Maintenance and Mediation in Database Federations

Ulf Leser
TU Berlin, Sekr. E-N 7, Einsteinufer 17,
D-10587 Berlin, Germany
E-mail: leser@cs.tu-berlin.de

ABSTRACT

This paper proposes a novel approach to the processing of global queries in a tight federation of heterogeneous relational databases. It is based on query correspondence assertions, which are declarative rules that describe semantic correspondences between different relational schemas. A mediator uses these rules to translate queries against the federated schema into semantically contained sequences of queries against source databases. We argue that this approach has significant maintenance advantages over traditional approaches.

1. INTRODUCTION

Many organisations today face the problem that information is spread over many distributed and heterogeneous databases. This is often inevitable, for instance to achieve high performance for mission-critical applications or due to organisational requirements, such if partner companies give access to their databases.

However, there is a growing need for integrated access methods. Decision support systems (DSS) typically need simultaneous access to many databases. Building a corporate data warehouse, which is an attractive alternative ensuring high performance of query execution, is not always feasible. Keeping a warehouse up-to-date can be impossible if highly time-critical data is required, such as stock market data. Information required by the DSS might be only available through commercial providers where queries are limited and costly. Such providers will permit specific queries, for instance to obtain information about a particular company, but it can be impossible or too expensive to download and integrate the complete database.

Another alternative is the construction of a tight federation of databases ([11]). This has to solve two main problems: the construction of the federated schema and the processing of queries against this schema. Solutions to both problems must be based on knowledge about correspondences between heterogeneous schemas which potentially contain many types of structural and semantic conflicts ([6]). Consider the example given in Tab.1. The three *sales* tables are semantically closely related, but not identical. For instance, S1 stores net sales, while S2 stores gross sales.

As components of the federation can autonomously change over time, maintenance is of crucial importance. The federation has to cope with evolving source schemas, changing user demands, temporarily unavailable sources etc. It is essential that maintainability is addressed already in the design of a federated system; ignoring this issue leads to expensive and time consuming adjustments and raises the danger of producing erroneous data.

In this work we propose a novel approach to the problem of processing global queries in a federation of relational database. We observe that many projects in the past have concentrated on schema integration techniques and use very simple query processing mechanisms. We argue that these systems lack flexibility in coping with evolving components and do therefore not support maintainability sufficiently. In section 2 we will discuss aspects of maintainability in database federations. Section 3 presents our modular system architecture. Section 4 introduces the concept of query correspondence assertions (QCA) and sketches the algorithm which answers global queries based on a set of QCAs. We

Federated schema	`sales(P#,Total,Ctr),prod(P#,Stock,Desc)`	Gross sales per country and product, together with stock data and product descriptions.
Source 1 (S1)	`sales(P#,Nettotal,Ptype),vat(Ptype,Vat)`	Net sales per product in UK only. VAT depends on type of product.
Source 2 (S2)	`sales(P#,Total,Stock,Ctr)`	Gross sales per country, includes product stock.
Source 3 (S3)	`product(P#,Stock,Desc)`	Products on stock.

Table 1: Three source and one federated schema. *P#* is a product key.

then evaluate our approach with respect to maintainability. Section 5 discusses related work and section 6 concludes.

2. MAINTENANCE IN DATABASE FEDERATIONS

Maintenance of database federations has many facets. We focus on logical aspects and ignore for instance technical or social issues. Maintaining federations is none-trivial because components are autonomous and can independently change over time. An informal list of situations requiring maintenance is given in the following:

Schema evolution in a source: we distinguish between changes that modify the scope of the source schema and changes that are only re-organisations. While the latter only affects schema correspondences, the former can require a change in the federated schema. If for instance S2 decides to use net sales, only the translation of queries is affected. But if S3 removes the *Desc* attribute, the federated schema needs to drop it too. On the other hand, if S2 would drop the *Stock* attribute, no change in the federated schema is necessary, as *Stock* information is also available from S3.

Changing requirements on the federation: such cases can trigger changes in the federated schema. The integration of parts of source schemas which were previously not contained in the federated schema can become necessary; other parts may become obsolete.

Addition of new sources: changing demands on the DSS can require the integration of new data sources, for instance to include data from another stock market. Organisational changes can extend the scope of the federation, for instance if products from a newly acquired company have specific, previously not necessary attributes.

Deletion of sources: in contrast to temporary unavailability, the permanent deletion of a source often needs to be reflected in schema changes on the federation level. If for instance S3 is removed, *Desc* must be dropped from the mediator schema.

