Using Source Capability Descriptions for the Integration of Digital Libraries

Eike Schallehn Martin Endig

Department of Computer Science, University of Magdeburg
P.O. Box 4120, D-39016 Magdeburg, Germany
{schallehn|endig}@iti.cs.uni-magdeburg.de

Abstract

For the integration of bibliographical data during the last years attempts were made to apply methods from the area of federated databases and mediators. In this paper we describe our approach using the loosely coupled federated system FRAQL. To integrate heterogeneous systems with limited query capabilities we introduce concepts for source description and their usage in query re-writing. This is presented in the context of a generic adapter that can be used in highly distributed scenarios which uses XML and related technologies.

1 Introduction

The integration of existing digital libraries and electronic catalogues is considered one of the main tasks in the digital library community. Various projects were started to realize this integration of bibliographical data. For this purpose newly developed methods and technologies from the research area of federated databases and mediator systems were applied successfully, though some problems remained unsolved. All approaches have a similar multi-tier architecture in common that provides adapters and/or wrappers for certain classes of sources to deal with interface heterogeneity and, depending on the approach, various issues of data representation.

As new sources may have to be included in an ongoing process, support for the development of adapters is a current research issue. Adapter generation and configuration techniques were considered conforming to the strategy of separating generic parts and source specific parts of such a component. All of these approaches have their limitations regarding interface heterogeneity and varying levels of cooperativeness of data providers. While there are suitable solutions for non-cooperative providers offering for instance only access to their data via Web interfaces, more efficient and less error-prone solutions are possible, if we can assume a certain level of cooperativeness.

The currently most applicable solutions for data integration are based on loosely coupled federated database systems in connection with a multi-database language and mediator based-systems [DD99]. Query languages supporting the integration of heterogeneous sources are for instance multi-database languages like MSQL [GLRS93] and SchemaSQL [LSS96]. Examples of system implementations are federated database systems like Pegasus [ASD$^+$91] or IBM DataJoiner [VZ98] as well as mediator-based systems like TSIMMIS [GPQ$^+$97]. The TSIMMIS successor MIX [BGL$^+$99] employs XML for data exchange between the mediator and the wrapper. The description of query capabilities for sources and its usage in query processing was for instance discussed related to the TSIMMIS [LYV$^+$98], Garlic [HKWY97] and Information Manifold [LRO96] systems.

2 Application Requirements

The integration of autonomous sources via public HTTP interfaces, while still error-prone, is often the only solution for integrating data from non-cooperative providers. More stable and more efficient solutions are conceivable if we assume a certain level of cooperativeness from the providers of bibliographical data. Therefore, we consider the following aspects and address them in our approach. *Minimal provider-side resource consumption:* Our approach takes the possibly limited query capabilities into consideration. Query processing is split between source and integration layer according to a source description. *Minimal provider-side implementation efforts:* On the one hand we simply rely on query capabilities of the source system that can easily be wrapped by the source-specific part of the adapter. On the other hand the provider supplies XML results conforming to their specific DTD. While we do not presume that all systems currently provide XML, this format can easily be created mainly because of the extensive tool support. *Efficient transfer:* One critical aspect is the size of results that have to be transfered over possibly slow network connections. This is done by moving sub-queries with high selectivity to the source where possible. *Efficient access for global applications:* By offering an object-relational interface to global applications we rely on applied and powerful standards. The integration layer offers additional query capabilities allowing the user to post standard SQL queries. Furthermore, it is flexible in terms of data to be imported and ways to integrate this data.

3 Integrating Bibliographical Data

For the integration of data an appropriate federated system is required. In principle the architecture of such a federated system comprises four layers: an application layer, a federation layer, an adapter layer and a data source layer. Our focus is on the *federation layer*, where part of the query processing and the integration of data from several data sources takes place, and on the *adapter layer*, where data source specific data models are transformed and interface heterogeneity is handled.

3.1 Federation Layer

The federation layer provides services for defining and querying integrated views on bibliographical data from different sources. These services are implemented by using the FRAQL language and query processor [SCS00]. FRAQL is a query language for object-relational database federations. It extends SQL by features for defining federations, accessing meta-data in queries, restructuring query results, and resolving integration conflicts. Furthermore, it is extensible by user-defined data types as well as functions and it supports dynamic integration of new sources.

In FRAQL a federation is a set of databases consisting of relations. A database can be provided by a full-featured DBMS or even by a Web source encapsulated by a wrapper. FRAQL is based on a simple object-relational data model: it supports the definition of object types and object tables derived from types. Types can be organized in a specialization hierarchy. Object tables represent global virtual relations of the federation. Here we distinguish between import and integration relations. An *import relation* is a projection of a local relation of a data source, where for each global attribute a mapping onto a local attribute can be defined. This mapping can be described as simple renaming, as value conversion by using a user-defined conversion function or with the help of an explicit mapping table. An *integration relation* is a view on other global relations combined by using operators like union, θ-join and outer join, which are extended by reconciliation functions for conflict resolution. The query processor evaluates a FRAQL query by decomposing it into sub-queries according to the definition of integration relations.

