

Integrating Autonomous Heterogeneous Information Sources

Rex Jakobovits

Dept. of Computer Science & Engineering
University of Washington

July 15, 1997

I. Introduction		
A. The Human Brain Project		3
B. Schematic Conflicts		4
II. Approaches to Achieving Interoperability		5
A. Overview of Multidatasource Systems		5
1. Federated Database Systems		6
2. Mediator Systems		7
3. Description-Logic Based Systems		7
4. Metadata Repository Systems		7
B. Integration Architectures		8
1. Static Integration Systems		9
2. Dynamic Integration Systems		10
C. Query Handling		12
III. _____ System		
		13
A. Multibase		14
B. TSIMMIS		15
C. HERMES		17
D. Information Manifold		18
E. Metadatabase		21
IV. _____ Alternative Approach		
		22
A. Point-to-Point Gateways		22
B. Data Warehouses		22
C. Distributed Objects		23
D. Shared Class Libraries		23
E. Shared Ontologies		24
V. Conclusion		
VI. _____ F		
		26

I. Introduction

A. *The Human Brain Project*

The human brain is the most complex system on earth, and scientists' efforts to understand the brain are now pushing the limits of database technology, and in the process generating the richest source of multimedia data that science has ever known. The data crosses all levels of the biological structural spectrum, including molecules, cells, systems of cells, and human behavior. The data are highly multidimensional and interconnected: findings at a molecular level may have implications for interpretation of behavioral data. In 1991, a committee of brain researchers and computer scientists recommended a grand undertaking: to develop a national network of databases mapping the brain and its function [CNN91]. In 1993 the Human Brain Project (HBP) was announced, representing an unprecedented coordination of funding from 16 federal organizations, to sponsor a wide range of independent projects for visualizing, simulating, and mapping knowledge about the brain. The field of *neuroinformatics* was born [KH97].

The HBP proposal was divided into two stages. The goal of the first stage was to develop tools and independent databases to aid in neuroscience research. Many of the projects have been extremely successful, generating a plethora of enabling technologies in the form of renderings, atlases, knowledge bases, and neural simulators with hundreds of users. Now the second stage has arrived, in which the individual research groups are mandated to integrate their systems and develop “querying approaches that will allow varied databases to be accessed with a single query, and retrieval of different types of data into a common information space” [KH97].

Computer scientists and informatics researchers have been exploring database integration techniques for twenty years [SL90], but never has their been a domain as challenging as

that posed by the Human Brain Project. As you will see from the systems described in this paper, most attempts at implementing a multidatabase system have had their hands full dealing with the heterogeneities of much simpler domains. This is crossover research, combining complex data type management with integration issues [SSU96]. This paper is a survey of the existing computer science research approaches to achieving interoperability between autonomous heterogeneous data sources.

B. Schematic Conflicts

Before describing approaches to integration, it is important to understand the nature of the challenge. When schema from heterogeneous data sources are compared, the local schemas will exhibit *schematic conflicts*, which naturally arise from uncoordinated representations of a given real world object. These conflicts must be addressed during schema integration or view generation. Six common types of schematic conflicts are listed here [KS91].

Generalization conflicts, in which a class or attribute in one database subsumes multiple classes or attributes in another database. For example, one Brain Map might distinguish between English language sites and Chinese language sites, whereas a different Brain Map might combine them into a single concept.

Data type conflicts, in which different types are assigned to semantically equivalent attributes. For example, a Patient ID could be represented as a string or an integer.

Structural conflicts, in which an entity in one system is modeled as an attribute in another system, e.g. a Surgery could be represented as a simple attribute of a Patient, or it could be modeled as a separate entity.

Missing attributes, such as when the age of a Patient is included in one system but not in another.

Naming conflicts, in which semantically equivalent classes or attributes are assigned different names.

Scale conflicts, such as when distances are measured in inches in one system and centimeters in another, or when different levels of precision are used.

II. Approaches to Achieving Interoperability

This section provides a categorical overview of existing systems, and discusses two major facets of every integration system: the integration architecture, and query handling strategy.

A. Overview of Multidatasource Systems

A “*multidatasource system*” (MDSS), to coin a term, is a collection of information sources which are integrated to some degree to support queries that span across multiple sources. I use the term “sources” as opposed to “databases” because some of the sources may be semi-structured or unstructured, and may not be updatable. A “*heterogeneous datasource system*” (HDSS) is a MDSS whose component sources don’t conform to a single structural model.

System	Classification	Institution	Reference	Common Data Model	Updates	Target Data Sources
Multibase	Tightly-Coupled FDBS	Computer Corp. of America	[DH84]	Functional + general-ization	Yes	Structured databases
MRDSM	Loosely-Coupled FDBS	INRIA, France	[LMR90]	Relational	Yes	Relational databases
TSIMMIS	Mediator System	Stanford	[PMU96]	Object Exchange Model (OEM)	No	Semi-structured or unstructured, dynamic
HERMES	Knowledge Amalgamation Mediator System	Univ. of Maryland	[AE95]	Two-Stage Entity Relationship	No	Knowledge bases, spatial and temporal reasoning systems
Information Manifold	Information Gathering Agent	AT&T Bell Labs	[LRO96]	Relational + class hierarchies	No	Web-based data sources
Metadatabase	Metadata Repository System	Rensselaer Polytechnic	[CC96]	None (but metadata expressed in	No	Enterprise databases, CIM

				ER model)		
--	--	--	--	-----------	--	--

Figure 1: Systems Overview

This paper focuses on four classes of HDSS: federated database systems, mediator systems, decision-logic based systems, and metadata repository systems. A brief overview of each class is defined below.

1. Federated Database Systems

A “*federated database system*” (FDBS) is an integrated collection of full-featured autonomous databases, in which the component administrators maintain control over their local systems, but cooperate with the federation by supporting global operations [LS90]. There are two kinds of FDBS: tightly coupled and loosely coupled.

In a “*tightly coupled FDBS*”, such as Multibase [DH84] or Pegasus [Ahm+91], the federation DBA provides the end-user with a predefined static view upon which to query and perform update operations. The federation DBA performs all schematic and semantic integration in advance. A tightly coupled FDBS may have either a *single federation* (one global schema), or *multiple federations* (schemas tailored to specific user groups).

