Federated Information Systems: Concepts, Terminology and Architectures

Susanne Busse, Ralf-Detlef Kutsche, Ulf Leser, Herbert Weber

Technische Universität Berlin, Fachbereich 13 Informatik
Computergestützte InformationssystemeCIS
Einsteinufer 17, D- 10587 Berlin
Email: {sbusse,rkutsche,leser}@cs.tu-berlin.de

Abstract

We are currently witnessing the emerging of a new generation of software systems: Federated information systems. Their main characteristic is that they are constructed as an integrating layer over existing legacy applications and databases. They can be broadly classified in three dimensions: the degree of autonomy they allow in integrated components, the degree of heterogeneity between components they can cope with, and whether or not they support distribution. Whereas the communication and interoperation problem has come into a stage of applicable solutions over the past decade, semantic data integration has not become similarly clear.

This report clarifies definitions and terms in the research area of integration of heterogeneous and distributed information systems. It gives classification criteria for such systems and particularly defines mediator-based information systems as we understand them. The definition of terms is accompanied by the identification of relevant concepts and reference architectures. We also closely relate our work to a number of other prominent classifications.

Contents

1 Introduction

The integration of data that is stored in different systems is an active research topic since many years. While in the 80th the focus was mainly to get data spread over many, proprietary and incompatible applications into a centralized database management system, the 90th have brought the urgent need to combine data stored in different DBMS. Some of the first work on such **multidatabase systems** was reported 1985 in [HM 85]. Later, the term **federated database** emerged to characterize techniques for proving an integrated access to a set of distributed, heterogeneous and autonomous databases [LMR 90], [SL 90]. Especially the work reported in [SL 90] has since then been used as a reference for further research. [SL 90] defines the nowadays classical 5-layer architecture of federated databases systems and clarifies important terminology, such as differing between multidatabase systems and federated systems and between loosely and tightly integrated systems.

However, the requirements for data integration have changed in the last years. In parallel, the techniques and methods have been further improved. For instance, **technical problems** of integration, emerging for instance from missing or incompatible networks, have almost disappeared with the success of the Internet. Physical distribution has also become manageable with standard tools such as CORBA and Java. In parallel, the number of potential data sources has increased tremendously, mainly due to the ubiquitous use of the World Wide Web as **data publishing** system. The pure number of sources, together with the rather limited capabilities of Web technology to cope with data access, have given the problem of heterogeneity new facets. For instance, schema integration is not an appropriate technology any more if data is not described through schemas, but rather through formats [ACM 93], and classical view mechanisms fail if data is not accessible through a query language [GMY 99]. Furthermore, an integration of web data sources is usually pursued without even notifying the sources of their "integration" in a federated information system. This gives the problem of autonomy a new dimension: if such a source changes its schema (or format), it can not notify integrating systems, even if it would be willing to do so. It is also increasingly acknowledged that the bulk of data produced for instance in scientific environments is simply not stored in database systems [MKTZ 99] but in different kinds of flat-file collections, and that new techniques are necessary for their integration.

At the same time, many paradigms of software development in general have changed. Centralized, monolithic applications are now considered dinosaurs, since modern systems are increasingly build from interoperating, but independently developed **components**. Client-server architectures are substituted by n-tier systems that exchange data in a number of different protocols (see fig. 1). Furthermore, it is now in general accepted that software systems as such should be, from the very start, build with their evolution in mind. This has many reasons, such as that their development requires a large investment and that they are frequently mission-critical components of a company [MW 98]. Experience has shown at the same time that evolution of software, i.e. the adoption to changing requirements and environments, is an extremely thorny and difficult topic. Therefore, software systems should explicitly take provisions to **react on change**. This is particularly true for data integration systems which have many potential incentives for changes: simply all the sources they integrate.

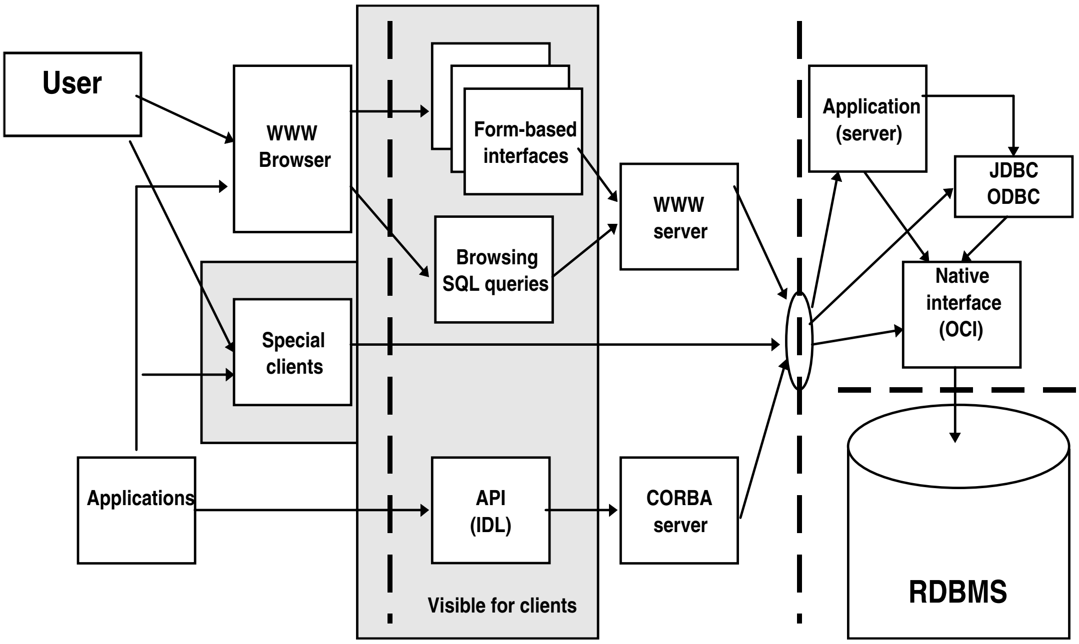

Fig. 1: Typical scenario of database access in current distributed environment.[1]

These developments have modified the requirements for database integration techniques. The very influential paper of Wiederhold [Wie 92] has sketched a scenario of a manifold of light-weight integration components, called **mediators**, that access sources or other mediators on-demand through interactive protocols. Sources are encapsulated by software adapters, called **wrapper**, which are able to present data in the form that a client, e.g. a mediator, needs it. These ideas of light-weight systems comprised of semiautomatics, but interacting components have found their way into many of the current research projects [ARPA 95].

We therefore believe that it is time to revise terminology and to try to systemize recent developments in the field of data integration[2]. This report is a first step in this direction. We do not want to replace existing criteria, as for instance given in [SL 90] or [Con 97], but refine and augment them, since we found that they are not sufficiently precise to characterize current systems. We therefore suggest several new distinguishing criteria to differ between different approaches. We also use **evolvability** of information systems as a primary dimension of their characterization, and we analyse several architectures in this light. Using this enriched set of distinguishing elements, we give precise definitions for common terms such as **federated database** or **mediator-based system**.

1. Databases are accessed through native call interfaces and also through special-purpose applications or application servers. CORBA and WWW server access the database either directly or through the application server. Remote accesses use e.g. WWW forms to submit (canned) queries or browse data, or make use of the CORBA API defined in OMG IDL. Proprietary clients might not use middleware but access the database through sockets. Dotted lines mark typical transitions in physical location. Arrows describe different possibilities. From the clients perspective, only the shaded area is visible.
2. The first “change” is the title: we deliberately use the term *federated information system* instead of *federated database system* to include sources that are not databases.

Structure

This report is structured as follows: chapter 2 revisits the three classical dimensions of the integration problem and adds evolvability as a fourth. We also distinguish the concept of federated database systems from distributed systems. Chapter 3 defines ten criteria that we use to characterize integration systems. Chapter 4 uses these criteria to precisely define our understanding of popular terms such as mediator-based information system. We also use these criteria to characterize five well-known integration projects. In chapter 5 we treat one particular problem of integration in more detail, namely the processing of global queries, and discuss several methods how this can be achieved. We compare our results with the work done by others and in other research directions in chapter 6. Finally, chapter 7 concludes.

2 Information Systems

As a trivial definition, an information system provides access to information, based on data managed somehow and somewhere in the system. Information systems can be characterized relative to the dimensions of autonomy, heterogeneity, distribution, and flexibility for evolution. A classification according to these dimensions will lead us to a first definition of federated information systems, being the focus of this paper. In particular, we shall see that heterogeneity is a very important aspect in this context which can be resolved using metainformation concepts.

2.1 Dimensions for IS classification

2.1.1 Autonomy

When the information system bases on multiple sites (called components[3] or component systems), the autonomy that integrated components may retain becomes a critical issue. We can distinguish several kinds of autonomy ([ÖV 99]):

- **Design autonomy** means that a component is independent from others in its design, regarding issues such as its universe of discourse, data model, naming concepts and so on. Design autonomy also entails the autonomy to change the design at any point in time, which is particularly difficult to handle within infrastructures of interoperating components.
- **Communication autonomy** is given when a component can decide independently with which other systems it communicates. Within federations of interoperating components communication autonomy means that each component can leave and enter the federation at any time.
- **Execution autonomy** denotes the component's independence in execution and scheduling of incoming requests. Execution autonomy is almost impossible to retain if a global trans-

3. In this context the term "component" denotes one source system participating in the federation. This is the usual understanding of this term in the database literature (see e.g. [SL 90]). It is differently used in other communities, such as software engineering, where it stands for a piece of software with certain properties. To discern the two concepts, we will use the term "software component" when we mean a part of a software architecture.

action management is involved.

Sometimes, association autonomy is discussed as a fourth type of autonomy in the literature. We refrain from doing so since we understand association autonomy as a mixture of design and communication autonomy.

2.1.2 Heterogeneity

Heterogeneity naturally occurs due to the fact that an autonomous development of systems always yields different solutions, for reasons such as different understanding and modeling of the same real-world concepts, the technical environment, particular requirements on the application, such as high performance for special queries, etc. Bridging heterogeneity is one of the main tasks of integration.

The literature has many classifications of heterogeneity on different levels of detail.[4] They sometimes coincide and sometimes differ. We distinguish syntactical, data model and logical heterogeneity with several subtypes:

Syntactical Heterogeneity

- **Technical heterogeneity** concerns differences in the technical aspects, like hardware platforms and operating systems (**hardware heterogeneity**), or access methods, such as:
    - Protocol, e.g. HTTP, SQL*Net, ODBC, CORBA etc.
    - Stateless or state-carrying connection
    - Security and log-on procedures
- **Interface heterogeneity** exists if different components are accessible through different access languages. We herein do not mean the technical aspect, such as whether a JDBC or a native interface is used, but refer to restrictions of the possible access methods. This includes:
    - **Language heterogeneity**: different query languages and language restrictions, e.g. no negation, no disjunctive conditions etc.
    - **Query restrictions**: only certain queries are allowed, only certain conditions can be expressed, joins can only have up to three relations, etc. [GMY 99], [TS 97]
    - **Binding restrictions**: certain attribute values must be specified to form a valid query. Although one could see this as a special form of query restrictions, we put it separately since it requires different techniques [RSU 95].

Data model heterogeneity

- **Data model heterogeneity** captures the fact that different data models have different semantics for their concepts. For instance, the relational model has no inheritance in contrast to object-oriented models.
  Although data model heterogeneity is a semantic conflict, we treat it separately because most systems handle it apart from other semantic conflicts, since data models typically contain only a handful of basic concepts. For instance, mediator-based systems require that wrappers hide data model diversity, but not semantic heterogeneity; the classical 5-

4. See e.g. [Wie 93] for an overview, and [Con 97], [Kim 95], [SP 91], [KS 95], [VJB+ 97] as classifications of logical heterogeneity.

layer architecture for federated databases has a separate layer to transform data models (the component schema).

Logical Heterogeneity

- **Semantic heterogeneity** concerns the semantic of the data and schema.

  Even schemas formulated in the same model can have different semantics[5]. Let us assume a schema consisting of relations and attributes. These relations and attributes have names and carry an implicit semantic, which is the **concept** they stand for. However, the schema contains in first place only the name, but not the concept. The interpretation of names by different people does not necessarily coincide. Therefore, different semantic conflicts can occur: equal names that denote different concepts (**homonyms**), different names for the same concept (**synonyms**), diverging understanding of a concept itself, etc. [VJB+ 97]. Attributes can also have the same semantic, but different units (DM or USD for prices)[6].

  The semantic of **data** values is defined through the schema element under which they are presented. We herein argue that the schema is the main place for encoding the semantic of a data source. Values do not carry semantic meaning, but are completely described through the schema portion that they belong to (i.e. the attribute and relation). This does not contradict the fact that in existing systems it might be necessary to analyze the values to extract the semantic of the schema.

