Integrating Heterogeneous Data Sources into Federated Information Systems – Work in Progress –

Harald Wehr

University of Applied Studies and Research
Friedrichstr. 57-59, D-38855 Wernigerode
hwehr@hs-harz.de

Abstract. Federated Information Systems are based on gathering information from different heterogeneous data sources. The Internet - a Meta FIS in one sense - demands for the ability to query many different information sources. Obviously, most of them are structured very differently. Considering databases, XML files, other structured text files or Web Services as information supplier the complexity of integrating the information with their various describing models is not easy to handle.
Even companies that only want to setup a global information system (e.g. a Data Warehouse) are faced with integration problems. As a lot of departments were able to develop their own local information pools there can be a variety of different sometimes redundant data sources even inside one company.
In this paper an new approach of building such a global information system is described. Besides the theoretical aspects it is mainly shown that the system is really working and can easily be adapted to the special needs, a certain Federated Information System has to fulfill.

1 Introduction

A lot of research has been done in the fields of integrating different heterogeneous data sources during the past years. Besides the integration of (mostly) relational databases it is also desirable to add semi-structured text files for instance XML files or other structured ASCII files. Considering Web Services as XML streams this kind of information sources could also be integrated like simple XML files.

In this paper we present an architecture of a system that provides uniform access to the mentioned kind of data sources. The system is very easy to configure and maintain. Adding additional data sources does not need restarting the system. By writing simple XML files, describing the structure and mapping of a new source, integration is as easy as possible.

The following work has been developed within the MobiHarz project[1]. The project looks for new concepts to increase the attractivity of public transport

[1] The project MobiHarz is part of the global research program "Freizeitverkehr" sponsored by the German Federal Ministry of Education and Research.

in tourism. Therefore, a system has to be developed that can easily integrate different information sources, such as accommodation data of hotel chains, data of certain points of interest and information of available public transport companies.

The paper is structured as follows. The next section contains an overview of existing theoretical approaches of Federated Information Systems. Section 3 describes the overall architecture of the system, while section 4 explains important parts in more detail. After that different related projects are enumerated. The current work is finally concluded in the last section with an overview of the next steps.

2 Federated Information Systems

Federated Information Systems are expected to be a completely new generation of software systems [1]. Their main task is to operate as a global layer over existing data sources. It is important to consider that these sources have certain characteristics making the integration process very difficult: [1][2][3]

Heterogeneity Local data sources are mostly developed for a special purpose. This often results in different solutions for storing information of the same real-world objects. Information can be stored in databases with different models (relational, object oriented), ASCII files or be available as Web Services. It is obviously that these kinds of sources are accessed through different interfaces, protocols and languages (*Syntactical Heterogeneity*). Even the same data model can cause mapping conflicts due to different understandings of the real world (*Logical Heterogeneity*).

Autonomy Data sources, integrated in a FIS do not give up their autonomy. First of all they keep their *Design autonomy*. It's up to them how the contained information is stored. Furthermore they are able to decide which other systems are allowed to communicate with them. The local data source should be able to enter or leave the federation at any time (*Communication autonomy*). Additionally each component is independent in deciding how the incoming requests are scheduled and executed. The federated system must not change execution priorities for its requests (*Execution autonomy*).

Distribution Sources that have to be integrated do not always reside on the same host. It's likely to be that they are on different hardware platforms and operating systems and can only be accessed through certain network protocols.

Based on [1][2] FIS can be divided into three different types:

- *Loosely coupled information systems* do not have a federated schema. They only provide a query language to access the different local sources. By implementing this kind of system the user is responsible for addressing the right information source with the specific desired elements.

– FIS that only contain database components are called *federated database systems (FDBS)*. A lot of research has been done in these kinds of integration systems. For a detailed overview see [4].
– *Mediator-based information systems* are not limited to certain kinds of components. The system is able to "mediate" between the user and structured as well as unstructured information sources.

According to the goals of our project our system architecture can be classified as a mediator-based system. It is not possible to limit the potential data sources to databases. Furthermore it is required to add or remove other components without changing the federated schema.

