1. INTRODUCTION

Databases have become an integral part of information systems. In the past two-decades, different database systems have been developed and maintained independently for different types of information systems. Databases are utilized by different groups of users and organizations for their daily functions. Each of these information systems are developed independently and customized to meet the particular requirements of the organization. However, today's interconnected networks have provided access and a need for integrated access to these independently developed information systems as a basic necessity for developing the next generation global information systems. Sophisticated user requirements (such as data warehousing, knowledge discovery and data mining, intelligent access to information on the Web) and the availability of a proliferation of data sources containing related information has created intelligent data access from heterogeneous distributed data sources a critically importance research issue. The focus of research into heterogeneous (multi-) database systems has been to provide such an integrated transparent access to a multitude of heterogeneous distributed data sources in a meaningful way. In order to better illustrate the problem domain, let us consider the following simplified example.

Example 1: Let us consider two database schemas of a university:

Schema 1: A relational database containing information about students and faculty of the university.

STUDENT

student-id	last_name	first-name	address

COURSE_ENROLLMENT

student-id	course-id	section-id	semester	year	grade

COURSE_OFFERING

course-id	section-id	semester	year	instructor-id

COURSE

course-id	course-name	description

INSTRUCTOR

id	last_name	first-name	address	dept-id

Figure 1. Schema of a Relational Database Developed for a University Application

Schema 2: A Semantic Database Schema [94] in the Computer Science department of the university consisting of information about students and the projects they are currently working at the department.

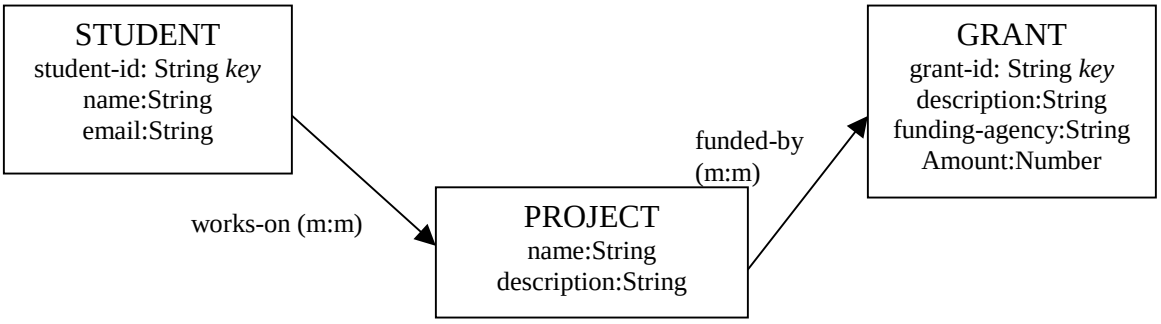


Figure 2. Semantic Database Schema in Computer Science Department of University Consisting Information of Students and Projects

Let us consider the following query:

Query 1: List students who are working in project ‘A’ of ‘Computer Science’ department and the courses they have taken at the university.

Query 1 is a simple query requiring information from two different databases. The results for query 1 require accessing *schema 2* to obtain information on students working for

project ‘A’ and also requires accessing *schema 1* to obtain course information of the students.

A straight-forward approach for obtaining the results for the above query is to write an application program that firstly accesses *Schema 2* to obtain the students working on project A. This requires the programmer to write a query in a native query language that the database engine supports, (in this case Semantic SQL [98] for Semantic Databases). Next, the program accesses *Schema 1* to obtain information on students and their courses. Similar to the previous database access, the programmer needs to access in a different query language pertinent to the database, (in this instance, SQL supported by the relational database engine). Finally, the program integrates the results of the two previous queries to obtain the result for query 1. The developer of the program needs to consider the distribution of data sources and program in different data communication protocols to obtain the results of the queries due to network and platform heterogeneity.

It is evident from the above description that the effort required to access distributed, heterogeneous data sources is significant and complex. It is not viable to expect an application developer to develop such applications accessing a large number of data sources for every required data access. Also, in today’s dynamic environments such as the World Wide Web where information sources change frequently, keeping these applications up-to-date is an enormous task and practically impossible. Some of the disadvantages of this approach are enumerated below:

(i.) Expensive: Need to hire an Application Developer to implement the query to access heterogeneous data sources.

(ii.) Non-generalized solution: That is, a different query requires the development of a new application program.

(iii.) Heterogeneous data models and query interfaces: The application developer must be familiar with the different data models and query facilities of the data sources.

(iv.) Distributed environment and different communication protocols: The application developer needs to consider the data distribution and communication protocols when accessing the data sources.

(v.) Considerable effort: Integration the results of different queries require substantial effort by the programmer.

(vi.) Not scalable: Integrating a new data source requires re-writing the application program.

(vii.) Non-optimal solution: Usually the application developer is not well versed with the query optimization techniques of distributed databases. Thus, (s)he, most probably, will develop the application using a non-optimal query processing strategy that may result in considerable degrading of system performance.

(viii.) Semantic and Schematic Heterogeneity Resolution: The application programmer requires to familiarize himself/herself with the data contained in all of component databases in order to decide the databases that are required to be accessed. In addition, the application developer needs to resolve schema-level heterogeneities that may occur due to different representations of similar data in different component databases.

(ix.) Data consistency problems: Changes of the data sources are not easily incorporated.

The solution proposed by the heterogeneous database researchers is to develop a homogenizing layer over the heterogeneous, distributed databases providing a single data model and query language. This allows the user to pose queries directly to the heterogeneous database system, which provides the illusion of a centralized homogeneous database system. This methodology avoids the disadvantages described in the previous approach.

1.1 Related Work

Three major approaches for building heterogeneous/multi-database systems can be identified in literature:

(i.) Global schema approach: Schemas of component database are exported to a global site and schema integration phase considers the creation of a global schema. In this approach, a global integrated schema, capturing the information content of the component databases in a single data model and query language, is presented to the user. Global schema approach has been discussed in [1], [9], [31], [33], [41], [56] and others.

(ii.) Federated database approach: This approach imports sharable schemas of remote databases and integrates with the local schema. Discussion on federated database approach can be found in [76], [105] and others.

(iii.) Multi-database query language approach: A powerful multidatabase language is provided to the users to manipulate data and meta-data in a multitude of non-integrated schemas. This approach is discussed in [62], [71], [72] and others.

We now investigate some of the strengths and weaknesses of each approach.

(i.) Global Schema Approach: Global schema approach provides users with a single global schema identical to the interface of a centralized database system, which provides a single schema (in a homogeneous data model and query language), containing the information content of all the integrated data sources. This is an ideal solution since this provides a well-known paradigm for heterogeneous distributed data access. However, the critics of this approach have pointed out the difficulty and inability to obtain such global schemas when a large number of data sources are integrated (i.e. solution is not scalable) and practically impossible (i.e. solution is not feasibile) to obtain the knowledge needed for the creation of such a global schema. Also, another limitation, which is not discussed in to a great extent, is the overhead of maintaining global schemas with dynamically changing component data sources.

(ii.) The federated database approach does not aim to provide a single global schema, rather integrates the local schema with other data sources containing relevant information. The creation of global schemas is avoided in this approach. However, the issue of identifying related information sources and resolving schematic heterogeneity must be addressed.

(iii.) Multidatabase language approach: In the multidatabase language approach, the system does not take part in the integration process. The major advantage is that the system is relieved from creating and maintaining global schemas. That is, the quality and completeness of content of information depends on the users' ability to specify his/her requirements adequately. The main limitation of this approach, is the assumption that the user have the expertise to express their intentions by using the complex language features provided for information sharing and exchange. Usually, a user of database system may be a naïve user with little knowledge of different data sources and technical know-how to manipulate the language.

1.2 Our Work

At High-performance Database Research Center (HPDRC [95]), we undertook the Heterogeneous Distributed Database Project, which aims at integrating information from a variety of distributed heterogeneous data sources (which includes structured data sources such as relational databases, semantic databases and semi-/un-structured data sources such as information from the World-Wide Web). This thesis describes the results of our efforts in designing the heterogeneous distributed database project at HPDRC.

A common issue to every approach discussed above is the resolution of semantic and schematic heterogeneity. That is, identifying related information from a multitude of heterogeneous data sources (i.e. resolve semantic conflicts) so as provide complete, coherent answers (i.e. resolve schema/data level conflicts) to users' requests. Our

approach provides a global knowledge base consisting of semantic knowledge to resolve semantic heterogeneity. The semantic heterogeneity resolution methodology applied is complete and unambiguous. We present a semi-automated, stepwise methodology to acquire semantic knowledge, thus reducing the effort required to gain initial knowledge for integration. The semantic heterogeneity resolution methodology presented can be easily adapted by global schema approach, federated database approach and multi-database approach, thus providing better means for intelligent data access. We applied this approach to a modified version of global-schema/federated database approach. Our approach provides a semi-automated way to create global views to the different user groups fulfilling their information requirements. Thus, we provide the ideal solution to each user group (that is, an integrated schema in a single query facility similar to a centralized database) accessing heterogeneous multiple data sources; however, we avoid creating and maintaining a single large global schema. Also, the creation and maintenance of global views is semi-automated, thus reducing the overhead. Since the approach is based on resolving semantic conflicts, it is scalable and has the ability to handle dynamic changes to a high degree. Also, most of the integration processes are automated, thus avoiding the need for great efforts and complexities inherent in the previous approaches. Further discussion on our semantic and schema level heterogeneity mechanisms are discussed in chapters 3 and 4. The next section briefly enumerates the contributions of this thesis.

1.2.1 Contributions of Thesis

The contributions of our research include the following:

- Architecture for Heterogeneous Distributed Database System: A scalable, easy-to-develop architecture using state-of-art technologies for the design of a heterogeneous database system is discussed.
- Semantic Binary Object-oriented Data Model and Semantic SQL query language: We have used an expressive data model and query language for the integration of heterogeneous data sources, namely Semantic Binary Object-oriented Data Model (Sem-ODM) [94] and Semantic SQL query language [98]. The ability to capture complex semantics and easy-to-use query facility made Sem-ODM and Semantic SQL excellent candidates for information integration and querying of heterogeneous data sources. Our approach is different from other approaches as we try to capture semantics of the data being integrated for easier integration and querying through our data model and other techniques.
- Semantic Heterogeneity Resolution Methodology: A major impediment for the ubiquitous use of multidatabase technology is the difficulty in resolving semantic heterogeneity between data sources. That is, identifying and managing semantically related information from heterogeneous distributed databases. We outline a methodology based on extents of meta-data constructs of schemas to resolve semantic heterogeneity. We outline the correctness and completeness of our methodology.
- Schematic Heterogeneity Resolution: A database system provides a schema (meta-data) describing the information content of the database. Similarly, a heterogeneous database system requires the definition of global views for the different user groups allowing access to the required information. Resolving schema-level heterogeneities (that occur due to different representation of semantically related information in

component schemas) is an issue that is addressed in the creation of global Sem-ODM schemas. A language to create global Sem-ODM schemas over a set component Sem-ODM schemas resolving schema-level heterogeneities is provided. The different schema conflicts and their resolutions using the language is illustrated.

- Heterogeneity Resolution Methodology and Knowledge Bases: As discussed earlier, our approach to resolving conflicts is unique since we take a step-wise process by firstly resolving semantic conflicts and then considering resolving schema-level conflicts. The semantic knowledge acquired during semantic heterogeneity resolution process is exploited to assist in resolving schema-level heterogeneities. The design of a knowledge base is a critical component to store and manage such semantic knowledge. Sem-ODM schemas for the storage component of the knowledge bases are presented. Rules that semi-automatically resolve conflicts and assist in the creation of global views are outlined. This simplifies the creation of global views significantly reducing the overhead in the global schema approach.
- Query processing and optimization: Query processing of Semantic SQL statements over a Sem-ODM global schema are presented along with strategies to optimize them utilizing semantic knowledge. Our optimizing strategies focus on utilizing semantic knowledge acquired during schema integration process to gain maximal system performance.
- A framework for Internet computing: Most database researchers see the Internet as a database system, which is loosely structured. However, a more natural perspective is to view the Internet as a distributed computing medium with heterogeneous data sources and services. We present a framework for Internet computing and present

how some of our concepts discussed can be applied in such a framework. This framework is presented as a future area of research with the goal being to achieve intelligent ubiquitous computing and communication on the Internet.

1.2.2 Limitations

Similar to other research projects and systems, we also have limitations in our research project. In this thesis, we have only considered read-only queries. Insert, delete and update queries including distributed transaction processing in a set of heterogeneous distributed data sources have been omitted from discussion in our thesis work.

1.2.3 Outline of Thesis

This thesis is organized as follows. Chapter 2 describes the high-level architectural design of the Heterogeneous Distributed Database System along with descriptions of the Semantic Binary Object-oriented Data Model and Semantic SQL query facility. Chapter 3 discusses the knowledge required for semantic heterogeneity resolution including a methodology for automated identification of semantic relations. Chapter 4 describes a language used for creation of Semantic Views over a set of component Sem-ODM database schemas resolving schema-level heterogeneities. Also, this chapter describes the Knowledge Base tool that assists in the creation of global views and Knowledge Base schemas used for the storage of such information. Chapter 5 discusses Semantic SQL query processing in a heterogeneous database system including strategies for optimizing using semantic knowledge. Chapter 6 provides an overview of a framework for distributed computing and communication paradigm in the Internet. This

chapter also details the use of techniques and methodologies developed in the previous chapters in the context of this paradigm. Chapter 7 provides the concluding remarks with some discussion on future research directions.

2. HETEROGENEOUS DISTRIBUTED DATABASE SYSTEM

The Heterogeneous Distributed Database System being developed at HPDRC provides access to a set of heterogeneous distributed data sources using the Semantic Binary Object-oriented Data Model and Semantic SQL query facility. Providing Semantic Access to a multitude of heterogeneous data sources is a more natural, expressive and rational approach for integrated data access. In addition, with the adaptation of SQL query language to Semantic Databases (called Semantic SQL), we have provided a popular declarative query facility for Semantic Data Access. These features and other advantages have illustrated that providing Semantic Access to a set of heterogeneous data sources reap significant benefits for data integration and query processing.

Subsequent sections of this chapter are organized as follows. Firstly, we provide an overview of some of the multidatabase systems that have been developed. Next, we introduce our prototype system by firstly discussing Semantic Binary Object-oriented Data Model. Semantic SQL query language and its adaptation to Sem-ODM schemas are discussed in the next section. System architecture of the Heterogeneous Distributed Database System including its main components is described in section 2.1.3. Finally, benefits of utilizing the proposed architecture and integration methodology are summarized.

2.1 Related Work

Many research prototypes and a few commercial heterogeneous/multidatabase systems have been developed. Following is a list of some multidatabase systems developed over the past two decades:

1. Multibase [63] is first designed and implemented multidatabase prototype. Multibase provides a uniform integrated interface for retrieving data from pre-existing heterogeneous distributed databases. It uses a global schema in a functional data model to create an integrated view of the data. A functional query language, DAPLEX [107], is provided as the query facility.
2. Amoco Distributed Database System (ADDS [15], [16]) is another industrial prototype of a multidatabase system. It provides both retrieval and update facilities on a set of component databases. The global schema is a relational schema with two relational query languages provided.
3. Pegasus ([102], [103]) is an effort by the Hewlett-Packard Laboratories in building a heterogeneous multidatabase management system. It provides an object-oriented data model schema with HOSQL query facility.
4. UniSQL/M [52] is a multidatabase system integrating relational and object-oriented schemas. A data definition and manipulation language, SQL/M, is used in the creation of global schemas.
5. Carnot [28] is an effort by Microelectronics and Computer Technology Corporation (MCC) to integrate heterogeneous data using the Cyc – knowledge base. Cyc knowledge base uses semantic information, in addition to structural knowledge in integration of different schemas. InfoSleuth [10] is an extension of

Carnot in a web environment with the use of agents and existence of unstructured dynamically changing data sources.

5. TSIMMIS ([23], [44], [49], [86]) is a system to integrate information from semi- and un-structured data sources. A common object model, Object Exchange Model (OEM), and a query language, OEM-QL, is developed. Major contributions of this research include the use of wrappers and mediators to obtain structure and query semi- and un- structured data sources.

6. Garlic [100], was an effort by IBM to integrate information systems and focuses on multi-media data and information. An object-oriented data modeling facility is used in the integration process.

Other projects include Mermaid ([19], [125]), Information Manifold [69], METU [36], OASIS [99].

It is evident from the above discussion that considerable effort has been focused into information integration during past two decades. We have observed a tendency to utilize expressive data models and querying capabilities to integrate heterogeneous data sources. For instance, earlier prototypes (i.e. Multibase) used functional data models and then relational data models were used (i.e. ADDS, etc.) and currently focuses on object-oriented data models (i.e. Pegasus, TSIMMIS, Garlic, etc.). However, with over two decades of research and systems being built using a multitude of methodologies, there has not emerged a generally accepted methodology or system in research or industry for integration and querying a set of heterogeneous data sources. Information integration is still a very active research area that has become critically important research issue

recently with the advent of the Internet allowing access to thousands and millions of heterogeneous distributed data sources and documents.

2.2 Our Work

Our approach to the development of a heterogeneous database system considers a scalable, easy-to-develop architecture using state-of-art technologies and use of Semantic Binary Object-oriented Data Model (as the canonical data model) and Semantic SQL query facilities (as the query facility to heterogeneous data sources).

2.2.1 Semantic Binary Object-oriented Data Model

Semantic Binary Object-Oriented Data Model (Sem - ODM) [94] combines the advantages of relational and object-oriented data models. Sem-ODM provides expressive data modeling capabilities, similar to object-oriented data models, but also has the simplicity of constructs similar to the relational data model [26] (which provides only one construct, namely *table*).

The central notion of the Semantic Model is an *object*. An object may be either *abstract* or *concrete*. An *abstract object* is any real world entity that may be stored in the database. An abstract object may be tangible (such as car, building, person) or intangible (such as idea, event). *Concrete objects* are printable objects (such as numbers, character strings or dates). Objects that possess common properties are grouped into classes called *categories*. The categories may or may not be disjoint allowing an object to belong to multiple categories simultaneously. Categories can be further divided into *Abstract* and

Concrete whose objects are always abstract or always concrete respectively. Categories can be inherited (called *subcategories*) from other categories (called *supercategories*). Objects of a subcategory are also objects of its supercategories. The category hierarchy does not contain a cycle for obvious reasons.

Every object in the real world contains properties ([21], [22]). Relationships between two categories are used to model properties (called *relations*). Relations have different cardinalities, such as *1:1*, *1:m*, *m:1* or *m:m*, specifying the maximum number of objects in the domain category and range category that may be related at any database instant via the relation. Also, a relation may be *total* which specifies that the existence of an object in the domain category requires the object to be related by the total relation. Further discussion on Sem-ODM can be found in [94]. In order to illustrate the expressiveness of the Semantic Model, we present the relational schema for the semantic schema presented in figure 2 of chapter 1. The semantic schema in figure 2 is a simple schema that does not contain expressive constructs such as sub-class/super-class relationships which cannot be directly represented in the relational data model.

STUDENT

student-id	name	email

WORK

student-id	project-id

PROJECT

project-id	name	description

FUNDED

project-id	grand-id

GRANT

grant-id	description	funding-agency	amount

Figure 3. Relational Schema Equivalent to the Sem-ODM Schema Presented in Figure 2

In order to represent *works-on* and *funded-by* relations which has cardinality *m:m*, two tables needed to be introduced (i.e. WORK and FUNDED). These tables are introduced to capture the relationships between tables rather than to provide means for storage of data. Also, in the table *PROJECT*, a field, *project-id*, which is the primary key field, is introduced. This spurious field was required in order to provide a relationship between tables GRANT and PROJECT. Note that *project-id* field does not capture any semantically useful information of the real-world. It is merely added as a means for creating a relationship.

This example illustrates a simple scenario of a Semantic Schema and its equivalent relational schema. It is apparent that the Semantic Schema captures semantically rich information set while relational databases require significant overhead to capture the same information and the end schema is not easy to understand by a human much less for a machine. Graphically, Sem-ODM schemas are represented as follows:

- Abstract categories are represented by rectangles with the name of category placed inside the rectangle;
- Subcategory relationships are represented by a dashed arrow from the subcategory to the supercategory;

- Relations are represented by thick arrows pointing from domain category to range category with constraints and cardinalities described in brackets;
- Attributes are placed in their respective domain categories with the range concrete category placed after ":" (semi-colon).

It is important to point out that Semantic Binary Object-oriented Data Model is a semantic data model, with object-oriented features incorporated. Semantic Data models are usually more powerful and more easy to use than current proposed object-oriented data models. They are especially more powerful in representing integrity constraints and various relationships. Object-oriented data models are generally based on class hierarchies and inheritance, plus their ability to represent the behavior of objects (for further details see [14]). Since Sem-ODM has object-oriented features incorporated, it allows the specification of methods and procedures in a category (see chapter 10 of [94]). However, the current implementation of Sem-ODB ([89], [91], [92], [97], [104], [117]) does not support the definition of methods. This has resulted in enabling us to adapt SQL-92 [110] for Semantic Database without any modifications to the syntax. Hence, in the subsequent chapters, we omit the discussion of methods and procedures. We can easily incorporate behavioral properties into the current Sem-ODB implementation and we are confident that we will be able to utilize up-coming standard SQL languages such as SQL-99 [20] when behavioral aspects are included into our implementation.

2.2.2 Semantic SQL Query Language

One of the major advantages contributing to relational databases' success is the standard query language, SQL, which is declarative in nature. Object-Oriented Databases' (OODB) query languages (such as OQL [13], and others) need to be correlated with an Object-Oriented Programming Language (OOPL) and/or are procedural in nature [58]. This has also resulted in the well-known problem of impedance mismatch. We have adapted SQL (specifically SQL-92 [110]) for Sem-ODM (called Semantic SQL [98]), thus providing a well-known declarative query language for Sem-ODM.

Semantic SQL [98] is the interpretation of SQL language, specifically SQL-92 [110], on Sem-ODM schemas. SQL-92 is a query language based on the relational data model. The basic constructs of the relational data model are *tables*. Basically, a SQL query statement is a set of operations on a set of tables of the relational schema. Thus, in order to adapt SQL for Semantic Binary Object-oriented Data Model, we provided a means to interpret *tables* from a Sem-ODM schema. The tables over a Sem-ODM schema are named *virtual tables*. Next, we adapted SQL for Sem-ODM, called *Semantic SQL*, which is SQL over virtual tables of a Sem-ODM schema.

Virtual tables of a Sem-ODM schema are named thus because they are never physically generated. Every virtual table is a finite representation of a spanning tree of Sem-ODM schema starting at a certain category in the schema. A formal recursive definition for virtual tables is presented in [98]. We provide it below for completeness.

Definition of virtual table T(C):

Let us consider virtual table T(C) where C is the starting category:

- C – attribute of T, range: C (m:1)

For every attribute A of T, for every relation *r* whose domain intersects with the range of A

- A__r – attribute of T, range: *range*(*r*) (m:1)

Note that this recursive definition may result in an infinite table (i.e. a table with an infinite number of attributes). A finite depth of this virtual table is determined by examining the query being processed. That is, we recursively generate the virtual table until all the attributes mentioned in the query statement are placed in the virtual table. After eliminating the extraneous paths traversed (i.e. paths traversed that do not contain the attributes mentioned in the query), the query is posed on the resultant virtual table. That is, the query is posed on the minimal virtual table containing all the attributes mentioned in the query statement.

