# Tool Support for the Design of Database Federations in SIGMA$_{FDB}$[1]

Kerstin Schwarz Ingo Schmitt Can Türker Michael Höding
Eyk Hildebrandt Sören Balko Stefan Conrad Gunter Saake

Otto-von-Guericke-Universität Magdeburg
Institut für Technische Informationssysteme
Postfach 4120, D–39016 Magdeburg, Germany
E-mail: {sigmafdb}@iti.cs.uni-magdeburg.de

June 1999

[1]This research was partially supported by the German Federal State Sachsen-Anhalt under FKZ: 1987A/0025 and 1987/2527R, the ESPRIT Working Group 23531 "Fireworks - Feature Integration in Requirements Engineering", and the ESPRIT Working Group 22704 "ASPIRE - Advanced modeling and SPecification of distributed InfoRmation systEms".

## Abstract

Federated database systems provide a homogeneous interface to possibly heterogeneous local database systems. This homogeneous interface consists of a global schema which result in a logical integration of the schemata of the corresponding local database systems and file systems. The **SIGMA**$_{FDB}$ project offers an approach to the design of federated database systems. In this paper, we sketch the integration process and describe a set of tools supporting the design process. Besides the classical database schema integration, the design process for federated information systems requires the integration of additional aspects like integrity rules, authorization policies and transactional processes. This paper reports on an integrated approach to tool support covering several of these integration aspects. The different integration facets are linked together via the central databases integration formalism GIM allowing a high degree of an automatic integration.

# Contents

# Chapter 1

# Introduction

The **SIGMA**$_{FDB}$ project is the focus of our research in the field of federated database systems [SL90, Con97, SCS99]. The acronym **SIGMA**$_{FDB}$[1] represents the two basic topics we have been working on since the very beginning of the project: schema integration and global integrity maintenance. The **SIGMA**$_{FDB}$ project officially started in 1995. In the meantime several further aspects of federated database systems, like file integration and security in database federations, have come into the scope of our research activities.

*Federated database systems* [SL90] provide a homogeneous interface to possibly heterogeneous local database systems. This homogeneous interface consists of a global schema which is the result of a logical integration of the schemata of the corresponding local database systems [BLN86]. Thus, our common view on federated database systems is a tightly coupled approach. This is motivated by the requirement of data consistency which can only be appropriately guaranteed in a tightly coupled environment. A federated database should correctly reflect the modeled real-world. Inconsistencies between the databases concerning to same real-world entities have to be removed. Correct conflict resolution builds the basis for a correct database integration.

Our research interests as well as the related implementation activities are stimulated by the following real application scenarios:

- During the first two years of the project, we focused on database integration scenarios in factory planning. This was stimulated by a close cooperation with engineers from the logistics and factory planning groups of our university and the Fraunhofer Institute in Magdeburg. In this field, different specific application programs (tools) are used for optimizing machine configuration and transport facilities in factories. These applications have been developed independently.

  Although some kind of system-based interoperability is obviously necessary, interoperability is currently performed manually. In most cases, interoperation is uni-

[1] **SIGMA**$_{FDB}$ (**S***chema* **I***ntegration and* **G***lobal integrity* **M***aintenance* **A***pproach for* **F***ederated* **D***ata* **B***ases*)

directional and primarily based on file-exchanges because most tools store their data in files using its own way of structuring the data. In consequence, data inconsistencies may occur which cause additional work for the factory designer. The result of analyzing the systems is a formal description of the used file formats in SGML and an integrated schema. Beside this, we developed a "hard-coded" demonstration prototype for illustrating problems and features of database integration using federated database systems.

- Our second integration scenario was motivated by joint work with our partners in the bio-informatics research group of our department. In this field, we are again faced with highly heterogeneous databases which contain bio-molecular data representing results of thirty years of experimental research. The integration of these partly overlapping databases promises a new quality of databases and enables new applications to access heterogeneous data in a uniform and transparent way.

  There are two main aspects being very specific for this scenario:

  - Firstly, most databases are based on files and often provide a WWW interface.
  - Secondly, only read access is possible.

  In this context, local (data) autonomy is important and cannot be relaxed. A more detailed discussion about the application of FDBS in the area of bio-molecular databases and a description of the implementation of the prototype **BioBench** can be found in [HHSS98]. The main focus of our research here is on methods for file integration.

- Our third (and most recent) integration scenario is given by the *Global-Info* project (funded by the German Ministry for Education and Research, BMBF) which aims at supporting digital libraries by federation services. As in the scenarios before, different information systems are available, mainly with WWW interfaces which usually restrict the access to pieces of information in a certain way. These systems provide related and often overlapping information.

  At the time being, a user has to access some or many information systems using his personal knowledge to find the correct (and needed) information. For example, even in the database research community, it is not trivial to assign the correct real-world person to the string "M. Scholl". Another problem concerns duplicate representation which is very hard to detect. For that, a federation layer can support the user in his work by offering services and meta-data. For instance, the Dublin Core [WL97] provides a standard for meta-data on publication and, thereby, allows a better exchange of concrete data.

An important aspect in conflict resolution between two or several information sources is to consider both intensional and extensional conflicts in a common framework. The GIM-Method [SS98] allows to deal with n-ary extensional assertions. Binary extensional assertions on which most other schema integration approaches are based cannot express all possible situations. Furthermore, the GIM-Method provides several algorithm to integrate different schemata. Since local integrity constraints determine the semantics of the local schemata, local integrity constraints have to be reflected on the global level. Of course, this approach contradicts the requirement of complete local autonomy. However, from our point of view it is the only way to provide a correct database to users. Even local database users take advantage of a correct database since the data quality will be improved. Moreover, considering local integrity constraints during schema integration can yield important hints for extensional assertions between local classes [TS98a].

A further step of the design process considers local integrity constraints as additional semantic information for schema integration. In addition, local access rights must also be considered in order to derive global access rights. Last but not least, we regard the design of global transactions and their decomposition into subtransactions as an important task which has to be supported for building federated information systems.

Federations can build the basis for cooperative systems. In this context, cooperation means collaboration to achieve a common goal. This was also proven by the first example scenario where consistency and cooperation are the main requirements. On the other hand, we found application scenarios where mainly reading access is possible (and necessary), e.g. in bio-informatics. In this case, a full-fledged FDBS seems to be oversized and/or impractical.

The remainder of this report is organized as follows. Chapter 2 briefly introduces GIM, the *Generic Integration Model*, which is the basis of our schema integration method.

The integration of integrity constraints is discussed in Chapter 3. Surprisingly, nearly all integration approaches proposed so far disregard explicit integrity constraints in schema integration, although explicit integrity constraints are part of database schemata. Our approach is based on general logical rules and can therefore easily be adapted to enhance other schema integration methods as well.

The fact that in a lot of application areas data is not stored and managed by fully-fledged database management systems imposes additional problems in building a database federation. In Chapter 4, we therefore discuss file integration in detail as an important task in design of federated systems.

A crucial aspect of databases is the management of confidential data. Of course, this also holds for federated databases. Therefore, Chapter 5 discusses the problem of authentication and authorization in federated database systems. Especially, our approach to the integration of access rights is presented.

The design process of federated databases also concerns global transactions. Naturally, the decomposition of global transactions depends on the result of the design process, the integrated schema. However, extensional assertions also influence the termination dependencies between the transactions executed on the local systems. This aspect is considered in Chapter 6.

Finally, the current functionality of the toolset $\mathbf{SIGMA}_{Bench}$is summarized in Chapter 7. This toolset supports the federated database designer in dealing with the aspects discussed so far.

# Chapter 2

# Schema Integration

Schema integration [BLN86] is the main step in federated database design. In correspondence to the five-level-schema architecture [SL90] the schemata of existing component databases must be integrated to the federated schema. Schema integration has to deal with heterogeneity on database model and on schema level. Tackling heterogeneity on database model level means to transform the given schemata into a common data model. More complex is the task to overcome schema heterogeneity. This step typically consists of detecting semantic conflicts and then of solving these conflicts by designing new schemata and corresponding schema mappings. When all conflicts are solved the schemata can be merged to a federated schema.

In our project we developed the new schema integration method *GIM-Method* (cf. [Sch98]). GIM is an acronym for *Generic Integration Model*, an intermediate data model. This model is designed for schema integration and does not restrict the database model of the federated schema. The GIM approach enables the usage of algorithms from formal concept analysis [SS98] increasing the part of schema integration which can be done automatically.

In the following we will use an example to sketch the GIM-Method. The schemata of two databases have to be integrated. One database is a library database storing information about publications, where books and journal papers are special types of publications. Each publication is either a book or a journal paper. This database has to be integrated with a project database which stores project publications, too. This database, however, distinguishes more types of publications. In contrast to journal papers, we assume that books and technical reports are non-refereed publications. `Book` and `Techn.-Rep.` are subclasses of the class `Non-Refereed`. Each object of the class `Publication` is an object of at least one of its subclasses. The corresponding schemata of both databases are given in Fig. 2.1.

Before we can integrate both schemata, we have to detect conflicts if they exist. In this paper we consider only extensional and intensional conflicts. We assume that other types of conflicts are resolved before as described for example in [NG82, LNE89, PBE95, LL95, YP95, Sch98].

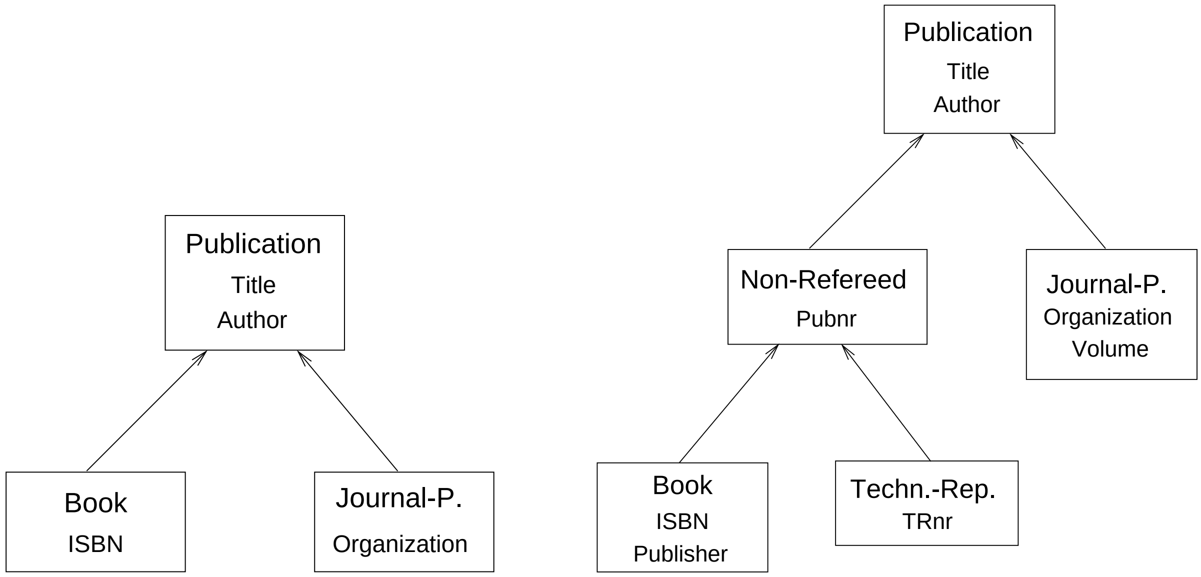


Figure 2.1: Library and Project database schema

Extensional conflicts correspond to the potential of redundancy among different classes whereas intensional conflicts are caused by different types of semantical related classes. Unfortunately, the information about extensional overlappings cannot be sufficiently derived from the given hierarchies. The schemata define only subset relationships between sub- and superclasses. It is impossible to conclude from given databases whether the extensions of an arbitrary set of classes from different databases can simultaneously contain objects modeling the same real-world object. Therefore, the designer has to give additional information about the extensional relationships among the classes. There are many approaches, e.g. [SP91, Dup94, LNE89], which help the designer to formulate such extensional relationships. In [ST98] we describe an iterative approach to find and formulate extensional relationships. This approach is implemented in the tool $\mathbf{SIGMA}_{Bench}$(see Chapter 7).

The designer has to specify extensional assertions. In general, between two classes $C_1$ and $C_2$ the following extensional assertions [SPD92] can be defined:

$$\begin{aligned}
\textit{Disjointness}\colon C_1 \varnothing C_2 &:\Leftrightarrow \forall S\colon Ext^S_{C_1} \cap Ext^S_{C_2} = \emptyset \\
\textit{Equivalence}\colon C_1 \equiv C_2 &:\Leftrightarrow \forall S\colon Ext^S_{C_1} = Ext^S_{C_2} \\
\textit{Containment}\colon C_1 \supset C_2 &:\Leftrightarrow \forall S\colon Ext^S_{C_1} \supseteq Ext^S_{C_2} \\
\textit{Overlap}\colon C_1 \Cap C_2 &:\Leftrightarrow \textit{true}
\end{aligned}$$

Two classes are extensionally disjoint, if there are no common objects in the extension of these classes in any database state. Extensionally equivalent classes have the same extensions in each database state. One class extensionally contains another class, if the extension of the first class is a superset of the extension of the other class in any database state. Finally, in case of overlap there are no restrictions on the extensions of the two related classes. Same extensions means here that the extensions model the same set of real-world objects.

For our example, the following assertions are specified:

$$\text{Lib.Book} \;\varnothing\; \text{Lib.Journal-P.} \tag{2.1}$$
$$(\text{Lib.Boo}k \cup \text{Lib.Journal-P.}) \;\equiv\; \text{Lib.Publication} \tag{2.2}$$

$$(\text{Proj.Non-Refereed} \cup \text{Proj.Journal-P.}) \;\equiv\; \text{Proj.Publication} \tag{2.3}$$
$$\text{Proj.Book} \;\varnothing\; \text{Proj.Techn.-Rep.} \tag{2.4}$$
$$(\text{Proj.Jounal-P.} \cap \text{Proj.Non-Refereed}) \;\subset\; \text{Proj.Techn-Rep.} \tag{2.5}$$
$$(\text{Proj.Book} \cup \text{Proj.Techn-Rep.}) \;\subset\; \text{Proj.Non-Refereed} \tag{2.6}$$

$$\text{Lib.Publication} \;\varnothing\; \text{Proj.Techn.-Rep.} \tag{2.7}$$
$$\text{Lib.Book} \;\varnothing\; \text{Proj.Journal-P} \tag{2.8}$$
$$\text{Lib.Journal-P.} \;\varnothing\; \text{Proj.Non-Refereed} \tag{2.9}$$
$$(\text{Proj.Journal-P.} \setminus \text{Proj.Techn.-Rep.}) \;\Cap\; \text{Lib.Journal-P.} \tag{2.10}$$
$$\text{Lib.Book} \;\Cap\; \text{Proj.Book} \tag{2.11}$$

The extensions of the classes `Publication` of both databases overlap. Technical reports are stored in the `Project` database only (2.7). Some publications stored in the `Project` database are published simultaneously as a journal paper and a technical report (2.5). Because there is no common subclass of the classes `Techn.-Rep.` and `Journal-P.`, the data model of the `Project` database has to support the role concept. That is to say, an object can be an instance of more than one class simultaneously.

A journal paper stored in the library database can be simultaneously stored only in the class extension of the class `P.Journal-P.` (2.10) and a library book can only be stored either in the class `L.Book`, in the class `P.Book`, or in both classes simultaneously (2.11).

Knowing the extensional overlappings the original extensions can be decomposed into disjoint base extensions. If, for example, the classes `A` and `B` overlap then their extensions are decomposed into three base extensions ($A \setminus B, B \setminus A, A \cap B$). The algorithm presented in [ST98] computes the base extensions from the extensional assertions and is implemented in the $\textbf{SIGMA}_{Bench}$.

The base extensions of our example are given in Tab. 2.1. Based on Tab. 2.1 the original extensions are partitioned into base extensions denoted by `1` to `9`. For example, the `Book` extension of the library database is partitioned into the two base extensions `1` and `3`. The base extension `3` contains books from the library database which are simultaneously stored in the project database whereas the base extension `1` contains the remaining ones. Each publication which is stored in the library or in the project database, can be assigned to exactly one base extension.

| Base ext. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| **Lib.Publication** | √ | √ | √ | √ | | | | | |
| **Lib.Book** | √ | | √ | | | | | | |
| **Lib.Journal-P.** | | √ | | √ | | | | | |
| **Pib.Publication** | | | √ | √ | √ | √ | √ | √ | √ |
| **Proj.Journal-P.** | | | | √ | | √ | √ | | |
| **Proj.Non-Refereed** | | | √ | | √ | | √ | √ | √ |
| **Proj.Book** | | | √ | | √ | | | | |
| **Proj.Techn.-Rep.** | | | | | | | √ | √ | |

Table 2.1: Extensional overlappings

Now we have to compare the intensional aspects of the schemata to be merged. For simplicity, we assume attributes with same names have identical semantics. With other words, *attribute conflicts* [LNE89] are already resolved. The resolution of attribute conflicts is in general an important step during the integration process but outside the scope of this paper.

