

SIXTH FRAMEWORK PROGRAMME
PRIORITY IST-2002-2.3.1.12

Technology-enhanced Learning and Access to Cultural Heritage



Information Society
Technologies



DELOS

Network of Excellence on Digital Libraries
NoE-G038-507618

“Foundations for Information Integration:
A State of the Art”

Workpackage	WP2		
Task	T2		
Date	28/02/2005		
File name	DelosWP2T2.pdf	Version 1.1	
Contributors Partners	Ioanna Koffina (FORTH-ISL) Giorgos Serfiotis (FORTH-ISL) Vassilis Christophides (FORTH-ISL)		
Contact Person	Vassilis Christophides ICS - FORTH Vasillika Vouton, P.O 1385 71110- Heraklion (Greece) Tel/Fax: +30 2810 391628 email: christop@ics.forth.gr		

Executive Summary

Digital libraries can be viewed as an infrastructure for supporting both the creation of information sources and the movement of information across global networks, and moreover the effective and efficient interaction among knowledge producers, librarians, and information and knowledge seekers. Typically, a digital library is a vast collection of objects, which are stored and maintained by multiple information sources, including databases, image banks, file systems, e-mail systems, the WWW, and others. These information sources are typically heterogeneous, in terms of how the objects are stored, organized and managed, and may exist on different platforms. Furthermore, the information sources in digital libraries are evolutionary in the sense that they may be included to or removed from the system. Thus, there is a need to provide the users with the ability to seamlessly and transparently access digital library objects in spite of the heterogeneity and dynamism among the information sources.

The aim of this survey is to thoroughly present and compare the different approaches, schemes, frameworks and systems proposed in the literature for supporting information integration.

Revision Information

Revision date	Version	Changes
1/06/2004	0.1	Draft Plan
1/09/2004	0.5	Detailed Plan
1/11/2004	1.0	First Version
28/02/2005	1.1	Final Version

1 Introduction

With the popularity of the Internet, access to data, independent of its physical storage location, has become highly facilitated. Additionally, users can access a variety of data sources that are related in some way, and combine the returned data in order to discover useful information, not physically stored into a single source.

Definition 1 *A data integration system, is a system that provides users with transparent access to a collection of related data sources as if these sources, as a whole, constitute a single data source. [7] □*

The main objective of a data integration system is to facilitate users to focus on specifying *what* data they want, rather than on describing *how* to obtain them. To achieve this, the system provides an integrated view of the data stored in the underlying data sources. In a data integration system, users are interested mainly in querying the integrated data rather than updating the data through the integrated view.

There are some major issues that should be considered in a data integration system related to its design, modeling and operation. Even if these issues concern every data integration system, they are differentiated according to the variety of data sources that they integrate. The most important issues concern autonomy, queries and heterogeneity of the sources.

Autonomy of the data sources regards the ability of the sources to choose their own design, their own schema (if any), their own data model and their own management of data. Usually data sources are created in advance of the integrated system and they can, autonomously, take decisions about their data. The data integration system is not, usually, informed for such changes a priori. Furthermore, data sources are able to choose whether and when they communicate with other components (communication autonomy). These kinds of autonomy are present in every integration system but they are differentiated. Issues, for example, concerning differences in choosing data model does not influence systems that integrate relational databases, since all the sources have a common data model.

Additionally, a significant issue is that of querying the integration system. Users pose queries formulated in terms of the provided integrated view and these queries should be translated into forms that are appropriate for the data sources. If the integrated sources are relational databases then user queries should be translated into a query language that is common for all the sources (i.e., SQL). However, when the data sources have different data formats this translation is not trivial. Furthermore, possible access limitations of the data sources for answering queries should be taken into account, since some sources may be able to answer only a small fragment of queries. Additionally, traditional query optimization techniques, that usually depend on statistical models, may not be applicable, since such statistical models are not (easily) available for all the sources.

Finally, the problem of the discrepancies between the data sources is highly important. Usually, the contents of data sources are related in some way, but they show diversity in many aspects. This diversity, which is usually referred as *heterogeneity*, causes the design

of a data integration system to be a challenging problem. Resolving the differences between the data sources is a crucial issue. There are different layers of heterogeneity beginning from *hardware* heterogeneities and continuing to discrepancies in the *operating systems* or *communication protocols*. On a higher level there is *logical* heterogeneity, which is the most intricate problem. Heterogeneity exists in any integration system, but in the case of older integration systems (relational database sources) some kinds of discrepancies do not exist. For example, there are no data model conflicts and every source has a schema, which is not ensured in nowadays integrations systems.

2 Data Integration Architectures

Data integration systems can be classified according to their approach for managing sources. One of the possible classifications may be the one based on whether the queries to the integration system are sent directly to the data sources or whether there are results of the queries that are pre-stored. The *virtual view* approach corresponds to the former technique, while the *materialized view* approach uses pre-stored results.

2.1 Virtual View Approach

In the virtual view approach, the data are accessed from the sources on-demand when a user submits a query to the data integration system. The two representative architectures of this approach are federated database systems, and mediated systems. Despite of the fact that mediated systems have many similarities with the federated databases, there are some basic differences:

- In mediated systems data sources are not necessarily databases.
- Sources in a mediated system can be added or removed easily.
- Usually, unlike the FDBSs (where access is read/write), in a mediated system access in the sources is read only [7]. This is due to the fact that sources in mediator-based systems are more autonomous.

2.1.1 Federated Database Systems (Read/Write Systems)

A federated database system (FDBS) consists of some semi-autonomous components that participate in federation to partially share data with each other. Each federated source can also operate independently from the others. These components are not fully autonomous in the sense that they are modified by adding an interface that allows communication with all other databases in the federation. Each component of the federation is either a centralized DBMS, a distributed DBMS or another federated database management system. There are two basic types of FDBS: *tightly coupled* FDBSs and *loosely coupled* FDBSs.

A tightly coupled FDBS has *one or more unified schemas* which can be produced automatically or manually (by a user). In this type of FDBS, domain experts should

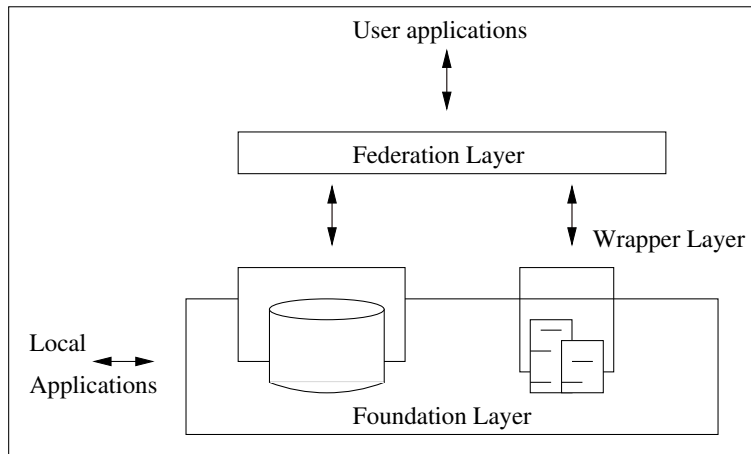


Figure 1: The architecture of a federated database system

undertake the arduous task of integrating all schemas of the federation into a global one. These FDBSs are static and it is very difficult to add or remove components from the integration system.

A loosely coupled FDBS does not have a unified schema, but it provides some *unified language for querying sources*. In this approach database components are more autonomous and they can decide how they will view all the accessible data in the federation. As there is no global schema, each source can create its own “federated schema” for its needs. Like the tightly coupled approach, logical heterogeneity should be resolved by domain experts.

The architecture of a federated database system is depicted in Figure 1. Federated databases is an approach appropriate to use when there is a small number of autonomous sources, and we want to retain their “independence”, allowing user to query them separately and let them collaborate with each other to answer a query.