- **Schematic heterogeneity** is the encoding of concepts at different elements of a data model. In the relational model, three types of such conflicts exist: relation ↔ attribute name, attribute name ↔ attribute value, and relation ↔ attribute value [KLK 91], [Mil 98]. An example for a “attribute name ↔ attribute value” conflict is shown in the tables below: While the first table models the courses a professor teaches as attribute names, the second table models them as values of the attribute ‘Course’.

Prof Name	Logic	DB	GIM
Maier	X		
Mueller	X	X	
Peters			X

Prof Name	Course
Maier	Logic
Mueller	Logic
Mueller	DB
Peters	GIM

Table 1: ’attribute name ↔ attribute value’ conflict

  Dealing with schematic heterogeneity typically requires query languages that are ‘syntactically second-order (of predicate logic), but semantically first-order’ [LSS 93], because they need variables that range over attribute or table names (examples are CPL [BDH+ 95], SchemaSQL [LSS 96], MSQL [LMR 90]). They are semantically first order because they always address a fixed schema.

- **Structural heterogeneity** exists if elements have the same meaning, are modeled with the same data model, and are schematically homogeneous, but structured in a different way, e.g. attributes are grouped into different tables.

5. We mean ‘intuitive semantic’, e.g. the meaning of a concept in the mind of a human being. Most data models do not allow to define the semantic of concepts in this manner. It can typically be found in documentation and handbooks.
6. Some authors call this ‘representation heterogeneity’ (e.g. [SPD 92]).

2.1.3 Distribution

A third, orthogonal problem to the autonomy and heterogeneity in data sources is their physical distribution. Since most computers are nowadays connected to some type of network, especially the Internet, it is natural to think of combining application and data sources that are physically located on different hosts, but that can communicate through the network.

Distribution is attacked by a number of successful techniques, such as HTTP or CORBA. Using for instance CORBA, applications can be developed that can ignore the physical location of components to a large degree. We therefore do not treat distribution as a separate problem in this report and assume that data is distributed over components which share neither memory nor disk ("shared nothing", [Dad 96], [Rah 94]).

2.2 Classification of Information Systems

Based on the two dimensions of distribution and heterogeneity, we separate three broad classes of information systems (IS):

- A **single (monolithic, centralized) information systems (SIS)** runs as one monolithic application on one computer. It offers one or more interfaces to its content. Single information systems can be:
    - **Database systems** which use a DBMS to store and manage data. In particular, it is based on a **data model** (such as relational, hierarchical or object-oriented); the data is structured according to a **schema**; and it is accessed through a **query language** (such as SQL or OQL).
    - **Non-database systems**, such as file systems, document collections, flat-file collections etc. These systems are usually not based on a standard data model and do usually not offer a query language. A non-database information system is **semi-structured** if the data it contains is not bound to a pre-defined structure; however, a concrete data item still may have a structure (which may be different from the structure of another item) [Bun 97].
- In a **distributed information system (DIS)** data is physically distributed over multiple sites which are connected with some kind of communication network.
- A **heterogeneous information system (HIS)** is a collection of information systems which differs in syntactical or logical aspects like hardware platform, data model or semantics.

If we add autonomy to the components, we come to our definition of federated information systems: A **federated information system (FIS)** consists of a set of distinct and autonomous information system components, the **participants** of this federation. Participants in first place operate independently, but have possibly given up some autonomy in order to participate in the federation.

Figure 2 shows the general three-tier architecture of a FIS. Application and users access a set of heterogeneous data sources through a **federation layer** which is a software component that offers a uniform way to access the data stored in data sources. The uniformity is reached with a specific interoperation strategy, e.g. this layer can offer a federated schema, a uniform query language, or a uniform set of source and content descriptions as metadata sets. The data sources usually are integrated into the infrastructure with wrappers which resolves some technical differences.

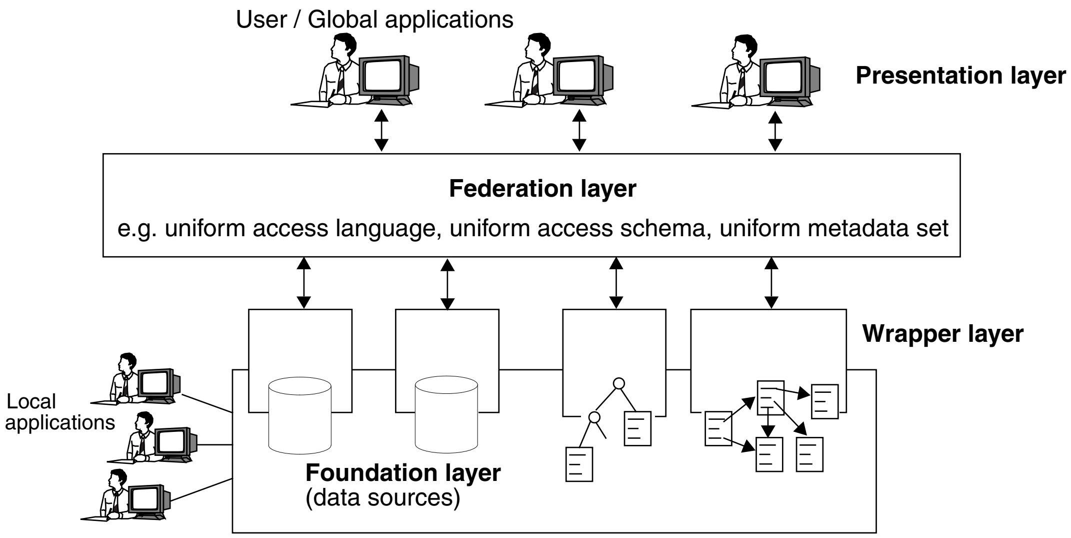

Fig. 2: Federated Information Systems

There are no strict borders between the classes of information systems described above. A FIS is typically constructed from a set of heterogeneous, semi-autonous and distributed components; however, if the components are for instance largely homogeneous, we would often still speak of a FIS. In this sense it is not clear what degree of heterogeneity components must reach for that the classification of an integrating layer changes from DIS to FIS. There are also no strict borders between database systems and non-database systems. Well-structured flat-file collections which enforce a strict format and offer some of declarative access language can be considered as, or at least be treated as, database systems [ACM 93]. On the other hand, DBMSs sometimes allow for BLOBs[7] and therefore give up the classical way of query capabilities w.r.t. parts of their contents.

2.3 Evolution of Information Systems

Besides the characteristics of autonomy, heterogeneity and distribution, most large-scale information systems solutions in business, science and administration today share another very important common characteristic as requirement for their future development: being subject to continuous change and evolution, they need capabilities of integrating legacy data sources and systems in an efficient and modular way (plug-in, plug-out). The concepts, architecture and realization of the whole information system, of its components and of the required services for integration (see fig. 2) have to allow for a high degree of maintainability and evolvability in the meaning of smooth change management.

Having the reference architecture of fig. 2 in mind, one recognizes two important reasons for evolution requests:

- at the bottom level, there will be a continuous change, emerging from newly developed,

7. BLOB means "binary large object", thus explicitly not showing the internal structure of its data content to the DBMS, e.g. as used in image processing, multimedia etc.

modified or additionally offered data sources. These new or changing components are in first place designed in the context of local requirements;

- from the top level, the desire for new information services will appear as soon as the global value of the given information structure is recognized by a relevant number of users, and an idea of possibly new sources exists.

The requests give rise to a number of requirements to the federation layer. In particular, it should be relatively easy to:

- integrate a new source, for instance by analysing its kind, structure and content and making it technically available to the global view by appropriate wrappers;
- develop a new service based on the knowledge of the information need on-top, the knowledge of the available contents at bottom, and relating these to each other;
- introduce new architectural components in the integration architecture, e.g. by recognizing more general patterns of wrapping or federation and therefore restructuring the "middleware".

In this report we abstract these requirements from the coding level towards the modelling level, where the most important contribution towards evolvable systems by appropriate modelling strategies and principles can be made, from which an architecture well-suited for evolution can be deduced, and then easily be realized.

2.4 Metadata for Information Systems

One important concept to deal with heterogeneity and distribution is the concept of metadata. Metadata is explicitly managed data describing other data or system elements to support their documentation, reusability and interoperation. As a result, this helps to develop more flexible and evolvable solutions.

Metadata is discussed in quite different contexts, see e.g. the initiatives to support retrieval in specific application domains[8] or metamodeling approaches (like CDIF or MOF [OMG 97]). Concentrating on federated information systems, especially the federation layer, we distinguish the following kinds of metadata:

1. **Technical metadata** describes information regarding the technical access mechanisms of components, such as the protocol, speed of connection, cost of queries, query capabilities and so on. It is used to bridge technical and interface heterogeneity.
2. **Logical metadata** relates to the schemas and their logical relationships. Logical metadata is e.g. available through the data dictionaries in RDBMS or as class diagrams in OODBMS. In particular, the relationships and dependencies between several schemas (of one data model) is one import kind of logical metadata in FIS.
3. **Metamodels** as metadata supports the interoperability of schemas in different data models. It addresses mainly data model heterogeneity.
4. **Semantic metadata** is information that helps to describe the semantic of concepts. In particular, **ontologies** and **thesauri** are used for this purpose. All domain-specific descriptions belong to this class.

8. See f.e. the dublin core initiative [DC 99] for document retrieval, the environmental data catalogue (UDK), FGDC [FGDC 94] for geo information, or HL7 [HL 7] in medicine.

5. **Quality-related metadata** describes source-specific properties of information systems regarding their quality, such as reliability, update frequency, actuality, comprehensiveness etc. This is used for ranking or optimization.
6. **Infrastructure metadata** helps users to find relevant data. This includes navigational aids like annotated bookmarks as well as a thesaurus structure (without concerning the thesaurus content – this is semantic metadata!). So, infrastructure metadata is used from the user or from information services of the presentation layer of a FIS.
7. **User-related metadat**a describes responsibilities and preferences of users of the information systems, e.g. user profiles.

A heterogeneous FIS (and only this one we will consider) typically use metadata in some aspects to gain flexibility. By describing component IS by a pre-defined set of metadata attributes it can implement generic methods to access these components. In all layers of a FIS some kind of metadata is needed. Usually it is stored as data in a **repository** (fig. 3).

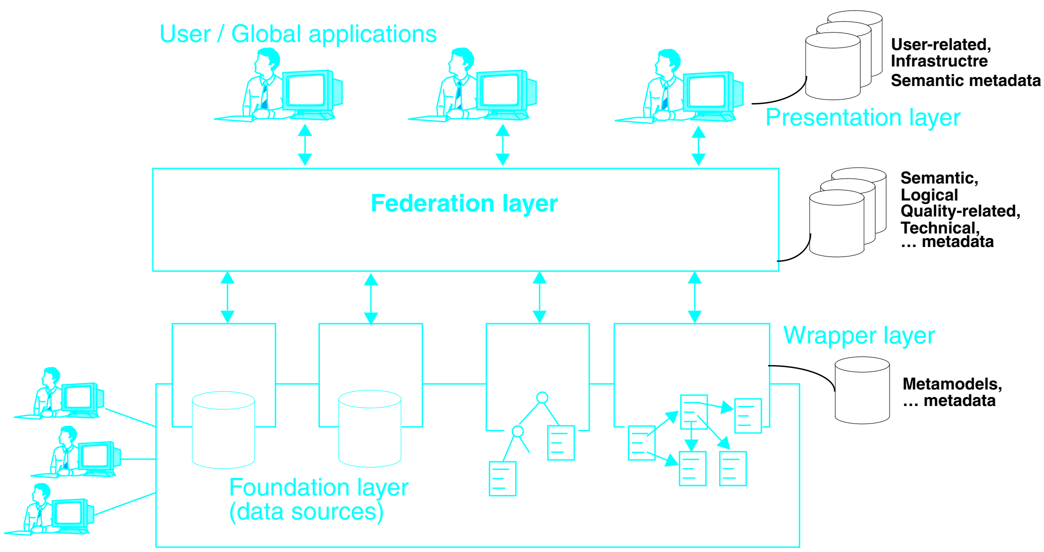

Fig. 3: Metadata for Federated Information Systems

3 Classification Criteria for Federated Information Systems

In the last chapter we identified autonomy as a common characteristic of components of federated information systems. Following "natural" tendencies, autonomous components will inevitably develop heterogeneous structures. It is the task of the federation layer to cope with the different types of heterogeneity.

In this section we discuss a number of criteria that are distinguishing between different approaches to the development of FIS. We will later use them to precisely characterize important classes of FIS (see next chapter). We distinguish FIS for instance by the kind of components they can integrate, the level of transparency that is achieved for a user, the degree of semantic integration and the overall development methodology. Although desirable, we find it impossible to give orthogonal criteria; therefore, some classifications regarding one dimension might require or induce a decision in another dimension as well. For instance, schema transparency requires the existence of an integrated schema; loose federations are necessarily build top-down; and IR-like query methods usually prevent a tight semantic integration. Nevertheless we found those criteria very useful to identify important properties of FIS.