Another aspect of the above definition is that attribute names for virtual tables are long in certain cases. Abbreviation of attribute names by eliminating the prefixes is allowed as long as no ambiguity arises. That is, attribute y of T is a synonym of the attribute X__y if T has no other attribute Z__y where depth(Z) ≤ depth(X) such that depth(x) represents the length of path x.

In order to understand the semantics for generating a virtual table, we re-iterate the definition of an extension of a virtual table from [98].

Definition of the Extension of a Table:

The virtual table T for a category C is logically generated as follows:

(1) Initially, t[C] = C, i.e. T contains one column called C whose values are the objects of the category

(2) For every attribute A of T, for every schema relation or attribute r whose domain may intersect range(A), let R be the relation r with its domain renamed A and range renamed A__r, let T be the left-outer-join of T with R (unlike a regular join, the outer join creates A__r = null when there is no match.)

Notice that during the creation of a virtual table, *null* values may be placed for every relation traversed. This issue is considered during query processing of Semantic SQL statements (see chapter 5).

Semantic SQL query language has identical syntax and semantics of SQL-92 with the exception that a Semantic SQL query statement is posed on (minimally projected) virtual tables generated for the Sem-ODM schema instead of actual physically resident tables in the database which is the case for relational database. There are many benefits of using Semantic SQL over Sem-ODM schema rather than SQL over its equivalent relational schemas such as the size of the resultant queries. To illustrate this feature, let us consider Semantic SQL query posed on the Sem-ODM schema provided in figure 2 and a semantically equivalent SQL query posed on the relational schema presented in figure 3.

Query: For every grant, obtain *grant-id*, the *names of project* that it funds and the *names of students* working for the project.

a. Semantic SQL:

```
SELECT    grant-id, funded-by___name, works_on___name
FROM      GRANT
```

b. SQL:

```
SELECT    GRANT.grant-id,  PROJECT.name,
          STUDENT.name
FROM      (((GRANT LEFT OUTER JOIN FUND ON
          (GRANT.grant-id = FUNDED.grant-id)) LEFT OUTER
          JOIN PROJECT ON(PROJECT.project-id =
          FUNDED.project-id)) LEFT OUTER JOIN WORK ON
          (PROJECT.project-id = WORK.project-id) LEFT OUTER
          JOIN STUDENT ON (WORK.student-id =
          STUDENT.student-id)
```

Figure 4. (a.) Semantic SQL Query posed on the Sem-ODM Schema (b.) SQL Query posed on the Equivalent Relational Schema

As apparent from figure 4, Semantic SQL queries posed on Sem-ODM schemas are much shorter than its counter-part SQL queries on an equivalent relational schema. Further discussion on benefits of using Semantic SQL is presented in section 2.4.

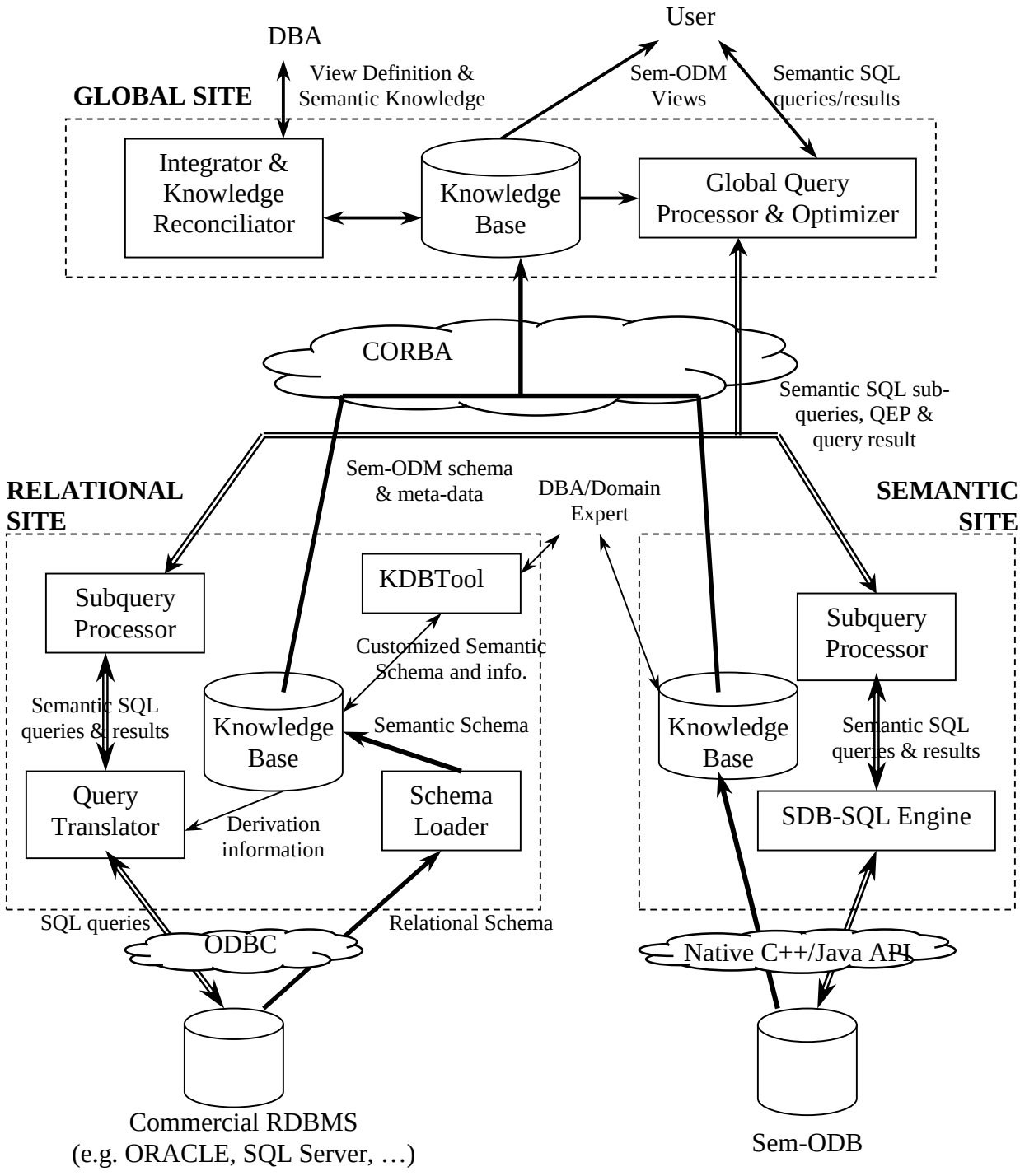


Figure 5. Architecture of Heterogeneous Distributed Database System

2.2.3 System Architecture

The system architecture of the Heterogeneous Distributed Database System is given in figure 5. It consists of three major components: (i.) Relational Site; (ii.) Semantic Site; and (iii.) Global Site.

- Relational Site: The relational site contains a relational database. The relational database is wrapped using a wrapper (i.e. SemWrap [74], [96], [97]) to provide the illusion of a Semantic Binary Object-oriented Database (Sem-ODB [92], [97], [104], [117]). The wrapper provides both a Sem-ODM schema and Semantic SQL query facility to the relational database. The components that make up the relational site are:

  (i.) Relational Database: The relational database is usually an existing commercial relational database (such as Oracle [85], SQL Server [109] or Access [2]). This commercial database is accessed using the Object Database Connectivity (ODBC [81]) protocol.

  (ii.) Schema Loader: This module imports the relational schema into the knowledge base. In addition, it creates an equivalent Sem-ODM schema for the relational database schema and stores this information along with derivation rules for schema mappings in the knowledge base. The schema transformation process is a bottom-up methodology similar to the reverse order of conversion described in [90].

  (iii.) Knowledge Base: This component acts as the interface among the Knowledge Base Tool (KDBTool), Schema Loader and Query Translator components. Knowledge Base stores schema information (semantic and relational

schemas), derivation information between relational schema and its equivalent semantic schema, and context information about constructs of the semantic schema. Further discussion about the knowledge base is presented in [74], [93] and subsequent chapters.

(iv.) Knowledge Base Tool (KDBTool): The relational schema does not have the ability to express complex semantics such as inheritance and *m:m* relations which are inherent to the Sem-ODM schemas. Hence, semantic schema generated by Schema Loader does not contain such complex structures. The DBA uses the KDBTool (also called Knowledge Base Editor) and Knowledge Base to add such complex features to the semantic schema along with derivation rules. The DBA also provides other semantic information such as context information, discussed in chapter 3, in order to resolve semantic heterogeneity.

(v.) Query Translator: The purpose of this module is to translate Semantic SQL queries based on the semantic schema to its equivalent SQL queries in the relational schema. To achieve this goal, the module uses the existing derivation information and schema information stored in the Knowledge Base. A detailed discussion on translation of Semantic SQL queries based on the Sem-ODM schemas to SQL queries based on the equivalent relational schema is presented in [74].

(vi.) Subquery Processor: This module receives the query execution plan (QEP) for the site and Semantic SQL subqueries. The Semantic SQL subqueries are passed to the query translator and the operations specified by the QEP are

performed on the query results and transmitted to the appropriate site. Note that postquery processing of the query results may be needed to resolve heterogeneities and/or integrate results from subqueries of remote data sources. These tasks are specified in the QEP and performed by the subquery processor.

- Semantic Site: This module implements the Semantic Database Engine (Sem-ODB [92], [97], [104], [117]), Semantic SQL interpreter ([97]), Knowledge Base and Subquery Processor. Sem-ODB engine is a multi-platform fully functional client-server database system (platforms include Solaris, HPUX, Linux, and various versions of Windows). Clients running on any platform can interact with one or more database servers running on the same or different platforms. Moreover, database files are fully compatible across platforms at binary level. Multiple clients can access server through network protocols such as TCP/IP or NETBIOS while some other clients can run locally as threads within the server process. In addition to the SQL-level access provided by Semantic SQL interpreter, the database engine provides a native C++ and Java API for elementary database access, similar to procedural access in an OODB. SQL-level access is provided by the SDB-SQL Server which interacts with the database engine via the elementary database access interfaces. Subquery Processor at the semantic site performs similar tasks as in the relational site. Knowledge Base interacts with the domain expert to obtain context information. Further discussion on Semantic Site including Sem-ODB and SDB-SQL Server can be found in [92], [97] and [117].

- Global Site: The significant tasks and processing for integration and global query processing are performed at the global site. Resolutions of heterogeneities such as semantic heterogeneities, global view definitions including resolving schematic heterogeneities are performed at this level. Also, global query processing and optimization of users' Semantic SQL queries are carried out. During the incorporation of a data source into the Heterogeneous Distributed Database System, a Sem-ODM schema is imported to the global site including its relevant context information. This meta-data is integrated to existing knowledge in a semi-automated methodology (described in chapter 3). The Integrator and Knowledge Reconciliator module performs the integration process, including semantic heterogeneity resolution and schema-level heterogeneity resolution. The global views are created by the DBA (see chapter 4) and stored in knowledge base along with relevant meta-data and semantic information. The users pose Semantic SQL queries on the Sem-ODM global views. The Global Query Processor and Optimizer module creates an optimized query execution plan and a set of subqueries to obtain the results for the users' query. In order to accomplish this task, this module uses knowledge acquired during integration process which is stored in the knowledge base. The subqueries along with the appropriate QEPs are transmitted to the relevant sites to obtain results for the query. Certain postquery processing of the results for subqueries are executed at the component and global sites as specified by the QEP.

It is significant to note the communication protocols that have been used in the architecture for inter-site communication and from the wrapper to relational database.

For inter-site communication, we have used Object Management Group's (OMG [84]), Common Object Request Broker Architecture (CORBA), which is an industry standard for application development within heterogeneous distributed environments. CORBA provides a network transparent distributed computing medium for developing applications on a distributed heterogeneous environment. CORBA consists of numerous features, including ORB Core, Interface Definition Language (IDL), Stubs, Skeletons, Services (such as Name Service, Query Service) and others. The main feature of ORB Core is its abstractions of the object implementations. Due to these features, the application developer need not consider the state of the object, how to communicate the remote object (such as TCP/IP, RPC, etc.) and other complexities. The use of CORBA has significantly reduced the effort and complexity in developing our system. Further information on CORBA and its use in Heterogeneous Distributed Database System are provided in [74]. We enumerate some of the benefits of using CORBA in section 2.4. In addition, we developed common interfaces (APIs) to access the Semantic as well as Relational sites thus re-using much of the code for accessing data sources.

At the relational site, Object Database Connectivity (ODBC) protocol and standard query language, SQL, is utilized. The use of such industry-wide standards has achieved portability and reusability to a very high-degree. We discuss these issues in the next section which outlines the benefits of using the architecture and methodology for the Heterogeneous Distributed Database System.

2.3 Benefits

Our approach in designing and developing the Heterogeneous Distributed Database System provides many advantages from the use of Semantic Binary Object-oriented Data Model, Semantic SQL query language, CORBA architecture, standard query languages such as SQL, standard protocols such as ODBC and other design considerations. We discuss these aspects below.

There are many advantages of using Sem-ODM as the canonical data model in the Heterogeneous Distributed Database System. They include:

(i.) A semantically expressive data model capturing the meaning of information content in a set of heterogeneous distributed data sources. Expressive modeling capabilities include *m:m* relations, disjoint categories, inheritance, arbitrary relations, multi-valued attributes and others.

(ii.) Due to the fact that Sem-ODM captures the semantics of the information content presented, it provides (a.) friendlier and more intelligent generic user interfaces; (b.) comprehensive enforcement of integrity constraints; (c.) greater flexibility; (d.) substantially shorter application programs; and (e.) easier query facility.

By adapting SQL for Sem-ODM, we have gained many benefits including,

(i.) A well-known declarative query language;

(ii.) The ability to use existing relational tools. That is, with the SQL interface provided to the Sem-ODM, we can reutilize tools that execute on relational database platforms without any modification;

(iii.) Easier query facility. We have gained an easier and less complex query facility. A Semantic SQL query over a Sem-ODM is significantly shorter and less complex than a SQL statement on an equivalent relational schema. This is, due to the ability to traverse relations in a semantic schema without specifying joins and the expressiveness of Sem-ODM when compared to relational schemas. This feature is demonstrated in [74], [96] and [97].

The use of CORBA as the communication protocol between component and global sites has:

(i.) Significantly reduced the complexity and effort required in developing the system;

(ii.) Resulted in faster development time. CORBA's ORB and Name Service is utilized to locate, identify and communicate to component data sources transparent of the network. This has resulted in less complexity and faster development time;

(iii.) CORBA's Object Model has resulted in modular design. Every data source is considered as a CORBA object with a common interface;

(iv.) Platform and network level heterogeneity is resolved. A common interface to all data sources avoids the use of different communication and/or network protocols;

(v.) Scalability: CORBA architecture provides scalability by allowing hundreds and/or thousands of data source to be seamlessly incorporated into the system.

During the development of the wrapper for relational databases, we utilized the standard query language SQL and ODBC protocol. This has resulted in many advantages including,

(i.) Reusability: That is, use of ODBC and SQL has resulted in enabling the wrapper to be plugged into any commercial relational database system (consisting of required ODBC Driver) without any modification;

(ii.) Portability: The wrapper can connect to databases residing on different platforms.

By providing a common interface (that is, Sem-ODB interface) to component database, we gain many benefits such as:

(i.) Extendibility: A new type of data source can be integrated into the Heterogeneous Distributed Database System by providing a Sem-ODM interface to the data source. Such an approach is considered in Data Extractor project [12], which integrates semi- and un- structured data from the Web data sources into the Heterogeneous Distributed Database System;

(ii.) Reusability: Since all data sources contain a common Sem-ODM interface, Subquery Processor module and CORBA IDL interfaces can be reused;

(iii.) Preserved autonomy of data sources: Our architecture preserves component databases' autonomy. That is, component data source does not require any changes in order to participate in the Heterogeneous Distributed Database System.

The transformation of component data schema to a common model and exportation of these schemas to the global sites has been discussed in [105]. The use of

CORBA as a distributed communication and integration medium for resolving network and platform heterogeneity is seen in METU ([36], [37]) and OASIS [99] multidatabase systems. OASIS uses a translation knowledge base similar to the relational wrapper, SemWrap [96]. Also, wrappers and mediators are used in TSIMMIS ([23], [44], [49], [86]). However, our architecture is unique with the fact that we have incorporated a Semantic Binary Object-oriented Data Model (i.e. a semantic object-oriented data model) with SQL query language (i.e. a well-known declarative query language) to provide a semantic access to a set of heterogeneous distributed data sources. The architecture described in this section is extendible, scalable, resolves platform and network heterogeneity, preserves autonomy of data sources and provides a common interface (data model and query facility) to component databases providing integrated access to heterogeneous distributed data sources. However, as mentioned earlier, ubiquitous deployment of heterogeneous database systems is hindered by the difficulty of resolving semantic heterogeneity. The next two chapters address this issue in detail.

3. SEMANTIC HETEROGENEITY RESOLUTION

As mentioned in chapter 1, a significant impediment for ubiquitous deployment of multidatabase technologies is the difficulty in identifying semantically related entities of different database schema. To illustrate this problem, we will use the following example.

Example 2: Let us consider the following semantic schema of a university application.

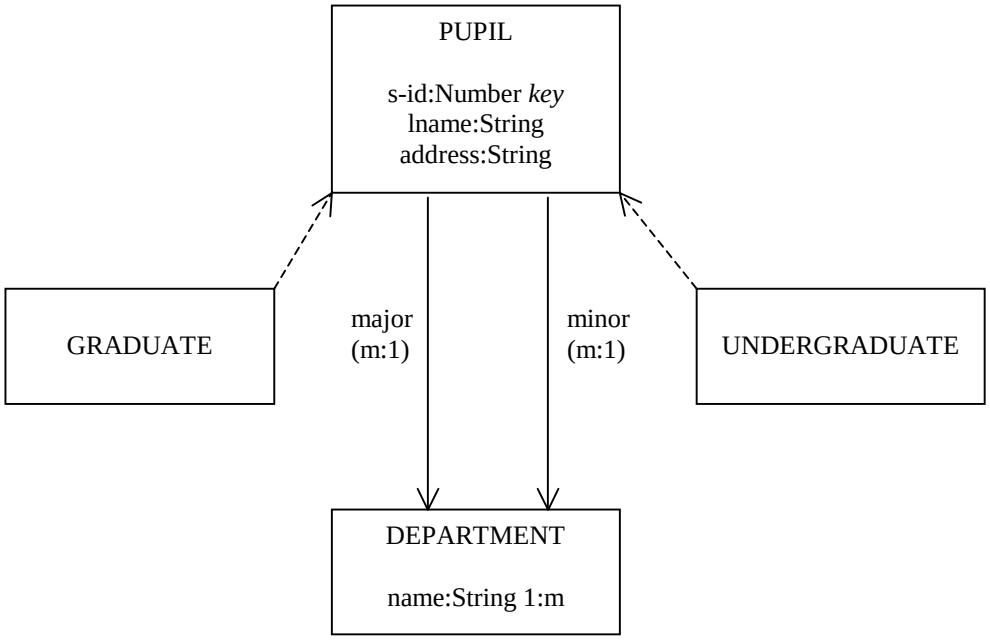


Figure 6. Semantic Schema of a University Application

Provided with schemas such as in figures 3 and 6, how do we integrate them? This is the problem faced by heterogeneous database researchers. This is an over simplified example. Consider a scenario of hundreds of schemas independently developed being provided and asked to integrate them. The answer to the above-mentioned problem necessitates two steps for its solution.

Step 1: Identify the constructs of the schemas that capture the same real-world concepts.

Step 2: Represent these constructs in a non-redundant, meaningful way.

Step 1 pertains to semantic heterogeneity resolution. Step 2 pertains to schema-level heterogeneity resolution. In this chapter, we will consider semantic heterogeneity resolution. In the next chapter, we discuss the schema-level heterogeneity resolution schemes.

Step 1 discussed above seems to be a simple problem. Let us investigate in detail. Looking at the schemas provided in figures 3 and 6, we *kind of* see that category PUPIL in schema of figure 3 is related to category STUDENT of figure 6. We figured this relationship based on our previous knowledge on what PUPIL and STUDENT meant and probably looking at the structures and relations within the schema which seem to correspond. Now if we take away all the pre-assumed and context knowledge based on which we made the previous conclusion, just taking into consideration the schema diagrams by themselves, can we conclude any relationship. The answer is obviously "No". Thus, it is clear that even humans are unable to conclude relationships between constructs of schemas without the appropriate knowledge to make these decisions. Thus, it is safe to conclude that computers & programs cannot determine relationships without providing the appropriate knowledge ("assuming that humans are intelligent than computers"). Practically, in real world situations, we are faced with schemas of a large number of legacy systems without adequate knowledge on what schemas capture in their data sources. Now it is clearer as to why heterogeneous database researchers are moving in the direction of utilizing expressive data models that capture more information to be used for integration. This is to obtain as much knowledge as possible so as to make intelligent design decisions in integration. In our approach, we used the Semantic Binary

Object-oriented Data Model instead of relational or object-oriented data model since we are convinced that Sem-ODM is expressive to capture the semantics of the data being modeled.

3.1 Related Work

In this section, we consider the existing approaches proposed for identification of related entities of different database schemas. Next, we discuss answer-completeness of queries and illustrate why current approaches fail to satisfy this requirement.

In the early work, such as the twelve approaches outlined in [9], [105] and others, we have seen the assumption that step 1 is resolved manually and focuses on techniques for resolving step 2, which is the representation issue. This approach may result in good integration, however require the integrator(s) to familiarize themselves with the schemas of component data sources and place much effort into integration. With a large number of schemas, this may be impractical and automation of semantic heterogeneity resolution is a highly desirable goal.

In [83], domains (extents) of the schema constructs in Entity-Category-Relationship (E-C-R) model ([42], [122]) are considered for resolving semantic heterogeneity. Also, methodologies for resolving schema level heterogeneities with different domain relationships are outlined. In [65], the authors present a heuristic method for determining the different domain relationships by using attribute equivalences based on the common principle of integrating attributes. A tool developed, using these

principles, to assist database designers in schema integration and modeling is discussed in [106].

In [18], a semantic heterogeneity resolution methodology for multidatabase language system is presented. The Summary Schemas Model is introduced which uses a global data structure to abstract information available in a multidatabase system. The use of linguistic theory for translation of users' queries to a set of system imprecise queries is discussed. The important feature is that semantic heterogeneity is resolved by translating users' queries to a set of semantically related system defined terms. The resultant queries are imprecise and provide imprecise answers unlike centralized homogeneous database systems. However, a global schema is not created thus avoiding this effort.

In [6], a stepwise methodology to obtain information from a remote schema is presented and integrated with the local schema. This methodology uses a Heterogeneous Semantic Data Model (HSDM) as the canonical data model to enrich the remote schema with semantic information. However, this methodology uses knowledge extensively from local domain expert and remote domain experts for integration decisions. For legacy systems, it may be difficult to obtain such information from domain experts.

In [70], neural networks are trained (based on field specifications and data contents) to identify equivalent attributes and similarity constructs of schemas. In [50], MUVIS system is introduced. MUVIS determines the degree of similarity and

dissimilarity of two objects based on comparing the field names of the attributes. Next, this tool provides recommendations on the integration process.

