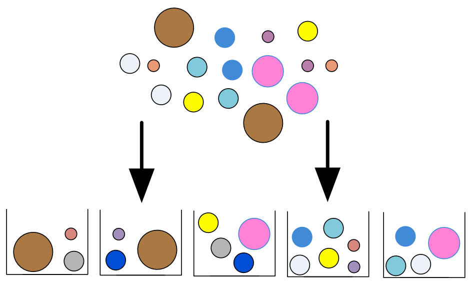


Fig. 2. Assignment of tokens to baskets

### A. Formal Description

To denote the assignment of tokens to baskets we define a decision variable $y$ as follows:

$$y_{t,i}^{(k)} = \begin{cases} 1 & \text{if token } t \text{ is in basket } i \\ 0 & \text{otherwise} \end{cases}$$

$$\text{with } k \in \{0,1\},\ t \in T,\ i \in N$$

The index $k$ identifies the first and the second token of the same color, respectively. An assignment of tokens to baskets $Y'$, the number of baskets $N$, and the set of tokens $T$ with their respective radiuses and colors is given as an input for this problem. The goal is to devise an algorithm which calculates a new assignment $Y$ (i.e. the transformation $f : Y' \to Y$).

Token radiuses increase and decrease as time progresses, although they are fixed for any given instance if the problem. The model is that a new instance of the problem is created each time one or more changes in token radius have been observed. We are looking for an algorithm that—when invoked—balances the sum of token sizes across all baskets and, at the same time, tries to minimize color co-appearance across the baskets. An optimal assignment in this respect would be such that no two colors appearing together in one basket appear together in any other basket at the same time. To do so, the algorithm moves tokens between the baskets.

The sum of all radiuses $r(s)$ of the tokens in a given basket $i$ is defined as

$$R(i) := \sum_{t\in T}\sum_{k=0}^{1} y_{t,i}^{(k)} r(t),\ i \in N.$$

To describe how good (or bad) a given assignment of tokens to baskets is w.r.t. color co-appearance, we introduce a penalty function $P(i)$. It is computed from the perspective of an individual basket and is defined as the sum of all token radiuses of co-appearing tokens in one of the other $N-1$ baskets. Since this value depends on which of the other $N-1$ baskets is chosen as a partner in this binary comparison, $P(i)$ is defined relative to the partner yielding the maximum value:

$$P(i) = \max_{j\in N}\left(\sum_{t\in T}\sum_{k=0}^{1} y_{t,i}^{(k)} y_{t,j}^{(1-k)} r(t)\right)$$
$$\text{with } i,j \in N, i \neq j$$

*1) Constraints:*

1) A valid assignment $Y$ must contain each color exactly twice.
$$\sum_{i\in N} y_{t,i}^{(0)} + y_{t,i}^{(1)} = 2,\ \forall t \in T$$
2) No basket must contain any two tokens of the same color.
$$y_{t,i}^{(0)} + y_{t,i}^{(1)} \leq 1,\ \forall t \in T,\ \forall i \in N$$
3) The sum of all token sizes in a basket $i$ must be less than or equal to the capacity of the basket.
$$R(i) \leq cap_i,\ \forall i \in N$$

*2) Objective Functions:*

1) All baskets shall be balanced w.r.t. aggregate token size (in addition to constraint no. 3, which only specifies an upper bound for the sum of all token radiuses within one basket $R_i$). One way of progressing towards a similar value for the different $R(i)$s is to minimize their variance:
$$\min\ Var(R(1), \ldots, R(|N|))$$
2) Co-appearances of colors in the baskets shall be minimized:
$$\min\ \sum_{i\in N} P(i),\ \forall i \in N$$
3) In addition to minimizing the co-appearance penalty per basket (the previous optimization goal), all baskets should have a similar penalty. One way of progressing towards a similar value for the different $P(i)$s is to minimize their variance:
$$\min\ Var(P(1), \ldots, P(|N|))$$

## B. Possible Extensions

The general formulation of the problem as posed above can be extended to make it even harder to devise good solutions.

*1) Varying number of baskets:* For the problem stated above we assume a fixed number of baskets $N$. It might be the case that the observed changes in token size create a situation in which the current number of baskets is not sufficient for finding an assignment of tokens to baskets such that none of the above constraints is violated. Given such a situation, the algorithm is allowed to create a new basket. Similarly, when the changes in token size result in a situation where all of the above constraints could be satisfied using fewer baskets, then the algorithm can empty a basket by migrating its tokens to other baskets and delete the basket. It would also be conceivable to trade-off the the number of baskets against the balancing of the $R(i)$s as well as the values and the balancing of the $P(i)$s.

*2) Minimizing the number of migrations:* So far we have not imposed a limit on the number of movements of tokens between baskets necessary to provide the transformation $f: Y' \rightarrow Y$. However, it would be conceivable to try to minimize the number of movements in this sequence. It would also be interesting to study how fast the other goal functions converge to an optimal value, varying the number of allowed migrations in $f$.

## III. COMPUTATIONAL COMPLEXITY

The tenant placement problem presented above is structurally similar to the general *bin packing with conflicts* (BPC) problem. In addition to standard bin packing with a given set $V$ of $n$ items, an instance of BPC also contains conflict graph $G = (V, E)$, where an edge $(i, j) \in E$ exists if items $i$ and $j$ are in conflict and may thus not be assigned to the same bin. The BPC problem is known to be NP-complete [9]. In order to proof that the tenant placement problem is also NP-complete, it must be shown that:

1) a candidate solution can be verified in polynomial time.
2) an instance of the BPC problem can be reduced to an instance of the tenant placement problem.

1 follows trivially from the observation that all constraints of the tenant placement problem are terms of quadratic or lower complexity. A formal proof of 2 is beyond the scope of this paper. However, the fact that the tenant placement problem only contains additional constraints in comparison to BPC suggests that a reduction is possible. In the remainder of this section we will elaborate on the combinatorial complexity of the tenant placement problem.

One possible way of enumerating all possibilities of assigning the two copies of a tenant to $N$ servers will now be briefly discussed. Given that both copies of a tenant must be on different servers and that both copies are equivalent, all possibilities to assign a single tenant to two servers can be done by creating a matrix with dimensions $N \times N$. The elements of this matrix are defined as follows:

$$(i,j) = \begin{cases} 1 & \text{if } i < j \\ 0 & \text{otherwise} \end{cases},\ i,j \in N$$

| # Servers | # Tenants | Search Space |
|---|---|---|
| 3 | 16 | 43,046,721 |
| 3 | 17 | 129,140,163 |
| 4 | 13 | 13,060,694,016 |
| 3 | 17 | 16,926,659,444,736 |

TABLE II
NUMBER OF COMBINATORIAL POSSIBILITIES FOR SMALL PROBLEM INSTANCES

An example for $N = 4$ servers:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

The number of possibilities for assigning two copies of a tenant to $N$ servers is thus $N(N-1)/2$. We refer to all 1-valued elements of this matrix as possible *configurations* of a tenant.

We developed a brute force solver for enumerating all possibilities to assign $T$ tenants to $N$ servers. As we have already established above, there are $N(N-1)/2$ possibilities to assign two copies of one tenant to $N$ servers. The number of combinatorial possibilities to assign $T$ tenants to $N$ servers is thus:

$$\left(\frac{N(N-1)}{2}\right)^T$$

In our implementation, we encode each combination using a sequence of length $T$ with each element of the sequence being an integer encoding a tenants *configuration*:

$$(a_0, \ldots, a_{T-1}),\ 0 \leq a_i \leq \frac{N(N-1)}{2} - 1$$

This sequence obeys lexicographic ordering in the sense that the next combination can by computed simply by incrementing the rightmost element of the sequence which is smaller than $(N(N-1)/2) - 1$. Brute force enumeration can nicely be parallelized: In our implementation, we use recursion to generate multiple ranges of the sequence with equal sizes. Each of those ranges is then enumerated in parallel. Our implementation uses Scala actors [10] and fits well with the large number of available cores in modern server machines. Nevertheless, a solution to the placement problem can only be found for very small instances of the problem using brute force. Table II show the steep increase of the search space size for four small instances.

## IV. RELATED WORK

The question of how to distribute data in a cluster of databases has a fundamental impact on the overall performance of the cluster. The data placement problem has been systematically studied in the context of parallel databases and, more recently, distributed databases for Web applications.

### A. Parallel Databases

In the parallel databases Teradata [11], Gamma [12], Bubba [13] and Tandem [14], the data placement problem entails distributing a fixed collection of relations across a fixed cluster of servers so as to minimize response times. Large relations are fully partitioned, also called *fully de-clustered*, across all servers and thus placement is straight-forward. For small relations, the placement problem is NP-hard [15] and various heuristics have been proposed. In Bubba, small relations are placed in decreasing order of their access frequency, or “heat”. At each placement step, the algorithm tries to balance the overall heat at each server. In a simulation study, [16] compares the Bubba algorithm with simple round robin placement of small relations. The conclusion is that, for large numbers of small relations, round robin performs as well as the Bubba algorithm. Round robin is in general preferable because it does not require knowledge of the workload.

Parallel databases generally maintain two copies of the data to ensure high availability. For small relations, a common approach to data placement is to simply treat each copy as a separate relation, with the additional constraint that the two copies cannot be placed on the same server. Bubba, for example, maintains the copies in different formats and tracks the heat of each copy independently.

For large relations, Teradata uses a technique called interleaved de-clustering. Each of the $N$ servers in a cluster is made responsible for the primary copy of one fragment of a relation. The secondary copy of each fragment is divided into $N-1$ sub-fragments that are distributed across the other servers in the cluster. Thus, when a server fails and the primary copy of a fragment becomes unavailable, the work is redistributed across $N-1$ other servers.

A disadvantage of interleaved de-clustering is that, if two servers in a cluster become simultaneously unavailable, one sub-fragment will have no active copy and the entire system has to be shut down. This problem gets worse as the cluster gets bigger, since there are more servers that can fail. As an alternative, Gamma uses a technique called chained de-clustering [16] in which the servers are organized into a logical ring. Each of the servers is made responsible for the primary copy of one fragment and the secondary is placed immediately after the primary in the ring. If a server fails, its workload is taken over by its successor and predecessor in the ring, which seemingly leads to an unbalanced system. The solution is that the workloads for the other fragments are incrementally shifted around the ring to re-balance the system. To shift the workload for a fragment, the boundary for the range of responsibilities between the two copies is shifted. Chained de-clustering makes it possible to survive the failure of any two non-adjacent servers in the ring.

In placement problem as presented in this paper, we are solely concerned with the placement of small relations. Interleaving is performed at that level rather than at the level of fragments and sub-fragments of large relations. In the parallel databases, the load on fragments of a large relation is correlated because every fragment is used to answer a query. The performance characteristics of our system differ because tenants submit queries independently. In our context, data placement is an on-line problem and entails adjusting the size of the cluster so response times requirements are met using as few servers as possible. When the load on a server becomes too high due to increases in the maximum resource consumption of tenants, our placement algorithms make local adjustments to the layout. Parallel databases can accommodate changes in the number and sizes of relations and the number of servers in the cluster, however such changes are infrequent and are applied more globally.

### *B. Web-Scale Databases*

Amazon Dynamo is a distributed key-value store that uses a multi-master-update, lazy-update-propagation model with conflict resolution [17]. Data is distributed across the servers using a variant of *consistent hashing* [18], [19]. The keys are hashed into a fixed circular ring and each server randomly chooses $T$ tokens in the ring. For each of its tokens, a server is responsible for the key range from that token to the next higher token of any server. The replicas for a key range are placed on the next distinct servers moving clockwise around the ring. Thus Dynamo interleaves data: if a server fails, its workload is distributed among $T$ other groups of servers. An essential characteristic of this system is that each key is accessed independently, the keys may be randomly partitioned, and such a partitioning results in a good load distribution for large numbers of keys. In contrast, for Enterprise SaaS and the algorithms studied in this paper, the partitions are fixed at tenant boundaries and the workload varies greatly between tenants.

In Google BigTable [20], rows are range-partitioned across servers by key. Rows with the same key prefix are guaranteed to be adjacent, hence the application can exert control over partitioning. Enterprise SaaS can be implemented on such a system by using the tenant ID as the leading prefix of the key. In BigTable, replication is performed at a lower level by the Google File System [21] and there is never more than one active node for a given data item. Nevertheless, if additional active copies were maintained *using different sort orders for the tenants*, interleaved replication would be provided. The problem with this approach is that the system would unnecessarily maintain the sort order of tenants in each replication group. Our approach is not subject to such a constraint and is not operationally more complex.

### *C. Data Placement in Other Areas*

Problems similar to data placement are also known outside the database community.

The so-called File Allocation Problem (FAP) is concerned with assigning files to nodes. If a file is on a node then access is cheap (local access), otherwise there is some communication cost attached to to accessing the file (remote access). This problem has been extensively studied in literature, and many different models using different objectives for optimization have been proposed. A survey providing a qualitative comparison of these models can be found in [22].

An extension of the FAP is the replica placement problem (RPP) for content delivery networks. As an example, consider an Internet service provider that wants to assign multimedia data to caches in the network topology. Here, the cost of remote accesses is extended by the notion of the distance to the closest node that has a copy of the requested object. [23], [24] surveys and compares known formalizations and solutions to RPP.

All approaches which can be subsumed by either FAP or RPP minimize cost functions which are designed with the aim of improving average data access performance. Utilization and the ability to provide service at a guaranteed performance in the presence of failures is not investigated.

## V. Conclusions and Future Work

In this paper, we have given a formal definition of the tenant placement problem, which we have identified as one of the key problems in cluster management for SaaS applications. This formalization is both a first step towards solving the placement problem in an efficient way and a helpful basis for comparing multiple solutions to the problem to each other. We have outlined a mechanism for the brute force enumeration of the complete search space. However, the brute force strategy is only useful for very small instances of the problem. As part of future work we thus plan to work on heuristic methods for solving the placement problem. In doing so, we plan to not only investigate and compare the theoretical properties of multiple placement algorithms but also to implement the algorithms in a multi-tenant database system and experimentally validate their practical usefulness.

## Acknowledgements

The authors would like to thank Michael J. Franklin from UC Berkeley for his valuable feedback and comments.

## References

[1] D. Jacobs and S. Aulbach, “Ruminations on Multi-Tenant Databases,” in *Datenbanksysteme in Business, Technologie und Web (BTW 2007), 12. Fachtagung des GI-Fachbereichs ”Datenbanken und Informationssysteme” (DBIS), Proceedings, 7.-9. März 2007, Aachen, Germany*, 2007, pp. 514–521.

[2] M. Myer, “Rightnow architecture,” in *HPTS*, 2007.

[3] “Amazon ec2 pricing,” http://aws.amazon.com/ec2/pricing/, last visited 2011-12-16.

[4] D. J. DeWitt, M. G. Smith, and H. Boral, “A Single-User Performance Evaluation of the Teradata Database Machine,” in *High Performance Transaction Systems, 2nd International Workshop, Asilomar Conference Center, Pacific Grove, California, USA, September 28-30, 1987, Proceedings*, 1987, pp. 244–276.

[5] J. Schaffner, B. Eckart, D. Jacobs, C. Schwarz, H. Plattner, and A. Zeier, “Predicting in-memory database performance for automating cluster management tasks,” in *Proceedings of the 27th International Conference on Data Engineering, ICDE 2011, April 11-16, 2011, Hannover, Germany*, 2011, pp. 1264–1275.

[6] J. Schaffner, D. Jacobs, B. Eckart, J. Brunnert, and A. Zeier, “Towards enterprise software as a service in the cloud,” in *Workshops Proceedings of the 26th International Conference on Data Engineering, ICDE 2010, March 1-6, 2010, Long Beach, California, USA*, 2010, pp. 52–59.

[7] J. Schaffner, B. Eckart, C. Schwarz, J. Brunnert, D. Jacobs, A. Zeier, and H. Plattner, "Simulating Multi-Tenant OLAP Database Clusters," in *Datenbanksysteme für Business, Technologie und Web (BTW), 14. Fachtagung des GI-Fachbereichs "Datenbanken und Informationssysteme" (DBIS), 2.-4.3.2011 in Kaiserslautern, Germany*, 2011, pp. 410–429.

[8] M. Grund, J. Schaffner, J. Krüger, J. Brunnert, and A. Zeier, "The effects of virtualization on main memory systems," in *Proceedings of the Sixth International Workshop on Data Management on New Hardware, DaMoN 2010, Indianapolis, IN, USA, June 7, 2010*, 2010, pp. 41–46.

[9] M. Gendreau, G. Laporte, and F. Semet, "Heuristics and lower bounds for the bin packing problem with conflicts," *Computers & OR*, vol. 31, no. 3, pp. 347–358, 2004.

[10] P. Haller and M. Odersky, "Actors That Unify Threads and Events," in *Coordination Models and Languages, 9th International Conference, COORDINATION 2007, Paphos, Cyprus, June 6-8, 2007, Proceedings*, 2007, pp. 171–190.

[11] D. Flynn and M. Adamson, "Capacity Planning on a Teradata DataWarehouse," in *34th International Computer Measurement Group Conference, December 7-12, 2008, Las Vegas, Nevada, USA, Proceedings*, 2008, pp. 93–104.

[12] D. J. DeWitt, S. Ghandeharizadeh, D. A. Schneider, A. Bricker, H.-I. Hsiao, and R. Rasmussen, "The Gamma Database Machine Project," *IEEE Trans. Knowl. Data Eng.*, vol. 2, no. 1, pp. 44–62, 1990.

[13] G. P. Copeland, W. Alexander, E. E. Boughter, and T. W. Keller, "Data Placement In Bubba," in *Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data, Chicago, Illinois, June 1-3, 1988*, 1988, pp. 99–108.

[14] P. Celis, "Distribution, Parallelism, and Availability in NonStop SQL," in *Proceedings of the 1992 ACM SIGMOD International Conference on Management of Data, San Diego, California, June 2-5, 1992*, 1992, p. 225.

[15] D. Saccà and G. Wiederhold, "Database Partitioning in a Cluster of Processors," *ACM Trans. Database Syst.*, vol. 10, no. 1, pp. 29–56, 1985.

[16] H.-I. Hsiao and D. J. DeWitt, "Chained Declustering: A New Availability Strategy for Multiprocessor Database Machines," in *Proceedings of the Sixth International Conference on Data Engineering, February 5-9, 1990, Los Angeles, California, USA*, 1990, pp. 456–465.

[17] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: amazon's highly available key-value store," in *Proceedings of the 21st ACM Symposium on Operating Systems Principles 2007, SOSP 2007, Stevenson, Washington, USA, October 14-17, 2007*, 2007, pp. 205–220.

[18] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin, "Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the world wide web," in *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*. New York, NY, USA: ACM, 1997, pp. 654–663.

[19] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 17–32, 2003.

[20] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A Distributed Storage System for Structured Data," *ACM Trans. Comput. Syst.*, vol. 26, no. 2, 2008.

[21] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles 2003, SOSP 2003, Bolton Landing, NY, USA, October 19-22, 2003*, 2003, pp. 29–43.

[22] L. W. Dowdy and D. V. Foster, "Comparative Models of the File Assignment Problem," *ACM Comput. Surv.*, vol. 14, no. 2, pp. 287–313, 1982.

[23] Y. Saito, C. T. Karamanolis, M. Karlsson, and M. Mahalingam, "Taming Aggressive Replication in the Pangaea Wide-Area File System," in *OSDI*, 2002.

[24] M. Karlsson and C. T. Karamanolis, "Choosing Replica Placement Heuristics for Wide-Area Systems," in *24th International Conference on Distributed Computing Systems (ICDCS 2004), 24-26 March 2004, Hachioji, Tokyo, Japan*, 2004, pp. 350–359.

# *Evaluation of Data Anonymization Tools*

Sergey Vinogradov
Corporate Technology
Siemens LLC
Saint-Petersburg, Russia
sergey.vinogradov@siemens.com

Alexander Pastsyak
Corporate Technology
Siemens LLC
Saint-Petersburg, Russia
alexander.pastsyak@siemens.com

***Abstract* — This survey became possible due to coming request of one of Siemens Business Units to look for data anonymization solutions being presented in the market today. The customer plans to implement and deploy it within software development projects to provide offshore team with a fully functional environment without any critical data in it. Critical data are, for instance, Personal Identifiable Information (PII), which is related to the nature of business application to be developed. In this survey paper, the introduction to data anonymization topic is given, the major challenges in data privacy an IT company may face during outsourcing of software development are considered, and the results of evaluation of data anonymization tools are provided.**

***Keywords-data anonymization; test data generation; pseudonymization; data masking, de-identification.***

## I. Introduction

The outsourcing of software development is becoming a common practice in Siemens due to global structure of the company. Either the whole development process or its particular phases like design, implementation or testing can be outsourced. Transfer of the testing process only to a remote location is still a rare practice since it requires special consideration regarding intellectual property, security and privacy [1]. In this paper, we consider the problem of data privacy during outsourcing of the testing process for those business domains where data management in applications is especially important [2]. Usually, such applications deal with a lot of private information such as names, addresses, phone numbers, customer names, bank accounts, transactions, so, it is very important to hide this information from off-shore test team [3]. On the other hand, the environment for the test team has to be as identical to the production as possible. In this situation, data anonymization solution may help.

In this work, we provide an introduction to data anonymization topic (Section II), review of major challenges related to data privacy that an IT company may face during outsourcing of software development (Section II), and evaluation results of data anonymization tools from vendors in SAP and non-SAP domains (Sections III and IV).

