MDDQL-Stat: data querying and analysis through integration of intentional and extensional semantics.

Epaminondas Kapetanios[1]
David Baer
Björn Glaus
Paul Groenewoud
Plirosoft Ltd., Semantic Technologies, Technoparkstr. 1, Technopark Zurick, Switzerland

[1]Epaminondas Kapetanios now works in the Harrow School of Computer Science, University of Westminster

MDDQL-Stat: Data Querying and Analysis through Integration of Intentional and Extensional Semantics

Epaminondas Kapetanios, David Baer, Björn Glaus, Paul Groenewoud
Plirosoft Ltd., Semantic Technologies
Technoparkstr. 1, Technopark Zürich
CH-8005 Zurich, Switzerland

Abstract

*We would like to present a prototype system enabling a rather empirical than a formal approach to the problem of posing queries to a semantically rich (quality aspects, semantic distance, etc.) data integration system {G,S,M} (**G**lobal schema, **S**ources, **M**ediation) through integration not only of intensional but also of extensional semantics. While the first is provided by an alphabet A_g, as given by an ontology based global schema G, and a high level query language (conjunction/disjunction + inequalities + statistical operations), the latter enables synthesizing of data source specific and previously transformed query results according to well-defined set operations for heterogeneous, distributed data sources. Our approach contrasts with other GAV (Global-As-View) related architectures for mediation of integrated read-only views, in that it simplifies query processing while preserving flexibility when adding new data sources, despite the inherited complexity of mappings due to enhanced semantic description of data (semantic distance, quality parameters, etc.) such that statistical results and comparisons become more meaningful.*

Keywords: *Data Integration, Ontology Driven Querying, Statistics, Mediation*

1. Motivation

Data integration systems are mostly characterized by an architecture based on some global schema and a set of sources with the first providing a reconciled, integrated, virtual view of the underlying sources [7]. They are mostly classified either into ***global-as-view*** approaches such as [1], where the global schema is expressed in terms of the data sources, or into ***local-as-view*** approaches such as [5], where the data sources are described in terms of the global schema. A comparison of the GAV and LAV approaches is given in [10]. A completely different approach is "InfoSleuth", which uses an agent-based concept for semantic information integration.

Formally speaking, a *data integration system* I is defined by the triple $< G, S, M >$ where G is the global schema expressed in a language L_g with an alphabet A_g , S is the source schema expressed in a language L_s with an alphabet A_s and M is the *mapping* between G and S. Therefore, querying of integrated data sources takes the form of either $q_s \rightarrow q_g$ or $q_g \rightarrow q_s$ depending on having the query expressed in terms of the language L_{M_s} or the language L_{M_g}, respectively.

In all these approaches, query answering over integrated data sources is bound to some kind of query rewriting as based on *views* [7, 8, 2]. In this paper, we introduce a data integration system as pursued in the MDDQL-Stat data integration framework, where querying also takes place in terms of a GAV-related data integration architecture, where mediation is provided for integrated read-only views as underlying data management policy [3]. MDDQL-Stat extends the query construction (ontology driven) mechanism, as presented in [4], through statistical operations and their application on query vocabulary terms.

The data integration system, however, does not make use of *views* in order to specify mappings between some *mediation schema* and *sources* and, therefore, avoids the inherited necessity of query rewriting, which usually lead to a *maximally contained rewriting* and not an *equivalent* one, as well as the difficulty of specifying usable views [2]. Furthermore, *maximally contained rewriting* assumption cannot be tolerated when query results are bound with statistical evaluations.

This is due to the fact that query processing relies on data source specific generation and distribution of high level, conceptual (sub)query trees through cloning and pruning of the original one, which reflects the submitted query q_g. To this extent, data source specific (sub)queries, such as SQL statements, are generates directly from the data source specific high level (sub)query tree. In addition, synthesis of the result takes place in terms of set-oriented operations, which

Proceedings of the 16th International Conference on Scientific and Statistical Database Management (SSDBM'04)
1099-3371/04 $ 20.00 © 2004 IEEE

have been specified and implemented in such a way that they comply with the semantics of *union*, *difference* and *intersection* of sets of returned results from each data source.

This is, in turn, enabled by the transformation of heterogeneous values into a commonly agreed-upon, user preferred value, prior to synthesis of the query result at the mediator level, such that statistical operations and analytical queries can be performed. In other words, there is no need of manually transforming data values in order to statistically analyze the data, especially when large data repositories are concerned.