3.1 Kinds of components: structured, semi-structured, unstructured

FIS differ in the types of components they can integrate. FIS can allow or disallow the integration of structured, semi-structured and unstructured components. Structured sources in our understanding have a pre-defined schema. All data items are intensionally defined through the schema element they belong to. Furthermore, the schema dictates the format of tall data items; items that do not fit into the schema can not be integrated into the data set.

A semi-structured data source has a structure, but this structure is not pre-defined in form of a strict schema [Bun 97]. Therefore, each single data item has to carry its own semantic definition, usually in the form of a label. At a given point in time, the sum of all labels of a semi-structured data source can be considered as its schema. However, this schema can potentially change every time new data is added, while in structured sources a schema change occurs much less frequently.

Unstructured data sources do not have any structure, such as textual documents.

3.2 Tight versus loose federation

We have so far not made any assumption about the form that the federation layer has. One can mainly distinguish two classes: tight and loose federations. A tight federation has a unified, global schema, which is the access schema for any user. A loose federation has no such schema, but only offers a unified language to query the data content of the components. Hence, tight federations offer schema, language and interface transparency (see below), while loose federation only offer the latter.

- **Tight federation** (fig. 4)

  A tight federation offers a unified schema (integrated or **federated schema**) as access interface to the federation. This schema can
  - be build through a (semi-)automatic schema integration process ([BLN 86], [Sch 98]) or can be created ad-hoc
  - semantically cover the components completely or only partially.

  In any case must the “semantic essence” of the federated schema be a subset of the union of the “semantic essences” of the component schemas. With semantic essence we mean the set of real-life concepts that are described through a schema.[9] This implies that a federated schema that is build in an ad-hoc fashion must also consider the content of the components to ensure this condition; but, in contrast to a schema integration process, it is not defined (and in particular not formalized) how they are considered.

9. This is similar to the notion of "information capacity" as used in [MIR 93].

The main task when using a federated schema is the resolution and handling of logical heterogeneity in the source schemas. It must be considered both during schema integration and during query processing. To ensure semantic equivalence, the FIS must know about **correspondences** (logical metadata) between queries, federated and component schemas. These correspondences can e.g. be expressed through ontological descriptions or rules. They can be defined by humans, by giving expressions in some language, or inferred automatically[10].

Tight federations are comfortable for users of the FIS, as they do not need to know about the schemas of all components, but only about the federated schema. On the other hand, users of a tight federation must rely on the translation mechanism and hence on the correspondences. Therefore, correspondences should be defined by a domain expert.

Using and offering global schemas are hence essential if very many sources are present, where it is impossible to expect every user to know all of them in detail, or if sources frequently evolve their schemas, and users cannot track all changes. Furthermore, certain situation unavoidably require the use of a global schema, namely, if standard schemas are used (e.g. STEP schemas [SM 98], [Sau 98] or OMG's domain standards, such as [BLL+ 99], [LSRDTF 98a]). In this cases, existing components must be fit into this standard schema.

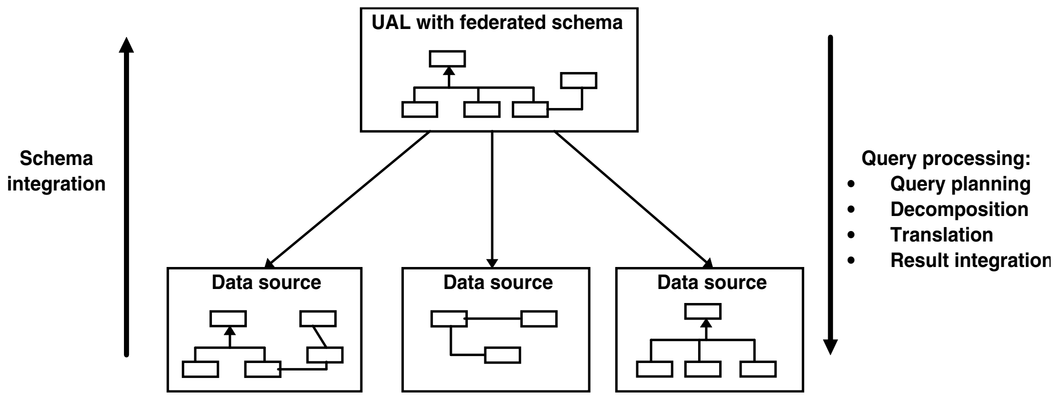

Fig. 4: Schemas in tightly integrated FIS

- **Loose federation**

  Loose federations do not offer a uniform schema for queries against the federation. But they offer a uniform query language (**multidatabase query language, MDBQL**) [LMR 90] which abstracts from the query languages of the components and hides technical and language heterogeneity. Hence, every user is itself responsible for handling logical heterogeneity in the components. To cope with schematic heterogeneity, MDBQL need to include the possibility to range over schema elements as if they were data [KLK 91], which is for instance not possible in SQL.

  Loose federations can only be build if the data sources themselves offer a query language access; binding patterns or access restrictions can not be modeled. In a tight federation,

10. We are not aware of any system that succeeds in this process. Cyc [Len 95] however is an attempt in this direction.

the federation can try to compensate for missing capabilities for sources; in a loose federation, this is much harder, since the query that is executed is directly determined by the user.

To avoid shifting the burden of the integration process completely to the users, systems based on MDBQL often use **integrating views**. This means that users can define views on the components and make them available for other users, which then use them as if they were global relations. These views are formulated in the MDBQL and *hardcode* the resolution of logical heterogeneity and result integration (see fig. 5).

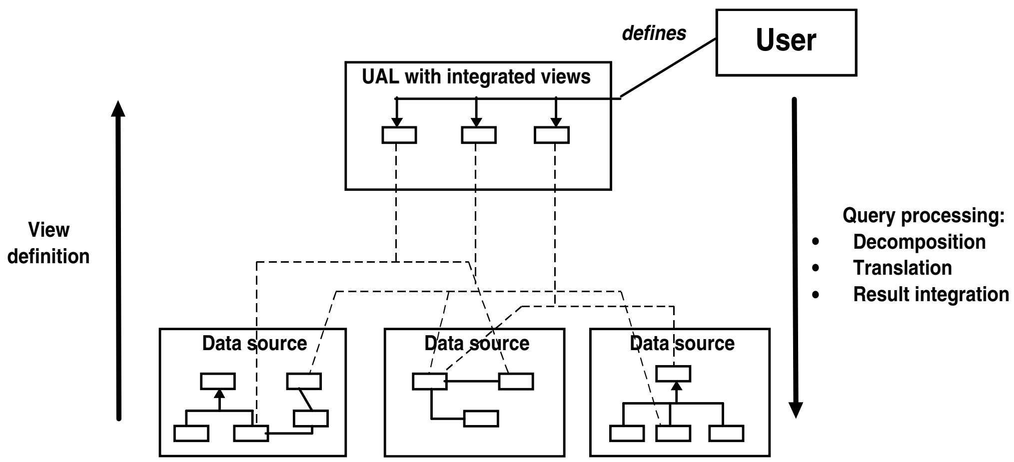

Fig. 5: Loosely integrated FIS

Loosely integrated systems that make extensive use of integrated views, possibly having them defined by domain experts, can almost be seen as tightly integrated FIS. The main criteria to discern tight from loose integration is therefore whether or not the component schemas are hidden for the users. In a tight integration, the source schemas are not visible any more; in a loose integration, source schemas are still visible, although additional views can be used to access them in a "pre-defined" fashion.

3.3 Data model of the FIS

The federation layer of a FIS must base on a specific data model, called the **canonical data model** [SL 90] or common data model [BLN 86]. The schemas of tight federations are schemas in this model. In loose federations, still the query language used to access the data sources bases on this data model.

The data model inherently restricts the kinds of components which can be integrated within the FIS, due to missing translation possibilities between some models. Table 2 gives the possible incompatibilities between popular types of data models. Incompatibility arises if semi-structured components shall be integrated into structured data. Problems also occur if semantically richer models shall be integrated into a poorer model. For instance, integrating object-oriented schemas into a relational schema is only possible with a loss of semantic knowledge. The other way round, in contrast, requires a process of **semantic enrichment** to obtain a proper representation [PBE 95], [HP 96], [SCG 91]. Note that the table assumes that it is tried to properly integrate data, i.e. without loss in semantic, and ignores e.g., the possibilities of canonical schemas.

	object-oriented FIS data model	**relational FIS data model**	**semi-structured FIS data model**
object-oriented component data model	+	(+)	(+)
relational comp. data model	+	+	+
semi-structured comp. data model			+

Table 2: Compatibility of global data model (horizontally) and component data models (vertically). A "(+)" indicates a potential semantic loss.

3.4 Kinds of semantic integration

The integration layer encapsulates data sources for the user. In contrast to loosely coupled FIS, where the data is in principle collected unchanged from the data sources, federation services of tightly coupled FIS may support a better semantic integration. We distinguish different kinds of semantic integration at the data level:

- **Collection**: Data of components are collected unchanged without matching equivalent data objects of different sources.
- **Fusion**: The integration of data of the component is done by a simple extraction (expressible with a query against the component schema); no further abstracting computations are done. But in contrast to a collecting approach, object fusion is performed to identify semantically equivalent entities coming from different sources [PAG 96][11] Furthermore, the FIS tries to determine a consistent representation, i.e., if sources report contradicting values for the same data item, such as different social security numbers for the same person, the FIS uses rules to remove the conflict [Ken 91b]. Note that data fusion is very difficult; frequently it is impossible to identify objects or to decide which data value is correct [Ken 91a].
- **Abstraction**: Hereby, the federated data base on extracted data of the components, but further functions may be applied to lift or lower the source data to the abstraction level of the federation schema. The need for abstraction is in general caused by semantic conflicts. It encompasses functions for aggregating data, reclassifying entities, or even more complex reasoning processes. Performing an abstraction during the integration implies that no writing operations are possible, since reverse operations are usually impossible to specify.
- **Supplementation**: Data is not only derived from data of the components, but some other data is added which describes the content or semantic context of the data (semantic metadata). Such an integration is used to handle implicit semantics of components. It is for instance necessary if data sources provide no schema, but the federation layer bases on a metadata schema.

11. Two data objects are semantically equivalent, if they describe the same real-world concept.

3.5 Transparency

We consider **transparency** for the end user as the ultimate goal of integration. A perfectly integrated information system would give the illusion that users interact with only one central, locally running, homogeneous and consistent information system. We distinguish the following types of transparency[12]:

- **Location transparency**: users do not need to know the physical location of information. This comprises the **host location**, e.g. defined by an IP number or host name, and the **name of the data source**, e.g. the name of the database if a RDBMS is used.
- **Schema transparency**: users do not need to know the different denotations that entities or attributes have in different data sources. Given a purely relational scenario, they do not need to be aware of the different relation and attribute names. In other words, all **logical conflicts** are masked. Clearly, schema transparency can only be achieved if a federated schema exists.
- **Language transparency**: users do not need to cope with different query mechanisms and languages. This comprises the **query language** and hence implicitly the data model, and the **access mechanism**, i.e. whether queries are finally executed by means of a declarative query language such as SQL or by some application methods started via RPC.

There is a clear relationship between the treatment of heterogeneity as described in section 2.1.2 and the level of transparency a FIS offers. Schema transparency comes down to hiding logical heterogeneity, while language transparency is achieved by hiding interface heterogeneity. Location transparency is related to technical heterogeneity.

Please note that achieving full transparency can be infeasible or even impossible. E.g., if a data source permits only certain types of queries, then it is sometimes impossible to compensate for this on the federation level. Compensation is in general only possible if the ‘complete’ data set is accessible, since then missing query capabilities in the source can then always be applied in a post-processing step – but downloading the complete data set can be prohibitively expensive. If the complete data set is not available, compensation can be impossible.

3.6 Query paradigm

Information systems are often classified by the types of queries they allow: **structured queries** or **information retrieval** (IR) queries. While the former assumes some structure in the information which is used to specify data items in a query, the latter performs similarity searches in documents (usually text documents, but research now also addresses information retrieval in multi-media objects such as video or sound). WWW search engines are typical IR-based systems; DBMS are typical query-based systems. A third class of query paradigms is the use of specific metadata that describes the data objects, though it might not be part of the data itself. For instance, document search might allow search criteria such as document size or creation date, which are not explicitly stored with the document.

12. *Transparency* in this context denotes the concept of *problem invisibility*; if the location is transparent, then an end-user cannot (or, at least, need not) see it.