## II. Data Anonymization

### *A. Definitions*

Data Anonymization (also referred as data obfuscation, data masking, de-sensitization, de-identification or data scrubbing) is the process that helps to conceal private data. It protects sensitive information in production data base so it can be transferred to a test team. Data anonymization can be classified to pure anonymization [2] and pseudo-anonymization [4]. Pure anonymization does not provide any possibility to reconstruct the initial data, while pseudo-anonymization indeed provides such possibility through special algorithms. The former approach is the most reliable when the highest security is required, while the latter one might be interesting in the situation when the issue found by the test team has to be reproduced with production data values.

Let us consider the following example: a database with personal information (names, birth dates, bank accounts) need to be transferred to the offshore test team. The following typical data anonymization approaches can be applied to hide sensitive information:

- Data generation. Completely new data are generated. Special cases for dates and bank accounts need to be properly handled.
- Data encryption. The data is simply encrypted. Can be restored if the key is saved.
- Shuffling. The data is shuffled in one column. In this case the combination (name, bank account) will not be real.

Also pseudo anonymization approach can be implemented with almost any anonymization technique in the following way: there is a special database, which keeps track of the changes during application of the anonymization algorithms. If the reverse operation is required the lookup over this database returns the old value.

In the real situation, the combination of the approaches applied for different data fields can be considered: data generation for card numbers, data encryption for some description field and shuffling for personal names and addresses.

In our tools evaluation, we consider different anonymization algorithms as well as other criteria described in sub-section B.

### *B. Criteria*

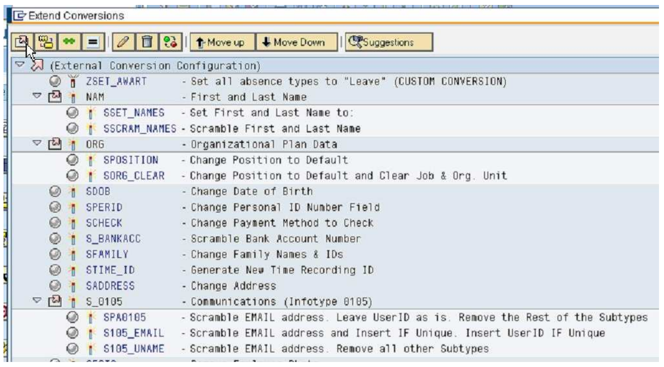
The following criteria have been used to evaluate data anonymization tools. We must note that the criteria described below belong to purely technical features of tools, while business-related requirements (like licensing, maintenance costs, etc.) of the tools overview are out of scope of this paper.

*SAP and non-SAP solution* – this criterion points to a tool's ability to cope with data management applications from SAP (SAP is a market leader in enterprise application software. SAP stands for Systems, Applications and Products in Data Processing) and non-SAP domains. Significant number of Siemens business divisions uses SAP products in their projects.

According to [5], businesses that run SAP face a common challenge: how to get real SAP data into non-production systems for testing, training and Production support. The key issue is that, while new data reflecting the latest business activity is constantly being added to Production, business users cannot easily access this data. Non-production updates are essential for testing newly-developed features, production-support issues, and support packs. However, using live SAP data for testing is becoming increasingly difficult. Client or system copies disrupt the landscape, require large amounts of disk space, take a long time to prepare, and increase technical-support overheads. These challenges have to be taken into account as far as the testing activities are planned to outsource where data privacy become critical. So it becomes obvious that data anonymization solution for SAP applications cannot be standalone and has to be integrated in overall data-copy solution.

*Out of the box SAP schemas support* – a tool provides out-of-the-box the solution for different SAP types of the system, i.e., SAP ERP (Enterprise Resources Planning System including such parts as HCM – Human Capital Management, FI – Financing. LO – Logistics) and SAP CRM & SRM (Custom Relationship Management System and Supplier Relationship Management System, accordingly). Important issue here is a specific set of data within each SAP scheme and a tool's ability to mask such data.

Such variety of types of application data makes us searching for vendors which provide support on data copy and data anonymization for the most of SAP system types. Especially it concerns the availability of pre-defined data transformation (conversion) rules in a solution. The examples of such conversions for HR data implemented in Data Sync Manager for HCM tool are shown in Fig. 1.



```
(External Conversion Configuration)
    ZSET_AWART     - Set all absence types to "Leave" (CUSTOM CONVERSION)
    NAM            - First and Last Name
      SSET_NAMES   - Set First and Last Name to:
      SSCRAM_NAMES - Scramble First and Last Name
    ORG            - Organizational Plan Data
      SPOSITION    - Change Position to Default
      SORG_CLEAR   - Change Position to Default and Clear Job & Org. Unit
    SDOB           - Change Date of Birth
    SPERID         - Change Personal ID Number Field
    SCHECK         - Change Payment Method to Check
    S_BANKACC      - Scramble Bank Account Number
    SFAMILY        - Change Family Names & IDs
    STIME_ID       - Generate New Time Recording ID
    SADDRESS       - Change Address
    S_0105         - Communications (Infotype 0105)
      SPA0105      - Scramble EMAIL address. Leave UserID as is. Remove the Rest of the Subtypes
      S105_EMAIL   - Scramble EMAIL address and Insert IF Unique. Insert UserID IF Unique
      S105_UNAME   - Scramble EMAIL address. Remove all other Subtypes
```

Figure 1. Conversions for HR data.

*Multi Database* – underlying database used in a project puts certain constraints on a data anonymization solution. Ability to work with different databases or independence of underlying database widens the application scope of a tool. This criterion is not relevant and would not be considered for SAP-dedicated data anonymization solutions.

*Resulting security* – since we want to protect our sensitive data the level of security needs to be assured. Usually test data generation tools have the highest security level since the data in non-production environment will be completely different from the source system.

In general, data anonymization algorithms can be grouped in three categories: data generation algorithms, algorithms which are dealing with already existing data and the combination of the above.

The first group includes all algorithms, which create completely new entries in the data base; they are used for anonymization of bank accounts, credit card numbers, social security numbers, generating random numbers, dates, etc.

The second group of algorithms operates with already existing data in the data base. The typical examples in this group are: shuffling of the fields in one or several columns, encryption, scrambling the letters in the string or figures in numbers and so on.

Also it is possible to make a combination of the algorithms from the above groups, e.g., shuffle the fields in one of the data base columns and add a random number as a string literal at the end of the field.

*Preserving application and data integrity* – data anonymization technologies should satisfy a simple, yet strict rule: the application that runs against masked data performs as if masked data is real [6].

This is a MUST requirement for every data anonymization tool since without conforming to this criterion the resulting database will be useless. The main focus during evaluation has been done on the tool ability to preserve data relations automatically (without user assistance).

*Support of roles assignment* – tool offers different roles to operate and use tool's functions. There might be system administrator role, super user role, user role with different sets of access rights to project data.

According to [6], there are four phases for data masking lifecycle identified: Data discovery and analysis, Data planning and modeling, Developing and Implementation.

The goal of Data Analysis phase is to identify the data that needs to be masked in order to sufficiently protect the data without compromising data utility. At this phase, the highest level of access a tool's user has to be provided assuming the work with data mining and analysis of customer data.

The Planning and Modeling phase is designed to set in place the criteria that will be used within your environment to mask the data and create context around the info that was discovered in first phase. The work to be done at this phase is not supposed to deal with critical data itself, rather than is related to selection of data anonymization rules and data anonymization strategy at all.

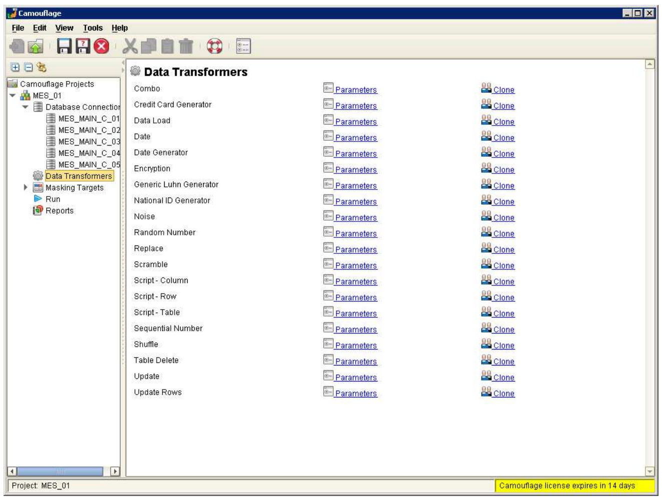
The Develop phase is designed to build data masking configuration suites based upon customer specific Functional Masking Needs. Again, the preparation of data anonymization scripts and the proper configuration of data masking rules within a tool do not assume the direct access to customer data.

The Implementation and Execution phase is designed to put in place a plan for integrating data masking into the overall production-to-non-production business process. This work mostly should be done in place (creating test database(s), moving masking scripts to source code implementation libraries, etc.); thus, requiring assignment of special persons to perform it and granting temporal access to the production database.

*Batch operation support* – due to possibly huge amount of sensitive data to be transferred to non-production environment, due to time-consuming tasks of creating non-production copies itself it would be useful to be able to plan and run such tasks from scripts.

*Algorithms for anonymization* – here, the different alternatives have been considered: test data generation, encryption, data masking within several masking rules, pseudo-anonymization, dynamic data masking, etc.

It has to be noted that most of the evaluated data anonymization tools provide possibility for customization of data masking algorithms. Such possibility allows for implementing almost every imaginable algorithm, however it requires significant time to study programming techniques (from simple Java$^{TM}$ subroutine in Camouflage Enterprise to ABAP routine in Data Sync Manager or even a pluggable C++ library in IBM Optim™.). That is why this particular criterion is focused on the algorithms which are provided by the tool out-of-the-box. The default list of data transformers available in the Camouflage Enterprise is presented on Fig. 2.



Data Transformers

| | | |
|---|---|---|
| Combo | Parameters | Clone |
| Credit Card Generator | Parameters | Clone |
| Data Load | Parameters | Clone |
| Date | Parameters | Clone |
| Date Generator | Parameters | Clone |
| Encryption | Parameters | Clone |
| Generic Luhn Generator | Parameters | Clone |
| National ID Generator | Parameters | Clone |
| Noise | Parameters | Clone |
| Random Number | Parameters | Clone |
| Replace | Parameters | Clone |
| Scramble | Parameters | Clone |
| Script - Column | Parameters | Clone |
| Script - Row | Parameters | Clone |
| Script - Table | Parameters | Clone |
| Sequential Number | Parameters | Clone |
| Shuffle | Parameters | Clone |
| Table Delete | Parameters | Clone |
| Update | Parameters | Clone |
| Update Rows | Parameters | Clone |

Project: MES_01

Camouflage license expires in 14 days

Figure 2. Default data transformers in Camouflage Enterprise

## III. Tools Analysis

There was a wide range of solutions found for data anonymization task in SAP and non-SAP domains. Moreover, in the market, we faced with several vendors who usually provide products in non-SAP domain, nevertheless they did not refuse to elaborate a data anonymization solution for SAP business applications (e.g., Grid Tools Ltd, UK). Other vendors, like IBM (US), are able to keep a separate product line to cope with data privacy projects in SAP domain. The tools found for each domain are summarized below.

There have been also found the test data generation solutions as well as non-commercial software.

Let us shortly introduce the found tools structured in a three domains: SAP, non-SAP and universal.

### *A. SAP tools*

**SAP® Test Data Migration Server (TDMS)** [7] software from the market leader allows for extracting data from the production system and creating test landscapes of lower volume of data. Despite of tight integration of this solution with SAP systems, such issues like unknown anonymization algorithm (sales department could not provide such information) and time-consuming installation and setup procedure to be performed only by SAP consultants compromised the perspectives of this tool.

**Accenture Clone and Test HCM (Accenture Software for SAP HCM)** [8] enables the simple configuration of reliable and realistic test environments using actual data from the current SAP ERP HCM system. Within copying data and creating clone the tool provides rule-based data scrambling. The solution deserves the serious attention due to no specific customization during setup and relatively fast copying procedure. The clear disadvantage of such tool is its limit of support of SAP schemas.

The product from **GASPARIN Software Solutions, hr.dat.copy** [9] is a pure transport of personnel data (HR data) across systems and/or clients. It allows for modification or anonymization of personnel numbers or other sensitive data; however the application scope of such solution is rather limited.

**BCV5™** solution from **(Enterprise Systems Associates, Inc, IBM affiliate)** [10] stands for fast, reliable copying and refresh / replication of DB2 data. It is rather fast solution customized for IBM mainframe.

**Data Sync Manager for HCM** from **EPI-USE** [5] pretends to be the most complete data anonymization solution for SAP systems (support can be extended to other schemas of SAP ERP as well as to SAP SRM and CRM). The solution is realized as a transport to both SAP source and target systems, thus no additional hardware and middleware required. Along with seamless integration with SAP system the setup of the tool requires no specific customization, and graphical representation of workflow for creating clones along with data masking procedures proved to be very clear and comprehensible.

### *B. non-SAP tools*

With **Oracle Enterprise Manager Data Masking Pack** [11] from **Oracle** sensitive information such as credit card or social security numbers can be replaced with realistic values, allowing production data to be safely used in development and testing or shared with out-source or off-shore partners for other non-production purposes. The clear advantage of this solution is a library of templates and format rules used, constantly transforming data in order to maintain referential integrity for applications. However, the possibility to use the tool with other DBs needs to be investigated.

**Camouflage Enterprise** from **Camouflage Software Inc., US** [6] provides complete tool-chain for data management tasks from data sub setting and data masking to creating copies of production system. It also delivers the highest level of customizable masking with Masking Engine, Scripting Engine and database specific transformers. Extendable and scalable for large organizations, with highly user-friendly interface this solution proves to be a key player in the market of non-SAP data anonymization solutions.

**Jumble DB** from **Orbium Software** [12] is a complete data scrambling solution for Oracle and SQL Server databases. Despite of moderate scalability and standard data masking features (scrambling), the vendor declares the tool’s capability to keep referential integrity intact.

**FieldShield** product from **Innovative Routines International (IRI), Inc.** [13] masks private data at the field level with obfuscation and encryption functions which are applied according to your business rules. It works with data in the format of sequential flat files extracted from database (Oracle and DB2 are supported). Despite of multiplatform support and wide set of data anonymization technologies (from encryption till masking via custom functions) this solution offers non-trivial workflow which threatens to be very time-consuming.

**DataVantage Global ®** [14] solution from **Direct Computer Resources, Inc.,** protects confidential health, financial, personnel and other data and uses data obfuscation and encryption methods for this. It declares multi-database support and provides only graphical user interface to perform data anonymization tasks.

**Data Masker** [15] software from **Net 2000 Ltd** removes sensitive data from test databases and replaces it with realistic looking false information. Along with existing data scrambling rules a user is allowed to define her ones. Tools editions are defined on a per DB type base (Oracle, SQL, DB2).

### C. Tools providing universal solutions

**Datamaker™** [3] solution from **Grid Tools Ltd** creates data from scratch, creates subset databases, de-identifies (mask or obfuscate) existing data and bulks up data for performance testing. The vendor declares the support of all known databases across multiple platforms. It is worth to be mentioned that based on existing products the solution for data anonymization in SAP domain can be found and developed as well.

**Tricryption®** [16] solution from **Eruces, Inc.** provides secure replacement of sensitive identifiable data with anonymous but unique alias/pseudonym labels. Pros and cons of pseudo anonymization were discussed in Section II. Nevertheless, the encryption technology used in this product might be rather powerful to gain the resulting security of data anonymization while preserving the possibility to recover data. Anyway it might be useful for so called “anonymization in place” and not relevant in the case of offshore development and testing.

**dgmasker™** [1] solution from **dataguise Inc.** masks data to help enterprises meet various compliance requirements such as PCI, HIPAA, GLBA, PII and SOX. The vendor declares the support of Oracle, DB2 and MS SQL Server databases, multiple advanced masking algorithms and tool’s capability to perform data anonymization of SAP applications.

**ActiveBase Security™** [17] product **ActiveBase Ltd** is another novel solution which offers a new approach to database security. It protects production environment by adding a security layer within and around business applications, masking or scrambling sensitive information in real time with no changes to applications or databases. Due to this independence of the solution from underlying data model in database SAP ERP, CRM applications are supported.

**Optim™** [18] from **IBM** presents the complete data management solution for all known databases and significant number of application types like SAP Apps, PeopleSoft, Jd Edwards EnterpriseOne, Siebel Apps etc. It delivers powerful data transformation capabilities to mask personal information within a structured workflow of extracting data from production and sending it to development, test and training systems on demand.

## IV. EVALUATION PROCEDURE

The evaluation procedure for the above tools has been separated into three different phases:

*Market Evaluation* – Comprised high level market analysis to choose most promising tools with regard to specified criteria for further technical evaluation.

This study has been performed using available marketing materials and calls to vendor representatives. 16 different tools described in Section III were found and screened according to the specified criteria. As the result of this phase 8 different tools proposed for further studies had been presented to the customer, who selected three of them for technical evaluation.

*Solution Design* – Comprised the preparation for technical evaluation of 4 tools-candidates selected at the previous project phase.

The technical evaluation of selected tools required the representative database, which has been defined and prepared during this phase. Two systems were used: Oracle database with MES data for evaluation of tools’ capabilities in non-SAP domain and system with SAP HR data for evaluation of tools in SAP domain.

*Technical Evaluation* – Comprised evaluation of the tools-candidates which was intended to perform on test system with real data. The study included application of selected solutions and clarification of its capabilities on sample datasets.

This phase has been carried out by applying the tools in the real data anonymization scenarios. Special use cases have been designed to verify tools capabilities: 10 use cases for non-SAP

TABLE I. CROSS COMPARISON OF EVALUATION CRITERIA

| Criteria | SAP and Non-SAP | SAP Scheme support | Multi Database | Multi Platform | Technical Features | Licensing Costs | Maintenance Costs | Setup Costs | Service / Support | Training costs | Customization costs | Required user profile | Points per Criterion | Relative Benefit Factor |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **SAP and Non-SAP** | - | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 8 | 0,06 |
| **SAP Scheme support** | 1 | - | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 18 | 0,14 |
| **Multi Database** | 2 | 1 | - | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 20 | 0,15 |
| **Multi Platform** | 2 | 1 | 1 | - | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 20 | 0,15 |
| **Technical Features** | 2 | 1 | 0 | 0 | - | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 17 | 0,13 |
| **Licensing Costs** | 1 | 0 | 0 | 0 | 0 | - | 2 | 1 | 2 | 2 | 1 | 2 | 11 | 0,08 |
| **Maintenance Costs** | 1 | 0 | 0 | 0 | 0 | 0 | - | 0 | 1 | 0 | 0 | 2 | 4 | 0,03 |
| **Setup Costs** | 1 | 0 | 0 | 0 | 0 | 1 | 2 | - | 1 | 0 | 0 | 2 | 7 | 0,05 |
| **Service / Support** | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | - | 1 | 0 | 2 | 6 | 0,05 |
| **Training costs** | 1 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 1 | - | 0 | 2 | 8 | 0,06 |
| **Customization costs** | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 2 | 2 | 2 | - | 2 | 12 | 0,09 |
| **Required user profile** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | - | 1 | 0,01 |

data and 12 use cases for SAP data. These use cases covered common difficulties in data anonymization process: foreign-primary key relationships, triggers, dependency between different columns (e.g., dates dependency), user-defined objects and fields in SAP system, bank accounts, application of custom anonymization algorithms, etc.

Quantitative assessment of capabilities of 4 tools-candidates turned to be possible at this phase. Firstly, the customer provided the cross comparison of the evaluation criteria in order to define the “relative importance” of each criterion. The results of such cross comparison are presented in Table I. Secondly, there was performed the benefit value analysis of the tools-candidates which received as an input the list of weighted criteria and the criteria values from our technical evaluation. The results of the benefit value analysis are summarized in Table II.

## V. SUMMARY

The results of data anonymization tools market evaluation have shown that there is a wide range of solutions available for data anonymization task in SAP and non-SAP domains. The solutions usually offer the longer list of functionality for creating non-production environment in addition to data anonymization, e.g., identification of sensitive data, data subsetting, application templates, and data copying. However in most cases practical applicability of the promising data anonymization solutions has to be confirmed in the course of more detailed study with involvement of trial versions of evaluated solutions.

**Camouflage DLM Suite (Camouflage Software Inc.)** and **DataVantage Global® tools** in non-SAP domain and **Data Sync Manager (EPI-USE)** for SAP domain have been preliminary selected as the most promising data solutions due to the following distinctive characteristics:

- User-friendly GUI and relatively short learning curve
- Comprehensive workflow and intuitive interface
- Available predefined conversions/masking rules for the most types of sensitive data
- Sensitive data identification can be provided by the vendor.

In this work data anonymization solutions only have been studied. Nevertheless, Camouflage DLM Suite aims at full functional support of creating non-production environment (sub setting, sensitive data identification, data masking) and provides the average learning curve for future users. Camouflage allows for masking the data in non SAP applications working with databases.

DSM covers SAP application domain providing transparent workflow for creating non-production environment and large set of pre-defined conversions for SAP ERP business objects.

