Information integration among Heterogeneous and Autonomous Applications

Benabdelkader, A.

Citation for published version (APA):
Chapter 2

Information Integration Approaches, Mechanisms, and Tools

2.1 Introduction

A wide variety of information management and data integration approaches, mechanisms, and tools are introduced and being used for diverse applications in the domains of life sciences, engineering, education, health care, business, tourism, and art. These approaches are mostly designed and developed to cope with the specific requirements of each application. Among the major requirements for today's and forthcoming applications, which are described in Table 1.1 of Chapter 1, we enumerate: site behavior, user facilities, security for access and visibility rights, collaboration and interoperation, use of standards and middleware solutions, system efficiency and minimization, and other advanced features. The diversities of the proposed approaches for information management and data integration are due to different aspects considered for each application, and which relate to the DBMS architecture, data storage approaches, and system interoperation mechanisms.

This chapter presents a survey on several research and development approaches that have directly or indirectly contributed to the issue of information integration and interoperation mechanisms among autonomous, distributed, heterogeneous systems. Considering the main emphasis of the thesis that addresses the integration of heterogeneous data sources and interoperation among autonomous sites, special emphasis is given to the data distribution, information exchange, and interoperability issues. Thus, the three concepts described below are used as the base for the classification of existing approaches and mechanisms for information integration:

- **DBMS architecture**: For most of the emerging applications, and considering the enormous improvement in networking and communication protocols, it is clear that the client/server architecture has become the key issue for application development. However, depending on the type of application, decision has to be taken regarding whether to use a centralized, distributed [SBD+83, TBD+87, PH 98, NDL+00], or even a federated approach [HM 85, SL 90, TA 93, HTH+99, GAH 01] for proper sup-
port of the application requirements in terms of information management and systems interoperability.

- **Data storage/access:** The manner in which data is being stored and accessed plays an important role in defining the proper integration mechanism, to be adopted for a given application. For instance, applications in the area of data mining and Online Analysis Processing (OLAP) [Sho 97, VS 99, PP 00, WB 97, FS 96, CD 97, and Kar 98] are based on the access to catalogs and repositories of data, which are reformatted and prepared for certain analysis tasks. While, in most applications from scientific and system engineering domains [WMP 98, PWD+99], the data processing mechanisms require that information has to be fetched on-line, processed on-line, and the proper decision has to be made on-line when necessary [BAK+00, ABK+00, AKB+01]. Similarly to the access mechanism, the data storage processing requirements may differ from one application to another. Furthermore, certain applications keep results locally and private, others publish the results immediately after generation, while a third type of application may require an evaluation time in order to validate the results before they get published and made available to the outside users.

- **System interoperation:** The interoperation mechanism for exchanging information and services among a set of users/applications within a collaborative community also defines the coupling mechanisms between these systems [LA 86, LMR 90, ZK 96, BA 98a, THB+98]. When data is integrated from external sources, it is very important for each system (e.g., data source) to clearly define the interaction mechanisms with other systems. Some criteria for this interaction involves handling the inter-linking among distributed data over different locations, and the format in which data is going to be exchanged.

It is clear that the complexity of the proposed approach for every advanced application depends very much on the specific characteristics of the application and its required level of integration. Therefore, **the more complex and higher is the requested level of integration in the application, the more complex and difficult becomes the development process.**

In this chapter, we study the main approaches introduced so far for information exchange and interoperation for different application domains. Furthermore, these approaches are discussed, evaluated, and those that can support some requirements for different applications and better fit the general integration purposes are validated for the general use cases. However, this is not a simple task since the requirements in terms of information handling are quite different in centralized applications from those that are distributed or federated applications. Some of the applications rely on the use of centralized archived data, others require homogeneous information replicated at different sites, a third type of applications may require run-time generation of information for decision support, and so on. Thus, **it becomes unrealistic to validate and conclude that one specific and concrete solution can support all types of today's and forthcoming applications.**

### 2.2 A Taxonomy for Information Integration

There are many classifications and initiatives aiming at the provision of a taxonomy for the information integration approaches. Some classifications look to the problem of integration/interoperation from the data access dimension, some others from the distribution/centralization dimension, and still some others from their heterogeneity and autonomy dimension [SL 90, RK 97, DD 99, ACM 00].
In order to describe other approaches suggested by research related to the subject of this thesis, and specially considering their wide diversity and distinct focus, we have resorted to the definition of certain classifications. In the proposed classification approach, since the main subject of the thesis consists of integrating heterogeneous data from distributed sources, we address the categorization architecture from the distribution/Integration point of view. The main purpose of this classification (and the names that are suggested and associated with each category) is to be able to discuss the common features among a group of systems that follow one general approach. As such, these “category names” might be defined differently in some other publications, since in general there is no common consensus among database researchers on the exact definition of these category names and the database systems they represent. Also this classification of information integration approaches is specifically focused on the main characteristics emphasized in the thesis, namely the design of a flexible environment to support many heterogeneous application domains, as it is the case for instance in scientific and engineering domains, like Virtual Laboratory environment.