From extensional and intensional information (Tab. 2.1) we are able to derive automatically a table, which assigns attributes to base extensions. A tick symbol for a column (base extension) and a row (attribute) means, that for potential objects of that base extension we would know the value of that attribute (Tab. 2.2).

Table 2.2: Extension-attribute relation

| Attr-Ext | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| **Title** | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| **Author** | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| **ISBN** | √ | | √ | | √ | | | | |
| **Organ.** | | √ | | √ | | √ | √ | | |
| **Volume** | | | | √ | | √ | √ | | |
| **Publisher** | | | √ | | √ | | | | |
| **TRnr** | | | | | | | √ | √ | |
| **Pubnr** | | | √ | | √ | | √ | √ | √ |

This table representation is a simplified GIM representation of the integrated schemata and can be directly used for further analyses. The order of columns and rows is irrelevant: we can exchange rows and columns. We modify the representation only. In such a representation a rectangle corresponds to a class: a union of base extensions having the same set of defined attributes.

Moreover, we can detect subclass relationships using the rectangle representation of classes. If rectangles overlap as shown in Fig. 2.2, then they are in a specialization relationship. In Tab. 2.2 we find, for instance, the class (rectangle) `A` with the extension $\{6, 7\}$ and attributes {`Organ.`, `Volume`}. Another class (rectangle) `B` has the extension

$\{7\}$ and attributes {`Organ.`, `Volume`, `TRnr`, `Pubnr`}. The class `B` is a subclass of class `A`.

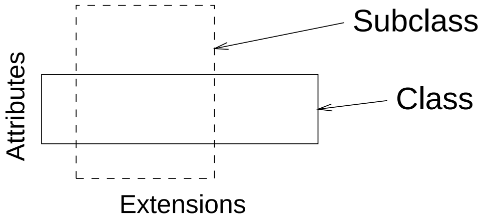


Figure 2.2: Class-subclass relation as overlapping rectangles

The detection of classes and subclass relationships using rectangles rely on the chosen row and column order (cf. [SS96]). Therefore, it is necessary to develop algorithms which perform this task independently of these sequences. Here, the algorithms of formal concept analysis (cf. [GW98]) come into play. They help to find all maximal rectangles and extensional subset relations between them. A maximal rectangle represents a class of the federated schema.

The algorithm presented in [SS98], which is a slightly modification of the concept analysis algorithm, finds the following classes (maximal rectangles) from the Tab. 2.2:

C1 = $(\{1, 2, 3, 4, 5, 6, 7, 8, 9\}, \{$Title,Author$\})$
C2 = $(\{1, 3, 5\}, \{$Title,Author,ISBN$\})$
C3 = $(\{2, 4, 6, 7\}, \{$Title,Author,Organ.$\})$
C4 = $(\{4, 6, 7\}, \{$Title,Author,Organ.,Volume$\})$
C5 = $(\{3, 5\}, \{$Title,Author,ISBN,Publisher,Pubnr$\})$
C6 = $(\{7, 8\}, \{$Title,Author,TRnr,Pubnr$\})$
C7 = $(\{3, 5, 7, 8, 9\}, \{$Title,Author,Pubnr$\})$
C8 = $(\{7\}, \{$Title,Author,Organ.,Volume,TRnr,Pubnr$\})$

| $<$ | **C1** | **C2** | **C3** | **C4** | **C5** | **C6** | **C7** | **C8** |
|---|---|---|---|---|---|---|---|---|
| **C1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **C2** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **C3** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **C4** | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **C5** | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| **C6** | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| **C7** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **C8** | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |

Table 2.3: Specialization relation matrix M

Now we have to build the matrix $M$ (Tab. 2.3), which represents the irreflexive binary relation $<$ defined on the generated class extensions expressing the specialization relation (subset relation) by comparing each pair of classes. Two classes are in specialization relation if their sets of base extensions are in a subset relation.

| $<$ | **C1** | **C2** | **C3** | **C4** | **C5** | **C6** | **C7** | **C8** |
|---|---|---|---|---|---|---|---|---|
| **C1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **C2** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **C3** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **C4** | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **C5** | -1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| **C6** | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| **C7** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **C8** | -3 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |

Table 2.4: Specialization relation without transitive specializations: matrix $\mathrm{M}_N$

The computation $M_N = M - M \times M$ (cf. Tab. 2.4) removes transitive specializations. Only the value '1' in a field of $M_N$ represents a direct specialization relation between two classes.

The classes C1, ..., C8 and the Tab. 2.4 can be directly used to produce the federated schema as depicted in Fig. 2.3. The names of the integrated classes correspond to the classes of the two existing databases.

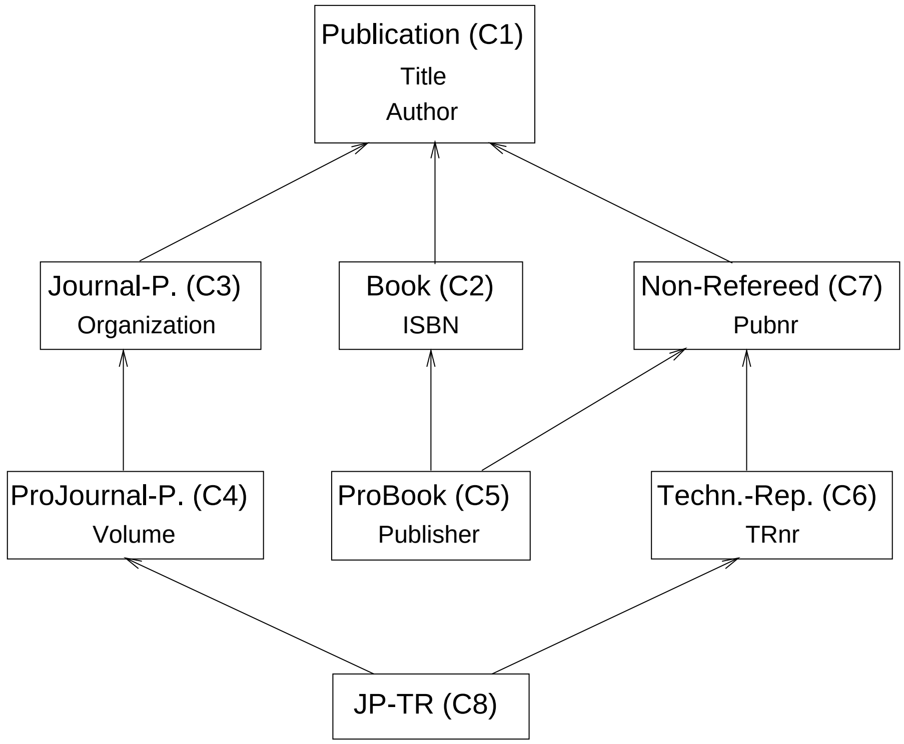


Figure 2.3: Integrated schema

# Chapter 3

# Integration of Integrity Constraints

Integrity constraints are an important part of database schemata. They describe properties of the modeled real world and thus define the semantics of a database schema. Their enforcement ensures the semantic correctness of the corresponding database states. In order to build and maintain semantically correct database federations, locally existing integrity constraints must be reflected in the global schema.

In this context, the notion of *global* and *local understandability* [TS98b, TS99a] play a central role. Global/local understandability means that a global/local transaction should not be rejected if it fulfills all integrity constraints of the corresponding schema. Let us consider the example depicted in Figure 3.1. There, we see a simple integration scenario where two extensionally equivalent classes are integrated into a common class in the global schema. The global insertion is rejected because the insertion does not fulfills the local integrity constraints. However, since the global user only sees the global schema, he cannot understand why the operation was rejected. From the global point of view, all global (or globally visible) integrity constraints are satisfied. The problem of global understandability is caused by an *information capacity augmenting* schema integration. Here, the information capacity [Hul86, MIR93] of the global schema is greater than the union of the information capacity of the local schemas, i.e., there are some global database states that cannot be mapped onto corresponding local database states. This problem occurs when existing local integrity constraints are not reflected in the global schema. Unfortunately, this is the usual case since most of the schema integration approaches do not consider integrity constraints at all. There are first integration approaches dealing with certain kinds of integrity constraints, e.g. [RPG95, AQFG95, VA96, RR97].

Analogously to the global insertion, the local insertion is rejected although it fulfills the integrity constraints of the corresponding local schema. Due to the equivalence assertion, the insertion must be done in both local databases. In this case, a propagation of the insertion is not possible because the object to be inserted does not fulfill the integrity constraints of the other local schema. The problem of local understandability is caused by an *information capacity reducing* schema integration. That is, the information capacity of the global schema is lower than the union of the information capacity of the local schemas. From a global point of view, some locally valid database states are not allowed anymore. This is due to the fact that the extensional assertion contradicts the local integrity constraints. Therefore, extensional assertions must be conform to the local integrity constraints [TS98a]. For instance, from disjoint integrity constraint sets we can reason that the corresponding classes have to be extensionally disjoint.

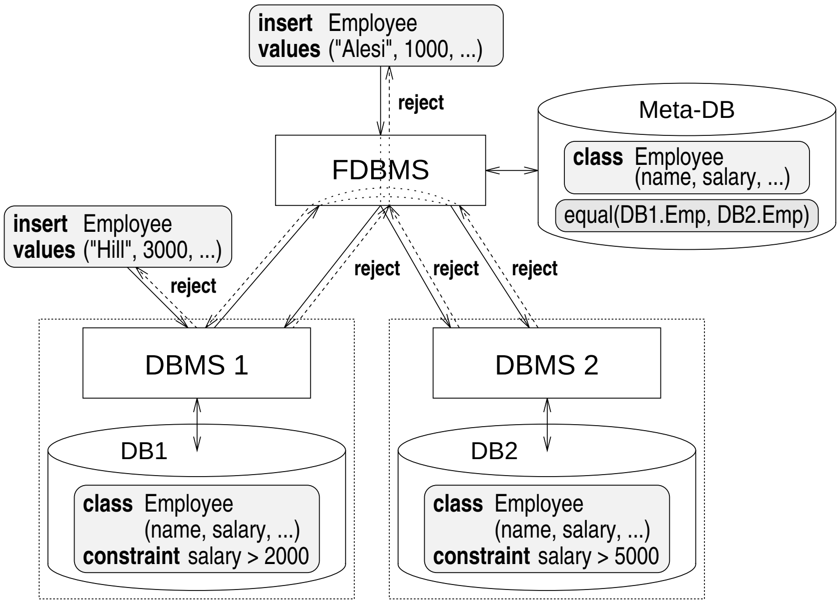


Figure 3.1: The Problem of Global and Local Understandability

Both problems may occur together if the schema integration is not *information capacity preserving*. However, this is a very difficult task. In particular, the computation of the relationship between two sets of integrity constraints in general is an undecidable problem (since testing for satisfiability or implication is undecidable) [Con86, BC86]. However, for the most interesting classes of constraints which frequently occur in database systems, the problem is decidable.

Our approach to dealing with integrity constraints during schema integration was discussed in [CHS+97, CST98] for object and uniqueness constraints and in [BT99] for aggregation constraints. In these papers, we have presented a framework for integrating integrity constraint based on a set of logical rules. Integrity constraints are usually given for a certain set of objects, i.e., they corresponds to class extensions. Therefore, we always consider constraints together with the extensions for which they are valid. As sketched in the previous chapter, our schema integration method consists of two steps. In the first step, disjoint base extensions are derived from the given class extensions and the extensional assertions defined by the database integrator. In the second step, class hierarchies are derived from these base extensions according to the underlying intensional relationships. In [CST98] we have provided rules for treatment of integrity constraints of the local schemata during these two steps. Here, we do not want to recall all these rules. Rather, we illustrate our approach by a small example.

Let us suppose that we want to integrate two classes Project1 and Project2 which contain data about employees participated in the projects "Project1" and "Project2", respectively. As depicted in Figure 3.2, these two classes are intensionally overlapping. Assume that the class Project1 contains the integrity constraints **unique**(name) and (salary $>$ 2000) whereas the content of class Project2 is restricted by the integrity constraints (salary $>$ 4000) and (0 $<$ bonus $<$ 5000). The employees in both classes are identified by their social security number, i.e., **unique**(ssn). The value of this attribute is used to identify same real-world objects. Moreover, we assume that both classes may be extensionally overlapping, which means that an employee can participate in both projects simultaneously.

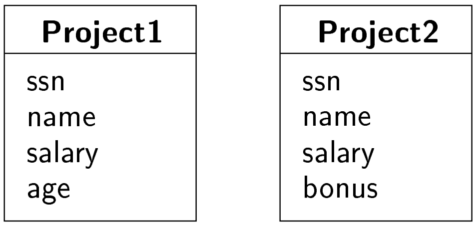


Figure 3.2: Example Classes

Based on the schema integration methodology described in the previous chapter, first the base extensions are derived from the given classes. Then, the respective integrity constraints are assigned to the base extensions. In our case, there are three base extensions are denoted: Ponly1, P12, and Ponly2. As the naming indicates, the base extension Ponly1 (Ponly2) contains only the employees that participate only in the project "Project1" ("Project2"). The base extension P12 contains all employees that are involved in both projects (see Table 3.1).

| **Base ext.** | **Ponly1** | **P12** | **Ponly1** |
|---|---|---|---|
| ssn | ✓ | ✓ | ✓ |
| name | ✓ | ✓ | ✓ |
| salary | ✓ | ✓ | ✓ |
| age | ✓ | ✓ | |
| bonus | | ✓ | ✓ |

Table 3.1: Intensional and Extensional Relationships of Example Classes

Now, we have to check which constraints of the original classes are still valid in the corresponding base extensions. Base extensions were created by an extensional decomposition operation. Therefore, we call an integrity constraint *decomposable* if it remains valid on any subset of the original class extension. Any constraint which is decomposable can be adopted unchanged to the corresponding base extensions. In our example we can adopt the constraints **unique**(ssn) and **unique**(name) because uniqueness constraints are always valid for each subset of an extension. Please note that the decomposition of class constraints leads to a loss of information (which have to be compensated later during the composition phase). Obviously, object constraints are also decomposable. Therefore, the constraints($\mathsf{salary} > 2000$) must hold also for the corresponding base extensions Ponly1 and P12. Since integrity constraints of the class Project2 are decomposable, they can be adopted to the base extensions Ponly2 and P12. Thus, the following integrity constraints hold on the base extensions:

**Ponly1:** This base extension forms a subset of the class Project1. Therefore, it contains all decomposable constraints of that class:

$$\forall o_1, o_2 \in \mathsf{Ponly1} : ((o_1.\mathsf{ssn} = o_2.\mathsf{ssn}) \vee (o_1.\mathsf{name} = o_2.\mathsf{name}) \Rightarrow o_1 = o_2) \wedge$$
$$\forall o \in \mathsf{Ponly1} : (o.\mathsf{salary} > 2000)$$

**P12:** All decomposable constraints of the classes Project1 and Project2 are combined conjunctively in the base extension P12. Since the constraint ($\mathsf{salary} > 4000$) implies the constraints ($\mathsf{salary} > 2000$) we may remove the latter one:

$$\forall o_1, o_2 \in \mathsf{P12} : ((o_1.\mathsf{ssn} = o_2.\mathsf{ssn}) \vee (o_1.\mathsf{name} = o_2.\mathsf{name}) \Rightarrow o_1 = o_2) \wedge$$
$$\forall o \in \mathsf{P12} : (o.\mathsf{salary} > 4000) \wedge (0 < o.\mathsf{bonus} < 5000)$$

**Ponly2:** Similar to the base extension Ponly1, all decomposable constraints of the class Project2 are adopted:

$$\forall o_1, o_2 \in \mathsf{Ponly2} : (o_1.\mathsf{ssn} = o_2.\mathsf{ssn} \Rightarrow o_1 = o_2) \wedge$$
$$\forall o \in \mathsf{Ponly2} : (o.\mathsf{salary} > 4000) \wedge (0 < o.\mathsf{bonus} < 5000)$$

When composing base extensions to global classes, we have to check whether the constraints are composable. An integrity constraint is called *composable* if it remains valid on any superset of the original extension. Unfortunately, none of the given constraints is composable (since all are decomposable) [BT99]. Therefore, we cannot adopt these constraints *unmodified* to the unified classes. If a class is composed by the union of two base extensions which is not associated with any class constraints, then the corresponding object constraints of the base extensions can be disjunctively connected. However, base extensions usually contain class constraints such as uniqueness constraints. In this case, we can solve the problem by introducing *discriminant*-attributes. Discriminant-attributes are used to assign the objects of the unified classes to their corresponding base extensions (see [CST98]). In our example, we may introduce the discriminant-attribute p. A discriminant-attribute value of 1 corresponds to objects of the base extension Ponly1; the values 2 and 3, respectively, refer to the base extensions Ponly2 and P12. Using discriminants the constraints for the unified classes can be defined as follows:

**P1orP2:** This class is composed by the union of the base extensions Ponly1 and Ponly2. Therefore, all objects in this class must fulfill either the constraints of Ponly1 or those of Ponly2. However, since both classes contain class constraints, we cannot integrate them by simply building a disjunction. In order to correctly represent the local integrity constraints, they are expressed by using the discriminant attribute. Please note that in case of **unique**(ssn) this is not necessary because this constraint is defined in both local classes:

$$\begin{aligned}
&\forall o \in \mathsf{P1orP2} : (o.\mathsf{p} \in \{1,2\}) \land \\
&\forall o_1, o_2 \in \mathsf{P1orP2} : (o_1.\mathsf{ssn} = o_2.\mathsf{ssn} \Rightarrow o_1 = o_2) \land \\
&\forall o_1, o_2 \in \mathsf{P1orP2} : (o_1.\mathsf{p} = 1 \land o_2.\mathsf{p} = 1 \land o_1.\mathsf{name} = o_2.\mathsf{name} \Rightarrow o_1 = o_2) \land \\
&\forall o \in \mathsf{P1orP2} : (o.\mathsf{p} = 1 \Rightarrow o.\mathsf{salary} > 2000) \land \\
&\forall o \in \mathsf{P1orP2} : (o.\mathsf{p} = 2 \Rightarrow ((o.\mathsf{salary} > 4000) \land (0 < o.\mathsf{bonus} < 5000)))
\end{aligned}$$

**P1andP2:** This class contains all three base extensions and therefore all constraints of the original classes:

$$\begin{aligned}
&\forall o_1, o_2 \in \mathsf{P1andP2} : (o_1.\mathsf{ssn} = o_2.\mathsf{ssn} \Rightarrow o_1 = o_2) \land \\
&\forall o \in \mathsf{P1andP2} : (o.\mathsf{p} \in \{1,2,3\}) \land \\
&\forall o_1, o_2 \in \mathsf{P1andP2} : (o_1.\mathsf{p} \in \{1,3\} \land o_2.\mathsf{p} \in \{1,3\} \land o_1.\mathsf{name} = o_2.\mathsf{name} \Rightarrow o_1 = o_2) \land \\
&\forall o \in \mathsf{P1andP2} : (o.\mathsf{p} =\in \{1,3\} \Rightarrow o.\mathsf{salary} > 2000) \land \\
&\forall o \in \mathsf{P1andP2} : (o.\mathsf{p} =\in \{2,3\} \Rightarrow (o.\mathsf{salary} > 4000) \land (0 < o.\mathsf{bonus} < 5000))
\end{aligned}$$

By integrating integrity constraint we can solve the global understandability problem. For solving the local understandability problem, the integration of integrity constraints must be done in a consistent way with respect to extensional assertions. However, since the extensional assertions are defined by the database integrator, the extensional assertions and the underlying local integrity constraints may be conflicting. Based on the idea that integrity constraints restrict the possible extension of related classes, the derivation of extensional relationships can be partly supported as shown in [TS98a]. For a deeper discussion on semantic integrity issues in federated databases we refer to [Tür99].