TABLE II. BENEFIT VALUE ANALYSIS

| **Tool Comparison** | | **Camouflage Enterprise (DLM Suite)** | | **IBM Optim™ Data Privacy 7.2** | | **IBM InfoSphere Optim TDM for SAP** | | **Data Sync Manager for HCM** | |
|---|---|---|---|---|---|---|---|---|---|
| **Criteria** | ***Relative Benefit Factor*** | ***Value for tool 1 (0-10)*** | ***Relative Value Tool 1*** | ***Value for tool 2 (0-10)*** | ***Relative Value Tool 2*** | ***Value for tool 3 (0-10)*** | ***Relative Value Tool 3*** | ***Value for tool 4 (0-10)*** | ***Relative Value Tool 4*** |
| **SAP and Non-SAP** | 0,06 | 0 | 0,00 | 0 | 0,00 | 0 | 0,00 | 0 | 0,00 |
| **SAP Scheme support** | 0,14 | 0 | 0,00 | 0 | 0,00 | 3 | 0,41 | 9 | 1,23 |
| **Multi Database** | 0,15 | 10 | 1,52 | 10 | 1,52 | 10 | 1,52 | 10 | 1,52 |
| **Multi Platform** | 0,15 | 10 | 1,52 | 10 | 1,52 | 10 | 1,52 | 10 | 1,52 |
| **Technical Features** | 0,13 | 10 | 1,29 | 8 | 1,03 | 3 | 0,39 | 10 | 1,29 |
| **Licensing Costs** | 0,08 | 9 | 0,75 | 7 | 0,58 | 4 | 0,33 | 5 | 0,42 |
| **Maintenance Costs** | 0,03 | 5 | 0,15 | 5 | 0,15 | 5 | 0,15 | 6 | 0,18 |
| **Setup Costs** | 0,05 | 9 | 0,48 | 7 | 0,37 | 7 | 0,37 | 5 | 0,27 |
| **Service / Support** | 0,05 | 7 | 0,32 | 6 | 0,27 | 4 | 0,18 | 9 | 0,41 |
| **Training costs** | 0,06 | 8 | 0,48 | 4 | 0,24 | 4 | 0,24 | 7 | 0,42 |
| **Customization costs** | 0,09 | 8 | 0,73 | 6 | 0,55 | 1 | 0,09 | 6 | 0,55 |
| **Required user profile** | 0,01 | 8 | 0,06 | 5 | 0,04 | 4 | 0,03 | 6 | 0,05 |
| **Total Benefit Value** | 1,00 | | 7,29 | | 6,27 | | 5,23 | | 7,83 |

## REFERENCES

[1] Why Add Data Masking to Your Best Practices for Securing Sensitive Data, Dataguise Inc., Whitepaper, 2009. www.dataguise.com <retrieved: October, 2011>

[2] www.datactics.com <retrieved: October, 2011>

[3] www.grid-tools.com <retrieved: August, 2011>

[4] www.sapior.com <retrieved: October, 2011>

[5] www.epiuse.com <retrieved: November, 2011>

[6] Data Masking Best Practices, Camouflage Inc., Whitepaper, March 2010. www.datamasking.com <retrieved: November, 2011>

[7] www.sap.com <retrieved: October, 2011>

[8] www.ehr-solutions.de <retrieved: October, 2011>

[9] www.gasparin.at <retrieved: August, 2011>

[10] www.esaigroup.com, http://ubs-hainer.com/en/db2-products/bcv5-fast-cloning-of-db2-data <retrieved: October, 2011>

[11] www.oracle.com<retrieved: October, 2011>

[12] www.orbiumsoftware.com <retrieved: October, 2011>

[13] www.iri.com <retrieved: October, 2011>

[14] Data Solutions for Data Privacy, Direct Computer Resources Inc., Whitepaper, June 2010. www.datavantage.com <retrieved: October, 2011>

[15] www.datamasker.com/index.html <retrieved: August, 2011>

[16] www.eruces.com <retrieved: August, 2011>

[17] www.active-base.com <retrieved: October, 2011>

[18] http://www-01.ibm.com/software/data/data-management/optim-solutions <retrieved: October 2011>

# Dynamic Stream Allocation with the Discrepancy between Data Access Time and CPU Usage Time

Sayaka Akioka
IT Research Organization
Waseda University
Tokyo, Japan
Email: akioka@aoni.waseda.jp

Yoichi Muraoka, and Hayato Yamana
Dept. of Computer Science and Engineering
Waseda University
Tokyo, Japan
Email: {muraoka, yamana}@waseda.jp

***Abstract*—Huge quantities of data arriving in chronological order are one of the most important information resources, and stream mining algorithms are developed especially for the analysis of the fast streams of data. A stream mining algorithm usually refers to the input data only once and never revisits them (read-once-write-once), while the conventional data intensive applications refer to the input data in a write-once-read-many manner. That is, once the stream mining falls behind, the process drops the input data until it catches up with the input data stream. Therefore, the fast execution of the stream mining leads the perfect analysis on all the input data, and it is very critical for the quality of the service. We propose a dynamic resource management for the stream mining in the distributed environment. The resource management utilizes the discrepancy between data access time and CPU usage time inside the stream mining, and speeds up the mining process. We implemented the methodology and proved successfully to process all the input data of such a fast data stream, whereas the serial execution drops more than 90% of the input data.**

***Keywords-Load Balancing; Resource Management; Stream Mining.***

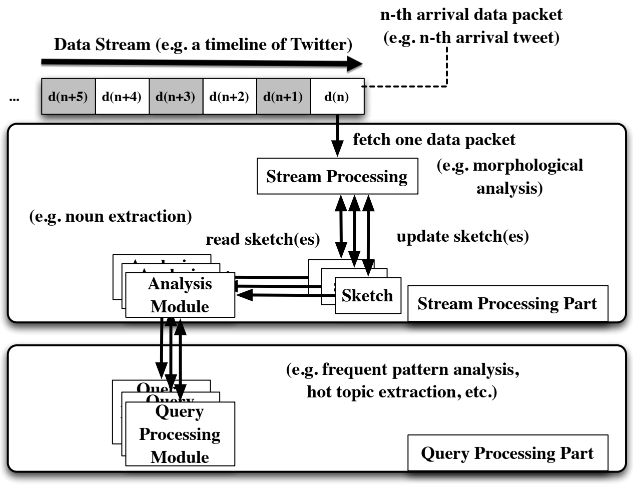


Fig. 1. A model of stream mining algorithms.

## I. Introduction

A data stream, which is a sequence of data arriving in chronological order, is one of the significant information resources. The data streams include consumer generated media and mini blogs and many projects are trying to extract useful information through the analysis of the data streams. One of the technical obstacles for the scalable and real-time analysis of the data streams is their characteristic data access pattern. A stream mining algorithm is often utilized for the analysis of the data streams, and the stream mining algorithm refers to the input data only once. The input data themselves are never stored anywhere and never revisited. Additionally, the data stream rate is not consistent in the actual situation, and the computational cost of each stage is hard to clarify in advance. Therefore, this paper proposes a dynamic resource management focusing on the discrepancy between data access time and CPU usage time for the purpose of the practical speed-up of the stream mining algorithm in the highly distributed computational environment.

The rest of this paper is organized as follows. Section 2 introduces a stream mining algorithm, analyzes its model, and defines the problem to address in this paper. Section 3 describes the resource management methodology, and Section 4 discusses the effect of the resource manager through some validations in the actual cloud environment. Section 5 covers the related work, and Section 6 concludes this paper.

## II. Problem Definition

### A. A model of the stream mining algorithms

A stream mining algorithm is an algorithm specialized for a data stream analysis. Gaber et al. reviewed the theoretical foundations of the stream mining algorithms [1]. There are also many variations of the stream mining algorithms including clusterings [2]–[11], classifications [6], [12]–[16], frequency countings [17], [18], and time series analysis [19]–[24]. Regardless of the difference among the details of the stream mining algorithms, general stream mining algorithms share a fundamental structure and a data access pattern as shown in Figure 1, which Akioka et al. proposed [25].

A stream mining algorithm consists of two parts, which are the stream processing part and the query processing part. The major responsibility of the stream processing part is to process each data unit and extract the essence of the data for the further analysis by the query analysis part. The stream processing part needs to finish the processing of the current data unit before the next data unit arrives, otherwise, the next data unit will be lost as there is no storage for buffering the incoming data in a stream mining algorithm. On the other hand, the query processing part takes care of the further analysis such as a frequent pattern analysis and a hot topic extraction based on the intermediate data passed by the stream processing part, which a database system stocks usually. This clear difference between the responsibilities of these two parts indicates that only the stream processing part needs to run on a real-time basis. The successful processing of all the incoming data simply relies on the speed of the stream processing part.

The process flow for a single data unit in the stream processing part is as follows. First, the stream processing module picks the target data unit, and executes a quick analysis over the data unit, such as a morphological analysis and a word counting. Second, the stream processing module updates the data cached in one or more sketches with the latest results through the quick analysis. That is, the sketches keep the intermediate analysis, and the stream processing module updates the analysis incrementally as more data units are processed. Third and finally in the stream processing part, the analysis module reads the intermediate analysis from the sketches, and extracts the essence of the data, which is passed to the query processing part for the further analysis.

### B. Motivation

There is a certain market requesting the complete analysis of one or more data streams on the fly. However, there is no solid solution for this problem targeting a large-scale, general stream mining algorithm yet. The model of a stream mining algorithm shown in Figure 1 indicates that the data access pattern of the stream mining algorithms are totally different from the data access pattern of so-called data intensive applications, which is intensively investigated in the high performance computing area. The data access pattern in the data intensive applications is write-once-read-many [26]. That is, the application refers to the necessary data many times during the computation; therefore, the key for the speed-up of the application is to place the necessary data close to the computational nodes for faster data access. On the other hand, in a stream mining algorithm, a process refers to its data unit only once, which is read-once-write-once style. Therefore, conventional techniques for the data intensive applications are not simply applicable for the speed-up or the complete analysis of the data stream.

## III. Methodology

Following the discussion up to Section 2, there are two technical challenges to address. One of the challenges is the resource management for a scalable and complete analysis of the fast data stream. The other challenge is a generic framework for various stream mining algorithms. We propose a resource manager as a solution for these challenges, which enables execution of stream mining algorithms in the pipeline. The resource manager accepts data streams and scatters analysis tasks over the distributed computational environment such as the cloud and the grid. The resource manager also accommodates the number of the computational nodes to utilize dynamically according to the speed of the input data stream and the load of the analysis process. The rest of this section describes the details of the resource manager focusing on the stream processing part. As we already discussed in Section 2, only the stream processing part needs to run in a real-time manner.

### A. Data Dependency and CPU occupancy

In a stream mining algorithm, one process of a data unit generates data dependency to the processes of the successive data units, and this is the major reason why the resource manager allocates the processes in a pipeline manner. As described in Section 2, the stream processing module updates the data in the sketches. That is, the sketches produce the data dependency across the processes. Figure 2 illustrates the data dependencies between two processes processing data units in line and the data dependency inside the process. More concretely, there are three data dependencies in Figure 2. One of the data dependencies is that the processing module in the preceding process should finish updating of the sketches before the processing module in the successive process starts reading the sketches (Dep.1 in Figure 2). Another data dependency is the processing module should finish updating the sketches before the analysis module in the same process starts reading the sketches (Dep.2 in Figure 2). The last data dependency is that the analysis module in the preceding process should finish reading the sketches before the processing module in the successive process starts updating the sketches (Dep.3 in Figure 2). These three dependencies are essential to keep the analysis results consistent and correct.

The data dependencies indicate that the access to the sketches reasonably divides the processing module into the two parts. That is, the stream processing module and the analysis module are reasonably managed independently, and these two modules in the same process are ideally allocated on the same CPU for the purpose to minimize the access cost to the sketches. Once we decide to allocate the processing module and the analysis module independently, the possible situation is that the processing module in one process occupies the CPU resource only waiting for the access to the sketch by the analysis module in the same process or the processing module in the successive process. This is the discrepancy between data access time and CPU usage time. The flexible resource management avoids this waste of the CPU resources. We revisit this problem and give the details in the next section.

### B. Dynamic Resource Management

Following the discussion above, the resource manager allocates tasks taking care of both the three data dependencies and the discrepancy between data access time and CPU usage time. Here, the unit of the task is the former half or the latter half of the stream processing part, as shown in Figure 3.

The resource manager maintains three tables, which are the processing modules table, the analysis modules table, and the CPUs table, as shown in Figure 4 in order to manage the resources and tasks. Each entry of the processing modules

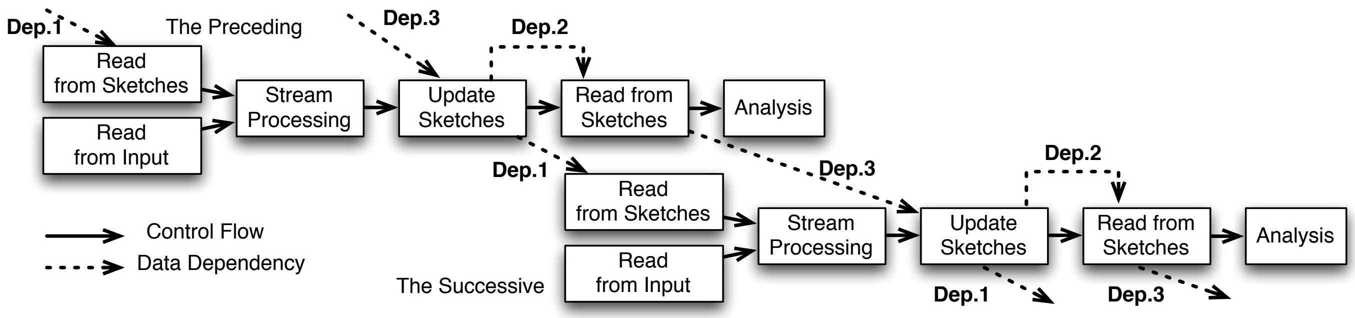


Fig. 2. The data dependencies between the two processes, which are processing two data units in line.

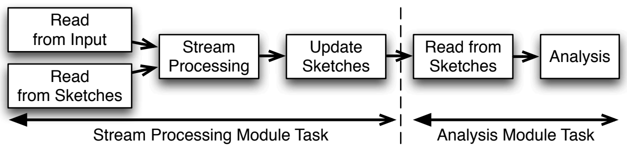


Fig. 3. The processing part consists of the two parts; the processing module and the analysis module. The resource manager recognizes each module as a task, and decides the resource allocation.

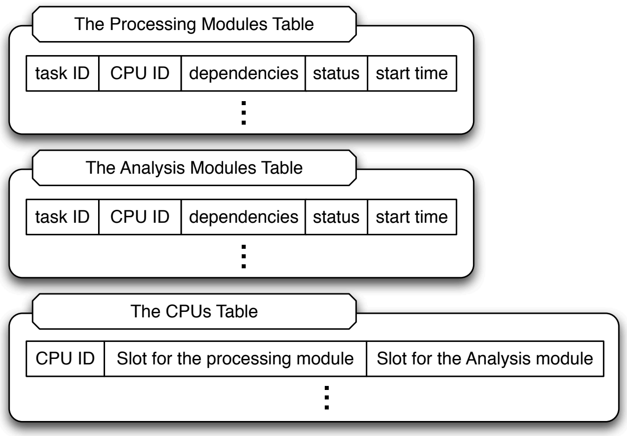

The Processing Modules Table
task ID
CPU ID
dependencies
status
start time
The Analysis Modules Table
task ID
CPU ID
dependencies
status
start time
The CPUs Table
CPU ID
Slot for the processing module
Slot for the Analysis module

Fig. 4. The resource manager maintains the three tables; the processing modules table, the analysis modules table, and the CPUs table. These tables tell the resource manager the availability of the CPU resources, and the statuses of the ongoing tasks.

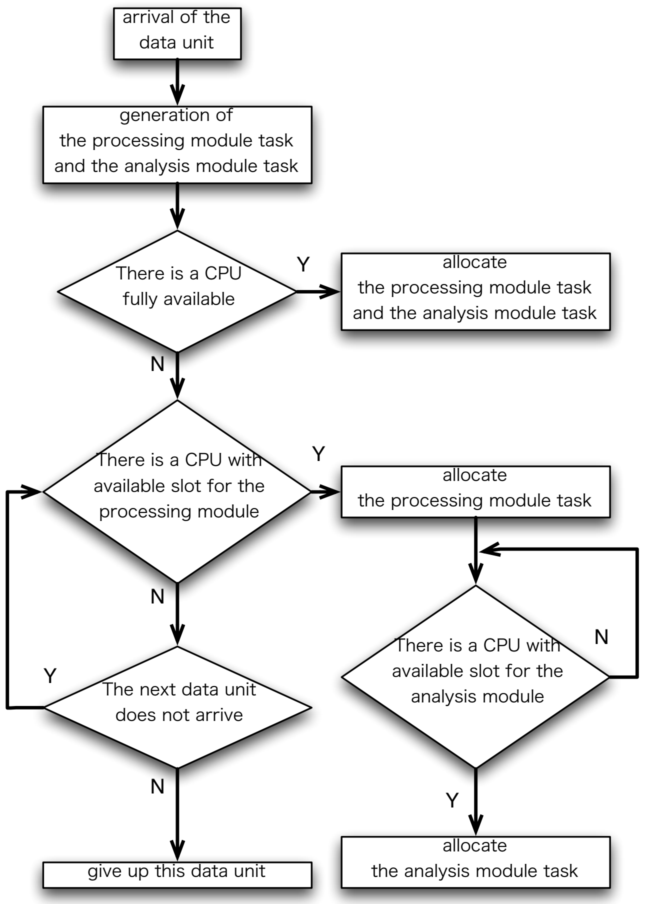

arrival of the data unit
generation of the processing module task and the analysis module task
There is a CPU fully available
Y
allocate the processing module task and the analysis module task
N
There is a CPU with available slot for the processing module
Y
allocate the processing module task
N
There is a CPU with available slot for the analysis module
N
Y
The next data unit does not arrive
N
Y
give up this data unit
allocate the analysis module task

Fig. 5. The procedure of the resource manager.

table and the analysis modules table represents each task. The entry consists of the task id of this task, the CPU id where this task runs on, the task ids those this task has data dependency with, the start time of this task, and the current status of this task. Each field of the entry of the tables lets the resource manager know the data dependencies among the ongoing tasks, and whether the ongoing task occupies the CPU resource only waiting for the data access or not. The resource manager also maintains the CPUs table, and each entry of the table represents each CPU resource. Each entry consists of two fields, which are one field for the processing module task, and the other field for the analysis module task. The CPU table tells the resource manager the availability of CPU units.

Figure 5 illustrates the procedure of the resource manager. On the arrival of the data unit, the resource manager generates two tasks. One task is for the processing module, and the other task is for the analysis module. The resource manager generates a task id and an entry of the corresponding table in Figure 4. Once the processing module task and the analysis module task are ready, the resource manager decides the

TABLE I
THE SPECS OF THE COMPUTATIONAL NODES AND THE RESOURCE MANAGER.

| Node Type | Num. of nodes | CPU+Memory |
|---|---|---|
| Node Type A | 32 nodes | Xeon X5550+32GB |
| Node Type B | 14 nodes | Xeon X5450+8GB |
| Node Type C | 6 nodes | Xeon X5430+8GB |
| Resource Mngr. | 1 node | Xeon E5520+24GB |

resource allocation for these tasks. Here, there are three possible situations. First situation is that there is an available CPU whose slots are available both for the processing module and for the analysis module. In this case, the resource manager allocates the corresponding tasks onto this CPU, and returns to wait for the successive data units. Second situation is that there is an available CPU only the slot for the processing module is available. This time, the resource manager allocates only the processing module onto this available CPU, and waits until the slot for the analysis module on any CPU becomes available. As the processing module of this data unit never starts before the analysis module of the preceding data unit starts, the allocation of the analysis module is not urgent. Third and the worst situation is that there is no available slot for the analysis module. In this situation, the resource manager tries to find an available slot until the next data unit arrives, and finally gives up the corresponding data unit when no slot turns to be available.

## IV. VALIDATION

We implemented the resource manager as a Java program and validated the effect of the resource manager in the actual computational cloud environment. We utilized 53 nodes of IBM Computing on Demand (CoD) as the cloud environment, and all the computational nodes are connected each other via Gigabit Ethernet. Table I summarizes the specs of the computational nodes of CoD. As shown on Table I, the computational environment was heterogeneous. However, this version of the resource manager does not consider the variations of the computational nodes, and the further functionality with the heterogeneous environment is reserved as future work.

The input data stream was generated inside of CoD under the assumption of a very fast data stream. More concretely, we took Twitter as a model of the input data stream. The data stream generator randomly chooses one post from the 1000 tweets pool, which is the actual sample from Twitter timeline. As the total number of the tweets per day is approximately 100 million, therefore, the tweet generator posts one tweet every millisecond in order to form a real Twitter like input data stream. We concluded this is the best way to simulate Twitter data stream, because collecting all the posts via Twitter API is practically almost impossible. Each computational node performs morphological analysis over the data unit in the processing module, and counts the appearances of the specific expressions such as URL or referred accounts in the analysis module. If either the processing module or the analysis module

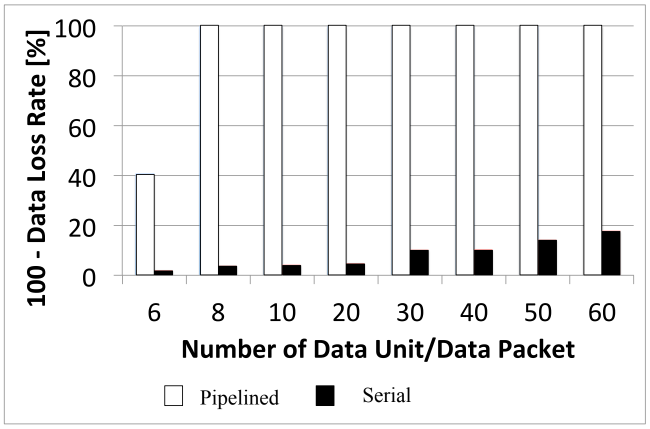


Fig. 6. The rates of the processed of the incoming data. The higher is better.

takes longer and results to break the dependencies discussed in Section 3, the corresponding data unit is given up as the analysis qualitywill not be ensured anymore. All the analysis modules are implemented with Java and Java RMI.