Later in Chapter 6 of this dissertation, we will use the results of our investigation represented in this chapter and our classification of approaches, for identification and validation of a set of generic methodologies that can better support the general collaboration processes among heterogeneous nodes in a network of databases and applications. Provision of a set of generic tools and methodologies for information management facilitates the collaboration process among distributed and heterogeneous nodes. In such methodology the use of standard tools and middleware solution will play an important role in solving many issues related to the use of multi-platform systems, different architectures, and various information modeling methodologies.

Figure 2.1 illustrates a taxonomy diagram for information integration approaches. As depicted in this figure, at the first level of the proposed hierarchy, the management of heterogeneous data sources (information) is classified into two main categories of Distributed Systems and Integrated Systems. Distributed systems typically support applications that share common database software at both DDBMS servers and their applications. Integrated systems however, support database applications that address similar tasks in different manners or using different representations and data modeling systems. Within each of these two categories several approaches are identified, studied, and evaluated based on the applications’ requirements.

At the second level of this taxonomy, different and various approaches for information integration are derived from the two categories illustrated at level 1 of Figure 2.1. On one hand, distributed database systems can follow horizontal fragmentation, vertical fragmentation, or hybrid fragmentation [EN 00, OV 99]. On the other hand, when the application becomes more complex and requires additional functionalities, most research on related approaches, focusing on the needs for data heterogeneity resolution, result in a variety of integrated systems. Although, a number of researchers in this area still consider all these approaches as heterogeneous distributed systems.

Within the integrated approach, Domenig and Dittrich [DD 99] present a taxonomy of systems for querying heterogeneous data in which, they distinguish between materialized and virtual approaches. Similarly, Florescu et al. [FLM 98] make an important distinction, in building data integration systems on whether to take a warehousing or a virtual approach. In our classification, we name the two approaches respectively Physical Integration and Virtual Integration. In a Physical Integration the data originating from local and remote sources are integrated into one single database on which all queries can operate. In Virtual Integration (also referred to as multidatabase systems in some literatures), data remains on
the local/remote sources, queries operate directly on them and information integration has to take place on the fly during the query processing.

At the third level of the taxonomy, the variety of the proposed approaches becomes more and more specified and complex, this is due to the advanced functionalities and complex features required by advanced applications emerging in the domains of system engineering, medicine, biology, etc. Nowadays, those applications are of wide use, significant importance, and real necessity.

- The physical integration expands into *centralized databases* and *data warehouses*. In a centralized database, information is migrated from various sources into a universal DBMS, while in data warehousing information may be imported in different form and volume than it exists in its originating sources.

- The virtual integration derives into *federated* and *non-federated* systems. Each of these systems can be either *loosely coupled* or *tightly coupled*.

At the fourth level of the taxonomy, we categorize the *intermediate data interchange format* and the *pair-wise translations* as examples of non-federated systems and *global schema representation* and *Node-to-Node federation* as examples of federated multidatabase systems.

The next sections present and discuss in more details these different approaches illustrated on Figure 2.1; more focus will be put on Virtual Integrated systems, due to their relation and direct impact on the subject of the thesis.

### 2.2.1 Distributed Systems

In distributed systems, the applications considered by the network have similar functionalities and share similar type of information. Furthermore, most of the involved sites use the
same software for data modeling, information management, and accomplishment of their
global functionalities. In such architecture, distributed database systems (DDBMS) are de-
defined as a collection of multiple logically interrelated databases distributed over a computer
network. A distributed database management system (DDBMS) is a software system that
manages the distributed database, while making the distribution transparent to the user.

Figure 2.2 illustrates a distributed database architecture, in which information is stored
within four distributed databases located at different remote sites and inter-linked through
the communication network. In such a system, distribution is transparent to the user in the
sense of hiding the details of where the data is physically stored within the system. Thus,
from the point of view of the operational details of the network, the user has the freedom
concerning the data location transparency and objects naming transparency [EN 00].

Figure 2.2: Distributed Database Architecture

Figure 2.3 presents an example of a simple database model. Its structure consists of two
classes: Product and Customer each characterized by a set of attributes defining the two
classes and a relationship among them. The example will be used within this section to
illustrate some of the concepts related to distributed database systems.

<table>
<thead>
<tr>
<th>Product</th>
<th>Customer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Name: String</td>
<td>Customer Name: String</td>
</tr>
<tr>
<td>Product Description: String</td>
<td>Customer Address: String</td>
</tr>
<tr>
<td>Product Category: Enum</td>
<td>Customer City: String</td>
</tr>
<tr>
<td>Product Price: Float</td>
<td>Customer Phone: String</td>
</tr>
</tbody>
</table>

Figure 2.3: Example of a Simple Database Model

2.2.1.1 Data Fragmentation and Allocation

Figure 2.4 illustrates an example of the three types of fragmentation, which are possible in
distributed systems:

1. **Horizontal fragmentation** distributes a relation into sets of tuples (rows). The
database model conserves the same structure, restriction however is applied to the set
of records (instances) based on some conditions.