2.1.2 Mediated Systems (Read Only Systems)

Mediated systems are an alternative architecture of data integration systems. They integrate data sources by providing a global virtual view. This global view is called *mediated schema*, and it is employed by the users in order to formulate their queries. The architecture of a mediated system is shown in Figure 2.

There are two basic software components of a mediated system: the *mediator* and one *wrapper* per data source. The former offers a common interface to a set of autonomous, independent and possibly heterogeneous data sources. The mediator (a.k.a *integrator*) performs the following actions: Firstly it receives a query formulated in terms of the unified schema and decomposes this queries into sub-queries. These queries are addressed to specific data sources. This decomposition is based on source descriptions, which play an important role in sub-queries’ execution plan optimization. Finally, the sub-queries are sent to the wrappers of the individual sources, which transform them into queries over the

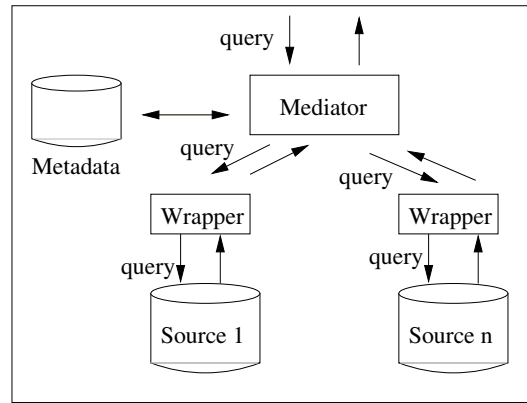


Figure 2: The architecture of a mediated integration system

sources. The results of these sub-queries are sent back to the mediator. At this point the answers are merged into one and returned to the user. Besides the possibility of asking queries, the mediator has no control over the individual sources.

The latter component (wrapper) is responsible for wrapping a data source in such a way that the source can interact with the rest of the integration system. It provides the mediator with data from the source that it is in charge of, as requested by the query execution engine. In consequence, it presents a data source as a convenient database, with the right schema and data, appropriate for being understood and used by the mediator. This presentation schema may be different from the real one, i.e., the internal to the data source. A wrapper hides low-level (protocol) and data model details of the data source from the mediator. It is an important component of both a mediator based architecture and a data warehouse (see Section 2.2).

A key element in the mediator architecture is the set of *source descriptions*, i.e., the descriptions of the available sources and their contents, which is achieved by establishing the relationships (*mappings*) between the global schema and the local schemas. These descriptions can be represented by a set of logical formulas, similar to the way in which views are defined in terms of base tables in the relational data model. The language usually chosen for expressing these mappings is Datalog. There are several fragments of Datalog that are used, based on the existence of recursion, negation and arithmetic comparisons. The most common framework is the one of conjunctive queries (Datalog with relational predicates) with neither recursion nor negation, and arithmetic comparisons limited to equalities. There are different approaches with respect to how mappings are defined, and will be discussed thoroughly in Section 3.

2.2 Materialized View Approach

In the materialized view approach (a.k.a data warehousing) some filtered information from data sources is pre-stored (materialized) in a repository and can be queried later. The single repository in which data are stored is called *data warehouse*.

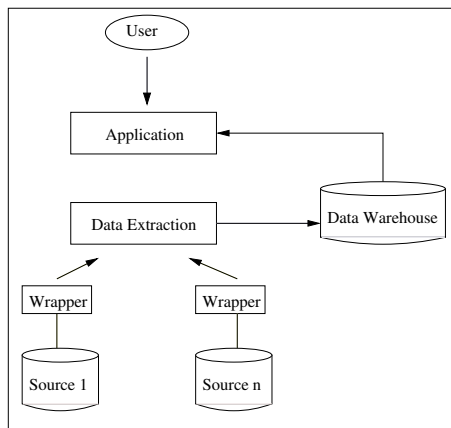


Figure 3: The architecture of a data warehouse

There are some important issues that should be taken into account for designing and maintaining a data warehouse. Firstly (designing phase) we need to decide what information from each source is going to be used, what views over these sources is going to be materialized and what global schema will be employed by the warehouse. Next (maintenance phase) we have to deal with how the warehouse is initially populated by the source data and how it is refreshed when the data in the sources are updated. Finally, there are some query processing, storage and indexing issues that should be taken into consideration. The architecture of the materialized view approach is depicted in Figure 3, where the Data Extraction component is responsible for the maintenance of the information stored in the Data Warehouse. This information is available to the user for querying and/or updating.

3 Modelling the Data Sources

Data integration systems can be classified according to *the way* in which the sources are described, i.e., how their content is related to the mediator global schema. There are two main approaches: the Global-As-View approach (GAV) and the Local-As-View approach (LAV). Furthermore, hybrid approaches based on both GAV and LAV have been recently proposed.

Definition 2 *A data integration system is a triple $\mathcal{I}\langle\mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G}, \mathcal{S}}\rangle$ [18] where*

- \mathcal{G} is the global schema expressed in the global language $\mathcal{L}_{\mathcal{G}}$ over alphabet $\mathcal{A}^{\mathcal{G}}$. The language $\mathcal{L}_{\mathcal{G}}$ determines the expressiveness allowed for specifying the global schema, i.e., the set of constraints that can be defined over it.
- \mathcal{S} is the set of the local schemas. It is modeled in the source language $\mathcal{L}_{\mathcal{S}}$ over the alphabet $\mathcal{A}^{\mathcal{S}}$. As in the case of global schema the language determines the set of constraints that can be defined over it.

- $\mathcal{M}_{\mathcal{G},\mathcal{S}}$ is the mapping between \mathcal{G} and \mathcal{S} . \square

In order to specify the semantics of a data integration system, we start with a set of data at the sources and specify which data satisfies the global schema. We start with a source database for \mathcal{I} , the database \mathcal{D} for the source schema \mathcal{S} . Based on this \mathcal{D} , we have to specify which information content of the global schema \mathcal{G} is. Any database for \mathcal{G} is called a global database for \mathcal{I} . A global database \mathcal{B} for \mathcal{I} is said to be *legal* with respect to \mathcal{D} , if:

- \mathcal{B} is coherent with \mathcal{G} , i.e., every constraint in schema \mathcal{G} is satisfied by \mathcal{B} .
- \mathcal{B} satisfies the mapping with respect to \mathcal{D} , that is its tuples respect the relationships defined between the global and the source schema.

Definition 3 Given a source database \mathcal{D} for \mathcal{I} , the semantics of \mathcal{I} with respect to \mathcal{D} , denoted $sem(\mathcal{I}, \mathcal{D})$, is defined as:

$$sem(\mathcal{I}, \mathcal{D}) = \{ B \mid B \text{ is a legal global database for } \mathcal{I} \text{ w.r.t. } \mathcal{D} \}$$

\square

Let us now turn our attention to queries. In order to define the semantics of a query q over the global schema \mathcal{G} , we have to take into account all the legal global databases for \mathcal{I} with respect to \mathcal{D} . We call *certain answers* of a query q of arity n with respect to \mathcal{I} and \mathcal{D} , the set $q^{\mathcal{I}, \mathcal{D}}$ of n -tuples t such that $t \in q^{\mathcal{D}\mathcal{B}}$ for every database $\mathcal{D}\mathcal{B} \in sem(\mathcal{I}, \mathcal{D})$. Certain answers is what we call, answers to a user query.

Given the above formal definitions, the mapping $\mathcal{M}_{\mathcal{G},\mathcal{S}}$ between the global schema and the sources is provided in terms of a set of assertions of the form $\langle R, V \rangle$, where R is a view over the global schema \mathcal{G} and V is the view over the source schema \mathcal{S} .

Associated to each mapping assertion $\langle R, V \rangle$ we have a specification $as(V)$ of which *assumption* to adopt for the view V , i.e., given a source database \mathcal{D} , how to interpret $R^{\mathcal{D}}$ with respect to the set of tuples in the answer to V over a global database \mathcal{B} , i.e., $V^{\mathcal{B}}$.