A “*loosely coupled FDBS*”, such as MRDSM [L+89], is a dynamic integration system in which the end-users interact with the component databases directly by means of a special data manipulation language. The end-users look at component schemas and formulate their own federated schema.

Note that there is no consensus of terminology in the federated database community. For example, the term “*multidatabase*” is used by [LMR90] to mean a loosely coupled FDBS, whereas [SL90] uses it to mean any collection of integrated databases.

2. Mediator Systems

A “*mediator system*” is a collection of information sources that are integrated to provide a uniform read-only interface to the end-user, and a set of tools for performing the integration tasks. There are three classes of users: the *domain integrator*, who translates an information source into the common data model, the *mediator author*, who integrates the translated sources, and the *end-user*, who submits queries to the integrated sources [PMU96]. The key difference between a mediator system and an FDBS is that the user cannot perform updates to the component sources. A mediator system is similar to a tightly coupled FDBS in that the end-user works with predefined static views. It is also like a loosely coupled FDBS, in that the component sources are highly autonomous.

Two mediator systems are discussed in section III. In TSIMMIS [PMU96, PMW95], the focus is integrating sources that are unstructured or semi-structured, i.e. that do not have a well-defined schema. In HERMES [AE95], the focus is integrating knowledge bases and reasoning systems.

3. Description-Logic Based Systems

The “*Information Manifold*” is a *description-logic-based* approach, which uses AI planning techniques to solve queries over multiple web-based information sources [LRO96B]. This approach is similar to the mediator systems, except that it imposes a single global view, whereas the mediator systems provide multiple views that can be layered and tailored to specific user groups.

4. Metadata Repository Systems

Another approach is the *metadata repository*, as demonstrated by the *Metadatabase* system [CC96]. In this system, queries are formulated dynamically by interacting with an on-line global dictionary of metadata. An intelligent user interface helps the user articulate queries by traversing the local models of the component databases, and assists in resolving semantic and schematic heterogeneities. This system is similar to a loosely coupled FDBS, in that there is no predefined integrated view. It is also like a mediator

system in that the user is limited to read-only queries. However, unlike all the other systems, there is no canonical data model.

In section III, I will discuss the data models, integration techniques, and query handling approaches of each of these systems in more detail.

B. Integration Architectures

The approaches described above can be broadly classified into two groups: *static integration systems* and *dynamic integration systems*. In static integration systems, schematic and semantic heterogeneities are resolved when a new component database is incorporated into the system. In dynamic integration systems, the heterogeneities are resolved by the end-user at query time. A general reference architecture for each class is described below.

System	Dynamic or Static	Global View	Metadata Resource	View Specification Language	Semantic Integration Techniques	Integration Tools
Multibase	Static	Superview (Global Schema)	Auxiliary Schema	DAPLEX and NQUEL.	Generalization	Language extensions
MRDSM	Dynamic	None	None	MSQL	implicit joins, dynamic attributes	Language extensions
TSIMMIS	Static	None	None	Mediator Specification Language (MSL)	Rule Specification, Virtual Objects	MedMaker
HERMES	Static	None	Yellow Pages	Generalized Annotated Program Framework	Amalgamation using Annotated Logic	Mediatory Programming Environment
Information Manifold	Static	World View	Source Descriptions	CARIN-CLASSIC	Correspondence functions.	Future work
Metadatabase	Dynamic	None	Global Information Resource Dictionary	MQL	rule processor, model traversal	Model Traversal Interface

Figure 2: Integration Issues

1. Static Integration Systems

In static integration systems, shown in Figure 3, there are three kinds of users: *domain integrators*, *mediator authors*, and *end-users*. As new component databases are integrated into the system, heterogeneities are resolved by the domain integrators and

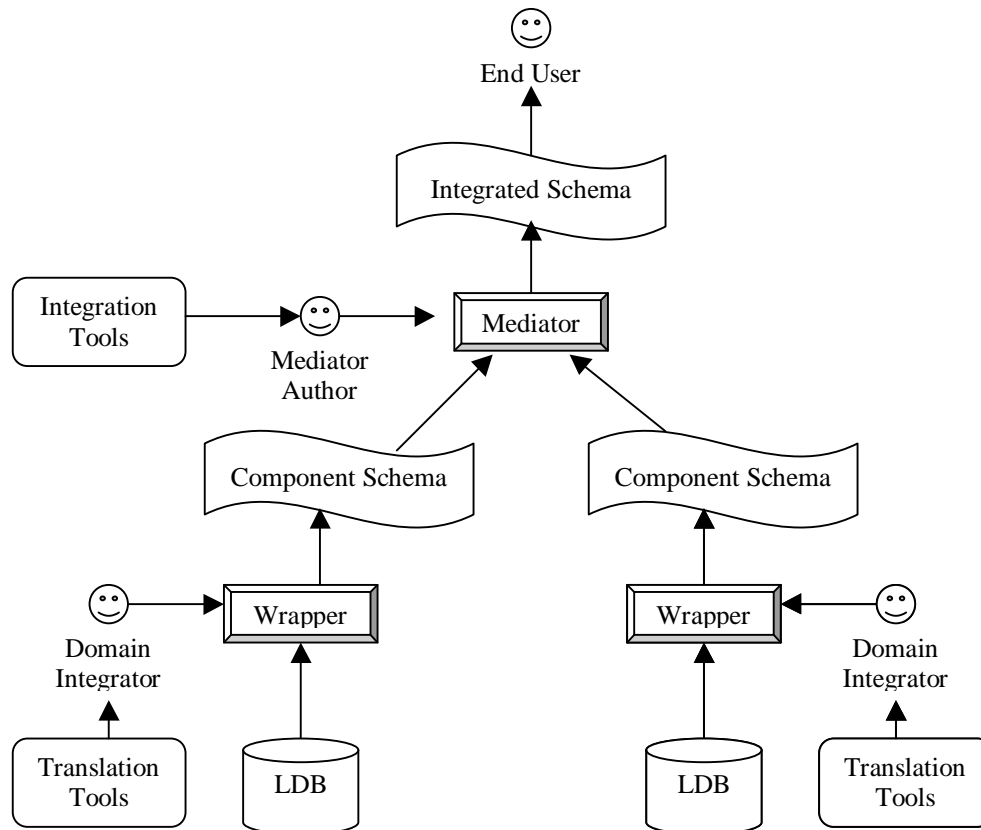


Figure 3: Static Integration Architecture

mediator authors during a two-stage “*schema integration process*”, resulting in an integrated schema, which hides the semantic and structural differences of the underlying component databases from the end-users.