All of the approaches discussed above provide with non-exact reasoning techniques for semantic heterogeneity resolution. For instance, methodology presented in [18] results in imprecise queries, techniques described in [50], [65], [70] uses heuristic based approaches and conclude via names and structures specification of attributes, thus it is possible to result in incomplete incorrect answers and/or recommendations. This leads to problems such as obtaining incomplete answers for queries. An example best illustrates the problem of answer-completeness.

Example 3. Let us consider accessing two databases (i.e. DB_1 and DB_2) with the following schema (figure 7):

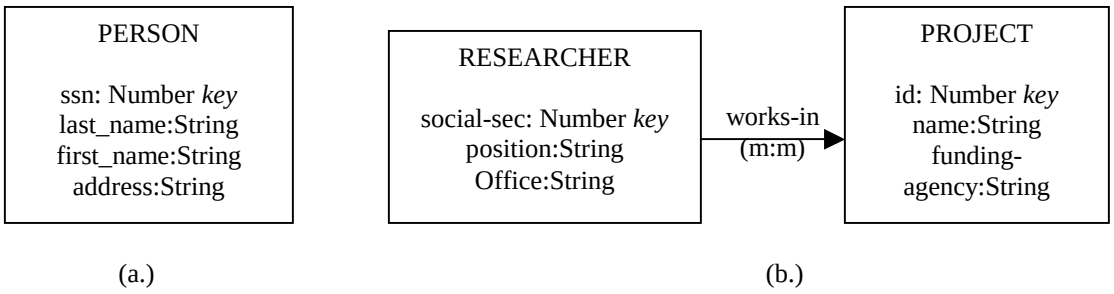


Figure 7. (a.) Schema of Database DB_1 in Administration Office of Company A
(b.) Schema of Database DB_2 in Lab L of Company A

Note that we consider only Sem-ODM schemas for integration since the architecture presented in chapter 2 provides a Sem-ODM schema of every component data source. However, our presentation of answer completeness problem can be extended to the any data model without loss of generality.

The category *PERSON* in database DB_1 contains objects describing the currently employed personnel at company A. Database DB_2 describes researchers and their projects at lab L of company A since its inception. The category *RESEARCHER* contains the researchers working or has worked at lab L of company A. The category *PROJECT* contains projects that the lab is currently working or already completed. The attributes *social-sec* and *ssn* in categories *PERSON* and *RESEARCHER* represent the social security number of a person.

Let us consider the above-mentioned approaches to schema integration. Accordingly, categories *PERSON* and *RESEARCHER* will be mapped as *equivalent* since they both represent personnel working at company A or mapped as a sub-category/super-category relationship because categories *RESEARCHER* represent a specialized class of all personnel working at company A represented by category *PERSON*. Thus, the integration process results in the following integrated schema:

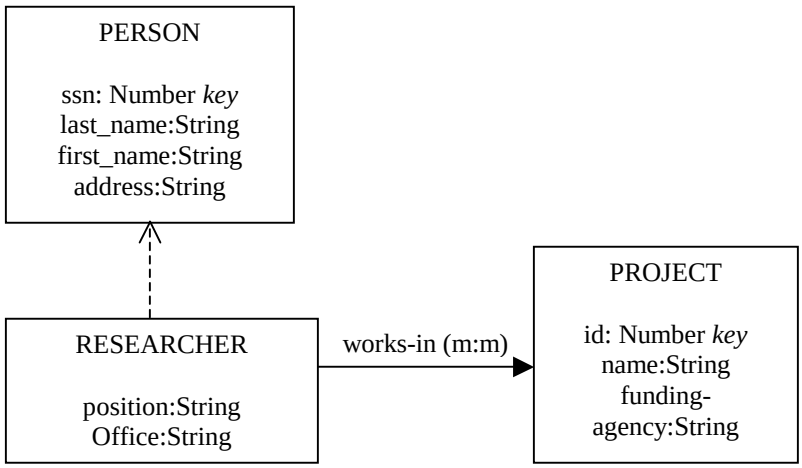


Figure 8. Integrated Schema for Schemas Presented in Figure 7

The derivation rules for the integrated schema are as follows. Note that → represents *is derived from* semantics:

$Rule_1$: PERSON → DB_1.PERSON

$Rule_2$: RESEARCHER → DB_2.RESEARCHER

$Rule_3$: PROJECT → DB_2.PROJECT

$Rule_4$: works-in → DB_2.works-in

Note that derivation rules for attributes are not presented here (as they are obvious). Equivalence condition for common objects of PERSON and RESEACHER is as follows:

DB_1.PERSON.ssn = DB_2.RESEARCHER.social-sec

The schema in figure 8 along with above-mentioned derivation rules can be considered as a reasonable result of integration using the approaches discussed above (such as based of name equivalences and structural equivalences).

Let us now consider the query, which obtains the last names of researchers who worked or are working at lab L, and the names of the projects they worked on or are working on. This query can be represented by the following Semantic SQL query on the integrated schema (see figure 8):

```
SELECT   RESEARCHER.last_name, RESEARCHER.works-in__name
FROM     RESEARCHER
```

The heterogeneous/multidatabase or mediator system translates this query (based on the derivation rules) to the set of operations depicted by the following SQL statement:

```
SELECT   DB1.PERSON.last_name, DB2.RESEARCHER__works-in__name
```

FROM DB_1.PERSON, DB_2.RESEARCHER

WHERE DB_1.PERSON.ssn = DB_2.RRESEARCHER.social-sec

Note that the result of this query only suffices to provide only a partial answer. Researchers who have worked on a project at lab L but **not currently employed** in company A are not represented in the query result. This aspect is known as *answer-completeness* [68], [82] of queries. This issue becomes a critical factor when dealing with multiple databases.

Our approach, based on extent of schema constructs, for database integration and query processing of multitude of data sources is guaranteed to avoid errors such as incomplete answers. A very desirable goal of heterogeneous databases users is obtaining relevant, complete, correct information from a variety of available heterogeneous distributed data sources. These factors translate to successful integration of data sources and answer-completeness of user's queries. Our approach addresses both these situations successfully. This approach is discussed in detail in section 3.3.

3.2 Our Work

The goal of our methodology is to achieve reliable, correct and complete answers to users' requests from a heterogeneous database management system, similar to centralized database system, through unambiguous, complete and correct integration. It is important to note that in achieving this goal, we incorporated many techniques discussed in previous approaches into our methodology. Our methodology is based on the extents of schema constructs similar to [83]. It is apparent from our previous discussion that

without appropriate knowledge, we are unable to achieve correct integration. We employ a step-wise process similar to [6] to gain such knowledge. In [18] a global data structure was used for matching users terms with systems concepts. In our approach we use shared ontologies to obtain the context meanings of schema constructs.

In describing our methodology, we will first introduce the foundations of semantic knowledge, which is the basis for integration. Next, we outline a methodology based on ontological concepts to semi-automatically obtain semantic knowledge.

3.2.1 Foundations of Semantic Knowledge

Our approach to resolving semantic heterogeneity assumes the existence of a schema describing the information content of a data source. The architecture (discussed in chapter 2) provides us a Sem-ODM schema for every component data source. The schema of a data source provides us with an unambiguous definition of the data content of the source, whether easily comprehensible or not. The schema captures the original database designer's intent of precisely what is stored in the database. The data is stored as a set of data items (extent) for each construct in the schema. Utilizing this information, we propose a set of relations, called semantic relations, which exploits both schema and its extent in database integration and query processing. This is similar to the domain relations discussed in [83]. However, we extend this concept in many ways to provide a complete basis for integration. With the use of the semantic relations as the basis in integration, we can easily preserve data quality attributes including completeness and

accuracy, which is not guaranteed in the approaches using heuristic methods based on name equivalences.

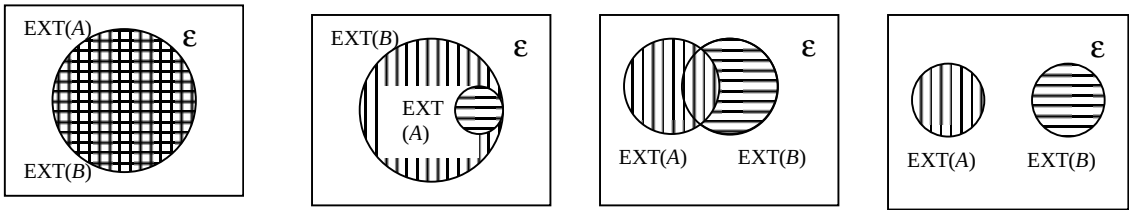
3.2.1.1 Semantic Relations

We have identified four semantic relations between entities of different schema. Before discussing the semantic relations, we introduce the notation, EXT(*A*) which is used to represent the extent of schema construct *A*. Let *A* be a construct of $Schema_1$ and *B* be a construct of $Schema_2$. We can derive four possible semantic relations between constructs *A* and *B* as follows: There are as follows:

1. Semantically Equivalent (*SEM_EQ*): *A* is semantically equivalent to *B* (represented as, *A SEM_EQ B*) if and only if EXT(*A*) = EXT(*B*) for all database instances at any given time t.
2. Semantically Subset (*SEM_SUB*): *A* is semantically subset of *B* (represented as, *A SEM_SUB B*) if and only if EXT(*A*) $\subseteq$ EXT(*B*) for all database instances at any given time t_1 and EXT(A) $\subset$ EXT(B) for some database instance at time t_2.
3. Semantically Overlap (*SEM_OVER*): *A* is semantically overlapping with *B* (represented as, *A SEM_OVER B*) if and only if EXT(*A*) $\cap$ EXT(*B*) $\neq \phi$ for some database instances at time t_1 and EXT(*A*) $\cap$ EXT(*B*) $\neq A$ or EXT(*A*) $\cap$ EXT(*B*) $\neq B$ for all database instances.
4. Semantically Disjoint (*SEM_DIS*): *A* is semantically disjoint with *B* (represented as, *A SEM_DIS B*) if and only if EXT(*A*) $\cap$ EXT(*B*) = ϕ for all database instances at any given time t.

Note that the semantic relations are disjoint. That is, if *A* r_1 *B* and *A* r_2 *B* where r_1, $r_2 \in$ {*SEM_EQ, SEM_SUB, SEM_OVER, SEM_DIS*}, then $r_1 = r_2$.

Proof Sketch: The completeness and correctness of the above semantic relations can be verified by examining all the possible scenarios of a Venn diagram for the extents of constructs *A* and *B* (see figure 9(a.)-(d.)). EXT(*A*) and EXT(*B*) are shaded in the figure.



(a.) (b.) (c.) (d.)

Figure 9. All Possible Scenarios for EXT(A) and EXT(B): (a.) EXT(A) = EXT(B); (b.) EXT(A) ⊆ EXT(B); (c.) EXT(A) ∩ EXT(B) ≠ ϕ; (d.) EXT(A) ∩ EXT(B) = ϕ

Note that ε represents the {domain of database containing construct A} ∪ {domain of database containing construct B}. Figure 9(a.) – (d.), depict all possible cases for semantic relations between any two database constructs *A* and *B*.

Some commutative rules and inference rules for semantic relations are enumerated below:

Rule 1: *A SEM_EQ B* ≡ *B SEM_EQ A*

Rule2: *A SEM_DIS B* ≡ *B SEM_DIS A*

Rule 3: *A SEM_OVER B* ≡ *B SEM_OVER A*

Rule 4: If *A SEM_EQ B* and *B SEM_EQ C* then *A SEM_EQ C*

Rule 5: If *A SEM_EQ B* and *B SEM_SUB C* then *A SEM_SUB C*

Rule 6: If *A SEM_EQ B* and *B SEM_OVER C* then *A SEM_OVER C*

Rule 7: If *A SEM_EQ B* and *B SEM_DIS C* then *A SEM_DIS C*

Rule 8: If *A SEM_SUB B* and *B SEM_SUB C* then *A SEM_SUB C*

Rule 9: If *A SEM_SUB B* and *B SEM_DIS C* then *A SEM_DIS C*

where *A, B, C* are constructs of different database schemas. The correctness of each rule can be directly verified using Venn diagrams or using set theory principles and thus not discussed any further.

The following example illustrates each semantic relation:

Example 4. Let us consider five constructs of different database schema in a university application.

Database	**Construct**	**Extent**
Registrar	*Employee*	contains all current employees of university A
Registrar	*Student*	contains all currently enrolled students of university A
Registrar	*Department*	contains all the departments of university A
Payroll	*Faculty*	contains all current faculty of university A
Payroll	*Emp*	contains all current employees of university A

By considering the extents, we can assume the following. In section 3.2.2, we discuss a methodology to automatically identify these relations.

- Registrar.*Employee SEM_EQ* Payroll.*Emp* (since both constructs represents the current employees of university A)
- Payroll.*Faculty SEM_SUB* Registrar.*Employee* (since *Faculty* construct contain the current faculty members of university A who are also employees of university A)
- Payroll.*Faculty SEM_OVER* Registrar.*Student* (assuming that the faculty member can also be registered to courses as students in university A)
- Registrar.*Department SEM_DIS* Payroll.*Emp* (since departments cannot be employees for obvious reasons)

Utilizing the rules described above, we can generate the following semantic knowledge from existing knowledge:

Payroll.*Faculty SEM_SUB* Registrar.*Emp* (Rule 5)

Registrar.*Department SEM_DIS* Payroll.*Employee* (Rule 7)

Payroll.*Faculty SEM_DIS* Registrar.*Department* (Rule 9)

The above-mentioned rules are important in gaining new knowledge from existing semantic relations and also for checking correctness and consistency of the existing knowledge in the knowledge base. Further discussion on the knowledge base can be found in chapter 4.

3.2.1.2 Object Equivalence

When two constructs, say *A* and *B*, are known to be semantically related by either *SEM_EQ*, *SEM_SUB* or *SEM_OVER*, it is possible for EXT(*A*) and EXT(*B*) to have the same real-world objects represented (i.e. this is the set of objects in EXT(*A*) ∩ EXT(*B*)). The identification of equivalent objects in different constructs is especially advantageous in schema integration. This allows extraction of extra information. To illustrate this fact, we provide the following example.

Example 5. Let us consider two databases schemas DB_1 and DB_2 consisting of students at university A:

Figure 10 (a.) Category of Database DB_1 Containing Information of Students in University A (b.) Category of Database DB_2 Containing Information of Students in University A

For simplicity, let us assume that DB_1.*PUPIL* SEM_EQ DB_2.*STUDENT* and attributes, *ssn* and *social_sec*, represent social security numbers in the same format and they are key attributes of categories DB_1.*PUPIL* and DB_2.*STUDENT* respectively. Hence, if DB_1.*PUPIL*.ssn match with DB_2.*STUDENT*.social_sec, implies that objects are equivalent (i.e. the same student).

Since *Pupil* SEM_EQ *Student*, every object in *Pupil* has a matching object in *Student* and vice-versa at every database instance. These matching objects are identified by comparing attributes DB_1.*PUPIL*.*ssn* and DB_2.*STUDENT*.social_sec. Hence, it is

possible to obtain a category, say STD, in global schema, which contains attributes: *social_security*, *address*, *gpa*, and *phone* for every student object of university A. This information cannot be obtained by accessing DB_1 or DB_2 individually. That is, it was possible to obtain additional information (i.e. address, gpa, phone attributes collectively) for every student in university A using an integrated access to DB_1 and DB_2. This example illustrates a simple scenario; this concept can be generalized for complex schemas.

In order to make the semantic relations useful for schema integration we obtain a condition which when satisfied will identify the common objects in entities A and B. The direct methodology is to identify a key attribute(s) that match in the two entities. For instance, in our previous example, we have equivalence condition as: DB_1.*PUPIL*.ssn = DB_2.*STUDENT*.social-sec. In the general case, obtaining such equivalent key attributes between entities DB_1.A and DB_{n+1}.B may not be possible. Then we try to gain an equivalence condition by using the following theorems.

Theorem 1. There exists an equivalence condition from entity DB_1.*A* to entity DB_{n+1}.*B* as follows:

(DB_1.*A*.KeyAttr = DB_2.CAT_2.KeyAttr) AND

(DB_2.CAT_2.KeyAttr = DB_3.CAT_3.KeyAttr) AND

….

(DB_k.CAT_k.KeyAttr = DB_{k+1}.CAT_{k+1}.KeyAttr) AND

….

(DB_n.CAT_n.KeyAttr = DB_{n+1}.*B*.KeyAttr)

where DB_k.CAT_k *sem_rel* DB_{k+1}.CAT_{k+1} such that *sem_rel* ∈{SEM_EQ, SEM_SUB} and DB_k.CAT_k.KeyAttr = DB_{k+1}.CAT_{k+1}.KeyAttr represents the equivalence condition between entities DB_k.CAT_k and DB_{k+1}.CAT_{k+1}.

Proof Sketch: Since DB_k.CAT_k and DB_{k+1}.CAT_{k+1} are related by SEM_EQ or SEM_SUB, joining the objects of DB_k.CAT_k and DB_{k+1}.CAT_{k+1} by the equivalence condition does not result in any loss of objects in DB_k.CAT_k. Thus by continuing on the path joining iteratively we gain attributes until finally, we gain an attribute which match the key attribute of DB_{n+1}.B.

Theorem 2: If there exists an equivalence condition from entity DB_1.*A* to entity DB_{n+1}.*B* as follows:

(DB_1.*A*.KeyAttr = DB_2.CAT_2.KeyAttr) AND

(DB_2.CAT_2.KeyAttr = DB_3.CAT_3.KeyAttr) AND

….

(DB_k.CAT_k.KeyAttr = DB_{k+1}.CAT_{k+1}.KeyAttr) AND

….

(DB_n.CAT_n.KeyAttr = DB_{n+1}.*B*.KeyAttr)

where DB_k.CAT_k *sem_rel* DB_{k+1}.CAT_{k+1} such that *sem_rel* ∈{SEM_EQ, SEM_SUB} and DB_k.CAT_k.KeyAttr = DB_{k+1}.CAT_{k+1}.KeyAttr represents the equivalence condition between entities DB_k.CAT_k and DB_{k+1}.CAT_{k+1}.

Then, the reverse traversal generates an equivalence condition from $DB_{n+1}.B$ to entity $DB_1.A$.

Proof Sketch: Proof idea for theorem 1 says that traversing from entity A to B does not result in any loss of common objects. Although, reverse traversal of joins may loose some objects in $DB_{n+1}.B$, it does not loose common objects (by previous proof). Thus common objects can be identified by reverse traversal.

3.2.1.3 Boundary Conditions

When either semantic relations, SEM_SUB or SUM_OVER relates two constructs, it is important to consider the boundary conditions on which the two constructs intersect. Considering these boundary conditions provides useful knowledge similar to object equivalence which otherwise is not explicit. The boundary conditions are rarely given importance (for instance, in [46] where only intersection classes are considered and not boundary classes). However, considering boundary conditions provide significant semantics which otherwise is lost. The example below illustrated this issue:

Example 6. Let us consider the scenario presented in example 3. Since category *PERSON* contains all the employees currently working for company A and category *RESEARCHER* contains all the persons who worked or are working in lab L of company A, by considering the extents, we can infer that *PERSON* SEM_OVER *RESEARCHER*. The persons currently working at lab L who are also employees of company A consists of EXT(*PERSON*) $\cap$ EXT(*RESEARCHER*). Current employees of company A not working

in lab L are in {EXT(*PERSON*) – EXT(*RESEARCHER*)}. Researchers who used to work at lab L, but are not presently employees of company A are in {EXT(*RESEARCHER*) – EXT(*PERSON*)}. This semantic knowledge allows us to extract more information as shown below.

For instance, we can now answer the query that asks for social security numbers of researchers who worked in lab L but have left company A (not currently working for company A) as follows:

SELECT DISTINCT DB_2.*RESEARCHER*.ssn

FROM DB_2.*RESEARCHER*

WHERE DB_2.*RESEARCHER*.ssn NOT IN

(SELECT DB_1.*PERSON*.ssn

FROM DB_1.*PERSON*)

This information could not be obtained by accessing the databases individually or without considering the extents. Note that in example 3, EXT(*RESEARCHER*) was considered a subset of EXT(*PERSON*) using previous methods. Thus, query mentioned above could not be posed or it would result in an empty result (i.e. incorrect answer). This example illustrates a simple case, but can be generalized for complex schemas. In addition boundary conditions is used for optimizing queries which is discussed further in chapter 5.

In our schema integration methodology, we consider semantic relations, object equivalences and boundary conditions. This will result in complete, correct and

unambiguous integration and querying of heterogeneous data sources. These aspects will be discussed in detail in section 3.3. Next, section discusses a methodology for automated identification of semantic relations.

3.2.2 Identification of Semantic Relations

Even though our approach provides complete, correct and unambiguous integration and querying, we have not discussed an easy way to identify the semantic relations discussed in the previous section. This becomes a bottleneck in the use of such knowledge during integration of a large number of schemas. Thus, this section investigates into this issue and provides a semi-automated methodology for identifying semantic relations. Our methodology is not based on heuristics, unlike previous attempts, and thus is guaranteed to provide correct results.

The main problem in resolving semantic heterogeneity and identifying semantic relations alike is the lack of appropriate knowledge and a need for automated identification of semantic knowledge. Usually, the intended extents are in the original database designer's mind, represented partially in conceptual models and to a lesser extent in the schemas. However, in most application domains, the only high-level knowledge available to the integrator is schema-level descriptions of legacy databases. The direct method of comparing extents is incorrect and impossible. In this section, we propose a stepwise methodology for identification of semantic relations without looking into all the possible combinations of attributes or the extent of classes. This methodology borrows many concepts from a variety of computer science research areas. Concepts

from ontology-based research, artificial intelligence (AI) and heterogeneous database research are incorporated. Section 3.2.2.1 outlines some of the relevant work from the different disciplines. Section 3.2.2.2 describes the methodology for semantic relations identification providing detailed discussion of each step.

3.2.2.1 Relevant Work

We have incorporated concepts from a number of disciplines including ontology (philosophy), semantic networks and classification techniques (artificial intelligence and biology) and databases. In this section, we describe these concepts prior to illustrating how they have been incorporated into our methodology for clarity. Section 3.2.2.1.1 discusses the ontological aspects. Section 3.2.2.1.2 discusses the semantic networks and classification of concepts.

3.2.2.1.1 Ontological Foundations

In this section, we describe some concepts from Bunge's ontological model ([21], [22]). His model articulates a set of high-level, abstract constructs that are intended to be means of representing all real-world phenomena. Bunge's ontological framework is well known and used by others ([118], [119], [120]) to analyze phenomena within computer science and information systems domains. Hence, we feel it is a good candidate for our work as well. In this section, we re-state some of the definitions and postulates of Bunge's ontology for completeness and clarity of our discussion. We also refer to some concepts from [118] in this section. A "*" (star) is placed to represent concepts of Bunge's ontological model.

Postulate 1*: The world is made of *things* that possess *properties*.

In OO modeling methodology, a thing is equivalent to an object. Thus, in a conceptual model it is an instance.

Rule 1: An *instance* in a conceptual model is a representation of a *thing* in the ontological model [118].

Postulate 2*: There are no things without properties. Moreover, properties are attached to things.

A property can depend on one or more things. A distinction is made between:

- *intrinsic* properties – properties that depend only on one thing only; and
- *mutual* or *relational* properties – properties that depend on two or more things.

For instance, the weight of a person is an intrinsic property, because it depends only on the existence of the person. The property of being an employee is a mutual property, because it depends upon the existence of both a person and a tertiary institution.

Rule 2: All attributes and relationships in an instance in conceptual model are representative of properties of things in ontological model [118].