Definition 4 The assumption we adopt for a view V , called $as(V)$, to each mapping assertion $\langle R, V \rangle$ is defined as follows:

When $as(V) = \text{sound}$, the extension of the associated global view R provides any superset of the tuples satisfying V . In other words, from the fact that a tuple is in $V^{\mathcal{D}}$ one can conclude that it satisfies the corresponding global relation R , while from the fact that a tuple is not in $V^{\mathcal{D}}$ one cannot conclude that it does not satisfy R . Formally a global database \mathcal{B} satisfies the sound view V if $V^{\mathcal{D}} \subseteq R^{\mathcal{B}}$

When $as(V) = \text{complete}$, the extension of the associated global view R provides any subset of the tuples satisfying V . In other words, from the fact that a tuple is in $V^{\mathcal{D}}$ one cannot conclude that such a tuple satisfies R . On the other hand, from the fact that a tuple

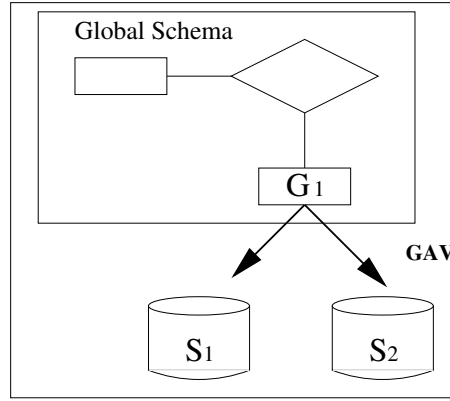


Figure 4: Global centric approach

is not in V^D one can conclude that such a tuple does not satisfy R . Formally, a global database \mathcal{B} satisfies the complete view V , if $V^D \supseteq R^{\mathcal{B}}$.

When $as(V) = exact$, the extension of the associated global relation R is exactly the set of tuples satisfying V . Formally, a global database \mathcal{B} satisfies the exact view V , if $V^D = R^{\mathcal{B}}$. \square

3.1 Global Centric Approach

In the Global-As-View approach (GAV) [30], the global schema is defined in terms of the local sources' schemas. That is, the global schema is defined as a *view* over the local sources' schemas. An overview of the approach is illustrated in Figure 4, where concept G_1 of the global schema is expressed in terms of the relations S_1 and S_2 of the local database sources. If both these schemas are relational, then one can write a *rule-based conjunctive query* over the source relations. This query specifies how to obtain the tuples for the global schema relations. Each query specifies that in order to compute the tuples in the relation in the *head* of the rule, one has to go to the *body* of the rule and compute whatever is specified there. The attributes appearing in the head indicate that they are the attributes of interest, thus the others (in the body) can be projected out at the end. If there is more than one rule to compute the same relation, we use all of them and we take the union of the results.

Definition 5 Given the Definition 2 mappings $(\mathcal{M}_{\mathcal{G},\mathcal{S}})$, in GAV approach, are in the form:

$$\mathcal{G}_i(\overline{X}) \leftarrow \mathcal{S}_1(\overline{X}_1, \overline{Y}_1), \mathcal{S}_2(\overline{X}_2, \overline{Y}_2), \dots, \mathcal{S}_k(\overline{X}_k, \overline{Y}_k)$$

where $\overline{X} = \cup_i \overline{X}_i$ and $\mathcal{G}_i, \mathcal{S}_i$ are global and local relations respectively. \square

Example 1 Supposing that we are interested in the cultural domain and both local and global schemas are expressed as relations. Our example global schema consists of three

relations:

Artists(Name, Year of Birth),

where the two attributes represent the name and the year of birth of each particular artist.

Artifact(Title, Style, Artist, Kind),

where Title is the name of the artifact, Style refers to the technique used for the creation of the artifact, Artist refers to the creator of the artifact and Kind is the type of the artifact (e.g., sculpture, painting).

ExhibitedInMuseum(Title, Museum, Period),

where the Title represents the name of the artifact, Museum is the name of the museum that the artifact is exhibited in, and Period is the time period that the artifact was exhibited in this particular museum.

Now assume that there are three local sources which consist of one relation $S1$, $S2$ and $S3$ respectively. $S1$ contains the name of the artist and the year of their birth. $S2$ contains the id of an artifact, its title, kind, style and the artist that created it. Finally, $S3$ contains information about the museums, the artifacts exhibited in these museums and the period of the exhibition.

The queries that specify how to obtain the tuples for the global schema relations are the following:

$Artist(Name, Year\ of\ Birth)$	\leftarrow	$S1(name, year\ of\ birth)$
$Artifact(Title, Style, Artist, Kind)$	\leftarrow	$S2(id, title, style, kind, artist)$
$ExhibitedInMuseum(Title, Museum, Period)$	\leftarrow	$S3(museum, id, period),$ $S2(id, title, style, kind, artist)$

These queries state that tuples for the global schema relation *Artist* are constituted by source $S1$, *Artifact* tuples are constituted by source $S2$ and *ExhibitedInMuseum* tuples are constituted by a join of the tuples originated from sources $S2$ and $S3$ ¹. \square

Definition 6 Given the formal definitions 4 and 5, the mapping $\mathcal{M}_{G,S}$ between the global schema and the sources, constitutes of the following assertions:

$r^B \supseteq V^D$ (sound source)

or

$r^B \equiv V^D$ (exact source)

where r is a relation of the global schema, and V is a view (query) over the local sources.

\square

It is an implicit assumption [19], in many GAV proposals, that the assertions above are exact. This assumption is true when in the global schema there are *no additional* constraints. Under these circumstances, the query rewriting in GAV approach is quite easy (this will be illustrated below). However, the possibility of specifying constraints in the global schema enhances the expressive power of GAV systems.

¹Global schema attributes' names start with a capital letter and the corresponding attributes from the local schemas start with a lowercase letter.

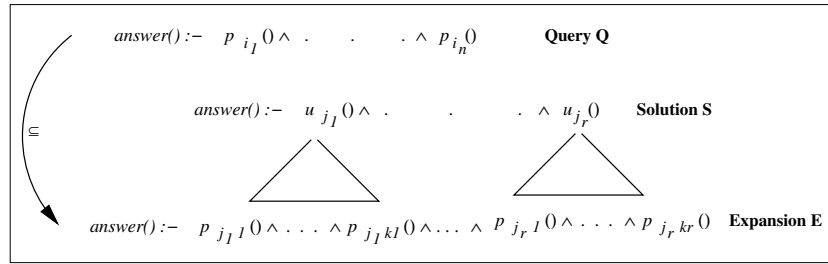


Figure 5: Query unfolding

As depicted in Figure 5 [30] the Global-As-View approach has the advantage of simple query rewriting. Due to the fact that the global schema is expressed in terms of the local schemas query rewriting consists of replacing each atom of the query with its definition. In this way the query is finally expressed in terms of the local sources. This substitution is called *query unfolding*.

Example 2 Based on Example 1 we are interested in a query (written as rule-based conjunctive query) returning the title, the style and the period of paintings, created by Da Vinci and exhibited in the Louvre:

$$\text{query}(\text{Title}, \text{Style}, \text{Period}) \leftarrow \text{ExhibitedInMuseum}(\text{Title}, \text{'Louvre'}, \text{Period}), \\ \text{Artifact}(\text{Title}, \text{Style}, \text{'Da Vinci'}, \text{'Painting'})$$

With the help of the descriptions for the global schema, the resulting query will be:

$$\text{query}'(\text{Title}, \text{Style}, \text{Period}) \leftarrow \text{S3}(\text{'Louvre'}, \text{id}, \text{Period}), \\ \text{S2}(\text{id}, \text{Title}, \text{Style}, \text{'Painting'}, \text{'Da Vinci'})$$

In query' relations *ExhibitedInMuseum* and *Artifact* have been replaced by *S2*, *S3* and *S2* respectively, but only one appearance of *S2* is kept. \square

The major drawback of this approach is its lack of flexibility with respect to the addition/deletion of the sources to the data integration system, or the modification of the sources schemas. This is due to the fact that each modification of a local source schema results in modification of global schema.