In the first stage, the domain integrators utilize system-supplied *translation tools* to implement a *wrapper* that translates the domain schema from their local data model into the system-wide canonical data model. In federated database terminology, the wrapper is

called a “*transforming processor*” [SL90]. The resulting schemas are called “*component schemas*”. By first translating into an intermediary component schema, the overall process of integration is greatly simplified. In federated database systems, the task of domain integration is often performed by the database administrators of the local databases, with the help of the federated database administrator.

In the second stage, the mediator authors use system-supplied *integration tools* to implement a *mediator* that combines multiple component schemas into an *integrated schema*, in which inconsistencies are resolved and duplicates are removed. In federated database terminology, the integrated schema is referred to as a “*federated schema*” or “*export schema*”. In some systems, all component schema are integrated into a single global schema, while in others, multiple integrated schema exist for different classes of end-users. Mediators may be combined in layers to provide cleaner views [PMU96].

The process of translating and integrating schemas generates mappings that will later be used by the query processor to modify global queries.

The tightly coupled federated database systems, mediator systems, and decision-logic based systems described in this paper are all static integration systems. They differ in their approaches to schema translation and mediator authoring. The design of semi-automatic translation and integration tools is currently an active research area [KWD97].

2. Dynamic Integration Systems

In dynamic integration systems, shown in Figure 4, the end-users are not provided with a predefined view. Instead, they are given direct access to the component schemas at query time by means of a multidatabase query language or a graphical user interface. A *metadata resource dictionary* [CC96] contains descriptions of the component schema (including conversion information). Using the metadata, the end-user constructs an integrated schema on the fly, and poses queries directly against the custom-built view.

Loosely coupled FDBMs and the Metadatabase are examples of dynamic integration systems.

Because integrated schemas can be promptly created and dropped on the fly, dynamic integration systems allow the end-user more flexibility and are much easier to maintain in the presence of evolving component schema. However, they require a more sophisticated

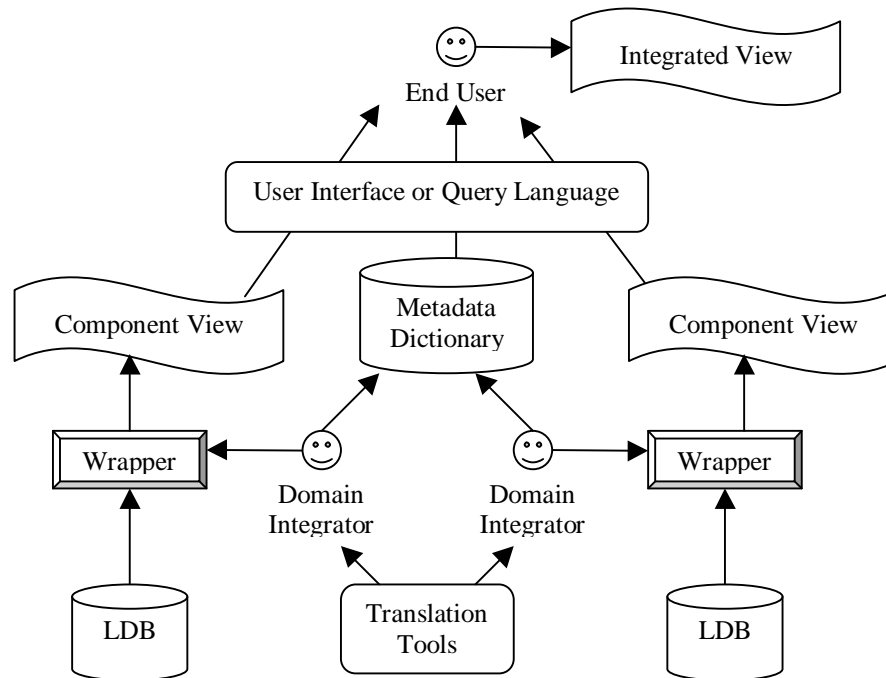


Figure 4: Dynamic Integration Architecture

end-user who understands the semantics of the component schemas and is responsible for integration.

By contrast, static integration systems tend to evolve in a gradual, controlled fashion, as they require negotiation between the three classes of users. However, the development of new integration techniques is allowing static systems to become more dynamic in nature, and the distinction between the two classes is beginning to blur. Some of these techniques are described in section III below.

C. Query Handling

Every system described in this paper follows the same basic 5-step approach to query handling: query specification, query modification, query translation, result translation, and result integration.

In the first step, *query specification*, the user articulates a request for data by specifying constraints on attributes. In all *static* integration systems, such as tightly coupled federated databases and mediator systems, end-users pose queries against a predefined integrated view using a declarative query language, or by entering constraints in a graphical interface. For example, in Multibase, queries are specified using an SQL-like language called NQUEL against a global *Superview*. In TSIMMIS, queries are formulated using an Object Exchange query language against an integrated view supplied by a mediator, tailored for that particular user. In HERMES, users pose queries as logical predicates. In Information Manifold, users select relations and supply constraints in the *World View* using a web-based interface, which generates a query expression consisting of a conjunction of clauses in a description logic language.

In *dynamic* integration systems, such as loosely coupled federated databases or metadatabases, end-users first generate a view and then pose their queries as in static integration systems. In MRDSM, queries are articulated in MSQL, a multidatabase query language.

In the second step, *query modification*, a query is decomposed into sub-queries, one for each component database to be accessed by the query. In federated databases and mediator systems, the query processor obtains a partition based on the integration functions that were used to resolve the heterogeneities on each attribute. Then for each subset in the partition, a local query is generated by replacing each global attribute with the corresponding component schema attribute. By contrast, decision-logic based systems such as Information Manifold perform query decomposition using a query planning algorithm. And in metadata repository systems, queries are decomposed

explicitly by the user via a model-assisted approach. Specific examples are given for the various systems in section III.

In the third step, *query translation*, the component sub-queries are translated into the syntax required by the corresponding local data source, whether it be query language commands, input to a web-based form, or invocation of external applications. The component information system reads the query and produces a result set. A large body of research is devoted to translating queries between data models.

