Integrating Bibliographical Data from Heterogeneous Digital Libraries *

Eike Schallehn, Martin Endig, and Kai-Uwe Sattler

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

Abstract. The integration of bibliographical data today is considered one of the most important tasks in the area of digital libraries. Various available sources of bibliographical information vary widely in terms of data representation and access interfaces. To overcome this heterogeneity during the last years attempts were made to apply methods developed for information system integration, like federated databases and mediators. In this paper we describe our approach using the loosely coupled federated system FRAQL. Furthermore, we present a generic adapter that can be used in highly distributed scenarios which uses XML and related technology for transfer and homogenization of data. As an application scenario we describe global citation linking for integrated digital libraries.

1 Motivation

Electronic publishing and the provision of information about publications and their availability are current trends that were supported by the growth of the Internet during the last ten years. Publishers, libraries and computer scientists engage in developing digital libraries and electronic catalogues to make the publications themselves (digital contents) and their accompanying meta-data (bibliographical data about the publications) available and to offer various services based on the provided information. Those libraries and catalogues currently serve certain areas of research, for instance the ACM and IEEE digital libraries providing meta-data and full text for computer science in general and the DBLP service providing meta-data for the field of database research.

The integration of existing digital libraries and electronic catalogues is considered one of the main tasks in the digital library community [SC99]. From a user point of view a single point of access would allow to look up publications more efficiently. Furthermore, the integration makes it possible to derive information about implicit relationships among data spread over various sources, like for instance citation linking information. For this purpose newly developed methods and technologies from the research area of federated database and mediator systems were applied successfully, though some problems remained unsolved. These problems exist because of the complexity resulting from the high level of heterogeneity regarding data representations, contents and interfaces. 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

* This research was partially supported by the BMBF (08SFB031) and DFG (FOR345/1-1)

solutions are possible, if we can assume a certain level of cooperativeness. In this paper we describe an integration architecture for this scenario, where the provider offers simple query capabilities and sends results in XML conforming to a source specific DTD.

2 Related Work

As integration of various sources for bibliographical information is one of the major tasks in the development of digital libraries. Concepts from the area of information system integration are adopted to this usage scenario. A federated database system is a distributed system consisting of different heterogeneous database systems or local file clusters [HM85]. The currently most applicable solutions are based on loosely coupled federated database systems in connection with a multi-database language and mediator based-systems. Good overviews are given in [DD99,BKLW99]. 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], Garlic [CHS+95] or Information Manifold [LRO96]. Some of the above mentioned systems were used in or developed in conjunction with integrated digital library systems, namely TSIMMIS, Garlic and Information Manifold. Another focus is on the integration of non-cooperative providers, especially those offering Web interfaces. The Unicats project [SS99] integrates digital libraries offering public HTTP interfaces. Other approaches include WebJDBC and techniques for wrapper generation introduced in [SH99]. TSIMMIS, Information Manifold and Garlic also focus on this approach to some extent. 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.

3 Application Requirements

While concepts for the integration of structured data in database systems have proven successful during the last years, in some application scenarios various aspects introduced new challenges resulting from the complexity and diversity of systems to be integrated. As more and more data became available on the Internet the integration of autonomous sources via public HTTP interfaces was considered as one choice. Because of various technical problems and a lack of stability in a changing environment especially the extraction of data still is and probably will remain error-prone. While this 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 based on commercial interests 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. The results are provided by the source as XML conforming to a source-specific DTD.

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. The design can be supported by tools using the source description. This is one of our current research interests. 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. XML is suitable as a transfer format, because it is an emerging standard for data transfer on the Internet and can provide type checking, compaction etc.

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. The global schema can be modified and additional information, for instance about citations, can be added to the schema.

In the following sections we describe our approach that offers a solution based on existing standards. Though it requires a minimum level of cooperativeness its intention is to minimize provider-side efforts by simultaneously using as much existing functionality as possible.

4 Integrating Bibliographical Data

The integrated bibliographical data of digital libraries must be supplied to the users via a uniform interface. For the preparation of such a single point of access to data an appropriate federated system is required. In principle the common architecture of a federated system comprises four layers: an application layer, a federation layer, an adapter layer and a data source layer. On the *data source layer*, all bibliographical data is provided preserving the autonomy of data sources. For the representation of data in a uniform data format special adapters are introduced on the *adapter layer*. With an adapter the data source specific data models are transformed. Part of the query processing and the integration of data from several data sources takes place on the *federation layer*. All relevant functionality is provided to the users in form of an object-relational interface on the *application layer*.