"The properties of a thing exist, whether or not humans are aware of them. Humans conceive of things, however, in terms of *models* of things. Attributes are characteristics assigned to (model of) things according to human perceptions. Depending upon circumstances, humans may use different models of the same thing and therefore assign different sets of attributes to the same thing" [118]. For instance, let us consider a

database in a university registrar office that model students records, including level of study of a student, transcript information and tuition payment information. Another database at a department in the university may model information regarding students' projects, papers published and other related information. The same student things are assigned different attributes and relationships since they model different perspectives.

Rule 3*: Properties themselves cannot have properties.

For instance, at first glance, the height of a person may seem to have a property associated with the time at which the height was measured. The "real" meaning here is that the person has a variable height (the property is not just *height* but *height at time t*). "The possibility of properties having properties is only contemplated when we have not fully specified (or properly understood) a property in the first place" [118].

Postulate 3: Humans conceive of properties of things in terms of the *attributes* of their conceptual models, and properties are known to humans only as *attributes* [118].

Postulate 4*: Every property in general can be represented by a prepositional (*attribute*) *function*: A: $T_1 \times \ldots \times T_n \times V_1 \times \ldots \times V_m \rightarrow$ Statement regarding A; and every specific property can be represented as an attribute function of the form: $A(t_1, \ldots, t_n, v_1, \ldots, v_m)$ where $t_i \in T_i$, $v_j \in V_j$, T_i $(i = 1, \ldots, n)$ represent the set of things and V_j $(j = 1, \ldots, m)$ represents the set of values.

For instance, the property "a student in a university" can be represented as S: $T_1 \times T_2 \times D \rightarrow P$, where T_1 is the set of students, T_2 is the set of universities, D is a set of dates, and P

is a set of statements of the form: "p (from set T_1) is a student of c (from set T_2) at d (from the set D)". We can represent this statement as an attribute function Student-of(p, c, d) meaning student p, is a student of university c, at date d.

Definition 1*: The *scope* of a property is the set of things that possess the property. That is, if θ is the set of things and P is the set of all properties, the *scope function* S is the mapping A: $P \rightarrow 2^{\theta}$.

Definition 2*: A subset of things, X, is called a *class* if and only if a property exists such that the subset is the scope of that property. That is, a subset X of the set of things θ is called a class of things iff $\exists p \in P$ such that $X = S(p) \in 2^{\theta}$.

Definition 3*: Let R be a set of properties. An *R-kind* is the intersection of all scopes of properties in R.

Definition 4: Any restriction on the set of properties of an R-kind is termed *law*.

Let R be a finite set of properties with possible laws on the values of its properties. Then, we term R-kind to be a *generic class*.

We can now map, a general class in conceptual model to a generic class in the ontological model.

Rule 5: A *class* in the conceptual model is representative of *generic class* in the ontological model.

The following definition formalizes the concept of inheritance.

Definition 4: A subset of things, X, is a *subclass* of another set of things, Y, if and only if X is a proper subset of Y. Conversely, Y is a *superclass* of X [118].

Corollary: If $S(p_1) = X$ and $S(p_2) = Y$, $p_1, p_2 \in P$, then X is a *subclass* of Y if and only if $S(p_1) \subset S(p_2)$ [118].

Let us consider a set of properties $P_1 = \{p_{11}, p_{12}, \dots, p_{1n}\}$ and a set of properties $P_2 = \{p_{21}, \dots, p_{2m}\}$. Let X be the class consisting of things $S(P_1)$ and Y be the class consisting of things $S(P_2)$. If $P_2 \subseteq P_1$, then we can represent Y as a subclass of X. This is generally true in conceptual modeling where subclasses contain specializing attributes and relationships (i.e. properties) than superclasses and superclasses generalize the concepts of subclasses.

Composition of things is outlined in Postulate 5.

Postulate 5*: Two things may associate to form another.

Based on this postulate, a thing is a *composite* if and only if at least two concrete things combine to form it. The reason for assembling these things to form composite things is the possibility to obtain *emergent* properties of the composite that is interesting, which is not a property of any of its component things [118].

Definition 5*: A property of a composite thing is *inherited* if and only if it is a property of any of its components; otherwise, it is *emergent*.

For instance, a computer is a composite thing since it is composed of main memory, processor, etc. Thus, the size of main memory is an inherited property because it is a property of main memory. However, the processing power of a computer is an emergent property because it is not a property of any of the individual components [118].

Ontology postulates that humans view an aggregation of things as a composite thing only if they are interested in at least one emergent characteristic of the composite:

Postulate 6*: Every composite thing possesses emergent properties.

In the next, section we discuss some concepts of classification techniques used in databases, AI and biology.

3.2.2.1.2 Classification Techniques

Classification techniques have been used in artificial intelligence as a means of knowledge representation and also in biological sciences to classify different types of plants and animals. An example of a classic AI technique that can be utilized for classification is the semantic network [87], originally developed as a way of representing human memory and language understanding.

The structure of a semantic network is shown graphically in terms of *nodes* and the *arcs* (links) connecting them. The nodes are generally used to represent physical objects, concepts, or situations. The links are used to express relationships. Two types of commonly used links are IS-A and A-KIND-OF, which are sometimes written as ISA and AKO [123]. IS-A means "an instance of" while A-KIND-OF means "specialization/generalization" relationship. Figure 11 depicts a semantic network with ISA and AKO links. For instance, in figure 11, node *University* represents the set of all universities and node $University_A$ represents an instance of university called $University_A$. We revisit this figure in subsequent discussions.

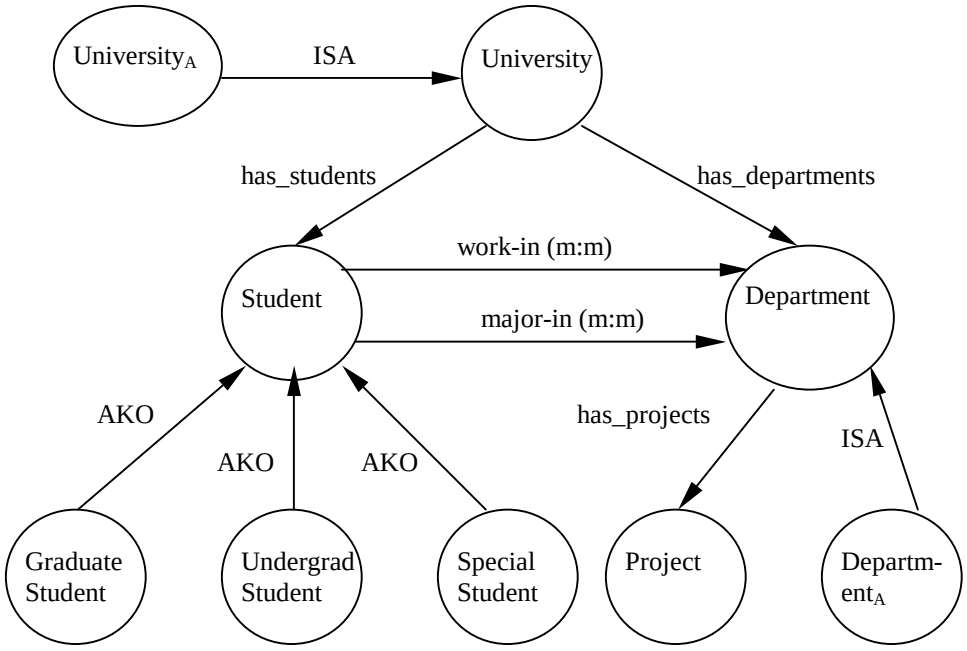


Figure 11. A Semantic Net with ISA and AKO Links

The extents of the different nodes (i.e. for non-instance nodes) are considered disjoint unless otherwise specified.

Having discussed some of the relevant work that is useful to our approach, let us now discuss the methodology for identifying semantically related items of different schemas. Identifying semantic relations between classes of different database schema is the basis for successful resolution of semantic heterogeneity during integrating. We focus on this issue in the next section.

3.2.2.2 Methodology

A significant impediment to identifying semantic relations in a definitive manner is the difficulty in obtaining knowledge of the extent of classes of database schema. Usually, the intended extents are in the original database designer's mind, represented partially in conceptual models and to a lesser extent in the schemas. However, in most application domains the only high-level knowledge available to the integrator is schema-level descriptions of legacy databases.

It is obvious that without the appropriate knowledge of the extents of classes, it is impossible to make a reliable decision as to the types of semantic relations that are present. Hence, our approach takes a stepwise process to obtain this information.

Before discussion of the steps of the proposed methodology, we present some obvious techniques for identifying semantic related classes and discuss why these techniques are infeasible. Discussing and comparing these techniques provides us with insights as to some problems related to semantic heterogeneity resolution and also provide justifications for the methodology we propose.

Technique 1: A brute force and obvious algorithm is to determine if two classes are SEM_EQ, SEM_SUB, SEM_OVER or SEM_DIS is to compare the extents for each class. This is practically impossible and theoretically incorrect. It is practically impossible because the sheer number of possible comparisons. It is theoretically incorrect, because we need to check for every database instance at time t and thus for instance, extents of class A and extents of class B being equal at the current moment do not necessarily mean that they will do so in future. Hence, this algorithm is impractical and incorrect.

Technique 2: Another approach would be to enumerate all the possible properties (discussed in section 2.1) for each class in the schema. Then comparison of matching properties of different classes enables to determine the semantic relationship between classes of different database schema. In terms of efficiency compared to technique 1, this approach is significantly efficient and may be practically possible for a small number of schemas. However, a main obstacle to this approach is the inability to verify if the set of all possible properties are generated for a particular class or whether to determine if this set is finite at all. Another technical hurdle is to find the matching properties in different classes.

The following sections describe our approach to identifying semantic relations.

3.2.2.2.1 Step 1: Conversion to Sem-ODM

The first step of an integration process is to reduce the heterogeneities that may occur due to different data models in which the schemas are represented. A generally accepted framework is to transform these schemas of different data models to a canonical data model (CDM) [105]. In the architecture presented in chapter 2, we convert the schemas to Sem-ODM through wrappers ([74], [96], [97]) for non-Sem-ODB data sources. The algorithm for automated translation from relational schema to semantic schema (by the Schema Transformer module of SemWrap [96]) is given below:

Algorithm:

- For each *table* in the relational schema
    - Create a *category* in the semantic schema with same name as in *table*.
    - For each *field* in the table
        - Create an *attribute* corresponding in the respective category with same name of field
- For each functional dependency (i.e. foreign key, primary key relationship) **except** when the primary key is composite (i.e. multiple fields make up the primary key field) and there are more than one functional dependency from foreign table to primary table.
    - Create a *relation* with cardinality m:1 from category corresponding to foreign table to category corresponding to primary table. The name of relation is "*DomainCategoryName*"_"*RangeCategoryName*"_# where # represents a number which is unique for relations between Domain and Range categories

Note that we avoided creating relations when there exists ambiguity in identifying the participating fields of primary and foreign tables due to composite primary key fields and multiple functional dependencies between the two tables.

3.2.2.2.2 Step 2: Obtaining Property Functions

The next step in the methodology is to obtain context information incorporated into the component schemas and have a common framework for sharing the semantic meaning of schemas. This step focuses on obtaining context information. The following example illustrates context information in detail:

Example 7: Let us consider a database containing information on students and major departments in a university. Figure 12 illustrates such a schema in Sem-ODM.

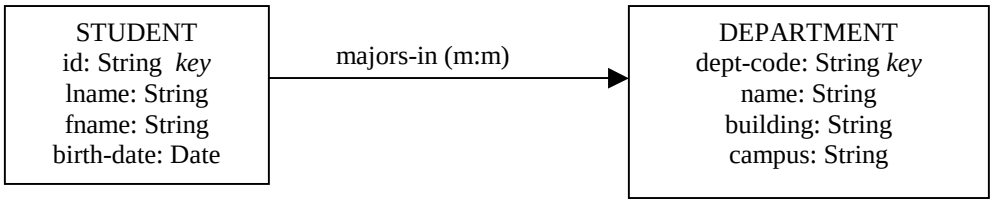


Figure 12. Schema in Sem-ODM

An important aspect that can be noticed from the schema is the lack of context information. Context information is the knowledge within the application domain that is not generally explicitly stated. However, when we bring out these schemas into the domains of other application areas for schema integration, specifying the context information is important. Context information is a generalized property which is common to every class within a database schema. For instance, schema in figure 12 describe a particular university A. This context information is not shown in the schema

but when comparing this schema with other schemas in a multidatabase environment this context information is significant.

We now extent some of the definitions from Bunge's ontological model in order to gain a formalism for defining the semantics of schema constructs.

Postulate 7: For every class C in conceptual model, there exists an abstract concept, C′ (called *general class of C*), such that EXT(C) $\subseteq$ EXT(C′)

Hence for every class, C, in Sem-ODM, there exists a general class for C′ that consists of at least the set of items represented by C.

Postulate 8: The extent of every class C can be defined unambiguously using a property function P. P (M, {f_1, … , f_n}) → Extent of C, where M (called *primary mapping*) is a mapping from C to one of its general classes C′, and f_i: C′ × A (i = 1, … , n) where A is an abstract concept.

Ontologically, this means that S(M) ∩ S(f_1) ∩ S(f_2) ∩ … ∩ S(f_n) = EXT(C) where f_i (i = 1, … , n) represent a property.

An example of a property function is shown below:

Example 8: Let us consider the schema in example 2. We have the following property functions for categories STUDENT and DEPARTMENT.

1. STUDENT:

   Property Function: (M, F) such that

   M : STUDENT → S

F : {Is-student-of($S \times U_A$), At-time($S \times T_S$)}

where STUDENT : { set of current students in University$_A$ }

S : {set of students of all universities at any instance of time}

U_A : {University$_A$}

T_S : {Current}

Therefore (M, F) represents "Set of current (T_S) students (S) in University$_A$ (U_A)". (M, F) unambiguously defines the EXT(STUDENT). That is, *S*(S) ∩ *S*(Is-student-of) ∩ *S*(At-time) = EXT(STUDENT) where *S* is the scope function.

2. DEPARTMENT:

Property Function: (M, F) such that

M : DEPARTMENT → D

F : {Department-of($D \times U_A$), At-time($D \times T_S$)}

where DEPARTMENT: {Set of departments in University$_A$}

D: {set of all departments of all universities at any instance of time}

U_A: {University$_A$}

T_S: {Current}

Therefore (M,F) represents "Set of current (T_S) departments (D) of University$_A$ (U_A). (M, F) unambiguously defines EXT(DEPARTMENT). That is, *S*(D) ∩ *S*(Department-of) ∩ *S*(At-time) = EXT(DEPARTMENT) where *S* is the scope function.

The next step in the methodology is to map the property function to a shared ontology representing the application domain, which is discussed in the following section.

3.2.2.2.2 Step 3: Mapping To Shared Ontology

This step tries to achieve a common language for sharing semantics between a set of component schemas so as to determine the semantically related constructs of the schema. The common medium for exchanging semantics is the use of a shared ontology represented by a semantic network (example shown in section 3.2.2.1.2). Let us now look at some previous work that provides justification for our claim (i.e. it is possible to build an ontology using a semantic network for a general application domain for which a database schema is designed). Previous work, such as in [101], uses a shared ontology for semantic interoperability. Thus, this is shown to be feasible. Our assumption that general conceptual models can be built for a general application domain is justified by previously demonstrated work such as in [111] which discusses a tool to automatically design schemas based user requirements. In [111], generalized schemas are stored in the Application Domain Base (ADB) and learning takes place when schemas from Application Case Base (ACB) are moved to the Application Domain Base. Empirical testing of the system provides favorable results. In [43], we have seen the use of ontology to describe information on Web pages. We postulate that similar techniques can be used to generate classification graphs for different domains. We feel that semantic network is a powerful expressive technique for representing shared ontology. This is justified by previous work such as [14] which argues that semantic network is powerful than OO

models. Also, we have seen in [112], the design of ontologies for general applications based on semantic networks.

This section describes a technique to map the property functions of the classes of each schema to shared ontologies of the application domains. These mappings enable to determine semantic relations between classes of different schemas (see step 4 below). Our technique is best illustrated by an example.

Example 9: Let us consider the schema in figure 12. The property functions for categories STUDENT and DEPARTMENT are given in example 8. Let us say that schema in figure 12 corresponds to DB_1 and the following schema (i.e. figure 13) corresponds to DB_2.

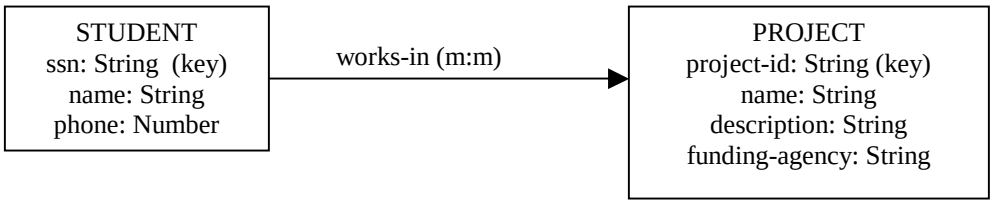


Figure 13. Schema for Database DB_2 in Example 1

The property functions for DB_2 are as follows:

STUDENT:

Property Function: (M, F) such that

$M : STUDENT \rightarrow S$

$F : \{ \text{Is-student-of}(S \times U_A), \text{Work-in}(S \times D_A), \text{At-time}(S \times T_S)\}$

where STUDENT : {set of students in $University_A$ who work in $Department_A$ of $University_A$}

S : {set of students of all universities at any instance of time}

U_A : {$University_A$}

D_A : {$Department_A$ of $University_A$}

T_S : {Current}

Therefore (M, F) represents "Set of current (T_S) students (S) in $University_A$ who work in $Department_A$ (D_A) of $University_A$".

PROJECT:

Property Function: (M, F) such that

M : PROJECT → P

F : {Project-of($P \times D_A$), At-time($P \times T_S$)}

where PROJECT : {set of current projects in $Department_A$ of $University_A$}

P : {set of all projects in all departments of all universities at any instance of time}

D_A : {$Department_A$ of $University_A$}

T_S : {Current}

Therefore (M, F) represents "Set of current (T_S) projects (P) in $Department_A$ of $University_A$ (D_A)".

We will map the property functions to semantic network given in figure 11. Note that node *Time* has been omitted in the semantic network. All nodes of the semantic network have relationship to *Time* node called *At-time*. The property *At-time* of every class is mapped to this relationship and is omitted from discussion below. T_S: {Current} is mapped to a node called *Current* which is an instance of *Time* (related by ISA).

	Matching property		Matched node of Semantic Net
DB_1.STUDENT	S	→	Student
DB_1.STUDENT	U_A	→	$University_A$
DB_1.STUDENT	Is-student-of	→	has-students
DB_1.DEPARTMENT	D	→	Department
DB_1.DEPARTMENT	U_A	→	$University_A$
DB_1.DEPARTMENT	Department-of	→	has-departments
DB_2.STUDENT	S	→	Student
DB_2.STUDENT	U_A	→	$University_A$
DB_2.STUDENT	D_A	→	$Department_A$
DB_2.STUDENT	Is-student-of	→	has-students
DB_2.STUDENT	works-in	→	work-in
DB_2.PROJECT	P	→	Project
DB_2.PROJECT	D_A	→	$Department_A$
DB_2.PROJECT	Project-of	→	has-projects

Having mapped to a common ontology presented in figure 11, let us now consider the derivation of semantic relations, which is the focus in the next section.

3.2.2.2.2 Step 4: Discovering Semantic Relations

This section discusses some rules that utilize the mapping information illustrated above to derive semantic relations. Application of these rules produce semantic relations.

Let A be a construct in schema 1 while B is a construct of schema 2.

Rule I: If the *primary mapping of class A* map to the same node as *primary mapping of class B* and all other properties of classes A and B map to the same set of nodes and links in the semantic network, then A SEM_EQ B (i.e. EXT(A) = EXT(B) at any given time t).

Proof Sketch: In postulate 8, we claimed that a concept is unambiguously defined using a property function. In step 3, we mapped the property function onto a shared ontology. Thus, if the mappings of two constructs correspond, then these concepts are describing the same concept. Hence, the extents are identical at any given instance.

Rule II: If the *primary mapping of class A* map to node in the semantic network which is disjoint with node in the semantic network to which the *primary mapping of class B* maps to, then A SEM_DIS B (i.e. $EXT(A) \cap EXT(B) = \varnothing$ for all instances of time t)

Proof Sketch: Let A′ be the primary mapping of class A and B′ be the primary mapping of B. Since A′ and B′ are disjoint $EXT(A') \cap EXT(B') = \varnothing$ for all instances of time t. By postulate 7, $EXT(A) \subseteq EXT(A')$ and $EXT(B) \subseteq EXT(B')$ for all instances of time t. Hence, $EXT(A) \cap EXT(B) = \varnothing$ for all instances of time t.

For instance, if we consider node “Department” is disjoint with node “Student” in figure 1, then we can conclude that DB2.STUDENT SEM_DIS DB1.DEPARTMENT by Rule II.

Rule III: Let $Nodes_A$ represent the set of nodes and links in the semantic network for which there is a mapping from a property of class A. Similarly, let $Nodes_B$ represent the set of nodes and links in the semantic network for which there is a mapping from class B.

If either *the primary mapping of class A* map to the same node as *primary mapping of class B* or if there is a path from *the primary mapping of class B* to *primary mapping of class A* using only AKO links and $Nodes_A \subseteq Nodes_B$, then B SEM_SUB A.

Proof Sketch: Let A′ be the primary mapping of class A and B′ be the primary mapping of B. Let $Nodes_A = \{n_1, n_2, \ldots, n_k\}$ and $Nodes_B = \{n_1, n_2, \ldots, n_k, \ldots, n_n\}$. Note that AKO represents a "specialization/generalization" relationship. Thus, if there exists a path from B′ to A′, then $EXT(B') \subseteq EXT(A')$. By postulate 8, $EXT(A) = S(A') \cap S(n_1) \cap \ldots \cap S(n_k)$ and $EXT(B) = S(B') \cap S(n_1) \cap \ldots \cap S(n_k) \cap S(n_{k+1}) \cap \ldots \cap S(n_n)$. Hence, $EXT(B) \subseteq EXT(A)$.

For instance, in example 4, $Nodes_{DB1.STUDENT}$ = {Student, $University_A$, has-students} and $Nodes_{DB2.STUDENT}$ = {Student, $University_A$, has-students, $Department_A$, work-in}. Primary mapping of classes DB_1.STUDENT and DB_2.STUDENT is "Student" node. Hence by Rule III, DB_2.STUDENT SEM_SUB DB_1.STUDENT (i.e. EXT(DB2.STUDENT) $\subseteq$ EXT(DB1.STUDENT)). This is true since DB_1.STUDENT represents students of university A, while DB_2.STUDENT represents student of university A who work for department A.

The rules presented by no means provide set of all semantic relations between constructs of database schema. However, these rules combined with rules presented in section 3.2.2.1 provide a basis for automated discovery of many semantic rules. Usually step 1 is automated with the domain expert customizing the automatically generated schemas. Steps 2-3 are performed at the component site with the interaction from domain expert. Step 4, combined with rules discussed in section 3.2.2.1 are executed at the global site by the Integrator and Knowledge Reconciliator (see chapter 2) automatically. This module further interacts with global DBA to identify further semantic knowledge and to create global views.