System	Query Specification	Query Decomposition	Result Integration Techniques	Optimization Heuristics
Multibase	NQUEL	Partitioning on conditional functions, subrange tables	simple union, duplicate elimination	not covered
TSIMMIS	OEM-QL	View Expansion, push selections down	logical datamerge rules	partial object fetches
HERMES	Mediatory clauses, special predicates	expert system invokes external programs	Information Pooling toolkits	Unspecified
Information Manifold	Conjunctive Queries in description logic.	query plans	simple union.	prune irrelevant sources
Metadatabase	MQL, visual query formulation, automatic completion	model-assisted decomposition	equijoin	minimize number of equijoins

Figure 5: Query Issues

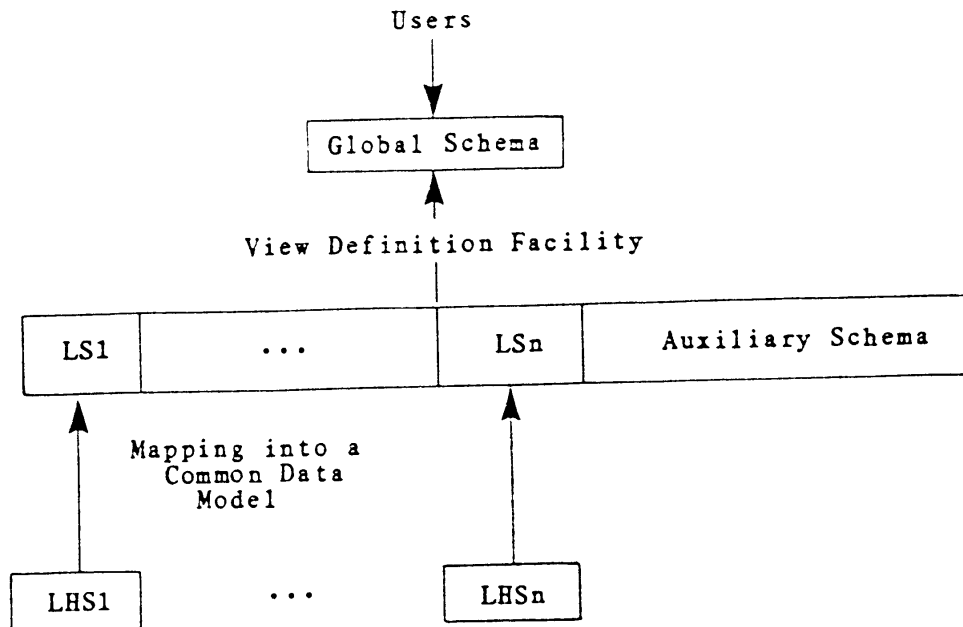
In the fourth step, each result is translated back into the common data model as necessary. In the fifth step, the multiple result sets are integrated and returned to the user. Each system has its own method for merging results, whether it is based on logical datamerge rules, information-pooling toolkits, or a simple equijoin.

III. System Details

In this section, I describe in five of the systems that were outlined above, focusing on the key aspects of each system's integration techniques and query handling strategies.

A. Multibase

Multibase is the classic tightly coupled federated database system. It is based on the functional data model, augmented with the concept of “*generalization*”, described below. The user is presented with a single global schema, called the *superview*, which provides the illusion of a homogenous database. View definition is performed by the federated database administrator, using the *DAPLEX* language, which is a functional query languages with loop constructs and nesting. The component DBAs translate their *local host schemas* into *local schemas* (i.e. component schemas) in the functional model,



recording metadata information in a global *Auxiliary Schema* (AS). The federated

database administrator then uses generalization to define the Global Schema (i.e. integrated schema) as a view of the LS's and AS.

Generalization is an integration technique that adds **ISA** inheritance relationships for both entities and functions. For example, if one database defines an employee as **EMP(name, sal, age)** and a second database defines it as **EMP(name, sal, address)**, generalization

allows us to define a *supertype* **EMP(name, sal)** that includes the common attributes, and subsumes two *subtypes* **EMP1(name, age)** and **EMP2(name, address)** which contain only the key value and specialized attributes. This is in contrast to the “*outer join*” integration technique, which would create **EMP(name, sal, age, address)** with NULL values padding the non-applicable fields.

The semantics for determining attribute values are specified as functions in the supertype. For example, suppose the two employee databases represent separate jobs, and an employee appearing in both should have a salary that is the sum of his two salaries. The following supertype definition, specified in NQUEL, does the trick:

```

DEFINE SUPERTYPE EMP BY
    EMP1 ISAe EMP, EMP2 ISAe EMP
    ID : Name
    FOR e IN EMP
        Sal := CASE
            e ISIN EMP1 – EMP2 => Sal1(e)
            e ISIN EMP2 – EMP1 => Sal2(e)
            e ISIN EMP1 Intersect EMP2 => Sal1(e) + Sal2(e)
        ENDCASE
    ENDFOR

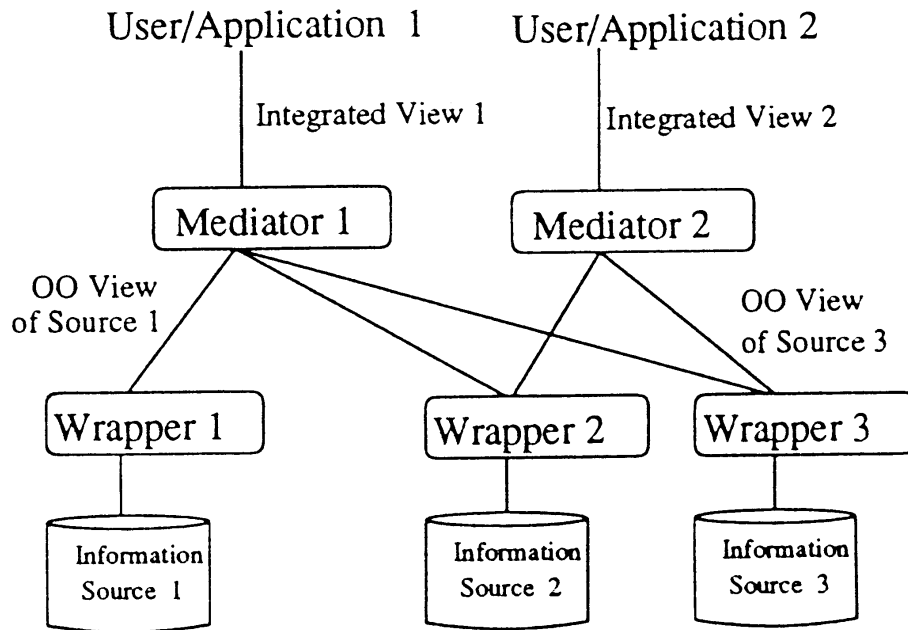
```