Figure 6 plots the rates of the processed of the incoming data units in comparison between the pipelined executions with the proposed resource manager and the serial executions. Here, let $n_{total}$ be the total number of the incoming data units, and let $n_{processed}$ be the total number of the successfully processed data units among the incoming data units. Following these definitions, the rate of the processed of the incoming data units $R_{processed}$ is defined as follows.

$$R_{processed} = 100(1 - \frac{n_{processed}}{n_{total}})$$

In Figure 6, the white bars represent the rates of the processed of the incoming data units with the pipelined executions, and the black bars represent the rates of the processed of the incoming data units with the serial executions. The x-axis indicates the numbers of tweets packed into one data unit, and the y-axis indicates the rates of the processed of the incoming data units in percentile. The higher is better. Ideally, each tweet should be located to one computational node, however, this scenario was not effective even with all the available 52 nodes in the experimental environment this time; both the pipelined version and the serial version lost more than 80% of the incoming data. Therefore, we decided to group several tweets into a data unit, and let each computational node process each data unit. Figure 6 indicates that the pipelined version dropped about 60% of the incoming data with six tweets packed into one data unit. However, the pipelined version processed all the incoming data when one data unit contains eight or more tweets. The drop rate decreases if you pack more tweets into one data unit, because you have longer time slot for the larger data unit, and you have more relaxed deadline for the dependencies discussed in Section 3.

On the other hand, the serial execution always drops more than 80% of the incoming data regardless of the number of tweets contained into a data unit. For example, we observed the

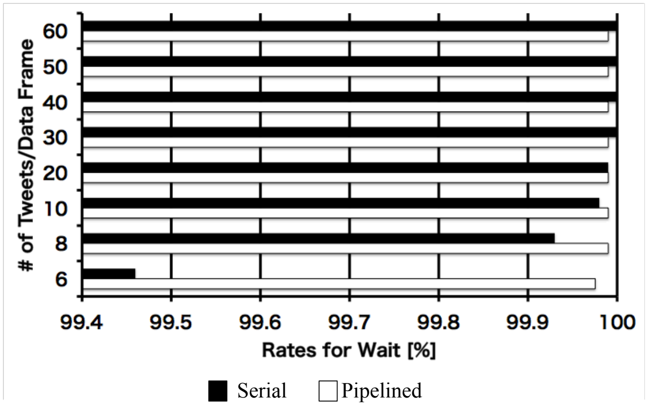


Fig. 7. The rates of waiting for the tasks for data dependencies.

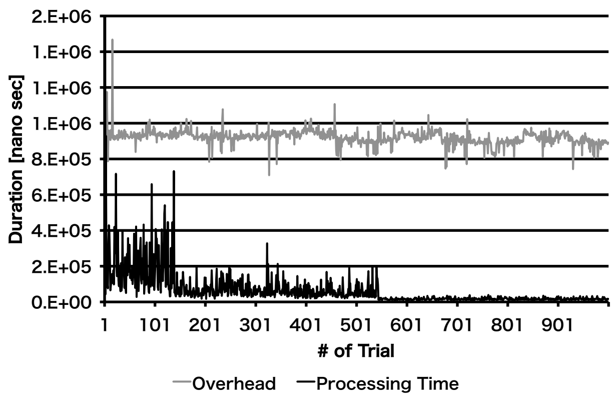


Fig. 8. Overheads of Java RMI calls and the actual time costs for the analysis processes on the incoming 1000 different tweets.

serial execution processed only less than 20% of the incoming data even with 60 tweets packed into one data unit. This situation is equivalent to the batch process of all the tweets every second, and the result indicates that the batch process invoked every second is not practical at all for the purpose of the realtime analysis of the actual fast data stream such as Twitter. The series of the experiments concludes that the pipelined execution is indespensable and effective for on-the-fly analysis of the realistic fast data streams.

Figure 7 plots the rates of the waiting in the pipelined execution with the resource manager. That is, the values in Figure 7 represent how many processes had to wait for another process to read from the sketch. The more idle tasks occupy more CPU resources as the rate for the wait increases. The x-axis represents the rates of the waiting and the y-axis represents the number of the tweets packed into one data unit. The black bars indicate the rates of the wait for the serial executions, and the white bars indicate the rates of the wait for the pipelined executions. Apparently, the pipelined tasks face the wait state more often than the serial tasks, which is natural because the pipelined version drops the incoming data less and tries to process all the incoming data. Figure 7, however, reveals that almost all the tasks had to wait for another task because of the data dependencies. We can assume that more tasks might be processed with less number of computational nodes once this situation is resolved and, therefore, we are planning to investigate this problem further in the future work.

We also recognize the experimental results shown here are possible to change according to the balance between the speed of the incoming data stream and the time cost for the analysis process on each tweet. Figure 8 shows the overheads of Java RMI calls and the actual times costs for the analysis processes on the incoming 1000 different tweets as a reference in order to give an overview of the balance of the time costs in the experiments in this section. In Figure 8, the gray line indicates the overheads of Java RMI calls, and the black line indicates the actual time costs for the analysis processes. The x-axis represents the number of the trials, and the y-axis represents the time costs in nano seconds. We also implemented a simulator to simulate the behavior of the resource manager in the distributed computational environment with the various settings of the speed of the incoming data stream, the time cost of the analysis process on each incoming data unit, and the overheads of the network communications and remote procedure calls. For the limitation of the space, we leave out the detailed results of the simulations and the further disucssion on the results here. We just add the simulation results showed quite similar tendency to the results already discussed in this section.

## V. Related Work

The high performance computing area has been energetically developed serveral speed-up techniques targeting the applications with the huge input data [26]–[31]. Such an application is called the data intensive application. However, these techniques are not simply applicable for the stream mining because of the data access patterns. A data intensive application accesses data in a write-once-read-many manner, and, therefore, the speed-up techniques for a data intensive application is to mainly for the purpose of the speed-up of the access to the well utilized data. On the other hand, a stream mining usually refer to the input data only once and, therefore, the speed-up techniques for the data intensive applications do not work well.

There are several researches on the speed-ups of the stream mining algorithms through static resource management [32]–[34]. Those resource managers request a solid estimation for each stage of the computation, and also put an assumption that the input data stream rate is consistent. As the actual data stream is not consistent and the estimation of the computational cost in advance is practically impossible in the actual applications and, therefore, a dynamic resource management just as proposed in this paper needs to be developed.

## VI. Conclusion

This paper proposed a dynamic resource management for the perfect analysis of the fast and realistic data streams such as Twitter, focusing on the discrepancy between data access time and CPU usage time. The proposed methodology was implemented as the resource manager in the acutual cloud environment, and validated its efficiency. The results demonstrated that our approach is indispensable to process the fast data stream without a drop; otherwise, 80-90% of the incoming data will be lost.

## ACKNOWLEDGMENTS

This paper is based in part on work supported by "Multimedia Web Analysis Framework towards Development of Social Analysis Software", funded by Japanese Science and Technology Agency (JST).

## REFERENCES

[1] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "Mining data streams: a review," *ACM SIGMOD Record*, vol. 34, no. 2, pp. 18 – 26, 2005.

[2] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *Proc. the 29th international conference on very large data bases (VLDB'03)*, 2003, pp. 81 – 92.

[3] ——, "A framework for projected clustering of high dimensional data streams," in *Proc. the 30th international conference on very large data bases (VLDB'04)*, 2004, pp. 852 – 863.

[4] B. Babcock, M. Datar, R. Motwani, and L. O'Callaghan, "Maintaining variance and k-medians over data stream windows," in *Proc. ACM SIGMOD/PODS 2003 Conference*, 2003, pp. 234 – 243.

[5] M. Charikar, L. O'Callaghan, and R. Panigrahy, "Better streaming algorithms for clustering problems," in *Proc. the 35th annual ACM symposium on Theory of computing (STOC'03)*, 2003, pp. 30 – 39.

[6] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proc. the sixth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'00)*, 2000, pp. 71 – 80.

[7] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan, "Clustering data streams: Theory and practice," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 3, pp. 515 – 528, 2003.

[8] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan, "Clustering data streams," in *Proc. the annual symposium on foundations of computer science*, 2000, pp. 359 – 366.

[9] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proc. the seventh ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'01)*, 2001, pp. 97 – 106.

[10] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani, "Streaming-data algorithms for high-quality clustering," in *Proc. the 18th international conference on data engineering (ICDE2002)*, 2002, pp. 685–694.

[11] C. Ordonez, "Clustering binary data streams with k-means," in *Proc. the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery (DMKD'03)*, 2003, pp. 12–19.

[12] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "On demand classification of data streams," in *Proc. the tenth ACM SIGKDD international conference on knowledge discovery and data mining (KDD'04)*, 2004, pp. 503 – 508.

[13] Q. Ding, Q. Ding, and W. Perrizo, "Decision tree classification of spatial data streams using peano count trees," in *Proc. the 2002 ACM symposium on Applied computing (SAC'02)*, 2002, pp. 413–417.

[14] V. Ganti, J. Gehrke, and G. Ramakrishnan, "Mining data streams under block evolution," *ACM SIGKDD Explorations Newsletter*, vol. 3, no. 2, pp. 1 – 10, 2002.

[15] S. Papadimitriou, A. Brockwell, and C. Faloutsos, "Adaptive, hands-off stream mining," in *Proc. the 29th international conference on very large data bases (VLDB'03)*, 2003, pp. 560 – 571.

[16] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proc. the ninth ACM SIGKDD international conference on knowledge discovery and data mining (KDD'03)*, 2003, pp. 226 – 235.

[17] G. Cormode and S. Muthukrishnan, "What's hot and what's not: tracking most frequent items dynamically," *ACM Transactions on Database Systems (TODS) - Special Issue: SIGMOD/PODS 2003*, vol. 30, no. 1, pp. 249 – 278, 2005.

[18] G. S. Manku and R. Motwani, "Approximate frequency counts over data streams," in *Proc. the 28th international conference on very large data bases (VLDB'02)*, 2002, pp. 346 – 357.

[19] Y. Chen, G. Dong, J. Han, B. W. Wah, and J. Wang, "Multi-dimensional regression analysis of time-series data streams," in *Proc. the 28th international conference on very large data bases (VLDB'02)*, 2002, pp. 323 – 334.

[20] V. Guralnik and J. Srivastava, "Event detection from time series data," in *Proc. the fifth ACM SIGKDD international conference on knowledge discovery and data mining (KDD'99)*, 1999, pp. 33 – 42.

[21] J. Himberg, K. Korpiaho, H. Mannila, J. Tikanmaki, and H. Toivonen, "Time series segmentation for context recognition in mobile devices," in *Proc. the 2001 IEEE International Conference on Data Mining (ICDM'01)*, 2001, pp. 203 – 210.

[22] P. Indyk, N. Koudas, and S. Muthukrishnan, "Identifying representative trends in massive time series data sets using sketches," in *Proc. the 26th International Conference on Very Large Data Bases (VLDB'00)*, 2000, pp. 363 – 372.

[23] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *Proc. the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery (DMKD'03)*, 2003, pp. 2 – 11.

[24] Y. Zhu and D. Shasha, "Statstream: statistical monitoring of thousands of data streams in real time," in *Proc. the 28th international conference on very large data bases (VLDB'02)*, 2002, pp. 358 – 369.

[25] S. Akioka, Y. Muraoka, and H. Yamana, "Data access pattern analysis on streaming mining algorithms for cloud computation," in *Proc. the 2010 International Conference on Parallel and Distributed Processing (PDPTA2011)*, 2011, pp. 36 – 42.

[26] I. Raicu, I. T. Foster, Y. Zhao, P. Little, C. M. Moretti, A. Chaudhary, and D. Thain, "The quest for scalable support of data-intensive workloads in distributed systems," in *Proc. the 18th ACM international symposium on High performance distributed computing*, 2009, pp. 207 – 216.

[27] S. Doraimani and A. Iamnitchi, "File grouping for scientific data management: Lessons from experimenting with real traces," in *Proc. the 17th International Symposium on High Performance Distributed Computing (HPDC'08)*, 2008, pp. 153 – 164.

[28] I. Raicu, Y. Zhao, I. T. Foster, and A. Szalay, "Accelerating large-scale data exploration through data diffusion," in *Proc. the 2008 International Workshop on Data-aware distributed computing (DADC'08)*, 2008, pp. 9 – 18.

[29] S. Y. Ko, R. Morales, and I. Gupta, "New worker-centric scheduling strategies for data-intensive grid applications," in *Proc. the 8th ACM/IFIP/USENIX international conference on Middleware (MIDDLEWARE2007)*, 2007, pp. 121 – 142.

[30] S. Venugopal and R. Buyya, "A set coverage-based mapping heuristic for scheduling distributed data-intensive applications on global grids," in *Proc. the 7th IEEE/ACM International Conference on Grid Computing (GRID'06)*, 2006, pp. 238 – 245.

[31] A. Ramakrishnan, G. Singh, H. Zhao, E. Deelman, R. Sakellariou, K. Vahi, K. Blackburn, D. Meyers, and M. Samidi, "Scheduling data-intensive workflows onto storage-constrained distributed resources," in *Proc. the Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid'07)*, 2007, pp. 14 – 17.

[32] B. Agarwalla, N. Ahmed, D. Hilley, and U. Ramach, "Streamline: a scheduling heuristic for streaming applications on the grid," in *Proc. the 13th Annual Multimedia Computing and Networking Conference (MMCN'06)*, 2006, pp. 77 – 91.

[33] W. Zhang, J. Cao, Y. Zhong, L. Liu, and C. Wu, "An integrated resource management and scheduling system for grid data streaming applications," in *Proc. the 2008 9th IEEE/ACM International Conference on Grid Computing (GRID'08)*, 2008, pp. 258 – 265.

[34] L. Chen and G. Agrawal, "A static resource allocation framework for grid-based streaming applications: Research articles," *Journal Concurrency and Computation: Practice & Experience - Middleware for Grid Computing*, vol. 18, no. 6, pp. 653 – 666, 2006.

# Relevance of Quality Criteria According to the Type of Information 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

***Abstract* — The assessment of Data Quality varies according to the information systems, quality properties, quality priorities, and user experience among others. This paper presents a number of user stereotypes and a study of the relevance of the quality properties from experienced users mainly at the industry. A Data Quality Manager prototype has been extended to suggest such quality properties and their corresponding priorities. The relevance of quality criteria according to the type of Information Systems is presented and validated.**

***Keywords-data quality; assessment; stereotypes.***

## I. Introduction

The analysis of data quality requires a number of indicators as a reference for the assessment of data quality. However, this is not an easy task for naive users with not enough experience. Data consumers of a Decision Support System (DSS) might prefer some data against other because of reputation of data producers, the credibility and relevance of data for the task at a hand, or the level of satisfaction they have on making strategic decisions effectively from using reliable data. Furthermore, data consumers of operational systems might be more interested in timeliness, response time, and accessibility of data for an effective On-Line Transaction Processing (OLTP).

Previous work has shown that the overall assessment of data quality depends on the quality properties chosen as quality indicators, and the priority of each quality property might change the final quality score. Refer to [4] for further information.

We have developed a Data Quality Manager (DQM) [1], [2], [3], [4] in order to assess data quality within heterogeneous databases. However, the DQM still required the specification of which quality properties and the priority of those quality properties in order to provide a global assessment [4]. Therefore, the assessment result depends on the user experience.

The purpose of the present research was to identify a set of relevant quality properties according to the type of Information Systems (IS) and their corresponding weights that shall be established for computing the global assessment of data sources. Therefore, we have suggested a number of quality properties according to the information systems and the type of user in [5]. We have also identified the data quality properties interdependencies within the data quality assessment. Therefore, we have decided to conduct a survey to validate or analyze such proposals, and to identify a general estimation of the priorities (weights) that should be considered within a data quality assessment.

As user experience is substantial within the data quality assessment, a survey was applied to OLTP and OLAP specialists on the web.

The identification of a set of user stereotypes with their corresponding weights for the assessment of data quality according to the type of Information Systems is a novelty and the contribution of the present paper.

The following section presents a number of data quality interdependencies identified for the assessment of quality indicators in [5]. The third section analyses the data quality stereotypes to be implemented within a Data Quality Manager. The fourth section presents a survey that was conducted to provide a ranking of quality properties according to the type of Information Systems from experienced users. Such ranking has been implemented as weights during the assessment of data quality within the DQM and presented in the fifth section. The last section concludes with main achievements and future work.

## II. Data Quality Interdependencies

From the Data Quality Reference Model presented in [3], we have identified a number of criteria whose measurement does not depend on other quality criteria as *Primary Quality Criteria*. Here we present very briefly the following data quality property definitions.

*Accuracy* is the measure of the degree of agreement between a data value o collection of data values and source agreed to be correct in [9].

*Timeliness* Is the extent to which the age of data is appropriate for the task at hand [6], and is computed in terms of currency and volatility.

*Completeness* is the extent to which data is not missing [11], [12], it is divided by two quality dimensions coverage, and density in [10].

*Currency* Time interval between the latest update of a data value and the time it is used [14].

The measurement of a quality criterion might be part of the measurement of an aggregate one. The quality dimensions, whose measurements derive from primary criteria, are identified as *secondary quality properties*. However, we have not established or tested any kind of correlation among them. We have identified some relationships between these quality properties based on their definitions from previous research as has been referenced on each quality property definition.

*The interpretability* dimension is the extent to which data are in appropriate language and units, and the data definitions are clear [15]. Thus, it depends on several factors: If there is any change on user needs, its representation should not be affected, this can be possible with a flexible format; The data value shall be presented consistently through the application and that the format is sufficient to represent what is needed and in the proper manner.

*Reputation* is the extent to which data are trusted or highly regarded in terms of their source or content [11]. Three factors shall be considered at measuring time: reputation of data should be determined by its overall quality. If authors of data provide inaccurate data then they are unreliable and their reputation shall be therefore decreased. Commonly reputation might be increased if authors have enough experience gained across the time. If data owners produce accurate data consistently, modify data as soon as possible when mistakes are found, and they in turn recommend authors of quality data.

*Accessibility* is the extent to which data is accessible in terms of security [15], availability and cost.

Data might be available but inaccessible for security purposes, or data might be available but expensive.

Data is *credible* as true [11] if it is correct, complete, and consistent.

*Usability* is the extent to which data are used for the task at a hand with acceptable effort. In other words, users prefer data that is useful and ease to use.

*Usefulness* is the degree where using data provides benefit on the performance on the job. In other words, the extent to which users believe data is correct, relevant, complete, timely, and provides added value.

*Easy to use* is the degree of effort user needs to apply to use data [12], because as less effort is easier to use. This effort is in terms of understand ability and interpretability as resources needed to achieve the expected goals. However, it is common that users use determined data sources, due to the reputation of authors.

The measurement of usability allow user to decide on the acceptance of data, and select a specific datum, data or data source among other alternatives.

Data is *reliable* if it is considered as unbiased, good reputation [14] and credible [6].

The *added value* is stated in terms of how easy is to get the task complete named as effectiveness; how long could the task take known as efficiency; and the personal satisfaction obtained from using data.

Fig. 1 presents the data quality interdependencies. These dependencies can help the measurement of such quality properties. There are some interdependencies very straight forward to compute. For instance, in order to compute timeliness, currency and volatility are required to be estimated and fused with an aggregation function as presented in Table 1.

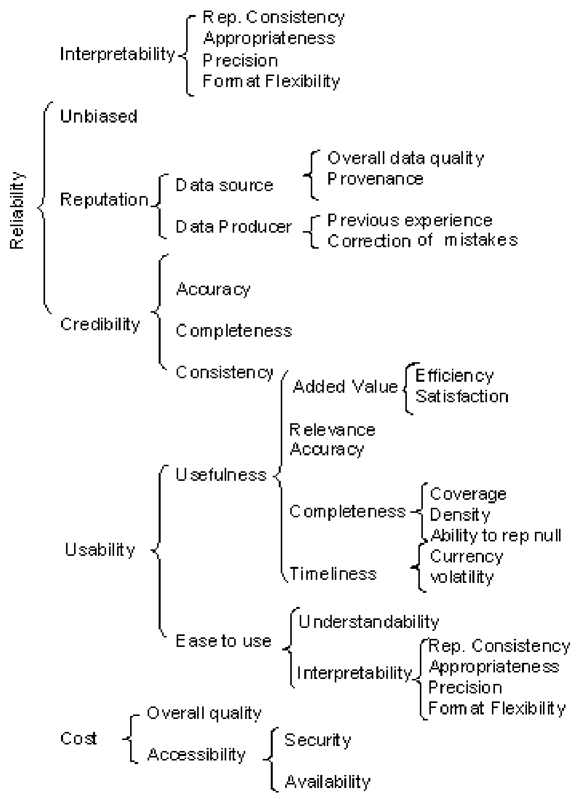


Figure 1. Data Quality Interdependencies

TABLE 1 EXAMPLE OF DATA QUALITY PROPERTIES

| **Currency** | **Volatility** | **Timeliness** |
|---|---|---|
| Cu(t)=Time Request – last update time | Vo(t)= Update frequency | T(t)= max(0,1-Cu(t)/Vo(t)) |

The present research has been focused mainly in quantitative data quality properties. For further information, refer to [4][5], where Measurement and Assessment model are detailed.