3.7 Bottom-up vs. Top-down

We distinguish between two fundamental ways of engineering a tightly coupled FIS: top-down, i.e. starting with a global information need and later plugging in sources that can contribute to this need, or bottom-up, which starts from the integration requirement of a set of sources:

Top-down Strategy

Top-down approaches are build according to a global information need. For instance, a company might want to offer the service to find the lowest book prices from different Internet stores; or a decision support system might want to integrate certain customer information that is spread over multiple department databases. In these cases, the actual schema of components does not matter for the design of the federated schema. From the four classical requirements for schema integration, i.e., completeness, correctness, understandability and minimality, two do not apply. First, there is no need to include schemas completely, if only customer data is required (completeness). Second, there is no need to represent data on the global level exactly as in the components, if only derived or abstracted data is required (correctness; the mapping becomes uni-directional). One might for instance decide to cluster customers into salary groups and not to store the precise income.

The global schema in top-down approaches can either be generated ad-hoc or be the result of a more formal analysis process, starting from use-case descriptions and ending by view integration techniques. The global schema might also be prescribed by a standard (see section 3.2). In any case, component schemas are only considered in a second step, when correspondences between the global schemas and sources schemas are established to allow for the translation of queries.

Top-down approaches have many advantages in scenarios where sources are quickly evolving; if it often happens that sources are removed or new sources are added; if schema integration is infeasible or too expensive; or if the global requirements themselves are changing [Les 98a]. This is mainly because schema integration based approaches are extremely vulnerable to any changes [Mot 98]. Anyway, top-down approaches typically result in a less ‘tight’ integration than bottom-up approaches.

Bottom-Up Strategy

Building a tightly integrated FIS bottom-up means that the initial requirement is the need to have an integrated access to a given number of data sources. A typical scenario is the need to have detailed and uniform access to all databases of a company to build global applications, possibly ahead of a migration. The manager here would say: ‘Give me an integrated access to (exactly and completely) these databases’, while the typical sentence for a top-down development is rather ‘Give me an integrated access to all customer data (no matter where from)’. In such a setting there is a stronger need to guarantee completeness and correctness of the integrated schema, and therefore (semi-) formal integration techniques are more appropriate. Bottom-Up integration leads to semantically well integrated systems because the components are assumed to be known completely before the integration process. A change in the configuration triggers a new integration process.

A particular problem in the construction of FIS is the necessity to update data in data sources through the global schema. Clearly, updates are only possible if the connection between the global and the component schemas are very tight, i.e., if updates can be propagated uniquely.

Top-Down approaches rarely offer this possibility, and also many bottom-up approaches do not retain unique correspondences.

3.8 Virtual vs. materialized integration

Federations can be differentiated whether or not data of the components is persistently stored at the integration layer. **Virtual integration** architectures only temporarily materialize the result of queries at the time the query is posed. This requires a mechanism to **translate queries** against the federated schema into one or more semantically meaningful and executable queries against components which are propagated dynamically to the sources (see section 5).

The other extreme are architectures that **materialize** sources completely or partially on the federation level. They are well known from the context of **data warehousing**.

Materialization has advantages:

- High performance for queries against the materialized data set
- Control over materialized data, especially curation is possible [LMR 90]

and disadvantages:

- Keeping the data up-to-date requires possibly complex update procedures, especially if no differential snapshots are provided ([Wid 95], [GM 95])
- The federation layer has to provide a possibly large amount of storage space.

There are also hybrid architectures that materialize only selected types of data, e.g. object names are hooks into data sources (e.g. [ZHK+ 95]).

3.9 Read-only or read-and-write access

We distinguish between FIS that allow the insertion (or updating) of data into component systems through the federation layer and those that do not allow this. Write access is often disregarded in integration projects because of the following points:

- many interfaces, e.g. WWW interface, do not allow a write-through;
- writing through integrated views raises all problems of updating data through views [BSKW 91];
- writing through an integrated schema e.g. raises the question of which source should be used if a class is present in more than one data source;
- global transactions require complex protocols.

In general, write-through access decreases source autonomy to a high degree. It is therefore often not considered in FIS projects.

3.10 Required access methods

Especially if data is dynamically obtained from data sources (virtual integration), the different access methods of component systems (or their wrappers) have to be considered. FIS can be distinguished in how far they can cope with potential restrictions on this level. A client does typically have one or more of the following possibilities to access the data:

1. Access through a **query language**, e.g. SQL or OQL

   This access can be granted through the native interface, through ODBC/JDBC, through a special method in a CORBA API[13] or through a special form in a WWW interface.

2. Access through **parameterized canned queries** (PCQ)

   Parameterized canned queries are predefined queries with some variable positions. Speaking in SQL, a PCQ typically has a fixed select- and from-clause, fixed join-conditions in the where-clause, and fixed types of other conditions in the where-clause, but allows free specification of the values with which attribute values are compared in these conditions.
   Canned queries are typically used in IDL method calls (using function parameters for the variable positions), WWW forms (using entry-elements for the variable positions) or arbitrary API-procedures.

3. Access through **browsing**

   Especially in a WWW environment data can often only be browsed rather than searched. In those cases, a search is typically restricted to a navigational traversing of the data.

Differences in access methods concern two types of heterogeneity: technical heterogeneity and access language heterogeneity. From the logical point of view, only access language heterogeneity matters. Note that is in particular very difficult to consider e.g. restrictions in the possible queries in a multidatabase query language, where clients can in principle formulate arbitrary queries.

4 Types of Federated Information Systems

Based on the criteria given in the last section, we give a definition of three types of federated information systems: loosely coupled information systems, federated database systems, and mediator-based information systems.[14] The following table briefly describes their characteristics.

	Loosely coupled Information Systems	**Federated Databases**	**Mediator-based Information Systems**
Types of heterogeneity addressed	Only technical and language heterogeneity	All, except query restriction heterogeneity; schema integration difficult for schematic heterogeneity	All
Loss of autonomy?	Execution autonomy	Execution autonomy; notification of schema changes	Execution autonomy
Transparency	Language	Location, schema and partly language	Location, schema and language

13. It is debatable if this is reasonable, especially when considering the return type of such a method [LTB 98].
14. Loosely coupled information systems are often classified as one kind of FDBS [SL 90]. We consider them separately because the main difference "federated schema or not" has very important affects on the characteristics of the information system.

	Loosely coupled Information Systems	**Federated Databases**	**Mediator-based Information Systems**
Kind of components	Structured	Structured	Any
Access methods	Query language	Query language	Any
Access restrictions	No	No	Yes
Write access?	Yes	Yes	No
Tight vs. loose integration	Loose	Tight	Tight
Kinds of sem. integration	Collection	Collection and fusions	Collection, fusion, sometimes abstraction
Necessary metadata	Technical, infrastructure	Logic, technical, semantic	Logic, technical, semantic
Bottom-up vs. top-down	n.a.	Bottom-up	Top-down
Virtual vs. materialized	virtual	Virtual	Virtual
Evolvability	High	Low	High

Table 3: Kinds of federated information systems

From the table it is clear that it is difficult to give a tree-like classification as in [SL 90], p. 45. A draft for a classification can be found in Fig. 6.

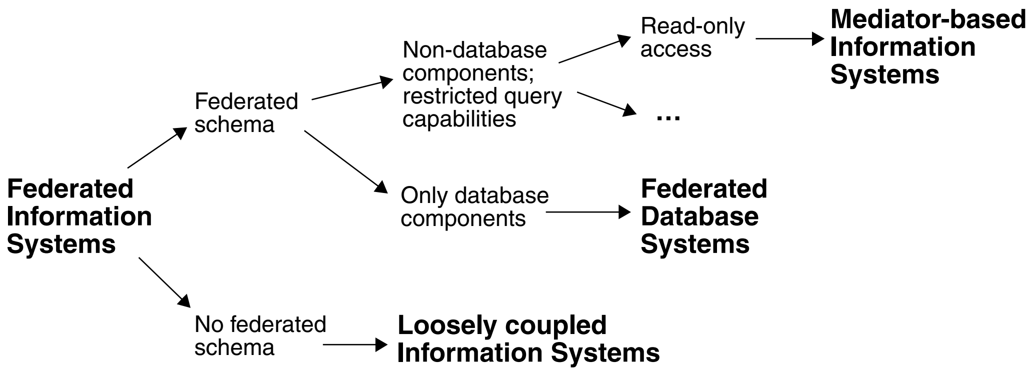

Fig. 6: Classification of Federated Information Systems

4.1 Loosely coupled Information Systems

Loosely coupled information systems do not offer a federated schema, but only a multidatabase query language to access the components. This has the advantage that components do not give up autonomy to participate in a federation. But on the other side, no location and schema transparency is offered: the user has to address the respective component and the particular element in the schema of the component within his queries.

Providing a uniform query language, technical and language heterogeneity is bridged by the FIS. All logical conflicts have to be resolved by the user or the services of the presentation layer. They are responsible for data integration with all its aspects of collection, fusion, and abstraction.

The federation layer is independent of the logical design of components. Since no global schema exists, changes of component schemas do not affect the system. But the missing logical integration leads to various dependencies between applications and component systems with all the negative effects on evolution known from two-tier systems.

In the literature loosely coupled information systems sometimes are called multidatabase systems (e.g. [Dad 96]). Other authors consider them as a special kind of federated database systems (see the next section). Because of the differences between loosely and tightly coupled FIS with their consequences to autonomy and evolution, we classify loosely coupled information systems separately.

4.2 Federated Database Systems (FDBS)

Federated database systems provide classical database system functionality. This includes a read-and-write access for data management. The term 'database' indicates the relationship to classical database systems: Components of federated database systems are structured sources, which are accessed through query languages. E.g. [SL 90] considers differences in the query language of components, but always assumes the existence of query language access. They do not consider restricted query capabilities.

Components typically give up some autonomy, e.g. notification of changes, access to logical metadata, or scheduling information for global transaction management.

Federated database systems are tightly coupled information systems. They are built bottom-up applying some schema integration techniques. The federated schema has to fulfill the requirements of completeness, correctness, minimality, and understandability [BLN 86], which is only possible with collection or fusion integration.

As a tightly coupled FIS, FDBS offer full location and schema transparency to their users. But FDBS typically have a static architecture with problems in system evolution because of the dependency on schema integration processes which do neither allow an easy plug-in or plug-out of components nor a flexible change management.

Figure 7 shows the classical 5-layer architecture of FDBS ([SL 90], p. 45). The right column defines the actions between the different layers. The left columns adopts terminology of the mediator community to this architecture (see next section).

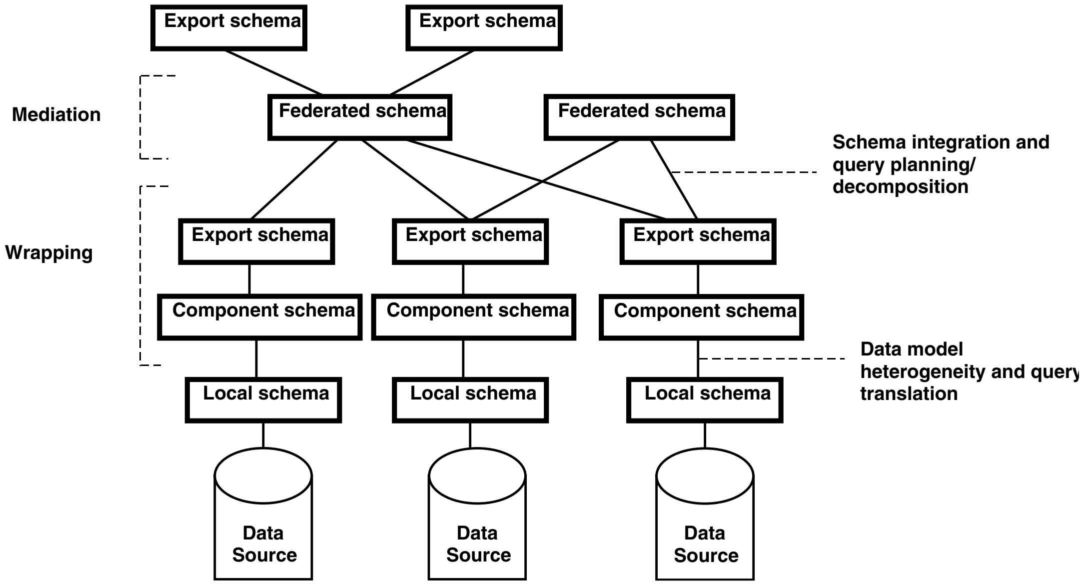

Fig. 7: Schema Architecture of FDBS

4.3 Mediator-based Information Systems

The term mediator was, to our knowledge, introduced by Wiederhold in [Wie 93] and is since then used in many publications on data integration projects and techniques. From the very first usage, there was no exact definition of a mediator, nor its relationship to FDBS. In general, a mediator should be a software component that mediates between the user and physical data sources. It should be **light-weight** ("manageable by a group, but not requiring a committee"), flexible and re-usable. In particular, mediators are designed to use other mediators as components; a feature that is also mentioned as a possibility in [SL 90], but was somehow ignored in the FDBS community.