B. TSIMMIS

TSIMMIS is a mediator system being developed by the Stanford database group, in conjunction with IBM. A major focus of the project is the development of tools to speed up the integration process by extracting properties from unstructured or semi-structured sources with no well-defined static schema, such as email documents or a finger facility. The key to their approach is the *Object Exchange Model (OEM)*, in which data items are *self-describing*. As opposed to relying on a schema, the object structure is included as labels embedded in every data item. Each OEM object has the structure **<Object-ID, Label, Type, Value>**. This is analogous to the *REPO* model of my Web-Interfacing Repository Manager [JB97], in which data objects are accessed as perl associative arrays. Like REPO, OEM differs from the Object-Oriented model in that it is much simpler: supporting only nesting and object-identity.

For example, here is a set of results from the finger operation (note the variable structure of the person elements):

```
<&g1, finger, set, {&g2, &g3, ...}>
  <&g2, person, set, {&g21, &g22}>
    <&g21, name, string, 'Linda Shapiro'>
    <&g22, login, string, 'shapiro'>
  <&g3, person, set, {&g31, &g32, &g33}>
    <&g31, name, string, 'Rex Jakobovits'>
    <&g32, plan, string, 'To pass generals!'>
    <&g33, login, string, 'rex'>
```



Because OEM forces no regularity on the data, it can easily handle structure irregularities and can handle schema evolution without the need for rewriting Mediators. The schema-less nature of OEM is useful when clients do not know in advance the structure of OEM objects, and can discover the structure as queries are posed. Users pose queries via OEM-QL, an SQL-like query language which supports wild-card pattern matching. Wild

cards can bind to both labels and attribute values. For example, to discover the structure of Rex’s finger entry:

```
SELECT *.* from FINGER
WHERE login = “rex”
```

In addition to the OEM, a major contribution of TSIMMIS is the *Mediator Specification Language* (MSL), a high-level declarative language for integrating data sources. MSL uses prolog-like rules and functions for translating objects. The tail of a rule specifies patterns found in the sources, while the head describes patterns of the top-level objects of integrated views.

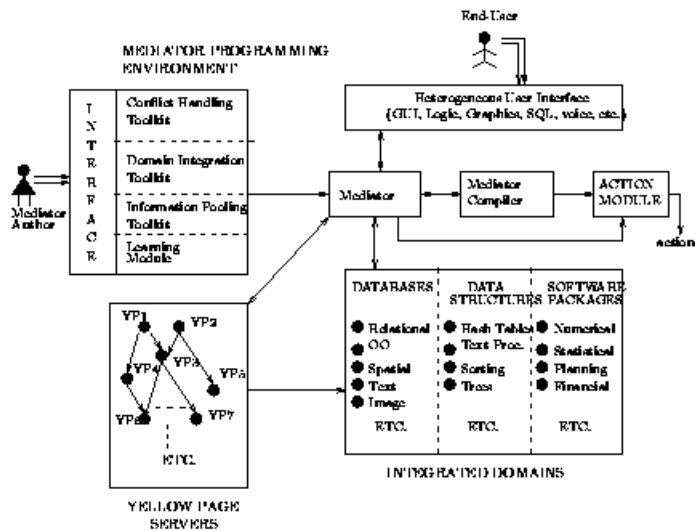


Figure 8: HERMES

C. HERMES

HERMES is a Mediator system being developed at the University of Maryland. Unlike TSIMMIS, which integrates semi-structured data sources, HERMES focuses on integrating knowledge bases and reasoning systems. Specifically, it provides support for amalgamating knowledge from relational, object-oriented, spatial, and temporal domains into a single reasoning system. Integration is achieved by hooking up each component system to the semantic model, called the Generalized Annotated Program (GAP)

Framework, an extension of logic programming. GAP supports domain-specific semantic expressions, such as:

SPATIAL:RANGE('brainmap', P, 5.0)

which finds all points within 5 units of point **P** in the spatial database called **brainmap**.

Like TSIMMIS, HERMES defines a rule-based mediator language, implemented within a *Mediator Programming Environment*. The Mediators are written as prolog-like clauses, with two special predicates:

- **in()**: executes a select statement on the target data source.
- **=()**: tries to unify two values.

The model also supports uncertainty, and atoms are marked with confidence factor values. Related systems are Carnot [CHS91] which uses the Cyc knowledge base, and SIMS, based on the Loom knowledge representation language.

D. Information Manifold

The Information Manifold (IM) is a decision-logic system for integrating web-based information sources, being implemented at AT&T Bell Labs by Alon Levy. As in a Mediator system, the end-user specifies their queries declaratively against a static view. Unlike TSIMMIS, however, the IM presents the user with a single global view, called the *World View*, which is a collection of virtual relations and classes that describes the contents of the information sources. The IM uses a relational data model, augmented with class hierarchies. A key feature of the IM model is that classes can be declared to be disjoint, guaranteeing that no object can belong to both.