## III. DATA QUALITY PROPERTIES ACCORDING TO IS

The identification and ranking of relevance for data quality properties according to the type of users and Information Systems is not straightforward. For instance, if we consider volatility as the update frequency the relevance of such quality property varies very remarkable according to the application domain, volatility is essential within operational systems, but not quite important within DSS where historical information is materialized.

An Executive Support System (ESS) is designed to help a senior management tackle and address issues and long-term trends to make strategic decisions for the business. It gathers analyses and summarizes aggregate, internal and external data to generate projections and responses to queries. Therefore, the main data quality problem on ESS relays on external data, so decisions depend on accuracy, timeliness, completeness and currency of the external data collected. Furthermore, users are interested in those quality properties that are very much related to their work role.

According to Lee and Strong [9], the responses from data collector, data custodian, and data consumer within the data production process determine data quality because of their knowledge.

Data consumers require friendly and usable tools in order to deal with making decisions only rather than the IS per se. Possible inconsistencies might be derived from different data sources so making decisions regarding which external data source to trust is an issue. Response time however, is not of great relevance when the analysis is on long-term trends.

### *A. Data Collector in DSS*

Within a Decision Support System, there are people, groups or even systems that generate, gather or save data to the information systems. Therefore, the role of data collector impacts on accuracy, completeness, currency and timeliness of data.

The quality properties identified as the most relevant within Decision Support Systems for data collectors are presented in Fig. 2. Therefore, accuracy, completeness and timeliness shall be presented to the collector user in order to help during the assessment of data quality. Furthermore, completeness is estimated by an aggregation function of coverage, density and ability to represent nulls. Same applies for the rest of the user stereotypes

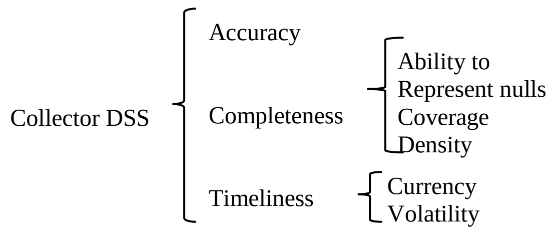


Figure 2. Quality properties for collectors within DSS

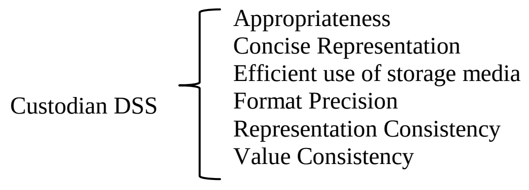


Figure 3. Quality properties for custodians within DSS

### *B. Data Custodian in DSS*

Data custodians are people who manage computing resources for storing and processing data. In the case of DSS, the process of extraction, transformation and load (ETL) of data within a data warehouse is mainly related to data custodians. The ETL process is a key data quality factor; it may degrade or increase the level of quality. Therefore, custodians determine the representation of data, value consistency, format precision, appropriateness of data for the task at a hand, the efficient use of storage media. Refer to Fig. 3 for the relevant quality properties among data custodians within Decision Support Systems. In other words, appropriateness, concise representation, efficient use of storage media, format precision, representation consistency and value consistency shall be evaluated and presented to them in order to help them decide which data source should be utilized.

### *C. Data Consumer in DSS*

Data consumers are involved in retrieval of data, additional data aggregation and integration. Therefore, they impact on accuracy, amount of data relevant for the task at a hand, usability, accessibility, reliability and cost of information in order to make decisions. An analysis on data quality properties in Data warehouses is presented in [8]; such quality properties are included in this work. Accuracy, amount of data, usability, accessibility, reliability and cost shall be considered during data quality assessment. Such quality properties are shown in Fig. 4.

### *D. Data Consumer OLTP*

As data consumers are involved in retrieval of data the quality properties usability, accessibility, believability, reputation of data sources are key factors for their job. Response time and timeliness [14] are essential within OLTP systems. From the data consumer perspective accessibility [15] and cost are also very important. The corresponding quality properties relevant to this role are shown in Fig. 5.

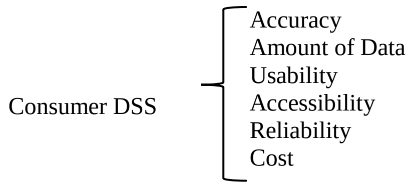


Figure 4. Quality Properties for data consumer within OLTP

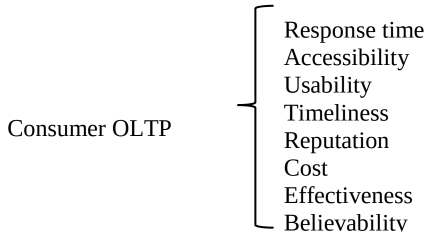


Figure 5. Quality properties for consumers within OLTP systems

### *E. Data Custodian in OLTP*

In transactional systems, data custodians are much related to accuracy, consistency at data value level, completeness [9], timeliness [13], and uniqueness. Therefore the set of quality properties they are interested on for analysis of data quality from their perspective are shown in Fig. 6.

### F. Data Collector in OLTP

As data collectors within OLTP systems are people who generate information, this role impacts on accuracy, completeness, currency, uniqueness, value consistency and volatility of data. Fig. 7 presents such relevant quality properties for collectors within OLTP systems.

Custodian OLTP
- Accuracy
- Value consistency
- Completeness
- Timeliness
- Uniqueness

Figure 6. Quality properties for custodian within OLTP systems

Collector OLTP
- Accuracy
- Completeness
- Currency
- Uniqueness
- Value consistency
- Volatility

Figure 7. Quality properties for collectors within OLTP systems

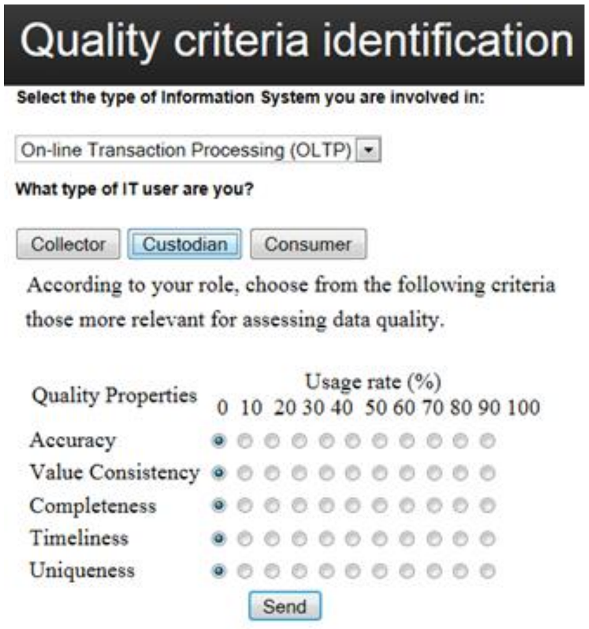
A number of quality properties have been identified according to the type of users and shown in the past 6 figures. However, there is no an assessment method to be considered to provide an overall data quality score. Therefore, we require identifying the priorities of such quality properties that is the aim to be achieved in the following section.

## IV. Identifying Data Quality Priorities According to Expert Users

### A. Design of Questionnaire

We have conducted an on-line survey requesting an order of importance among the quality properties according to their corresponding experience within a specific Information System.

The questionnaire requires the type of information system and what role do users play. According to these two characteristics, the questionnaire presents a set of quality properties and a percentage of relevance these properties should be assigned during quality assessment. The questionnaire was designed to be briefly answered. For instance, we present in Fig. 8 an example of the one developed to find out the relevance of quality properties for custodian users within OLTP systems.



Quality criteria identification

Select the type of Information System you are involved in:

On-line Transaction Processing (OLTP)

What type of IT user are you?

Collector [✓ Custodian] Consumer

According to your role, choose from the following criteria those more relevant for assessing data quality.

| Quality Properties | 0 | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Accuracy | ● | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| Value Consistency | ● | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| Completeness | ● | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| Timeliness | ● | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| Uniqueness | ● | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |

Usage rate (%)

Figure 8 Questionnaire for custodian users within OLTP systems

In order to obtain unbiased results, we have invited a number of specialists in operational and DSS information systems around the world. The following groups were invited to participate within the survey: the University Network of Contribution in Software Engineering and Databases, the Professionals of Business Intelligence Group, the Very Large Database Group, the Data Quality Pro Group, and the Information Technology and Communications Group.

The on-line survey was opened for six months in order to allow experts the specification of those quality priorities within the analysis of data quality.

### B. Results of Experiments

Quality is a very subjective conception, depending on user experience, information system, and business sector among others. Therefore we have decided to collect opinions and to identify the most common ranking of such quality properties during data quality assessment. The results of the on-line survey were analyzed and are presented in this section. There were collected 136 responses.

Concerning Decision Support Systems there were 82 responses, 22 from user collectors, 33 from data custodians, and 27 from DSS consumers.

In the case of data collectors, they take into consideration accuracy and completeness followed by currency and timeliness. Data custodians prefer to consider as the first option the value consistency of data followed by an efficient use of storage media, the appropriateness of data, concise representation and format precision. Data consumers on the other hand relay their decisions on sufficient amount of usable and accurate data. Refer to Fig. 9 for the data quality prioritization within DSS.

Regarding operational systems there were 54 responses, 19 from user collectors, 13 from data custodians, and 22 from data consumers.

Data collectors trust the most on accurate, complete and non duplicated data, followed by current and consistent information. Furthermore, data custodians also prefer accurate, unique and consistent rather than timely data. However, data consumers require fast response time, accessible, timely and usable data. Refer to Fig. 10 for the corresponding data quality prioritization.

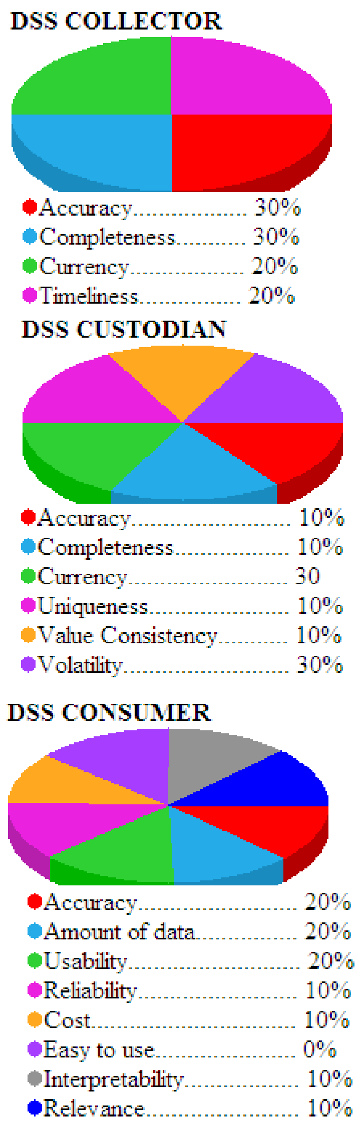


Figure 9. Data Quality prioritization within DSS

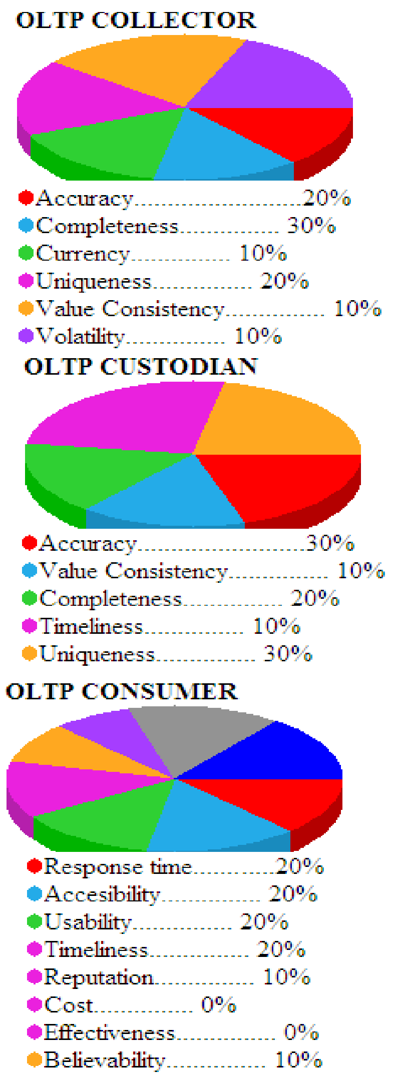


Figure 10. Data Quality prioritization within OLTP systems

The final percentages are obtained through the ranking of the quality properties according to the responses collected. However, the present research is looking forward to have more responses in the future by incorporating more specialists groups that allow being more precise with the outcomes and also to test the effectiveness of the stereotypes presented.

## V. DATA QUALITY MANAGER IMPROVEMENT

We have developed a Data Quality Manager as a prototype for the assessment of data quality within heterogeneous databases in [1], [2], [3], [4].

An improvement of such prototype consisted in the implementation of the data quality stereotypes to be suggested to inexperienced users to assist them with the analysis of a number of data sources to identify and query the best ranked data sources and make informed decisions.

The stereotypes implemented are the result of the experiments conducted through the analysis of the results obtained from the online survey and briefly explained in the previous section.

### *A. Suggestion of priorities for quality priorities according to the information systems*

This section presents very briefly the improvement of the DQM prototype for the assessment of data quality by suggesting a set of quality properties and their priorities to naive users. In the case of experienced users they still allowed to indicate explicitly their preferences.

For instance, Fig. 11 shows the DQM main menu and the selection of data quality assessment within Online Transaction Processing System conducted by inexpert custodian user.

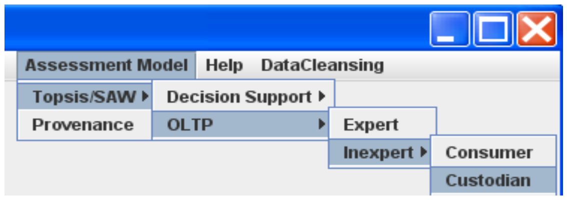
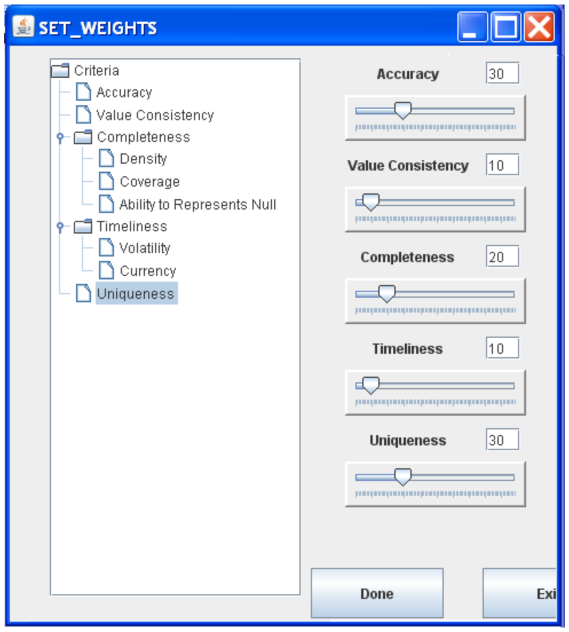


Figure 11. DQM main menu for selecting IS and type of user

The DQM prototype presents in Fig. 12 the most relevant quality properties within a DSS and their corresponding percentages. For instance, accuracy and uniqueness are the most relevant quality properties with 30%, then completeness with 20%, and Timeliness, Uniqueness with 10%.

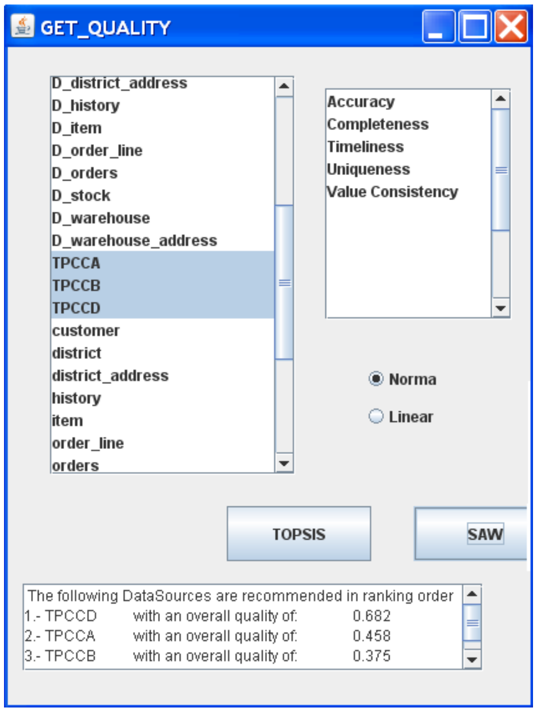


Accuracy 30
Value Consistency 10
Completeness 20
Timeliness 10
Uniqueness 30

Figure 12. DSS most relevant quality properties

### *B. Assessment of Data Quality*

Fig. 13 shows the assessment of data quality properties of three data sources obtained from the TPCC benchmark [16] named TPCCA, TPCCB and TPCCD, where TPCCD contains the best overall data quality.



(● Norma) ○ Linear

```
The following DataSources are recommended in ranking order
1.- TPCCD      with an overall quality of:      0.682
2.- TPCCA      with an overall quality of:      0.458
3.- TPCCB      with an overall quality of:      0.375
```

Figure 13. Ranking of data sources according to their quality

## VI. CONCLUSIONS AND FUTURE WORK

This document has presented how users might prefer some quality properties during the assessment of data quality according to their role within specific information systems.

We have previously identified data quality interdependency in [5] and now by an on-line survey, the prioritization of such quality properties to the Information Systems.

A number of user stereotypes have been suggested by a Data Quality Manager prototype that is meant to help naive users within the data quality analysis.

The DQM allows the identification of which quality criteria shall be used based on the application domain and the type of users. Furthermore, the user stereotypes presented correspond to data consumer, data collectors, and data custodians. However, more information from specialists is required in order to corroborate the prioritization and testing of the effectiveness of the stereotypes identified is part of future work.

## REFERENCES

[1] P. Angeles and L.M. MacKinnon, “Detection and Resolution of Data Inconsistencies, and Data Integration using Data

Quality Criteria", Proceedings of QUATIC 2004: Conference for Quality in Information and Communications Technology, Instituto Portugues da Qualidade, ed., pp. 87-94., ISBN 972-763-069-3, Porto, Portugal, 2004.
[2] P. Angeles and L.M. MacKinnon, "Tracking Data Provenance with a Shared Metadata", Proceedings of PREP 2005: Postgraduate Research Conference in Electronics, Photonics, Communications and Networks, and Computing Science, pp. 120-121, Lancaster England, U.K., 2005.
[3] P. Angeles and L.M. MacKinnon, "Quality Measurement and Assessment Models Including Data Provenance to Grade Data Sources", International Conference on Computer Science and Information Systems" at the Athens Institute for Education and Research, pp. 101-118, Athens, Greece, 2005.
[4] P. Angeles and L.M. MacKinnon, "Management of Data Quality when Integrating Data with Known Provenance", Herriot-Watt University, pp. 1-199 Edinburgh, UK, April 2007.
[5] M.P. Angeles and F. Garcia-Ugalde, "User Stereotypes for the Analysis of Data Quality According to the Type of Information Systems", International Association for Scientific Knowledge, E-Activity and Leading Technologies, pp. 207-212, Madrid Spain, December 2008.
[6] Ballou, D.P., Wang, R.Y., Pazer, H. and Tayi, G.K. "Modeling information manufacturing systems to determine information product quality". Management Science 44, 4 April,1998, 462–484.
[7] C.Cappiello, C.Francalanci, B.Pernici, P.Plebani, and M.Scannapieco, "Data Quality Assurance in Cooperative Information Systems: a Multi-dimension Quality Certificate". Proceedings of the ICDT 2003 International Workshop "Data Quality in Cooperative Information Systems" (DQCIS 2003), pp. 64 -70 Siena, Italy, 2003
[8] M. Jarke, M.A. Jeusfeld, C. Quix, and P.Vassiliadis "Architecture and Quality in Data Warehouses an extended Repository Approach", Journal on Information Systems, Vol. 24", no.3, pp. 229-253, 1999, URL "citeseer.ist.psu.edu/jarke99architecture.html", retrieved: December, 2011.)
[9] Lee Y. and Strong D. "Knowing-Why about Data Processes and Data Quality", Journal of Management Information Systems, Vol. 20, No. 3, pp. 13 – 39. 2004.
[10] F. Naumann, J. Freytag, and U. Lesser, "Completeness of Information Sources", Workshop on Data Quality in Cooperative Information Systems (DQCIS2003), pp. 583-615, Cambridge, Mass., 2003.
[11] L. Pipino, W.L. Yang, and R. Wang, "Data Quality Assessment", Communications of the ACM, Vol. 44 no. 4e, pp.211-218, 2002.
[12] Redman "Data Quality for the Information Age", Boston, MA., London : Bartech House, 1996.
[13] D.M. Strong, W.L. Yang, and R.Y. Wang, "Data Quality in Context", Communications of the ACM, vol. 40, no. 5, pp. 103-110, 1997.
[14] R.Y. Wang, M.P. Reedy, and A. Gupta, "An Object-Oriented Implementation of Quality Data Products". Workshop on Information Technology Systems, O.1993.
[15] Wang R. Y. and Strong D.M. "Beyond accuracy: What data quality means to Data Consumers", Journal of Management of Information Systems, vol. 12, no 4 1996, pp. 5 -33.
[16] TPCH, TPC Benchmark ™ H, Standard Specification Revision 2.3.0 Transaction Processing Performance Council www.tpc.org.info 2006 (retrieved: December, 2011.)