4.1 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. 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.

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. *Object types* describe the structure of objects as sets of attributes and their domains. *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 in the form *g_attr* **`is`** `@`*tbl (l_attr, src, dest, default)*. This means that the database table *tbl* is used for mapping the values from the local attribute *l_attr*. The value of the global attribute *g_attr* is obtained by looking for the value of attribute *l_attr* in column *src* and retrieving the corresponding value of column *dest*. The field *default* denotes a default value, either as literal or as local attribute, which is assigned to the global attribute, if the value of *l_attr* is not found in the 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 [SCS00]. In addition, the standard SQL operations selection and projection are provided, too. FRAQL is implemented as part of a federated query processing system that consists of the following main components: the query parser, the decomposer and the global optimizer, the query evaluator, the Java VM for evaluating user-defined functions, and the catalog. The query processor evaluates a FRAQL query by decomposing it into sub-queries according to the definition of global relations. These sub-queries are sent to the adapters and evaluated by the data source. Finally, the results are combined in the query processor.

4.2 Adapter Layer

Adapters provide a generic interface to a class of heterogenous data sources. Our focus is on the XML-based adapter for cooperative providers. On the one hand, the access to a data source depends on the local query interface and the local query capabilities of the source. Hence, 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. For this transformation process appropriate mapping information is needed, too.

Source Descriptions: We currently consider two kinds of sources. If the source system is a fully-fledged relational database system we can simply pass through the rewritten sub-query. Sources with limited query capabilities include proprietary systems of providers of bibliographical data. 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 queryable attributes of relations in $\mathcal{R}$, the source description $SrcDesc$ can be defined as follows.

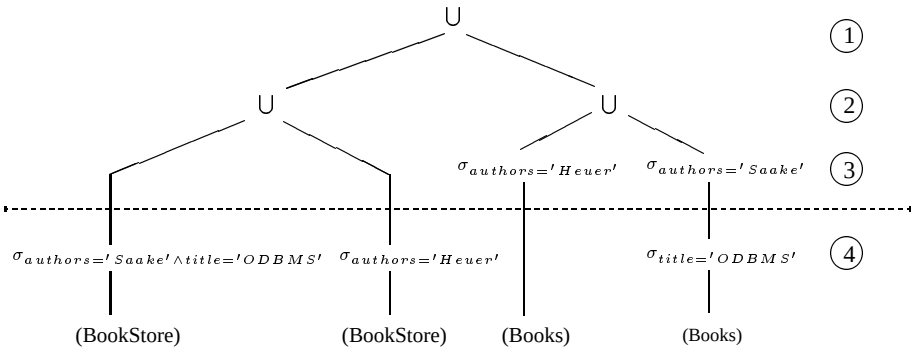

Fig. 1. Re-written query example for source systems

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

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 reconciliations or by mapping functions during result set integration have to be processed afterwards.

Result Transformation: The approach for the generic adapter is based on the assumption that 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 formats are considered. Therefore 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 usage of the general format the introduction of a mapping between the XML conforming to a provider specific DTD and the adapter specific XML format is required. For this purpose we propose the application of XSLT in the adapter. XSLT is a stylesheet language for transforming XML documents into other XML documents. It is used for transforming provider specific XML data into XML data suitable for processing by the adapter. Hence, not only the special XML data but also an appropriate XSLT file is required as an input for the adapter.

For instance, the execution of a simple query on a data source can produce the following result conforming to the provider-specific DTD:

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

For this special document type an appropriate XSLT file is required in order to transform it into an adapter-specific XML document. A special parser capable of handling XSLT searches the input document for special patterns and applies the substitutions described in the templates. Using the XSLT file the following result is created in the adapter:

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

This document can be processed in the generic adapter, which passes it through to the integration layer for further processing.