On the other side, this approach preserves the flexibility of adding and describing data sources, as known to be the advantage of the LAV-based approaches. This also holds even when advanced semantic descriptions of data sources such as data quality aspects, or advanced mappings such as *semantic distances*, in terms of unclear or fuzzy specification of meanings (very often with legacy systems), are given.

The system has been implemented in *Java* by using a standard client-mediator-wrapper architecture, which addresses all JDBC compliant DBMS's. Since our focus has been semantic heterogeneity, we are, currently, making use of Oracle DBMS as a data repository, however, with different user or database schemas (data sources). Having a user or database schema provided by another DBMS, e.g., SQL-Server, it is simply considered as an additional data source.

The data sources are described in RDF[1]. The integrated result is enriched with information about the origin and the quality of the data source. Quality measures have been introduced. The prototype has been implemented in Java. In addition, a Java based tool supports a semi-automatic description of a data source (relational, object-relational databases).

2. The Data Integration System

Client: The Global Schema and Query Language The global schema G and queries q_g are expressed in the MDDQL-Stat ontology driven query language with an alphabet A_g as provided by *OntoContext*, a contextualized ontology description language. For the sake of simplicity, we will address only conjunctive queries with inequalities and statistical operators within the MDDQL-Stat data integration framework.

Construction of queries takes place in a human-computer interaction driven by the system and in any selected natural (sub)language, according to meaningful suggestions as drawn by an inference engine operating upon the application domain semantics as described by *OntoContext*.

1 Resource Description Framework

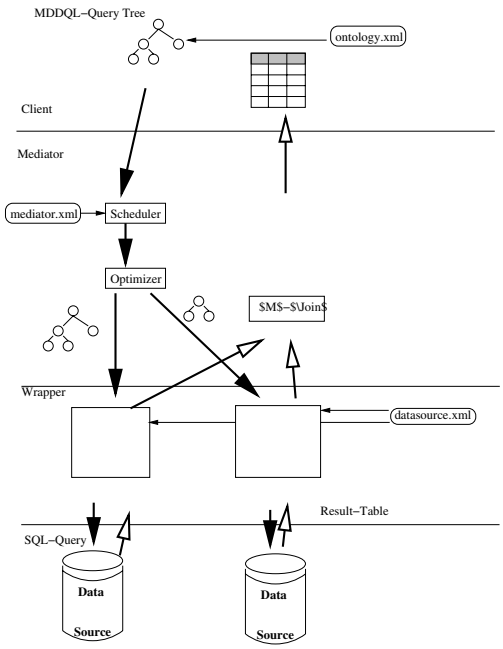

Figure 1. The "big picture" of the system architecture.

More details about the query construction mechanism can be found in [4].

The query language A_g alphabet is, therefore, provided by the terms as defined by the ontology modeling constructs, such as *class*, *instance*, *property*, etc. Each term in the ontology, however, has an additional slot indicating the unique *Mediator Identifier* **MID**, which maps a particular term into a set of various data source elements at the mediator level, such as tables, attributes, values, etc., which might semantically overlap.

Each constructed query by the client, however, is reflected and represented by a high level (conceptual) tree, which is specified by the following constraints: the root of the query tree is always a *Class* or an *Instance* term node. A *Class* or *Instance* term node might have as children other *Class*, *Instance* or *Property* term nodes. A *Datatype Property* term node might have as children other *Datatype Property* term nodes or *Value* term nodes. An *Object Property* term node, i.e., relationship between two *agents*, MUST have children, which are *Classes* or *Instance* term nodes. An *Object Property* term node, i.e., relationship between two *agents*, might have as children *Property* term nodes. A *Value* term node might have as children only *Value* nodes.

Comparison and/or unary statistical operators are assigned to *Value* and *Datatype Property* term nodes only, respectively. Binary or n-anary statistical operations are as-

signed to the whole query tree. An example of a high-level query tree as depicted at the top of figure 2, reflecting the query where one asks for "the `symptoms`, `date of birth` and `weight` for `female` patients. Nodes marked with an "E" refer to *Class* or *Instance* term nodes, nodes marked with an "R" refer to *Object Property* term nodes, nodes marked with "P" to *Datatype Property* term nodes and nodes marked with "V" to *Value* term nodes. However, all values referring to "female" in different data sources should be presented in the final query result as "F".