3.2 Local Centric Approach

In the Local-As-View approach (LAV) [23] the global schema is defined *independently* of the local sources schemas. Each source is described in terms of the global schema relations. That is, the sources are described as *materialized view* of the global schema. An overview of the LAV approach can be seen in Figure 6 where S_1 can be seen as a view over concepts G_1 and G_2 , while S_2 can be seen as a view over G_2 .

Definition 7 Given the Definition 2 mappings $(\mathcal{M}_{G,S})$, in LAV approach, are in the form:

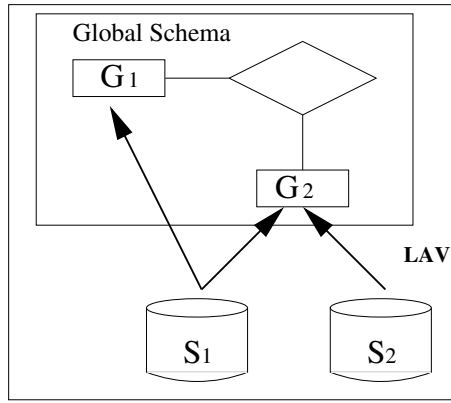


Figure 6: Local centric approach

$$\mathcal{S}_i(\overline{X}) \leftarrow \mathcal{G}_1(\overline{X}_1, \overline{Z}_1), \mathcal{G}_2(\overline{X}_2, \overline{Z}_2), \dots, \mathcal{G}_j(\overline{X}_j, \overline{Z}_j)$$

where $\overline{X} = \cup_i \overline{X}_i$ and $\mathcal{G}_i, \mathcal{S}_i$ are global and local relations respectively. \square

Example 3 Let's assume again that the global schema consists of the three relations defined in Example 1. Suppose that there are two sources $S1$ and $S2$ that are integrated in this data integration system. The descriptions of these sources are:

$S1(\text{Name}, \text{Title})$	\leftarrow	$\text{Artifact}(\text{Title}, \text{Style}, \text{Artist}, \text{Kind}),$ $\text{Artist}(\text{Name}, \text{Year of Birth}),$ $\text{Name} = \text{Artist}, \text{Year of Birth} > 1900$
$S2(\text{Title}, \text{Period})$	\leftarrow	$\text{Artifact}(\text{Title}, \text{Style}, \text{Artist}, \text{Kind}),$ $\text{ExhibitedInMuseum}(\text{Title}, \text{Museum}, \text{Period}),$ $\text{Kind} = \text{'Painting'}, \text{Museum} = \text{'Louvre'}$

From the descriptions above we can conclude that relation $S1$ contains the title of artifacts and the name of their creator, only for creators born after 1900, while source $S2$ contains the titles of paintings along with the period of their exhibition in the Louvre museum. \square

Definition 8 Given the formal definitions 4 and 7, the mapping $\mathcal{M}_{G,S}$ between the global schema and the sources, constitutes of the following assertions:

$$s^{\mathcal{D}} \subseteq V^{\mathcal{B}} \text{ (sound source)}$$

or

$$s^{\mathcal{D}} \equiv V^{\mathcal{B}} \text{ (exact source)}$$

where s is a relation of the local schema, and V is a view (query) over the global schema relations. \square

Due to the fact that the source relations are expressed as views over the global schema, each modification/addition/deletion of sources is costless since the local sources' schemas

are the only things that should change. However, query rewriting in this approach is quite complex. The user of the data integration system poses a query in terms of the global schema and this query should be translated in terms of the local ones. The Local-As-View approach provides the system with a description of the local sources over the global schema and no description in the other “direction”, as in the Global-As-View approach. So, it is not possible for the query to obtain the answers by a simple and direct computation of the body of the query. However, the data reside in the sources, and the query should be translated against them. This problem is widely known as the problem of *rewriting queries using views*.

The problem of query rewriting using views will become clearer if we consider the following: Let’s assume that we have a collection of views V_1, V_2, \dots, V_n , whose contents have already been computed, and cached or materialized. When a new query Q arrives, instead of computing its answers directly, we try to use the answers stored to V_1, V_2, \dots, V_n . A problem to consider lies in determining *how much* from the real answer we get by using the pre-computed views only; and also determining *the maximum* we can get in terms of the kind of views we have available.

3.3 Query Reformulation Algorithms

Definition 9 *With respect to the Definition 2 we assume a query Q expressed over the alphabet \mathcal{A}^G and a rewriting Q' expressed over the alphabet of the sources \mathcal{A}^S . Given this query Q and a set of views $V = \{V_1, \dots, V_n\}$ over the \mathcal{A}^G , the expression Q' defined over \mathcal{A}^S is a rewriting of Q if $Q' \cup V$ is contained in Q and Q' does not refer to the predicates of \mathcal{A}^G .*

The problem of query containment for conjunctive queries is NP-complete [29]. Below, we study four algorithms that deal with the problem of query rewriting.

3.3.1 The Bucket Algorithm

The main idea underlying the Bucket Algorithm [22] is that we can reduce the number of query rewritings that needs to be considered if we consider each subgoal in the query separately to determine which views may be relevant to each subgoal. Given a query Q the Bucket Algorithm finds a rewriting of Q in two steps:

1. The algorithm creates a bucket for each subgoal in Q that contains the views (i.e., data sources) that are relevant to answering that particular subgoal.
2. The algorithm tries to find query rewritings that are conjunctive queries, each consisting of one conjunct from every bucket. For each possible choice of element from each bucket, the algorithm checks whether the resulting conjunction is contained in the query Q , or whether it can be made to be contained if additional predicates are added to the rewriting. If so, the rewriting is added to the answer. Hence, the result of the Bucket Algorithm is a union of conjunctive rewritings.

The algorithm above can become more clear with the following example:

Example 4 *Let's assume again that the global schema consists of the three relations defined in Example 1. Suppose that there are two sources S1 and S2 that are integrated in this data integration system. The descriptions of these sources are the same as in Example 3. Suppose that the query posed by the user to the global schema calculates the sculptures' title, along with their creator and creator's year of birth:*

$$Q(\text{Title}, \text{Name}, \text{Year of Birth}) \leftarrow \begin{array}{l} \text{Artifact}(\text{Title}, \text{Style}, \text{Artist}, \text{Kind}), \\ \text{Artist}(\text{Name}, \text{Year of Birth}), \\ \text{Kind} = \text{'Sculpture'}, \text{Artist} = \text{Name} \end{array}$$

The buckets that are constructed for this example using the first step of the Bucket Algorithm are:

$$\begin{array}{c|c} \text{Artifact}(\text{Title}, \text{Style}, \text{Artist}, \text{Kind}) & \text{Artist}(\text{Name}, \text{Year of Birth}) \\ \hline \text{S1}(\text{Name}, \text{Title}) & \text{S1}(\text{Name}, \text{Title}) \end{array}$$

The first line shows the subgoals of the user query whereas the second line shows the sources S_i that are related to these subgoals. It is true that subgoal *Artifact* is also related with $S2$ but the restriction of *Kind* ($\text{Kind} = \text{'Sculpture'}$) in the query does not allow us to add the source $S2$ ($\text{Kind} = \text{'Painting'}$), since they are contradictory.

The second step of the algorithm chooses one view from each bucket and it combines them into a new query. In this example we have already one entry per bucket, so there is only one combination of views. In general, we should construct one query per possible combination of entries and test for containment in the original query. Then, the result would be the union of all the contained queries.