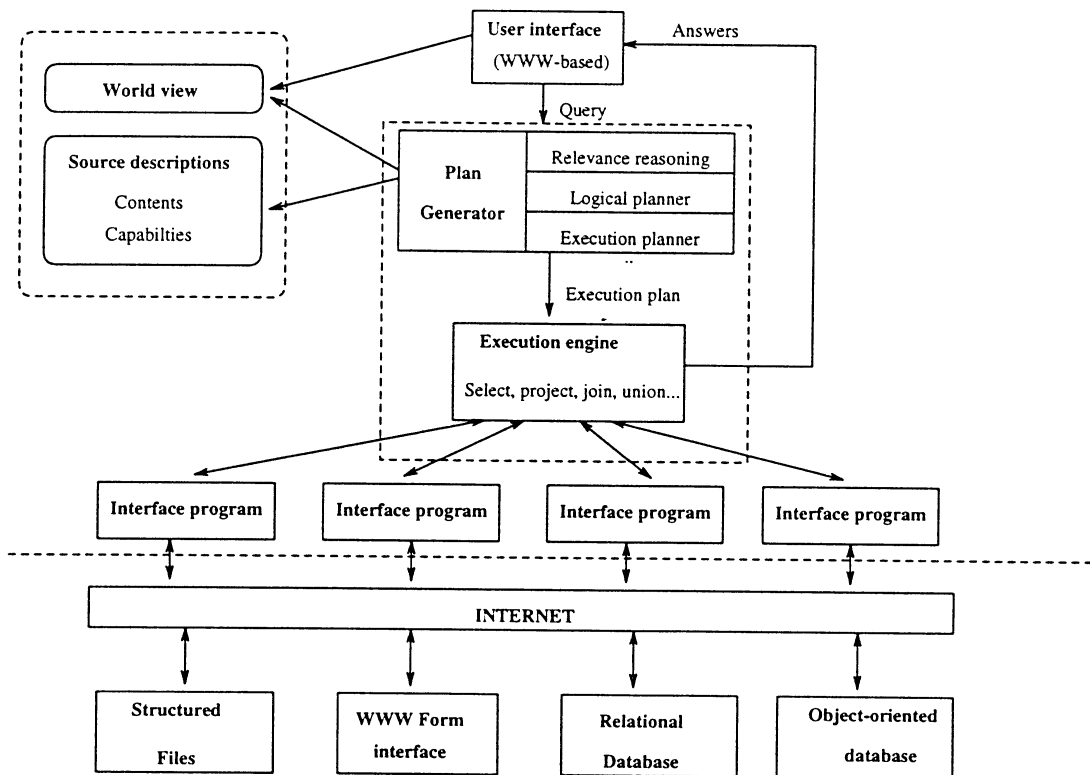
Using the CARIN-CLASSIC description logic language, the integrator specifies *source descriptions* for each data source, consisting of *content records* and *capability records*. The content record of a given source identifies the world view attributes which can be

found in that source. For example, consider the following content record for a source which describes a database of car reviews for models newer than 1985:

V(product, year, review):

Review(model, year, review) \wedge Car(model) \wedge year \geq 1985

Because web-based sources often have limited query capabilities, a capability record describes which attributes can be used as binding and selection criteria. A capability record is a tuple $\langle S_{in}, S_{out}, S_{sel}, \min, \max \rangle$ which specifies that the elements of S_{in} can be used as input parameters, bindings must be supplied for at least **min** of the input parameters, the elements in S_{out} are the possible output parameters, and the elements of



S_{sel} are parameters on which the source can apply inequality selections. For example, if

the car review query form allows reviews to be retrieved by model or year, the capability record might look like this:

({model, year}, {model, year, review}, {year}, 1, 2)

User queries are formulated in terms of the world-view relations by means of *conjunctive queries*. For example, the following query requests 1992 or later model cars for sale and their reviews:

Q(model, yr, price, rev) : CarForSale(model, yr, price) \wedge Review(model, yr, price) \wedge yr > 1992

The IM uses *query plans*, consisting of conjunctive subgoals, to prune out irrelevant sources and decide on the order of query execution.