2. **Vertical fragmentation** distributes a relation into sub-relations where each sub-
relation is defined by a subset of the columns of the original relation. All objects are
partially present in the fragmented class, restriction is made for some attributes of the
class (table).
Hybrid fragmentation partitions a relation by applying the horizontal and vertical fragmentation strategy one after the other. Restriction is made on both attributes and instances of the fragmented class.

Figure 2.4: Fragmentation in Distributed Database Systems

Figure 2.4 illustrates the three types of data fragmentation:

- Horizontal fragmentation: the class Product is restricted at the instances level to only preserve products of ‘Price < 25’.
- Vertical fragmentation: the class Product is restricted at the attribute level to Name and Price.
- Hybrid fragmentation: the class Product is restricted at the attribute level to Name and Price, and at the instances level to only hold products of category ‘Home Care’.

In distributed systems failure at a single site does not make the entire system unavailable to all users: when the data and software fail on one site the other sites continue to operate, which improves both system reliability and availability\(^1\) in comparison to centralized approach. Furthermore, safeness can also be achieved and improved by replicating data and software at multiple sites.

Figure 2.5 gives a global overview and an example on how fragmentation and replication can be established within a distributed system. Here, the data fragmentation and distribution is illustrated using the simple data model presented in Figure 2.3. In addition to the three types of fragmentation, a mirror site is defined as a part of the distributed system. The mirroring site assures in fact the availability and safeness of information by holding a replicate of all the data, which is distributed among the other sites.

The distributed database architecture is still valid and used, a representative example and a good candidate for distributed systems is the banking application environment where all the distributed sites share the same database schema and participate in achieving a global and unique task. Another example with a small difference in the distributed database model can be the case of administrating a main hospital with many branches and care centers performing similar activities.

However, distributed systems are not fully appropriate for instance in emerging scientific applications, since it is very hard to guarantee the usage of the same data model and software tools for all cooperating/integrating sites.

\(^{1}\)Reliability is defined as the probability that a system is running at a certain time point, whereas availability is the probability that a system is continuously available during a time interval [EN 00]
2.2.2 Integrated Systems

In the main area of integrating heterogeneous and distributed information sources, the information integration generally implies uniform and transparent access to data managed by multiple databases. The task of an integrated database system is to answer queries that may require extracting and combining data from multiple local/remote data sources. An important distinction in classifying the different strategies for the manipulation of the data is whether to take a physical or a virtual integration approach [FLM 98].

1. In the physical integration, data from multiple sources is loaded into one single comprehensive new database on which all queries are applied. This requires that the new database needs to be updated when data changes, but the advantage is that adequate performance can be guaranteed at query time.

2. In the virtual integration, the data remains at the local/remote sources where it belongs. Queries to the integrated system are decomposed at run time into sub-queries to be applied to the local/remote sources, and then the integration of the sub-queries results must take place on the fly during the query processing. In this approach data is not replicated and is guaranteed to be up-to-date at query time.

The following sections will describe in more details both the physical and virtual integrations and address different categories derived from these two approaches. The variations in manipulating and handling the exchange of information reside in the manner the data is gathered and accessed.

2.2.2.1 Physical Integration – warehouses, malls, or marts

The physical integration (also known as materialized approach in some literatures) involves extracting data from a variety of heterogeneous distributed systems and applications, standardizing it to fit a global format, transporting it from the place where it belongs, and loading it in the local database in desired formats. There are essentially two variants of materialized systems [DD 99]:

a. Data from the remote systems is extracted, integrated, and stored in a centralized database, therefore, the remote systems are not used after data is extracted. In such a system, in case of updates to both data and meta-data of every remote site, the centralized database must be adjusted to reflect eventual changes in the remote systems. The main drawback of this approach is that existing applications have to be rewritten to fulfill the new database model.
b. Data from the remote sources are imported into one DBMS, the data warehouse. The difference with the previous case is that the underlying data sources are still operational and the warehoused data is typically not imported directly in the same format and volume as it exists in local systems. It is mostly transformed, cleaned, and prepared for certain analysis tasks, like data mining and OLAP (Online Analysis Processing).

The physical integration model is considerably adopted and used within data warehousing environments [FS 96, CD 97, WB 97, Kar 98] and OLAP databases [Sho 97, VS 99, PP 00]. Recently, several commercial DBMS products provide certain solutions for data warehousing and OLAP. To name a few of them, Oracle Migration Workbench\(^2\) [Daly 01], DB2 Warehouse Manager [ED 01], Sybase Warehouse Studio [Syb 99], SAP/R3 Hummingbird [Hum 00], Hyperion [Hyp 01], Cognos [Cog 00], and Comshare [Com 00]. They provide capabilities that specifically address the specifications and requirements of data warehousing. Furthermore, these software houses provide complete open warehouse design and meta-data management environments that simplify the process of building and managing warehoused data while delivering impressive flexibility.