# Chapter 4

# File Integration

Many databases containing useful data are based on files. Traditional FDBS design approaches fail in integrating data files since there is neither a data model nor a database schema which can be exploited for schema integration. Nevertheless, the importance of such systems for a federation is confirmed by our analysis of the described application scenarios.

A first approach to integrating file-based systems was presented in [Höd96]. It is important to mention that the integration has to cover all possible files, and should not be restricted to only one or some example files. In this context, we use the term *file cluster* to refer to these file databases which cover all files of an application system. For the integration of such file clusters we suggest the derivation of a structure description, e.g. a grammar, describing the physical structure and the syntax of the data files and a conceptual schema, modeling structure, and semantics in database terms. The structure description could be a SGML-DTD (SGML Document Type Definition) or a context-free grammar according to YACC [Höd96, Höd97]. The conceptual schema should be based on the common data model of the FDBS (see also Figure 4.1).

The derivation of these two schemata cannot be done automatically. However, in order to reduce the cost of the very expensive, manual modeling of a structure description and a conceptual schema as well as to improve the quality of derived adapters or wrappers, the process has to be supported by tools. As discussed in [HC98], the use of data-mining techniques seems to be very promising. In [HHSS98] we deal with the integration of WWW-based data sources which is closely related to file cluster integration.

In the following section we will discuss several approaches and tools for adapter implementation. First we focus on three approaches for adapter implementation. After this we discuss techniques and tools for semi-automatic description or grammar derivation.

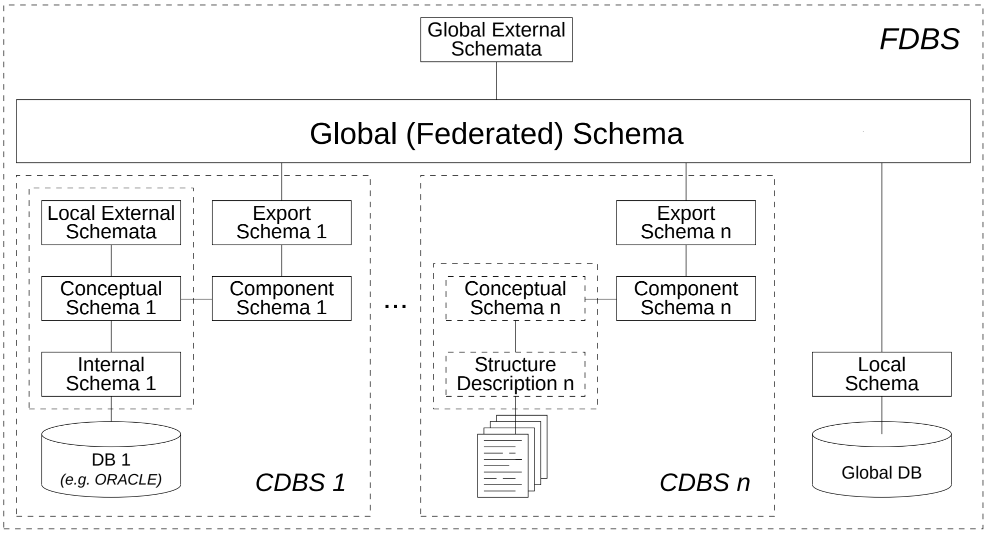


Figure 4.1: Extended Schema Architecture

## Traditional Approach: Programming Adapters

Accessing to (foreign) files is a really historic problem in information system design. Normally software developers focus on the eternal application problems, like calculation or simulation and sometimes also to comfortable graphical user interfaces. Persistent data management is often neglected. In many cases, some textual or binary file format was quickly designed to store information into files. A very specialized loading function, programmed for reading the specific file format, is reconstructing data structures from the files for later usage in the application system.

The traditional approach is programming a similar specific loading function. Obviously (real) computer scientists would use techniques as grammars and parser generators (cf. section 4). However most adapter developers are programmers which are lacking this knowledge. They use the try and error method, sketched in figure 4.2.

The main problem is mostly the use of a traditional third generation programming language neglecting tools or libraries for parsing files. The programmers like to build up the adapter on a low level file interface, supporting only basic file operations. In that way the code becomes very large and causes difficulties in maintenance. Experiences with the programming approach show, that most implementation errors are some how programming errors which are not related to the specific file format. These are mainly compilation errors, i.e. errors regarding the syntax or semantic of the programming language or errors regarding the usage of libraries (link errors). Errors regarding to the syntax of the file format can only be found in the late step of execution checking errors messages or test output. For the detection of semantic errors the programmer has to check the output.

This causes unnecessary compilation cycles compared with the approach in the following section. The most of try and errors cycles are dedicated in boring detection of

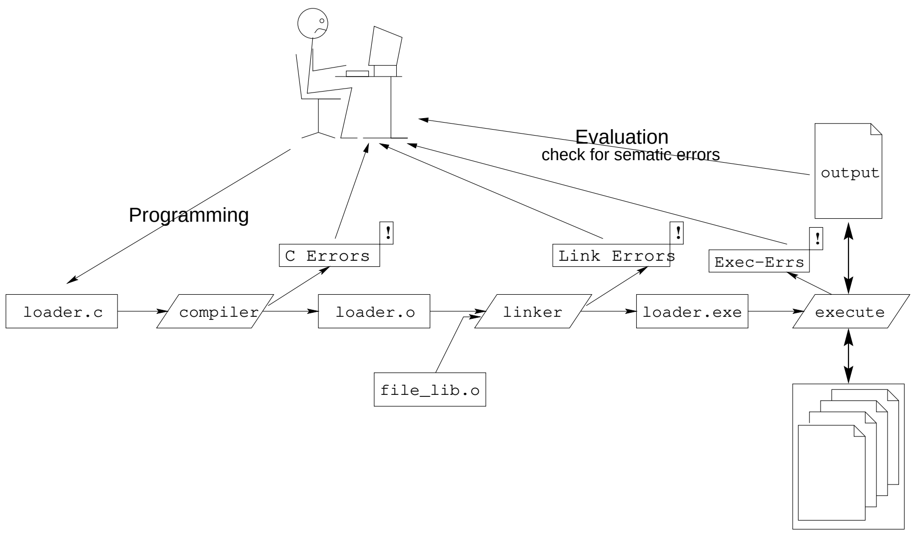


Figure 4.2: Extended Schema Architecture

programming language errors. The programming approach lacks in specific tools supporting the developer in the specific task of building a file cluster adapter.

# Generative Approach

The generative approach is based on the idea of generating the adapter source code using a descriptive specification of the file format. The developer of an adapter describes the structure and semantics of the file format using a higher level description language [HHSS98]. In difference to the programming approach he or she does not describe the operations for opening a file, reading bytes, etc.

Based on the description a *generator* derives source code for a programming language, containing the operational part of the adapter. A number of parser generators, e.g. YACC or JavaCC, support the derivation of parser modules according to a given grammar (description). The methodology for building grammars and parsers is well founded. Using such techniques one can avoid a major number of programming errors, especially regarding the operational part of the loader module. Now the main class of errors are made regarding the syntax and semantic of the specification language. The following steps of compilation and linking will be succeed without errors. A major class of errors are execution errors. The will mostly occur when the description does not match the syntax of the file format. Checking remaining semantic errors is again a main manual task for the intelligent designer.

The advantage of the generative approach is the delegation of human work to the intelligent part of adapter implementation process. For the design of description different languages were discussed. First works discuss the use of SGML as a framework [Höd96, Höd97]. The lack of powerful tools for SGML processing lead to the use of YACC

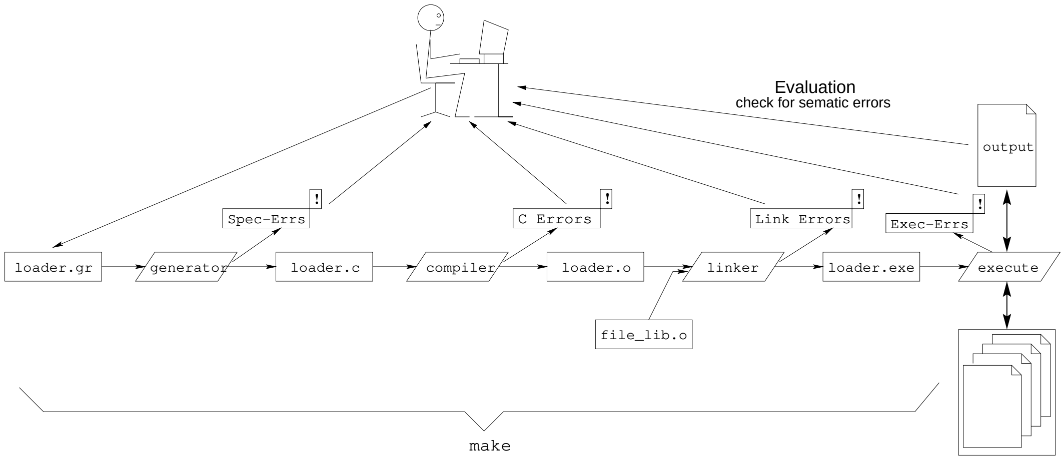


Figure 4.3: Generative Approach for Adapter Development.

based techniques, as discussed in [ACM93]. However, YACC grammars have to be very detailed and support neither concepts for repetition of elements with given cardinality nor semantic specification of schema objects. The last part has to be programmed in traditional manner and therefore one has (partially) to deal with the problems mentioned above.

In [EHH99] we described a generative approach merging YACC syntax with additional constructs for a higher level description. Some of the concepts, e.g. cardinality and repetition of elements, are influenced by SGML. Beside the specification of the file format syntax by the grammar, also classes for an appropriate ODMG schema and mapping to these classes can be defined. The generator succeed two steps. First it translates the specification into a YACC grammar and generates the ODMG classes in a repository. Second the executable or linkable adapter module is derived using YACC and C. The existing implementation is part of the FIHD framework for database integration [HP97].

Future work has to discuss the use of XML and RDF for description purposes. Up to now only some less important systems support XML DTD. However, promoting XML to a standard could result in a common use of XML also for legacy file clusters.

We have to point out that the generative approach also comes along with a major disadvantage. First the definition of a grammar is a quite costly process and has to be done by a developer who owns reasonable knowledge in adapter design as well as the application data, contained in the file. He or she has to define the description as a hypothesis about the file format. The second problem to mention is the late evaluation of the hypothesis, after generation and compiling.

## Configuring Adapters

The configuration approach also uses the fact that the main part of an adapter is independent from the specific file format. In difference to the generation approach here

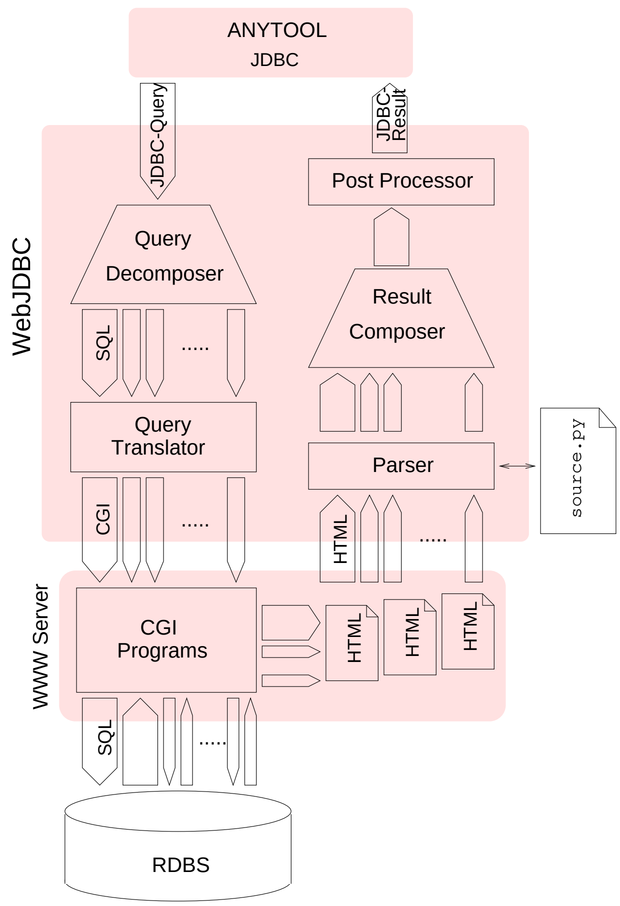


Figure 4.4: Architecture of WebJDBC.

the compiled adapter is available and accesses a file cluster according to a given source description. In programming language terms, here we have to deal with an interpretative approach in contrast to the compiling approach presented in the previous section.

An example for configurable adapters is WebJDBC, designed and implemented by Sattler [SH99]. This system is dedicated to Web-Databases offering CGI interfaces for querying and supporting semistructured textual data files (cf. figure 4.4). For such systems WebJDBC provides a database-like interface according to JDBC. Since WebJDBC is focusing on Web data, it does not use a grammar. Instead of that only the structure of the useful inner part of the textual result-WWW-page is described by patterns and utility functions. Therefore Sattler calls it an *operational* description.

Once implemented a common WebJDBC adapter, the development of a description for the file format is the major problem. The advantage of the configuration approach is the immediate evaluation of the file format description. In that way a highly interactive development of a source description provides good productivity. This fact was illustrated by experiences with the FLORID system [HL98]. Here a F-Logic program, interpreting web sources, can be designed step-by-step.

The configuration approach is a promising solution for file access. Especially for accessing relationally structured data[1], that can be mapped into a relational schema, configurable adapters are a suitable solution.

## Describing strongly structured files

The previous section discussed several approaches and system architectures to file adapter implementation. However the "magic task" of refining structure information and other meta data has still to be performed manually by the intelligent developer. We will discuss tools which can support the developer during this sophisticated process. The next section deals with semistructured data sources. This section is dedicated to strongly or completely structured files.

Completely structured files can be defined as files, which can be modeled with a grammar. Several languages for grammar definition are existing, supported by tools like lexical scanners and parser generators. Such grammars can be directly compiled into adapters. The usability of a grammar definition framework is important for the productivity of the adapter development process. Old fashioned style YACC grammars, based on the Backus-Naur-Form, are difficult to handle. For example a file contains many book objects which could be followed by a publisher object, one has to define the following rules:

```
file : book_publisher file |
       book_publisher;
book_publisher : book |
               : book publisher;
```

To define that a Manager has a name and four phones one has to write:

```
manager: name phone phone phone phone;
```

Alternatively other grammar definition languages, e.g. the Enhanced-Backus-Naur-Form provided by JavaCC, support enhanced features. In the $FIHD_{FDL}$ language of Ebert the given examples can be easily defined as follows:

```
file : (book publisher?)+
manager: name [phone]4
```

[1] We have to point out the difference between semistructured data and semistructured files. Semistructured data means it is difficult or impossible to define a suitable set of classes for an object society, i.e. every object has his own specific set of properties. Semistructured files are files that contain some how structured data objects as well as informal textual information or additional data. Semistructured files often contain strongly structured data. For example, a result-web-page generated form a relational database, e.g. as the result of a search query to an online book seller, contains textual information as well as structured data.