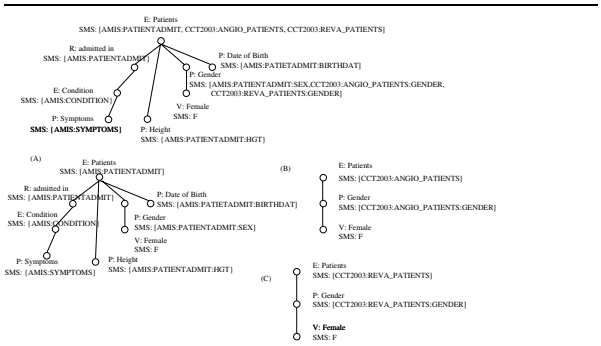

Figure 2. Generating data source specific, high level sub-query trees

Given that the high level query term nodes are constructed implicitly from the ontology modeling constructs, they also carry on the corresponding MID, which is replaced with the corresponding set of SMS's, into which the MID has been mapped, when query arrives at the mediator, as described below.

Mediator: The Mappings and their Description The mediator is responsible for mapping the MID (Mediator Identifier) and, implicitly, the concept in the ontology to the corresponding data source elements. In order to preserve flexibility, when new data sources are added, as well as to alleviate the task of describing semantically rich mappings, we

- *introduced* a "dot" based notation of describing the data source elements, which we call Storage Medium Symbol (SMS). An SMS is defined by the constraints: a) all SMS constituents underly a sequence order <data source>:<table>:<attribute>:<value>, from left to right, indicating *inclusion*, b) for the sake of simplicity, SMS's referring to values are represented by :<value> only, since inclusion is implied by the parent node, which is always a datatype property term node.
- considered as being modeling structures themselves rather than logic-based descriptions. This enables the assignment of attributes to *mappings* such as *semantic distance*.

An example of SMS's is given in figure 2 and for the example of the high level query tree. They are retrieved from an XML based description of the modeling constructs of the mappings. An example of a partial description of the mappings is given in the following:

```
<mddql:mediator>
  <mddql:mid mid="m100">
    <mddql:sms distance="1.0">
      AMIS:PATIENTADMIT
    </mddql:sms>
    <mddql:sms distance="1.0">
      CCT2003:ANGIO_PATIENTS
    </mddql:sms>
    <mddql:sms distance="1.0">
      CCT2003:REVA_PATIENTS
    </mddql:sms>
  </mddql:mid>
  <mddql:mid mid="m501">
    <mddql:sms distance="1.0">
      AMIS:PATIENTADMIT:SEX
    </mddql:sms>
    <mddql:sms distance="1.0">
      CCT2003:ANGIO_PATIENTS:GENDER
    </mddql:sms>
    <mddql:sms distance="1.0">
      CCT2003:REVA_PATIENTS:GENDER
    </mddql:sms>
  </mddql:mid>
  . . . . . .
</mddql:mediator>
```

In the example presented above, all data source elements have their origin in two data sources, namely *AMIS* and *CCT2003*. Both of them have a table with an attribute, i.e., "SEX" in "AMIS", "GENDER" in "CCT2003", which are mapped to the ontology concept (property) *Gender* (`mid="m501"`). Similarly, the concept "Patients" is reflected in both data sources. In "AMIS", the respective table is called `PATIENTADMIT`. In the "CCT2003" source, it exists in two different tables (`ANGIO_PATIENTS` and `REVA_PATIENTS`). Therefore, [CCT2003:ANGIO_PATIENTS] and [CCT2003:REVA_PATIENTS] are considered as being two *virtually different data sources*. Gender contains only the two categorical values "F" (`mid="m1000"`) and "M" (`mid="m1001"`).

To this extent, adding a new data source becomes quite flexible and easy, in comparison with the GAV-related approaches, since all we need to express is the data source element in terms of an SMS and position it into the corresponding modeling construct of a mapping. In addition, further attributes of a mapping can also be expressed such as the attribute *semantic distance* [9] through a coefficient $d \in (0, 1]$.

The mediator, subsequently, generates a series of high level sub-queries under the following condition: assign to each data source a particular high level query tree, which includes only nodes with SMS's as refer to that particular data source (see also figure 2, where *three* high level (sub)query trees (A), (B) and (C) are generated, with (B) and (C) for the two virtual data sources). This can be recognized easily by having a look at the first constituent of an SMS, which always refers to the data source. Data source

specific high level (sub)query trees are generated through cloning and pruning of the global, high level query tree.

In addition, the mediator synthesizes the query result according to set-oriented operations, which are, however, defined for the needs of sets of tuples as returned from the execution of the sub-queries at the different data sources. We call these operations M-Join, M-Union and M-Difference, since they are defined at the mediation level and for sound but not complete, partly overlapping data sources as the extensional semantics of the MDDQL-Stat data integration system.