3.3 Summary

In this chapter, we introduced the problem of semantic heterogeneity. We discussed some of the existing approaches that address this issue. Almost all of the approaches use heuristic means to acquire knowledge and resolve semantic heterogeneity. These approaches can result in incorrect results during integration (such as incomplete answers to queries). We incorporate many techniques mentioned in previous approaches and propose a methodology for semantic heterogeneity resolution. Our methodology is based on the acquisition of semantic knowledge for resolving semantic heterogeneity. The basis for acquiring semantic knowledge is determining semantic relations between entities of different component schemas. The completeness and correctness of these relations are outlined. We extend the semantic knowledge by acquiring object equivalences and boundary conditions for certain types of semantically

related entities. This methodology resolves semantic heterogeneity and provides correct, complete and unambiguous integration (will not result in incomplete query results). The acquired semantic knowledge can be exploited during the creation of global schemas (see chapter 4) and for optimizing queries posed on the global schema (see chapter 5). An automated methodology for identifying semantically related entities is highly desirable. We investigate into ontological research and knowledge representation techniques in designing a semi-automated step-wise methodology for identifying semantic relations. Investigating into techniques for easy specification of property functions and discovering rules for identifying semantic relations are future research directions we consider for improving our methodology.

4. SCHEMATIC HETEROGENEITY RESOLUTION

As mentioned in chapter 1, the ideal situation is to provide an interface similar to a centralized database system to the multidatabase users. This requires the definition of global schemas/views from a set of component database schemas. An issue that needs to be addressed when creating a global schema/view is the resolution schema-level conflicts (known as schematic heterogeneity). Schematic heterogeneity occurs when semantically related (in our case SEM_EQ, SEM_SUB, SEM_OVER) schema constructs are represented differently in different component schemas. For instance, the address of a person may be represented by a *category* in one schema and as an *attribute* in another. The price of an item may be represented in 'US Dollars' in one schema while the price of the same item may be represented in 'British Pounds' in another schema. During global schema/view definition, a single representation schema for data items must be decided and a conversion from different representations of the component schemas to the representation of the global schema must be defined. This chapter focuses on the resolution of schematic heterogeneity and global view definition including knowledge management issues for database integration.

In section 4.1, a brief discussion into related work regarding schema-level heterogeneity resolution and database integration is presented. Section 4.2 discusses our approach to global schema definition and schema integration. Benefits of our approach to database integration when compared with previous approaches are outlined in section 4.3.

4.1 Related Work

Early research into multidatabase systems has focused on the schema-level heterogeneities. A plethora of approaches for resolving schema level conflicts are presented in literature ([9], [17], [31], [33], [53], [54], [55], [59], [83], [105] and others). Many broad classes of schema-level conflicts have been identified and resolved. Almost all of these methodologies have focused on the relational and object-oriented data models. In [55], an exhaustive enumeration of schematic conflict types and their resolutions for integrating relational and object-oriented schemas has been presented. However, we have not found any previous work regarding schema-level heterogeneity resolution taking Semantic Data Models into consideration. We focus on this issue in the subsequent sections.

4.2 Our Work

This section outlines our schematic heterogeneity resolution including knowledge management for database integration as a whole. The organization of this section is as follows. Firstly, a language, called SemOSQL/M, for defining global Sem-ODM views over component Sem-ODM schemas is introduced. In section 4.2.2, the use of SemOSQL/M to resolve each type of schema-level conflict resolution during global view definition is illustrated. A desirable and advantageous goal is to store and manage the semantic knowledge and schema-resolution knowledge in a centralized manner for global schema definition and query processing. In section 4.2.4.1, schemas designed for Knowledge Bases to store and manage such information are presented. Next, a tool that assists the process of global view definition using the existing knowledge in the

Knowledge Base is presented. Finally, section 4.3 discusses the benefits of our approach to existing approaches.

4.2.1 SemOSQL/M

This section introduces SemOSQL/M which is a language used in the creation of SemODM global views over a set of component SemODM schemas. SemOSQL/M is similar to SQL but extended in certain aspects to incorporate features of multidatabase systems.

In SemOSQL/M, the definition of a category in the global schema has two components. The first component is the signature of the global category. The second component is a list of SQL like statements that specify a methodology to derive information for the categories from component schemas. The second component includes one query for each of the component database (CDB) entities being integrated.

Following is the syntax for the category definition:

CREATE CATEGORY *category_name*
[SUPERCATEGORY super_category {,super_category}*]
attr_def_list
AS SELECT *selection_list*
FROM *entity_spec_list*
[WHERE *search_conditions*]
[GROUP BY *selection_list*]
[HAVING *search_conditions*],

……

SELECT *selection_list*

FROM *entity_spec_list*

[WHERE *search_conditions*]

[GROUP BY *selection_list*]

[HAVING *search_conditions*],

entity_spec_list ::= *cdb_entity_name* [*variable*]

{, *cdb_entity_name* [*variable*] }

cdb_entity_name ::= [*cdb_name.*]*entity_name*

The *attr_def_list* consists of attributes and their domains, along with methods and relations. A comma separates each component query. The *selection_list* is an extension to SQL to handle schematic and data heterogeneities. The *entity_spec_list* determines the various entities from different CDBs against which the query is to be evaluated. The *search_conditions* are identical to those in SQL. The *super_category* defines name of the super category in the inheritance hierarchy.

The syntax for the definition of a relation in SemOSQL/M is as follows:

CREATE RELATION *relation-name*

(DOMAIN *domain-category* RANGE *range-category*

[CARD *cardinality*][TOTAL])

AS FROM *selection-list*

WHERE *join-conditions*

…….

FROM *selection-list*

WHERE *join-conditions*

The *relation-name* contains the name of the relation in the global schema. The *domain-category* denotes the category name of the domain of the relation. The *range-category* denotes the category name of the range of the relation. The DBA can specify the *freest* (*freest* is described below) cardinality in *cardinality* as *m:1*, *1:m, 1:1* or *m:m*. If the cardinality is not specified, then it is assumed to the default (*m:m*). The totality of a relation is specified by TOTAL. If not specified, it is assumed to be not total by default. The *selection-list*, in the case of a semantic schema as the CDB schema, will contain a relation on the schema. In the case of relation spanning across component schemas, *selection-list* will contain two category names in different schema with the *join-conditions* specifying a condition to satisfy in order for the objects to be considered related in the domain and range categories.

The schema-level conflicts and their resolutions specified using SemOSQL/M statements are illustrated in the next section.

4.2.2 Schema-level Conflicts and Resolutions

A number of efforts to resolve schema-level conflicts in object-oriented schemas and relational schemas have been discussed previously in literature. However, schema conflicts among a set of Sem-ODM schemas are lacking. In this section, we present resolutions for different types of schema-level conflict among Sem-ODM schemas. Note

that in our presentation, there is an overlap of certain ideas with previous work (such as in [55]), however we included them for clarity and completeness of our discussion.

This section describes the different types of schema conflicts and their resolutions. Each conflict type will be illustrated with an example and its resolution will be specified using SemOSQL/M. Note that in all of the examples, we assume that the categories represented are semantically equivalent (i.e. SEM_EQ) and we create a semantically equivalent global category for the presented component schemas' categories unless stated otherwise. This assumption simplifies our presentation and does not restrict in any aspect. The ideas presented can be easily extended to SEM_SUB and SEM_OVER without any loss of generality. The only difference is that we may need to specify the boundary conditions in the WHERE clause for category definitions appropriately.

4.2.2.1 Naming Conflicts

Conflict: Semantically equivalent categories and attributes may have different names in the component database schema

Resolution: Renaming entities and attributes in the global schema and mapping them to their corresponding entities and attributes in CDBs.

Example 10: Let us consider the following schemas:

Figure 14. Schemas with Naming Conflicts

The SemOSQL/M statement looks as follows:

```
CREATE CATEGORY STUDENT
        (SocialSecurity:INTEGER, address:STRING)
AS      SELECT ssn, address
        FROM   cdb1.STUDENT
        SELECT SocialSec, address
        FROM   cdb2.GRAD_STUDENT
```

The extension of category STUDENT in the global schema can be obtained by visiting either categories, cdb1.STUDENT or cdb2.GRAD_STUDENT (since STUDENT, cdb1.STUDENT and cdb2.GRAD_STUDENT are related by SEM_EQ relation). However, our system encourages the global DBA to define all the semantically related entities of data sources in deriving the global construct as this information can be used in deriving intelligent query optimization strategies (discussed in chapter 5).

4.2.2.2 Data Conflicts

Conflict: Data conflicts occur, when semantically equivalent data are represented differently.

Resolution: Homogenizing the representations. In the global schema, the data are represented in one form (same expression, same unit, same precision). This may lead to loss of accuracy/precision. For instance, converting marks from 1-100 scale to grade 'A','B','C','D' and 'F'. Homogenizing representations are allowed in SemOSQL/M using

arithmetic operators and DBA defined functions (Note that DBA defined functions are preceded by “dba.” string).

Example 11: Let us consider the following schema of two semantically related (i.e. SEM_EQ) entities:

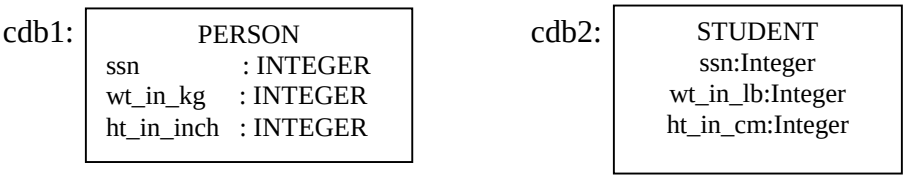


Figure 15. Schemas with Data Conflicts

Let us assume that weight is specified in kg (by *wt_in_kg* attribute) in cdb1.PERSON and height is specified in inches (by *ht_in_inch* attribute). Also, weight is specified in lbs (by *wt_in_lb* attribute) and height in centimeters (by *ht_in_cm* attribute).

Following is a SemOSQL/M statement that resolves the data conflicts:

```
CREATE CATEGORY PERSON
        (ssn:INTEGER, wt_in_lb:INTEGER, ht_in_in:INTEGER)
AS      SELECT ssn, dba.change_lb(wt_in_kg), ht_in_inch
        FROM   cdb1.PERSON
        SELECT ssn, wt_in_lb:INTEGER, ht_in_cm/2.54
        FROM   cdb2.STUDENT
```

In the above definition, to convert to lbs in the first SELECT statement, we use a DBA defined function (i.e. dba.change_lb()). In the second SELECT statement, the division (/) operator is used. Note that there is a loss of precision when converting from cm to inch.

An important aspect to note that is not explicitly stated is that, in defining conversion functions to resolve conflicts, we also need to specify reverse functions to perform the opposite conversion. For instance, for the *dba.change_lb(),* we need to specify a function *dba.change_kg()* which takes as input a field in *lbs* and outputs the value in *kg*. This function is used during query processing to translate the queries to the format of component database. For instance, let consider the following query:

```
SELECT      ssn
FROM        PERSON
WHERE       wt_in_lb > 100
```

In order to translate this query into a query of cdb1, we need to first convert value '100' mentioned in the WHERE clause of the global query to units in kgs for which the query processor utilizes function *dba.change_kg().*

4.2.2.3 Attribute Type Conflicts

Conflict: Semantically equivalent attributes may have different types.

Resolution: Type coercion. Most of the types may be coerced. Type coercion is allowed in SemOSQL/M with the keyword AS or by DBA defined functions.

Example 12: Assume that for above example attribute *ssn* is represented as a STRING in cdb1.

```
CREATE CATEGORY PERSON
        (ssn:INTEGER, wt_in_lb:INTEGER, ht_in_in:INTEGER)
AS      SELECT ssn AS INTEGER, dba.change_lb(wt_in_kg),
```

ht_in_inch

FROM cdb1.PERSON

SELECT ssn, wt_in_lb:INTEGER, ht_in_cm/2.54

FROM cdb2.STUDENT

The above SemOSQL/M statement using AS keyword to convert the type from STRING to INTEGER.

4.2.2.4 Attribute Granularity Conflicts

Conflict: A single attribute in a CDB is equivalent to a group of attributes in another CDB.

Resolution: There are two alternatives for this conflict type. (i.) Concatenate the attributes and represent it as a single attribute in the global entity. (ii.) Simplify the complex attribute to multiple attributes, and represent it as a group of attributes. In SemOSLQ/M, square brackets ([]) are used for attribute concatenation of strings; arithmetic operators for defined operand types and DBA defined methods for attribute simplification.

Example 13: In the following schemas, cdb1.PERSON.name contains both last name and first name.

cdb1: PERSON(name:STRING, ssn:INTEGER)

cdb2: EMPLOYEE(lastname:STRING, firstname:STRING,

ssn:INTEGER)

Alternative 1: Concatenating the attributes for strings.

CREATE CATEGORY PERSON

(ssn:INTEGER, fullname:STRING)

AS SELECT ssn, name

FROM cdb1.PERSON

SELECT ssn, [lastname,firstname]

FROM cdb2.EMPLOYEE

Alternative 2: Simplifying the attributes

CREATE CATEGORY PERSON

(ssn:INTEGER, firstname:STRING lastname:STRING)

AS SELECT ssn, dba.extractFirst(name),

dba.extractlast(name)

FROM cdb1.PERSON

SELECT ssn, lastname, firstname

FROM cdb2.EMPLOYEE

4.2.2.5 Missing Attribute Conflicts

Conflict: There may be attributes missing in the entities of CDB schemas.

Resolution: There are 3 ways to resolve this type of conflict. (i) Coerce non-existent attributes with NULL ("values not known"). Note that having a NULL value does not mean that every object in the CDB entity will obtain a NULL value for the missing attribute. If there exists a non-NULL value for an attribute of a semantically equivalent object from a different CDB entity, these values will be replaced instead of NULL values. (ii.) Exclude the extra attributes from the selection list of the component query for the

CDB entity, which has more attributes than other CDB entities with which it is being integrated. (iii) Model the entity with fewer attributes as the super-category of the others, provided that all entities being integrated are related by the inclusion relationship (that is, every object in the subcategory must also be an object in the super category).

Examples of the first and third resolutions are given below.

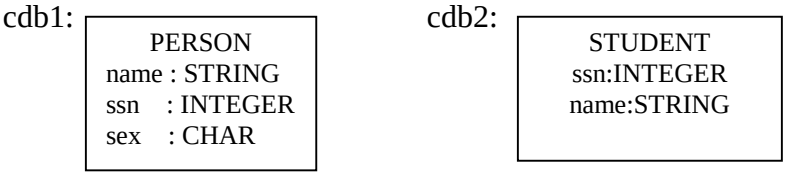


Figure 16. Schemas with Missing Attribute Conflicts

Alternative 1: Using NULL values.

```
CREATE CATEGORY PERSON
        (name:STRING, ssn:INTEGER, sex:char)
AS      SELECT name, ssn, sex
        FROM   cdb1.PERSON
        SELECT name, ssn, NULL
        FROM   cdb2.STUDENT
```

Alternative 3: Using super/sub categories.

```
CREATE CATEGORY PERSON
        (name:STRING, ssn:INTEGER)
AS      SELECT name, ssn
        FROM   cdb1.PERSON
```

SELECT name, ssn

FROM cdb2.STUDENT

CREATE CATEGORY PERSON_1 SUBCATEGORY PERSON

(sex:CHAR)

AS SELECT sex

FROM cdb1.PERSON

4.2.2.6 Missing Attributes with Implicit Values

Conflict: This type of conflict has entities with missing attributes which are implicit (such as context information); hence not included in the CDB schema.

Resolution: An expression *cdb_attr_name = value* is included as an element of the *selection_list* of a component query in the definition of the global category, where cdb_attr_name is the name of the missing attribute in a CDB entity, and its value is the implicit default value.

Example 14: Let us look at the following schemas. In this case, cdb1.STUDENT, student_type denotes whether a given student is an undergraduate or graduate student.

Figure 17. Schemas having Missing Attributes with Implicit Value Conflicts

```
CREATE CATEGORY GRADUATE_STUDENT
                (ssn:INTEGER, name:STRING, student_type:CHAR)
    AS      SELECT ssn, name, student_type
            FROM   cdb1.GRAD
            SELECT ssn, name, student_type = 'G'
            FROM   cdb2.GRAD_STUDENT
```

4.2.2.7 Basic Relations

Conflict: Importing relations, between entities of component database schemas, to the global schema.

Resolution: We can import relations between entities of same component database and represent them as a relation in the global schema. The totality of a relation and cardinality conflicts are resolved as follows.

- *Total*: The relation defined in the global schema is total iff every sub-relation it is based on is total, otherwise it is not total.
- *Cardinality*: The cardinality is determined as the *freest* possible case of the sub-relations. By freest, we mean that we take the largest value for the right hand and left hand of the cardinality from the set of cardinalities of the sub-relations. For instance, if we have the following cardinalities (m:1) and (1:1) then freest cardinality is (m:1) (that is m is largest on r.h.s. and 1 in l.h.s.).

Example 15: Following are component schemas of cdb1 and cdb2.

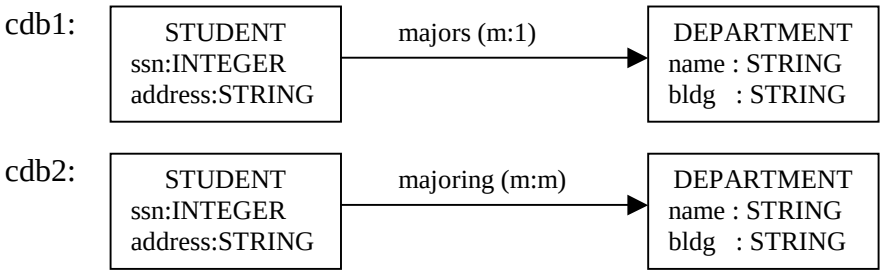


Figure 18. Schemas with Basic Relations

Let us assume that the following categories have already been created in the global schema.

```
CREATE CATEGROY STUDENT
        (ssn:INTEGER, address:STRING)
AS      SELECT ssn, name
        FROM   cdb1.STUDENT
        SELECT ssn, name
        FROM   cdb2.STUDENT

CREATE CATEGROY DEPARTMENT
        (name:STRING, bldg:STRING)
AS      SELECT name, bldg
        FROM   cdb1.DEPARTMENT
        SELECT name, bldg
        FROM   cdb2.DEPARTMENT
```

Now let us consider how the "majors" relation is defined between category STUDENT and DEPARTMENT in the global schema.

CREATE RELATION majors

(DOMAIN STUDENT RANGE DEPARTMENT)

(CARD m:m)

AS FROM cdb1.STUDENT.majors

FROM cdb2.majoring

4.2.2.8 Composite Relations

Conflict: Importing a composition of relations, between entities of component database schemas, to the global schema.

Resolution: We can import a composition of relations between entities of same component database and represent them as a relation in the global schema. The totality of a relation and cardinality conflicts are resolved as follows:

- *Total*: The relation defined in the global schema is total *iff* the direction of the relation is the same as the direction of every sub-relation in the composite relation and each sub-relation is total.
- *Cardinality*: The highest value from the left hand side of cardinality of every sub-relation is taken for the left-hand value of the cardinality for the composite relation. The highest value from the right-hand value is taken for the right-hand value of the cardinality for the composite relation assuming the same direction of composite relation as the global relation. For instance, if we have the following cardinalities (m:1) and (1:m) of two sub-relations which make a composite relation then we take cardinality as (m:m) for the cardinality of composite relation. Next we find the freest possible relation from all local relations as the cardinality of the global relation.

Example 16: Following is the schema of component database cdb1.

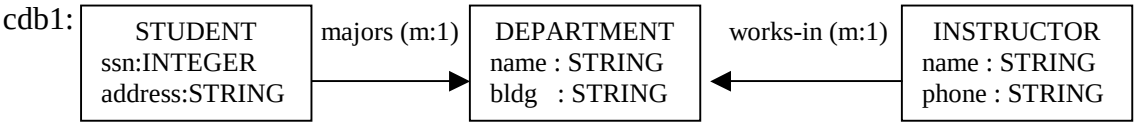


Figure 19. Schemas with Composite Relations

Let us assume that the following categories have already been created in the global schema.

```
CREATE CATEGROY STUDENT
        (ssn:INTEGER, address:STRING)
AS      SELECT ssn, name
        FROM   cdb1.STUDENT

CREATE CATEGROY INSTRUCTOR
        (name:STRING, phone:STRING)
AS      SELECT name, phone
        FROM   cdb1.INSTRUCTOR
```

Now let us consider how the "major-dept-inst" relation is defined between category STUDENT and INSTRUCTOR in the global schema.

```
CREATE RELATION major-dept-inst
        (DOMAIN STUDENT RANGE INSTRUCTOR)
        (CARD m:m)
AS      FROM  cdb1.STUDENT.STUDENT__majors__works_in___
```

Note that we have used a similar syntax as Semantic SQL where two underscores (i.e. '__') means direct relation (non-inverse) while three underscores (i.e. '___') means inverse relation. It is also important to point out that the cardinality of 'major-dept-inst' is (m:m) since 'majors__' cardinality is (m:1) and 'works-in___' cardinality is (1:m).

4.2.2.9 Inter-schema Relations

Conflict: Defining relations between entities in different component database schemas.

Resolution: When defining a relation between entities in different component databases, the relation cannot be declared as total. This is because the CDBs are autonomous and we cannot guarantee the existence of a RANGE object. Also, the relation has cardinality (m:m) due to the same reason.

Example 17: Let us look at the following schemas.

Figure 20. Schemas with Inter-schema Relations

Let us assume that the following categories have already been defined.

```
CREATE CATEGROY STUDENT
        (name:STRING)
AS      SELECT name
        FROM   cdb1.STUDENT

CREATE CATEGROY DEPARTMENT
```

(name:STRING, bldg:STRING)

AS SELECT name, bldg

FROM cdb2.DEPARTMENT

Now let us consider the inter-schema relation *majors* from category STUDENT to DEPARTMENT.

CREATE RELATION majors

(DOMAIN STUDENT RANGE DEPARTMENT)

AS FROM cdb1.STUDENT s, cdb2.DEPARTMENT d

WHERE s.major-dept = d.name

Note that, a join-condition (i.e. major-dept = name) needed to be specified for an inter-schema relation. A similar methodology can be used to create a relation between categories of the same CDB schemas where a relation between the categories does not exist (called join-relation conflict).

4.2.2.10 Category Inclusion Conflicts

Conflict: A category in a CDB is semantically subset with a category in another component database.

Resolution: Two categories are defined in the global schema with one category being a subcategory of the other. The DBA must ensure that the entity inclusion relationship is preserved (that is every object is in the subcategory is also a member of the super-category).

Example 18: Let us look at the following schemas.

Figure 21. Schemas with Category Inclusion Conflicts

Let us assume that cdb2.GRAD_STUDENT is SEM_SUBSET of cdb1.STUDENT. Following are the SemOSQL/M statements that define these categories in the global schema.

```
CREATE CATEGORY STUDENT
        (ssn:INTEGER, address:STRING)
AS      SELECT ssn, address
        FROM   cdb1.STUDENT
        SELECT ssn, address
        FROM   cdb2.GRAD_STUDENT

CREATE GRAD_STUDENT SUBCATEGORY STUDENT
        (major:STRING)
AS      SELECT major
        FROM   cdb2.GRAD_STUDENT
```

4.2.2.11 Attribute Inclusion Conflicts

Conflict: There is an inclusion relationship between attributes in semantically equivalent entities.

Resolution: We use inheritance to resolve the conflict.