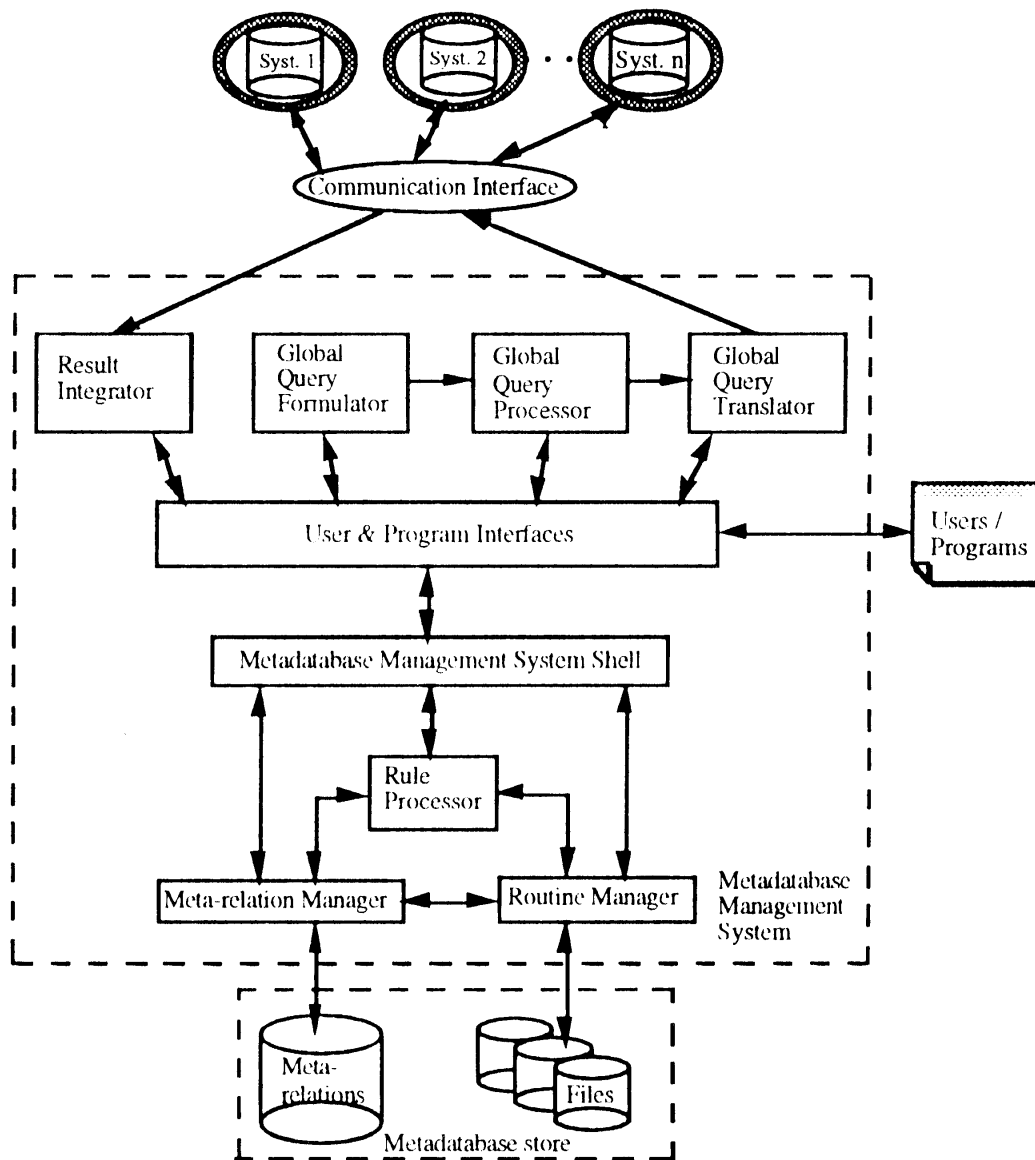


Figure 10: Metadatabase

E. Metadatabase

The Metadatabase is a dynamic integration system developed at the Rensselaer Polytechnic Institute. Unlike the majority of systems covered here, the Metadatabase does not provide the user with a unified schema or data model, but rather provides an on-line repository of enterprise metadata to facilitate global query formulation. The user

interface allows the user to articulate queries directly by visually traversing the local data models. The user picks items and attributes directly from menus rather than entering their names. The system uses a detailed metadata repository to provide assistance in traversing models, checking semantics, and deriving implied data. The user's dialog with the interface results in the generation of a query in the *Metadatabase Query Language (MQL)*. The Metadatabase includes a table of conversion rules for use during query processing. A *Rules Processor* searches for applicable rules and fires them.

The metadata about local models are represented using the *Global Information Resource Dictionary (GIRD)* model. GIRD supports four categories of metadata: functional models, structural models, software and hardware resources, and application families. GIRD itself is implemented using a *Two-Stage Entity Relationship (TSER)* method, in which metadata are defined as a network of SUBJECTs (functional objects), ENTITIES (structural objects), and CONTEXTs (rule-based descriptions of processes).

IV. Alternative Approaches

The four classes of systems described above define the cutting edge of integration research, and all of them seek to provide a framework for schematic integration while maintaining the autonomy of the component systems. But a number of other approaches are being used in industry, and they deserve consideration here.

A. Point-to-Point Gateways

Rather than attempt to integrate multiple systems under a single query interface, pairs of systems can be integrated by building customized interfaces between them. The adoption of an intermediary call-level-interface standard, such as ODBC [Gei96] or JDBC [PP96], provides a uniform way to communicate, although schematic conflicts must still be handled on a case-by-case basis.

B. Data Warehouses

A data warehouse is a centralized repository of information extracted from multiple data sources [Wid95]. The difference between a data warehouse and the integration systems

described in this paper is that a data warehouse is a static copy of the integrated data, and updates to the component databases are not readily reflected in the warehouse. The warehouse is a kind of *instantiated view*. The view tends to be subject-oriented, containing histories of the data changing over time, and focuses on categorizing the data across various dimensions. Data is integrated at the time of gathering or by “data cleaning” techniques. The disadvantage of this centralized approach is that it is hard to keep the warehouse in synch with local systems from which the data was copied without severely restricting their local autonomy.

But in spite of their static nature, data warehouses have proven to be extremely valuable to corporate enterprises. They provides users with the “bigger picture”, and facilitate knowledge discovery and on-line analytical processing. And being an industry solution, they come with industrial strength tools: high-level applications for developing the warehouse, cleaning the data, deploying web interfaces, and facilitating data mining. And most importantly, because query execution does not involve data translation and communication with remote sources, complex queries can be executed easily and efficiently.

C. Distributed Objects

The opposite approach to integration is the adoption of standards. The Human Brain Project could encourage its constituents to adopt a distributed object model for their applications and databases. The distributed object model allows objects to interact without knowing anything about their location [OHE95]. Objects are encapsulated entities that are accessed by means of well-defined, self-describing interfaces. Local DBA’s could still define their own internal data models, data structures, schema, implementation languages, system platforms, etc. But to participate in the “federation” their systems would need to comply with an *object request broker* standard, such as CORBA [Vin97].

D. Shared Class Libraries

This is the most restrictive approach of all with regard to component autonomy. The community agrees on a common application language (e.g. Java or C++) and develops a

set of domain-specific superclasses from which applications may inherit their data structures and interfaces. Using an object-oriented database (i.e. a persistent programming language), data objects naturally gain persistence.

A classic example of this approach is the *Image Understanding Environment* (IUE), an ARPA project to develop a common object-oriented software environment for facilitating the exchange of research results within the Image Understanding community [IUE-url].

The approach has the obvious benefits of a common data model, and independent projects can avoid “reinventing the wheel”. It greatly facilitates the design applications that use multiple sources, and programmers can benefit from other people’s tools at the application level. For example, a system for handling spatial queries over atlases can be made into a “spatial_query_processor” class, which operates on “spatial_query_objects” and “spatial_data_objects” which are specified as classes that multiple groups may use.

E. Shared Ontologies

Perhaps the greatest promise of all lies with share common knowledge ontologies, which are essential for integration at any level. Every approach described in this paper would benefit from a detailed knowledge ontology. An ontology is more than just a controlled vocabulary: it should contain deep domain knowledge and form a *conceptual* standard. Our Structural Informatics Group has proposed that anatomy is the ultimate framework for organizing biological knowledge [Ros+97].

V. Conclusion

The question remains, which approach is best for the Human Brain Project? Because the various approaches differ greatly in their models presented to the user, we must first identify the **target users** of the HBP databases. Then we can identify the requirements of each user group, and adopt an approach (or combination of approaches) accordingly.

An immediate user class is the *local database developer* at each HBP site: when designing their local applications, they could benefit from a global model. However, they probably value their autonomy more than the convenience of a global model. Another user will be the *federation DBA*: the person responsible for managing the “mother of all databases”, which could require a deep knowledge of the local schemas.

A broader class of users are the *HBP researchers* themselves, who are not necessarily working on the database aspects of their systems, but would have a high interest in learning more about the other projects’ data.

An important class of users are the “*funders*”: members of Congress and grant issuers, who will use the federated database as a means of evaluating the success of the entire brain project effort.