Our definition of mediator-based information systems (MBIS) however rather considered the state-of-the-art of systems that call themselves "mediator-based" (see section 6.1), rather then on the original suggestions of Wiederhold (see section 6.2). One obvious difference between MBIS and FDBS is the read-only access to the data sources. MBIS are tightly coupled information systems, so a federated schema is used to provide integrated access to data of different components (semantic heterogeneity). In contrast to FDBS the federated schema is usually build top-down according to the information needs. Related to this is the understanding of mediators as **services** that are constructed and offered to costumers. This implies the important requirement of flexibility regarding evolution of the system. At least it must be possible to easily plug-in and plug-out components because of sources in a MBIS typically keep complete communication autonomy.

A further classification of MBIS reveals some differences. In the literature, both structured and semi- or un-structured components are considered. Heterogeneity of access methods is an important research topic, for instance regarding binding patterns (ubiquitous in the integration of WWW sources) and restricted query capabilities (important in many scientific domains, such as spatial databases). MBIS can integrate several integration mechanisms like abstraction, aggregation or metainformation approaches. Typically, a mediator does not address all these aspects, but it should handle at least one of them.

The figure below shows the classical mediator-wrapper architecture adapted from the general architecture for FIS (see fig. 8). The federation layer contains several mediators providing mediation services; therefore this layer is now called mediation layer. Each mediator has its own federated schema, and mediators can use other mediators as data sources (mediator networks). Wrappers hide technical and data model heterogeneity; how they access their data sources is transparent for a mediator. Queries against federated schemas (user queries) are dotted, queries against wrapper (source queries) are lines.

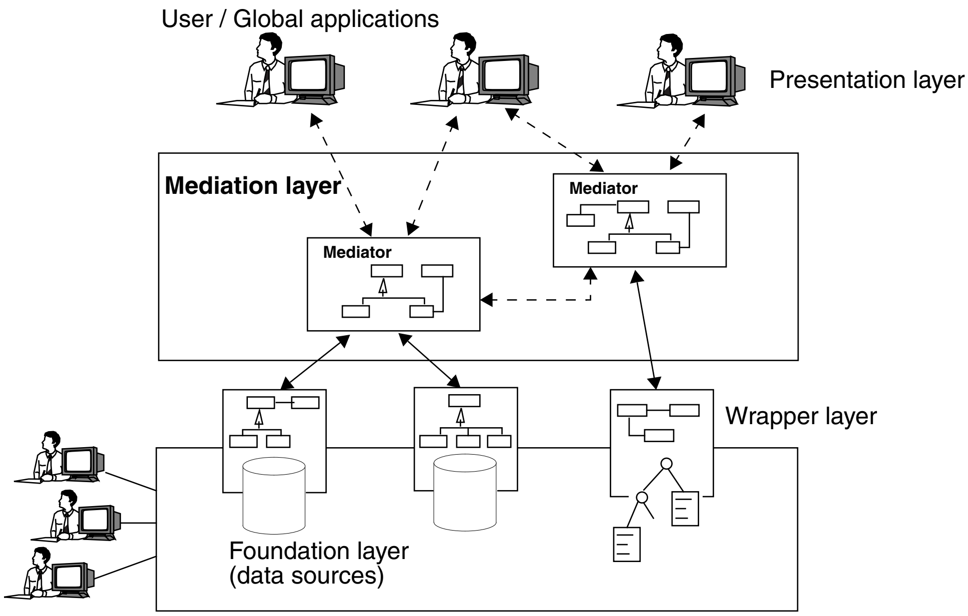

Fig. 8: Architecture of Mediator-based Information Systems (MBIS)

5 Query Mediation in Federated Information Systems

In this chapter we will take a closer look at a particular task in FIS, namely the treatment of queries in mediator-based systems with fusion integration level. We coin the term ‘query mediation’, which denotes the process of answering a query against a schema by translating it into queries against other schemas, in the presence of query restrictions and semantic, structural and schematic conflicts between the addressed schemas (see fig. 9).

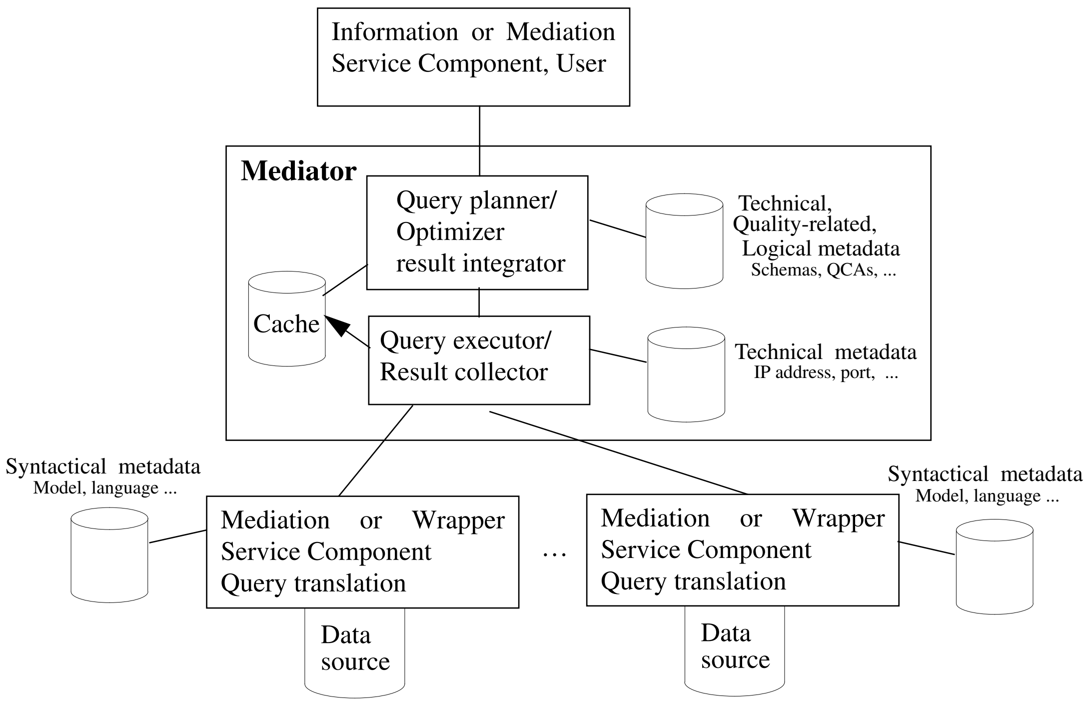

Fig. 9: Query Mediation

Note that we restrict our notion of mediation in this chapter to **query mediation**. Another topic of interest, which is not treated here, is service mediation or trading (see section 6.5).

5.1 Steps in query mediation

Only in very rare cases user queries against the global schema can not be answered directly by sending them entirely to one source. Instead, the mediator has to find combinations of queries against sources that, if combined in a meaningful way, together yield correct results to the original query (query rewriting). We call a combination of source queries that is considered as a possible solution for a user query a **plan**. We say a plan is **correct** if it obtains only semantically correct answers to the user query. Note that a single plan will in general not obtain all correct answers. We will show later with which techniques a mediator can decide whether or not a plan is correct.

On this background, we define **query mediation** as consisting of three major steps:

- **Query planning** is the process of finding a correct plan of executable source queries for a given query against the federated schema. Query planning must be based on predefined correspondences between queries or concepts in different schemas, and it must account for query restrictions in the different sources.

- **Plan execution** is the process of executing a plan. This comprises (1) optimization steps that decide which query operations are performed by which component, (2) the shipping of subqueries and the collection of the results, and (3) applying potentially necessary post-processing such as computing inter-source joins inside the mediator.
- **Result integration** finally tries to homogenize the obtained data by removing redundancy, identifying identical objects and resolving inconsistent data values (i.e. perform fusion integration, see section 3.6).

5.2 Plans and correspondences

Any query planning must be based on some description of the source content with respect to the global schema, for instance by means of views as defined in SQL. We call the language to express these correspondences *correspondence specification languages* (CSL). Hence, query planning must find correct plans by exploiting the semantic knowledge that is expressed in rules of a certain CSL (see e.g. [CL 93], [SPD 92]). Finding such correspondences is usually the task of a human operator, since they encode the semantic relationships between concepts.

There are two basic classes of CSLs. Following the **Global-as-View** (GaV) paradigm, the global schema is defined by having one or more views over the sources schemas for each class (see also section 3.2). Hence, each correspondence rule has a single global class on one side and defines its semantic equivalence to a source query on the other side (of the rule). The situation is reversed in the **Local-as-View** paradigm, where the classes of the source schemas are described by giving equivalent views on the global schema [Hul 97]. Again, each rule has a class and a query; but here, it is a local class and a global query (see fig. 10)

Apart from technical implications (see below), this mainly expressed a difference in the perception of the global schema: while GaV sees the global schema as something artificial that must be filled with life by accessing sources, the LaV rather assumes each source as a certain part of overall, global information space.

Global–as-View

Query translation in a GaV approach basically requires the expansion of the classes in a user query into the corresponding source queries [MY 95]. The expansion step "global class → source query" is hardcoded in the definition of the classes (as views). Usually, also the information fusion rules are contained in this definition. For instance, the TSIMMIS approach builds complex restructuring and priority rules into their view definition language MSL [PGW 95], [GMPQ+ 97].

Local-as-View

LaV query planning requires a more complex process, because it is a-priori unclear which parts of a given user query are defined through a view. Every single global view can potentially contribute to a plan for the query. This problem, also known as "answering queries using only views" [LMSS 95], [Qia96], [CKPS 95], is shown to be NP-complete already for conjunctive queries and conjunctive view definitions in [LMSS 95]; it can be solved by enumerating a possibly exponential number of view combinations, and testing query containment for each of these combinations [Les 98b], [Les 98c]. The problem quickly becomes undecidable, e.g. if negation is allowed [Ull 97].

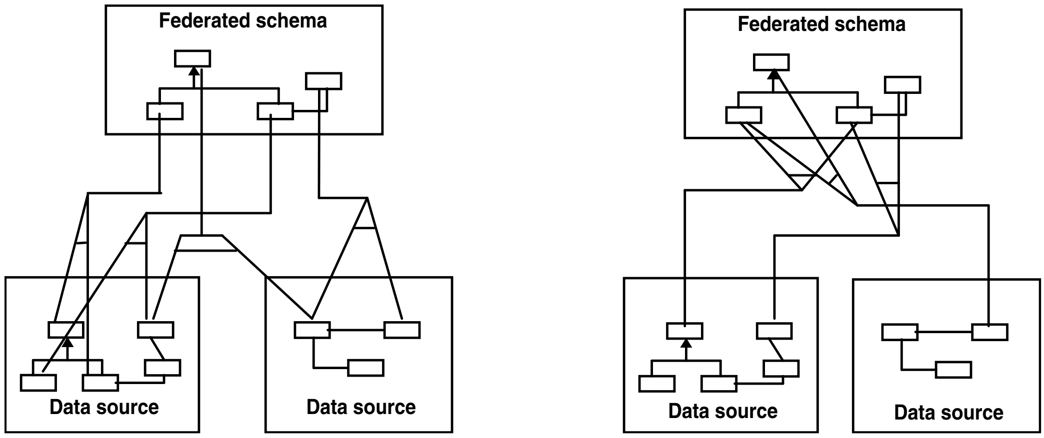

Fig. 10: GaV versus LaV approach. Angles indicate view definitions. Left-hand side is GaV, where federated classes are defined as view on wrapper schema, right-hand side is LaV, where wrapper classes are defined as view on the mediator schema.

We can see that the question of whether a plan will obtain semantically correct answers to a query is in GaV case solved trivially, while it must be explicitly tested in the LaV approach. Please note that the expressiveness regarding semantic heterogeneity in GaV and LaV are different; both can express situation which the other can not. Examples can be found in [Les 99].

5.3 Properties of query mediation

Approaches to query mediation can be further classified regarding a number of properties that the mediation algorithm has or lacks.

- **Semantically contained versus semantically equivalent plans**

  In LaV approaches, plans that are only semantically contained, but not equivalent, can be either considered or discarded. For instance, if a user query asks for the prices of cars younger than 1990, it is questionable whether a source storing data about cars younger than 1995 shall be queried or not. Certainly, the results will be correct; on the other hand, other plans will obtain "more" correct results, which is important if resources, such as time, are limited.

  Note that this problem does not occur in GaV approaches, since a situation as above simply cannot be expressed through GaV rules.