Example 19: Let us look at the following schema.

cdb1: PEOPLE(name:STRING, age:INTEGER, childname:STRING)

cdb2: PERSON(name:STRING, age:INTEGER, son_name:STRING)

The following SemOSQL/M statements define the inheritance hierarchy:

```
CREATE CATEGORY PARENTS
        (name:STRING, age:INTEGER, child:STRING)
AS      SELECT name, age, childname
        FROM   cdb1.PEOPLE
        SELECT name, age, son_name
        FROM   cdb2.PERSON

CREATE CATEGORY PARENTS_OF_MEN SUBCATEGORY PARENTS ()
AS   SELECT *
        FROM   cdb2.PERSON
```

4.2.2.12 Category-versus-Attribute Conflicts

Conflict: These conflicts arise when a concept is represented as a category in one CDB, however as a set of attributes (belonging to a semantically equivalent entity) in another CDB.

Resolution: The category may be split into multiple parts, or two categories (or parts thereof) may be integrated into one.

Example 20: Let us look at the following schemas.

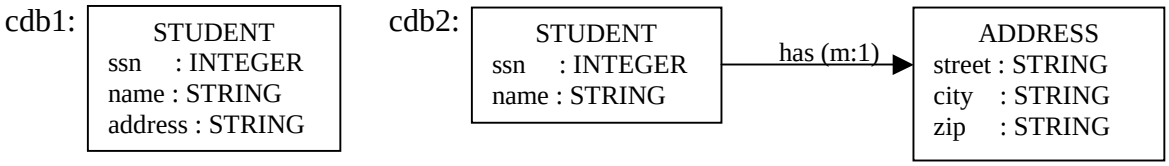


Figure 22. Schemas with Category-versus-Attribute Conflicts

Following are the SemOSQL/M statements:

Alternative 1: Splitting

CREATE CATGEORY STUDENT

(name:STRING, ssn:INTEGER)

AS SELECT name, ssn

FROM cdb1.STUDENT

SELECT name, ssn

FROM cdb2.STUDENT

CREATE CATEGORY ADDRESS

(address:STRING)

AS SELECT address

FROM cdb1.STUDENT

SELECT [street, city, zip]

FROM cdb2.ADDRESS

CREATE RELATION ADDRESS_OF

(DOMAIN STUDENT RANGE ADDRESS)

AS FROM cdb1.STUDENT a, cdb1.STUDENT b

WHERE a.STUDENT = b.STUDENT

FROM cdb2.STUDENT__has__

Note that we used the surrogate (STUDENT attribute) to identify common objects in WHERE clause.

Alternative 2: Integrating

CREATE CATEGORY STUDENT

(ssn:INTEGER, name:STRING, address:STRING)

AS SELECT ssn, name, address

FROM cdb1.STUDENT

SELECT ssn, name, [has__street, has__city, has__zip]

FROM cdb2.STUDENT

Note that in alternative 2, we use the relation *has* to query category cdb2.ADDRESS.

Usually, when resolving schema-level conflicts a combination of the above resolutions may need to be applied. The DBA creating the global views firstly determines the semantic relation between global schema construct and one of the component database schema constructs. Next, Knowledge Base tool interacts with the DBA to assist in creating a semantically correct global view from the component database schemas. In the next sections, we will consider the methodologies of managing the meta-data and

knowledge gained in integration for intelligent design decisions and query processing strategies.

4.2.3 Handling Inconsistent Data

In previous discussions, we did not consider the issue of inconsistent data. That is, similar concepts in different component databases having different data values. For instance, let us consider a category EMPLOYEE derived from categories cdb1.EMPLOYEE and cdb2.EMP. Both categories contain an attribute called *salary* which represents the salary of an employee. Let us assume that the equivalence condition is cdb1.EMPLOYEE.social-sec = cdb2.EMP.ssn . It is possible that the attribute *salary* in categories cdb2.EMP and cdb1.EMPLOYEE may contain different values when cdb1.EMPLOYEE.social-sec = cdb2.EMP.ssn . That is, same employee may have different *salary* values. Until now, we choose either cdb1.EMPLOYEE.salary or cdb2.EMP.salary value even for equivalent objects assuming that they are consistent. Another resolution is to obtain the combined values of cdb1.EMPLOYEE.salary and cdb2.EMP.salary using some aggregate functions. Let us look at the SemOSQL/M statement for this case:

```
CREATE CATEGORY EMPLOYEE
        (ssn:INTEGER, name:STRING, salary:NUMBER)
AS      SELECT cdb1.EMPLOYEE.social-sec, cdb1.EMPLOYEE.name,
                SUM(cdb1.EMPLOYEE.salary, cdb2.EMP.salary)
        FROM    cdb1.EMPLOYEE, cdb2.EMP
        WHERE cdb1.EMPLOYEE = cdb2.EMP BY EQ_COND
```

SELECT social-sec, name, salary

FROM cdb1.EMPLOYEE

WHERE cdb1.EMPLOYEE BY BOUNDARY

(cdb1.EMPLOYEE – cdb2.EMP)

SELECT ssn, name, salary

FROM cdb2.EMP

WHERE cdb2.EMP BY BOUNDARY (cdb2.EMP – cdb1.EMPLOYEE)

Note that in the above instance, we create a global category EMPLOYEE which is a union of objects of cdb1.EMPLOYEE and cdb2.EMP which themselves are related by SEM_OVER (i.e. cdb1.EMPLOYEE SEM_OVER cdb2.EMP). Hence, when considering objects in category EMPLOYEE which are common to both categories cdb1.EMP and cdb2.EMPLOYEE by object equivalence (presented in SemOSQL/M as EQ_COND), we take the SUM of cdb1.EMPLOYEE.salary and cdb2.EMP.salary to obtain the salary of EMPLOYEE (i.e. handling inconsistencies). In other cases, we obtain either cdb1.EMPLOYEE.salary or cdb2.EMP.salary for the boundary cases. The equivalence and boundary cases are specified in the WHERE clause.

4.2.4 Knowledge Management in Database Integration

It is advantageous to store and manage the meta-data and knowledge mentioned earlier in a centralized manner. This will allow easily verifying the consistency of existing knowledge, acquiring new knowledge and maintaining the knowledge. The use of semantic dictionaries, global thesauruses and other techniques has been proposed in literature. We adopt Knowledge Bases for this purpose. The schemas for the storage

component of the Knowledge Bases are discussed in section 4.2.4.1. A tool that utilizes the knowledge in the knowledge base in creating the global views is outlined in section 4.2.4.2.

4.2.4.1 Knowledge Base

A Knowledge Base (KB) is used as a means for storage and manipulation of meta-data and knowledge discussed previously. In the architecture (see chapter 2), Knowledge Bases act as the interface between integration/schema definition phase with query processing phase at the Global and Relational Sites. The knowledge acquired during the integration/schema definition processes is made available through the Knowledge Base for query processing. In our architecture (refer chapter 2), three Knowledge Bases can be identified. Knowledge Bases used at the Relational and Semantic Sites and a Knowledge Base used at the Global Site. The Knowledge Bases store and manage different types of information at the different sites. Due to the fact that Sem-ODM is an expressive and powerful data model, it was natural to utilize a Sem-ODB for the storage component for the Knowledge Bases. The semantic schema designs for each Knowledge Base are presented in appendices 1-5. In the following sections, we describe each KB schema.

4.2.4.1.1 Knowledge Bases at Component Sites

The KB at the Relational Site subsumes the information content captured by the KB at the component Semantic Site. Hence, we will discuss the Knowledge Base at the Relational Site and provide descriptions for the common portions with the knowledge

base at Semantic Site. The knowledge base schema at the Relational Site captures the following information: (i.) relational database schema; (ii.) transformed Sem-ODM database schema; and (iii.) mapping information between the relational database schema and its transformed Sem-ODM schema. This information is crucial for both schema transformation and query translation. Also, semantic enrichment of the transformed schemas (which includes incorporating context information through ontology and property functions) is stored in the Knowledge Base. The Integration and Knowledge Reconciliator module uses this information for semantic heterogeneity resolution. The sub-schemas that make up the knowledge base schema of the Relational Site are presented in appendices 1 – 4. Each sub-schema is described below.

Sem-ODM consists of *category*, which may be inherited and *relation*, which is a relationship between categories. Appendix 1 presents the meta-schema of Sem-ODM. Note that this sub-schema covers only the main concepts of the Sem-ODM data model. A detailed meta-schema and its descriptions can be found in [117]. The primary constructs of Sem-ODM are *CATEGORY*s and *RELATION*s. A *CATEGORY* can be either *ABSTRACT* or *CONCRETE*. *ABSTRACT CATEGORY*s represent objects that are explicitly created representing real-world concepts, ideas or objects. *CONCRETE CATEGORY*s represents printable values. *CONCRETE CATEGORY* captures the different types of datatypes present in Sem-ODM. These can be different number ranges (represented by *NUMBERS RANGE*), string ranges (represented by *STRINGS RANGE*), user enumerated types (represented by *ENUMERATED TYPE*) or binary range types (represented by *BINARY*). A *RELATION* is a mapping between objects in the domain to

objects in the range. A *RELATION* having a range of a *CONCRETE CATEGORY* is termed an attribute of the domain. This subschema has the ability to store any semantic schema.

The meta-schema of a relational database schema is shown in appendix 2. This sub-schema contains *TABLE*s, *FIELD*s which belong to tables and their respective *DATATYPE*s. Primary and foreign keys are represented by categories *PRIMARY KEY FIELD* and *FOREIGN KEY FIELD* respectively. The functional dependencies are represented by relation *refers-to*. It is evident how the relational schemas are stored in this sub-schema and hence not discussed further.

The subschema illustrated in appendix 3 represents the mapping information between the transformed Sem-ODM and component relational schemas. Categories *META OBJECT* and *COMPONENT META OBJECT* are the same categories represented in appendices 1 and 2 respectively. It is significant to note that category *META OBJECT* is not directly derived from *COMPONENT META OBJECT*, instead from category *VIEW META OBJECT*. *VIEW META OBJECT* is categorized to *COMPONENT META OBJECT* and *VIEW SPECIFICATION*, which is further categorized to categories *VIRTUAL CATEGORY*, *VIRTUAL RELATION* and *VIRTUAL ATTRIBUTE*. If the object in the Sem-ODM schema is directly derived from the relational schema object, then *META OBJECT* is derived from an object in category *COMPONENT META OBJECT*. However, it is quite possible to contain different types of schematic heterogeneities between Sem-ODM schema and relational schema similar to schematic heterogeneities

between Sem-ODM component and global schemas (see section 4.2.2). Similar to the creating global Sem-ODM schemas from component schemas, the different types of schematic heterogeneities are specified in SemOSQL/M and translated and stored in the knowledge base schema. Note that we will not discuss schema level heterogeneity resolutions between Sem-ODM and relational schemas as the issue of schematic heterogeneity resolution between OO and relational data models have been discussed extensively in previous work and can be directly applied to Sem-ODM as described in [7].

The fact that we store the derivation information specified in SemOSQL/M in the knowledge base schema, allows to conveniently derive the semantics of the conflicts resolutions. The schematic heterogeneity resolutions are stored in the categories represented by *VIEW SPECIFICATION*. Category *VIRTUAL CATEGORY* captures schema level heterogeneities that may occur between a category in Sem-ODM schema and a set of tables in the relational schema. For instance, a category *GRAD_STUDENT* in Sem-ODM schema represents the graduate students. However, this is derived from table *STUDENT* which represents both graduate and undergraduate students. Hence, an object in *VIRTUAL CATEGORY* will represent the extraction of only graduate students from table *STUDENT* via the attribute *VIRTUAL_CATEGORY::where* clause (this example describes the boundary condition specification using the *where* clause). Similarly, a *relation* in the Sem-ODM schema is derived from a functional dependency in component relational schema. This is specified by category *VIRTUAL RELATION*. An *attribute* in the Sem-ODM schema may have schema level conflicts with fields of the relational

schema (similar to conflicts between component Sem-ODM schemas and global Sem-ODM schema specified in sections 4.2.2). This information is captured by category *VIRTUAL ATTRIBUTE*. Likewise, different types of heterogeneities are resolved with the addition of middle-level categories between transformed schema and component schema.

For clarity and illustration purposes, we describe the following example.

Example 21. Let us consider the following relational schema.

STUDENT(soc-sec, last-name, first-name, birth-year, major-dept, minor-dept)

INSTRUCTOR(soc-sec, last-name, first-name, birth-year)

DEPARTMENT(dept-code)

DEPARTMENT_NAMING(name-key, main-name)

WORK(instructor-id-in-key, department-main-name-in-key)

The names of *tables* are represented outside the brackets in capital letters and *fields* of each *table* are placed inside the appropriate brackets. The *primary key fields* of each *table* are underlined. The fields *major-dept* and *minor-dept* of table *STUDENT* are foreign keys referring to *dept-code* field of *DEPARTMENT* table representing the student's majoring and minoring departments respectively. The *dept-code* field of table *DEPARTMENT* is a foreign key referring to the *name-key* field of table *DEPARTMENT_NAMING*. The field *instructor-id-in-key* of table *WORK* is a foreign key field referring to *soc-sec* field of table *INSTRUCTOR* and *department-main-name-in-key* is a foreign key field referring to field *dept-code* of table *DEPARTMENT*.

Let us assume that the KDBTool and DBA at the component Relational Site generated the Sem-ODM schema shown in figure 23, which represents the transformed relational schema represented above. The storage of the relational schema and its transformed Sem-ODM schema in the meta-schemas presented in appendices 1 and 2 are self-explanatory and is omitted from discussion. We consider the mapping between relational and Sem-ODM schema which is captured by the sub-schema presented in appendix 3. This mapping information and the respective categories of sub-schemas shown in appendices 1-3 are given in table 1.

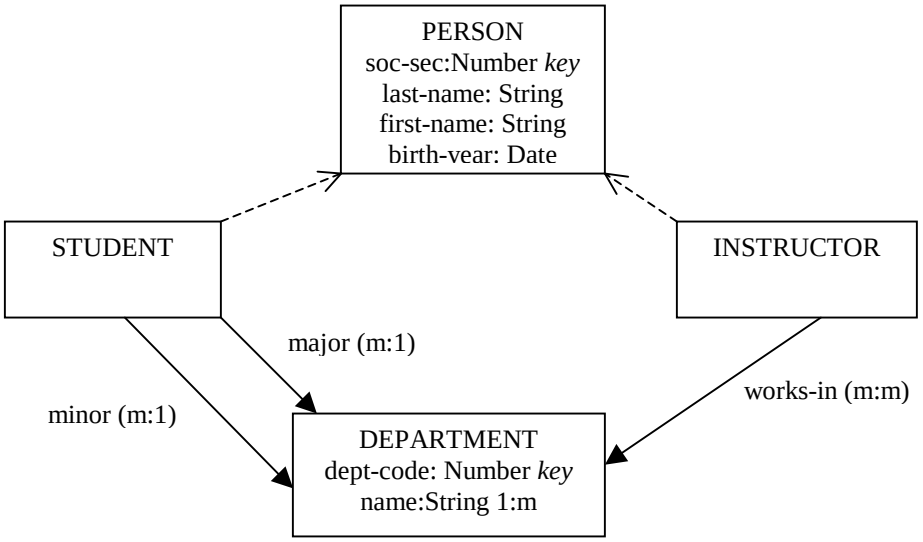


Figure 23. Semantic Schema Created By Transforming a Relational Schema

Semantic Schema Entity	**Relational Schema Entity (*is-derived-from*)**	**Category in Knowledge Base Schema**
PERSON	STUDENT INSTRUCTOR	TABLE TABLE
PERSON.soc-sec	STUDENT.soc-sec INSTRUCTOR.soc-sec	FIELD FIELD
PERSON.last-name	STUDENT.last-name INSTRUCTOR.last-name	FIELD FIELD
PERSON.first-name	STUDENT.last-name INSTRUCTOR.last-name	FIELD FIELD
PERSON.birth-year	STUDENT.birth-year INSTRUCTOR.birth-year	FIELD FIELD
STUDENT	STUDENT	TABLE

INSTRUCTOR	INSTRUCTOR	TABLE
DEPARTMENT	DEPARTMENT	TABLE
DEPARTMENT.name	DEPARTMENT_NAMING. main-name *from*: DEPARTMENT, DEPARTMENT_ NAMING *where*: DEPARTMENT.dept- code = DEPARTMENT _NAMING.name-key	VIRTUAL_ATTRIBUTE (represented by VIRTUAL_ RELATION which is joined by VIRTUAL_ATTRIBUTE::has relation)
STUDENT::major	*from*: STUDENT, DEPARTMENT *where*: STUDENT.major-dept = DEPARTMENT.dept-code	VIRTUAL_ RELATION
STUDENT::minor	*from*: STUDENT, DEPARTMENT *where*: STUDENT.minor-dept = DEPARTMENT.dept-code	VIRTUAL_ RELATION
INSTRUCTOR::work	*from*: INSTRUCTOR, WORK, DEPARTMENT *where:* (INSTRUCTOR.soc-sec = WORK.instructor-id-in-key) AND (WORK.department- main-name-in-key = DEPARTMENT.dept-code)	VIRTUAL_ RELATION

Table 1. Mappings between Sem-ODM and relational schema

The above example illustrates the use of middle-layer categories in the mapping schema (Appendix 3) for resolving conflicts. The attribute *DEPARTMENT.name* has cardinality *1:m* which is not possible to be represented directly by a relational schema. Thus, *VIRTUAL_ATTRIBUTE* category is used for resolving this conflict. Similarly, *relations* in semantic schema cannot be directly mapped to relational schema constructs and hence we introduce the category *VIRTUAL_RELATION*.

Appendix 4 depicts sub-schema capturing ontology information and mappings from Sem-ODM schema to the ontology. The ontology is composed of a set of *META CONCEPT*s which are either *CONCEPT*s or *RELATIONSHIP*s between concepts. According to the discussion in chapter 3, a semantic network is used as a representation

scheme for ontologies. Thus, in sub-schema of appendix 4, the category *CONCEPT* maps to the *node* of the semantic network while category *RELATIONSHIP* maps to the *links* between the nodes. A meta-object of Sem-ODM schema maps to the ontology via a property function represented by category *PROPERTY FUNCTION*. In the general case, attributes and categories in Sem-ODM are mapped to a *CONCEPT* while relations in Sem-ODM are mapped to an *OTHER* relationship. Property functions have a *primary mapping* and a set of restrictions which is captured by relation *restricted-by*. The mapping from a Sem-ODM schema to ontology is discussed in chapter 3 and hence not explained further. The current implementation of SemWrap [96] does not support ontology mappings, however plans to include ontology concepts in the future versions.

Knowledge Base at Semantic Site is similar to Knowledge Base schema at relational site. However, it contains only sub-schemas shown in appendices 1 and 4 as it does not need to contain relational schemas similar to the KB at Relational Site. However, similar to Relational Site's Knowledge Base, it contains mappings to the ontological concepts. The next section describes the Knowledge Base at the Global Site that stores integration information from a set of Sem-ODM schemas.

4.2.4.1.2 Knowledge Base at Global Site

The heterogeneities between a set of Sem-ODM schemas are resolved at the Global Site. This process requires (i.) identification of semantic relations between constructs of component schemas; (ii.) acquiring means for determining object equivalences for related constructs; (iii.) determining boundary conditions of related

entities; and (iv.) resolving schematic heterogeneities. The KB at Global Site focuses on the storage of these types of knowledge. The concepts mentioned in (i.)-(iii.) have been discussed in chapter 3 and (iv.) in the previous sections of this chapter.

A sub-schema of the KB schema at Global Site is presented in appendix 5. The category *META OBJECT* is part of a sub-schema similar to appendix 1 (not shown due to redundancy). Category *VIEW SPECIFICATION* has subcategories *VIRTUAL RELATION*, *VIRTUAL CATEGORY* and *VIRTUAL ATTRIBUTE* similar to sub-schema in appendix 3. These categories are not shown in this schema to avoid complexity and redundancy. Attributes of each category are omitted as well from discussion to avoid complexity and ease of discussion. Category *SEMANTIC RELATION* captures the different types of semantically related entities. Semantic relation "semantically disjoint" (i.e. SEM_DIS) is not represented and assumed by default. *Object equivalences* (described in section 3.2) are represented by category *OBJECT EQUIVALENCE PATH*. The *boundary conditions* (discussed in section 3.2) are specified in category *BOUNDARY CONDITION* and are represented by an object of *INTEGRATED META OBJECT* (using relation *represented _by*). Similar to appendix 4, *INTEGRATED META OBJECT* is mapped to ontology. This sub-schema is omitted since it is discussed in appendix 4. The *INTEGRATED META OBJECT* category contains all the concepts of component databases as well as global views. A *META OBJECT* belongs to (represented by *belongs-to* relation) either a global view (represented by categories *GLOBAL*) or component database schema (represented by categories *COMPONENT DATABASE*). A global meta object is derived from an *INTEGRATED META OBJECT* category (via relation *is-derived-from*) similar to *META*

*OBJECT*s being derived from *VIEW META OBJECT* of subschema of appendix 3. Most of the discussions on previous schemas in the above sections overlap with this schema and hence we avoid further discussion of this schema. A pre-cursor to the work presented in this section can be found in [93].

For illustration purposes, we describe the following example that describes the different types of semantic knowledge stored at the KB of Global Site.

Example 22. Consider two schemas from different component databases, DB_1 and DB_2. Let us assume that schema of DB_1 has the Sem-ODB schema presented in figure 23 of $university_A$. Let DB_2 have the Sem-ODM schema of $university_A$ shown in figure 24. We have the following semantic relations from schemas of DB_1 and DB_2. The semantic relation, SEM_DIS, is assumed by default:

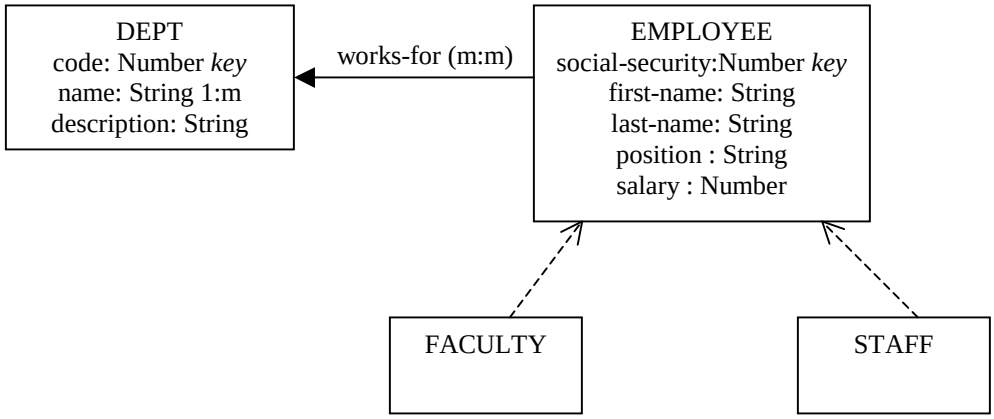


Figure 24. Sem-ODM Schema of DB_2

DB_1:

[1] DB_1.*STUDENT* SEM_SUB DB_1.*PERSON* (ISA relationship)

[2] DB_1.*INSTRUCTOR* SEM_SUB DB_1.*PERSON* (ISA relationship)

DB_2:

[3] DB_2.*STAFF* SEM_SUB DB_2.*EMPLOYEE* (ISA relationship)

[4] DB_2.*FACULTY* SEM_SUB DB_2. *EMPLOYEE* (ISA relationship)

Semantic relations between DB_1 and DB_2:

[5] DB_1.*PERSON* SEM_OVER DB_2.*EMPLOYEE* (since category *PERSON* contains the faculty and students of $university_A$ while category *EMPLOYEE* contains faculty and staff of $university_A$).