These sub-queries are sent to the adapters and evaluated by the data source. Finally, the results are combined in the query processor.

3.2 Adapter Layer

Several adapters for the FRAQL system exist, that allow access to various database systems. Our focus is on an XML-based adapter for cooperative providers. On the one hand, the access to a data source is depended on the local query interface and the local query capabilities of the source. For global query re-writing a general description of the query capabilities is necessary. On the other hand, in each local data source a specific data format is used, that must be transformed into a general format.

Given an integration layer offering an object-relational database interface we currently consider two kinds of sources. If the source system is a fully-fledged relational database system we can simply pass through the re-written sub-query. The second class of sources consists of systems like Internet databases offering Web interfaces. Usually these sources support only constant selections with predefined comparison operators. Conforming predicates can be combined conjunctively. If $\mathcal{R}$ is the set of names of relations exported by source $\mathcal{S}$ and $Attr$ is the set of query-able attributes of relations in $\mathcal{R}$, the source description $SrcDesc$ can be defined like as follows.

$SrcDesc \triangleq \mathcal{R} \times RelDesc$	
$RelDesc \triangleq \mathbb{P}(Pred) \times ComDescr$	(description of export relation R)
$Pred \triangleq Attr \times \mathbb{P}(\Theta)$	(supported constant selections)
$ComDescr \subseteq \mathbb{P}(Attr)$	(possible predicate combinations)
$\Theta = \{>, \geq, =, \neq, <, \leq, LIKE\}$	(globally supported comparison operators)

As an example consider two sources exporting the relations *Books* and *BookStore*, both conforming to a global type *Publications*. Source 1 exports relation *BookStore* and supports equality matches on $author$ or $title$ or a combination of both, i.e.

$$SrcDesc_1 = (\{(BooksStore, \{(authors, \{=\}), (title, \{=\})\}, \{\{authors\}, \{title\}, \{authors, title\}\})\})$$

Source 2 exports relation *Books* and supports equality match on title, i.e.

$$SrcDesc_2 = (\{(Books, \{(title, \{=\})\}, \{\{title\}\})\})$$

According to these source descriptions request are created. These requests can be transmitted as a simple HTTP request. If the following query is posed the query processing is like described below and displayed in Figure 1.

```
select *
from Publications
where title = 'ODBMS' and
    authors = 'Saake' or
    authors = 'Heuer';
```

The sources process all sub-queries consisting of supported conjunctive predicates corresponding to the source description (4). In this example we assume that fetching the relation $Books$ is possible without specifying a predicate. This can not always be done easily and may require additional provider-side functionality. Unsupported predicates are processed in the federation layer (3). This includes attribute selections involving attributes from this relation only. The selection criterion for each source relation is in disjunctive normal form (2). Results from various source relations are integrated (1) either by a simple union or by applying same-object identification and reconciliation of result tuples.

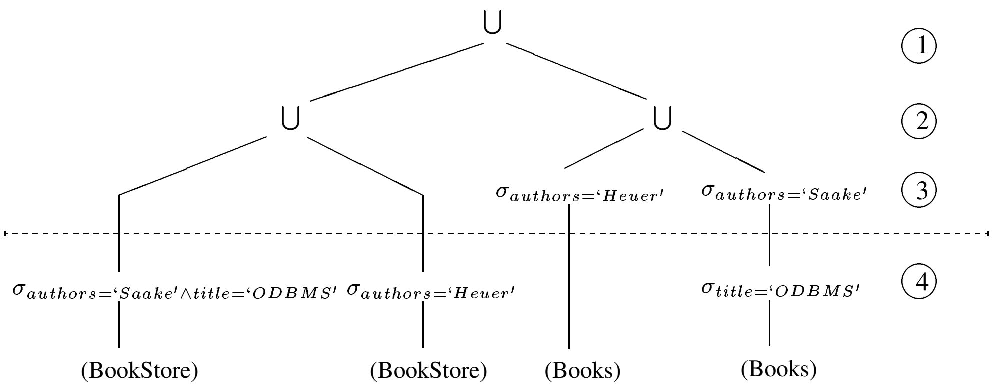

Figure 1: Re-written query example for source systems

In this simple example only one global relation was involved. Further processing like joins, projections, attribute selections crossing various global relations and constant selection involving attributes that are modified by either reconciliation or by mapping function during result set integration have to be processed afterwards.