Building a FIS can be done by two different approaches [5]. The so-called *bottom-up strategy* is often used for developing a FDBS. Every component is completely integrated into the federated schema. This procedure often leads to a very complex global schema. Adding a new source always leeds to a new and bigger global system. The *top-down strategy* is mainly used for mediator-based systems. It does not provide a complete and exact access to all information of the integrated sources. The system is built for a specific information need it has to fulfill. Schemas of the local components do not affect building issues of the federated schema. They will only be considered during the step of mapping elements of data sources to elements of the global schema. By following this approach there is no need for altering the federated schema. Additional sources can be integrated without any effects to the federated model. Of course it has to be considered that the global schema can't be as complete as the one developed with the *bottom-up strategy*.

3 System Architecture

An overview of the general architecture of MobiHarz is depicted in figure 1.

As mentioned above the project provides a homogeneous interface to tourists hiding the underlying heterogeneous information sources. Several output formats like HTML, WML or PDF have to be offered as different Internet enabled client devices shall be able to obtain information from the federated system [6]. The content management system is therefore responsible for generating dynamic content in the desired output formats based on the business objects residing in the federated system. All business objects are kept in random access memory (RAM). A thread with lower priority builds a complete new object system at regular intervals that replaces the old one. This procedure entails several advantages:

– The response times of the system are very small as the underlying sources are not queried at each incoming request.
– If a source is temporarily unavailable its information can still be brought to the customer.
– Additional data sources can be integrated without restarting the system [9].

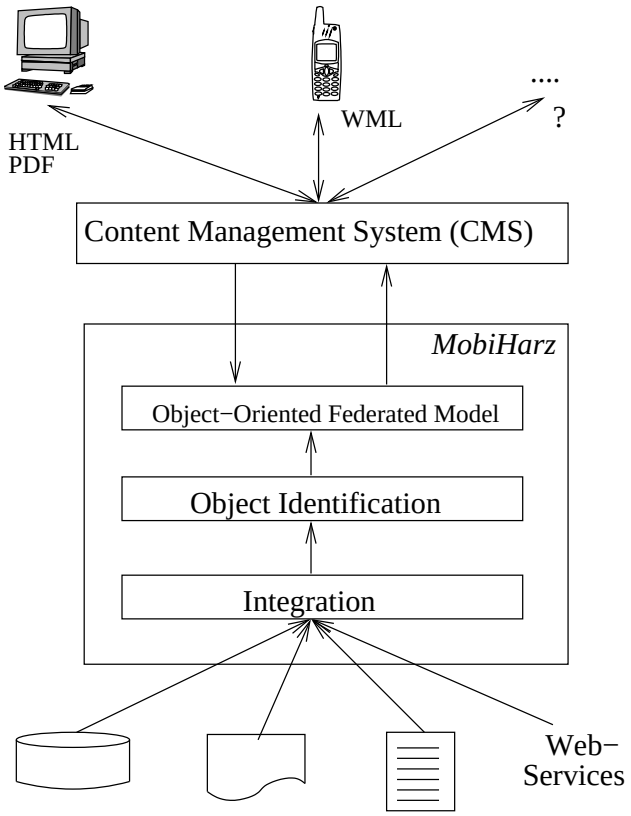


Fig. 1. Architecture of the MobiHarz system

Besides these advantages it is clear, that very big sources cannot be easily integrated due to limited RAM. This problem could be solved with an additional database that consequently fits to the developed business objects and temporarily stores the integrated information as it is done in data warehouses.

In the following section the process of integrating different sources is described in more detail as this step has already been implemented in our FIS.

4 Integrating the Sources

Integration of heterogeneous data sources causes several conflicts that have to be resolved. They can be divided into *semantic conflicts*, *descriptive conflicts*, *heterogeneous conflicts* and *structural conflicts* [7][8]. Solving these conflicts the MobiHarz project uses a 'semi-automatic' approach for the process of integration [10]. Each source has to be described in a simple configuration file containing