There + means one or many, ? means optional and 4 is an explicit cardinality. In that way a generator providing a high level grammar definition is a powerful tool to support the designer.

A promising task is the semiautomated derivation of grammar and schema. For that one has to consider approaches from machine learning, using grammatical inference methods [AMN94]. The given example in [AMN94] describes the derivation of DTD (document type definition, which are the SGML grammars) for SGML documents lacking a DTD.

Another quite interactive approach was implemented in the NoDoSe-Tool of Adelberg [Ade98]. Here the developer defines the grammar top down, marking data in a text window and assigning them with data structures defined in a grammar or schema window. The tool calculates all candidates for tokens, identifying the data. In that way a structure description can be "mined" step-by-step.

Our own approach, as a part of the $\mathbf{SIGMA}_{Bench}$ is bottom-up. The first step is the syntactical analyze of a given set of files. The idea is to separate the words[2] in a file by delimiters. The set of delimiters can be defined explicitly. The next step is the counting of number of occurrences for each word and the sorting of the list of words according to the count. Based on the hypothesis, that the most frequently words are good candidates for tokens or key words, this list can support the developer in grammar design. However, the experiments prove, that also non key words can occur very frequently and key words, e.g. identifying classes with very few instances in the files, can occur less frequent. Therefore the developer has to improve the list manually. Alternatively a number of similar files could be analyzed and in that way the rare key words accumulate to a higher count. The next syntactical analyze step is finding bracket pairs[3] or block delimiters.

Based on a set of bracket pairs, the $SIGMA-File-Analyzer$ (cf. fig 4.5) constructs a parse tree for the files. Now the parse tree has to be translated into a grammar tree. This is done bottom-up, beginning with the leaves of the tree. The goal is to find similar structured blocks in all levels of the tree. Therefore for each word in a block two alternative representations have to be considered. The first one is the pure representation of a word. I.e. `NAME` is represented as `NAME`. In contrast the second representation is a meta-representation by a type or class. For example `Berlin` is commonly represented as `String`. Checking all combinations of both representations should result in a reduced number of equivalent blocks. The following example illustrates the simplification of inner blocks. The parse tree contains the following inner blocks:

```
NAME Kerstin AGE 29
NAME Gunter AGE 38
NAME Can AGE 29
NAME Michael AGE 31
NAME Ingo AGE 32
```

[2] A word can also be a single character.

[3] Brackets can also be "long" words, e.g. BEGIN or END.

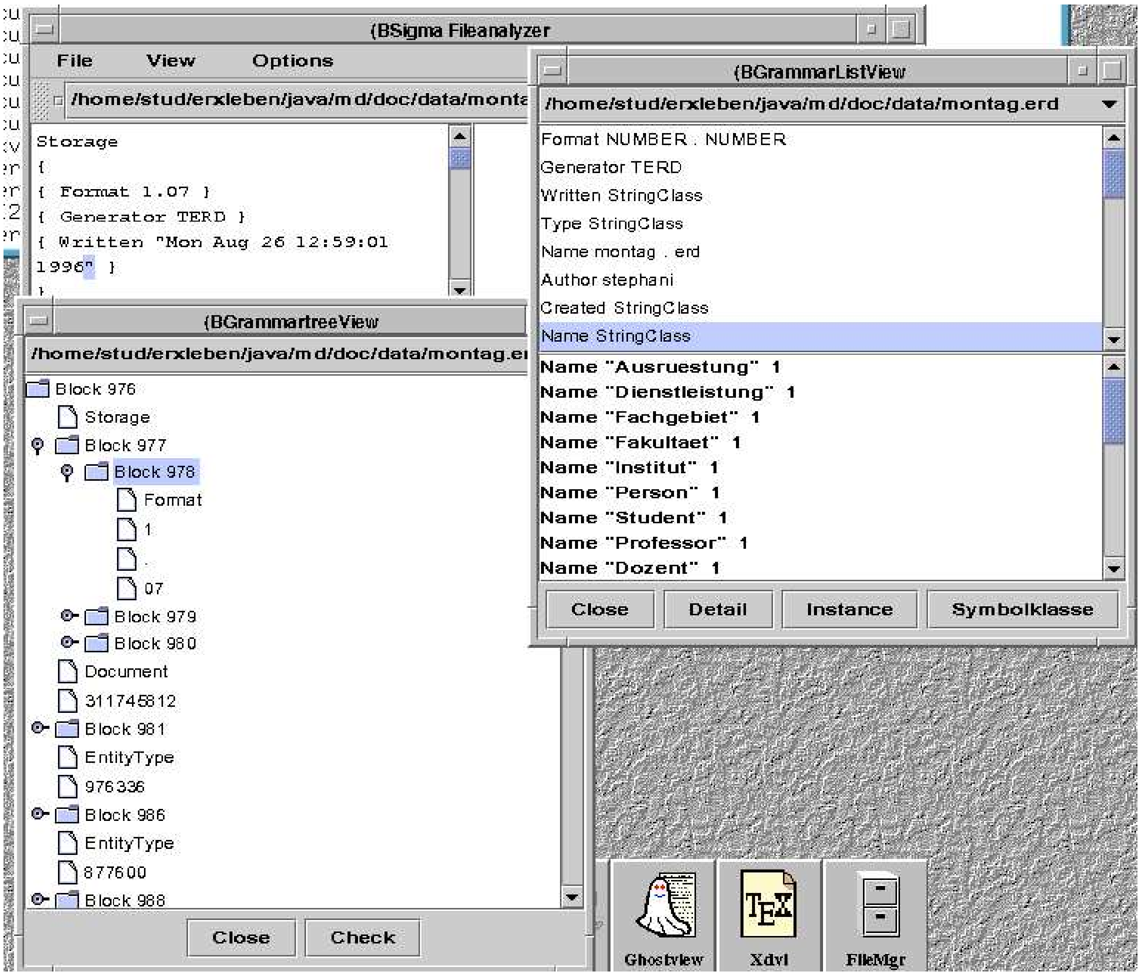


Figure 4.5: Screenshot of the SIGMA-File-Analyzer.

```
NAME Stefan AGE 32
NAME Soeren AGE 24
```

Regarding the number of four words in the blocks one can derive the complete set of 16 combinations. The first combination (`pure pure pure pure`) is the source combination. Setting the first word to meta does not reduce the number of blocks:

```
String Kerstin AGE 29
String Gunter AGE 38
String Can AGE 29
String Michael AGE 31
String Ingo AGE 32
String Stefan AGE 32
String Soeren AGE 24
```

The goal is to calculate the best combination. Best means as least as possible different blocks with as least as possible switches from `pure` to `meta`. As easy can be seen for the example the best combination is `(pure meta pure meta)`, resulting in:

```
NAME String AGE Number
```

In that way it is possible to reduce the number of different inner blocks, that now can be represented by the calculated combinations.

This algorithm can be applied recursively to each level of the parse tree. The result is a grammar tree which is equivalent to the parse tree. The grammar tree can now be translated into a grammar for a adapter generator. The approach is partially implemented and produces good results for strongly structured files using blocks for structuring. Future work will enhance the technique to files where bracket-pairs are tokens with key word semantic or value semantic.

## Describing semistructured files

As argued in [SH99] a grammar is not a suitable approach for semistructured files. For instance, a HTML file, that is generated by a CGI script accessing a relational database, in most cases contains a huge volume of unnecessary and unstructured information. This could be control sequences for formatting of output or textual information, e.g. advertising of products, additional services, etc. Constructing a complete grammar for such a kind of file is unnecessary and expensive.

A more suitable approach for this kind of semistructured files is the application of pattern matching techniques, e.g. with regular expressions. As mentioned above, this is done in the FLORID prototype as well as in the WebJDBC system.

The derivation of patterns and extraction rules for WebJDBC is supported with the Tool $SIGMA - Web - Analyzer$ (cf. fig. 4.6). For that we propose a semi-automatic approach, according to the following steps:

1. Define one or more example URLs, representing queries to the data source (interactive)

2. Load example Web pages (automatic)

3. Extract one or more example tuples from the example Web pages (interactive)

4. Find *prestrings*, *instrings*, *poststrings* for all example tuples in the example Web pages (automatic)

5. Evaluate and unify prestrings, instrings, poststrings; derive *prefix*, *infixes*, *postfix* (automatic)

6. Derive extract-functions for adapter configuration (automatic)

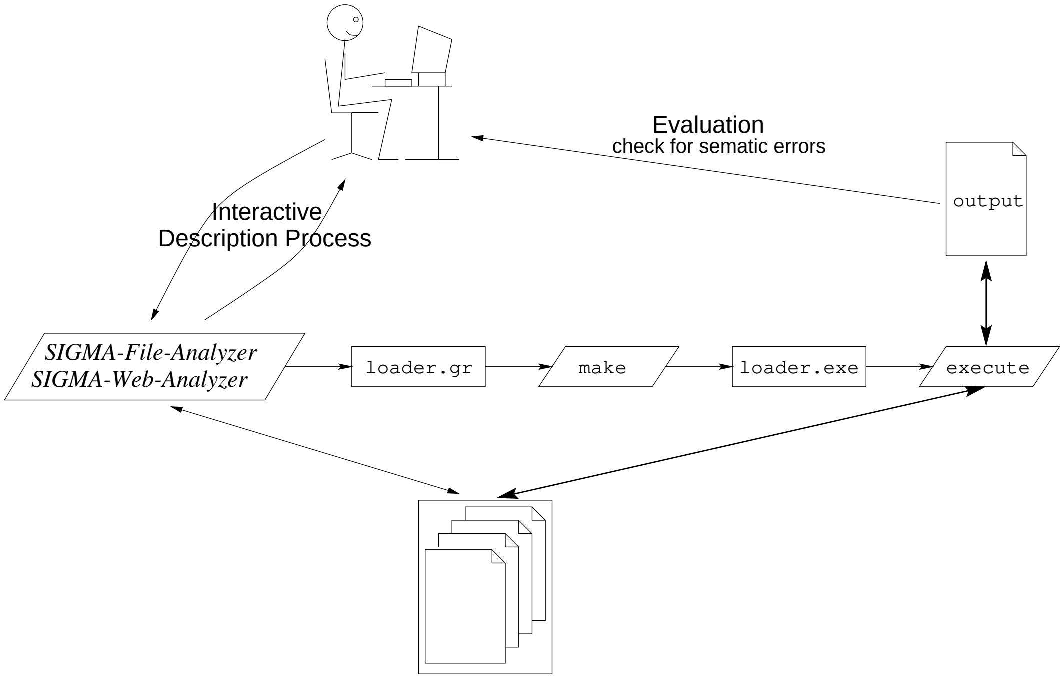


Figure 4.6: Using interactive tools for modeling.

7. Solve multi-valued instrings, derive specific split-functions for adapter configuration (automatic)

Preconditions for the algorithm are the existence of constant prefixes and postfixes, which determine begin and end of data areas (attribute values or tuples) and a suitable set of examples. The term *suitable* is quite fuzzy. For instance the example should contain neighboring tuples. If the algorithm fails, the structure of tuples in the Web documents is not homogeneous and can not be parsed by WebJDBC adapters. However, we have to point out, that Web pages generated by CGI programs accessing a database system generally support the necessary regular structure.

The following set of rules illustrates the main ideas of the extraction rule derivation[4]. The algorithm calculated a unique prefix and postfix for the tuple. Based on this it derives the following rule representing the structure of a result Web page containing a set of tuples.

```
tuple = split(extract(www_page,'tuple_prefix.$1'),
              'tuple_postfix')
```

This rule defines the outer structure of a tuple. Defining the inner structure of the tuple one has to take care of set-valued attributes and not-constant instrings.

[4]For easy read of the rules we separate postfix, placeholders, etc. by '.'. These dots are not part of the rules in the resulting source.py file.

Beginning with flat tuples containing only atomic attributes which are separated by constant infixes, that are calculated by step 5 of the algorithm, the following extraction function is derived:

```
att1, att2, att3 = extract(s,'$1.infix1.$2.infix2.$3')
```

Obviously, separators are not always constant. Often separators, given by example tuples, contain additional but unused data. In this case the algorithm derives a constant left side and right side which function as postfix and prefix. In the following example the separator `tmp` between `att1` and `att2` is variable:

```
att1, tmp, att2, att3 =
    extract(s,'$1.postfix1.$2.prefix1.$3.infix2.$4')
```

The handling of set-valued attributes is more difficult. First there are two opportunities to define attribute values for such attributes in the example tuples. The first is to instantiate the value with the complete set of example attributes including separators. In that way the set-valued attribute can be handled in the same way as an atomic attribute. Beside this the attribute has to be defined as 'set-valued' manually. In an additional step the separator has to be computed. A disadvantage is, that in numerous cases the complete string representing the set in the example is really long. This could be caused by extensive use of HTML tags (especially links) or additional but unused information. Therefore we discuss the use of only one example element of the set for each example tuple. (In the case of existence of set-valued first or last attributes, e.g. one author of a list of more authors, these approach influences the calculation of tuple prefixes or tuple postfixes.) For the calculation of prefix and postfix of the set and the separators the algorithm needs example elements from the beginning, the middle, and the end of an example set. Separators can be again constant and variable. In the following example `att2` is set-valued and the set elements are separated by a (constant) infix:

```
att1, tmp, att3 = extract(s,'$1.infix1.$2.infix2.$3')
a_set = split(tmp,'$1,set_infix')
```

To support variable separators an additional extraction rule has to be derived as illustrated before.

The algorithm is implemented in Java and supports the semiautomatic and interactive derivation of extraction patterns for semistructured files.

# Conclusion

File integration is a necessary task in FDBS design. We discussed the main approaches for adapter development in conjunction with alternatives for file structure description.

Commonly, adapter development cannot be completely automated. An intelligent developer have always to analyze the file cluster and model the file structure. Moreover he or she has to evaluate the results of parsing. Basic tools, e.g. parser generators or configurable adapters, has to relieve the designer from routine work. Enhanced tools for source analyzes can support the intelligent work of file description derivation partially.

There is a number of additional problems which has to be considered. These are transactions on files, updating file clusters, simple file formats (e.g. standard line or comma separated files, use of standard interfaces (JDBC, ODBC), etc.

# Chapter 5

# Security in Federated Databases

A critical feature of database systems concerns the management of confidential data. For that, we need powerful mechanisms for user management, authorization and authentication. In traditional database systems there are many of these mechanisms implemented in different ways. In a federated environment they have to work together with the analogical global ones to ensure security. Hereby, we define security as the state that occurs when all requirements regarding the secrecy, integrity and availability of the managed data are met (for a further introduction in the area of database security we refer to [CFMS95]). In accordance with our component oriented approach (section 7) we present in this chapter some components for user authentication and authorization in federated databases.

Before we go into details we have to discuss some basic assumptions about security requirements in federated databases. Several factors determine these requirements. Firstly, there is the factor of autonomy. This means, that all local systems have the right to implement and enforce their own security measures. As a result, the global system has to respect, to support and to cooperate with the local mechanisms. Secondly, there is the factor of heterogeneity. This means, that the local measures rely on different mechanisms and that the local systems have or more or less trust in a person regarding secrecy and integrity. As a result, we need global security measures which can provide

- a uniform security interface for all federated users based on a locally accepted federated security policy,
- approaches to resolve security conflicts that arise from the syntactic and the semantic heterogeneity among the local systems and
- approaches to resolve special federated security problems like undesired aggregation of access rights.

It is obvious that we need the trust of the local systems in these global security measures to ensure a secure and working federated system. This assumption leads to the requirement that the global security component has to build on mechanisms that are at least as trustworthy as these from the local ones.

The remainder of this chapter is organized as follows: first, we lay down the basis for authentication and authorization and, second, the foundations of our component based approach. Afterwards, we present three different components to resolve the problem of user authentication in federated database systems. Similarly, the following section provides three components for federated authorization. Finally, we summarize our approaches and compare them with existing ones.

# Preliminary Considerations

The core of every security component in operating or database systems are the mechanisms for the authentication of users and the implementation of the authorization process. Because we want to deal with the realization of these two aspects in a federated environment, we have to lay down some basic facts about these aspects in the first part of this section. Further on, we describe some considerations about the principal realization of a federated security component.

## Security Concepts

Before we describe the processes of user authentication and authorization in traditional centralized database systems we define some principal notions.

**Users** are natural persons with the principal right to access a database system.

**Identity** or *login* is the unique attribute with which a user is attached to a determined system.

**Identifier** is an attribute that can be uniquely assigned to a determined user and proves his identity via the system.

**User profile** contains all information about a user that is managed and stored by the database system.

**Identification** is the assertion of a user to possess a determined identity.

**Authentication** is the process of proving that a user possesses the asserted identity.

**Access Rights** are statements about the rights of subjects (users, programs a.o.) to access objects (pieces of data) in certain ways (read, write a.o.) inclusive the right to pass them on to other subjects or not.