Within the Virtual Laboratory\(^3\) project, the physical integration approach can be validated for some applications to support data archiving, cataloguing, and publishing, where organizations with similar working environments are able to gather and collect their finalized data in a global repository to be used for other activities. Archives in scientific applications provide storage of information off-line, and make it available when necessary to be used as input for experiments during a mining session, or an analysis process. Published data in the area of scientific experimentation refers to the final successful experiment results that an organization/system wishes to make available to outside users. Both archived and published data represent a huge volume of data that is not susceptible to be changed frequently.

However, the physical integration is not fully suitable for large scale interoperable applications, in which the design and development of a real federated system proves to be more appropriate.

### 2.2.2.2 Virtual Integration - Multi-Database Systems

The main aim in the virtual integration is to give the user the impression of working with a single DBMS, while, in fact, the data is managed by several individual DBMS. This approach is more appropriate for building systems where the number of databases is large, the data is changing frequently, and there is little global control over the participating local/remote data sources.

In the area of implementing integrated information systems, in which different systems need to collaborate, the development of software layers and tools turn out a necessity. These tools facilitate the exchange and the sharing of data among collaborative systems by translating data between different applications and making the information distribution transparent to the users. From the implementation point of view of multidatabase systems, many different approaches, dealing with this issue, are designed and developed.

---

\(^2\)http://otn.oracle.com/tech/migration/workbench/content.html

\(^3\)The Virtual Laboratory project (1999-2003) supported by the Dutch ICES/KIS foundation aims at designing and developing hardware and software reference architecture and a digital library framework to enable scientists and engineers to work on their problems via experimentation in the field of technical and scientific applications, making optimum use of by modern Information Technology.
Within the next sections we will present and discuss different approaches that have been investigated in the area of integrating heterogeneous and distributed data sources, namely, the pair-wise translation, the intermediate data format, global schema definition, and the node-to-node federation.

**Pair-wise Translation**

Using the pair-wise translation, we keep and preserve the various representations of different applications then, we provide translation tools from one application to another. As depicted in Figure 2.6, within this approach we have to develop a two-side dependent translator tool for every application within the system. Any change in any of the applications, requires rewriting some parts of code, the integration of a new application requires the development of $2*N$ new translators in which $N$ represent the number of communicating applications in the network.

![Figure 2.6: Two Side Dependent Translation](image)

The two-side dependent translation approach provides the following advantages [AE 95]:

- Every application will be designed in a manner that optimizes the considered tasks.
- Every module may be extended without any necessary updates in the other parts of the system.
- The specific translation to each application reduces the volume of information that needs to be exchanged, since every module can execute locally some tasks and send the results in a condensed format.
- A more flexible exchange for data and services can be obtained using database standards for data modeling and information access/exchange (e.g. ODL, OQL, and XML). These standards facilitate the logical links and interfaces between different pieces of information within those interconnected applications.

The pair-wise translation approach better suits in small environments where the number of communicating applications is very limited.
Intermediate Data Interchange Format - Common Data Definition Model

The integration approach through intermediate data format develop a general representation for all the anticipated data types, this representation must include some dependencies among different types. The intermediate data model also makes assumption for its entities both for their representation and the methods of handling them. An entity, e.g. a Triangle for instance, can be represented by its three points in application_1 and by its three segments in application_2. Thus, the handling methods for this entity change from one representation to another.

Figure 2.7 illustrates the architecture used by the intermediate data format, in which every application participating in the system must have at its disposal two translators:

a- The pre-processor from the application to the intermediate format, which translate data from the format used by the application and make it available according to the intermediate common format,

b- The post-processor from the intermediate format to the application, which takes information available in the intermediate format and translates it to fit the local format adopted by the application.

Examples of standards that deploy the intermediate data format and consider it for modeling object structures and representations include STEP, IGES, DXF, SET, VDA, etc. [Fow 95, TM 96]. These standards (also referred to as neutral formats) are designed to fulfill a number of high level industry requirements, and are based on a number of fundamental principles for shared product databases.

The advantage of using the common intermediate definition over the two-side dependent translation is that each application has to communicate only with the intermediate format instead of communicating with all other applications, thus reducing the number of necessary developed translators, especially when the number of applications grows.

Nowadays, the deployment of middleware solutions and standards can extend this approach, so the inter-modules communication becomes transparent. The node-to-node integration approach, presented in Chapter 6, will benefit from a part of this architecture, in which, database standards will play a role similar to the intermediate data model, in order to unify the schema definition, the query language, and the data exchange processes within the integrated applications.
The Global Schema Representation – Schema and Data Integration

The global schema representation (also known as universal representation in some literatures) is based on the development of one single representation that manages all types of data required by the networked applications and completely integrates multiple data sources into one global database in order to provide a single and coherent view on the data [SP 94]. As depicted in Figure 2.8, the global schema representation requires that all system components, accessing the single database, shall conform to the shared format and create their specific views based on it.