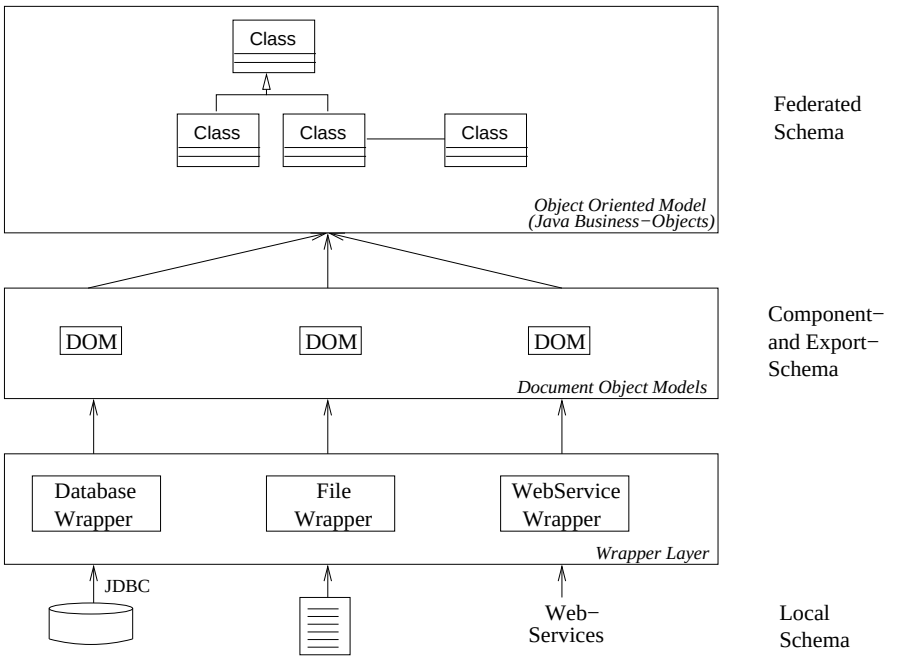


Fig. 2. Integration of different data sources

field names, data types and optional primary and foreign keys. A second file is responsible for the mapping between one source and the project's target business model. Figure 2 shows the way of integrating different kinds of sources.

Transforming the Sources

According to the five-level schema architecture of an FDBS ([2]) each source is transformed into the component schema by certain wrappers. One source and its information is hold in one or more Document Object Model (DOM) [11]. They represent the so-called *canonical* or *common data model* as the starting point for generating the corresponding Java objects. The different sources are transformed in the following ways:

Databases All relational databases that provide a "Java Database Connectivity" (JDBC) driver can be integrated into the system. Each relation in a database is added to the *Document* of the source. Every row is appended as further *Element*. Elements can reference other elements to reflect relationships between the former tables. As almost all database vendors develop the mentioned JDBC driver MobiHarz is able to handle a large number of different relational databases.

Text Files There is no common rule how many *Documents* are produced from a certain text file. To simplify matters each XML file could result in only one document. Other files could of course be represented by several documents. It is possible to write additional wrappers for files that cannot be transformed with the default ones provided by the MobiHarz' system.

Web Services As Web Services deliver XML over HTTP they can be transformed as easy as normal XML files into one *Document*.

It is important to consider that the process of describing the structure of one source (relations, fields, data types) often can be automated. JDBC drivers are able to provide meta data about the connected database. Document Type Definitions (DTD) inform about the structure of a certain XML file. Even Web Services have their own *Web Service Description Language (WSDL)* that gives further information about the structure of the content.

The Business Model

As it is intended to use the federated system not only for touristic concerns the mapping into certain Java business objects must not be hard coded. By using the federated system the administrator can provide his or her desired business classes. Each class or one of its super classes must inherit from the abstract `DefaultBusinessClass` in order to provide all objects with the following required attributes:

Source: the source from which a certain object has been instantiated,

LocalObjectId: an Id that clearly identifies an object within its source (sometimes optional),

AttributesFillable: information whether certain attributes could be filled by the source or not,

Instances: class variable that holds all instances coming from one source.

Furthermore it is important to create the appropriate relationships between the business objects. Unary as well as binary OneToOne-, OneToMany-, ManyToOne- and ManyToMany-relations should be supported. In order to build a required relationship the business classes have to follow these restrictions:

- Objects on the "One"-side of a relationship must have an attribute whose type is of the class of the other participating object.
- Objects on the "Many"-side of a relationship must have an attribute whose type implements the interface `java.util.List`. This container holds the references to the participating objects in the relationship. Certain implemented mapping actions (see next section for details) make sure that the attributes are filled correctly.