- **Structurally complete or partial plans**

  In both GaV and LaV approaches, the views will in general not obtain all attributes of global relations. This for instance occurs if a global class has some attributes that the same class in a particular source does not have. Therefore, a correct plan can potentially be structurally incomplete, i.e. not compute values for all required attributes of a user query. Mediator systems differ in whether or not such plans are considered as valid. A *cooperative mediator* might even suggest the changing of a query if for instance only one single attribute or conditions prevents the finding of a correct plan.

- **Globally complete or partial answers**

  In general, there will be many plans to answer a given query. In contrast to centralized databases, where all plans to answer a query will obtain the same tuples[15], different plans will obtain different results in the FIS setting since different plans in general use different sources. A mediators can either generate and execute all correct plans, only one or up to a certain stop criteria, for instance induced by some quality reasoning [NLF 99]:
  - In settings with highly consistent data sources (e.g. book information: author, ISBN number, list of authors and publisher of a book will probably be equal in all potential data sources) the mediator might decide to execute only one of the set of correct plans.
  - If values are expected to be less consistent, the mediator might decide to execute only a certain number of plans, which will be chosen using some selection criteria, such as source quality, cost, etc. He could e.g. use quality and completeness assessments of data sources to generate and execute plans until a certain threshold of the overall completeness is reached.
  - Finally, the mediator can decide to execute all correct plans. This ensures *globally complete* answers, with respect to the sources and their descriptions used by the mediator.

6 Related Approaches

Within the last chapters we already referenced particular literature about specific topics. Now we will relate our terminology in a broader scene. We will relate our classification criteria to existing projects in this context, and will discuss the definition of mediator-based information systems from Wiederhold and the I^3 reference architecture, respectively. Besides, the relationship to cooperative information systems and – in a more technical direction – to component integration technologies are described.

6.1 Research projects

We will characterize four typical and prominent MBIS with a short summary and comparison in tabular form. Naturally, we can not consider all relevant projects in this context.[16] In particular, we do not address mediators which use materialization. This includes metadata-based approaches with content-based metadata for integration because it is quite difficult to compare these with the others.

15. And where query optimization hence is the task of finding the optimal (quickest, cheapest etc.) out of a set of equivalent plans.
16. Other relevant projects are e.g. DISCO [TRV 96], InfoMaster [DG 97], DIOM [LP 97], COIN [GMS 94], or Araneus [AHK 96].

Summary of relevant projects

- **Garlic** ([HKWY 97], [TS 97])

  Garlic is a project of IBM Research. It addresses large-scale multimedia information systems by considering specialized component systems to store and search for particular data types like image management systems. The component export schemas are tightly integrated within an object-oriented data model. Heterogeneity in schemas is not considered, but differences in query capabilities are handled very flexible by a powerful query optimizer. Even changes of capabilities do not affect the mediator. Garlic requires quite powerful wrappers, since query execution depends on a interactive communication between mediator and wrappers about the component's capabilities. Once such wrappers are implemented, the autonomy of components is high.

- **Information Manifold** ([LSK 95], [LRO 96a], [LRO 96b])

  The Information Manifold prototype developed of AT&T in 1995/96 integrates structured data sources on the WWW. Main topics of the project were source descriptions and query processing. The heterogeneous components are hidden through a mediator schema (the "world model") which is designed according to the information needs on top of the system (top down). Each concept of a component schema is related to the mediator schema using a powerful declarative language following the local-as-view approach. Given a user query, the system uses the descriptions to identify relevant sources, executes the sub-queries and collects the results. Further computation or object fusions have to be applied by the user. Because of the independence between mediator and component schema and the explicit description of their relationship, the components keep their autonomy and the system can evolve easily.

- **SIMS** ([AHK 96], [AKS 96])

  SIMS ("Search in Multiple Sources") uses an ontology to provide a global access point to heterogeneous components. This ontology uses a description logic as data model. The high expressiveness of the description logic allows to integrate sources with quite different data models. The ontology-based approach particularly addresses semantic heterogeneity. Query processing in SIMS considers replicated data sources, so that queries can be answered even if a component is not available. As the Information Manifold, SIMS allows for the autonomy of components and for the evolvability of the system: components are integrated by relating source concepts to the mediator ontology independent of other components and the mediator schema itself (Local-as-View). Necessary modifications or extensions of the mediator ontology can be done independently.

- **TSIMMIS** ([CGH+ 94], [PAG 96], [PGU 96])

  TSIMMIS ("The Stanford-IBM Manager of Multiple Information Sources") supports the integration of heterogeneous data sources. It addresses structured as well as semi-structured components. In contrast to the projects described above, TSIMMIS does not provide a mediator schema, but propagates all schemas of the component's wrappers to the user. Data model heterogeneity is resolved using the semi-structured "Object exchange model" (OEM), a simple model with objects and object nesting (but e.g. no inheritance and no classes). To resolve semantic conflicts between components, a dictionary service was proposed, but not implemented. One component of TSIMMIS allows the specification of integrating view using the **Mediator Specification Language** (MSL). Therein particularly mechanisms for object fusions or abstractions are available. Interestingly, TSIMMIS often uses quite non-standard semantics for terms. In their language, a mediator is simple one integrating view. Furthermore, they promote "thick" wrappers, in that the shift many tasks that are usually considered to lie in the mediator's responsibility into the wrapper, such as the decomposition of queries and the compensation of missing query capabilities

Comparison

The table below summarizes the characteristics of these projects according to the dimensions we identified in this paper. Note, that TSIMMIS is a loosely coupled information system and not a MBIS in out terminology.

We omit two criteria: first, all approaches are based on queries (query paradigm, section 3.6). Second they all assume sources to be encapsulated by wrappers that compensate for critical restrictions in query methods (access method, section 3.10). For illustration, we added the principle mechanism of query processing as one criteria.

	Garlic	**Information Manifold**	**SIMS**	**TSIMMIS**
Autonomy	High (medium wrapper)	High (thin wrapper)	High (thin wrapper)	High (thick wrapper)
Heterogeneity	Interface	Structural, semantic	Structural, semantic	Only data model
Evolvability	Good for queries capabilities, bad for schemas	Very good	Very good	Very good
Kinds of components	Structured	Structured	Structured	All
Tight vs. loose	Tight	Tight	Tight	Loose
Data model of the FIS	ODMG-93	Relational with extension	Description Logic LOOM	OEM
Kinds of semantic integration	Collection	Collection	Collection, Abstraction	Collection, Fusion, Abstraction
Transparency	Language	Complete	Complete	Language; location and schema with MSL views
Bottom-up vs. Top-down	—	Top-down	Top-down	Top-down
Virtual vs. materialized	Virtual	Virtual	Virtual	Virtual
Read-only?	Yes	Yes	Yes	Yes
Query processing principle	GaV; cost-based query optimization	LaV; semantic query containment	LaV; query subsumption in description logic	GaV; MSL view expansion

Table 4: Comparison of projects

6.2 "Mediators" according to Wiederhold

Our definition of MBIS has some differences to [Wie 94]:

- “Mediation is the principle means to resolve problems of semantic interoperation. It recognizes the autonomy and diversity of data sources” is a definition that is compatible with ours, although we give a much more precise meaning to this sentence.
- “Mediation is the task of reducing data to information by applying knowledge about resources, search strategies and user requirements”. This is different from our definition, which could be rephrased as “finding only relevant data from a wide range of sources”, and hence reduces the amount of information by filtering. Since we have no clear understanding of the expression "reducing data to information", we omit it. Mediation in our definition is furthermore user independent, apart from the fat that users formulate queries against the global schema.

Wiederhold gives the following possible "added-value services" for mediators. We define which of these service are an obligatory part of our definition (and not an added service), and which we are not considering (of course, one could define them as added value for our definition as well):

[Wie 94], [WG 97]	Part of our definition?
Selection of relevant sources	Yes (implied in tight approach with location transparency)
Resolution of scope mismatches	Yes
Abstraction to same level of granularity	Yes
Integration of data	Yes (fusion-level integration)
Assessment of data quality	No
Ranking in terms of quality	No
Omission of replicated data	Yes (implied in fusion-level integration)
Seeking exceptions from trends	No
Transformation of material	Transformation to data model and schema of the mediator
Adaptation to bandwidth and media capabilities	Adaptation to query capabilities: yes Adaptation to bandwidth: no
Optimization towards small response time	Yes

Table 5: Definitions of mediators

[Wie 92] gives another definition of mediator and mediation:

> “A mediator is a software module that exploits encoded knowledge about certain sets or subsets of data to create information for a higher layer of applications”.
>
> “... This term [mediation] includes the processing needed to make the interface work, the knowledge structures that drive the transformations needed to transform data to information, and any intermediate storage that is needed”.

The definition of mediators is compatible to ours.

[Wie 92] gives also a very interesting list of desirable features of mediators. We can only underline them as desired, but not required features:

- small and simple to increase maintainability

- inspectable for users to aid their decision which mediator to use. Inspectability requires some declarative specification.
- hierarchical organization in networks
- Supporting flexible application interfaces
- Reusability and shareability
- Pruning of the search space
- Representation of vague and uncertain information

6.3 The I^3 Reference Architecture

The I^3 reference architecture is one result of the Program on Intelligent Integration of Information (I^3) of the Advanced Research Projects Agency (ARPA). The program includes several projects developing "… methods and tools to combine information from autonomous and heterogeneous information resources. […] The approach is to develop standards for constructing and integrating such information sources." ([ARPA 95]).

The information technology vision taken from the I3 community includes solutions for source integration, information sharing, and collaboration management. According to that, the reference architecture describes services organized in four service families [ARPA 95]:

- Wrapping services "are used to make information sources comply with an internal or external standard."
- Semantic integration and transformation services "support the semantic manipulations needed when integrating and transforming information to satisfy an I^3 task, as well as the capabilities needed to re-use program components."
- Coordination and Management Services "provide support, whether ad hoc or automated, for programming I^3 configurations."
- Functional Extension services "augment functionalities of other I^3 services."

These services are similar to our three-tired architecture with technical integration (wrappers), semantic integration (mediators) and information services on the presentation layer. But we concentrate on semantic integration and transformation services as mediation tasks as a basis for coordination services. Besides, we do not address configuration tasks of system components. So, the I^3 reference architecture addresses a much broader universe of discourse.

On the other side, the I^3 reference architecture only gives an overview on relevant services: "A reference architecture model describes a system in terms of the interconnection of basic functional elements and the interfaces between them. It clarifies where protocols must be defined and identifies groupings of functionality. It does not imply a physical implementation" ([ARPA 95]). As a consequence, identified I^3 services are only described informally – some of them are only named.

Finally, we notice that this reference architecture gives an overview of many relevant problems of federated information systems, which can be taken to integrate concrete projects in a broader scene. But it does not say anything about concrete solutions.

6.4 Cooperative Information Systems

In the 90s the new term "cooperative information systems" (CoopIS) was born for the vision of next generation information systems (see the conference CoopIS, e.g. [PS 98]). The manifesto [MDJ+ 98] proposes a framework for CoopIS with three interrelated facets:

- The system facet includes information, workflow, and other computer-based systems supporting some specific tasks. Within the system facet, problems of heterogeneity and interoperation of systems have to be solved.
- The group collaboration facet addresses how people work together on a common business process or other project; it is influenced from the research area of computer-supported cooperative work (CSCW).
- The organizational facet "addresses global organizational concerns, including organizational objectives and business goals, policies, regulations and resulting workflow or project plans." It does not regard by whom or with what technology these objectives are reached.

These three facets show that CoopIS consider information systems technologies as we have discussed them in this paper in a larger social and organizational context. Similar to the I^3 vision, the coordination and cooperation is more important than a simple sharing of information.

[MDJ+ 98] shows that change management and continuous evolution is the central issue related to and relating all three facets. We noted the importance of evolution which has to be considered when information systems are designed, too. Here much more work has to be done, wherein we also will concentrate on the influences on the system facet.

6.5 Related interoperation paradigms

Obviously, the intention of information federation and of mediation today is not only tackled from the classical database and informations systems viewpoint, but also under quite different paradigms of "open distributed computing". The term "mediation" can be related to "brokering" as it is used in classical middleware architectures, cf. [OMG 95a], and also to "trading", which was introduced in the conceptual context of 'Open Distributed Processing' ODP, cf. [ODP 95]. Moreover, the "agents" paradigm gives another approach to the class of requirements under discussion in our report: here, the search for information, possibly its integration and further processing, is done by a number of active subjects, acting and collaborating in a very autonomous manner.

A first differentiation between these approaches can be made by introducing the notion of service: A service provides useful functionality in a given system context. Examples of services we have discussed are the mediation of information or mediation of queries in a given information infrastructure, cf. fig. 2 in section 2. Here, a clear focus lies on the information as a rather static resource, as usually applied in the database context.