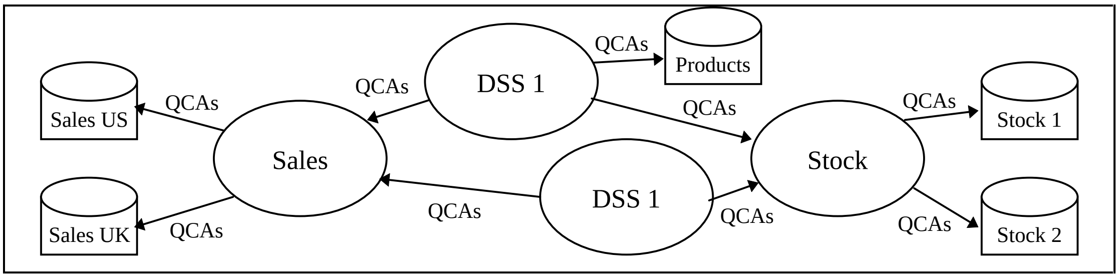

Fig.1 A mediator network forming a modular federation. Mediators (ovals) are be used by other mediators as data sources. Arrows define 'uses'-relations; the correspondences are described by QCAs.

Unavailability of sources: sources can become temporarily unavailable, for instance due to network problems. Such problems should not affect the execution of global queries that do not require those sources. For instance, unavailability of S1 should not affect queries for sales in the USA.

In general, we call situations that require a change in the federated schema *schema-effective*. Changes that remove (add) attributes from (to) a source that are not present in another source but (shall) appear in the federated schema are always schema-effective. In contrast, neither the deletion of S1 nor the addition of a similar source for the USA would be schema-effective.

3. A MODULAR ARCHITECTURE FOR DATABASE FEDERATIONS

We suggest a modular architecture for database federations (Fig. 1). One key idea is the decomposition of the integration problem into small units, another is the declarative representation of schema correspondences. We refrain from building a large, complex global schema, but instead propose the creation of a set of small schemas where each captures a specific aspect of the DSS, such as sales, customers, stock etc. Each is maintained by a software agent, termed *mediator* ([13]), which is able to answer queries against his schema by accessing relevant data sources. Knowledge about these data sources, in particular about their schema, is specified in the form of *query correspondence assertions* (sec. 4). A mediator can rewrite queries against his mediator schema into an executable and semantically contained sequence of source queries using an algorithm which interprets the QCAs (sec. 4.1). Therefore, a mediator can be treated as a virtual database and can be used by other mediators.

We do not make assumptions about how a mediator schema is created in first place. It could for instance be created by using schema integration ([2]). However, we believe that a source-independent creation of mediator schemas has several advantages. To do this, an expert, having in mind the available and the required entities, builds a mediator schema from scratch and relates it to the appropriate source schemas in a second step by defining QCAs. We will show in the next section that QCAs are sufficiently expressive to allow full query processing in such a setting. The advantage of this approach is that it enables homogeneous and well-structured mediator schemas which are easier to query and maintain than the large structures resulting from many schema integration methods.

4. QUERY CORRESPONDENCE ASSERTIONS

Each mediator needs knowledge about the correspondences between tables and attributes in his schema and tables and attributes in the source schemas. A human administrator must provide this knowledge in the form of QCAs. A QCA is syntactically a set-equation between two views, one defined as a query against the mediator schema (*mediator query*), and one defined as a query against one source schema (*source query*). The views are necessary to project out attributes that are used by only one of the queries. Semantically, a QCA defines the intensional equivalence between the queries and states that the extension of the source view is a subset of the (only virtually existing) extension of the mediator view. Technically, the mediator query must be a conjunctive query ([1]) with comparisons between variables and constants, while the source query can be any arbitrary relational query. The schema correspondences in our example are expressed by the following QCAs (we use DATALOG style):

```
(R1) sales(P#,T,'UK') =: v1(P#,T) ⊇ s1(P#,T) := sales(P#,X,Pt),vat(Pt,V),P=X*V;
(R2) sales(P#,T,C),prod(P#,S,D) =: v2(P#,T,S,C) ⊇ s2(P#,T,S,C) := sales(P#,T,S,C);
(R3) prod(P#,S,D) =: v3(P#,S,D) ⊇ s3(P#,S,D) := product(P#,S,D);
```

Mediator queries are on the left, source queries on the right hand side. R1 defines that the extension of the query *sales(P#,T,C),C='UK'*, projected to *P#* and *T*, is a superset of the extension of the corresponding source query. But R1 cannot simply retrieve the value of *T*, but needs to calculate it from other values to bridge semantic differences (*T* including resp. excluded VAT). Note that semantically related data is spread over different databases. For instance, the extension of the global *sales* table is the union of the extensions of s1 and s2, as specified with R1 and R2.