![Figure 2.8: Access to the Global Shared Database](image)

Modular and extensible systems make this approach not attractive for the following reasons [AE 95, EK 91]:

- There is no guarantee that new applications do not need different formats for their internal operations,
- In today’s applications, there is always the introduction of new methods that require particular information structure to be reflected in the underlying database model of the applications,
- The use of such a representation, which supports an exhaustive structure, represents a supplementary load for systems that manage simple applications.

In addition, the global integrated schema is hard to maintain and to automate. Furthermore, systems autonomy is often sacrificed in order to solve semantic problems.

Node-to-Node Federation

The information dealt with in federated systems consists of different pieces of information gathered from disparate local and remote sources and integrated into one coherent view, known as integrated schema. Therefore, the integrated schema is constructed by merging the structure of the local schema with the various imported schemas. The imported schemas represent the part of information that other systems wish to share with the outside world.
There are two main features distinguishing a federated system from a traditional database system [FLM 98]:

1. A system within the federation does not communicate directly with a local storage manager. Instead, the query execution engine communicates with the corresponding wrappers on top of it. Those wrappers are defined within the federated layer developed for each system.

2. The user does not pose queries directly on the schema in which the data is stored. Instead, the user poses queries on an integrated (or mediated) schema defined at the federated layer of his system.

In the federation approach, every site/user is responsible for integrating the schemas they need for their applications. Support is therefore provided by a federated or multidatabase language that contains the syntactic constructs needed for accessing and manipulating disparate and autonomous databases [BBE 99]. In addition to these constructs, a federated system requires a set of sources defining the mapping rules, the semantics description, and eventually the data exchange format. These sources also define the manner in which information will be imported, integrated, and accessed.

The way in which information is imported, integrated, and accessed in multidatabase systems differs from one approach to another. They all consider the data distribution, the heterogeneity of information, and the autonomy of the systems participating in the collaboration. However, visibility level for the data and the heterogeneity at the DBMS level are not fully considered within most of those systems. In addition, extensibility for new DBMSs to be considered within the federation is not supported.

Some of the commercial products such as Sybase, Oracle, Ingres/Star, and UniSQL/M have extended the functionalities of their systems to support some of the general requirements of the multi-database systems [HBP 94] [SYE+90]. Therefore, each of these DBMSs provides some type of federation 4 where the federated schema is commonly defined for all the nodes that participate in the collaboration. The so-called federated schema is created based on each node decision about what information is commonly shared. A node can also decide at any time to extend, restrict, or remove its shared data. However, the federated approach provided by those systems represent in fact a very limited federation where a node is only allowed to share the same data with all the other nodes or do not share any thing.

The UniSQL/M, for instance, is an object-relational multi-database system that enables interoperability of multiple databases. Currently, UniSQL/M supports the integration of the following other database types: UniSQL/X, Oracle, Informix, and Sybase. UniSQL/M gives its applications and users access to those databases through a single SQL interface, however, it does not address the visibility level of shared data at the local databases. Through the registration process, a database has to decide to share every thing or do not share; thus, there is no support for full federation process as required in different types of applications. Furthermore, user access is defined at the global schema without support for any access level, which is very critical in today's applications.

The multidatabase approach, as addressed and provided by some commercial products, better suits specific applications within the same organization in which, different software tools need to adjust their input/output. In such a case (within the same environment), visibility levels may not be highly required and heterogeneity of used systems and data modeling is not very complex.

---

4 Federation provides applications and queries with access to data stored in multiple databases without requiring knowledge of their distributed location.
Table 2.1 illustrates and evaluates some of the commercial database systems in terms of integrating information from different sources. The table shows the product name, the used data model, support for information visibility levels, and the consideration of standards in terms of information modeling, query language, and information exchange. The evaluation illustrates that most of these products are extension to the existing relational DBMSs, in which the use of standards is well considered, while the information visibility levels and the node-to-node federation are not fully taken into account. Data is usually collected from different sources and locally (physically) stored according to the global schema definition.

<table>
<thead>
<tr>
<th>Product</th>
<th>Data Models</th>
<th>Global Schema</th>
<th>Visibility Levels</th>
<th>Use of Standards</th>
<th>Data Export</th>
<th>Query Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>UniSQL/MM</td>
<td>Relational, UniSQL, Oracle, Informix, Sybase</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>SQL</td>
</tr>
<tr>
<td>Sybase</td>
<td>Relational</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>SQL-Like</td>
</tr>
<tr>
<td>Oracle</td>
<td>Relational</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>PL/SQL</td>
</tr>
<tr>
<td>Ingres/Star</td>
<td>Relational</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>SQL-Like</td>
</tr>
</tbody>
</table>

Table 2.1: Commercial Systems Evaluation in Terms of Information Integration

The remaining of this section will address three examples of federated systems for which the adopted architecture is close to satisfy the requirements of today's applications emerging from scientific and industrial domains. The prototypes concern:

1. **PEER system** [ATW+94]: an object-oriented federated information management system supporting the import, export, and integration of heterogeneous and autonomous schemas.