Integrating the sources

As mentioned above every data source must be provided with a mapping file that describes how certain fields are used to build the attributes in the target business

objects. This mapping-file is again a simple XML-file. Neither the mapping of the Business-Objects nor their attributes are hard-coded in Java. As we strongly use the reflection API the mapping process can be described and easily altered by writing simple text files.

At this point of integration it is not longer to be seen which kind of source holds the information. As the mapping process takes places at the export schema (see figure 2) the original structure of the integrated source is hidden. Certain already implemented *actions* make sure that the listed attributes can be set correctly from different sources with different structures. In addition the user is able to write own actions that belong to certain sources. Thus it should be possible to integrate even those sources whose structure completely differs from the structure of the federated schema.

Following steps have to be done for integrating a new source:

1. First of all a new source must be introduced to the federated system. The file `sources.xml` contains all available sources with their names, responsible wrapper classes, access information, locations and so on.
2. According to the component schema, information of the source have to be transformed into documents. The name of the wrapper class and the corresponding implemented class must be available. Wrappers for JDBC databases and XML files already exist in the federated system and can be used by their names and Java's dynamic class loading. Obviously, some sources can't be transformed by the supplied standard wrappers. In this case the user must implement a new one for the desired source.
3. Additionally each source must be provided with two configuration files. The first one contains information about the fields and data types of the component schema. The second one informs about the mapping actions that have to be executed. In some cases it may be necessary to implement further actions for a source that can't be mapped by the default ones.

Having done these steps for a new source during the next 'refresh' of the system all sources (including the new one) will be integrated. New objects will be instantiated and have to pass through the way of the identification process as depicted in figure 1. Afterwards a set of objects is waiting for requests from the content management system.

5 Related Projects

In the last years a lot of research has been done in the fields of federated information systems. Three different projects shall be described here in further details.

– **TSIMMIS (The Stanford-IBM Manager of Multiple Information Sources)**
  Following the descriptions of the project integration of heterogeneoues data sources is possible. As global federated schema the developed 'object exchange model' is used. This model is able to represent different objects that

can contain further objetcs. Inheritance is not possible within this model [12]. The successor of TSIMMIS is called MIX. It is completely based on XML as global schema. Queries against this schema are done by using the XMAS-language [13].

- **Information Manifold**
  This prototype has been developed by AT&T in 1995/96. The federated model is mainly a relational one provided with some extensions. Like TSIMMIS the federation is virtual. Every time a user accesses the system it identifies relevant sources and executes sub-queries to them. After that the user is responsible for cleaning overlapping information or performing object fusion mechanisms [14].
- **SIMS (Search in Multiple Sources)**
  The federated schema in this project uses the hierarchical terminological knowledge base (Loom). It consists of different nodes representing each class of objects and relationships between them. An incoming user query is translated by certain mapping descriptions in a corresponding local query. Again a mechanism of avoiding overlapping information is missing [15].

6 Conclusions

The outlined architecture satisfies almost all requirements of a federated information system. As printed in the title the described work is still in progress. The depicted architecture is the result of a former prototype shown in [6]. Besides the integration of different kinds of data sources it now offers a more flexible way of extending the system.

Our federated system currently provides following features:

- Several heterogeneous data sources can be easily integrated, updated or just removed from the global information system by simply changing text files.
- A large amount of available databases, structured text files and Web Services are supported due to already available wrappers. Of course it is possible to write own wrappers that import other currently not supported data sources.
- The mapping process is done by certain mapping actions. The system contains already implemented actions. Again, the mechanism of dynamic classloading enables the user to write his own actions for special kind of data sources with an uncommon structure that does not fit into the global business model.
- Due to the JAVA Reflection-API the object-oriented federated model can be provided by the user. It is possible to use this system in nearly every environment where integration is needed.