The new query, written in terms of the sources is:

$$Q'(\text{Title}, \text{Name}, \text{Year of Birth}) \leftarrow \text{S1}(\text{Name}, \text{Title})$$

In this query, the second appearance of $S1$ has been eliminated. \square

This algorithm is able to reformulate queries of the form:

$$Q(\bar{X}) : - C(\bar{Y}), S_1(\bar{X}_1), \dots, S_k(\bar{X}_k)$$

It concerns, conjunctive queries where $S_i(\bar{X}_i)$'s are first-order predicates and $C(\bar{Y})$ is a conjunction of *order constraints* on the variables of the query. Order constraints are of the form $\alpha_1 \theta \alpha_2$, where α_1 and α_2 are variables or constants, and $\theta \in \{<, \leq, =, \neq\}$.

It is guaranteed to find a maximally contained rewriting of the query if it does not contain arithmetic comparison predicates and there are no functional dependencies over the global schema. The Bucket algorithm has to perform a lot of containment tests and this is quite expensive, since testing containment of conjunctive queries is NP-complete [15].

3.3.2 Inverse Rules Algorithm

The key idea underlying this algorithm is to construct a set of rules inverting the view definitions, i.e., rules that show how to compute tuples for the mediated schema relations from tuples of the views [10] [12] [20]. One can think of this process as obtaining GAV definitions out of LAV ones. One inverse rule is constructed for every subgoal in the body of the view. While inverting the view definitions, the existential variables that appear in the view definitions are mapped using Skolem functions to ensure that the value equivalences between the variables are not lost.

Example 5 *Let's consider a slightly different description of the source $S1$ given in Example 3:*

$$S1(\textit{Artist}, \textit{Period}) \leftarrow \textit{Artifact}(\textit{Title}, \textit{Style}, \textit{Artist}, \textit{Kind}), \\ \textit{ExhibitedInMuseum}(\textit{Title}, \textit{Museum}, \textit{Period})$$

For every subgoal of the view definition we should write an inverse rule:

$$\textit{Artifact}(f_1(\textit{Artist}, \textit{Period}), f_2(\textit{Artist}, \textit{Period}), \textit{Artist}, f_3(\textit{Artist}, \textit{Period})) \\ \leftarrow S1(\textit{Artist}, \textit{Period})$$

$$\textit{ExhibitedInMuseum}(f_1(\textit{Artist}, \textit{Period}), f_4(\textit{Artist}, \textit{Period}), \textit{Period}) \\ \leftarrow S1(\textit{Artist}, \textit{Period})$$

In this algorithm, the attributes \textit{Title} , \textit{Style} , \textit{Kind} and \textit{Museum} are replaced by a Skolem function f_i which takes all the attributes of the view ($S1$) head as input. The rewriting of a query Q using the set of views V is the Datalog program that includes the inverse rules for V and the query Q . Given the query:

$$Q(\textit{Artist}, \textit{Period}) \leftarrow \textit{Artifact}(\textit{Title}, \textit{Style}, \textit{Artist}, \textit{Kind}), \\ \textit{ExhibitedInMuseum}(\textit{Title}, \textit{Museum}, \textit{Period})$$

This query is reformulated to (where for simplicity we assume that $T=\textit{Title}$, $S=\textit{Style}$, $A=\textit{Artist}$, $K=\textit{Kind}$, $M=\textit{Museum}$, and $P=\textit{Period}$):

$$Q(A, P) \leftarrow \textit{Artifact}(f_1(A, P), f_2(A, P), A, f_3(A, P)), \\ \textit{ExhibitedInMuseum}(f_1(A, P), f_4(A, P), P), \\ T=f_1(A, P), S=f_2(A, P), K=f_3(A, P), \\ M=f_4(A, P)$$

By using the inverse rules defined previously we conclude into:

$$Q(A, P) \leftarrow S1(A, P), S1(A, P), \\ T=f_1(A, P), S=f_2(A, P), K=f_3(A, P), \\ M=f_4(A, P)$$

By eliminating what we don't need and by replacing the variables introduced before, we come up with:

$$Q'(Artist, Period) \leftarrow S1(Artist, Period)$$

□

This algorithm reformulates recursive conjunctive queries in presence of functional or full dependencies. It is guaranteed to find a maximally contained rewriting in polynomial time in the size (number of predicates) of the query and the views.

3.3.3 MiniCon Algorithm

MiniCon Algorithm [28] is an improved version of the Bucket Algorithm. As in the Bucket Algorithm, there are two steps: computing the buckets (one for each subgoal of the query) and then computing the rewritings by using the buckets. Additionally, MiniCon Algorithm is attentive to the interaction of the variables in the query and in the view definitions, in order to prune some of the views that will be added into the buckets. This way, the number of views to be considered for the rewriting step is reduced. MiniCon Algorithm considers conjunctive queries, as Bucket Algorithm does.

Example 6 Let's consider a slightly different description of the sources given in Example 3:

$S1(Name, Style) \leftarrow \begin{aligned} &Artifact(Title, Style, Artist, Kind), \\ &Artist(Name, Year\ of\ Birth), \\ &Name = Artist, Year\ of\ Birth > 1900 \end{aligned}$
$S2(Title, Period) \leftarrow \begin{aligned} &Artifact(Title, Style, Artist, Kind), \\ &ExhibitedInMuseum(Title, Museum, Period), \\ &Kind = 'Painting', Museum = 'Louvre' \end{aligned}$

Suppose that the query posed by the user to the global schema calculates the title of the paintings and the period of their exhibition:

$$Q(Title, Period) \leftarrow \begin{aligned} &Artifact(Title, Style, Artist, Kind), \\ &ExhibitedInMuseum(Title, Museum, Period), \\ &Kind = 'Painting' \end{aligned}$$

The buckets that are constructed for this example with the help of the first step of the Bucket Algorithm are:

$Artifact(Title, Style, Artist, Kind)$	$ExhibitedInMuseum(Title, Museum, Period)$
$S2(Title, Period)$	$S2(Title, Period)$
$S1(Name, Style)$	

The first line shows the subgoals of the user query whereas the other lines show the sources S_i that are related to these subgoals.

By examining the variables of the source S_1 , it becomes obvious that it does not contribute to the query, since it does not contain the variable in which the subgoals of the query Q join (Title). So, it can be removed from the bucket, leaving only source S_2 .

The new query, written in terms of the sources is:

$$Q'(Title, Period) \leftarrow S_2(Title, Period)$$

In this query, the second appearance of S_2 has been eliminated. \square

The key advantage of the MiniCon Algorithm is that, in the end, it considers fewer combinations than the cartesian product of the views in the buckets created by the Bucket Algorithm. The complexity of the algorithm is exponential in terms of the number of the views, but in general it outperforms both the Bucket and the Inverse Rules Algorithm.

3.3.4 The Shared Variable Bucket Algorithm

This algorithm [26], like the MiniCon, also aims at recovering the weak aspects of the Bucket Algorithm to achieve a more efficient query reformulation. It examines conjunctive queries, like the ones taken into consideration on the Bucket Algorithm. The idea is again to examine the shared variables and reduce the bucket contents so, that the number of view combinations to be considered is reduced in the second phase of the algorithm. The complexity of the algorithm is in the worst case exponential with respect to the size of the query and the views.

3.4 Combining Global and Local Approach

As discussed earlier, both GAV and LAV have some drawbacks that should be overcome. Thus, an approach has been proposed that combines global and local approach. It is called *GLAV* [14]. In this approach we are able to express a local source in terms of the global schema (LAV), a global source in terms of the local sources (GAV) and *additionally* a whole view (“part”) of the global schema in terms of the local sources. An overview of this approach can be seen in Figure 7 where the whole “relation” of G_1 and G_2 is described by the local sources relations, S_1 and S_2 .