However, the combination of data and functionality, as e.g. pursued in the object oriented paradigm, immediately gives rise to different styles of integration and, as a result, interoperation in a large information infrastructure. Not only the static data is the matter of interest in the mediation, but also functions that operate on data.

We shall briefly summarize the basic aspects of these concepts as related to mediation.

Brokering & Object Interoperation

The most prominent broker architecture has been introduced by the Object Management Group, namely CORBA, the Common Object Request Broker Architecture [OMG 95a]. We consider CORBA as today being state-of-the-art in object-oriented distribution and interoperation w.r.t. its object model, the object management architecture, and the object services [OMG 95b].

Particularly, the set of services established around the persistence issue, i.e. the former Persistent Object Service (POS), the new Persistent State Service (PSS), the Portable Object Adaptor (POA) and the Object Transaction Service (OTS) provide a bunch of functionality which, related to the general service provision philosophy of CORBA, gives a perfect technological background to cope with mediation as another (more high-level) service on an architectural level.

The basic philosophy of the broker itself is to establish an easy-to-use transparent layer of object interoperation and related services – the process of mediation, its semantics and its application context explicitly is left out of the scope of brokerage. However, query mediators can be established on-top of these middleware services [DDO 98].

Component Integration Technologies

Closely related to the CORBA interoperation model, there are component models, based on OMG's CORBA itself, based on Microsoft's alternative COM/DCOM [Box 98], [Ses 97], or as introduced in the Java/Web area by Java Beans [OH 98], addressing larger granules than objects for the interoperation: software components.

Middleware and software components, in general, in first place only remove technical heterogeneity. In some cases also data model heterogeneity is addressed, e.g. if data is wrapped in OMG IDL interfaces. But logical heterogeneity is not treated per-se. Although common middleware technologies support the definition of services for further purposes (see the CORBA services mentioned above), there do not exist sufficient services for semantic data integration.

In this sense, a mediator will use a middleware technology to access data sources. If all sources were available through one component integration infrastructure, many problems of technical heterogeneity would disappear. Some would however remain, such as the question whether a connection is stateless or not. If e.g. CORBA is used and data objects are represented as CORBA interface, format heterogeneity would also not exist any more.

Trading & Service Mediation

In contrast to brokering, the concept of trading is rather that of a mediation, here understood in the meaning of mediating services between service clients and service providers.[17] Traders, cf. the ODP approach [ODP 95b], have to "know" the clients' or customers' requirements, as well as the service offers of the providers and then relate them to each other. The request therefore carries certain requirements to and descriptions of the requested service (e.g. name, quality-of-

17. A *service*, in this context, has to be understood in the meaning of some piece of functionality, offered by a service provider to some client interested in. This relates well to the notion of 'client-server-computing' and to the (object-oriented) middleware, as discussed above.

service requirements etc.). Based on the request and a repository of available service descriptions the trader will select the ‘optimal’ service, whether it might be a choice of only one or of many providers.

In an idealistic intention, there should be semantics included in the trading process, similar to our discussion of the mediators. In practice, however, trading is done on a merely syntactical level by name matching and, in positive cases, additional type information and Q-o-S descriptions on a more technical level.

Comparing traders with mediators, one can recognize the following differences:

- Traders only select services, but they do not integrate or homogenize services.
- Traders, in general, do not combine different services, but only choose the optimal service from a set of potentially applicable services when receiving a request.
- Traders do not combine services to handle specific client requests. Of course, traders may trade integration services, but they do not carry out this task themselves.
- After selecting a service, the trader is not any more participating in the connection, while, e.g. a query mediator will translate each query (= service invocation)

7 Summary

The main objective of this report is a clarification of terminology and concepts in the area of information integration. To this end, we have defined the notion of federated information system as the broadest term for such systems. We distinguish FIS from distributed databases through the high degree of autonomy that components can retain. We then gave ten different classification dimensions under which FIS approaches can be classified. Using this catalog, we tried to clearly separate the terms "federated database", "mediator-based system" and "loosely coupled system".

Another main focus of this report was a characterization of mediator-based systems. The definition we finally found is mainly based on three fundaments: the initial definition of Wiederhold [Wie 92], which we however consider as too vague for our purpose, and a thorough analysis of systems that call themselves "mediator-based" [LBW 99]. In short, our definition is:

> *A mediator-based information system is a system that offers a homogeneous, virtual and read-only access mechanism to a dynamically changing collection of heterogeneous, autonomous and distributed information sources. Access is carried out through a query language and uses a global schema. MBIS are typically lightweight systems that are developed top-down, addressing all types of heterogeneity. This results in a methodological concentration on the mechanisms for query processing, since components typically are not full-fledged database management systems and often offer only very limited query capabilities. The main software components of a MBIS are wrappers, which encapsulate sources and remove technical and data model heterogeneity, and mediators, which resolve logical heterogeneity.*

This definition makes deliberately no statement about whether or not a MBIS can use a semi-structured data model, where interface heterogeneity is resolved and what degree of semantic integration is offered. We consider these criteria as distinguishing issues in-between the class of MBIS.

MBIS can be distinguished from tightly integrated federated databases in that they do not consider write access, and can hence use much more flexible mechanisms for the correlation of heterogeneous schemas. We believe that in particular the requirement for write-access have led to the predominant concentration on schema integration methods in research on FDBS. If this requirement is not necessary, the restrictions imposed by schema integration are not worth the benefits in our experience. We can not find a distinguishing difference between MBIS and cooperative information systems. CoopIS typically concentrate more on interactive aspects between components, which is closer to the vast field of research on autonomous agents. In the last section, we also tried to shed some light on a broader perspective of mediation in software systems that is not restricted to the structural information viewpoint we took throughout the rest of this report.

References

[ACM 93] S. Abiteboul, S. Cluet,T. Milo, *Querying and Updating the File*. 19th Conference on Very Large Databases, Dublin, Ireland pp. 73-84, 1993.

[AHK 96] Y. Arens, C.-N. Hsu, C.A. Knoblock, *Query Processing in the SIMS Information Mediator*, in: A. Tate (ed.), *Advanced Planning Technology*, AAAI Press, pp. 61-69, 1996.

[AKS 96] Y. Arens, C.A. Knoblock, W.-M. Shen, Query *Reformulation for Dynamic Information Integration*, Jounal of Intelligent Information Systems, Special Issue on Intelligent Information Integration, Vol. 6, No. 2/3, pp. 99-130, 1996.

[ARPA 95] Advanced Research Projects Agency, Program on Intelligent Integration of Information, *Reference Architecture for the Intelligent Integration of Information*, Version 2.0 (Draft), Aug. 1995.

[Box 98] D. Box, *Essential COM*, Addison-Wesley, 1998.

[Bun 97] P. Buneman, *Semistructured Data*. 16th ACM Symposium on Principles of Database Systems, Tuscon, Arizona pp. 117-121, 1997.

[BDH+ 95] P. Buneman, S. Davidson, K. Hart, G.C. Overton, *A Data Transformation System for Biological Data Sources*, Proc. 21st Conference on Very Large Data Bases (VLDB'95), pp. 158-169, 1995.

[BLL+ 99] E. Barillot, U. Leser, P. Lijnzaad, C. Cussat-Blanc, K. Jungfer, F. Guyon, G. Vaysseix, C. Helgesen,P. Rodriguez-Tome , *A Proposal for a Standard CORBA Interface for Genome Maps*, Bioinformatics 15: 157-169, 1999.

[BLN 86] C. Batini, M. Lenzerini, S.B. Navathe, *A Comparative Analysis of Methodologies for Database Schema Integration*, ACM Computing Surveys, Vol. 18, No. 4, pp. 323-364, Dec. 1986.

[BSKW 91] T. Barsalou, N. Siambela, A. M. Keller,G. Wiederhold, *Updating Relational Databases through Object-Based Views*. ACM SIGMOD Int. Conference on Management of Data 1991, Denver, Colorado pp. 248-257, 1991.

[Con 97] S. Conrad, *Föderierte Datenbanksysteme: Konzepte der Datenintegration*, Springer, 1997.

[CGH+ 94] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. D. Ullman, J. Widom, *The TSIMMIS Project: Integration of Heterogeneous Information Sources*, 16th Meeting of the Information Processing Society of Japan, pp. 7-18, Tokyo, Japan, 1994.

[CKPS 95] S. Chaudhuri, R. Krishnamurthy, S. Potamianos,K. Shim, *Optimizing Queries with Materialized Views*. 11th Int. Conference on Data Engineering, Los Alamitos, CA, IEEE Computer Soc. Press pp. 190 - 200, 1995.

[CL 93] T. Catarci, M. Lenzerini, *Representing and Using Interschema Knowledge in Cooperative Information Systems*, Journal for Intelligent and Cooperative Information Systems, Vol. 2, No. 4, pp. 375-399, 1993.

[Dad 96] P. Dadam, *Verteilte Datenbanken und Client/Server-Systeme – Grundlagen, Konzepte und Realisierungsformen*, Springer, 1996.

[DC 99] *The Dublin Core home page*, last visited at 16-Feb-99, http://purl.oclc.org/dc/

[DDO 98] A. Dogac, C. Dengi, M. T. Özsu, *Distributed Object Management Platforms*, Communications of the ACM, Vol. 41, No. 9, pp. 95-103, 1998.

[DG 97] O.M. Duschka, M.R. Genesereth, *Query Planning in InfoMaster*, 12th ACM Symposium on Applied Computing, San Jose, CA, 1997.

[FGDC 94] Federal Geographic Data Committee, *Content Standards for Digital Geospatial Metadata*, Washington, Jun. 1994.

[FS 98] L. Faulstich, M. Spiliopoulou, *Building HyperNavigation Wrappers for Publisher Web-Sides*, Proc. 2nd European Conf. on Digital Libraries; LNCS 1513, pp. 115-134, 1998.

[GM 95] A. Gupta, I.S. Mumick, *Maintenance of Materialized Views: Problems, Techniques and Applications*, IEEE Quarterly Bulletin on Data Engineering, Special Issue on Materialized Views and Data Warehousing, Vol. 18, No. 2, pp. 3-18, 1995.

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

[GMS 94] C.H. Goh, M.E. Madnick, M.D. Siegel, *Context Interchange: Overcoming the Challenges of Large-scale Interoperable Database Systems in Dynamic Environments*, in: N. Adam, B. Bhargava, Y. Yesha (eds.), *3rd International Conference on Information and Knowledge Management*, ACM Press, pp. 337-346, 1994.

[GMY 99] H. Garcia-Molina, R. Yerneni, *Coping with Limited Capabilities of Sources*. 8th GI Fachtagung: Datenbanksysteme in Buero, Technik und Wissenschaft, Freiburg, Germany, Springer Verlag pp. 1-19, 1999.

[Hul 97] R. Hull, *Managing Semantic Heterogeneity in Databases: A Theoretical Perspective*, Proc. ACM Symp. on Principles of Databases Systems PODS'97, 1997.

[HKWY 97] L.M. Haas, D. Kossman, E.L. Wimmers, J. Yang. *Optimizing Queries Across Diverse Data Sources*, 23rd Conference on Very Large Database Systems, Athen, Greece, 1997.

[HL 7] *Health Level 7 Standards Page*, http://www.mcis.duke.edu/standards/HL7/hl7.htm; last visited 15 March 1999.

[HM 85] D. Heimbigner, D. McLeod, *A Federated Architecture for Information Management*, ACM Transactions on Office Information Systems, Vol. 3, No. 3, pp. 253-278, July 1985.

[HP 96] U. Hohenstein, V. Plesser, *Semantic Enrichment: A First Step to provide Database Interoperability*, Workshop Föderierte Datenbanken, Magdeburg, pp. 3-17, 1996.

[Ken 91a] W. Kent, *The Breakdown of the Information Model in Multi-Database Systems*, SIGMOD Record 20(4): 10 - 15, 1991.

[Ken 91b] W. Kent, *Solving Domain Mismatch and Schema Mismatch Problems with an Object-Oriented Database Programming Language*. 17th Conference on Very Large Databases, Barcelona, Spain, 1991.

[Kim 95] W. Kim (ed.), *Modern Database Systems*, Addison-Wesley, 1995.

[KLK 91] R. Krishnamurthy, W. Litwin,W. Kent, *Language Features for Interoperability of Databases with Schematic Discripancies*. ACM SIGMOD Int. Conference on Management of Data 1991, Denver, Colorado pp. 40 - 49, 1991.

[KS 95] V. Kashyap, A. Sheth, *Semantic and Schematic Similarities between Database Objects: A Context-based approach*, Sept. 1995; an abridged version appears in the VLDB Journal, Vol. 5, No. 4, pp. 276-304, Oct. 1996; 1995.