Statistical operations are all implemented as being *classes*, since semantics of statistical operators vary considerably when more than one data source is considered, e.g., *the average of data source specific averages does not equal the average of all returned tuples from the data sources*. They are made available at the mediator level, however, one can have them running on the client site too. The latter option is recommended when the amount of data upon which the statistical operations are performed and, therefore, which needs to be transferred to the client, is rather low.

Prior to synthesis of the query result, the generated high level (sub)queries are delegated to the relevant data source where they get transformed into the data source specific query language such as SQL with respect to the required value transformations as described below.

Wrapper: Data source description Two main tasks have to be accomplished by the wrapper: (a) transformation of the MDDQL-Stat high-level *(sub)*query tree into a database specific query language such as SQL, (b) transformation of the results into the desired (as stated by the high level query) formats, i.e., integration of extensional semantics. For both tasks, the wrapper needs to access the data source (database) description (database schema elements such as tables, attributes, primary/foreign keys, data types, measurement units, etc., enhanced by quality parameters such as *completeness* and *soundness*), as provided in an RDF like syntax [6].

For task (a), generation of data source specific query statements takes place in terms of traversing the MDDQL-Stat high level (sub)query tree in a depth-first strategy, where the generation of the statement is conceived as an automaton having as an initial state $q_0 = \{S_0, F_0, W_0\}$, where $S_0 \equiv F_0 \equiv W_0 \equiv \{\}$, with S, F, W representing the SELECT, FROM, WHERE (conditional) parts of the statement. In order to accomplish this task, however, the wrapper needs the information about the database schema elements as provided by the data source description.

Following the example given in figure 2, the generated SQL queries at each relevant data source, which correspond the high level sub-queries (A), (B) and (C), would have taken the form:

SELECT {ID}, C.SYMPTOMS, P.HGT, P.BIRTHDAT **FROM** PATIENTADMIT P, CONDITION C **WHERE** P.PATIENT_ID = C.PATIENT_ID **AND** P.SEX = 'female'

SELECT {ID} **FROM** ANGIO_PATIENTS A **WHERE** A.GENDER = 'F'

SELECT {ID} **FROM** REVA_PATIENTS R **WHERE** R.GENDER = '1'

For task (b), the wrapper transforms the (sub)query results into a common format based on the information about the scale, the unit, etc., as well as access to a unit transformation table. The quality parameters are not used for the query at this stage, but will be added to the result, leaving it to the mediator and the user to decide how to use them.

Acknowledgments: We would like to express our thanks to Prof. Dr. Gustavo Alonso, Dept. of Computer Science, ETH-Zürich, for his comments and valuable feedback.

References

[1] C. Goh, S. Bressan, S. Madnick, and M. Siegel. Context Interchange: New Features and Formalisms for the Intelligent Integration of Information. *ACM Transactions on Information Systems*, 17(3):270–293, 1999.

[2] A. Halevy. Answering Queries Using Views: A Survey. *VLDB Journal*, 2001.

[3] R. Hull. Managing semantic heterogeneity in databases: A theoretical perspective. In *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*. ACM Press, 1997.

[4] E. Kapetanios and P. Groenewoud. Query Construction through Meaningful Suggestions of Terms. In T. Andreasen, A. Motro, H. Christiansen, and H.-L. Larsen, editors, *Flexible Query Answering Systems*, volume 2522 of *Lecture Notes in Artificial Intelligence*, pages 226–239, Copenhagen, Denmark, October 2002. Springer.

[5] T. kirk, A. Y. Levy, Y. Sakiv, and D. Srivastava. The Information Manifold. In *Proc. of the AAAI 1995 Spring Symp. on Information Gathering from Heterogeneous, Distributed Environments*, pages 85–91, 1995.

[6] O. Lassila and R. R. Swick. Resource Description Framework (RDF) model and syntax specification. Technical report, World Wide Web Consortium, 1999. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.

[7] M. Lenzerini. Data integration: a theoretical perspective. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 233–246. ACM Press, 2002.

[8] Y. Papakonstantinou and V. Vassalos. Architecture and Implementation of an XQuery-based Information Integration Platform. *Data Engineering Bulletin*, March March 2002.

[9] L. Pontieri, D. Ursino, and E. Zumpano. An approach for the extensional integration of data sources with heterogeneous representation formats. *Data and Knowledge Engineering*, 45:291–331, 2003.

[10] J. D. Ullman. Information Integration Using Logical Views. In *Proc. of the 6th Inter. Conf. on Database Theory (ICDT 97)*, volume 1186 of *Lecture Notes in Computer Science*, pages 19–40. Springer, 1997.