We will not enumerate all the possible semantic relations between entities of DB_1 and DB_2 due to space limitations. These semantic relations are obtained using rules described in chapter 3. Also, the global DBA can identify them as well. For this example, *object equivalence* conditions and *boundary conditions* for relation [5] are illustrated below.

Assuming that key attributes *social-security* of DB_2.*EMPLOYEE* match key attribute *soc-sec* of DB_1.*PERSON* are both referring to the social security numbers of a person, we can directly obtain the following equivalence condition:

DB_1.*PERSON*.*soc-sec* = DB_2.*EMPLOYEE*.*social-security*

Hence, the same real world persons in both DB_1.*PERSON* and DB_2.*EMPLOYEE* can be determined by comparing their social-security numbers.

The boundary conditions for [5] include the following. Note that in the following rules, we have assumed that all objects of category DB_2.*EMPLOYEE* are either objects of category DB_2.*FACULTY* or DB_2.*STAFF* which are disjoint categories.

1. EXT(DB_1.*PERSON*) ∩ EXT(DB_2.*EMPLOYEE*): contains the set of instructors/faculty at $university_A$, which is represented by entities DB_2.*FACULTY* and DB_1.*INSTRUCTOR* which are related by semantic relation SEM_EQ.
2. EXT(DB_1.*PERSON*) – EXT(DB_2.*EMPLOYEE*): contains the set of students of $university_A$, which is represented by entity DB_1.*STUDENT*.
3. EXT(DB_2.*EMPLOYEE*) – EXT(DB_1.*PERSON*): contains the set of staff members of $university_A$, which is represented by entity DB_2.*STAFF*

This information will be stored in the Knowledge Base of Global Site and used in the creation of global schemas/views and query processing. The creation of global views with the assistance of a tool is discussed in the next section.

4.2.4.2 A Tool used for Global View Definition

A major overhead in the global schema approach is the need for creating and maintaining a global schema over the component heterogeneous databases. In our approach, we feel that a single global schema is not required. This is because of the large number of component schemas are integrated into the system and hence it is usually the case that a single user/user group may not require access to all of such information. Hence, we propose the creation of global views for different groups of users meeting their requirements. This reduces the complexity of creating and maintaining a global schema significantly. Also, a tool to semi-automatically create the global views is described in this section, thus reducing the complexity and overhead further. This tool uses the semantic knowledge, stored in the Knowledge Base, to provide intelligent design decisions in creating global views.

Similar to creating a Sem-ODM schema for an application, we first proceed creating the constructs (i.e. categories and relations) for the global view. Once, we have defined the global view for the particular user group or application, the next step is to define the meaning for each construct of the global schema. This is performed, by mapping each construct onto the ontology. The knowledge base tool, then obtains all the semantically related constructs (i.e. SEM_EQ, SEM_SUB, SEM_OVER) from the component database schemas by referring to the knowledge base. Also, it provides with information as to the degree of completeness of information in the component database constructs (since the semantic relations are based on extents of schema constructs). This provides the global DBA to express the extents for each construct in the global schema.

For instance, in the scenario of example 22, we create a category *TEACHER* in the global schema which represents the set of instructors in $University_A$. Then, the tool by referring the Knowledge Base will present categories *INSTRUCTOR* and *PERSON* of DB_1 with the appropriate boundary conditions and means to create object equivalences. Also, it will present the DBA with categories *STAFF* and *EMPLOYEE* with the relevant information. The DBA need not learn any semantics of the component database schema or specific information about the component schemas. The DBA does not require searching all the constructs of the component database schemas to find schema constructs, which contain relevant information. Basically, the problem of semantic heterogeneity is resolved. This is one of the major obstacles for ubiquitous use of multidatabase technology and this factor is resolved. Next, the DBA defines the global

view resolving schema-level heterogeneities. The tool makes important suggestion at this stage as well. By considering the different types of schematic heterogeneities present in the Knowledge Base, the tool is capable of suggesting intelligent design choices for schema definition.

Another limitation of the global schema approach is that the necessity for all semantic knowledge before-hand in creating a global schema. Our approach allows incremental acquisition of knowledge about component schemas as they progress and learn about the component database schema. For instance, certain component schema may not be mapped to ontologies initially due to lack of information. They may be represented as semantically disjoint entities from semantically related entities. When knowledge is acquired incrementally over a period of time, these new semantic relations are identified. Note that this process is transparent to the users' global schema which is designed in top-down methodology similar to centralized database design. As new semantically related entities are gained they are mapped to constructs of the global views, hence made available to global users without the need to change their applications.

There are many other advantages of using our approach. It is quite probable that the schemas of the underlying databases may change. In our approach, the changes have a minimal effect of the global users with the least overhead to the global DBA. When the semantics of a component schema construct change, say a construct is deleted, the changes are reflected in the property functions and ontology mappings, which are transmitted to the global site. This change may result in a change of semantic relations

between component schema construct and global schema construct. Then this mapping is deleted or altered as necessary and informed to the DBA. The users of global view do not see any effect. A query on the global construct will obtain results from a different component database. Also, the completeness of the query results can be viewed notified to the user if the system is unable to provide complete answers due to the change in semantics. For instance, in the previous example, it is learnt that DB_1.*INSTRUCTOR* category represents instructors of $University_B$ rather than $University_A$. This change is noted in the ontology mapping and transmitted to the global site. At the global site, there will be change from global category *TEACHER* being related by SEM_EQ to DB_1.*INSTRUCTOR* to *TEACHER* SEM_DIS DB_1.*INSTRUCTOR* assuming that instructors in $University_A$ cannot be instructors of $University_B$. This change is propagated within the Knowledge Base using rules described in chapter 3 to make the knowledge consistent. The mapping from global category *STUDENT* to DB_1.*INSTRUCTOR* is deleted automatically. Hence, now a query to obtain instructor information will be directed to only component database DB_2. This change is transparent to the user and minimal work by the DBA. The DBA is informed of the change by the tool.

The next section summarizes this chapter with a description of the advantages of the discussed approach to database integration when compared with other approaches.

4.3 Summary

In this chapter, we introduced a language called SemOSQL/M to define global views over a set of Sem-ODM component schemas. An exhaustive list of the different

types of schematic heterogeneities that may occur during integration of a set of component Sem-ODM schemas is discussed and their resolutions are presented. Knowledge management in our architecture is discussed. Knowledge Base schemas for global and component sites are extensively discussed. Finally, a tool that assists in creating global views is described.

Our approach to integration is unique from other approaches as it combines the semantic conflict resolution techniques (described in chapter 3) with schema-level conflict resolution. We have seen a similar approach in [51] which combines context information to schema level heterogeneity resolution. However, our approach to semantic heterogeneity resolution is based on semantic relations (similar to [83]) in contrast to [51].

As we mentioned in the first chapter, the ideal situation is to provide an interface similar to a centralized database system to the multidatabase users. Similar to a centralized database system, this requires the creation of global schemas/views. A major disadvantage in global schema multidatabase approach, which the critics of this approach have pointed out, is the significant overhead in the creating and maintaining of a single global schema. This factor is significantly reduced in our approach by allowing different views to be created for different user groups. This factor avoids the creation of a single global schema. Also, the need for complete semantic knowledge of all schemas is avoided. The methodology allows step-wise incremental approach to gain knowledge about the component schemas, which minimizes the effect to the global users whose

global views are designed in a top-down manner to meet their requirements. Also, an intelligent tool, utilizing the semantic knowledge acquired during integration process is outlined which automates a significant portion of creating a global view. This tool resolves the semantic heterogeneity problem.

Also, in today's dynamically changing environments, efficient means for incorporating changes must be considered. Since our integration is based on semantic knowledge, the changes in semantics are easily propagated without affecting the global users or applications of the global views. Also, it is possible to guarantee the completeness of user's query results, since our semantic knowledge is based on extents. When complete answers are impossible, the degree of completeness can be efficiently measured.

In summary, the benefits of our approach include (i.) ideal solution to heterogeneous data access problem by providing global views for users' accessing multiple heterogeneous data sources in an expressive data model and query language (similar to centralized database systems); (ii.) our methodology avoids the overhead of creating and maintaining a single global schema by allowing the creation of multiple global views for each user group; (iii.) global view definition is performed in a top-down manner similar to database design methodologies of a centralized system, meeting the requirements of each user group. Next, these schemas are mapped to ontologies from which automatic semantic heterogeneity resolution takes place. This is significant improvement from existing approaches; (iv.) the ability of our methodology to gain

semantic knowledge in a step-wise, incremental manner without affecting the global users is beneficial. This has significant advantages from previous attempts, which require all semantic knowledge before creating a global schema. Obtaining such knowledge before-hand may be impossible and impractical for legacy database system due to the un-availability of domain experts; and (v.) our approach can handle dynamic changes of component database schemas in a very graceful manner (in contrast to most previous approaches).

The above-mentioned factors are significant improvements from the existing approaches. Hence, our approach to database integration provides an ideal situation for distributed, heterogeneous database access reducing the overhead incurred previously. The next chapter focuses on the query processing aspects of the Heterogeneous Distributed Database System. Query optimization algorithms try to exploit the semantic knowledge for efficient query processing providing results that adhere to data quality attributes such as completeness.

5. QUERY PROCESSING

We have seen extensive work on query processing and optimization in database systems, centralized and distributed alike, in the past two-three decades. This has resulted in important and well-understood principles and approaches to query processing in general. Query processing in multidatabase systems include several steps. Firstly, a query based on a global view is decomposed into a set of subqueries, along with a query execution plan (postquery processing [79]) to combine the results of the subqueries. Next, the translation and optimizing of subqueries at the local sites takes place. Query processing in multidatabase systems borrows concepts and techniques used in query processing of centralized and distributed database systems. However, certain unique features of multidatabase systems require developing innovative techniques for query processing and optimizing.

In this chapter, we will focus on Semantic SQL query processing in the Heterogeneous Distributed Database System introduced in the previous chapters. Our approach to Semantic SQL query processing and optimizing incorporates many existing techniques discussed in literature related to centralized database query processing, distributed database query processing and multidatabase query processing. We discuss some related work in section 5.1. In section 5.2, we discuss our approach to Semantic SQL query processing in a multidatabase environment. The adoption of techniques initially developed for relational SQL query processing to Semantic SQL query processing is described. A unique feature of our integration process (see chapters 3-4) is the acquisition of semantic knowledge. Such semantic knowledge can be exploited for

optimization of Semantic SQL queries. We present some techniques that utilize the semantic knowledge for query optimization. Finally, concluding remarks for the chapter is presented.

5.1 Related Work

The relational model was introduced by Codd in [26] (extended later in [27]). There onwards we have seen relational algebra being used to query a relational schema. Next, we have seen the development of declarative query languages for the relational model and finally the standard query language SQL [110] which has been extended from time to time (SQL-89, SQL-92, SQL-99). Optimization of such queries specified in these query languages has taken shape in a centralized environment. Usually, a SQL query statement is broken into blocks of SQL statements (each block containing SELECT-FROM-WHERE-GROUP BY clause). Next, each query block is transformed into a set of relational algebra operators specifying the query. Logical optimization takes place at this level (rule-based optimization – such as pushing down selects). Physical optimization is performed (considering index structures, access paths, etc.) to reduce data retrieval costs. Many database textbooks (such as [32], [40], [45], [88], [116] and others) discuss this process in detail.

Distributed database query processing considers the data distribution costs, transmission costs and horizontally and vertically partitioned relations. Many approaches for distributed database query processing and optimization (such as semi-joins [11]) are described in literature [124]. Some of these techniques can be directly applied for query

processing and optimizing in multidatabase systems as well. However, distinctive features of multidatabase systems give rise to unique issues which do not arise in query optimization for homogeneous distributed database systems. Some unique features of multidatabase systems from distributed database systems include [73]:

(i.) Site autonomy: Component database retains complete control over local data and processing. This fact has many implications such as *communication autonomy* which means that component sites independently determines what information it will share with the global system, what global requests it will service, when it will participate in the multidatabase system, and also when it will stop participating. This adds complexity to the query processing due to dynamical changes of the data sources. Another implication is *design autonomy* which means that component databases are free to optimize access paths and query processing methods without any obligation to the multidatabase system. Also, statistical and relevant information may not be available for the global query optimizer. Thirdly, *execution autonomy* where the global system is considered as another user and no cooperation between global query processing and component database query processing modules to correlate optimizing procedures similar to distributed database query processing.

(ii.) System heterogeneity: System heterogeneity occurs at several different levels. Component databases may reside on computer systems with different architectures, connected via different types of networks, use different types of communication protocols and support different types of data models. Hence, the

assumption in distributed databases that the local sites equal in terms of the processing capability no longer holds.

(iii.) Semantic heterogeneity: Same real-world object may be stored in different component databases. This situation is avoided in the homogeneous distributed database approach.

Query processing in multidatabase systems needs to address these issues.

The issues of system and semantic heterogeneity in multidatabase systems have been discussed extensively in previous chapters. We discuss some related work in query processing and optimizing in multidatabase systems. Due to the different data models of the component databases, a global query after decomposing into a set of subqueries needs to be translated into the appropriate query language supported by the component database. Query translation between different query languages have been studied extensively by many researchers ([25], [64], [80], [126], and others). Most of the work is in translation from either object-oriented databases to relational databases or relational queries to queries against hierarchical and network databases. Translation of Semantic SQL queries to relational queries in SemWrap [96] has been detailed in [74]. Due to site autonomy, obtaining statistics from component database systems is difficult for query optimization. Different approaches to acquire cost parameters for accessing component databases have been discussed in [38], [128] and others. Query optimization utilizing different operations (such as semi-joins, outer joins) to obtain inexpensive query execution plans (QEPs) have been discussed in [24], [34], [35], [39]. Exploiting parallelism for query processing optimization is discussed in [113]. Due to the schema

conflicts, it is possible for query answers to be partial (i.e. not certain). A methodology for understanding uncertain answer tuples for global queries is presented in [115]. Query optimization in multidatabase systems taking into consideration the different schema conflicts is discussed in [66]. These methodologies take into consideration different aspects for optimizing queries in a multidatabase system. In the next section, we will discuss query processing of Semantic SQL queries in the Heterogeneous Distributed Database System and consider optimization techniques utilizing semantic knowledge.

5.2 Our Work

This section discusses the steps involved in processing a Semantic SQL query statement posed on a global Sem-ODM view. The steps in Semantic SQL query processing include the following: (i.) A Semantic SQL query statement is firstly scanned, parsed and checked for semantic errors. The semantic checking phase un-abbreviates the abbreviated Semantic SQL query. This un-abbreviation process produces the virtual tables on which the Semantic SQL query is posed. This step is discussed in section 5.2.1. (ii.) Next, the Semantic SQL statement is transformed to a relational algebra expression based on the virtual tables. Consequently, relational algebra expression is optimized based on logical optimization techniques (rule-based optimization). This step is described in section 5.2.2. (iii.) Thirdly, an internal representation of the virtual table is presented and further simplification of virtual tables is discussed. Logical optimization of QEP is carried out further. This step is described in section 5.2.3. (iv.) Next, the constructs of the global schema are replaced by component database constructs. At this step, optimization strategies are considered to efficiently acquire the query result. Application of existing

techniques for query optimization is presented. Also, innovative query optimization strategies based on semantic knowledge are discussed. Section 5.2.4 discusses these issues. (iv.) Finally, generation of subqueries from the QEP and query execution plans for integrating the results of subqueries at different sites are discussed.

5.2.1 Step 1: Scanning, Parsing and Semantic Checking

Similar to standard query processing, error checking, scanning, parsing and semantic checking procedures takes place in processing of Semantic SQL queries. A distinction between Semantic SQL query processing and relational SQL query processing is that semantic checking phase un-abbreviates the abbreviated Semantic SQL statement, in addition to checking the correctness of the query. That is, this process determines the size and the attributes of each virtual table that the query references. An algorithm that produces the virtual tables and the un-abbreviated Semantic SQL query statement is given below:

Algorithm:

Input - Abbreviated Semantic SQL query (*AbbSemSQLQuery*) and a Sem-ODM view (*SemanticSchema*) on which the query is posed

Output - Unabbreviated Semantic SQL query (*UnAbbSemSQLQuery*) and a set of virtual tables (*VirtualTables*) on which the query is posed.

1. *UnAbbSemSQLQuery* ← Algorithm_A(*AbbSemSQLQuery*, *SemanticSchema*)
2. if *UnAbbSemSQLQuery* ≠ NULL then

   VirtualTables ← Algorithm_B(*UnAbbSemSQLQuery*, *SemanticSchema*)

   else

return NULL, NULL

To achieve its goal the algorithm uses two sub-modules, namely $Algorithm_A$ and $Algorithm_B$ which are described below.

$Algorithm_A$: The function of $Algorithm_A$ is to un-abbreviate the abbreviated attribute names in the original Semantic SQL query. $Algorithm_A$ outputs NULL for an erroneous Semantic SQL query, hence it is checked for NULL in step 2 of the algorithm given above. The high-level pseudo-code of $Algorithm_A$ is provided below:

1. Scan, parse and semantic check for any errors in *AbbSemSQLQuery*. If *Error,* return NULL
2. For each abbreviated attribute, *n*, in *AbbSemSQLQuery*
    - Find the shortest path, *p*, from the starting category *C* to *n* in the *SemanticSchema*, where *C* is a category or alias name in the FROM clause of *AbbSemSQLQuery.*
    - Concatenate *p* to *n* to obtain the full name of *n*
3. Return the unabbreviated query

Note that the abbreviated attribute, *n*, discussed in step 2 above is identified as follows. Let us consider an attribute, *m*, mentioned in *AbbSemSQLQuery* after step 1.

- Firstly, we check if attribute, *m*, begins with a category/alias name mentioned in the FROM clause of corresponding SQL statement
    - If yes, then we determine *m* to be un-abbreviated.
    - If not, we determine *m* to be abbreviated

Finding the shortest path to an abbreviated attribute, *m*, is processed as follows:

- For every category/alias name in the corresponding FROM clause of *m*

  We first calculate the shortest path (via relations) to a prefix of *m*, say *s*, where *s* is either a relation or attribute name. This is processed using a modified version of Dijkstra's shortest path algorithm.

- Next, we check if there exists more than one shortest path to attribute *m*
  - If there exists more than one shortest path to *m*, then return NULL
  - If there exists only one shortest path then $p \leftarrow$ shortest path
  - If there is no path, then return NULL

Algorithm$_B$: The unabbreviated Semantic SQL query (*UnAbbSemSQLQuery*) and Semantic Schema (*SemanticSchema*) are the input parameters for Algorithm$_B$, which outputs a set of virtual tables. The pseudo-code for Algorithm$_B$ is given below:

1. *VirtualTables* $\leftarrow \phi$
2. For each category or alias of category, *C*, in the FROM clause of *UnAbbSemSQLQuery*
   a. Create a virtual table named *C*
   b. For each attribute *m* in *UnAbbSemSQLQuery*

      Create a column named *m* in *C* if *m* begins with *C* (i.e. attribute of virtual category *C*)

   c. *VirtualTables* $\leftarrow$ *VirtualTables* $\cup$ {virtual table *C*}
3. Return *VirtualTables*

Example 24: For instance, let us consider the following abbreviated Semantic SQL query posed on the Sem-ODM schema presented in figure 24 of chapter 4.

```
SELECT    first-name, salary, name
FROM      EMPLOYEE
```

The un-abbreviated query is as follows:

```
SELECT    EMPLOYEE.first-name, EMPLOYEE.salary,
          EMPLOYEE.works-for__name
FROM      EMPLOYEE
```

where EMPLOYEE is the virtual table which is given below:

EMPLOYEE(first-name: String; salary: Number; works-for__name: String)

At the end of this step, we obtain an unabbreviated Semantic SQL query (output of $Algorithm_A$) and a set of virtual tables (output of $Algorithm_B$). The set of virtual tables can be interpreted as a relational schema and the unabbreviated Semantic SQL statement can be interpreted as a relational query on the relational schema defined by the set of virtual tables. The interpretation of the original Semantic SQL statement over the Semantic Schema is equivalent to the interpretation of the un-abbreviated Semantic SQL statement over the set of virtual tables. Thus, this step has converted the Semantic SQL query to an equivalent relational SQL query based on a set of virtual tables. This conversion process enables us to apply the existing relational algorithms/knowledge to process this query. These aspects are discussed in the following sections.

5.2.2 Step 2: Relational Algebra and Logical Optimization

The previous section converted the Semantic SQL statement into an equivalent SQL query based on the virtual tables. This has enabled us to apply the existing approaches of SQL query processing to the un-abbreviated query. Similar to query processing in a relational database, we convert the Semantic SQL statement into an extended relational algebra expression. We discuss this process briefly for completeness. Further details of this process can be found in database textbooks (such as [32], [40], [45], [88], [116] and others).

Firstly, the Semantic SQL query is parsed into a collection of query blocks. A *query block* is an SQL query with no nesting and exactly one SELECT clause and one FROM clause and at most one WHERE clause (in conjunctive form), GROUP BY clause, HAVING clause and ORDER BY clause.

Secondly, each block is expressed as a relational expression. We consider an extended set of relational algebra operators (which include DISTINCT, GROUP BY, HAVING, ORDER BY operators) in addition to the basic relational algebra operators (i.e. union – "$\cup$", intersection – "$\cap$", difference - "-", selection - "σ", projection – "π", product – "$\times$" and joins "$\triangleright\triangleleft$, …"). The meanings of extended relational algebra are similar to the respective clauses of SQL which the name specifies and hence not explained further. Every SQL query block can be expressed using extended algebra expression. For instance, let us consider the following query:

SELECT $a_1, \min(a_2), \ldots, a_n$

FROM $T_1, T_2, \ldots, T_n$

WHERE Condition1

GROUP BY $g_1, g_2, \ldots, g_m$

HAVING Condition2

ORDER BY $s_1, s_2, \ldots, s_k$

This SQL query can be represented by the following extended relational algebra expression:

$$\pi\, a_1, \min(a_2), \ldots, a_n\ (\text{ORDER BY}_{s_1, s_2, \ldots, s_k}\ (\text{HAVING}_{\text{Condition2}}\ (\text{GROUP BY}\ g_1, g_2, \ldots, g_m\ (\sigma_{\text{Condition1}}(T_1 \times T_2 \times \ldots \times T_n)))))$$

For instance, the query statement discussed in example 24 can be written directly as:

$$\pi_{\text{first-name, salary, works-for\_\_name}}(\text{EMPLOYEE})$$

The relational algebra expression can be used interchangeably as a query execution plan (QEP) that specifies the operations in obtaining the result of the query.