**Authorization** is the process of proving, granting, revocation and management of access rights.

For centralized database system we distinguish two kinds of authentication components. Firstly, the task is done by the operating system or a hardware component outside the control of the database system with accepted and trustworthy mechanisms. Hereby, the database system has no access to user identifiers and shares the response for security with the operating system. Secondly, the database system realizes and controls an own component for this task. This allows an easier coordination with the other parts of access control.

On the other side there are authorization components. These rely mainly on the chosen security policy and model. They are mostly software moduls directly integrated into the database management system and connected with a data dictionary containing the access rights in the form of the chosen model. They perform the following tasks:

**Proving:** The component checks every query against the stored access rights. Dependent on the result the query will be performed completely, partly, or not.

**Management:** The component manages and regulates the access to the stored access rights. It also secures their consistency in compliance with the implemented policy. Therefore, it has to check the insertion, the update and the deletion of access rights manually performed by administrators and users.

**Granting and Revocation:** The component supports the granting and revocation of access by users or administrators following an implemented policy.

Finally, we state that an authorization component is always as complex as the underlying policy and model.

## A Secure Federated Architecture

Besides the presented security concepts we need another premise to start our investigation. We have to define a basic concept for a secure federated architecture. It should be so generic and flexible that we can on the one hand integrate our authentication and authorization components and that we are on the other hand independent in face of different requirements and conditions. We present and discuss such a concept shortly in this section.

We start with the consideration how we can establish a security system into the federated database architecture presented in Figure 4.1. It should fulfill the requirements stated before. This lead us to the following components and their role within the federated security system:

**Local Security Components:** We have to respect their tasks because of the valid autonomy. Depending on the cooperation of the local system we can use different strategies to integrate these components into the federated security system. The main hindrance hereby is the heterogeneity described before.

**Global Security Components:** The tasks of these components were described at the beginning of this chapter. Their realization depends on the concrete situation of the specific federated database system. Principally, they consist of existing software modules to perform tasks like a global password check and special software modules to resolve the specific federated security problems.

**Interface:** We need also a special interface that connect the local and the global components. It has to perform the following tasks:

- access to, transformation and integration of local security data like user profiles and access rights,
- transmission and translation of messages between the local and global components.

We can see that we have herewith an instruction for the development of specific federated security systems which again can easily be integrated into the whole federated database system. Figure 5.1 shows an example of a federated database system with a security system that consists of several local and global components for authentication, authorization and auditing purposes. The lines between the local and global components represent hereby the required interfaces. We left out all other necessary components of a federated database system except the transaction manager. We deal with the realization of federated components for authentication and authorization in the next two sections.

## Authentication

The identification and authentication of users in federated database systems is a more complex process than in traditional ones. We now discuss the main reasons and the resulting tasks:

**Heterogeneity:** The local authentication components can base on the variety of security concepts. As a result users have to pass through all these different procedures to gain access. Besides, the identity of a user can vary from system to system. The task is to overcome this heterogeneity without a violation of security. That means, each user is authenticated once but correct to all relevant participating systems per session in a federated environment.

**Autonomy:** The local authentication components decide dependent on the delivery of the correct identifier about a users access wish. The maintenance of this kind of autonomy can be necessary to secure the trust of the local systems.

**Population control:** A user can operate in a federated database system with different identities and identifiers. Otherwise, a user should be handled as a single subject independent of the identity he logs in. The authentication of users without local identities can be a task in environments with closed local populations.

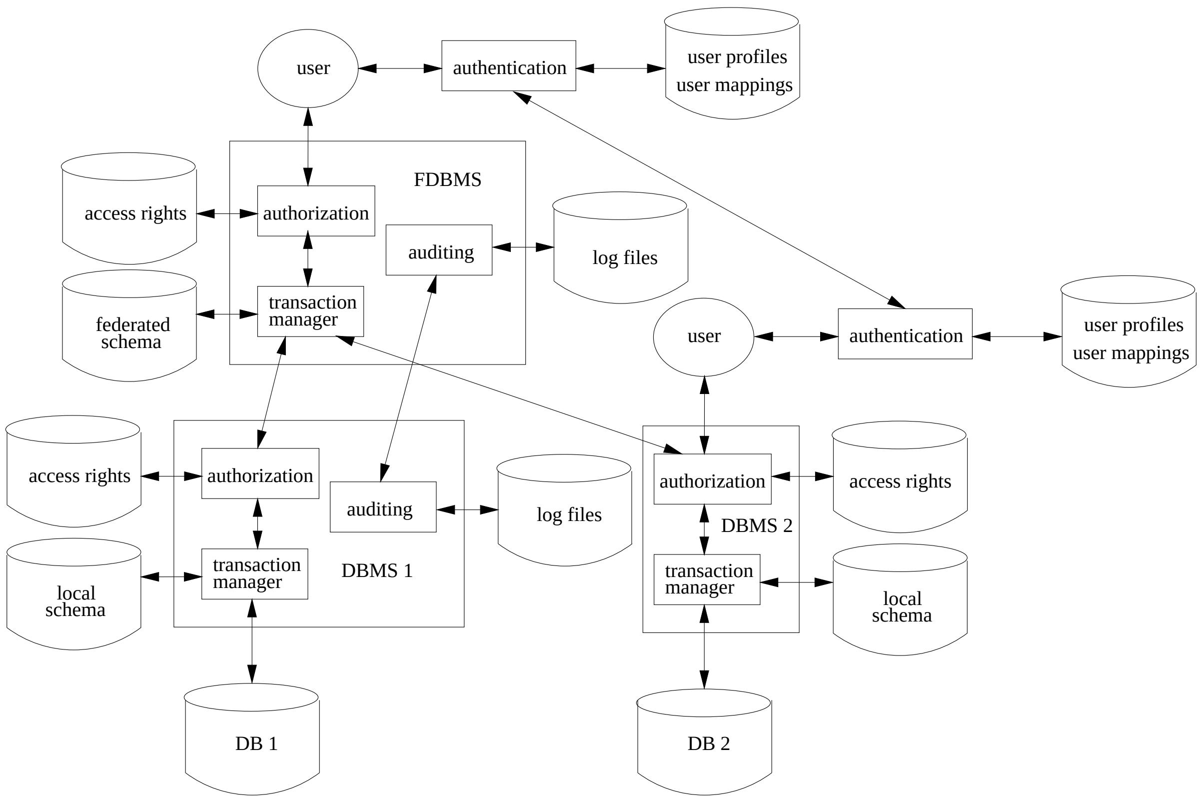


Figure 5.1: A database federation with security components

The heterogeneity and the requirements regarding the maintenance of autonomy vary in different application environments. In consequence, we need several approaches to handle this process. We present three of them below. But first, we turn to the task of population control with the presentation of a tool for the management of users in a federated database system.

## User Management

In this subsection we deal with two aspects. On the one hand, we introduce our user concept. On the other hand, we describe a global tool for the management of users in a federated environment. Herewith, we lay the basis for a successful authentication process.

For simplification we assume that each local system associates each of its users with exactly one identity and one identifier. We do not consider groups of users like for example roles or other possible subjects like programs. They all can be easily backtracked to an individual user. Each user of the global system receives a global identity and a global identifier. So, we can recognize the following three classes of users in a federated database system:

- local users with one identity per affiliated system,
- global users with only a global identity,
- federated users with local and global identities.

The following events cause dynamic transitions between these classes:

- acquisition of a global identity,
- acquisition of one or more local identities
- dispossession or return of local or global identities

The granting and withdrawal of identities for users regulates the principal access to a determined system. This is the task of the responsible administrator of the system. In federated database systems we need therefore a coordinated policy to prevent unauthorized access and to reach the goal of population control. For the granting of global identities we present the following principles:

- The global system grants a global identity to each local user who wants to use the federated system. If possible this identity should be equal or similar to the local ones to prevent unnecessary heterogeneity.
- The global system grants a global identity to persons with no access to local systems if they fulfil at least one of the following requirements:
  - one or more local identities should be granted,
  - access to global data should be granted,
  - access to determined local data should be granted without the required local identity.

The last requirements can occur if the enrolment of new users in a local system is prevented by a limited capacity or a determined policy. We discuss different possibilities for the realization of this requirement with the corresponding authentication approaches. The withdrawal of a local or global identity should be propagated to all other participating systems where the user belongs to. Besides this more or less flexible rules the following rule has to be enforced through an intensive cooperation between the local and the global administrator:

**Identity association:** For every federated user should exist a global managed reference between his global and all of its local identities.

One consequence of this rule is the postulation that all local systems have to release user profiles and other information about users to the global system. Herewith, we have also an important requirement for our user management tool.

The global user management tool should be a part of the administration modules of the federated database management system. The main task of the tool is to support the administrator on the management of global user profiles and identity references as well as the enforcement of the user access control policy. Therefore, we need a new schema for the global data dictionary to represent the information about the users. We introduce two new classes (or tables) according to the underlying data model. The first class *global-user* contains all necessary information about global and federated users including real name, identity and identifier. The second class *local-user* covers up all necessary local information about federated users. A 1–to–n reference between these classes expresses the desired identity association. Figure 5.2 shows this schema.

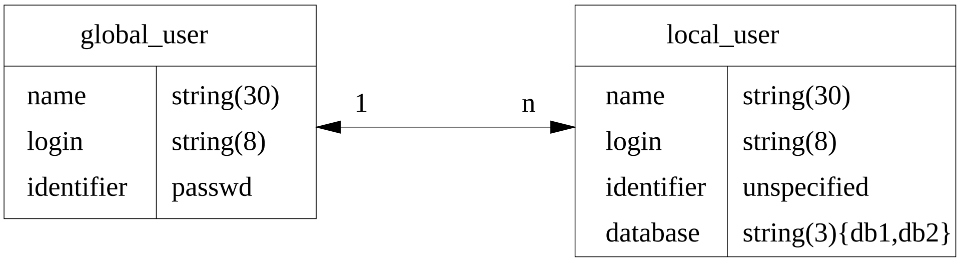


Figure 5.2: Schema of the federated user profiles

We now turn to the description of the necessary functions of this tool:

**Addition of users:** The tool allows both the manual addition of new global or federated users as well as the semiautomatic integration of local users into the federated database system. The global administrator inserts the values for *global-user* objects. The local administrator provides the information for *local-user* objects. The global administrator sets the references with the help of the local administrator.

**Updates of user profiles:** The tool supports also the global administrator by all necessary updates of the user profiles at runtime. This occurs in three ways. Firstly, it prohibits all updates which are in contradiction to the stated access control policy. Secondly, it receives or accesses local updates, converts them in the global user profiles and informs the administrator. Third, it informs the local administrators about updates of profiles which regard their own users.

Currently, we are implementing these ideas as part of the security tool for the **SIGMA**$_{Bench}$.

## Direct, Indirect and Global Authentication

The direct and indirect approaches have one thing in common. A federated user who wishes to access the federated database system has to prove both his global and his local identities. With the direct approach the user authenticates himself directly at all involved systems. With the indirect approach the global system delivers the necessary identifiers of a federated user to the relevant local systems. The third approach is different because there is mainly the global system involved while the local systems only provide trust. We discuss now the realization of fitting components, the appropriate area of application and the advantages and disadvantages for these three approaches.

The autonomy of the local authentication components is fully preserved with the direct approach. The global component can be a simply password–based approach if there is no need to protect global data. The interface has the task to provide secure channels to secure that local information is released only to the authenticated user. Figure 5.3 shows the architecture and an example for this approach. We use this example also for the other approaches. It should be applied in cases where high local protection

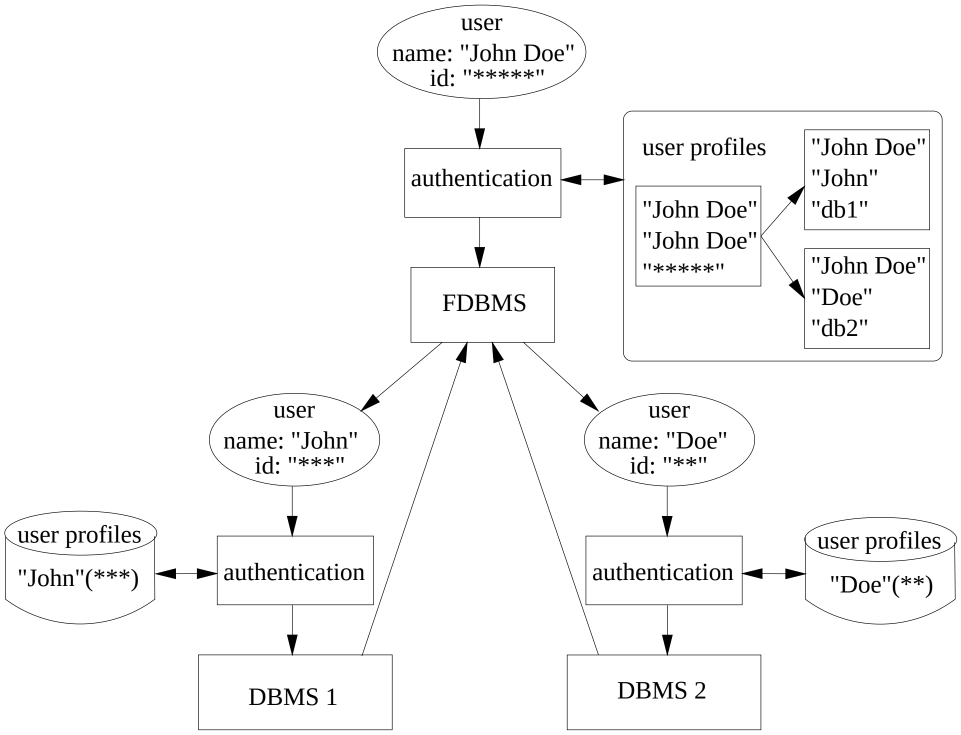


Figure 5.3: Direct authentication

requirements or hardware–based authentication components exist. The main advantages are the preservation of the local autonomy and the herewith connected high level of protection. But on the other hand the heterogeneity is also preserved and this approach is very uncomfortable for the federated users.

With the indirect approach the local authentication components have to accept the delivery of identities and identifiers through the global system in behalf of the attached user. Therefore, adequate adaptions have to be implemented into these component. This approach also requires a high degree of trust in the global system. So, we need an appropriate global authentication component. The interface realizes the necessary transmissions with the help of cryptographic mechanisms. There are two possibilities to make the local identifiers available. Firstly, we can store them durable in the local-user objects of our user profiles. Secondly, the user has to enter them with every session. Figure 5.4 shows the components and the process of this approach by means of our example. We can

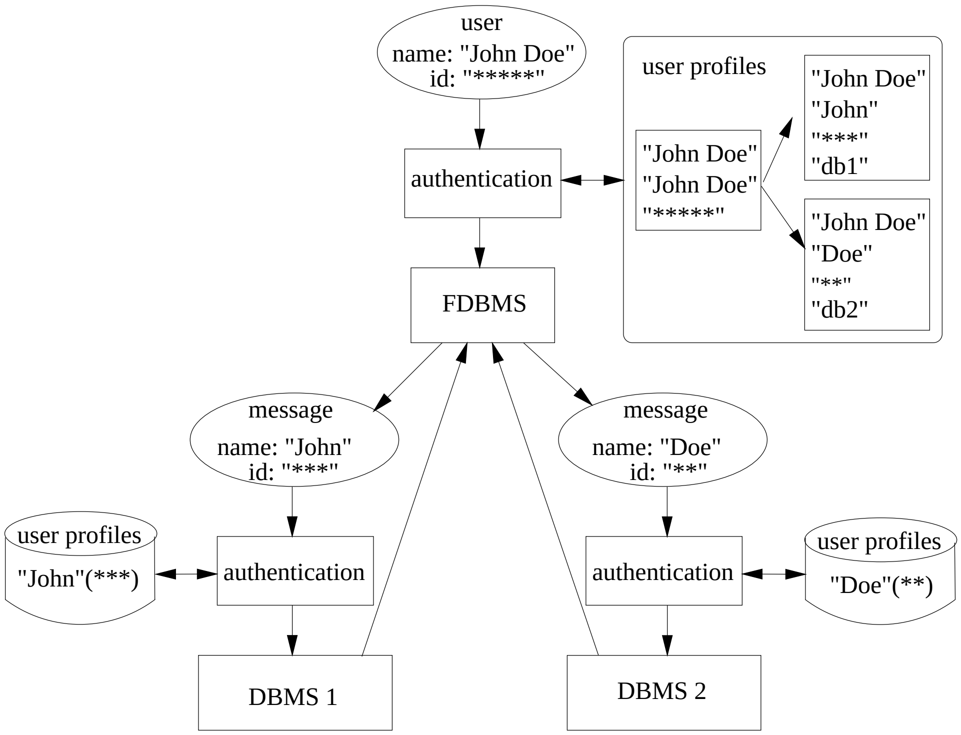


Figure 5.4: Indirect authentication

implement indirect authentication in every federated database system where the local systems gather the necessary trust. This approach is a compromise between preservation of autonomy and overcoming of heterogeneity. But it is very expendable to realize.