4.1 Answering Queries using QCAs

A *user query* is an arbitrary query against the schema of a mediator. The mediator answers user queries by finding sequences of source queries (*plans*). A plan is *valid* if its result, which is the union of its views, is semantically contained in the user query. Valid plans are guaranteed to obtain only correct tuples for the user query. Finding valid plans is equivalent to the problem of answering queries using views and is known to be NP-complete ([9]). The mediator in principle needs to enumerate all combinations of mediator views and test each combination for semantic containment (see e.g. [1]). We need to restrict mediator queries to conjunctive queries because the algorithm can prove semantic containment only in this case. For space limitations we can not go into details. The complete algorithm which also performs several optimisations can be found in [8], together with a discussion on the integration of non-relational sources.

We explain the mechanism by giving an examples. Imagine the user query *sales(P#,T,'UK'),prod(P#,S,D)*. To find valid plans, the mediator first determines all mediator views whose defining queries contain a relation of the user query: for *sales* these are {v1,v2}, for *prod* these are {v2,v3}. Then it calculates all combinations: <v1,v2>, <v1,v3>, <v2,v2>, <v2,v3>. <v2,v2> and <v1,v2> are immediately excluded because they do not retrieve values for *D*. The remaining two plans are *expanded* by replacing them with their definitions:

$$\texttt{v1(P\#,T),v3(P\#,S,D)} \rightarrow \texttt{sales(P\#,T,'UK'),prod(P\#,S,D)}$$

$$\texttt{v2(P\#,T,S,C),v3(P\#,S,D)} \rightarrow \texttt{sales(P\#,T,C),prod(P\#,S}_1\texttt{,D}_1\texttt{),prod(P\#,S}_2\texttt{,D}_2\texttt{)}$$

To check whether an expanded plan is valid, we need to find a *containment mapping* from the symbols of the user query to the symbols of the plan. For both plans such a mapping exists. However, if the user query would contain the condition *C='USA'*, no mapping could be found for the first plan, because containment mappings cannot map a constant to another constant. Note that the second plan contains two instances of the attributes *S* and *D*. The mapping will map *D* and *S* from the user query to D_2 and S_2, because D_1 is projected out by v2.

Once a containment mapping is found, the mediator executes the corresponding source queries of all QCAs in the plan and retrieves tuples according to the signatures of the views. These values are stored in virtual mediator tables as defined by the mediator query of the QCA. Finally, the original query is executed on these virtual tables.

If we also restrict source queries to conjunctions, we can use the algorithm to translate queries against a source schema in queries against the federation. Imagine an arbitrary query against the schema of one source S[1]. We first construct all valid plans by building combinations of source queries of QCA describing S. Note that this uses QCAs in the reverse order than above because now we rewrite the initial query using source queries, not mediator queries.

[1] Note that sources are usually described by sets of QCAs, unlike in our example.

In a second step, each such plan is re-translated into sequences of source queries of *arbitrary* sources. Imagine an application A, developed for S2, that calculates statistics about sales. A will not be easily adaptable to other sources because the probably hard-coded queries will not function any more. Using the procedure described above, we can use A for the entire federation without having to change the queries. This allows users or applications of a specific source to access the federation through the schema they are familiar with.

4.2 Maintaining Federations built on QCAs

The approach we suggest in this work has several maintenance advantages. In general, small schemas are always easier to maintain than large schemas. Changes in sources mostly affect only those mediators that use these sources directly. Because correspondence relationships are explicitly specified, we can react on many changes by adjusting, deleting or adding rules. If for instance S2 chooses to store net totals and VAT separately, only R2 needs to be changed. As another example, imagine the acquisition of a new company selling a new set of products. Their sales database can be integrated by simply adding appropriate rules to a sales-mediator. User queries requiring sales data will immediately obtain answers that fully consider the new data source. Mediators using this sales-mediator need not be aware of the change.

However, schema-effective changes need to be propagated through the network. Note that a change in the schema of a mediator M is not necessarily schema-effective for mediators that use M as data source.