But in the long term, the ultimate users are those not directly affiliated with the HBP: hard core neuroscientists, students, and clinicians.

It is clear that the wide range of users will exhibit different levels of sophistication with regard to domain knowledge, internal data models, and programming sophistication. Therefore a combination of the approaches must be undertaken simultaneously.

In the face of the complexity of the data, integration is a daunting task. But we don’t have to start from scratch: a number of sub-communities already exist, sharing schemas and data at point-to-point level, and groups have already begun adopting their own protocols and standards. The route to success must start from two directions: *top-down*, in which a data warehouse or global view is gradually constructed, and *bottom-up*, in which closely related research groups gradually increase the amount of sharing and cross-fertilization. At every step of the way, the design and operational autonomy of the independent groups must be preserved.

VI. References

- [AE95] S. Adali and R. Emery. A uniform framework for integrating knowledge in heterogeneous knowledge systems. *Proc. of the Eleventh International Conference on Data Engineering*, pp. 513-520, 1995.
- [Ahm+91] R. Ahmed et. al. The Pegasus heterogeneous multidatabase system. *IEEE Computer* 24(12):1991, 19-27.
- [CC96] W. Cheung and H. Cheng. The Model-Assisted Global query system for multiple databases in distributed enterprises. *ACM Transactions on Information Systems*. 14(4): 421-470, 1996.
- [CCT94] Wesley Chu, A. Cardenas, and R. Taira. KMeD: A knowledge-based multimedia medical distributed database system. *Information Systems*, 20(2): 75-96, 1994.
- [CHS91] C. Collet, M. Huhns, and W. Shen. Resource integration using a large knowledge base in Carnot. *IEEE Computer* 24(12):55-63, Dec. 1991.
- [CNN91] Committee on a National Neural Circuitry Database. Mapping the brain and its functions: integrating enabling technologies into neuroscience research. National Academy Press, Washington D.C., 1991.
- [DH84] U. Dayal and H. Hwang. View definition and generalization for database integration in MULTIBASE: A system for heterogeneous distributed databases. *IEEE Trans. Software Engineering*, SE-10, 6, 628-644, 1984.
- [Gei96] Geiger, Kyle. Inside ODBC. Microsoft Press, 1996.
- [GMS94] C. Goh, S. Madnick and M. Siegel. Context interchange: overcoming the challenges of large-scale interoperable database systems in a dynamic environment. *Proceedings of the Third International Conference on Information and Knowledge Management*, Dec. 1994.
- [IUE-url] <http://www.aai.com/AAI/IUE/IUE.html>
- [JB97] R. Jakobovits and J.F. Brinkley. Managing medical research data with a web-interfacing repository manager. Submitted to *1997 JAMIA Fall Symposium*.
- [KH97] S. Koslow and M. Huerta. Neuroinformatics: an overview of the human brain project. Lawrence Erlbaum, Mahwah, NJ, 1997.

- [Kim95] W. Kim. Modern database systems: the object model, interoperability, and beyond. ACM Press, New York, 1995.
- [KS91] W. Kim and J. Seo. Classifying schematic and data heterogeneity in multidatabase systems. *IEEE Computer* 24(12) 12-18, Dec. 1991.
- [KWD97] N. Kushmerick, D. Weld and R. Doorenbos. Wrapper induction for information extraction, *IJCAI-97*, 1997.
- [L+89] W. Litwin, et. al. MSQL: a multidatabase language. *Information Science*, 49, 59-101, 1989.
- [LMR90] W. Litwin, L. Mark, N. Roussopoulos. Interoperability of multiple autonomous databases. *ACM Computing Surveys*, 22, 3, 267-293, 1990.
- [LRO96] A. Levy, A. Rajaraman, and J. Ordille. Querying heterogeneous information sources using source descriptions. *Proceedings of the 22nd VLDB Conference*, 1996.
- [LRO96b] A. Levy, A. Rajaraman, and J. Ordille. Query answering algorithms for information agents. *Proceedings of the 13th National Conference on Artificial Intelligence*, August 1996.
- [OHE95] B. Orfali, D. Harkey and J. Edwards, Intergalactic client/server computing. *BYTE*, April 1995.
- [PMU96] Y. Papakonstatniou, H. Garcia-Molina, and J. Ullman. MedMaker: A mediation system based on declarative specifications. *IEEE 12th Int. Conference on Data Engineering*, pp. 132-141, 1996.
- [PMW95] Y. Papakonstatniou, H. Garcia-Molina, and J. Widom. Object exchange across heterogeneous information sources. *11th International Conference on Data Engineering*, pp. 251-260, 1995.
- [PP96] P. Patel. Java database programming with JDBC. Coriolis Group, 1996.
- [Ros+97] C. Rosse, R. Jakobovits, B. Modayur, and J. Brinkley. Motivation and organizational principles for the Digital Anatomist Symbolic Knowledge Base: an approach towards standards in anatomical knowledge representation. To appear in *Journal of AMIA*.
- [SL90] A. Sheth and J. Larson. Federated database systems for managing distributed, heterogeneous and autonomous databases. *ACM Computing Surveys*, 22, 3, pp. 183-236, 1990.

- [SSR94] E. Sciore, M. Siegel, and A. Rosenthal. Using semantic values to facilitate interoperability among heterogeneous information systems. *ACM Transactions on Database Systems* 19(2): 254-290, June 1994.
- [SSU96] A. Silberschatz, M. Stonebraker, J. Ullman. Database research: achievements and opportunities into the 21st century. *SIGMOD Record*, 25(1):52-63, 1996.
- [Sub94] V. Subrahmanian. Amalgamating Knowledge Bases. *ACM Transactions on Database Systems*, 19, 2, pp. 254-290, 1994.
- [Vin97] S. Vinoski. CORBA: integrating diverse applications within distributed heterogeneous environments. *IEEE Communications Magazine*, 35(2): 46-55. Feb. 1997.
- [Wid95] J. Widom, "Research Problems in Data Warehousing." *Proceedings of the 4th Int'l Conference on Information and Knowledge Management (CIKM)*, November 1995.
- [Wie93] G. Wiederhold. Intelligent integration of information. *Proceedings of the ACM SIGMOD Conference* pp. 434-437, May 1993.