With the last approach a user authenticates self only via the global system. The global system on the other hand authenticates itself via all local systems with his own special local identity and identifier. These should be granted by the local systems. Hereby, this is their only task. For the interface we need only one secure channel for the information flow between one local and the global system. On the global side we distinguish two cases depending on how we use the local-user objects. Firstly, we consider these informations by global authenticating all local identities of a federated user. This can be done, if we can implement a global authentication component which enables different authentication processes. Secondly, we renounce these informations where we have again two cases. Firstly, an independent dedicated authentication authenticates all users in the federated database environment. The components of the participating systems can be dissolved. Secondly, there are no federated users. The authentication process is strictly separated in global authentication for global users and local authentication for local users. Figure 5.5 shows the architecture and process for the last variant of this approach. It is obvious

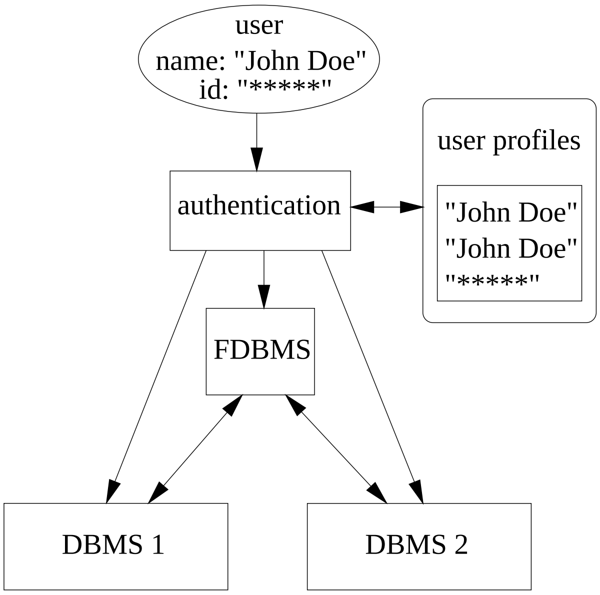


Figure 5.5: Global authentication with independent server

that this approach with its different variants is suitable for environments with special requirements.

For a deeper discussion of these approaches and the generally problem of authentication in federated database systems we refer to [HS98].

## Authorization

The problem of authorization in federated database system is not only more complex than in centralized ones but it also depends on much more factors. Therefore, we describe at first the main aspects we have to consider in the following:

**Heterogeneity:** The main objects of authorization are access rights. Local systems may use heterogeneous ways to express, to grant or to revoke access rights. The reason is that they apply different security models and policies. Beside these syntactic conflicts there are semantic ones if a certain user can access an object in one local system but not the equivalent object in another one. In the federated database system we must find solutions for both problems. That means, we need an uniform presentation and integration concept for access rights.

**Autonomy:** There are two cases of local autonomy. Firstly, there is the operation of an authorization component which proves and manages access rights. Secondly, there is the enforcement of a security policy which regulates the granting and revocation of access rights. Their preservation is always an important goal. But, we have hereby to regard different interests and requirements of the participating systems. So, we have to develop several approaches with different levels of autonomy restrictions.

We divide this problem in two parts. Firstly, we have to solve the conceptional problem how to treat local and global access rights in a federated environment. Then, we turn to the architectural side and propose three different approaches to handle the authorization process. We exclude at this place the problem of granting and revoking of access rights and other questions regarding a federated security policy.

## Presentation and Integration of Access Rights

In this subsection we describe a strategy that allows developers to find an adjustable treatment of access rights for their federated database system. This strategy forms a process tightly connected with the ones from schema and integrity treatment. Four steps (selection of a global security model, transformation, filtering and integration of access rights) form this process. We discuss these steps shortly in the following.

The first step and the basis for all other steps is the selection of a suitable global security model. All access rights which are treated at the global system have to be expressed in this model. We have developed the following four criterions for the selection which take into consideration different circumstances:

- The global security model should be tightly related to the types of the local security models so that access rights can be easily transformed.
- The model should address different granularities of data objects (classes, attributes, references) in an access right with regard to filtering and partitioning during schema integration.
- The selection is limited by the underlying global data model. This means, access rights can only consists of subjects, objects and access types from this model.
- The selection is also limited by the global system itself. This means, we can only use a model that we can implement into the global system.

Both discretionary [JD96, EKP+95] and mandatory [Per93] security models were proposed and applied as global security models. We use for our considerations a very simple model.

Within the federated authorization process it is necessary to transform access rights expressed in a local security model into access rights in the form of the global security model and reversed. The following aspects should hereby considered:

- The resulting access rights should correspond to the local or the component schema to avoid semantic losses.
- It should always be possible to backtrack any subject to one or more individual users.
- If some corresponding concepts like access types are incompatible the developer should always choose the more restrictive concept.

We ourselves have till now examined above all the transformation of different role concepts.

Within the federated authorization process there exists also the task of filtering all access rights which affect only local users and local data that isn't exported to the federation. We have hereby to consider the following aspects:

- The resulting access rights should correspond to the export schema.
- The access rights of not exported users should not be affected by the filtering.

The most complex step is the integration of access rights from different local database systems into access rights of the federated database system. Hereby, we have to deal with two fundamental conflicts.

**A semantic conflict** may arise if a user has the right to access a certain data object in one local system but not the same or overlapping data object in another local system.

**An aggregation conflict** may arise if the access rights of a user from different local systems aggregate to an undesired level at the global system.

We provide here now some resolution strategies for semantic conflicts. An example for such a conflict is easy to imagine. Consider our federated user *John Doe* from the authentication approaches. If he has the right to read objects from a class *book* as user *John* in system *1) but not the right to read objects from an equal class* book *as user* {*Doe* at system *2* and these classes are integrated into a class *book* at the federated system then we have a semantic conflict. The designer of the federated system has now the following three possibilities to resolve such a conflict in cooperation with the local administrators:

- The access should be denied if this is the request of the administrator from system *2*.
- The access should be permitted if this is accepted by the administrator from system *2*.
- The right is directly granted at system *2*. The conflict vanishes.

The recognition of semantic conflicts must be directly linked to the schema integration process.

### Local, Global and Distributed Authorization

As stated above we limit the federated authorization process to the two tasks of the proving and the management of access rights. In each case the name of our approach indicates the place where these tasks are executed. Like in the case of authentication we want to introduce and shortly discuss these approaches.

Local authorization means that all access rights are managed and proved by the local systems where they belong. All global queries as well as global updates are transformed into local queries and local access rights by the global system. The local authorization component checks them. The results of accepted queries will be released and transmitted to the global system. An update that corresponds to the local security policy will be locally stored. If an semantic conflict occurs the global system can decide whether the information in question should be released or not. The main advantage of this simple to realize approach is the preservation of the local autonomy. On the other hand, we can't realize a uniform federated security policy with this approach and special problems like aggregation can't be treated.

Global authorization means that all access rights are managed and proved by the global system. Therefore, all locally stored access rights have to be transformed, filtered and transmitted to the global system. This concerns also all local updates of these access rights. At the global system these access rights will be integrated and managed. All global queries will be checked by the global system against these access rights. If it is possible we recommend to pass by the local authorization component. Global updates of these access rights have to accord the stated federated security policy. If they are accepted by the global system they will be propagated to the accordingly local ones. The main advantage of this expensive to realize approach are the overcoming of heterogeneity and the possibility to establish a federated security policy. On the other hand, we need a high effort to maintain the consistency and the redundancy between the globally and locally managed access rights.

Because both approaches aren't applicable in most federated database systems, we propose a compromise between them which we call distributed authorization. This approach should be flexible to different security and autonomy requirements of the local systems. Like the name says, there are all local and global components involved in the task. At this time we examine this approach and will present details later.

## Conclusion

In this chapter we have presented some thoughts about security in federated database systems. Besides a short introduction into the area of database security we have described the problems and tasks which arise in federated environments regarding the protection of sensible data. We have proposed a basis architecture for the development of federated security components. We have introduced a tool for the management of federated users. We discussed six approaches for authentication and authorization. Before we present an outlook on future work we discuss the connections of our approaches with other work.

Other architectures for secure federated database systems were described in [BMR+95, TF97]. We have developed the presented authentication approaches on the basis of authentication schemes for federated databases from [JD94]. We added among other things some advancements to the indirect version and developed variants of the global one. There exists many work in the area of authorization for federated database systems. We refer particularly to [CDF97, DS97] because of its fundamental significance.

In future work we want to investigate further in the area of architectures and components for trustworthy federated database systems. A particular aspect hereby is to establish a link between the authentication and authorization approaches. However, our main work lies in the development of a federated security policy. We base this policy on the security properties and the behaviour of users in a federated environment.

# Chapter 6

# Transaction Dependencies in Federated Databases

In a federated database environment, *global transactions* are able to transparently access and manipulate data located in different local database systems [BGS92]. At the same time, pre-existing local applications access and manipulate the data of only one local database by submitting *local transactions*. Depending on the underlying global schema, and particularly on the fixed extensional assertions, a global transaction is decomposed into a set of *global subtransactions* which operate on the local database systems.

A global access on a global class of the global schema result in an access on one or more local classes of probably different local databases. For example, suppose the global class $C_g$ is the union of the local class $C_1$ of database $LDB_1$ and the local class $C_2$ of database $LDB_2$. The global access is initiated by a global transaction. The operations on the local classes are performed by the global subtransactions $t_1$ on class $C_1$ and $t_2$ on class $C_2$. During schema integration extensional assertions between local classes are fixed. These extensional assertions influence the dependencies between the global subtransactions of the same global transaction [TS99b, TSS99].

Figure 6.1 represents a scenario where two local classes Employee are integrated into one global class Employee. For simplicity, we assume that both classes have the same attributes. During schema integration the designer of the global schema has fixed that the local classes are extensionally equivalent, i.e., these classes have to contain the same (semantically equivalent) objects in each database state. In consequence, a global insertion transaction $t_g$ must be transformed into two global subtransactions $t_1$ and $t_2$ which try to perform the insertions in the corresponding local database systems. Due to the extensional assertion, both global subtransactions have to be successful in order to commit the global transaction. In case one global subtransaction commits whereas the other one aborts, the requirement that both classes are extensionally equivalent is violated. In order to correctly terminate a global transaction, a global commit protocol is needed.

To overcome the heterogeneity and to maintain global correctness each of the local

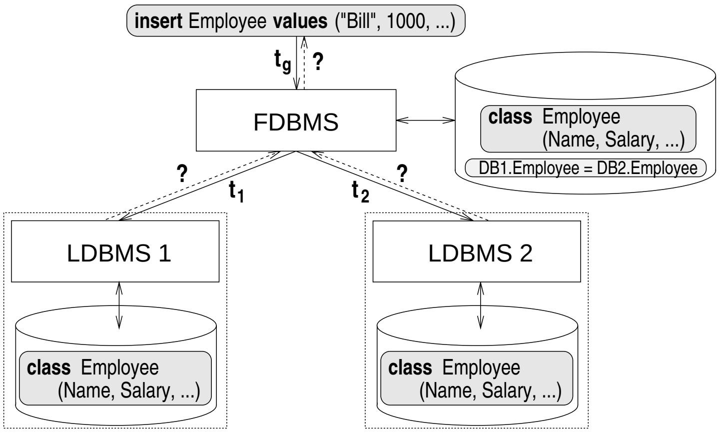


Figure 6.1: Global Insert into Equivalent Classes

database systems participating in the federated database system has to support a visible Prepare-State. Otherwise we cannot guarantee the atomic commitment of global transactions, as proven in [MEKS92]. To treat the problem of global serializability we assume to apply an approach like the ticket method [GRS91, GRS94]. The ticket method enables us to derive the local serialization order from the local commit order of the global subtransactions at each local system. The problem concerning the local serialization order goes back to interleaved access of local transactions and global subtransactions on data of the same database. On the federation level the local transaction are not visible.

The focus in this chapter lies on the dependencies between global subtransactions of the same global transaction which are based on global extensional assertions (see Chapter 2).

Depending on the extensional assertions, different dependencies between the global subtransactions are derived and different *global commit rules* are needed to correctly terminate a global transaction [TSS99]. For instance, in some cases a commit of only one global subtransaction is sufficient to commit the global transaction, i.e., a global transaction can commit although some of the global subtransactions abort. In order to get a grasp of the meaningful global commit rules, we have analyzed the underlying transaction termination dependencies. For that, we use the transaction closure framework [STS98b, STS98a] where we have defined a complete set of the possible termination dependencies.

In the following, we exemplary consider the decomposition of a global transaction $t_g$ and the corresponding global commit rule for this transaction. The global transaction $t_g$ consists of the following insert and delete operations on the global classes $C_{g_1}$ and $C_{g_2}$ (where the classes $C_1$ and $C_3$ are managed in one local database system while the classes $C_2$ and $C_4$ are managed in another local database system):

Insert into $C_{g_1} = C_1 \cup C_2$ with $C_1 \varnothing C_2$;
Delete from $C_{g_2} = C_3 \cup C_4$ with $C_3 \equiv C_4$;

The insertion into class $C_{g_1}$ is successful if the corresponding object is inserted in exactly one of the local classes $C_1$ and $C_2$. In contrast, an object is deleted from class $C_{g_2}$ if the object is deleted from all of the local classes. In this case we have to check each of the local classes $C_3$ and $C_4$ whether the object is part of the class or not.

We suppose an access to a global class consisting of a set of local classes is performed by global subtransactions on each local class. In our example, the global transaction is decomposed into two global subtransactions $t_1$ and $t_2$. Transaction $t_1$ performs the insertion into the local class $C_1$ and deletion from the local class $C_3$ whereas the transaction $t_2$ executes the insertion into the local class $C_2$ and the deletion from the local class $C_4$. The question is now:

> *In which case the global transaction is executed successfully? If both global subtransactions commit or if exactly one global subtransaction commits?*

A commit of both global subtransactions means that the insertion is performed in both local classes $C_1$ and $C_2$ and the deletion is executed on both local classes $C_3$ and $C_4$. However, in this case the extensional relationship between the classes $C_1$ and $C_2$ is not disjoint anymore, i.e., the extensional assertion cannot be guaranteed. Therefore, both global subtransactions cannot commit together. On the other hand, if both global subtransactions cannot commit together, the deletion cannot be successfully executed. *Thus, this global transaction cannot be performed by only two global subtransactions.*

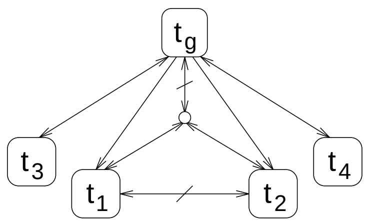


Figure 6.2: Termination Dependencies in our Example

Figure 6.2 presents a possible solution to the example discussed above. For each global operation two global transactions have to be initiated. The global subtransactions $t_1$ and $t_2$ realize the local insert operations whereas the global subtransactions $t_3$ and $t_4$ perform the local delete operations. The insertion transactions are terminated using the **exactly one** commit rule whereas the deletion transactions are terminated by the **all** commit rule. The corresponding termination dependencies are as follows:

$$
\begin{aligned}
& c_{t_g} \Rightarrow ((c_{t_1} \wedge a_{t_2}) \vee (a_{t_1} \wedge c_{t_2})) \wedge c_{t_3} \wedge c_{t_4} \\
\equiv\; & (a_{t_1} \wedge a_{t_2}) \vee (c_{t_1} \wedge c_{t_2}) \vee a_{t_3} \vee a_{t_4} \Rightarrow a_{t_g} \\
\equiv\; & \textbf{vital}(t_1, t_2, t_g) \wedge \textbf{vital}(t_g, t_1) \wedge \textbf{vital}(t_g, t_2) \wedge \\
& \textbf{exc}(t_1, t_2) \wedge \textbf{vital\_dep}(t_g, t_3) \wedge \textbf{vital\_dep}(t_g, t_4)
\end{aligned}
$$

The summarized termination dependencies are defined as follows (between the transactions $t_i$, $t_j$ and $t_k$):

$$
\begin{aligned}
\textbf{vital\_dep}(t_i, t_j) \quad &:\Leftrightarrow \quad (a_{t_i} \Rightarrow a_{t_j}) \wedge (a_{t_j} \Rightarrow a_{t_i}) \\
\textbf{vital}(t_i, t_j) \quad &:\Leftrightarrow \quad (a_{t_i} \Rightarrow a_{t_j}) \\
\textbf{exc}(t_i, t_j) \quad &:\Leftrightarrow \quad (c_{t_i} \Rightarrow a_{t_j}) \vee (c_{t_j} \Rightarrow a_{t_i}) \\
\textbf{vital}(t_i, t_j, t_k) \quad &:\Leftrightarrow \quad (a_{t_i} \wedge a_{t_j}) \Rightarrow a_{t_k}
\end{aligned}
$$

In Figure 6.2 these dependencies are also illustrated. A termination dependency **vital** is represented by a single arrow ($\rightarrow$). The direction of the arrow corresponds to the direction of the abort dependency. For example **vital**$(t_g, t_1)$ means that the abort of transaction $t_g$ leads to the abortion of $t_1$. Thus, the dependency **vital_dep** is represented by a bidirectional arrow. Two transactions involved in a dependency **exclusive** only allows to commit one of these transactions if the other on aborts. This case is represented by a bidirectional arrow with a crossed line. In contrast to the dependency **vital_dep**, the commit (not the abort) of one of the corresponding transactions leads to the abortion of the other transaction. Finally the ternary dependency **vital** is represented by three arrows connected over a small cycle (cf. Figure 6.2).