Adapter Architecture: In Figure 2 the architecture of our generic adapter is shown. In principle, the adapter architecture consists of three parts: a query translator, an XML parser and a result translator. By using a *Query Translator*, requests to the adapter are transformed into specific requests for special data sources and those requests are transmitted. The description of the source capabilities is used here for generating requests. The source description is part of the adapter configuration and is currently stored in a separate XML file. The requests are performed by the data source. The results are made available for the adapter as an XML document. By means of the generic *XML Parser*

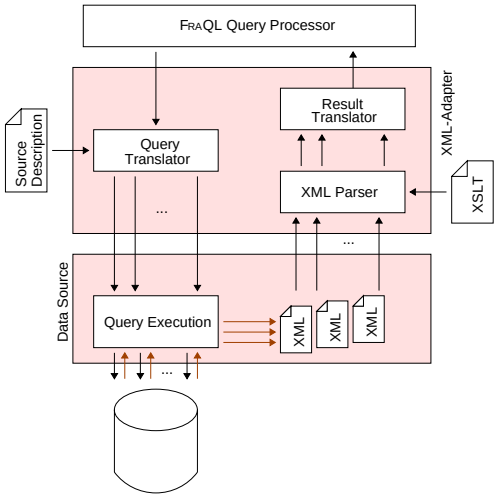

Fig. 2. Architecture of the XML Adapter

and the usage of appropriate XSLT information the special results are transformed into an internal representation. Based on this the results are transformed in compliance with the request in the *Result Translator*.

Using the programming language Java and the ORACLE XML Parser classes, a prototype of the generic adapter was implemented. In this prototype, the approach based on describing the source capabilities and the usage of XSLT for the generic adapter was verified. We are currently investigating optimization issues based on real world scenarios with existing bibliographical data from publishers and electronic catalogues.

5 Citation Linking as an Application Scenario

As a first example as well as for evaluation purposes we have implemented a simple solution for *citation linking*. Usually citations are used within scientific publications in order to link the contents to related work or publications containing basic or more detailed descriptions of mentioned concepts. These citations must be considered part of the meta-data for a given document. This problem is known as citation or reference linking and discussed in more detail in [SES00]. In the following we assume several data sources containing bibliographical data in a relation of the form:

$$Publ(Key, Attr_1, \ldots, Attr_n, \{Ref_1, \ldots, Ref_m\})$$

Here Key denotes a document identifier, $Attr_i$ a bibliographical attribute (author, title, journal etc.) and Ref_j the citation as a document reference. Because of the currently limited capabilities of the FRAQL data model—it lacks support for multi-valued attributes—this relation has to be transformed into normal form. For this purpose, two

object types `PublType` for the “publication” entities and `RefType` for the “references” relationship are defined:

```
create type PublType (
  id varchar(20),
  source varchar(20),
  title varchar(100),
  authors varchar(100));
```

```
create type RefType (
  id varchar(20),
  source varchar(20),
  ref_id varchar(20),
  ref_source varchar(20));
```

In both type definitions the attribute pairs (`id`, `source`) and (`ref_id`, `ref_source`) form the globally unique document identifier. Furthermore, the source relation is transformed into first normal form by the adapter. So, an example relation exported by the adapter may look like relation `src1.Publ` shown in Fig. 3.

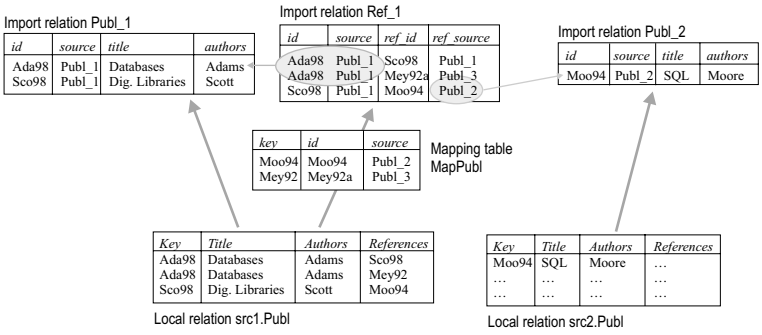

Fig. 3. Citation Linking

Based on these types the FRAQL import relations are specified. First of all, for each source *N* the relation `Publ` is partitioned into a relation `Publ_`*N* of type `PublType` and a relation `Ref_`*N* of type `RefType`. Furthermore, the global attribute `source` of relation `Publ_`*N* is initialized with a literal name identifying the source (in our case the name of the relation `Publ_`*N*):

```
create table Publ_1 of PublType as
  import from src1.Publ (
    id is Key,
    source is 'Publ_1',
    title is Title,
    authors is Authors);
```