Definition 10 Given the Definition 2 mappings $(\mathcal{M}_{\mathcal{G},\mathcal{S}})$, in *GLAV* approach, are in the form:

$$\mathcal{G}_1(\overline{X}_1, \overline{Z}_1), \mathcal{G}_2(\overline{X}_2, \overline{Z}_2), \dots, \mathcal{G}_j(\overline{X}_j, \overline{Z}_j) \leftarrow \mathcal{V}(\overline{X}, \overline{Y})$$

where $\overline{X} = \cup_i \overline{X}_i$, $(\cup_i \overline{Z}_i) \cap \overline{Y} = \emptyset$, \mathcal{G}_i are global relations and $\mathcal{V}(\overline{X}, \overline{Y})$ is a conjunction of source relations. \square

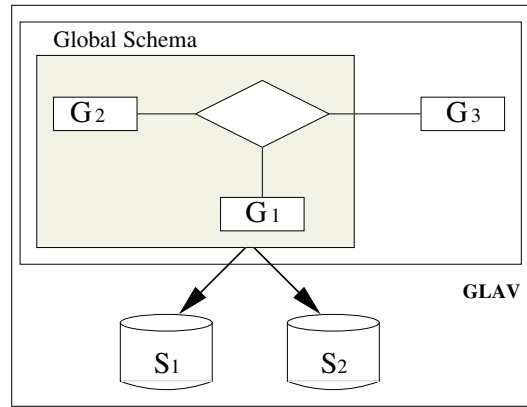


Figure 7: GLAV approach

Example 7 Let's assume that the global schema consists of the three relations described in example 1 and that there are two local sources $S1$ and $S2$. $S1$ describes the artifacts, while $S2$ describes the artifacts that are exhibited in a museum for a certain period and the artist that created them. The queries that specify how to obtain the tuples for the global schema relations are the following:

$Artifact(Title, Style, Artist, Kind),$	$\leftarrow S2(title, artist, museum, period)$
$ExhibitedInMuseum(Title, Museum, Period)$	
$Artifact(Title, Style, Artist, Kind)$	$\leftarrow S1(id, title, style, kind, artist)$

The second query describes the global relation $Artifact$ in terms of the local source $S1$ (typical GAV approach), while in the first query the **conjunction** of the two relations ($Artifact$ and $ExhibitedInMuseum$) is described in terms of the source $S2$. This type of descriptions differentiates the GLAV approach. \square

Definition 11 Given the formal definitions 4 and 10, the mapping $\mathcal{M}_{G,S}$ between the global schema and the sources, constitutes of the following assertions:

$$R^B \supseteq V^D \text{ (sound source)}$$

or

$$R^B \subseteq V^D \text{ (complete source)}$$

or

$$R^B \equiv V^D \text{ (exact source)}$$

where R is a view (query) of the global schema, and V is a view (query) over the local sources. \square

The GLAV approach combines the expressive power of GAV and LAV. Additionally it reaches the limits of the expressive power of a data source description language. This is because slight additions to the expressive power of GLAV would make query answering co-NP-hard in the size of the data in the sources. Query rewriting in this approach is shown to be no harder than it is for the LAV approach. It should be noted that GLAV is

also of interest for data integration independent of data sources, because of the flexibility it provides in integrating heterogeneous sources.

3.4.1 Another Hybrid Approach: BAV

BAV is another data integration approach. BAV transformation sequences are partially derived from LAV or GAV view definitions. BAV is a rich integration framework, which is based on the use of reversible sequences of primitive schema transformations, called *transformation pathways*. It is an expressive data integration language, since it allows the expression of mappings in both directions. Another major advantage of using the BAV approach is that it supports the evolution of both global and local schemas, in contrast to taking either a GAV or LAV approach. With the BAV approach it becomes possible to extract a definition of the global schema as a view over the local schemas and vice versa. BAV combines the benefits of LAV and GAV in the sense that any reasoning or processing which is possible with the view definitions of GAV or LAV will also be possible with the BAV definition. However, BAV is likely to be more costly to reason with and process than the corresponding LAV, GAV or GLAV view definitions would be. The most representative system that implements this approach is AutoMed [3] [16].

3.5 Comparison of the Approaches

In conclusion, there are some interesting facts that should be noted about the approaches above. It is true that both LAV and GAV, which are the first approaches proposed, have advantages and disadvantages. LAV is really flexible in addition/deletion of the local sources that participate in the integration system; this is the main drawback of the GAV approach, since every addition/deletion leads to a new rewriting of the global schema description. Exactly the same occurs when it is necessary to add some more complicated constraints on sources, since LAV demands only the addition of the necessary changes to the source, while GAV demands the rewriting of the global schema description. These constraints usually concern the availability of the data during querying the sources. However, query answering is quite simple in GAV, whereas it is harder in LAV. In the GAV approach, query rewriting can be achieved by unfolding the source descriptions of the global “parts” of the query. In the LAV approach, this is not feasible since such descriptions are not available. In the GLAV approach, the source evolution and addition/removal is easier, since, in fact, both directions are implemented (LAV and GAV) and in this case it is appropriate to use the LAV approach. Additionally, query answering is not harder than in LAV. However, GLAV gives the ability of more expressive mappings.

An interesting observation, as stated in [4], is that, under certain circumstances, the existence of constraints (e.g., keys or foreign keys) in the global schema can turn the GLAV mappings into GAV ones, and thus take benefit of the query reformulation algorithm proposed for the GAV approach. Unfortunately, despite their expressiveness, GLAV mappings introduce new challenges. Recent work [24] has shown that the composition of GLAV mappings may be undecidable in certain cases (e.g., the composition of GLAV rules, which map

non CQ_k^2 queries over a source schema to non CQ_k queries over a target schema, may result in an infinite set of mappings). Additionally, as illustrated in [13] the composition of two GLAV rules does not always imply a new GLAV rule that maps a conjunctive query to another conjunctive query. There are cases (e.g., when we compose two finite sets of non full³ source-to-target dependencies used for the interpretation of the mappings) where the composition is definable only with the use of *existential second-order formulas*. In these formulas, new function symbols that guarantee the presence of the existentially quantified variables appearing in the dependencies, are introduced.

3.6 Representative Systems

Below we discuss briefly the most representative systems of the above source modeling approaches, namely Tsimmis, MIX, Nimble, Information Manifold, Infomaster, Agora, MARS, Dis@Dis.

Table 1 summarizes the basic characteristics of these information integration systems.

3.6.1 GAV Systems

One of the first systems (and the most representative of GAV approach) is Tsimmis system [6]. This system uses the OEM (Object Exchange Model) [27] to convey information between the components of the system. The first basic component of a mediated system, the mediator, is specified using MSL (Mediator Specification Language). It is a logic-based, object-oriented language that can be seen as a view definition language, targeted to the OEM data model. The second one is the wrappers which are specified using WSL (Wrapper Specification Language). WSL is an extension of MSL, supporting the description of source contents and source query capabilities. End users pose queries, which are written in LOREL [1], an extension of OQL targeted to semi-structured data. LOREL queries are translated to MSL queries, which are forwarded to the mediator. The architecture of the Tsimmis is illustrated in Figure 8. An important assumption in the Tsimmis system, is that sources to be integrated have a common way to identify their objects.

Other GAV systems are: MIX and Nimble. MIX (stands for Mediation of Information using XML) [2] is a successor of Tsimmis. The basic difference from Tsimmis is that XML is used as the language (i) to represent the global schema and (ii) to exchange data between the mediator and the XML sources (instead of OEM). The query language of MIX is XMAS (XML Matching And Structure Language) instead of LOREL. XMAS uses features from several XML query languages and it allows object fusion and pattern matching on XML data. XMAS queries are formulated in terms of the mediator schema, and are rewritten into XMAS queries that refer to the source views exported by the wrappers. These queries are then sent to the wrapped sources for evaluation.

²Informally, CQ_k is the class of conjunctive queries in which every nested expression has at most k variables [24].

³A dependency is full if no existentially quantified variables occur in it.