# Cloud-based Medical Image Collection Database with Automated Annotation

Anthony Maeder and Birgit Planitz
School of Computing and Mathematics
University of Western Sydney
Sydney, Australia
a.maeder@uws.edu.au, birgit@planitz.net

*Abstract* - **Typical medical image annotation systems use manual annotation or complex proprietary software such as computer-assisted-diagnosis. A more objective approach is required to achieve generalised Content Based Image Retrieval (CBIR) functionality. The Automated Medical Image Collection Annotation (AMICA) toolkit described here addresses this need. A range of content analysis functions are provided to tag images and image regions. The user uploads a DICOM file to an online portal and the software finds and displays images that have similar characteristics. AMICA has been developed to run in the Microsoft cloud environment using the Windows Azure platform, to cater for the storage requirements of typical large medical image databases.**

***Keywords - medical imaging; content based image retrieval; cloud computing***

## I. INTRODUCTION

Large medical image collections provide essential source datasets for research on population health or disease cohort studies, using patient phenotype information derived from the images. For example, we may wish to select cases where brain morphology of a certain kind is conjectured to be related to deposition of plaques linked with the onset of dementia, and finding regions in those images where there is a higher expectation of plaque formation. Presently, *annotation of medical images* for such work is conducted either manually by highly specialist expert viewers or radiologists [1], or by complex proprietary software such as computer-assisted-diagnosis. This is a time consuming process, subject to error and bias. Selection of images according to established Content Based Image Retrieval (CBIR) criteria such as “similar to a given image” or “containing range of characteristics” would be far more objective if it could be conducted automatically. To achieve this for medical images requires a set of functions which cater for the typical components of image similarity, and can be tuned to suit different medical image types.

The *Automated Medical Image Collection Annotation (AMICA)* toolkit for implementing the above CBIR criteria has been developed to address this need. It is intended to contribute to advances in clinical research by permitting wider use of medical images than is currently practiced, such as:

*a) Population health and epidemiological studies* that rely on content analysis of images and related patient information will be much easier to conduct via an electronic medium than by using expert human readers, and availability of compatible computer processible image data widely from various sources will allow a much fuller analysis which truly covers most of the population;

*b) Cohort studies* for cases with particular physiological or phenotypical profiles will be able to source and include enough cases to provide high statistical power, allowing more individualised risk factors to be assessed and thus allowing screening and staging processes to be optimised. Cases will also be selectable on a wider basis than is achievable now from patient information in electronic records, by use of image content analysis;

*c) Education and training/credentialing* of radiographers, radiologists and other clinicians who are involved with image interpretation will be more effective because it will be possible to select instances of images which demonstrate particular visual aspects, or correspond to types of cases where reading performance improvement is desirable for that individual.

The AMICA software is appropriate for the above situations because both the medical image database and automated annotation tools are stored in a cloud environment. That is, we make use of the flexibility and scalability of the cloud to grow our application to suit researchers’ needs. We envisage that our application will grow beyond its initial pilot implementation to a wide ranging application suitable for a range of modalities and anatomical regions.

This paper provides a brief review of existing medical image collections and annotation tools, and describes how AMICA differs from other software. We discuss how Microsoft technologies have been used to deploy our web application in the cloud. We also detail the CBIR functions used for image annotation; our CBIR algorithm works on the principle of “find other images like my given image”. We conclude with a discussion on leveraging the cloud environment to expand our pilot implementation of AMICA.

## II. RELATED RESEARCH

Many widely used digital medical image collections have previously been established but these are generally used as raw data source only, without related toolsets. Providing associated functionality to allow specific types of operations to be performed on these images has proved beneficial in some cases (e.g., brain image registration [2]; brain atlases [3]). However, toolset development for image analysis functions on medical images has tended to be ad hoc, with Open Source options proliferating.

Several major organisations, particularly in the U.S., such as the National Cancer Institute, have made medical images databases and associated search tools available to the wider research community. A popular project is the Visible Human, which is a database of transverse CT, MR and cryosection images of three dimensional (3D) anatomical representations of male and female cadavers [4]. The National Institute of Health has made these datasets available for study, and also developed ITK, an open source toolkit for image registration and segmentation [5]. The Biomedical Informatics Research Network (BIRN) has made a list of tools available for data storage and medical image processing [6], and has also developed a downloadable Human Imaging Database [7]. The National Cancer Institute developed Annotation Imaging Markup, a downloadable manual annotation tool that enables easy and automated image searching [8]. The Institute also produced the National Biomedical Imaging Archive, which includes a manual annotation option and a web portal for accessing medical images [9]. Other large medical image databases (e.g., the Singapore National Medical Image Resource Centre [10]) are searchable via keywords but not image content.

CBIR has been applied to medical image datasets, however this has generally been limited to specific cases. For example, tumour detection [11], retrieval of lung images [12], 3D MR images [13], 3D MR and CT images [14], or image shape for retrieving pathology [15]. In a project with broader scope, Principal Component Analysis (PCA) has been used for medical image classification [16]. In a system similar to ours, PCA was applied to find closest matches between features of candidate brain images and a set of test images. However, the system is a form-type application, rather than web-based, therefore limiting its capacity to a local setting. Also, results were limited to brain datasets. Mojsilovic and Gomes developed a web-based system for classifying numerous image datasets according to modality [17]. Their program searched large medical image databases and performed the classification automatically. The objective of the modality classification was to categorise images so that domain-specific CBIR functions could later be applied to datasets in each modality. The approach is flexible in that it encompasses existing online databases without users having to load their own. However, this system becomes superfluous in the case of DICOM data, where modality information is stored in the data file.

It would appear that existing medical image repositories and retrieval systems were typically designed with specific applications in mind. Alternatively, our AMICA software provides a simple but comprehensive toolset that could be established as a baseline. Our software takes an image and finds similar images according to some basic CBIR criteria. We believe that this approach, coupled with our leveraging of Microsoft's cloud environment makes for a flexible and scalable tool for a wide variety of medical images.

## III. Cloud-based Automated Annotation Tools

This project uses the Windows Azure Platform for data management by exploiting the cloud model. By hosting the AMICA software in the cloud, we tap into the potential of growing the application to use significant volumes of medical images. Individually, medical images tend to be of large sizes (e.g., >25MB for a mammogram), so a substantial collection quickly causes an explosion in server storage requirements. Our objective is to make our database scalable and to provide sufficient storage space for users to upload the datasets that they wish to annotate automatically. Using the cloud, we have the benefit of flexibility, because we can scale up our storage needs as required. We also run our web interface and worker role (for CBIR) in the cloud, where they are stored multiply to ensure that the web interface is constantly available.

Our software consists of three components:

1. Storage Table (SQL Azure)
2. CBIR (Azure Worker Role)
3. Website (Azure Web Role)

We have set up the application using .net and C# in the Visual Studio environment, which extend naturally when deploying an application to Azure Platform.

### *A. Storage Table*

Our first important design choice was whether to store medical image data within Azure's storage environment, or to use a SQL Azure table. We consulted the MSDN developer magazine, which advises [18]:

"If you have an application that requires data processing over large data sets, then SQL Azure is a good choice. If you have an app that stores and retrieves (scans/filters) large datasets but does not require data processing, then Windows Azure Table Storage is a superior choice."

Hence, we have used a SQL Azure table because our project involves significant image processing on large numbers of medical images.

We designed our data storage such that DICOM files and their associated image attributes, which are retrieved using CBIR, are stored as rows in a table, as shown in Table I.

Each entity (table entry) has a unique ID, which is generated automatically and consists of DICOM header elements.

We list Modality and Anatomical Region for each DICOM file explicitly. This reduces our initial search function, i.e., we assume that users wish to find images similar to their given image, which is of a specific body part that was captured using a specific modality. In terms of image processing, it also makes sense to segment the database in this manner, as different image processing functions apply to different image types.

TABLE I. MEDICAL IMAGE DATABASE TABLE

| Column Name | Data Type |
|---|---|
| ID | int |
| Modality | nvarchar |
| AnatomicalRegion | nvarchar |
| DicomInfo | text |
| DicomImage | image |
| ImageAttributes | text |

DICOM files are separated into information and image files. The aim of separating the files is also to speed up the search function. For example, a user may be studying dementia, and thus may want to only find brain images (that are like their given image) for patients of a specific age. In a future implementation, we will apply a pre-filter to scan DICOM information and retrieve relevant DICOM files, before performing more time-consuming image processing.

DICOM Images are stored as image data types, which are data types that hold any type of binary data. We read BLOBs (Binary Large Objects) in as streams and manipulate/display images according to the information (e.g., Bit Depth) extracted from the DICOM Info file.

The ImageAttributes text file is the file produced by the CBIR function. Image features and their values (e.g., foreground/background intensity threshold) are stored so that they can be compared to a given image in the retrieval process.

## *B. CBIR*

The CBIR toolset applies specific image analysis tasks on medical digital images, providing information on fundamental visual appearance characteristics of the images for metadata annotation purposes. The tools support two different types of annotations: (i) overall image characteristics and (ii) location and extent of specific features (i.e., regions of interest). The provision of these annotations allows matching to be performed between pairs of candidate images, to extract groups of images within a prescribed similarity envelope. For example, given a sample mammogram indicative of a breast with dense texture, other images with a comparable texture density can be extracted by comparison of the breast texture annotation values. We have not fully defined the range of ideal or conventional annotations as this aspect is work in progress.

Typically, in medical research, image subsets are extracted from image collections based on the appearance to human observers of image visual content characteristics such as statistical properties (e.g., textures) or structural features (e.g., regions of interest). For an automated annotation process, application of a sequence of software tools is required to allow users to undertake the following tasks on a given image collection:

(i) Define foreground (i.e., zone of relevance or region of interest) versus background (i.e., zone to ignore) image components;

(ii) Select and apply parameters for characterizing overall image properties (e.g., statistical texture/intensity profiles);

(iii) Select and apply parameters for characterizing region of interest features (e.g., statistical density/edge properties);

(iv) Select and display images having candidate annotations for properties/features within specified ranges;

(v) Iterate in repeated cycle of reapplying the above steps with parameter variations to refine the results.

A major advantage of our approach is that these types of annotation helps avoid the need to reapply image analysis computations each time the image collection is searched: it is analogous to pre-scanning a text data file for keywords and saving these as indexed tags for the file to be available to search tools. Allowing image subsets to be extracted rapidly and consistently using these annotations will improve speed and ease of use in research projects that rely on them.

The CBIR toolset is being developed as a Worker Role in Windows Azure. Images are dispatched to a queue for background processing [19]. The worker role takes images from the queue and processes them; it generates thumbnails for each image in our current pilot implementation.

## *C. Website*

The AMICA website has functions for:

(i) Uploading DICOM files (MR Brain and Mammograms in the current implementation, as described below); and

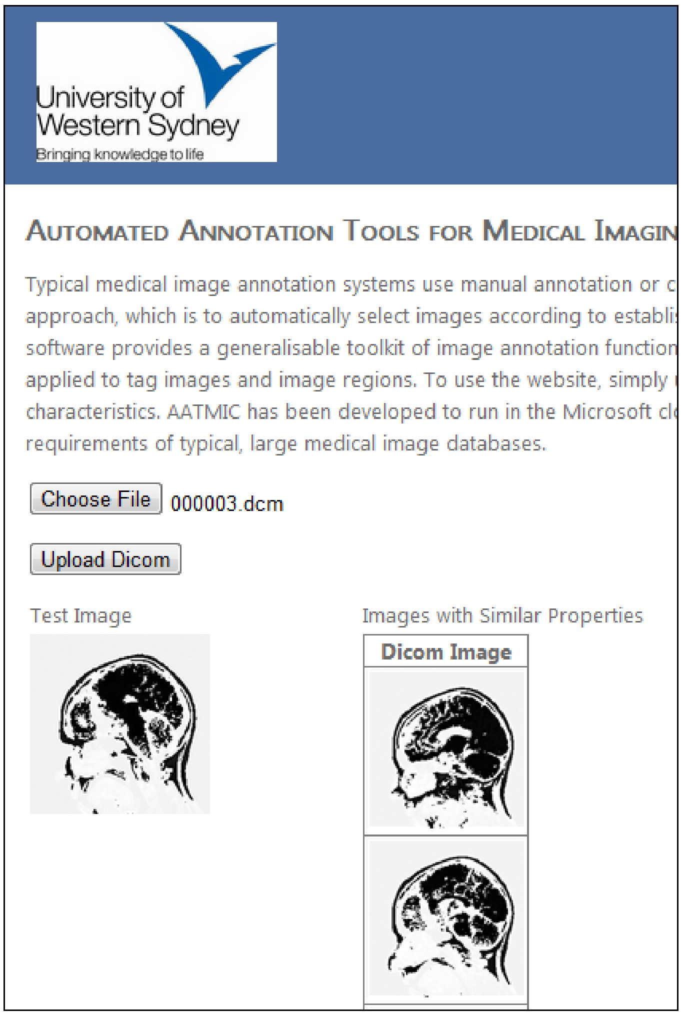
(ii) An output panel that displays (as thumbnails) the user's loaded image on the left and similar images, as found by CBIR, as on the right hand side (see Fig. 1).

We are tuning and testing our pilot implementation by using two distinct medical image types.

We use MR brain images as these constitute one of the most prolific forms of medical image data present across the health sector. Datasets, such as the Vasari collection are downloadable and are used to test image annotation algorithms, specifically, "to validate the use of medical images as predictive biomarkers for cancer diagnosis" [20]. These DICOM files are suitable for testing Cloud based web and worker applications because the images are small and thus enable efficient testing. We also test with a Matlab brain MR dataset for the same reason [21].

We are also testing and tuning with mammograms because large scale collections exist de facto in countries where there is a national breast cancer screening program. Mammogram image collections were first developed using scanned films in 1980s by MIAS, and subsequently USF, which has become the benchmark data set with several thousand images. Some proprietary collections exist for commercial research by CAD and imaging manufacturers (e.g., SECTRA, R2). The possibility exists that our AMICA software can be used for data management and annotation such that national coordination of breast screening images (e.g., as proposed in the UK e-Diamond project [22]) could ensue.

In Australia, the wide establishment and stimulation of eResearch infrastructure offers an opportunity contribute to emerging platforms and tools for basic data repository functions, and to enhance these by developing some more sophisticated ones suited to this application area. Most eResearch tools developed locally have been concerned with text or symbolic data annotation and analysis, so our software may contribute comparatively novel tools by addressing the digital image space.



AUTOMATED ANNOTATION TOOLS FOR MEDICAL IMAGIN

Typical medical image annotation systems use manual annotation or c
approach, which is to automatically select images according to establi
software provides a generalisable toolkit of image annotation function
applied to tag images and image regions. To use the website, simply
characteristics. AATMIC has been developed to run in the Microsoft cl
requirements of typical, large medical image databases.

Figure 1. Partial screenshot of AATMIC Website

Like our SQL table and CBIR worker role, the AMICA web interface also operates from Windows Azure. Although the storage requirements for the online portal are not overwhelming (we display thumbnails for all but individually selected images), we make use of Azure's triplicate storage of the website. That is, the redundancy ensures that the website is always up and running should one, or even two, servers be down [24].

## IV. DISCUSSION

Our short term goal is to complete the pilot implementation of the AMICA software. Then, we will extend the implementation as follows:

(1) The final pilot implementation will be tuned and tested using a greater number of Vasari MR brain datasets and our entire collection of mammograms (approx. 500 x 50MB image sets). Further testing will therefore cover a much wider range of different image content variations. These results will be used to evaluate computational efficiency of the tools and success of the automated annotations, as rated by comparison with human expert opinions.

(2) We will then extend our CBIR toolset to include a wider range of modalities and anatomical regions (e.g., CT brain, CT chest, CR chest).

(3) To cope with the predicted increase in data, we will provide a separate form for pre-filtering DICOM files. As mentioned earlier, before any image processing takes place, DICOM Information files could be pre-filtered so that only a relevant subset of images is returned for a given search. The pre-filtering function should be turned off if the user wishes to find test images in the entire database for a given Modality and Anatomical Region. We have not yet implemented this function, but see the need to do so as the medical image database grows and as searches become more specific (e.g., the dementia project mentioned earlier).

(4) In terms of CBIR, further benefits could be obtained by selection of similar cases to that in a given image, and selection of archetypical images for certain image features, using feature vectors and classification methods on certain combinations of annotation fields. This would contribute to fuller understanding of how such combinations occur, as well as providing suitable sample images for clinical education and training purposes.

## V. CONCLUSION

This paper has described an ongoing project in automated medical image annotation for management of image collections. The software that is currently under development addresses problems hampering existing systems, which either use manual annotation or complex proprietary software. Alternatively, we have designed the AMICA software to automatically select images according to established Content Based Image Retrieval (CBIR) criteria. Images are tagged and managed via a generalisable toolkit of image annotation functions. AMICA's website, storage and worker roles are being developed to run in the Microsoft cloud environment, to exploit the storage requirements of typical large medical image databases. Also, we ensure that the website is always accessible to users via Azure's triple redundancy storage system. The approach could be generalized to use other cloud environments, provided they offer similar functionality.

## ACKNOWLEDGMENT

The support of Microsoft Research in funding this project is gratefully acknowledged.

## REFERENCES

[1] C. E. Chronaki, X. Zabulis, and S. C. Orphanoudakis, "I2net medical image annotation service," *Informatics for Health and Social Care*, vol. 22, pp. 337-347, 1997.

[2] J. M. F. Jay West, et al., "Comparison and evaluation of retrospective intermodality brain image registration techniques," *Journal of Computer Assisted Tomography*, vol. 21, pp. 554-566, 1997.

[3] Wieslaw L. Nowinski, et al., "Multiple Brain Atlas Database and Atlas-Based Neuroimaging System," *Computer Aided Surgery*, vol. 2, pp. 42-66, 1997.

[4] National Library of Medicine. *The Visible Human Project* [website]. Available: http://www.nlm.nih.gov/research/visible/visible_human.html (accessed: Nov 2011)

[5] National Library of Medicine. *ITK Insight Toolkit* [website]. Available: http://www.itk.org/ (accessed: Nov 2011)

[6] BIRN. *Tools*. [website]. Available: http://www.birncommunity.org/resources/tools/ (accessed: Nov 2011)

[7] NITRC. *Human Imaging Database (HID)*. [website]. Available: http://www.nitrc.org/projects/hid/ (accessed: Nov 2011)

[8] National Cancer Institute. *Annotation Imaging Markup (AIM)*. [website]. Available: https://wiki.nci.nih.gov/display/AIM/Annotation+Imaging+Markup+%28AIM%29 (accessed: Jan 2012)

[9] National Cancer Institute. *National Biomedical Imaging Archive*. [website] Available: https://wiki.nci.nih.gov/display/ImagingKC/Imaging+Knowledge+Center (accessed: Jan 2012)

[10] Guo-Liang Yang and C. T. Lim, "Singapore National Medical Image Resource Centre (SN.MIRC): A World Wide Web Resource for Radiology Education," *Annals Academy of Medicine*, vol. 35, pp. 558-563, 2006.

[11] Philip Korn, Nicholas Sidiropoulos, Christos Faloutsos, Eliot Siegel, and Z. Protopapas, "Fast and Effective Retrieval of Medical Tumor Shapes," *IEEE Trans. on Knowledge and Data Engineering*, vol. 10, pp. 889-904, 1998.

[12] C. E. B. Chi-Ren Shyu, Avinash C. Kak, Akio Kosaka, "ASSERT: A Physician-in-the-Loop Content-Based Retrieval System for HRCT Image Databases," *Computer Vision and Image Understanding*, vol. 75, pp. 111-132, 1999.

[13] G. S. Jérôme Declerck, Jean-Philippe Thirion, Nicholas Ayache, "Automatic retrieval of anatomical structures in 3D medical images," *LNCS: Computer Vision, Virtual Reality and Robotics in Medicine*, vol. 905, pp. 151-162, 1995.

[14] Yanxi Liu, Frank Dellaert, and W. E. Rothfus, "Classification Driven Semantic Based Medical Image Indexing and Retrieval," Robotics Institute, Carnegie Mellon University, Technical Report CMU-RI-TR-98-25, 1998.

[15] Dorin Comaniciu, David Foran, and P. Meer, "Shape-based image indexing and retrieval for diagnostic pathology," Proceedings of the 14th International Conference on Pattern Recognition, pp. 902-904, 1998.

[16] Usha Sinha and H. Kangarloo, "Principal Component Analysis for Content-based Image Retrieval," *RadioGraphics*, vol. 22, pp. 1271-1289, 2002.

[17] A. M. a. J. Gomes, "Semantic based categorization, browsing and retrieval in medical image databases," Proceedings of the 2002 International Conference on Image Processing, pp. III-145 - III-148, 2002.