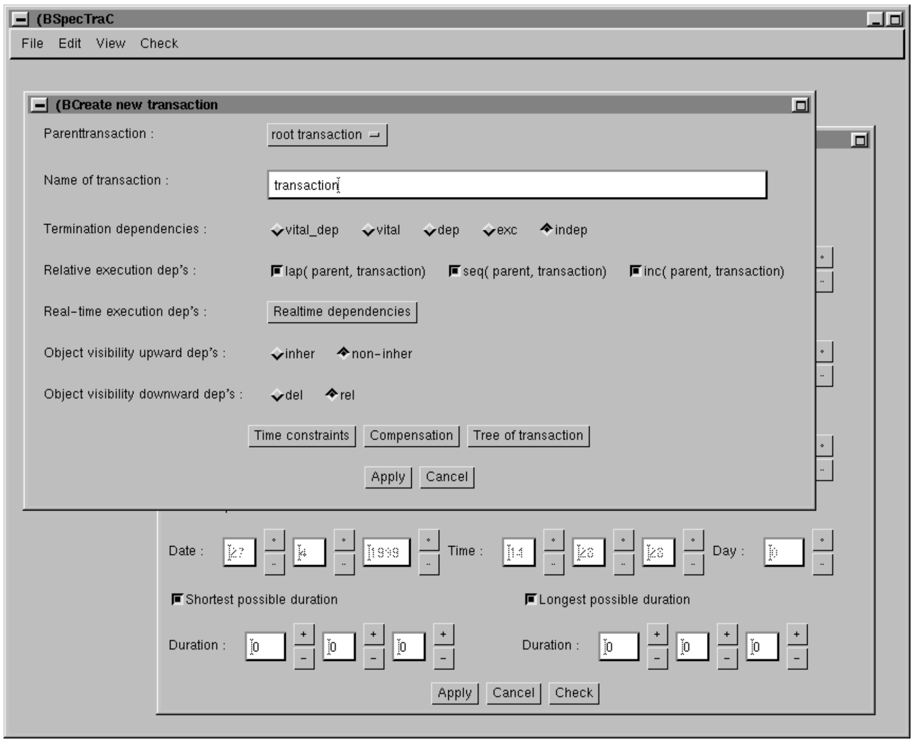


Figure 6.3: SpecTraC

The previous example shows that the assumption in the literature concerning the transaction management in federated database systems, e.g. [BGS92, AT91, BST92, GRS94], cannot be hold since local integrity constraints are not known and considered on the global level. The tool SpeCTraC[1] enables us to specify arbitrary sets of transactions on the global level (cf. Chapter 7). These are connected over transaction dependencies such as termination dependencies and are called transaction closures. Due to the global extensional assertions of the global schema, different commit rules can be derived for these global transactions.

[1]A Tool for **Spe**cifying **C**onsistent **Tra**nsaction **C**losures

# Chapter 7

# SIGMA$_{FDB}$-Tools

To verify and demonstrate the various algorithms introduced so far, we are implementing different tools. These tools support different issues of the federated database design process.

This includes programs for schema editing and viewing, definition of extensional relationships, and schema integration. In addition, we implemented a database loader to demonstrate the import of schema meta-data into our prototype. Most of the methods are part of the integrated Design-Toolset **SIGMA**$_{Bench}$. This toolset bases on the integration of database schemata using the GIM-approach. The **SIGMA**$_{Bench}$ is developed according to the client-server architecture. The central interoperation platform is the **SIGMA**$_{Bench}$ repository that is implemented using the relational DBMS *YARD* (cf. Figure 7.1). Clients for dedicated tasks are developed in *Java* and, thereby, provide platform independence.

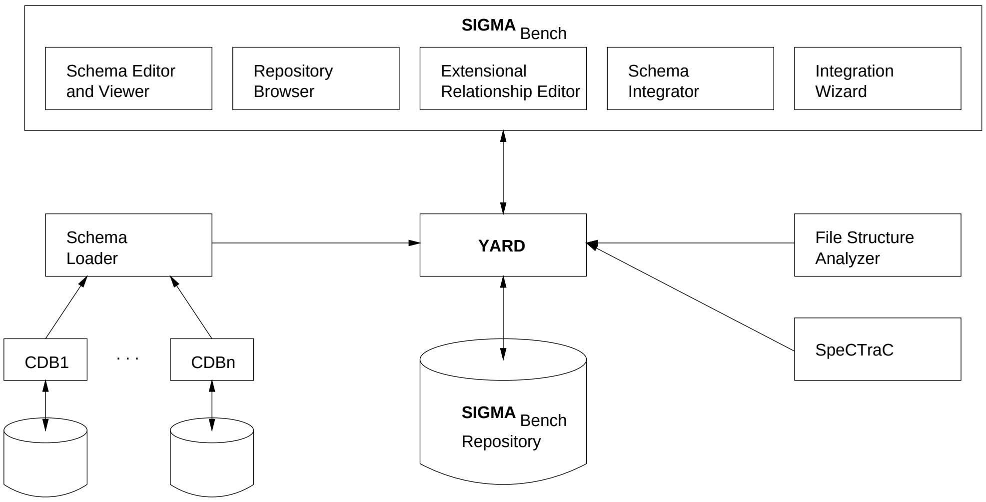


Figure 7.1: **SIGMA**$_{FDB}$-Tools

**Schema Editor and Viewer** The schema editor allows the definition of database schemata for storing them in the repository. Of course, in "real" integration scenarios the schemata should be imported from the local databases. Therefore, the main issue of the schema editor is to make schemata available for demonstration purposes. Thus, the editor offers beside classes, relationships, and specialization also integrity constraints. Currently, three kinds of integrity constraints are supported:

1. uniqueness constraints, e.g. **unique**($Employee.Name$)
2. comparison of an attribute and a constant, e.g. $Employee.Salary > 3000$
3. comparison among two attributes, e.g. $Book.ISBN \neq Journal.ISBN$

These constraints can even be composed to more complex integrity constraints using the logical operators AND, OR, and NOT.

Since its OMT-like graphical notation of classes is descriptive, the schema editor is also often used as a viewer of local and integrated schemata.

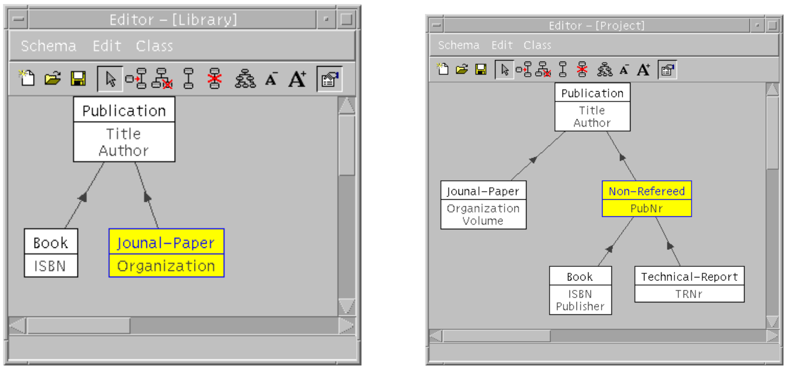


Figure 7.2: Editor with Schemata Library and Project

Figure 7.2 depicts the schemata "Library" and "Project" as defined in the schema editor of **SIGMA**$_{Bench}$.

**Repository Browser** This tool displays all schemata, classes, attributes, and integrity constraints stored in a database. Here, a tree-like graphical notation is used to depict the hierarchical relations among the mentioned objects. Even if this tool is not explicitly necessary for FDBS design, it is useful to explore available schemata before they are integrated. Currently, the repository browser is extended to cover local user profiles and access rights, as well.

**Extensional Relationship Editor** Manual definition of extensional relationships is a quite costly task. Different approaches to an intuitive graphical user interface were implemented and tested. In the current version we are implementing the derivation of extensional relationships from integrity constraints (e.g. Journal.ISBN<"1-2345-6789-0" **and** Book.ISBN>"2-6789-9876-4") and extensional assertions (e.g. Library.Book **is disjoint to** Project.Book). The extensional relationships are expressed by the GIM-matrix which correlates disjoint extensions and classes of the participating schemata. Our tool also allows the direct manipulation of the matrix.

**Schema Integrator** The internal part of **SIGMA**$_{Bench}$ is the integration tool which automatically derives integrated object-oriented schemata using the source schemata in the repository and the modeled extensional relationships. This tool covers extensional and intensional conflicts and supports the integration of integrity constraints. The integration of access rights is currently in development.

**Integration Wizard** To automate the design of federated schemata, some necessary steps (schema selection, definition of extensional relationships and assertions, GIM-matrix generation, selection of integration methodology, and presentation of the integrated schema) are combined in this tool. Thus, it improves the understandability of design of federated schemata and helps avoiding design errors.

Figure 7.3 depicts all mentioned steps implemented in the **SIGMA**$_{Bench}$integration wizard. This tool is particularly useful for demonstration purposes since it guides the user through the necessary steps of schema integration. We are currently working also on extensions to integrate local schema import/definition into this tool.

**Schema Loader** The schema loader imports schemata from relational databases into the **SIGMA**$_{Bench}$repository database using the catalogue tables of the underlying RDBMS. This includes tables, their attributes, data types, and integrity constraints. After importing a local database schema, it can be used for the integration. Since we use *Oracle* in our department for various tasks in administration, research, and teaching, and can therefore access a rather extensive, existing repository, we decided to support this particular DBMS at first. It should, however, be fairly simple to port this tool to other RDBMS, as well.

**File Structure Analyzer** The file structure analyzer comprises different tools for the file structure analysis. Key words, tokens, and brackets can be found semi-automatically based on text analysis. This way, the design of grammars, describing the structure of file clusters, is supported.

**SpeCTraC** This tool SpeCTraC allows to specify (global) transactions with different dependencies among them. The tool comprises a consistency checker which supports the designer in specifying consistent dependencies. If an inconsistency occurs, alternative dependencies are proposed. Due to the fixed extensional assertions of the global schema, different commit rules can be derived for global transactions (see also Chapter 6).

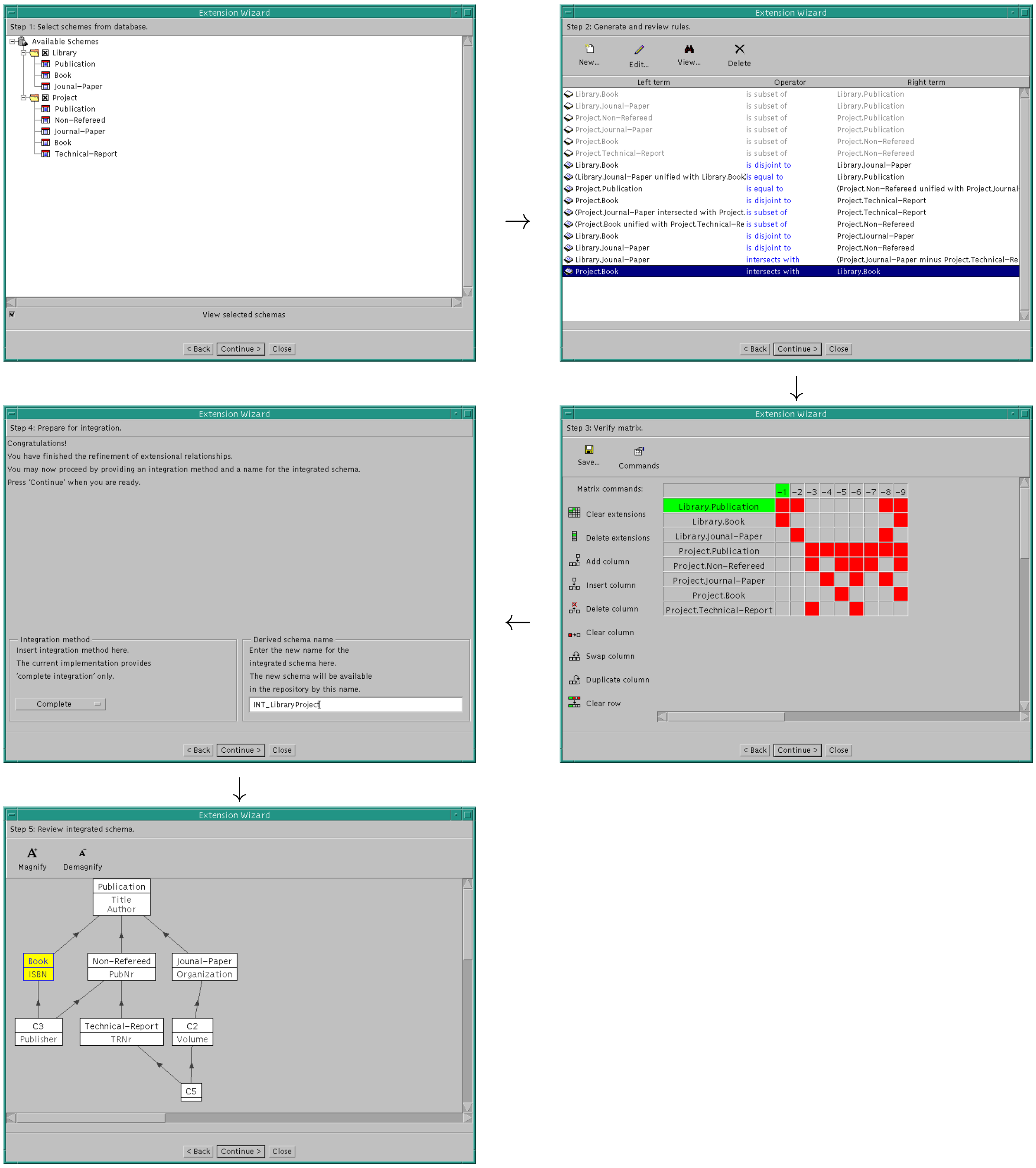


Figure 7.3: Design Steps in the Integration Wizard

# Chapter 8

# Conclusions

Building and maintaining federated databases is a hard task which cannot be done completely in an automatic way. Our main goal is to support the schema designer of a federated database during the design process. We work on different areas of this complex problem and developed different solutions. The online documentation of the **SIGMA**$_{Bench}$ can be found at:

http://wwwiti.cs.uni-magdeburg.de/~sigmafdb/

From a scientific point of view, there are still a lot of open problems in database integration. One main problem is the extraction of semantic information from existing systems. Our approach has done first steps in analyzing file structures, database schemata and existing explicit integrity constraints. However, schema re-engineering and extraction of implicit integrity constraints can never be completely automated implying a need for interactive semantic enrichment steps.

Besides the concrete federation aspects shortly described in this paper there are still a high number of other facets where methods and tool support are needed. Examples are integration of stored procedures, other databases models, general behavior integration, etc. Our aim is to build an open tool environment where developments in these areas can be 'plugged in' to enrich the design support offered by our tools.

# Bibliography

[ACM93] S. Abiteboul, S. Cluet, and T. Milo. Querying and Updating the File. In S. Baker, R. Agrawal, and D. Bell, editors, *Proc. of the 19th Int. Conf. on Very Large Data Bases, VLDB'93, Dublin, Ireland, August 24–27, 1993*, pages 73–84. Morgan Kaufmann Publishers, San Francisco, CA, 1993.

[Ade98] A. Adelberg. NoDoSE - A Tool for Semi-Automatically Extracting Structured and Semistructured Data from Text Documents. In L. Haas and A. Tiwary, editors, *SIGMOD'98, Proc. of the 1998 ACM SIGMOD Int. Conf. on Management of Data, June 1–4, 1998, Seattle, Washington, USA*, ACM SIGMOD Record, Vol. 25, pages 283–294, ACM Press, June 1998.

[AMN94] H. Ahonen, H. Mannila, and N. Nikunen. Generating Grammars for SGML Tagged Texts Lacking DTD. In *Proc. of PODP'94 – Workshop on Principles of Document Processing*, 1994.

[AQFG95] R. M. Alzahrani, M. A. Qutaishat, N. J. Fiddian, and W. A. Gray. Integrity Merging in an Object-Oriented Federated Database Environment. In C. Goble and J. Keane, editors, *Advances in Databases, Proc. of the 13th British National Conf. on Databases, BNCOD 13, Manchester, UK, July 1995*, Lecture Notes in Computer Science, Vol. 940, pages 226–248. Springer-Verlag, Berlin, 1995.

[AT91] R. Alonso and A. Tal. Dealing with Failures in Heterogeneous Databases. In *Int. Workshop on Interoperability in Multidatabase Systems, Kyoto, Japan*, pages 356–358, IEEE Computer Society Press, April 1991.

[BC86] J. Biskup and B. Convent. A Formal View Integration Method. In C. Zaniolo, editor, *Proc. of the 1986 ACM SIGMOD Int. Conf. on Management of Data, Washington, D.C.*, pages 398–407, ACM SIGMOD Record, Vol. 15, No. 2, ACM Press, June 1986.