	Approach	Sources	Global View	Query Language	Query Reformulation Algorithm
Tsimmis	GAV	Text files RDB OODB	OEM	LOREL/ MSL	Query Unfolding
MIX	GAV	XML	XML	XMAS	Query Unfolding
Nimble	GAV	XML	XML	XML-QL	Query Unfolding
Information Manifold	LAV	Structured files,OODB RDB, www form-based bases	Relational	SQL	Bucket Algorithm, MiniCon
Infomaster	LAV	Semi- structured	Relational	conjunctive queries	Inverse Rules Algorithm
Agora	LAV	RDB/XML	XML DTD	XQuery	Query rewriting using SQL views
MARS	GLAV	RDB/XML	XML	XQuery	Chase/ Backchase
Dis@Dis	GLAV	Relational	Relational	union of conjunctive queries	Variant of inverse rules
AutoMed	BAV	Relational XML and structured files	Relational	IQL	Query Unfolding

Table 1: Basic characteristics of the information integration systems

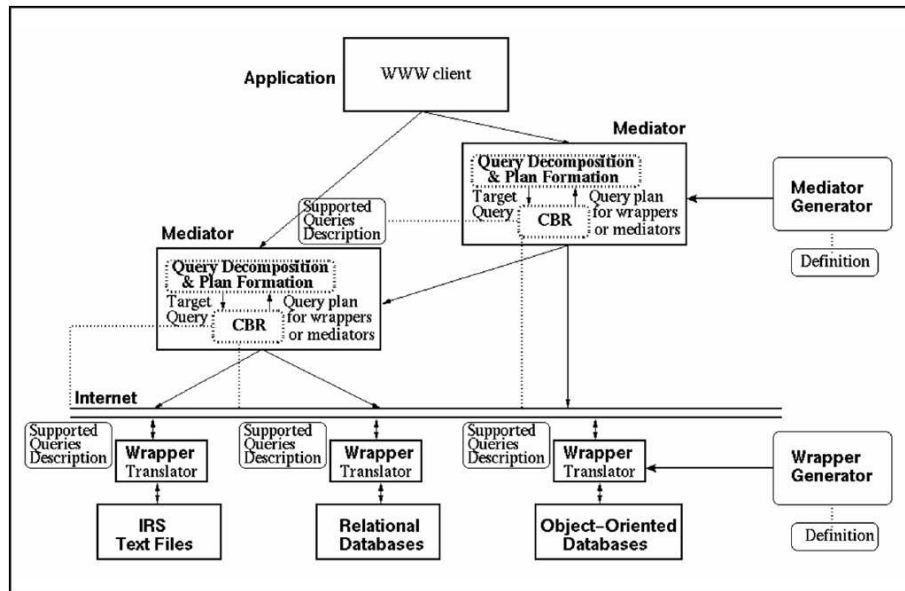


Figure 8: The basic components of Tsimmis System

Similar to the MIX system is the commercial system called Nimble [9]. Nimble integrates XML sources too. The architecture of Nimble system is based on a set of mediated schemas, which are defined as views over the schemas of the data sources. Similar to Tsimmis, the mediated schemas can be built in a hierarchical fashion, that is, a mediated schema can be defined on another mediated schema or on an exported source schema. The query language used by the Nimble system is XML-QL. When a query is posed to the integration system, it is decomposed into multiple source queries based on the data sources. The compiler translates each such query into the appropriate query language for the destination source.

3.6.2 LAV Systems

Information Manifold [17] [21] [22] handles the problem of data integration by providing a mechanism to describe declaratively the contents and the query capabilities of the information sources. The contents of the sources are expressed with the help of source descriptions, which are used efficiently by the system to prune the information sources that do not provide any answer to a user query. In Information Manifold system the global schema is relational. A source description is a conjunctive query over the global schema relations, which will be referred to as view. User queries, posed in Information Manifold, are conjunctive queries like source descriptions. They are expressed in terms of the global schema relations. Query rewriting is done by the Bucket algorithm, or an improved version of it, the MiniCon Algorithm, both discussed in Section 3.3. The architecture on Information Manifold is illustrated in Figure 9.

Infomaster [11] is an information system which provides integrated and uniform access

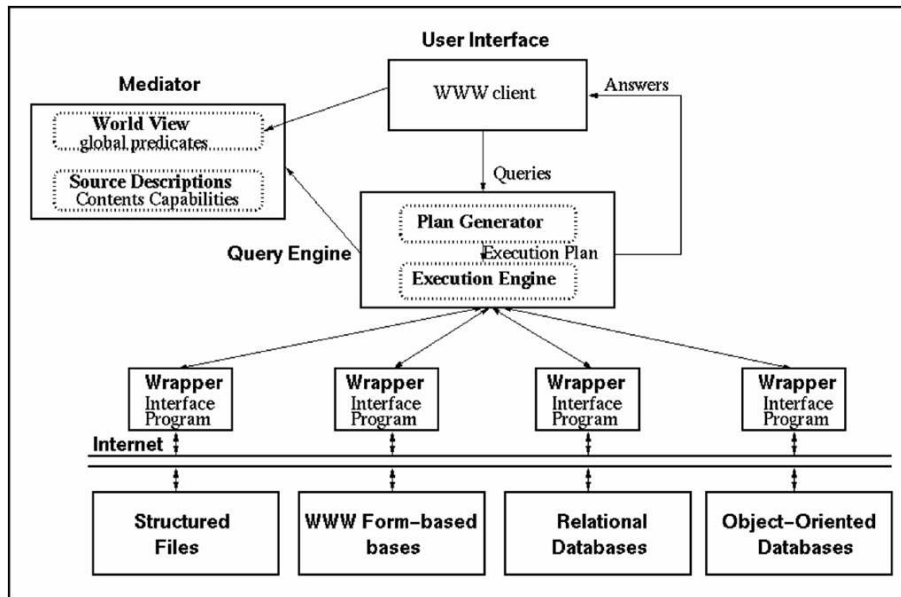


Figure 9: Information Manifold Architecture

to multiple distributed, heterogeneous, structured sources. It uses a three level abstraction hierarchy for modeling the global schema and the sources. Data available in a source is also seen as a set of relations, called site relations. Between site relations and interface relations, a set of base relations is defined. Site relations are described as views over the base relations (following the LAV approach), while interface relations are defined as views on the base relations (following the GAV approach). User queries are expressed in terms of the interface relations and are firstly transformed in terms of the base relations (query unfolding following GAV approach). Then these queries are rewritten in terms of site relations using the Inverse Rules Algorithm (see Section 3.3.2).

Agora [25] system supports querying and integration of heterogeneous relational and XML information sources. The global schema is an XML DTD and a virtual relational schema is used as an interface between the sources and this schema. Relational and XML resources are modeled as SQL queries over the relational global schema. Users formulate XQuery queries in terms of this global DTD. These queries are normalized and translated into an SQL query over the generic relational schema. Query rewriting is based on existing methods and on algorithms for rewriting queries using SQL views.

3.6.3 GLAV Systems

There are two basic representative systems for the GLAV approach: MARS and Dis@Dis.