[18] J. Fultz. SQL Azure and Windows Azure Table Storage. *MSDN Magazine*. Available: http://msdn.microsoft.com/en-us/magazine/gg309178.aspx (accessed: Nov 2011)

[19] Microsoft Corp. *Windows Azure Platform Training Course*. [website]. Available: http://msdn.microsoft.com/en-us/windowsazure/wazplatformtrainingcourse.aspx (accessed: Nov 2011)

[20] National Cancer Institute. *VASARI*. [website]. Available: http://cabig.cancer.gov/action/collaborations/vasari/ (accessed: Jan 2012)

[21] J. Mather. *DICOM Example Files*. [website]. Available: http://www.mathworks.com/matlabcentral/fileexchange/2762-dicom-example-files (accessed: Nov 2011)

[22] Michael Brady, David Gavaghan, Andrew Simpson, Miguel Mulet Parada, and R. Highnam, "eDiamond: a Grid-enabled federated database of annotated mammograms," in *Grid Computing – Making the Global Infrastructure a Reality*, Fran Berman, G. Fox, and T. Hey, Eds., Chichester: John Wiley & Sons, pp. 923-944, 2003.

[23] N. B. C. Foundation. *Lifepool*. [website]. Available: http://www.lifepool.org/ (accessed: Nov 2011)

[24] Charlie Kaufman and R. Venkatapathy, "Windows Azure™ Security Overview," 2010.

# Algebraic Constructs for Querying Provenance

Murali Mani, Mohamad Alawa, Arunlal Kalayanasundaram
University of Michigan, Flint
303 E. Kearsley St, Flint, MI 48502
{mmani, malawa, arunlalk}@umflint.edu

*Abstract*— **Provenance that records the derivation history of data is useful for a wide variety of applications, including those where an audit trail needs to be provided, where the trust-level attributed to the sources contribute to determining the trust-level in results etc. There have been different efforts for representing provenance information, the most notable being the Open Provenance Model (OPM). OPM defines structures for representing the provenance information as a graph with nodes and edges, and also specifies inference queries that can be expressed in Datalog/SQL. However, the requirements of a query language for provenance information go much beyond those that can be expressed using only inference queries. In our work, we build on OPM and propose two classes of algebraic constructs for querying provenance information: content-based operators that operate on the content of nodes and edges, and structure-based operators that operate on the graph structure of the provenance graph. An user can express a query as a workflow by composing these content-based and structure-based operators. Our operators are powerful, and an user can express a wide variety of interesting queries on the provenance data, that go much beyond simple inference queries as expressible using Datalog/SQL. As part of our evaluation, we show different queries and how they can be expressed using our constructs.**

***Keywords-Provenance; graph; data model; query language; algebraic operators***

## I. Introduction

Provenance (also referred to as lineage, parentage, pedigree, genealogy, or filiation) describes the steps by which data was derived. Provenance has been found useful for a wide variety of scenarios, where one needs to determine the attribution of a data item. For instance, a doctor may need to find the sources from which a data item was obtained to determine whether the item is clinically valid; in data gathering and analysis, the sources from which individual data items are obtained is used for later verification.

In [6], the authors describe two different granularities of provenance. In workflow provenance, which is coarse-grained, each process in the workflow manipulates a set of data items producing another set of data items; the granularity for provenance recording and querying is at the level of data items. In data provenance, which is fine-grained, the queries deal with identifying which pieces of a data item contributed to form a piece of another data item. A well-studied example of data provenance is in the context of SQL queries, where users may want to find why a particular row is in the result (or in some cases, why it is not in the result) [4, 7, 9, 10, 13, 18, 24]. Our focus in this work is on workflow provenance as studied in [1, 5, 15, 16, 22]. In [12] as well as in [14], the authors describe a single system that supports both data and workflow provenance.

Initial works on provenance were tightly integrated with the application they dealt with, such as scientific workflows [1, 5, 8, 15]. OPM provenance model [20] was developed to serve as an application-independent model for representing provenance. OPM represents the provenance information of an application as a graph, where nodes represent artifacts (data items), processes (that manipulate data items), and agents (that control processes), and edges represent causal dependencies. A simple fictitious example of milk powder production, distribution and export is shown in Fig. 1. It shows collection of milk, production of milk powder, distribution at restaurants and at retail stores, exporting, and candy manufacturing and distribution. As in OPM, an ellipse is used to represent an artifact, and a rectangle is used to represent processes. The causal dependencies are shown in a simplified manner than OPM for easier understanding.

While OPM explains how provenance information can be represented, the manipulation of provenance information is limited to inference queries, such as determine all the items that are directly or indirectly produced by a particular data item. An example based on the example application shown in Fig. 1 is to determine the candy bars that are affected by a particular set of milk powder. These queries can be represented in Datalog or using SQL [17].

However, limiting ourselves to such inference queries as provided by OPM is insufficient for a variety of applications, and several interesting queries cannot be expressed. Two example queries that cannot be expressed using inference queries as provided by OPM are described below.

Query1 (shortest path): Suppose there is a time duration associated with processes in Fig. 1, that shows the time it takes for the process from start to finish. If a particular set of milk powder is found to be contaminated, we may want to find the earliest set of products in domestic market that are produced by this contaminated set of milk powder. This requires shortest path type of queries.

Query 2 (sub-graph matching): Suppose there is a pattern that the user is looking for, which can be specified as a graph (and using regular expressions to describe paths). A user may want to find the occurrences of this pattern in the provenance graph. An example with regards to Fig. 1 is finding cases of milk powder produced in Michigan being used for candy manufacturing in France and sold in French retail stores.

There is considerable work in the graph database community on graph query languages and graph algebras, such as [11]. However, these are too generic and often not practical for provenance data. Furthermore, these languages focus mainly on sub-graph isomorphism, and cannot express

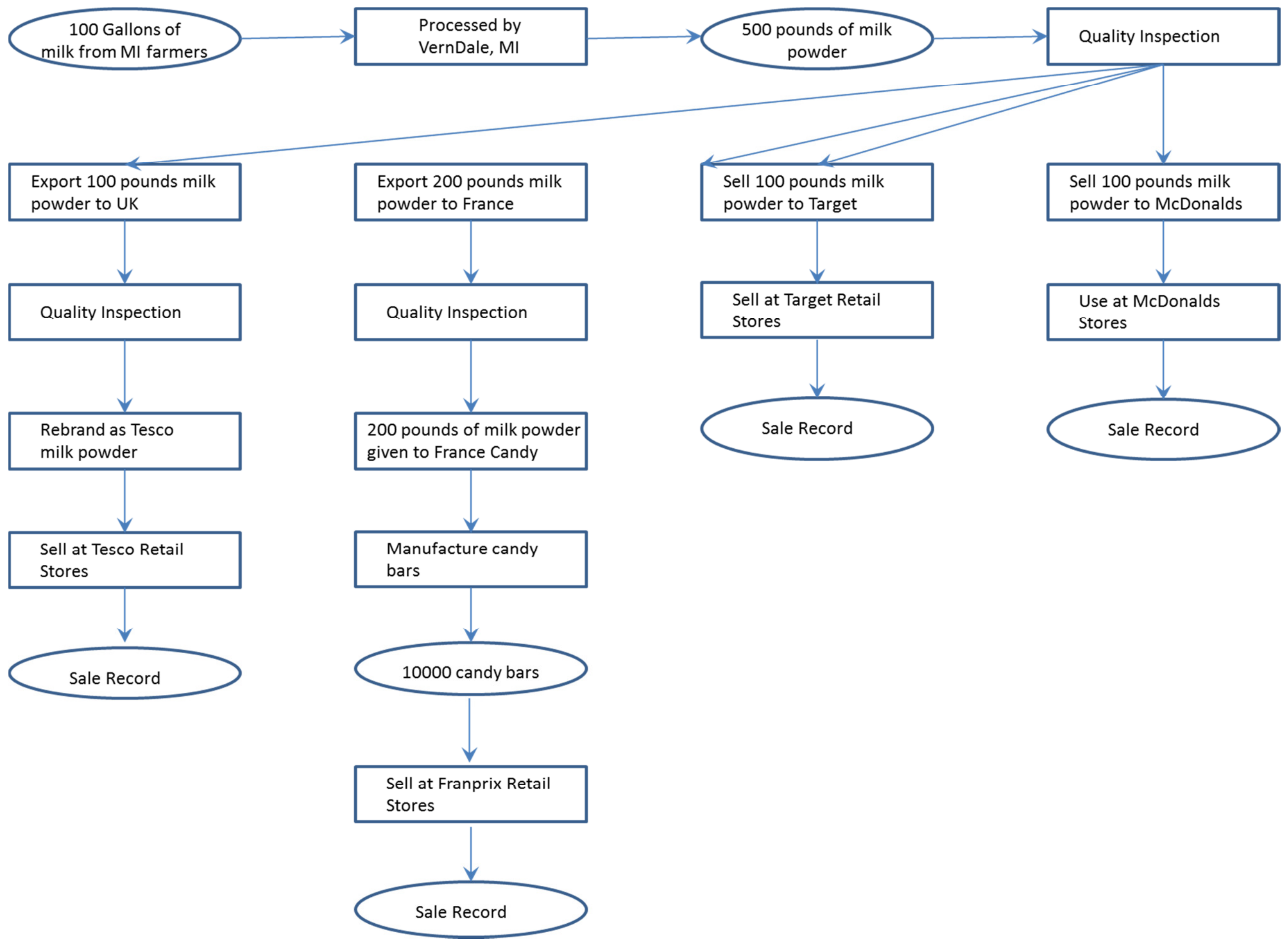

100 Gallons of milk from MI farmers
Processed by VernDale, MI
500 pounds of milk powder
Quality Inspection
Export 100 pounds milk powder to UK
Export 200 pounds milk powder to France
Sell 100 pounds milk powder to Target
Sell 100 pounds milk powder to McDonalds
Quality Inspection
Quality Inspection
Sell at Target Retail Stores
Use at McDonalds Stores
Rebrand as Tesco milk powder
200 pounds of milk powder given to France Candy
Sale Record
Sale Record
Sell at Tesco Retail Stores
Manufacture candy bars
Sale Record
10000 candy bars
Sell at Franprix Retail Stores
Sale Record

Figure 1. An example application illustrating provenance data. Of the 500 pounds of milk powder produced, 200 pounds are used for local sales, and 300 pounds are exported. Of this, 200 pounds exported to France are used to manufacture candy bars, which are then sold at retail stores.

queries such as shortest path. Recently, OPQL [16], a query language for provenance data was developed to support user friendly languages for manipulating provenance; their query language is motivated by [11]. An OPQL query is translated into an SQL query for processing; this implies that OPQL cannot express wide range of queries such as shortest path queries, which cannot be expressed in Datalog/SQL.

In this paper, we propose generic algebraic operators for querying provenance. We classify our operators into two categories: content based operators that query the content information, and structure based operators that query the graph structure of provenance. A user can express a query as a workflow that integrates these two categories of operators seamlessly. This provides a powerful framework that can be used for answering a wide variety of useful provenance queries, including shortest path and sub-graph matching.

Our contributions in this paper are as follows:

1) We define a set of algebraic operators for querying the content of nodes and edges in provenance graph, as well as the structure of the provenance graph. Our operators are powerful, yet simple, and we believe they will form the basis for any future provenance query languages.
2) A user can express a query as a workflow that integrates both content-based and structure-based operators. This provides a powerful framework that can express a wide variety of useful provenance queries, much beyond what can be expressed using Datalog/SQL based languages.
3) We evaluate the expressiveness of our query model. We consider our own example application (Fig. 1), and show how different queries can be expressed. We also consider some queries from the third provenance challenge [23], and study how these queries can be expressed using our query model.

**Outline:** The outline for the rest of the paper is as follows. In Section 2, we outline the OPM data model [20] that we use for representing the provenance data; we simplify the OPM data model so that readers who are not aware of the OPM data model are not carried away by the details, and can focus on the query model. In Section 3, we describe the algebraic constructs that we define. We divide this section into three parts; we first describe the content based constructs, we then describe the structure graph based constructs, finally we describe how a user can express a query by integrating these constructs. In Section 4, we describe our evaluation, where we examine different queries and how they can be expressed in our model. Section 5 discusses related work, and Section 6 concludes this paper.

## II. Representation of Provenance Information

Let us first examine in some detail as to how the provenance information is represented. Graph based data models have been used for provenance in the past, including for OPM [20], and also other works such as [13]. OPM describes three kinds of nodes: *artifacts*, which represent a state of an object at a particular time instant and is immutable; *processes*, which represent an action or a series of actions performed on artifacts and producing new artifacts; *agents*, which represent contextual entities that act as a catalyst for a process. In OPM, an artifact is represented as an ellipse, a process as a rectangle and an agent as an octagon. In our running example in Fig. 1, the 500 pounds of milk powder is an artifact, quality inspection is a process. Any artifact, process or agent may have annotations, which are additional information associated with these entities. The annotations are represented as a set of property-value pairs. Example annotations that can be added to the example shown in Fig. 1 include: range of serial numbers of the milk powder cans produced; time it takes for the process that processes milk and produces milk powder.

OPM also describes five kinds of edges (edges are called dependencies in OPM): *used* representing processes that used an artifact; *wasGeneratedBy* representing artifacts that were generated by a process; *wasControlledBy* representing processes that were controlled by an agent; *wasDerivedFrom* representing artifacts that were derived from an artifact (used for a dataflow view of provenance); *wasTriggeredBy* representing processes that were trigged by a process (used for a process oriented view of provenance). In Fig. 1, the process "Processed by Verndale" *used* 100 gallons of milk; 500 pounds of milk powder *wasGeneratedBy* the process "Processed by Verndale". The process "Sell at Tesco Retail Stores" *wasTriggeredBy* the process "Rebrand as Tesco Milk Powder". In our data model, we do not explicitly indicate the type of the edge; the type of an edge can be immediately inferred based on the nodes that the edge connects.

OPM also describes multi-step inferences and multi-step edges, which are based on transitive closure kind of operators (widely used in the context of directed graphs). For instance, the process "Sell at Tesco Retail Stores" has a multi-step *wasTriggeredBy* dependency on the process "Export milk powder to UK". Also two automatic completion rules are described in OPM.

Our work focuses on query language constructs for provenance. In this paper, we continue to use the same types of nodes and edges as described in OPM. However, we simplify the model and do not distinguish between the different types of edges (as mentioned above, the type of an edge can be inferred based on the types of nodes that the edge connects).

## III. Algebraic Constructs

In this section, we define our query language constructs. We divide our query language constructs into two categories: content based constructs that work on the annotations associated with the nodes and edges, and structure based constructs that work on the graph structure of the provenance graph. We will then describe how these two sets of constructs can be integrated to form a powerful query model, that can express a wide variety of interesting queries.

### A. *Constructs for Querying Content*

The content-based query constructs are used to select nodes or edges based on the annotations associated with them. In short, these constructs operate on a set of entities (which can be nodes or edges), and produce a set of entities as a result; the graph structure of provenance such as edges or paths between two nodes is not used by these constructs.

Some of the operations that are expressible using content based query constructs are listed below. See that our content based query constructs are similar to those in relational algebra, and we borrow notations from relational algebra to represent the different query constructs.

1. Select a set of entities based on a selection condition (used to select nodes and edges based on annotations). This is expressed as $\sigma(S, c)$, where $S$ is a set of entities, and $c$ is the condition. The result of selection is $S' \subseteq S$; each entity in *S'* satisfies the condition specified in *c*.

   Example 1 (Select nodes based on a filtering condition): In Fig. 1, select all processes that are involved with "selling at a retail store". The result of this would include: selling of milk powder at Tesco retail stores, selling of milk powder at Target retail stores, and selling of candy at Franprix retail stores.

   Suppose the process "Selling at Tesco Retail" is as follows (other processes have similar representation):
   process-id = PTESUK101
   annotation:

   process type: Sell
   description: Sell at Retail Store
   duration of process: Jan 10, 2011 – Jan 25, 2011
   details of store: Tesco Retail, Cardiff

   Then the selection condition can be represented as annotation.description LIKE "Sell at Retail Store". Note that the above is very similar to the select operator in relational algebra; we need to use path expressions similar to XPath (XML path language) as annotations can be arbitrarily nested.

2. Project properties, such as ids or annotations from a set of nodes. This is similar to the project operator in relational algebra, and is denoted as $\pi_{aList}(S)$, where $S$ is the set of entities and *aList* is the set of properties of these entities that are to be projected. The result is a set of entities corresponding to the *aList* that is projected. Because annotations can be arbitrarily nested, we use a path expression to indicate annotations to be projected.

   Example 2 (projection of process-id): Find all the process-ids from the set of processes shown in Fig. 1.

3. Set operations such as union, intersection, difference, cross product of two sets of nodes.

4. Aggregate operations such as min, max, sum, count, average on a set of entities.

   Example 3 (aggregation): How many processes are involved with "selling at a retail store". This involves first selecting processes that are involved with "selling at a retail store" (Example 1), and then counting the number of processes that are the result of the selection.

## B. *Constructs for Querying Structure*

The constructs for querying structure use the graph structure of the provenance graph as the basis of querying. Remember that the provenance graph consists of a set of nodes and edges, with each edge connecting two nodes (as shown in Fig. 1). Let us first examine a set of basic functions that are used to further define additional operators. The basic functions include:

1. *from(e)*: returns the node from where the edge e starts.
2. *to(e)*: returns the node where an edge e ends.
3. *from$^{-1}$(n)*: returns the edges that start from node n.
4. *to$^{-1}$ (n)*: returns the edges that end in node n.
5. *next(n)*: returns all the nodes such that there is an edge from node n to it.
6. *prev(n)*: returns all the nodes, such that there is an edge from that node to n.

Example 4: In Fig. 1, let QIMich denote the node depicting the quality inspection process that took place in Michigan. The query next(QIMich) produces four nodes: export 100 pounds to UK, export 200 pounds to France, sell 100 pounds to Target, sell 100 pounds to McDonalds.

We can write quite complex queries using the above functions. For instance, to find all nodes that have a path of length 2 from QIMich, we can combine next operators; first, a next operator finds all the nodes reachable from QIMich by a path of length 1; use the next operator again to find paths of length 1 from these nodes (implying a path of length 2 from QIMich). These can be expressed in Datalog (requires a join). Assume the relation nextTable(d, s) indicates that node d is a node in next(s). The relation pathOfLen2 defines the nodes that are at a path of length two from the node QIMich.

pathOfLen2(y) :- nextTable(x, QIMich), nextTable(y, x).

In our work, instead of defining such operators, we define a very powerful general selection operator on the structure graph of provenance, that takes a structure graph as input, and produces another structure graph as output. Let *G(N, E)* denote a provenance graph with *N* as the set of nodes and $E$ as the set of edges. Let $f_n$ be a selection condition on the nodes, and $f_e$ be a selection condition on edges. The selection operator, denoted as $\sigma_g$, is defined as:

$\sigma_g(G, f_n, f_e) = G' = (N', E')$ , where
$N' \subseteq N$, is the set of nodes that satisfy the condition in $f_n$,
$E' \subseteq E$, is the set of edges that satisfy the condition in $f_e$, and the nodes connected by an edge in *E'* are both in *N'*.

See that the selection operator takes a provenance graph *G* as input, along with $f_n$ and $f_e$; the result of the selection is another graph *G'*, which is a sub-graph of the original graph. We require that edges chosen must only be those such that the nodes connected by an edge must be in *N'*. The reason for this is that we need to select both the nodes connected by an edge for that edge to be selected. There are no restrictions on how to specify the conditions $f_n$ and $f_e$. This allows complex conditions to be specified, much beyond what can be expressed in Datalog/SQL. At the same time, our operator maintains a lot of useful properties including the elegance of a simple algebraic operator with clear semantics. Several queries, including reachability, shortest path, sub-graph matching etc can be expressed using this general selection operator. A few example queries that can be expressed using this general selection operator are given below.

Example 5 (descendant): Given a structure graph *G = (N, E)* (graph *G* has *N* nodes and *E* edges), and a set of *N'* nodes, return the structure graph including the nodes in both *N'* and *N*, the nodes reachable in *G* starting from any node in *N'*, and include the edges between these nodes. Expressing this as a general selection operator, $f_n$ selects the nodes:
$$(N \cap N') \cup \{n \mid n \in N, \exists s \in N', there\ is\ a\ path\ from\ s\ to\ n\ in\ G\}$$
$f_e$ selects edges on a path in *G* starting from a node in *N'*.

For instance in Fig. 1, we can find all the descendants of the process: "Processed at Verndale, MI" if we want to determine what are all the processes and artifacts that were directly or indirectly caused by this process. The result of this selection will be the entire graph except for the first node "100 gallons of milk from MI farmers" and the edge starting from this node. One can define similarly an ancestor operator with appropriate conditions for $f_n$ and $f_e$.

Example 6 (in-betweener): Given structure graph *G = (N, E)* (graph *G* has *N* nodes and *E* edges), and a set of *N'* nodes, return the structure graph including the nodes in both *N'* and *N*, all the nodes that are on a path between two such nodes, and all the edges on these paths.

This can be expressed as a general selection operator where $f_n$ selects the nodes: $(N \cap N') \cup \{n \mid n \in N, \exists a, b \in N', n\ is\ a\ node\ on\ a\ path\ from\ a\ to\ b\ in\ G\}$
$f_e$ selects edges on a path in *G* between two nodes in *N'*.

For instance in Fig. 1, we can find the graph in between the nodes "Proceessed at Verndale, MI", "Sell at Tesco Retail", and "Sell at Franprix Retail". Another example of the in-betweener operator is shown in Fig. 3.

Example 7 (path matching): For this, we first define a labeling function, that maps nodes and edges to labels. The path matching operator works on the graph after the labeling and selects all paths in the original graph that match the path pattern. For this, the selection condition $f_e$ selects the edges:
$$\{e \mid \exists p, p\ is\ a\ path\ in\ G\ after\ labeling, p\ matches\ the\ path\ pattern, and\ e\ is\ an\ edge\ in\ p\}$$
$f_n$ selects the nodes such that each edge *e* that is selected either starts from the node or ends in the node.