2. **DIMS of PRODNET II** [CA 99]: Distributed Information Management System for an IT platform supporting industrial virtual enterprises and providing mechanisms for inter-operation and information exchange in real time.

3. **WebFINDIT** [BB99a]: a system supporting the database equivalent to the World Wide Web and addressing interoperability in Web accessible Databases.

**Example 1: PEER Federated System**

**PEER** [ATW+93, ATW+94] is an object-oriented federated information management system designed and developed at the University of Amsterdam to support the management, sharing, and exchange of heterogeneous information in a network of loosely/tightly coupled nodes. Using PEER, each node in the federation network can autonomously decide about the information that it locally manages, and which part of its local information it wishes to export and share with other nodes. Each node can import information that is exported by other nodes and then transform, derive and integrate (a part of) the imported information to fit its interest and corresponds to the local interpretation. PEER is a pure federated system; namely there is no need for a single global schema to be defined on the information to be shared by different nodes, and there is no global control among the nodes.

The PEER integration infrastructure helps the human users in a cooperative team, by supporting their information integration at different levels of granularity, e.g. to support

---

5 More details related to the PEER federated system and its application in Waternet will be later provided in Chapter 3.
the global task, or among different activities and sub-activities. The PEER system provides an environment for the cooperation and information exchange among different nodes in a network, where every node is composed of one server process and may consist of several client processes.

The PEER information management strategy supports the sharing and exchange of information among nodes, without the need for data redundancy and/or creation of one global schema. Therefore, the problems of data consistency, referential integrity and update propagation, and the need for a common glossary of concepts and definitions are eliminated.

As depicted Figure 2.9, every node in the PEER layer is represented by several kinds of schemas: one local schema, a number of import and export schemas, and one integrated schema.

- The local schema is the schema that models the information that is available and stored locally within the node,
- The various imports schemas model the information that the node needs to access from other local/remote nodes,
- The export schema models the information that a node wishes to make accessible to other nodes, and
- The integrated schema presents coherent pool of information on all accessible local and remote information.

---

**Example 2: DIMS of PRODNET II**

The European ESPRIT project PRODNET II (1996-1999) [CA 99] designs and develops an open IT platform to support industrial virtual enterprises, with special focus on the needs of Small- and Medium-sized Enterprises (SMEs). The PRODNET project provides a VE support infrastructure, in which the involved SME companies are able to inter-operate and exchange information in real time so that they can work as a single integrated organization, while at the same time keeping their own independence and autonomy.

The proposed infrastructure for PRODNET II [GR 01], depicted in Figure 2.10, is composed of three main components: the Internal Module (including the PPC and other engineering modules), the Advanced Coordination Functionalities, and the PRODNET Cooperation Layer (PCL). More details related to the description of different components of PRODNET can be found in [GAH 01, FAG+00, KRS+99, CL 99, GCL 99, Sch 99, and OAB
This section however, will give a brief description of the DIMS component of the PCL layer, due to its relation with the subject of the thesis.

The DIMS (Distributed Information Management System) is responsible for modeling and managing the exchange of all integrated VE cooperation-related information, while preserving the autonomy and information privacy of the involved enterprises [GAH 01, FAG+00].

The general reference architecture for DIMS embodies the following components (see figure 2.11):

- **VCL Integrated Schema**: provides a unified definition of both the local and the distributed VE information that can be accessed by end-users and applications at each VE node. Other VCL components and external enterprise modules issue federated database queries on this schema through the DIMS server agent, which takes care of the interaction with the federated query processor, or with the local database schema.

- **Export Schema Manager and Tool**: encloses the functionality to create and maintain
the hierarchy of export schemas that are defined on the VCL local schema, based on the visibility of access that need to be specified for a given node.

- **Federated Query Processor**: transparently supports the access to data distributed over the nodes of the VE network, taking into account the specific visibility access rights (represented by export schemas) defined for every node.

- **DIMS Server Agent**: receives and dispatches all the DIMS service requests issued by other VCL modules. The Server Agent first determines the nature of the service requests and then triggers the activation of the involved DIMS internal components.

- **Internal DIMS Database Manager**: represents the server tier that provides the fundamental functionalities expected from a database management system including: transaction management, data storage and retrieval facility, stored procedures managemen, SQL support, database triggers, etc.

- **DIMS Kernel Configurator**: allows the user to specify the configuration of certain DIMS operation parameters (e.g. DIMS users and access security definitions (accounts and passwords), Communication port number of DIMS server, and Timeout duration for distributed queries).

### Example 3: WebFINDIT a System for Querying Web Databases

**WebFINDIT** [BBO+99a] [BBH+99][BBO+99b] focuses on the design and implementation of an architecture to support appropriate tools to manage the description of, location, and access to data in the context of highly dynamic networks of information sources. It proposes the use of flexible organizational constructs called coalitions and service links, to facilitate data organization, discovery, and sharing among Internet accessible databases.