[BGS92] Y. Breitbart, H. Garcia-Molina, and A. Silberschatz. Overview of Multidatabase Transaction Management. *The VLDB Journal*, 1(2):181–240, October 1992.

[BLN86] C. Batini, M. Lenzerini, and S. B. Navathe. A Comparative Analysis of Methodologies for Database Schema Integration. *ACM Computing Surveys*, 18(4):323–364, December 1986.

[BMR+95] B. T. Blaustein, C. D. McCollum, A. Rosenthal, K. P. Smith, and L. Notargiacomo. Autonomy and Confidentiality: Secure Federated Data Management. In *The Second International Workshop on Next Generation Information Technologies and Systems (NGITS'95), June 27–29, 1995, Naharia, Israel*, 1995.

[BST92] Y. Breitbart, A. Silberschatz, and G. R. Thompson. Transaction Management Issues in a Failure-Prone Multidatabase System Environment. *The VLDB Journal*, 1(1):1–39, July 1992.

[BT99] S. Balko and C. Türker. Integration of Aggregation Constraints. 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 5–24, infix-Verlag, Sankt Augustin, 1999.

[CDF97] S. Castano, S. De Capitani di Vimercati, and M. G. Fugini. Automated Derivation of Global Authorizations for Database Federations. *Journal of Computer Security*, 5(4):271–301, 1997.

[CFMS95] S. Castano, M. G. Fugini, G. Martella, and P. Samarati. *Database Security.* ACM Press, Addison-Wesley, Reading, MA, 1995.

[CHS+97] 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*, Lecture Notes in Computer Science, Vol. 1271, pages 200–214. Springer-Verlag, Berlin, 1997.

[Con86] B. Convent. Unsolvable Problems Related to the View Integration Approach. In G. Ausiello and P. Atzeni, editors, *Proc. of the 1st Int. Conf. on Database Theory, ICDT'86, Roma, Italy*, Lecture Notes in Computer Science, Vol. 243, pages 141–156. Springer-Verlag, Berlin, 1986.

[Con97] S. Conrad. *Federated Database Systems: Concepts of Data Integration.* Springer-Verlag, Berlin/Heidelberg, 1997. (In German).

[CST98] S. Conrad, I. Schmitt, and C. Türker. Considering Integrity Constraints During Federated Database Design. In S. M. Embury, N. J. Fiddian, A. W. Gray, and A. C. Jones, editors, *Advances in Databases, 16th British National Conf. on Databases, BNCOD 16, Cardiff, Wales, July 1998*, Lecture Notes in Computer Science, Vol. 1405, pages 119–133. Springer-Verlag, Berlin, 1998.

[DS97] S. De Capitani di Vimercati and P. Samarati. Authorization Specification and Enforcement in Federated Database Systems. *Journal of Computer Security*, 5(2):155–188, 1997.

[Dup94] Y. Dupont. Resolving Fragmentation Conflicts in Schema Integration. In P. Loucopoulos, editor, *Entity-Relationship Approach — ER'94, Proc. of the 13th Int. Conf. on the Entity-Relationship Approach, Manchester, UK, December 1994*, Lecture Notes in Computer Science, Vol. 881, pages 513–532. Springer-Verlag, Berlin, 1994.

[EHH99] A. Ebert, U. Hohenstein, and M. Höding. An Approach for Generating Comfortable File Interfaces. In C. L. Liu, A. Chen, and Lochowsky F. H.., editors, *Database Systems for Advanced Applications '99, Proc. of the 6th Int. Conf., DASFAA'97, Taiwan, ROC, April 19–22, 1999*, World Scientific Publishing, Singapore, 1999. *To appear.*

[EKP+95] W. Essmayr, F. Kastner, S. Preishuber, G. Pernul, and A.M. Tjoa. Access Controls for Federated Database Enviroments - Taxonomy of Design Choices. In R. Posch, editor, *Proc. of the Joint IFIP TC6 and TC11 Working Conf. on Communications and Multimedia Security, Graz, Austria*, pages 117–132, Chapman&Hall, London, 1995.

[GRS91] D. Georgakopoulos, M. Rusinkiewicz, and A. Sheth. On Serializability of Multidatabase Transactions through Forced Local Conflicts. In N. Cercone and M. Tsuchiya, editors, *Proc. of the 7th IEEE Int. Conf. on Data Engineering, ICDE'91, April 8–12, 1991, Kobe, Japan*, pages 314–323. IEEE Computer Society Press, Los Alamitos, CA, 1991.

[GRS94] D. Georgakopoulos, M. Rusinkiewicz, and A. Sheth. Using Tickets to Enforce the Serializability of Multidatabase Transactions. *IEEE Transactions on Knowledge and Data Engineering*, 6(1):166–180, February 1994.

[GW98] B. Ganter and R. Wille. *Formal Concept Analysis*. Springer-Verlag, Berlin/Heidelberg, 1998.

[HC98] M. Höding and S. Conrad. Data-Mining Tasks in Federated Database Systems Design. In T. Özsu, A. Dogac, and Ö. Ulusoy, editors, *Issues and Applications of Database Technology (IADT'98), Proc. of the 3rd World Conf. on Integrated Design and Process Technology, July 6-9, 1998, Berlin, Germany*, Vol. 2, pages 384–391, Society for Design and Process Science, 1998.

[HHSS98] M. Höding, R. Hofestädt, G. Saake, and U. Scholz. Schema Derivation for WWW Information Sources and their Integration with Databases in Bioinformatics. In W. Litwin, T. Morzy, and G. Vossen, editors, *Advances in*

*Databases and Information Systems, Proc. Second East-European Symposium, ADBIS'98, Poznań, Poland, September 1998*, Lecture Notes in Computer Science, Vol. 1475, pages 296–304. Springer-Verlag, Berlin, 1998.

[HL98] R. Himmelröder and B. Ludäscher. Querying the Web with FLORID. In M. H. Scholl, H. Riedel, T. Grust, and D. Gluche, editors, *Kurzfassungen — 10. Workshop "Grundlagen von Datenbanken", Konstanz (02.06.–05.06.98)*, Forschungsbericht No. 63, pages 94–98, Universität Konstanz, Fachbereich Informatik, 1998.

[Höd96] M. Höding. An Approach to Integration of File Based Systems into Database Federations. In *Heterogeneous Information Management, Prague, Czech Republic, 4–5 November 1996, Proc. of the 10th ERCIM Database Research Group Workshop*, pages 61–71, ERCIM-96-W003, European Research Consortium for Informatics and Mathematics, 1996.

[Höd97] M. Höding. Federating Databases and Files: An Approach to a Uniform and Comfortable Interface to Heterogeneous Tourist Information Systems. In A.-M. Tjoa, editor, *Information and Communications Technologies in Tourism 1997, Proc. of the Int. Conf., Edinburgh, Scotland*, pages 140–149. Springer-Verlag, Wien, New York, 1997.

[HP97] U. Hohenstein and V. Pleßer. A Generative Approach to Database Federation. In D. W. Embley and R. C. Goldstein, editors, *Conceptual Modelling — ER'97, Proc. of the 16th Int. Conf., Los Angeles, California, USA, November 1997*, lncs, Vol. 1331, pages 422–435. Springer-Verlag, Berlin, 1997.

[HS98] E. Hildebrandt and G. Saake. User Authentication in Multidatabase Systems. In R. R. Wagner, editor, *Proc. Ninth Int. Workshop on Database and Expert Systems Applications, August 26–28, 1998, Vienna, Austria*, pages 281–286, IEEE Computer Society Press, 1998.

[Hul86] R. Hull. Relative Information Capacity of Simple Relational Database Schemata. *SIAM Journal on Computing*, 15(3):856–886, 1986.

[JD94] D. Jonscher and K.R. Dittrich. An Approach For Building Secure Database Federations. In J. B. Bocca, M. Jarke, and C. Zaniolo, editors, *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94)*, pages 24–35, Morgan Kaufmann Publishers, San Francisco, CA, 1994.

[JD96] D. Jonscher and K.R. Dittrich. Argos – A Configurable Access Control System for Interoperable Enviroments. In D. L. Spooner, S. A. Demurjian, and J. E. Dobson, editors, *Database Security, IX: Status and Prospects, Proc. of the IFIP WG 11.3 Working Conf. on Database Security, Rensselaerville, New York, USA, 13–16 August, 1995*, pages 43–60, Chapman&Hall, London, 1996.

[LL95] M. L. Lee and T. W. Ling. Resolving Structural Conflicts in the Integration of Entity-Relationship Schemas. In M. Papazoglou, editor, *OOER'95: Object-Oriented and Entity-Relationship Modeling, Proc. of the 14th Int. Conf., Gold Coast, Australia, December 1995*, Lecture Notes in Computer Science, Vol. 1021, pages 424–433. Springer-Verlag, Berlin, 1995.

[LNE89] J. A. Larson, S. B. Navathe, and R. Elmasri. A Theory of Attribute Equivalence in Databases with Application to Schema Integration. *IEEE Transactions on Software Engineering*, 15(4):449–463, April 1989.

[MEKS92] J. G. Mullen, A. K. Elmagarmid, W. Kim, and J. Sharif-Askary. On the Impossibility of Atomic Commitment in Multidatabase Systems. In P. A. Ng, C. V. Ramamoorthy, L. C. Seifert, and R. T. Yeh, editors, *Proc. of the Second Int. Conf. on System Integration, ICSI'92*, pages 625–634. IEEE Computer Society Press, Los Alamitos, CA, 1992.

[MIR93] R. J. Miller, Y. E. Ioannidis, and R. Ramakrishnan. The Use of Information Capacity in Schema Integration and Translation. In S. Baker, R. Agrawal, and D. Bell, editors, *Proc. of the 19th Int. Conf. on Very Large Data Bases, VLDB'93, Dublin, Ireland, August 24–27, 1993*, pages 120–133. Morgan Kaufmann Publishers, San Francisco, CA, 1993.

[NG82] S. B. Navathe and S. G. Gadgil. A Methodology for View Integration in Logical Database Design. In D. MacLeod and Y. Villasenor, editors, *Proc. of the 8th Int. Conf. on Very Large Data Bases, VLDB'82, Mexico City, September 8–10, 1982*, pages 142–164, Morgan Kaufmann Publishers, 1982.

[PBE95] E. Pitoura, O. Bukhres, and A. K. Elmagarmid. Object Orientation in Multidatabase Systems. *ACM Computing Surveys*, 27(2):141–195, June 1995.

[Per93] G. Pernul. Canonical Security Modeling for Federated Databases. In D. K. Hsiao, E. J. Neuhold, and R. Sacks-Davis, editors, *Interoperable Database Systems, Proc. of the IFIP WG 2.6 Database Semantics Conf., DS-5, Lorne, Victoria, Australia, November, 1992*, pages 207–222. North-Holland, Amsterdam, 1993.

[RPG95] M. P. Reddy, B. E. Prasad, and A. Gupta. Formulating Global Integrity Constraints during Derivation of Global Schema. *Data & Knowledge Engineering*, 16(3):241–268, July 1995.

[RR97] V. Ramesh and S. Ram. Integrity Constraint Integration in Heterogeneous Databases: An Enhanced Methodology for Schema Integration. *Information Systems*, 22(8):423–446, 1997.

[Sch98] I. Schmitt. *Schema Integration for the Design of Federated Databases*, Dissertationen zu Datenbanken und Informationssystemen, Vol. 43. infix-Verlag, Sankt Augustin, 1998. (In German).

[SCS99] G. Saake, S. Conrad, and I. Schmitt. Database Design. In J. G. Webster, editor, *Wiley Encyclopedia of Electrical and Electronics Engineering*, pages 540–567, Vol. 4, John Wiley & Sons, 1999.

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

[SL90] A. P. Sheth and J. A. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, 22(3):183–236, September 1990.

[SP91] S. Spaccapietra and C. Parent. Conflicts and Correspondence Assertions in Interoperable Databases. *ACM SIGMOD Record*, 20(4):49–54, December 1991.

[SPD92] S. Spaccapietra, C. Parent, and Y. Dupont. Model Independent Assertions for Integration of Heterogeneous Schemas. *The VLDB Journal*, 1(1):81–126, July 1992.

[SS96] I. Schmitt and G. Saake. Integration of Inheritance Trees as Part of View Generation for Database Federations. In B. Thalheim, editor, *Conceptual Modelling — ER'96, Proc. of the 15th Int. Conf., Cottbus, Germany, October 1996*, Lecture Notes in Computer Science, Vol. 1157, pages 195–210. Springer-Verlag, Berlin, 1996.

[SS98] I. Schmitt and G. Saake. Merging Inheritance Hierarchies for Database Integration. In M. Halper, editor, *Proc. of the 3rd IFCIS Int. Conf. on Cooperative Information Systems, CoopIS'98, August 20–22, 1998, New York, USA*, pages 322–331. IEEE Computer Society Press, Los Alamitos, CA, 1998.

[ST98] I. Schmitt and C. Türker. Refining Extensional Relationships and Existence Requirements for Incremental Schema Integration. In G. Gardarin, J. French, N. Pissinou, K. Makki, and L. Bougamin, editors, *Proc. of the 7th ACM CIKM Int. Conf. on Information and Knowledge Management, November 3–7, 1998, Bethesda, Maryland, USA*, pages 322–330. ACM Press, New York, 1998.

[STS98a] K. Schwarz, C. Türker, and G. Saake. Execution Dependencies in Transaction Closures. In M. Halper, editor, *Proc. of the 3rd Int. Conf. on Cooperative Information Systems, CoopIS'98, August 20–22, 1998, New York City, USA*, pages 122–131. IEEE Computer Society Press, Los Alamitos, CA, 1998.

[STS98b] K. Schwarz, C. Türker, and G. Saake. Transitive Dependencies in Transaction Closures. In B. Eaglestone, B. C. Desai, and J. Shao, editors, *Proc. of the 1998 Int. Database Engineering and Applications Symposium (IDEAS'98),* *July 8–10, 1998, Cardiff, Wales, UK*, pages 34–43. IEEE Computer Science Press, Los Alamitos, CA, 1998.

[TF97] Z. Tari and G. Fernandez. Security Enforcement in the DOK Federated Database System. In P. Samarati and R. S. Sandhu, editors, *Database Security, X: Status and Prospects, Proceedings of the IFIP WG 11.3 Working Conference on Database Security, Como, Italy, 22–24 July, 1996*, pages 23–42, Chapman&Hall, London, 1997.

[TS98a] C. Türker and G. Saake. Deriving Relationships between Integrity Constraints for Schema Comparison. In W. Litwin, T. Morzy, and G. Vossen, editors, *Advances in Databases and Information Systems, Proc. Second East-European Symposium, ADBIS'98, Poznań, Poland, September 1998*, Lecture Notes in Computer Science, Vol. 1475, pages 188–199. Springer-Verlag, Berlin, 1998.

[TS98b] C. Türker and G. Saake. Exploiting Integrity Constraints and Extensional Relationships for Semantic Schema Integration. Preprint No. 15, Fakultät für Informatik, Universität Magdeburg, 1998.

[TS99a] C. Türker and G. Saake. Consistent Handling of Integrity Constraints and Extensional Assertions for Schema Integration. In *Advances in Databases and Information Systems, Proc. Third East-European Symposium, ADBIS'99, Maribor, Slovenia, September 1999*, Lecture Notes in Computer Science. Springer-Verlag, Berlin, 1999. *To appear.*

[TS99b] C. Türker and K. Schwarz. Dependencies between Transactions in Federated Database Systems. In A. P. Buchmann, editor, *Datenbanksysteme in Büro, Technik und Wissenschaft, BTW'99, GI-Fachtagung, Freiburg, März 1999*, Informatik aktuell, pages 271–290. Springer-Verlag, Berlin, 1999. (In German).

[TSS99] C. Türker, K. Schwarz, and G. Saake. Commit Protocols for Global Transactions in Federated Database Systems. Preprint No. 2, Fakultät für Informatik, Universität Magdeburg, 1999.

[Tür99] C. Türker. *Semantic Integrity Constraints in Federated Database Schemata.* Dissertation, University of Magdeburg, Germany, 1999. *In preparation.*

[VA96] M. W. W. Vermeer and P. M. G. Apers. The Role of Integrity Constraints in Database Interoperation. In T. M. Vijayaraman, A. P. Buchmann, C. Mohan, and N. L. Sarda, editors, *Proc. of the 22nd Int. Conf. on Very Large Data Bases, VLDB'96, Bombay, India, September 3–6, 1996*, pages 425–435. Morgan Kaufmann Publishers, San Francisco, CA, 1996.

[WL97] S. Weibel and C. Lagoze. An Element Set to Support Resource Discovery - The State of the Dublin Core. *Journal on Digital Libraries*, 1(2):176–186, 1997.

[YP95] J. Yang and M. Papazoglou. A Frame-Based Approach for Interoperation Support in Multidatabase Systems. In N. Revell and A. M. Tjoa, editors, *Database and Expert System Applications, Proc. of the 6th Int. Conf., DEXA'95, London, UK, September 1995*, Lecture Notes in Computer Science, Vol. 978, pages 48–57. Springer-Verlag, Berlin, 1995.