An example of path matching is shown in Fig. 2.

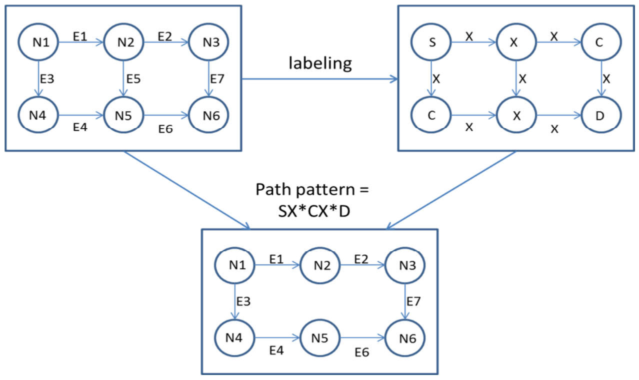


Figure 2. Illustrating path matching. Given a provenance structure graph, first the labeling is applied, and the result of matching the path pattern SX*CX*D is shown.

We call the different selection operators, each with its own definition of $f_n$ and $f_e$ as a variation of the general selection operator $\sigma_g$. The shortest path operator is such a variation, where $f_e$ selects the edges that are in the shortest path given two nodes, and $f_n$ selects the nodes that are along the shortest path between the nodes. Sub-graph isomorphism as in [11] is another variation, as the result is a sub-graph of the original provenance structure graph; another variation is an operator that returns a sub-graph pruning away nodes whose in-degree/out-degree are above/below a threshold.

We define union and intersection of provenance graphs; union of two provenance structure graphs returns a new graph that includes all the nodes in these graphs, and all the edges in these graphs; intersection of two provenance structure graphs returns a new graph that includes only the nodes present in both these graphs and the edges that are present in both these graphs.

Even though the general selection operator is powerful, there are queries that require new nodes or edges that cannot be expressed using our selection operator (because the selection operator only returns a sub-graph of the original graph). An operator, called the abstraction operator allows introducing new edges in some situations (such as transitive closure). This operator takes a provenance structure graph $G = (N, E)$ and a set $N'$ of nodes as input, and produces a graph which includes all the nodes in $N \cap N'$. If there is an edge between 2 nodes in $N \cap N'$ in $G$, that edge is kept. If there is a path between 2 nodes in $N \cap N'$ in $G$, and no intermediate node of this path is in $N \cap N'$, then a new edge is added to the resulting structure graph (Fig. 3).

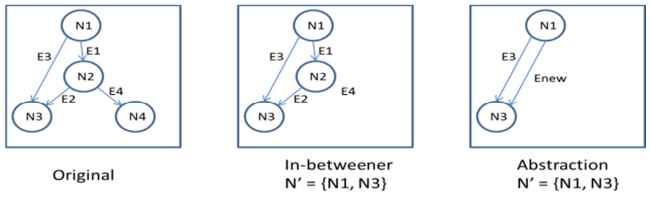


Figure 3. Example showing in-betweener operator (expressed using the general selection operator), and abstraction; both these operators consider the nodes {N1, N3} as input.

## C. Integrated Query Model

In the previous two sections, we examined different operators for querying the content of provenance entities and for querying the structure of provenance graph. We need to integrate these two operator sets to be able to express a wide range of interesting queries. For this, we define one more operator that takes a structure graph and projects the set of provenance entities (nodes and edges in the graph). This projection operator is denoted as $\pi_g(G)$; it returns the set of nodes and edges in G. Different content-based constructs can be applied on this set of provenance entities. Also note that the basic structure functions (from, to, next etc) can be used to select entities from a structure graph.

In addition, any implementation of our query model will provide a pre-defined set of variations of the general selection operator. For instance, an implementation may provide descendant, ancestor and shortest-path variations of the general selection operator. Furthermore, for any operator, multiple physical level implementation alternatives can be provided.

Let us look at a fictitious, but complete example based on the provenance structure graph shown in Fig. 1. Assume that a certain batch of milk powder is found to be contaminated, and we need to find all candy bars that are affected (so that they can be recalled). Suppose we also want to find the total financial loss (the total worth of the affected candy bars).

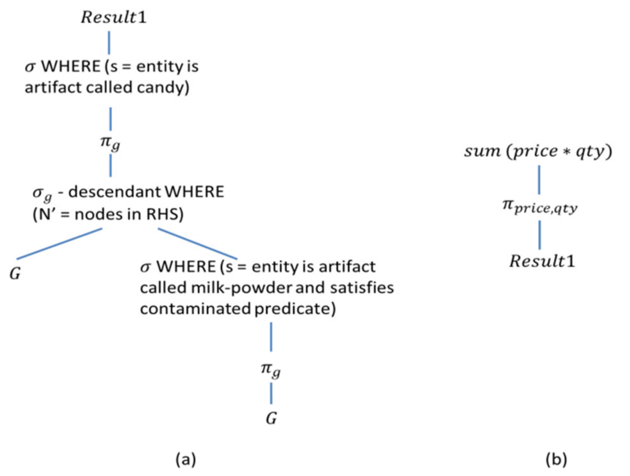

Result1
σ WHERE (s = entity is artifact called candy)
$\pi_g$
$\sigma_g$ - descendant WHERE (N' = nodes in RHS)
G
σ WHERE (s = entity is artifact called milk-powder and satisfies contaminated predicate)
$\pi_g$
G
sum (price * qty)
$\pi_{price,qty}$
Result1
(a)
(b)

Figure 4. (a) computes the candy bars that are affected; (b) computes the total financial loss

Fig. 4 shows how this query can be answered. From the structure graph, G, the $\pi_g(G)$ operator first computes the set of entities in G; then the content-based selection operator selects the artifacts that represent the contaminated milk-powder. Now the general selection operator $\sigma_g$- descendant selects the sub-graph of G that represent descendants of the contaminated milk-powder entities. We apply $\pi_g(G)$ to get all the entities in this graph; then the content-based selection operator selects all artifacts that are candy bars. To find the total financial loss, we project (content-based projection) the price and qty of each of the selected candy bar entities from above, and perform aggregation to find the total loss.

## IV. Evaluation

For our evaluation, we consider two more examples from our milk powder example in Fig. 1. We will further consider some example queries from the Third Provenance Challenge [19], and describe how they can be expressed.

Query 1: From Fig. 1, determine how a problematic set of milk powder was produced, transported and processed to make an affected brand of candy.

Fig. 5 (a) shows Query 1. An alternate option (to using two content-based selection operators) is to use one selection operator with predicates combined using OR. Fig. 5 (b) shows Query 2 using shortest-path operator.

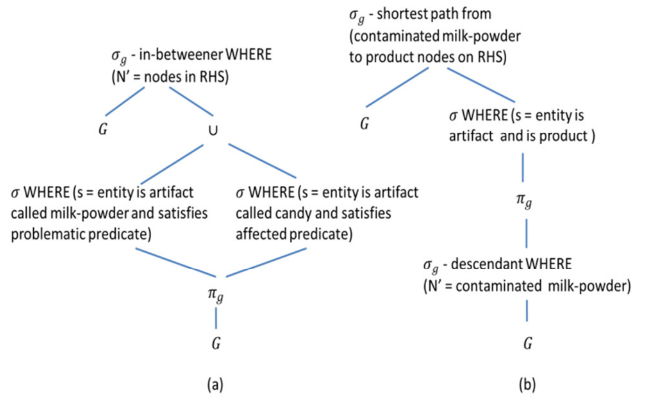

$\sigma_g$ - in-betweener WHERE (N' = nodes in RHS)
$G$
∪
$\sigma$ WHERE (s = entity is artifact called milk-powder and satisfies problematic predicate)
$\sigma$ WHERE (s = entity is artifact called candy and satisfies affected predicate)
$\pi_g$
$G$
(a)
$\sigma_g$ - shortest path from (contaminated milk-powder to product nodes on RHS)
$G$
$\sigma$ WHERE (s = entity is artifact and is product )
$\pi_g$
$\sigma_g$ - descendant WHERE (N' = contaminated milk-powder)
$G$
(b)

Figure 5. (a) Query 1 (b) Query 2 using shortest-path operator

Query 2: From Fig. 1, determine the earliest products produced from known contaminated milk powder.

The Third Provenance Challenge [23] provided provenance information about data from the Pan-STARRS project, that continuously scans the visible sky once a week and builds a time-series of data. The aims of the project include: help detect moving objects that may potentially impact with earth, build a massive catalog of solar system and stars. These are some of the queries from this data.

Query 3 (from [23]): Given a particular detection which files contributed to it?

This is similar to Fig. 4 (a), except that the first content-based selection selects the detection, the general selection is an ancestor operator, and the second content-based selection selects the files.

Query 4 (from [23]): The user finds a particular table with data that they do not expect. Was the range check performed on this table?

Query 4 can be answered using the path matching operator as shown in Fig. 6 (a). Fig. 6 (b) shows Query 5, also using the path matching operator.

Query 5 (from [23]): The user executes a workflow many times over different data sets. Find which of these executions halted.

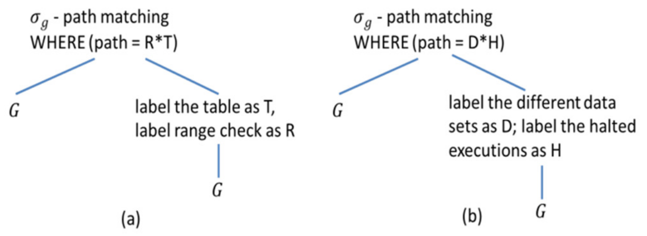

$\sigma_g$ - path matching WHERE (path = R*T)
$G$
label the table as T, label range check as R
$G$
(a)
$\sigma_g$ - path matching WHERE (path = D*H)
$G$
label the different data sets as D; label the halted executions as H
$G$
(b)

Figure 6. (a) Query 4, if there are no paths selected by the path matching general selection operator, that means range check was not performed; if there is a path selected, then range check was performed (b) Partial Query 5, showing that path matching operator can be used to find all paths resulting in a halted execution starting from the specified data sets.

From our evaluation section, we can conclude that a wide range of queries, including shortest path queries can be expressed using the constructs that we defined. Further, the content based operators and the provenance structure graph based operators are combined to express several interesting queries that cannot be expressed using Datalog/SQL.

## V. Related Work

There has been lots of interest in provenance data management in the recent past. The initial works on provenance were tightly integrated with the application they considered [1, 5, 15]. The provenance community realized that it would be beneficial to have a uniform model for representing provenance and this led to the development of the Open Provenance Model (OPM) [20]. OPM describes how the provenance information can be represented in a general fashion; OPM also describes inference queries, such as the nodes that are ancestors or descendants of a node.

Most of the previous works on provenance also had some query mechanism, VisTrails [15] used the VisTrails query language called vtPQL; Kepler [1] provides a query language called QLP, which works on Kepler's proprietary provenance model; ZOOM [5] provides an interface for users to query provenance information similar to inference queries in OPM; Taverna [25] allows query specification using SPARQL query language; Karma [22] supports provenance queries using XPath and SQL. These works are tightly coupled with their underlying provenance representation mechanism, and hence is not general. Also, they are based on SQL or XPath kind of languages, and hence the expressiveness is limited by the expressiveness of these languages.

In [21], the authors propose an algebra for their provenance data model called provenir, which is defined using OWL-DL. The algebraic operators supported are provenance(), for obtaining the complete provenance (which can be expressed using ancestor/descendant operators); provenance_pathway(), which returns a subset of the information returned by provenance(), provenance_context(), which uses a user specified context (can be expressed using a combination of content-based operators for manipulating the context and ancestor/descendant operators). However, there are several other operators that we need for provenance metadata processing which are not supported. In [19], the authors consider scoping of provenance, because the entire provenance might be overwhelming for the user. The authors consider the scenario where the user can explicitly exclude a part of the provenance. In our system, such scoping can be expressed either by developing a variation of the general selection operator that excludes part of the provenance, or by expressing it as first computing the complete provenance (using ancestor/descendant operators) and then excluding the portions that the user wants to exclude. In [26], the authors use the linking of data as supported by Linked Open Data (LOD) cloud; such context can be used for queries and for the interpretation of experimental data. In our work, such joins across multiple graphs can be expressed by unioning the graphs and then performing a join, or by selecting nodes/edges from one graph and using that as a context to select a subgraph of the second graph.

OPQL [16] introduces a query language that is directly defined over the Open Provenance Model (OPM). Here, the authors specify six types of graph patterns based on the patterns described in [11] and define three types of graph matching based on these graph patterns; the authors then define algebra operators for extracting sub-graphs that match a pattern, and for performing set operations such as union, intersection and difference. The implementation of OPQL uses SQL and hence their expressiveness is limited by what is expressible in SQL. Another work that considers querying data provenance is [13]. Here, the authors are concerned with data provenance; however, their query language is general and is applicable to workflow provenance as well. Their query language introduces XQuery style FOR-WHERE-RETURN expressions, with an additional INCLUDE PATH clause which specifies the paths that need to be included in the resulting graph. However, the WHERE clause is restricted so that shortest path kind of queries cannot be expressed, though it is possible to express ancestor/descendant queries. For manipulating graph data models, [11] specifies graph patterns and studies extracting sub-graphs as specified in the pattern.

The works described above [11, 13, 16] all manipulate graphs and return graphs as result. However, the expressiveness is limited to those expressible using Datalog/SQL; queries such as shortest paths are not considered. Also [11] is a query language for graphs in general, and not limited to provenance graph queries; some of the queries that may be interesting to general graphs, may not be interesting to provenance graphs.

Graph querying to a limited extent is supported by commercial database software such as PostgreSQL (http://pgfoundry.org/docman/view.php/1000262/505/README.txt) and MySQL (http://openquery.com/graph/doc). Here, graphs are represented as first class objects (not as tables), and an arbitrary set of graph operations are supported. For instance, MySQL supports shortest path, but does not support sub-graph isomorphism or path pattern matching. Also, they come up with SQL like syntax for all these operations, which we believe will be cumbersome if we want to perform a series of operations on a graph, as in a query plan. In short, we believe that support for graphs within databases is in its infancy, and most implementations are supporting only an arbitrary subset of graph operations. In our work, we provide a mechanism to query graphs that is general, where the users express their query as a workflow (thus are not limited by a text based language like SQL), and one query can express a series of operations to be performed on the graph.

In [2, 3], the authors consider querying of semantic associations from data represented using OWL. Here, the authors talk about given two nodes, how to determine the relationship between the nodes. One type of relationship referred to as $\rho$-pathAssociated (where there is a path between two nodes) can be expressed using our in-betweener operator, which is a type of the general selection operator. For other types of $\rho$-queries (relationships between nodes), we need to define other types of the general selection operator. However, note that in [2] the authors define a language exclusively for querying path information. However, for provenance data, we need more operators than just for querying the path information; we therefore use a workflow based approach, where the user specifies the query as a workflow consisting of any number of operators (as needed). One of the query constructs that we propose is a general selection operator for the structure graph of provenance; the only requirement is that this operator returns a sub-graph of the original graph; no restrictions are imposed on how the selection condition for nodes and edges are specified. Therefore, we are able to express a wide variety of queries including shortest path queries, sub-graph or path matching, ancestor, descendant, etc., while still maintaining a simple and elegant formalism of a selection operator.

## VI. Conclusions and Future Work

In this work, we studied algebraic constructs that can be used for provenance queries. Several interesting queries cannot be expressed using today's graph languages, or provenance query languages, as they rely on translating their queries into SQL/Datalog. We proposed two sets of algebraic constructs for querying provenance: content based operators manipulate the content (or annotation) of the nodes in a provenance graph; structure based operators manipulate the structure of a provenance graph. One of the powerful algebraic operators that we propose is a general selection operator that selects a sub-graph of the provenance structure graph, based on general restrictions on nodes and edges. An user expresses a query in our query model as a workflow by integrating these algebraic constructs. Our query model can express a wide range of interesting queries.

As part of future work, we are currently investigating optimization opportunities for the various operators and for provenance storage. Further, we are considering whether constructs that allow arbitrary addition of nodes and edges are useful for provenance queries, and how they can be supported.

## Acknowledgment

We would like to acknowledge Yaobin Tang, who did initial work on provenance with us, and whose master's thesis led to many of the insights that we describe in this paper. We would also like to acknowledge the discussions with the members of the University of Michigan, Flint, and Ann Arbor database groups, and discussions with Chunhyeok Lim and Shiyong Lu at Wayne State University, who are currently pursuing research on provenance. Last, but not the least, we would like to acknowledge NSF for partially supporting this research.

## REFERENCES

[1] M. Ananad, S. Bowers, and B. Ludascher, Techniques for Efficiently Querying Scientific Workflow Provenance Graphs, In EDBT 2010, pp. 287 – 298.

[2] K. Anyanwu, A. Maduko, and A. P. Sheth, SPARQ2L: Towards Support for Subgraph Extraction Queries in RDF Databases, In WWW 2007, pp. 797 – 806.

[3] K. Anyanwu and A. P. Sheth, $\rho$ -Queries: Enabling Querying for Semantic Associations on the Semantic Web, In WWW 2003, pp. 690 – 699.

[4] O. Benjelloun, A. D. Sarma, A. Y. Halevy, M. Theobald, and J. Widom, Databases with Uncertainty and Lineage, VLDB Journal, 17 (2), 2008, pp. 243 – 264.

[5] O. Biton, S. C. Boulakia, S. B. Davidson, and C. S. Hara, Querying and Managing Provenance through User Views in Scientific Workflows, In IEEE ICDE 2008, pp. 1072 – 1081.

[6] P. Buneman and W-C Tan, Provenance in Databases, In ACM SIGMOD 2007 (Tutorial), pp. 1171 – 1173.

[7] A. Chapman and H. V. Jagadish, Why Not? In ACM SIGMOD 2009, pp. 523 – 534.

[8] A. Chapman, H. V. Jagadish, and P. Ramanan, Efficient Provenance Storage, In ACM SIGMOD 2008, pp. 993 – 1006.

[9] Y. Cui and J. Widom, Lineage Tracing for General Data Warehouse Transformations, In VLDB Journal, 12 (1), 2003, pp. 41 – 58.

[10] T. G. Green, G. Karvounarakis, and V. Tannen, Provenance semirings, In ACM PODS 2007, pp. 31 – 40.

[11] H. He and A. K. Singh, Graphs-at-a-time: Query Language and Access Methods for Graph Databases, In ACM SIGMOD 2008, pp. 405 – 418.

[12] R. Ikeda and J. Widom, Panda: A System for Provenance and Data, In IEEE Data Engineering Bulletin, 33 (3), 2010, pp. 42 – 49.

[13] G. Karvounarakis, Z. G. Ives, and V. Tannen, Querying Data Provenance, In ACM SIGMOD 2010, pp. 951 – 962.

[14] D. Koop, E. Santos, B. Bauer, M. Troyer, J. Freire, and C. T. Silva, Bridging Workflow and Data Provenance Using Strong Links, In SSDBM 2010, pp. 397 – 415.

[15] D. Koop, C. E. Scheidegger, J. Freire, and C. T. Silva, The Provenance of Workflow Upgrades, In International Provenance and Annotation Workshop (IPAW), 2010, pp. 2 – 16.

[16] C. Lim, S. Lu, A. Chebotko, and F. Fatouhi, OPQL: A First OPM-Level Query Language for Scientific Workflow Provenance, In IEEE SCC 2011, pp. 136 – 143.

[17] C. Lim, S. Lu, A. Chebotko, and F. Fatouhi, Storing, Reasoning and Querying OPM-Compliant Scientific Workflow Provenance Using Relational Databases, In Future Generation Computer Systems, 27 (6), 2011, pp. 781 – 789.

[18] A. Meliou, W. Gatterbauer, J. Y. Halpern, C. Koch, K. F. Moore, and D. Suciu, Causality in Databases, In IEEE Data Engineering Bulletin, 33 (3), 2010, pp. 59 – 67.

[19] S. Miles, Electronically Querying for the Provenance of Entities, In IPAW 2006, pp. 184 – 192.

[20] The OPM Provenance Model (OPM), available at http://openprovenance.org/, retrieved: December, 2011.

[21] S. Sahoo, R. Barga, J. Goldstein, and A. Sheth, Provenance Algebra and Materialized View-based Provenance Management, Microsoft Research Technical Report, 2008, available at http://research.microsoft.com/apps/pubs/default.aspx?id=76523, retrieved: December, 2011.

[22] Y. Simmhan, B. Plale, and D. Gannon, Karma2: Provenance Management for Data Driven Workflows, International Journal of Web Services Research (IJWSR), 5 (2), 2008, pp. 1 – 22.

[23] Third Provenance Challenge, available at http://twiki.ipaw.info/bin/view/Challenge/ThirdProvenanceChallenge, retrieved: December, 2011.

[24] A. Woodruff and M. Stonebraker, Supporting Fine-Grained Data Lineage in a Data Visualization Environment, In IEEE ICDE, 1997, pp. 91 – 102.

[25] J. Zhao, C. Goble, R. Stevens, and D. Turi, Mining Taverna's Semantic Web of Provenance, In Concurrency and Computation: Practice and Experience, 20 (5), 2008, pp. 463 – 472.

[26] J. Zhao, S. S. Sahoo, P. Missier, A. P. Sheth, and C. A. Goble, Extending Semantic Provenance into the Web of Data, IEEE Internet Computing, 15 (1), 2011, pp. 40 – 48.