In order to achieve broad and flexible access to those remote information sources, WebFINDIT provides the WebTassili as language that supports the definition and manipulation of information using two levels mechanism for Querying Web Databases. At the meta-data level, users/applications can explore meta-information about a particular database, while at the data level, they can query actual information stored in databases. WebTassili also educates users about the information available and focuses here on those aspects of the language designed specifically for locating information sources and educating users. The WebTassili framework translates WebTassili queries to the native local languages, and translates results from the native systems format to WebTassili.

The WebFINDIT prototype has been implemented using advanced object technology, including CORBA as a distributed computing platform, Java, and Database Connectivity Gateways to access native databases and to connect to databases. The combination of technologies such as CORBA and Java offers a compelling middleware infrastructure to implement wide-area enterprise applications, in which CORBA is used to provide a robust communication infrastructure while Java is used to allow a dynamic deployment of the system over the Web. Different database management systems\(^7\) have been used as a test-bed for the prototype.

As shown in Figure 2.12 [BBO+99a], the WebFINDIT components are grouped in four interactive layers:

1. The **Query Layer** gives users access to WebFINDIT services through two components: the browser and the query processor.

\(^7\)According to [BBO+99a]. The actual WebFINDIT prototype interconnects 26 databases, which are implemented using four different DBMSs: Oracle, mSQL, DB2, and ObjectStore.
2. The **Communication Layer** manages the interactions among WebFINDIT components and mediates requests between the query processor and co-database servers,

3. The **Meta-data Layer** consists of a set of co-database servers that store meta-data about the associated databases (e.g. information type, location, and coalitions).

4. The **Data Layer** has two components: databases and Information Source Interfaces (ISIs). An ISI provides access to a specific database server by delivering requests from the communication layer and retrieving results from the database.

**Discussion of the three Examples**

In comparison to the main aspects of the approach suggested in this dissertation for information integration, hereafter, we briefly discuss the three examples presented above. These examples are discussed in accordance to the detailed requirements of distributed, heterogeneous, and autonomous sites, as described in section 1.1.

The PEER system uses its specific languages for modeling and querying the data, which make the knowledge of PEER a must for every application participating in the federation community. The fact that PEER is not based on standards, makes the process of learning its languages and maintaining its applications a hard task that require appropriate skills and expertise in the field. In addition, PEER by itself is a full DBMS that is required to be used as the database management system for each application within the federation community. In our opinion, a better approach to information integration must preserve the involved systems with their existing DBMSs, and provide a higher level of integration that logically links those DBMSs in a suitable manner.

The DIMS approach, proposed for PRODNET, provides in fact a comprehensive federated solution for a certain type of applications that cooperate together towards the achievement of a common and global goal of the VE. Within the DIMS approach of PRODNET, users and their access rights are well defined, however, the heterogeneity issue of different
sites of the collaboration (e.g. legacy systems) is left to be resolved by the sites themselves. In other words, different collaborative sites share a common data model and a common query language.

WebFIXDIT provides an architecture to manage and query data in the context of highly dynamic networks of information sources. Similar to many other solutions, WebFIXDIT uses its specific languages for data coalitions and information retrieval, which also require appropriate knowledge and expertise of these specific languages and modeling concepts. In addition, users authentication and their access rights to the information sources are not addressed to the required level. From the global architecture point of view, it seems that WebFIXDIT proposes a global solution to access data from external sources, rather than providing an integration mechanism where different application can cooperate and share each others data and services.

### 2.3 Further Classifications and Categorizations

In this section we evaluate some of the approaches for information integration based on two classifications variant, namely, (1) a classification based on the developed approaches, and (2) a classification based on the application requirements.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Application Area</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distributed Systems</td>
<td>Banking, hospital chains, etc.</td>
<td>Availability, Reliability</td>
<td>Off time updates at central nodes, Performance</td>
</tr>
<tr>
<td>Data Warehouse</td>
<td>Archiving, statistics, decision support, and OLAP</td>
<td>Uniform access to data Adequate performance guaranteed at query time</td>
<td>Updates need to be reflected Data is accessed off-line, so not up to date</td>
</tr>
<tr>
<td>Common Data Definition Model</td>
<td>Archiving, statistics, and decision making</td>
<td>Uniform access to data Data is always fresh</td>
<td>Hard to automate Hard to maintain</td>
</tr>
<tr>
<td>Pair-Wise Translation</td>
<td>Specific Applications</td>
<td>Well optimized</td>
<td>Specific interfaces</td>
</tr>
<tr>
<td>General Federated Approaches</td>
<td>Business, and scientific applications</td>
<td>Uniform access to disparate databases</td>
<td>Visibility level is not fully addressed</td>
</tr>
<tr>
<td>Node-to-Node Federation</td>
<td>Scientific domains, Business applications</td>
<td>Full federation, Good visibility level, Data is not replicated, Data is guaranteed to be fresh at query time</td>
<td>Hard to build and to maintain, Difficult to reach an agreement</td>
</tr>
</tbody>
</table>

Table 2.2: Approaches Evaluation based on the Developed Systems

Table 2.2 evaluates some of the approaches for information integration as developed to fulfill the specific requirements in different application domains. It also outlines the advantages and disadvantages of each approach. Therefore, the deployment of an approach.
to be used by an application domain for their information management, is strongly based on the specific requirements of every application. In addition, the final decision concerning the approach to follow also rely on the evaluation and estimation of the gained benefits and advantages against the sacrificed features, which relate to system autonomy, data freshness, and information privacy.