[Len 95] D.B. Lenat, *Cyc: A large-scale investment in knowledge infrastructure*, Comm. ACM, Vol. 38, No. 11, pp. 33-38, Nov. 1995.

[Les 98a] U. Leser, *Maintenance and Mediation in Federated Databases*. 8th Workshop on Information Technology and Systems, Helsinki, Finland, TR-19, University of Jyvaeskylae pp. 187-196, 1998.

[Les 98b] U. Leser, *Combining Heterogeneous Data Sources through Query Correspondence Assertions*. 1st Workshop on Web Information and Data Management, in conjunction with CIKM'98, Washington, D.C. pp. 29-32, 1998.

[Les 98c] U. Leser, *Query Mediation for Heterogeneous Data Sources*. 3rd Workshop 'FÖderierte Datenbanken', Magdeburg, Germany, Shaker Verlag, Aachen pp. 33-44, 1998.

[Les 99] U. Leser, *Query Mediation in Federated Information Systems*. Technische Universitaet Berlin, Technical Report, in prepreration, 1999.

[LBW 99] U. Leser, S. Busse, H. Weber (eds.), *Mediator-basierte, heterogene verteilte Informationssysteme*. Abschlußbericht zum gleichnamigen Seminar, TU Berlin, WS 1998/99. Unpublished. Available at (in german): http://cis.cs.tu-berlin.de/Lehre/WS-9899/Sonstiges/i3-pages/endbericht.ps.gz, 1999.

[LMR 90] W. Litwin, L. Mark, N. Roussoupoulos, *Interoperability of Multiple Autonomous Databases*, ACM Computing Surveys, Vol. 22, No. 3, pp. 267-293, Sep. 1990.

[LMSS 95] A.Y. Levy, A. O. Mendelzon, Y. Sagiv, D. Srivastava, *Answering Queries using Views*. 14th ACM Symposium on Principles of Database Systems, San Jose, CA pp. 95-104, 1995.

[LP 97] L. Liu, C. Pu, *Dynamic Query Processing in DIOM*, IEEE Quaterly Bulletin on Data Engineering, Special Issue on Improving Query Responsiveness, Vol. 20, No. 3, 1997.

[LRO 96a] A.Y. Levy, A. Rajaraman, J.J. Ordille, *Querying Heterogeneous Information Sources Using Source Descriptions*, Proc. 22nd Conf. on Very Large Databases, VLDB, Mumbai (Bombay), India, pp. 251-262, 1996.

[LRO 96b] A.Y. Levy, A. Rajaraman, J.J. Ordille, *Query Answering Algorithms for Information Agents*, 13th AAAI National Conf. on Artificial Intelligence, Portland, Oregon, Aug. 1996.

[LSK 95] A.Y. Levy, D. Srivastava, T. Kirk, *Data Model and Query Evaluation in Global Information Systems*, Journal of Intelligent Information Systems, Special Issue on Networked Information Discovery and Retrieval, Vol. 5, No. 2, pp. 121-143, 1995.

[LSRDTF 98a] Life Science Research, Domain Task Force of the OMG, *Genomic Maps RFP*. Object Management Group , Request for Proposals , OMG Document lifesci/98-11-07, 1998.

[LSS 93] L.V.S. Lakshmanan, F. Sadri,I. N. Subramanian, *On the logical foundation of schema integration and evolution in heterogeneous database systems*. 2nd Int. Conf. on Deductive and Object-Oriented Databases, Phoenix, Arizona, Springer Heidelberg, Berlin, New York pp. 81-100, 1993.

[LSS 96] Lakshmanan, L. V. S., F. Sadri,I. N. Subramanian, *SchemaSQL: A Language for Interoperability in relational Multidatabase Systems*. 22nd Conference on Very Large Databases, Bombay, India, Morgan Kaufman Publishers, pp. 239-250, 1996.

[LTB 98] Leser, U., S. Tai, S. Busse, *Design Issues of Database Access in a CORBA Environment*, Workshop on Integration of Heterogeneous Software Systems, Magdeburg, Germany pp. 74-87, 1998.

[Mil 98] R.J. Miller, *Using Schematically Heterogenous Structures*, in: L.M. Haas, A. Tiwari, *ACM SIGMOD Int. Conference on Management of Data 1998*, Seattle, Washington, pp. 189-200, 1998.

[Mot 98] R. Motz, *Propagation of Structural Modifications to an Integrated Schema*. 2nd East European Symposium on Advances in Databases and Information Systems; LNCS 1475, Poznan, Poland pp. 163-174, 1998.

[MDJ+ 98] G. De Michelis, E. Dubois, M. Jarke, F. Matthes, J. Mylopoulos, M.P. Papazoglou, K. Pohl, J. Schmidt, C. Woo, E. Yu, *Cooperative Information Systems: a Manifesto*, in: [PS 98], pp. 315-363, 1998.

[MIR 93] R.J. Miller, Y. Ioannidis, R. Ramakrishnan, *The Use of Information Capacity in Schema Integration and Translation*, 19th Conf. on Very Large Databases, Dublin, Ireland, pp. 122-133, 1993.

[MKTZ 99] N.M. Mattos, J. Kleewein, M. Tork Roth,K. Zeidenstein, *From Object-Relational to Federated Databases*. 8th GI Fachtagung: Datenbanksysteme in Buero, Technik und Wissenschaft, Freiburg, Germany, Springer Verlag pp. 185-209, 1999.

[MW 98] H. Müller, H. Weber (eds.), *Continuous Engineering of Industrial-Scale Software Systems*, Seminar Report #98092, March 2-6, 1998, IBFI, Schloß Dagstuhl, 1998.

[MY 95] W. Meng, C. Yu, *Query Processing in Multidatabase Systems*, in: W. Kim, *Modern Database Systems*, Addison-Wesley, pp. 551-572, 1995.

[NLF 99] F. Naumann, U. Leser, J. C. Freytag, *Quality-driven Integration of Heterogeneous Information Systems*. Humboldt University, Technical Report, Informatik Bericht 117, 1999.

[ODP 95] ISO/IEC International Standard 10746-1...4, *Reference Model of Open Distributed Processing – Parts 1 (Overview), 2 (Foundations), 3 (Architecture), 4 (Architectural Semantics Amendment)*, 1995.

[ODP 95b] ITU-T Recommendation X.9tr, ISO/IEC JTC1/SC21 DIS 13235, ODP Trading Function, draft, 1995.

[OH 98] R. Orfali, D. Harkey, *Client/Server Programming with Java and CORBA,* 2nd ed., Wiley & Sons, 1998.

[OMG 95a] Object Management Group, *The Common Object Request Broker: Architecture and Specification*, Revision 2.0, OMG Inc., Framingham, Massachusetts, July 1995.

[OMG 95b] Object Management Group, *CORBA services: Common Object Services Specification*, OMG Document 95-3-31, Revised ed. 2.0, Framingham, Massachusetts, March 1995.

[OMG 97] Object Management Group, *Meta Object Facility (MOF) Specification*, Joint Revised Submission, OMG Document ad/97-08-14, 1997.

[ÖV 99] M.T. Özsu, P. Valduriez, *Principles of distributed database systems*, 2nd edition, Prentice Hall, 1999.

[PAG 96] Y. Papakonstantinou, S. Abiteboul, and H. Garcia-Molina, *Object Fusion in Mediator Systems*, in: T.M. Vijayaraman, A.P. Buchmann, C. Mohan, N.L. Sarda (eds.), *22nd Conf. on Very Large Databases*, VLDB, Mumbai (Bombay), India, 1996.

[PBE 95] E. Pitoura, O. Bukhres, A.K. Elmagarmid, *Object Orientation in Multidatabase Systems*, ACM Computing Surveys, Vol. 27, No. 2, pp. 141-195, Jun. 1995.

[PGU 96] Y. Papakonstantinou, H. Garcia-Molina, J. Ullman, *Medmaker: A Mediation System Based on Declarative Specifications*, International Conference on Data Engineering, New Orleans, February, pp. 132-141, 1996.

[PGW 95] Y. Papakonstantinou, H. Garcia-Molina, J. Widom, *Object Exchange across Heterogeneous Information Sources*. 11th Conference on Data Engineering, Taipei, Taiwan, IEEE Computer Society pp. 251-260, 1995.

[PS 98] M.P. Papazoglou, G. Schlageter (eds.), *Cooperative Information Systems – Trends and Directions*, Academic Press, 1998.

[Qia96] X. Qian, *Query Folding*. 12th Int. Conference on Data Engineering, New Orleans, Louisiana pp. 48-55, 1996.

[Rah 94] E. Rahm, *Mehrrechner-Datenbanksysteme: Grundlagen der verteilten und parallelen Datenverarbeitung*, Addison-Wesley, 1994.

[RSU 95] A. Rajaraman, Y. Sagiv, J.D. Ullman, *Answering Queries using templates with binding patterns*. 14th ACM Symposium on Principles of Database Systems, San Jose, CA, ACM Press, pp. 105-112, 1995.

[Sau 98] G. Sauter, *Interoperabilitaet von Datenbanksystemen bei struktureller Heterogenitaet*. Sankt Augustin, Infix, 1998.

[Sch 98] I. Schmitt, *Schemaintegration für den Entwurf föderierter Datenbanken*, Infix Verlag, Sankt Augustin, 1998.

[Ses 97] R. Sessions, *COM and DCOM: Microsoft's Vision for Distributed Objects*, John Wiley & Sons, 1997.

[SCG 91] F. Saltor, M. Castellanos, M. García-Solaco, *Suitability of data models as canonical models for federated databases*, SIGMOD Record, Vol. 20, No. 4, pp. 44-48, ACM, Dec. 1991.

[SL 90] A.P. Sheth, J.A. Larson, *Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases*, ACM Computing Surveys, Vol. 22, No. 3, pp. 183-236, Sep. 1990.

[SM 98] J. Sellentin, B. Mitschang, *Data-Intensive Intra- & Internet Applications - Experiences Using JAVA and CORBA in the World Wide Web.* 14th Int. Conference on Data Engineering, Orlando, Florida pp. , 1998.

[SP 91] S. Spacciapietra, C. Parent, *Conflicts and Correspondence Assertions in Interoperable Databases*, SIGMOD Record, Semantic Issues in Multidatabase Systems, Vol. 20, No. 4, pp. 49-54, 1991.

[SPD 92] S. Spaccapietra, C. Parent, Y. Dupont, *Model Independent Assertions for Integration of Heterogeneous Schemas*, The VLDB Jounal, Vol. 1, No. 1, pp. 81-126, Jul. 1992.

[TRV 96] A. Tomasic, L. Raschid, P. Valduriez, *Scaling Heterogeneous Databases and the Design of DISCO*, 16th Int. Conf. on Distributed Computing Systems, Hong Kong, IEEE Computer Society, pp. 449-457, 1996.

[TS 97] M. Tork Roth, P.M. Schwarz, *Don't Scrap It, Wrap It! – A Wrapper Architecture for Legacy Data Sources*, 23rd Conference on Very Large Database Systems, Athen, Greece, 1997.

[Ull 97] J.D. Ullman, *Information Integration using Logical Views*, 6th Int. Conf. on Database Theory, Delphi, Greece, 1997.

[VJB+ 97] P.R.S. Visser, D.M. Jones, T.J.M. Bench-Capon, M.J.R. Shave, *An Analysis of Ontology Mismatches; Heterogeneity versus Interoperability*, AAAI 1997 Spring Symposium on Ontological Engineering, Stanford University, USA; also appeared as AAAI Technical Report SS-97-06, 1997.

[Wid 95] J. Widom, *Research Problems in Data Warehousing*, Proc. 4th Int. Conf. on Information and Knowledge Management (CIKM), Nov. 1995.

[Wie 92] G. Wiederhold, *Mediators in the Architecture of Future Information Systems*, IEEE Computers, Vol. 25, No. 3, pp. 38-49, Mar. 1992.

[Wie 93] G. Wiederhold, *Intelligent Integration of Information*, SIGMOD Record, Vol. 22, No. 2, pp. 434-437, ACM, Jun. 1993.

[Wie 94] G. Wiederhold, *Interoperation, Mediation, and Ontologies*, Proc. Int. Symposium on 5th Generation Computer Systems (FGCS'94), Workshop on Heterogeneous Cooperative Knowledge-Bases, Vol. W3, pp. 33-48, Tokyo, Japan, 1994.

[WG 97] G. Wiederhold, M.R. Genesereth. *The Conceptual Basis for Mediation Services*, IEEE Expert, Vol. 12, No. 5, 1997.

[ZHK+ 95] G. Zhou, R. Hull, R. King, J.-C. Franchitti, *Data Integration and Warehousing using H2O*, IEEE Quarterly Bulletin on Data Engineering, Special Issue on Materialized Views and Data Warehousing, Vol. 18, No. 2, pp. 29-40, 1995.