Temporary unavailability of sources affects each mediator only to a minimal degree. Mediators will simply ignore the corresponding QCAs during query planning. Queries not requiring this source remain answerable. If for instance S1 goes off-line, queries for sales in countries others than the UK are still possible.

4. RELATED WORK

Database federations have been extensively studied for many years. To our best knowledge, the important issue of maintainability was not addressed adequately before. Many approaches have concentrated on schema integration. They usually address query processing by refining the federated schema until direct correspondences between tables can be established. Global queries can then be rewritten by simple substitution (e.g. [4], [5]). Apart from yielding large and complex schemas, this approach has the disadvantage that evolution in the sources requires a new schema integration, which potentially leads to completely different global schemas [10]. This is particularly critical if a modularised approach is pursued.

Our architecture is not novel in itself. Federations using other federations as data source are for instance described in [11]. [13] describes mediator networks and emphasis the maintainability of *thin mediators*. However, both works do not present actual methods to achieve modular federations.

Other authors have also observed the advantages of using a declarative representation of correspondences. [12] uses *schema correspondences* for schema integration. The rules used in [3] are effectively restrictions of QCAs to single tables on both sides. [7] introduces *virtual views* which are QCAs where the mediator query must be a single relation.

5. CONCLUSIONS

We proposed a novel mechanism for the representation of semantic correspondences between heterogeneous schemas. Used in a modular federation architecture, integrated systems emerge that can be maintained with less effort than in approaches concentrating on schema integration. We showed several cases where changes can be treated by changing the set of rules.

Some questions remain open. For instance, we have only given an informal description of maintainability problems in database federations. We have no precise criteria that help to identify schema-effective modifications. However, we believe that most changes in daily routine are not schema-effective. Furthermore, we ignored issues of user awareness - if for instance S1 is unavailable, queries for all countries are technically still executable, but do not obtain complete results any more. Users must be notified of such problems.

ACKNOWLEDGEMENT:

This work was supported by the German Research Council, grant no. GRK 316. We are grateful to the referees for helpful comments on the original version of this paper.

REFERENCES:

[1] Abiteboul, S., R. Hull, et al. (1995). „Foundations of Databases." Reading, Massachusetts, Addison-Wesley Publishing Company.

[2] Batini, C., M. Lenzerini, et al. (1986). "A Comparative Analysis of Methodologies for Database Schema Integration." ACM Computing Surveys **18**(4): 323-364.

[3] Catarci, T. and M. Lenzerini (1993). "Representing and Using Interschema Knowledge in Cooperative Information Systems." Journal for Intelligent and Cooperative Information Systems **2**(4): 375-399.

[4] Hammer, J. and D. McLeod (1993). "An Approach to Resolving Semantic Heterogeneity in a Federation of Autonomous, Heterogeneous Database Systems." Journal for Intelligent and Cooperative Information Systems **2**(1): 51-83.

[5] Huhns, M. N., N. Jacobs, et al. (1992). „Enterprise Information Modeling and Model Integration in Carnot." 1st Int. Conf. on Enterprise Integration Modeling, MIT Press, Cambridge pp. 290-299.

[6] Kashyap, V. and A. Sheth (1996). "Semantic and Schematic Similarities between Database Objects: a context-based approach." The VLDB Journal **5**(4): 276-304.

[7] Kim, W., I. Choi, et al. (1995). „On Resolving Schematic Heterogeneity in Multidatabase Systems." In *Modern Database Systems*. W. Kim, ed. New York, ACM Press, Addison-Wesley Publishing Company**:** 521-550.

[8] Leser, U. (1998). „Combining Heterogeneous Data Sources through Query Correspondence Assertions." 1st Workshop on Web Information and Data Management, in conjunction with CIKM'98, Washington, D.C. To appear.

[9] Levy, A. Y., A. O. Mendelzon, et al. (1995). „Answering Queries using Views." 14th ACM PODS, San Jose, CA pp. 95-104.

[10] Motz, R. (1998). „Propagation of Structural Modifications to an Integrated Schema", 2nd East European Symposium on Advances in Databases and Information Systems; LNCS 1475; Poznan, Poland. pp. 163-174

[11] Sheth, A., J.A. Larson (1990). "Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases." ACM Computing Survey **22**(3).

[12] Spaccapietra, S., C. Parent, et al. (1992). "Model Independent Assertions for Integration of Heterogeneous Schemas." The VLDB Journal **1**(1): 81-126.

[13] Wiederhold, G. (1992). "Mediators in the Architecture of Future Information Systems." IEEE Computer **25**(3): 38-49.