MARS [8] is a system for publishing as XML data from mixed (relational and XML) proprietary storage, while supporting redundancy in storage for tuning purposes. In this system XML and relational integrity constraints are taken into consideration. It accepts client XQueries formulated against the public schema and with the help of rewriting-

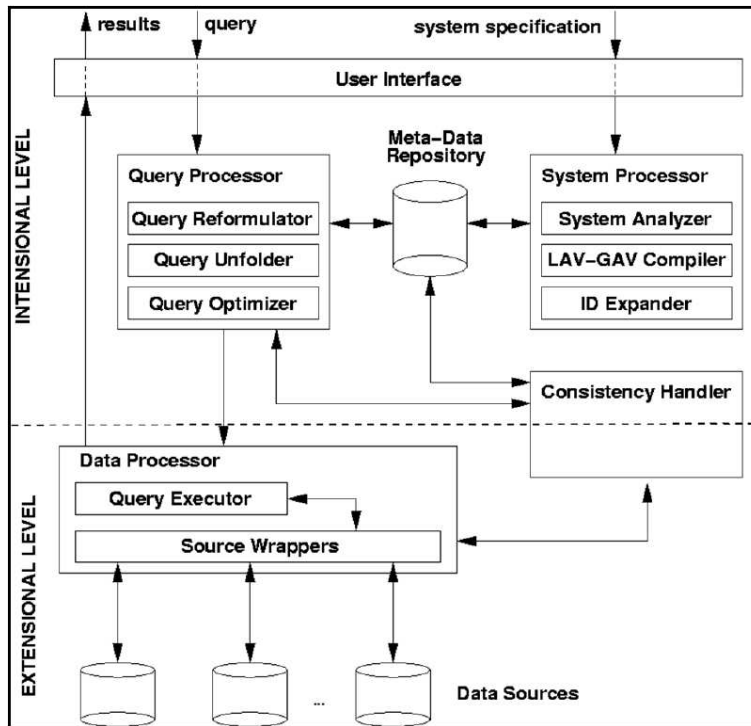


Figure 10: Dis@Dis Architecture

with-views, composition-with-views and query minimization under integrity constraints it achieves optimal reformulation against the proprietary schema. To be able to combine GAV and LAV, MARS is switched to a different, “direction-neutral”, representation of the views, compiling them into constraints.

Dis@Dis [5] deals with the problem of query answering when data does not satisfy integrity constraints. This happens, for instance, in data integration, where integrity constraints that enrich the semantics of the global view are violated by data in the sources. Dis@Dis provides a thorough analysis of data integration (both in the GAV and in the LAV approach) in the presence of both key dependencies (KDs) and inclusion dependencies (IDs). More specifically, system results advance the state-of-the art in data integration in terms of: (i) decidability results for general or key-consistent database instances; (ii) new algorithm computing query answering in the presence of both KDs and cyclic IDs; (iii) a modular treatment of GAV and LAV systems in the presence of integrity constraints, through the compilation of LAV into GAV, and of KDs and IDs, through the separability property of non-key-conflicting (NKC) dependencies; (iv) effective techniques for data integration, which exploit both off-line and intensional processing of queries (this processing is independent of the data in the sources) (Figure 10). Experiments have shown that the intensional processing of queries is actually performed very fast, despite the complexity of the tasks that are computed (which in the worst case require time exponential in the size of the query). Future Dis@Dis extensions include more complex forms of dependencies, e.g., functional and exclusion dependencies.

References

- [1] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener. The Lorel Query Language for Semistructured Data. *International Journal of Digital Libraries*, 1996.
- [2] C. Baru, A. Gupta, B. Ludaescher, R. Marciano, Y. Papakonstantinou, and P. Velikhov. XML-Based Information Mediation with MIX. In *Exhibition Program of ACM SIGMOD Int. Conference on Management of Data, June, 1999*.
- [3] M. Boyd, C. Lazanitis, S. Kittivoraviktul, P. Mc Brien, and N. Rizopoulos. An Overview of The AutoMed Repository. Technical report, Department of Computing, Imperial College, London SW7 2AZ, 2004.
- [4] A. Cali, D. Calvanese, G. De Giacomo, and M. Lenzerini. Data Integration under Integrity Constraints. In *Proceedings of the International Conference on Advanced Information Systems Engineering (CAiSE)*, 2002.
- [5] A. Cali, D. Lembo, and R. Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 2003.
- [6] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS Project: Integration of heterogeneous information sources. In *Proceedings of IPSJ Conference, Tokyo, Japan, October, 1994*.
- [7] C. Convey, O. Karpenko, and N. Tatbul. Data Integration Services, 2001.
- [8] A. Deutsch and V. Tannen. MARS: A System for Publishing XML from Mixed and Redundant Storage. In *Proceedings of the 29th VLDB Conference, Berlin, Germany, 2003*.
- [9] D. Draper, A. Levy, and D. Weld. The Nibble System. In *Proceedings of the International Conference on Database Engineering*, 2001.
- [10] O. Duschka. *Query Planning and Optimization in Information Integration*. PhD thesis, Stanford University, 1997.
- [11] O. Duschka and M. Genesereth. Query Planning in Infomaster. In *Proceedings of the International Conference on Management of Data, SIGMOD, San Jose, CA, USA*, pages 109–111, 1997.
- [12] O. Duschka, M. Genesereth, and A. Levy. Recursive Query Plans for Data Integration. *Journal of Logic Programming*, 43(1):49–73, 2000.

- [13] R. Fagin, P. Kolaitis, L. Popa, and W. Tan. Composing Schema Mappings: Second-Order Dependencies to the Rescue. In *Proceedings of the 23th ACM Symposium on Principles of Database Systems, Paris*, 2004.
- [14] M. Friedman, A. Levy, and T. Millstein. Navigational Plans for Data Integration. In *Proceedings of the sixteenth national conference on artificial intelligence and eleventh innovation applications of AI conference on Artificial intelligence and innovative applications of artificial intelligence*, pages 67–73, 1999.
- [15] H. Garcia-Molina, J. D. Ullman, and J. D. Widom. *Database Systems: The Complete Book*. Prentice Hall, 2001.
- [16] E. Jasper, N. Tong, P. McBrien, and A. Poulouvasilis. Generating and Optimising Views from Both as View Data Integration Rules. In *Proceedings of 6th Baltic Conference on Database and Information Systems (DBIS'04), Riga, June*, 2004.
- [17] T. Kirk, A. Levy, Y. Sagiv, and D. Srivastava. The Information Manifold. In *Proceedings of the AAAI Spring Symposium on Information Gathering in Distributed Heterogeneous Environments, Stanford, CA, March*, 1995.
- [18] D. Lembo, M. Lenzerini, and R. Rosati. Review on models and systems for information integration. Technical report, Universita di Roma “La Sapienza”, 2002.
- [19] M. Lenzerini. Data Integration: A Theoretical Perspective. In *Proceedings of the ACM PODS, Madison, Wisconsin, USA, 3-6 June*, 2002.
- [20] A. Levy. Answering Queries Using Views: A Survey. *The International Journal on Very Large Data Bases*, 2001.
- [21] A. Levy, A. Rajaraman, and J. J. Ordille. Querying Heterogeneous Information Sources Using Source Descriptions. In *Proceedings of International Conference on Very Large Databases (VLDB), Bombay, India, September*, pages 251–262, 1996.
- [22] A. Levy, D. Srivastava, and T. Kirk. Data Model and Query Evaluation in Global Information Systems. *Journal of Intelligent Information Systems*, 1995.
- [23] A. Y. Levy. Logic-Based Techniques in Data Integration. In Jack Minker, editor, *Workshop on Logic-Based Artificial Intelligence, Washington, DC, June 14-16*, College Park, Maryland, 1999. Computer Science Department, University of Maryland.
- [24] J. Madhavan and A. Halevy. Composing Mappings Among Data Sources. In *Proceedings of the 29th International Conference on Very Large Databases(VLDB), Berlin, Germany*, 2003.
- [25] I. Manolescu, D. Florescu, and D. Kossmann. Answering XML Queries over Heterogeneous Data Sources. In *Proceedings of the 27th VLDB Conference, Roma, Italy*, 2001.

- [26] P. Mitra. Algorithms for Answering Queries Efficiently Using Views. Technical report, Infolab, Stanford University, September, 1999.
- [27] Y. Papakonstantinou, H. Garcia Molina, and J. Widom. Object Exchange across Heterogeneous Information Sources. In *Proceedings of the International Conference on Data Engineering (ICDE)*, March, 1996.
- [28] R. Pottinger and A. Levy. A Scalable Algorithm for Answering Queries Using Views. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, Cairo, Egypt, September, 2000.
- [29] Y. Sagiv and M. Yannakakis. Equivalences among Relational Expressions with the Union and Difference Operators. *Journal of the ACM*, 27:633–655, 1980.
- [30] J. Ullman. Information Integration Using Logical Views. *Theoretical Computer Science*, 239(2):189–210, 2000.