In comparison with the related projects our current prototype uses the so-called 'materialized integration'. On the federation level the content of the local data sources is completely mirrowed in business objects. Following this way of integration update procedures have to be developed. Another disadvantage of the current approach is that all the information are kept in main memory. Of course

this can be changed by installing an object-oriented database at the federated level. According to our project needs this is not necessary at the moment.

This paper mainly refers to the problem of integrating different sources. After this step the conflicts of object identification have to be solved. As the sources may contain overlapping information several instantiated objects may refer to the same "real-world" object. Furthermore the business objects must provide a common interface to the mentioned content management system to support different output formats. These tasks still have to be done in future.

Acknowledgement:
I would like to thank Bernd Müller for a lot of useful comments on this paper and the work in the project MobiHarz.

References

1. Susanne Busse, Ralf-Detlef Kutsche, Ulf Leser, and Herbert Weber. Federated Information Systems: Concepts, Terminology and Architectures. Technical Report 99-9, Forschungsberichte des Fachbereichs Informatik, 1999.
2. A. Sheth and J. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing, Springer Verlag (Heidelberg, FRG and NewYork NY, USA)-Verlag Surveys, ; ACM CR 9107-0567*, 1990.
3. M. Tamer Ozsu and Patrick Valduriez. Distributed database systems: Where are we now? *IEEE Computer*, 24(8):68–78, 1991.
4. S. Conrad. *Föderierte Datenbanksysteme: Konzepte der Datenintegration.* Springer-Verlag, Berlin/Heidelberg, 1997.
5. W. Hasselbring. Top-Down vs. Bottom-Up Engineering of Federated Information Systems. In S. Conrad, W. Hasselbring, and G. Saake, editors, *Proc. 2nd Int. Workshop on Engineering Federated Information Systems EFIS'99, Kühlungsborn, Germany, May 5–7, 1999*, pages 131–138. infix-Verlag, Sankt Augustin, 1999.
6. Bernd Müller and Harald Wehr. Uniform Information Processing with Heterogeneous Data Sources. In *5th International Conference on Business Information Systems BIS 2002*, Poznan, Poland, 2002. Witold Abramowicz (Ed.).
7. S. Conrad, M. Höding, G. Saake, I. Schmitt, and C. Türker. Schema Integration with Integrity Constraints. In C. Small, P. Douglas, R. Johnson, P. King, and N. Martin, editors, *Advances in Databases, 15th British National Conf. on Databases, BNCOD 15, London, UK, July 1997*, volume 1271, pages 200–214, Berlin, 1997. Springer-Verlag.
8. Stefano Spaccapietra, Christine Parent, and Yann Dupont. Model Independent Assertions for Integration of Heterogeneous Schemas. *VLDB Journal: Very Large Data Bases*, 1(1):81–126, 1992.
9. Harald Wehr and Bernd Müller. Providing Uniform Access to Heterogeneous Data Sources - A Positional Paper. To appear in: *International Conference on Internet Computing* June 2002, Las Vegas.
10. E. Rahm and P. Bernstein. On matching schemas automatically. Technical Report MSR-TR-2001-17, Microsoft Research, Redmon, WA., 2001.
11. W3C Document Object Model: http://www.w3.org/DOM/.

12. Yannis Papakonstantinou, Hector Garcia-Molina, and Jennifer Widom. Object exchange across heterogeneous information sources. In P. S. Yu and A. L. P. Chen, editors, *11th Conference on Data Engineering*, pages 251–260, Taipei, Taiwan, 1995. IEEE Computer Society.
13. C. Baru, A. Gupta, B. Ludascher, G. Marciano, Y. Papakonstantinou, P. Velikhov, and A. Yannakopulos. Xml-based information mediation with vamp.
14. Alon Y. Levy, Divesh Srivastava, and Thomas Kirk. Data model and query evaluation in global information systems. *Journal of Intelligent Information Systems - Special Issue on Networked Information Discovery and Retrieval*, 5(2):121–143, 1995.
15. Yigal Arens, Chun-Nan Hsu, and Craig A. Knoblock. Query processing in the SIMS information mediator. In Michael N. Huhns and Munindar P. Singh, editors, *Readings in Agents*, pages 82–90. Morgan Kaufmann, San Francisco, CA, USA, 1997.