In Table 2.3, we classify the application domains into four categories of Chain Systems, Data Archiving and Cataloguing, Data Publishing, and large scale Applications. The classification, which is based on the main characteristics of each application domain, also determines the application type and provides examples of these applications.

<table>
<thead>
<tr>
<th>Application Type</th>
<th>Approach</th>
<th>Characteristics</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chain Systems</td>
<td>Distributed Systems</td>
<td>Similar environments at geographically distributed sites. Applications may use similar data models and management systems. Data is susceptible to change.</td>
<td>Banking, hospitals, supper-markets, and insurance organizations.</td>
</tr>
<tr>
<td>Data Archiving and Cataloguing</td>
<td>Centralized Databases</td>
<td>Data is gathered from different sources in order to be archived and used by a third party (service). Archived data is not susceptible to change frequently. Archives are usually copyrights protected.</td>
<td>Scientific results. Biology, reference databases, and product information.</td>
</tr>
<tr>
<td>Data Publishing</td>
<td>Data Warehousing</td>
<td>Data is collected from distributed sources. cleaned, filtered, and integrated into one format. Data is susceptible to change and the new updates need to be reflected.</td>
<td>Data mining, and On-line Analysis Processing (OLAP).</td>
</tr>
<tr>
<td>Large scale Applications</td>
<td>Federated Systems</td>
<td>Data needs to be accessed from different remote resources. Data needs to be fresh.</td>
<td>Science environment, health care, business, and industry.</td>
</tr>
</tbody>
</table>

Table 2.3: Approaches Evaluation based on the Application Requirements

2.4 Discussion

Several approaches, among the ones presented in this chapter, address the main aspects of multidatabase systems. Namely, heterogeneity, distribution, and autonomy are addressed with a certain level of details. However, the requirements of today's applications are more complex than what is provided, more precisely:

- **Distribution** must be addressed in a manner were data remains at its originating source while access to it is gained on-line, when needed and in the requested format. Under the normal consideration, up-to-date data must be gathered from multiple disparate data sources.

- **Heterogeneity** must be supported at the level of data representation (including semantic and syntactic heterogeneity) and at the level of DBMSs (e.g. relational model.
CODASYL network model, object-oriented model, and flat files).

- Each system within the collaboration community must fully preserve its local autonomy in terms of controlling its information management.

- The import/export of information at each node must be well controlled and very flexible. A node must be able to define as many export schemas as required by different applications. Each export schema shall represent the part of local information the node wishes to make available for other specific applications. Similarly, a node shall import as many import schemas as needed from its different applications: moreover, information is imported in the format desired by the requesting applications.

- **Bilateral agreements** need to be established between the members of a collaboration network in order to define the information to share, the manner to access it, and the circumstances under which the shared information will be used.

In addition, generally most of the proposed approaches lack the means for generalization and only address specific domain-dependent cases. Adding a new site to the cooperation requires considerable expertise and effort in order to interface it with all communicating systems. Thus, support for applications extensibility and evolutions are not guaranteed in most of these approaches. For instance, the PEER federated system provides a loosely coupled federated environment where autonomy is preserved, users visibility rights are supported, and its federated schemas are well defined. However, the existing implementation of PEER requires that, for their interaction/cooperation, all the nodes (agents) within the federation community use the same data model (the PEER model) and the same query language (PEER language) in their "cooperation layer". Similarly, WebFINDIT proposes an architecture that allows dynamic couplings of Web accessible databases using a common data model and a common query language called WebTassili. Thus, for both systems, expertise is required to build cooperation layers when initiating a new collaboration network or extending the existing federated nodes to support new members.

The approach we propose in chapter 6 is mainly based on and extends the PEER architecture in defining the type of information managed by each node (namely local, import, export, and integrated schemas). In addition, the approach allows different applications to follow widely understood formats and common languages for modeling and querying the data. Further, it preserves all application's autonomy and builds a federated layer on top of each application. Different than the original development of the PEER database system, the information integration approach, proposed in chapter 6, supports the inter-nodes communication based on the usage of middleware solutions and standards: namely, ODL for schema definition, OQL for query formulation, and XML/OIF for data exchange. Moreover, the solution we propose can be considered as a higher abstract level that is used to interconnect different information sources, rather than being a new DBMS intending to replace the existing DBMSs in all networked applications.