In order to map the inter-source references we have to provide a mapping table, which relates the local reference identifiers to the global identifiers. This mapping table is stored in a local component database of the federation and has to be built by hand. For our example table `MapPubl` contains values shown in Fig. 3. It can be used for mapping the reference attributes (`ref_id`, `ref_source`) of the import relation `Ref_1`:

```
create table Ref_1 of RefType as
  import from src_1.Publ (
    id is Key,
    source is 'Publ_1',
    ref_id is @MapPubl (References, key, id, References),
    ref_source is @MapPubl (References, key, source, 'Publ_1'));
```

After importing the source relations into a homogeneous model we are able to integrate them by defining the integration relations `Publications` and `References` as union of all `Publ_`*N* relations and `Ref_`*N* relations, respectively. With these both relations the retrieval of documents referenced by a given document is straightforward:

```
select t2.id, t2.title, t2.authors
from References t1, Publications t2
where t1.id = 'Ada98' and
      t1.source = 'Pub_1' and
      t1.ref_id=t2.id and
      t1.ref_source=t2.source;
```

Obviously, building the mapping table by hand is not an appropriate solution for large databases. However, in a stable scenario as described in section 3 and in combination with tool support for analyzing bibliographical data and their relationships as well as deriving the mapping from these results this approach provides a reliable alternative to heuristic-based automatic mappings.

6 Conclusion and Outlook

We presented our approach for the integration of bibliographical data that employs the FRAQL federation service for loosely-coupled integrated systems. Furthermore, we introduced concepts for adapters that can be used on a common class of source systems and allow a seamless integration with minimal provider-side effort. The adapter is based on XML for data transfer and XSLT for data restructuring, which are suitable in a distributed scenario and keep the transformation process comprehensible. Apart from the XSLT mapping information a query capability description for sources is part of the adapter configuration. In summary, our approach represents 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 and returning the results in XML.

For further research we investigate tool support for implementing provider-side components and for specifying the XSLT mapping. Another focus is on adding efficient mechanisms for the detection of identical objects during result set integration.

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.

[BKLW99] S. Busse, R.-D. Kutsche, U. Leser, and H. Weber. Federated Information Systems: Concepts, Terminology and Architectures. Technical Report Forschungsberichte des Fachbereichs Informatik 99-9, Technische Universität Berlin, 1999.

[CHS+95] M.J. Carey, L.M. Haas, P.M. Schwarz, M. Arya, W.F. Cody, R. Fagin, M. Flickner, A. Luniewski, W. Niblack, D. Petkovic, J. Thomas II, J.H. Williams, and E.L. Wimmers. Towards Heterogeneous Multimedia Information Systems: The Garlic Approach. In Omran A. Bukhres, M. Tamer Özsu, and Ming-Chien Shan, editors, *Proc. RIDE-DOM '95, 5th Int. Workshop on Research Issues in Data Engineering - Distributed Object Management, Taipei, Taiwan*, pages 124–131. IEEE-CS, 1995.

[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, 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.

[HM85] D. Heimbigner and D. McLeod. A Federated Architecture for Information Management. *ACM Transactions on Office Information Systems*, 3(3):253–278, July 1985.

[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*, 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*, 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.

[SC99] B. Schatz and H. Chen. Digital Libraries: Technological Advances and Social Impacts. *Computer*, 32(2):45–50, February 1999.

[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)*, 2000. *To appear*.

[SES00] E. Schallehn, M. Endig, and K.-U. Sattler. Citation Linking in Federated Digital Libraries. In *Proceeding of the 3rd International Workshop on Engineering Federated Information Systems (EFIS'00)*, 2000. *To appear*.

[SH99] K. Sattler and M. Höding. Adapter Generation for Extraction and Querying Data from Web Sources. In *Proc. of 2nd ACM SIGMOD Workshop WebDB'99*, 1999.

[SS99] B. Schmitt and A. Schmidt. METALICA: An Enhanced Meta Search Engine for Literature Catalogs. In *Proceedings of the 2nd Asian Digital Library Conference (ADL'99)*, Taipei, November 1999.

[VZ98] S. Venkataraman and T. Zhang. Heterogeneous Database Query Optimization in DB2 Universal DataJoiner. In A. Gupta, O. Shmueli, and J. Widom, editors, *Proc. of 24rd Int. Conf. on Very Large Data Bases*, pages 685–689. Morgan Kaufmann, 1998.