All relevant results of special requests are received as provider specific XML documents. During the exchange of bibliographical data different kind of publications with different kind of attributes in different document type definitions are considered. For the realization of the adapter the introduction of a general XML format *and* an appropriate transformation is needed. Due to the used concepts and methods on the federation layer, the general format is derived from its object-relational model. For the mapping between the XML conforming to a provider specific DTD and the adapter specific document type we use XSLT in the adapter. XSLT is a stylesheet language for transforming XML documents into other XML documents.

Provider-side XML:

```
<ROWSET>
 <ROW num="1">
  <ID>1</ID>
  <TITLE>ODBMS</TITLE>
  <AUTHORS>Heuer</AUTHORS>
 </ROW>
</ROWSET>
```

Adapter-side XML:

```
<COLLECTION>
 <RESULTSET tablename="PUBLICATIONS">
  <RECORD>
   <FIELDVALUE name="ID"> 1 </FIELDVALUE>
   <FIELDVALUE name="TITLE"> ODBMS </FIELDVALUE>
   <FIELDVALUE name="AUTHORS"> Heuer </FIELDVALUE>
  </RECORD>
 </RESULTSET>
</COLLECTION>
```

For a special document type an appropriate XSLT file is required as part of the adapter configuration in order to transform it to the adapter-specific XML document.

4 Conclusion and Outlook

We presented our approach for the integration of bibliographical data that employs the FRAQL federation service for loosely coupled integrated systems. It comprises a suitable solution for the integration of a variety of different sources where the providers accept to cooperate in providing a simple wrapper for query processing based on a description of their query capabilities. For further research we investigate tool support for implementing provider-side components and for specifying the XSLT mapping.

References

[ASD+91] R. Ahmed, P. De Smedt, W. Du, W. Kent, M.A. Ketabchi, W. Litwin, A. Rafii, and M.-C. Shan. The Pegasus Heterogeneous Multidatabase System. *IEEE Computer*, 24(12):19–27, December 1991.

[BGL+99] C. Baru, A. Gupta, B. Ludäscher, R. Marciano, Y. Papakonstantinou, P. Velikhov, and V. Chu. XML-based Information Mediation with MIX. In A. Delis, C. Faloutsos, and S. Ghandeharizadeh, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMod-99)*, volume 28,2 of *SIGMOD Record*, pages 597–599, New York, June 1–3 1999. ACM Press.

[DD99] R. Domenig and K. R. Dittrich. An Overview and Classification of Mediated Query Systems. *SIGMOD Record*, 28(3), 1999.

[GLRS93] J. Grant, W. Litwin, N. Roussopoulos, and T. Sellis. Query Languages for Relational Multidatabases. *VLDB Journal*, 2(2):153–171, 1993.

[GPQ+97] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. D. Ullman, V. Vassalos, and J. Widom. The TSIMMIS Approach to Mediation: Data Models and Languages. *Journal of Intelligent Information Systems*, 8(2):117–132, March/April 1997.

[HKWY97] L. M. Haas, D. Kossmann, E. L. Wimmers, and J. Yang. Optimizing Queries Across Diverse Data Sources. In *VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases*, pages 276–285, 1997.

[LRO96] A. Levy, A. Rajaraman, and J. Ordille. Querying Heterogeneous Information Sources Using Source Descriptions. In T. M. Vijayaraman et al., editors, *Proceedings of the Twenty-second International Conference on Very Large Data Bases, September 3–6, 1996, Mumbai (Bombay), India*, pages 251–262, Los Altos, CA 94022, USA, 1996. Morgan Kaufmann Publishers.

[LSS96] L.V.S. Lakshmanan, F. Sadri, and I.N. Subramanian. SchemaSQL - A Language for Interoperability in Relational Multi-Database Systems. In T. M. Vijayaraman, A.P. Buchmann, C. Mohan, and N.L. Sarda, editors, *VLDB'96, Proc. of 22th Int. Conf. on Very Large Data Bases, 1996, Mumbai (Bombay), India*, pages 239–250. Morgan Kaufmann, 1996.

[LYV+98] C. Li, R. Yerneni, V. Vassalos, H. Garcia-Molina, Y. Papakonstantinou, J. Ullman, and M. Valiveti. Capability Based Mediation in TSIMMIS. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 27(2), 1998.

[SCS00] K. Sattler, S. Conrad, and G. Saake. Adding Conflict Resolution Features to a Query Language for Database Federations. In *Proc. 3nd Int. Workshop on Engineering Federated Information Systems, EFIS'00, Dublin, Ireland, June*, 2000. *To appear.*

[VZ98] S. Venkataraman and T. Zhang. Heterogeneous Database Query Optimization in DB2 Universal DataJoiner. In A. Gupta, O. Shmueli, and J. Widom, editors, *VLDB'98, Proc. of 24rd Int. Conf. on Very Large Data Bases, 1998, New York City, New York, USA*, pages 685–689. Morgan Kaufmann, 1998.