The next step is to optimize the extended relational algebra expression (i.e. QEP represented by the relational algebra expression). This is usually processed using pre-defined rules. Some rules that are applied for optimizing the extended relational algebra expression include [88]:

Rule 1:	$\sigma_{c1 \wedge c2 \wedge \ldots cn}(R) \equiv \sigma_{c1}(\sigma_{c2}(\ldots(\sigma_{cn}(R))\ldots))$	Cascading selections
Rule 2:	$\sigma_{c1}(\sigma_{c2}(R)) \equiv \sigma_{c2}(\sigma_{c1}(R))$	Commutative
Rule 3:	$\pi_{a1}(R) \equiv \pi_{a1}(\pi_{a2}(\ldots(\pi_{an}(R))\ldots))$	Cascading projections

Rule 4: $R \times S \equiv S \times R$ Commutative

Rule 5: $R \rhd\lhd S \equiv S \rhd\lhd R$ Commutative

Rule 6: $R \times (S \times T) \equiv (R \times S) \times T$ Associative

Rule 7: $R \rhd\lhd (S \rhd\lhd T) \equiv (R \rhd\lhd S) \rhd\lhd T$ Associative

Rule 8: $\pi_a (\sigma_c (R)) \equiv \sigma_c (\pi_a (R))$ if selection operation involves only attributes that are retained by the projection

Rule 9: $R \rhd\lhd_c S \equiv \sigma_c (R \times S)$ Combine

Rule 10: $\sigma_c (R \times S) \equiv \sigma_c (R) \times S$ if attributes mentioned in c appears only in R and not in S

Rule 11: $\sigma_c (R \rhd\lhd S) \equiv \sigma_c (R) \rhd\lhd S$ if attributes mentioned in c appears only in R and not in S

Rule 12: $\pi_a (R \times S) \equiv \pi_{a1} (R) \times \pi_{a2} (S)$ where ‘a1’ is the subset of attributes in ‘a’ that appear in R, and ‘a2’ is the subset of attributes in ‘a’ that appear in S

Rule 13: $\pi_a (R \rhd\lhd_c S) \equiv \pi_{a1} (R) \rhd\lhd_c \pi_{a2} (S)$ where ‘a1’ is the subset of attributes in ‘a’ that appear in R, and ‘a2’ is the subset of attributes in ‘a’ that appear in S

There are many other rules that are not enumerated above that preserve equality (see database textbooks). These rules can be utilized for optimizing the logical query execution plan. Rule-based logical query optimization has been extensively discussed in database textbooks and hence not discussed any further.

This section briefly discussed the translation of Semantic SQL queries to extended relational algebra expressions and logical optimizing using rules. This is the same approach taken by standard relational query optimizers. The fact that we were able to convert the Semantic SQL query to an equivalent relational SQL query based on virtual tables enabled to utilize the existing knowledge and techniques (for relational databases' SQL query processing) in Semantic SQL query processing.

5.2.3 Step 3: Expanding the Virtual Tables

During optimization in the previous section, we assumed that the virtual table is a single entity (i.e. one table). We can represent the virtual table as a tree structure where nodes represent *categories* in the global Sem-ODM schema and links from parent to child nodes represent the *relations* traversed to reach the categories containing attributes in the virtual table. In such a representation the root node is the starting category (refer chapter 2) of the virtual table. For illustration purposes, we provide the following example.

Example 25: Let us consider following Semantic SQL query statement over the Sem-ODM schema given in figure 23 of chapter 4:

```
SELECT    name, major___last-name
FROM      DEPARTMENT
WHERE     works-in___last-name = 'Kim'
```

The virtual table, DEPARTMENT for the above query is as follows:

DEPARTMENT (name:String; major___last-name:String; works-in___last-name:String)

We can represent this virtual table as a tree structure as follows (see figure 25). A similar tree representation of a virtual table used in translating from Semantic SQL queries to relational SQL queries is discussed in [74].

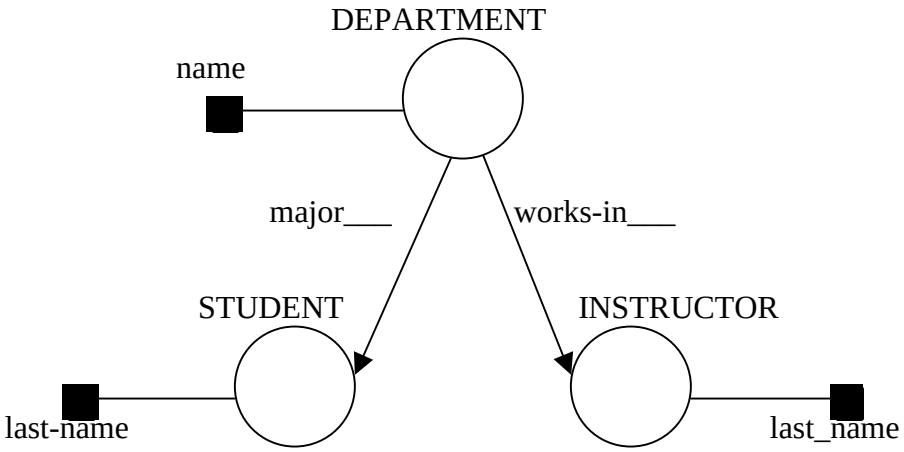


Figure 25. Tree Representation of a Virtual Table

Considering each node in the tree as a table, we can replace each link by an (left/right) outer join operation according to the semantics of extension of a virtual table. Using this procedure, we replace every instance of virtual table in the optimized extended relational algebra expression by a set of tables that are joined through outer join operations. For instance, the query in example 25 is represented as shown below using relational algebra operators:

$$\pi_{\text{name, major\_\_\_last-name}} (\sigma_{\text{works-in\_\_\_last-name = 'Kim'}} (\text{DEPARTMENT}))$$

After replacing the virtual table with outer-joins, the following relational algebra expression is obtained:

$$\pi_{\text{DEPARTMENT.name, STUDENT.last-name}} (\sigma_{\text{INSTRUCTOR.last-name = 'Kim'}} ((\text{DEPARTMENT } \text{LOJ}_{\text{majors\_\_}} \text{ STUDENT}) \text{ LOJ}_{\text{works-in\_\_}} \text{ INSTRUCTOR}))$$

where LOJ means *left-outer-join* and relation names are used as the join conditions.

This procedure has further simplified the extended relational algebra expression. Further optimization on this simplified expression can be performed (using rules) and also considering outer-join optimization techniques (such as techniques discussed in [67]).

At the end of step 3, we have obtained a simplified, logically optimized extended relational algebra expression. This expression moreover represents a query execution plan (QEP) to obtain the results for the global query. However, all of the above-mentioned optimizations were posed assuming that the global view was a single centralized database. Next section focuses on the decomposition of this QEP to a set of subqueries, based on component databases, and efficient integration of subquery results (postquery processing plans) to generate the results for the global query.

5.2.4 Step 4: Global Query Optimization

Previous sections have discussed the methodologies to include current relational database query processing techniques to process and optimize QEPs for Semantic SQL queries. This approach enabled us to exploit existing well-known and tested methodologies for Semantic SQL query processing. An important consideration during this optimization phase was that the global view is a centralized database system. However, this is not the case in Heterogeneous Distributed Database System. In this section, we focus on query decomposition and the generation of a QEP for postquery processing of subquery results to acquire the global query result.

It is important to comprehend the total cost of processing a global query, before trying to optimize it. The following equation is generally regarded as the formula for calculating the cost of processing a global query:

$$\text{Total Cost of Global Query} = \sum_i \text{Cost}(\text{Subquery}_i) + \text{Cost}(\text{Data Transmission}) + \text{Cost}(\text{Postquery Processing})$$

That is, the total cost of processing a global query is the sum of the costs of processing local queries, the costs of transmitting data across different databases, and the costs of postquery processing.

In order to estimate the cost of each subquery, certain cost parameters of the component databases are needed. Such information may not be available to the global system due to autonomy of the component database. This complicates the process of the global query optimizer which chooses a good strategy for executing a global query. Three methodologies for obtaining local cost parameters by the global query optimizer have been outlined in [128]:

- Performing some testing queries to test the black box;
- Guessing necessary information subjectively based on external characteristics of and previous knowledge about the black box;
- Monitoring the behavior of the black box at run time and dynamically collecting necessary information.

We can find discussion using the third approach in [73]. Techniques belonging to the first approach are explained in [38]. In [127], a query sampling method, belonging to the first group is presented (extended in [128]). For our purposes, we feel that using such

techniques outlined in literature and also developing a technique following the second approach is possible. Currently, our architecture integrates relational and Sem-ODM component data sources. If the global query optimizer is provided with information about the type of data source (i.e. relational database or Sem-ODM database), this information can be exploited for obtaining local cost parameters. Access methods used for Sem-ODB and relational databases are well-known. Hence, we feel that a technique to acquire local cost parameters, using the second approach is plausible. However, this is out of scope of this research, and in our discussion below, we assume that local cost parameters have already been acquired.

Reducing data transmission and postquery processing costs have been extensively discussed in distributed database query optimization techniques [124] and multidatabase query processing techniques. Optimizing through parallel execution of queries [113], considering schema conflicts of queries [66], optimizing through improved execution strategies and simplification of operators [24] are some existing methodologies. Incorporating these optimization techniques is beneficial for Semantic SQL query processing. However, none of these methodologies try to optimize queries based on semantic knowledge obtained during integration. To the best of our knowledge, we have not come across methodologies that use semantic knowledge (such as extents of constructs) for multidatabase query optimization.

In this section, we discuss some query optimization techniques, based on semantic knowledge acquired during integration (see chapter 3), for optimizing global Semantic

SQL queries. These techniques can be applied for optimizing relational SQL multidatabase queries without loss of generality.

Technique 1: During the integration process, semantic relations between constructs were distinguished. This information can be utilized in querying the minimum and most efficient set of component databases. To illustrate this technique, an example is presented:

Example 26: Let us consider the following scenario.

Global view: STUDENT(ssn:Integer, name:String)

Component Databases (DB_1, DB_2): DB_1.PUPIL(ssn:Integer, key; name:String)

DB_2.PERSON(ssn:Integer, key; name:Sting;

isStudent:Bool)

We have the following semantic relations:

STUDENT *SEM_EQ* DB_1.PUPIL (i.e. EXT(STUDENT) = EXT(PUPIL))

STUDENT *SEM_SUB* DB_2.PERSON (i.e. EXT(STUDENT) $\subseteq$ EXT(PERSON))

Let us assume that category STUDENT is derived from categories DB_1.PUPIL and DB_2.PERSON (with condition isStudent = TRUE) and attribute STUDENT.name is derived from DB_1.PUPIL.name and DB_2.PERSON.name. Likewise for attribute *ssn*.

Let us consider the following query posed on the global view:

SELECT name FROM STUDENT

By considering the semantic relations, it is evident that querying either DB_1.PUPIL or DB_2.PERSON is sufficient to obtain the results for this query (by their extents). Hence, the more efficient component database is queried.

Assuming that accessing DB_1.PUPIL is efficient than DB_2.PERSON (estimated through local cost parameters), the above query is transformed into the following subquery to obtain the result for the global query:

SELECT DB_1.PUPIL.name FROM DB_1.PUPIL

However, if we do not consider the semantic relations (similar to previous approaches), it is not possible to determine if all objects of STUDENT are in DB_1.PUPIL or DB_2.PERSON. Hence, the query decomposition algorithm queries both component databases. The query execution plan not considering semantic relations is shown in figure 26.

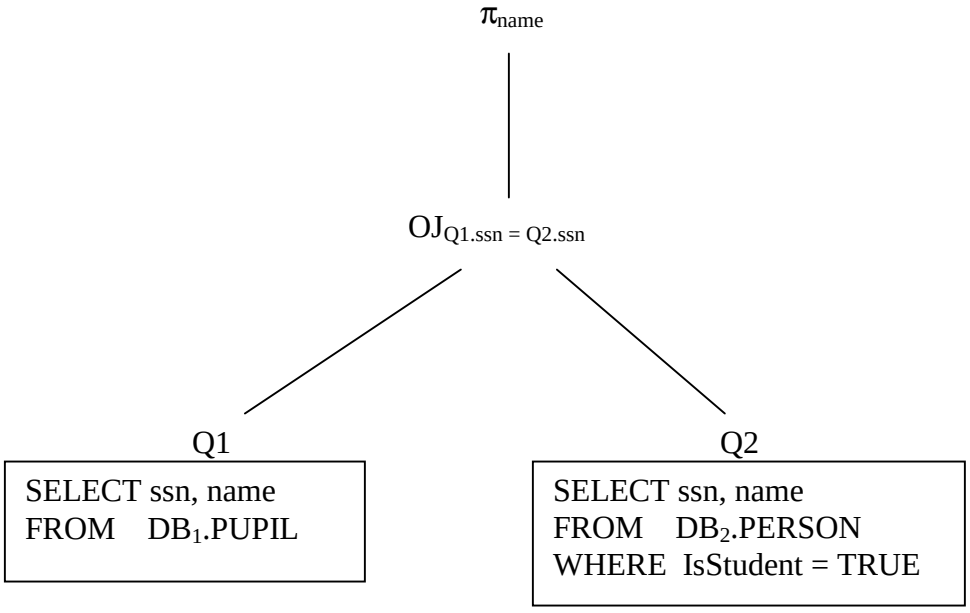


Figure 26. Query Execution Plan Without Considering Semantic Relations

In the above QEP, the subqueries obtaining information from component databases, DB_1 and DB_2, are shown as Q1 and Q2 respectively. At the next level, the common objects are eliminated using an outer-join operation. Finally (at the root), the attribute *name* is projected.

It is obvious that the improvement in query processing execution is significant due to the consideration of semantic relations. The above-example is a simple case. These improvements are applicable directly to the more general and complex queries. The fact that semantic relations provide information as to what objects of the global schema are present in which component databases allows intelligent decisions as to which information sources to query. A minimal and most efficient number of component data sources are queried. Another advantage is that the quality of the query results is improved. If the global query processor is unable to provide complete answers to global queries (say due to some component data sources withdrawing from participating in the global database system), it is possible to specify the missing data utilizing semantic relations and boundary conditions. This is not the case with other approaches.

Technique 2: In this technique, we transform complicated operations (such as NOT IN, EXIST, ALL, ANY, SOME, etc.) into simpler operations utilizing semantic knowledge. We illustrate transformation of NOT IN operation in the following example.

Example 27: Let us consider the following scenario.

Global schema: STUDENT(ssn:Number, key; name:String)

GRAD_STUDENT(ssn:Number, key)

Component database (DB_1): DB_1.PUPIL(ssn:Number, key; name:String)

DB_1.UNDERGRAD() subcategory of DB_1.PUPIL

DB1.GRAD() subcategory of DB_1.PUPIL

Assume the following semantic relations:

STUDENT *SEM_EQ* DB_1.PUPIL

GRAD_STUDENT *SEM_EQ* DB_1.GRAD

DB_1.GRAD *SEM_SUB* DB_1.PUPIL (by subcategory relation)

DB_1.UNDERGRAD *SEM_SUB* DB_1.PUPIL (by subcategory relation)

Assume the following boundary conditions (where all objects of PUPIL are either objects of UNDERGRAD or GRAD):

Boundary condition1: DB_1.PUPIL - DB_1.GRAD: is represented by DB_1.UNDERGRAD in the Knowledge Base

Boundary condition2: DB_1.PUPIL - DB_1.UNDERGRAD: is represented by DB_1.GRAD in the Knowledge Base

Let us consider the following global query:

```
SELECT name
FROM   STUDENT
WHERE  ssn NOT IN (SELECT ssn FROM GRAD)
```

Direct conversion of this query (without considering boundary conditions) will result in the following subquery:

```
SELECT name
FROM   DB1.PUPIL
```

WHERE ssn NOT IN (SELECT ssn FROM DB_1.GRAD)

Considering the boundary conditions, we can obtain the same result by the following subquery:

SELECT name FROM DB_1.UNDERGRAD

This was possible because the global query aims to obtain the names of STUDENTs who are not GRAD_STUDENTs. These objects are represented by boundary condition1. It is well known that processing operator NOT IN is significantly less efficient than processing a projection. Hence, the second option is a superior solution.

Technique 3: This technique optimizes processing of aggregate functions (in a multidatabase environment) utilizing semantic knowledge. The following example demonstrates the processing of *count* aggregate function. This methodology can be directly applied for other aggregate functions such as *avg*, *min*, *max*, *sum* etc.

Example 28: Let us consider the following scenario.

Global schema: STUDENT(StID:Number, key; name:String; GPA:Number)

GRADUATE(ThesisTitle:String) subcategory of STUDENT

UNDERGRAD() subcategory of STUDENT

Component databases (DB_1, DB_2):

DB_1.GRAD_STUDENT(StID:Number, key; name:String;

GPA:Number; ThesisTitle:String)

DB_2.UNDERGRADUATE(StID:Number, key; name:String;

GPA:Number)

Assume the following semantic relations:

I DB_1.GRAD_STUDENT *SEM_SUB* STUDENT

II. DB_2.UNDERGRADUATE *SEM_SUB* STUDENT

III. DB_1.GRAD_STUDENT *SEM_DIS* DB_2.UNDERGRADUATE

Let us assume that there exists an *INTEGRATED META OBJECT* in the knowledge base, which contains the union of objects of DB_1.GRAD_STUDENT and DB_2.UNDERGRADUATE, called X. Also, assume that STUDENT *SEM_EQ* X.

Let us assume that

(i.) STUDENT ← {DB_1.GRAD_STUDENT, DB_2.UNDERGRADUATE}

(ii.) GRADUATE ← {DB_1.GRAD_STUDENT}

(iii.) UNDERGRAD ← {DB_2.UNDERGRADUATE}

where ← means “is derived from”.

Now, we will consider the following global query:

```
SELECT    count(last-name)
FROM      STUDENT
```

If we directly translate this query, without considering semantic relations above, we obtain the following QEP (see figure 27):

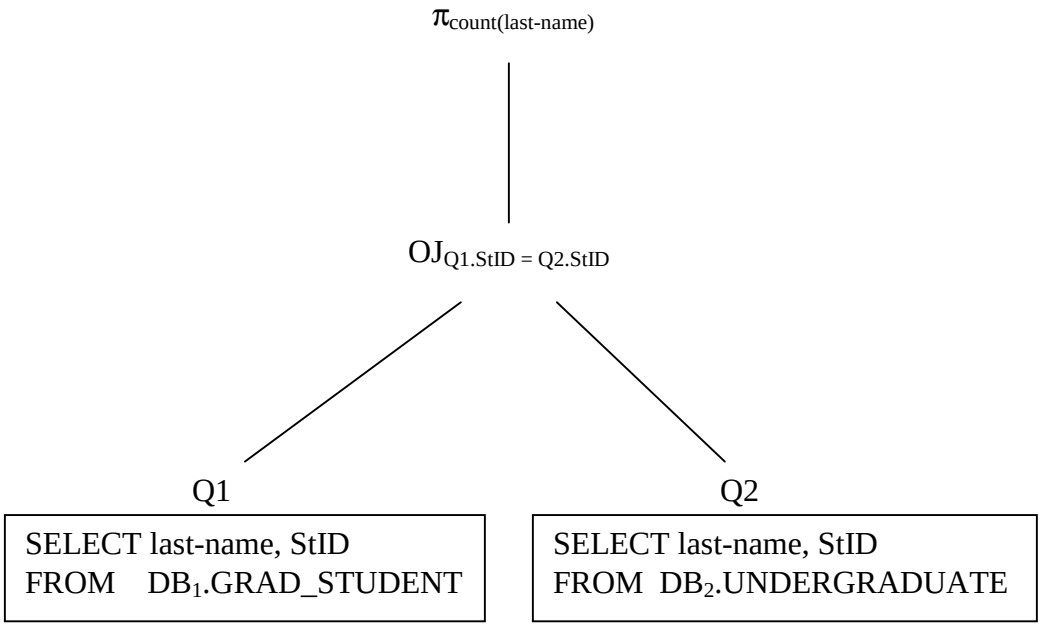


Figure 27. Query Execution Plan Without Considering Semantic Relations

Considering the semantic relations, we obtain the following QEP:

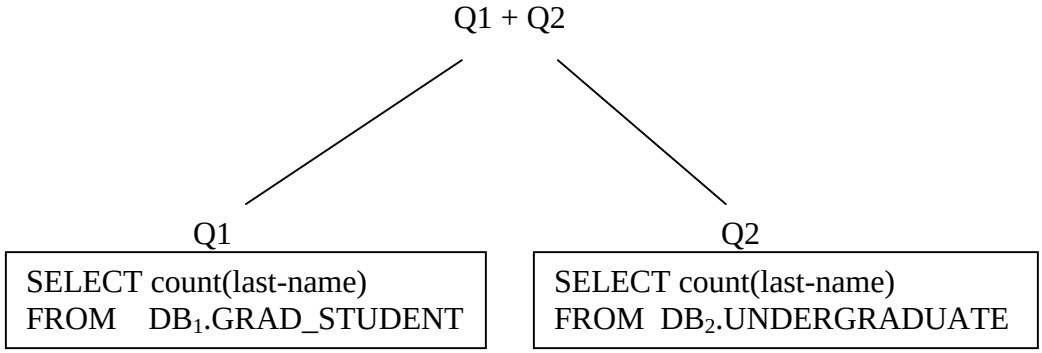


Figure 28. Query Execution Plan Considering Semantic Relations

Note that by considering that semantic relation III, it is provided that there are no common objects between DB_1.GRAD_STUDENT and DB_2.UNDERGRADUATE. This enables us to directly add the result of Q1 and Q2. This methodology has significantly reduced the postquery processing efforts and reduced cost of data transmission. This

method can be applied for other aggregate operators (such as SUM, AVG, MIN, MAX) without loss of generality.

The use of semantic knowledge for query optimizing in a multidatabase environment has significant potential for improving query performance. This aspect is not exploited in current multidatabase systems. It is possible to generate more optimization strategies for global queries in a multidatabase environment using semantic knowledge discussed in chapter 3. However, in this section we do not enumerate all the possible techniques, but will consider these factors in our future work. The next section discusses the query decomposition and creation of subqueries.

5.2.5 Step 5: Generating Subqueries

The final step in global query processing is to generate the subqueries in Semantic SQL from the query execution plan. The previous step obtained an optimized query execution plan (as an extended relational algebra expression) for the global query based on the component database constructs. The leaves of QEP refer to categories of the component database schema. The simplest methodology is to obtain the tables from database via select statements and allow the processing of the query at the Subquery Processor (see chapter 2). However, it is more efficient to incorporate the maximal number of operations into a single query of a component database which will be executed by the local query processor in the database engine (as this will allow physical optimization of queries according to internal representations and access paths by the local query optimizer) and process inter-database operations at the Subquery Processor. In

order to achieve this goal, we obtain the largest sub-tree in the QEP which references only categories in one component database and try to generate Semantic SQL statement(s) equivalent to the operations specified in the subtree (similar to the reverse of translation from SQL to algebra). It is possible that the subtree may not be converted to exactly one Semantic SQL query in which case a QEP to combine the results of the set of subqueries is generated. A minimal number of Semantic SQL queries are obtained for the QEP subtree. These queries along with the QEP for the subtree are combined into a single transaction before transmitting to the Subquery Processor of the component site. QEPs passed to the component sites may contain interdatabase operations which are processed by the Subquery Processor module of the component site. A goal in generating the subqueries is to place as many operations within one component database into Semantic SQL queries so as to minimize the postquery processing of the query results from a single component database at the Subquery Processor.

5.3 Summary

In this chapter, we first discussed some of the existing work in SQL query processing in a centralized database, query processing and optimizing in distributed databases, and existing work on query optimizing in multidatabase systems. Secondly, we discussed the different steps taken to process and optimize Semantic SQL queries. In the first step (section 5.2.1), the Semantic SQL query based on a global Sem-ODM schema was translated into an SQL statement over a set of virtual tables. This transformation allowed applying the existing approaches for SQL query processing to process and optimize Semantic SQL queries. In section 5.2.2, similar to SQL query processing in a

centralized relational DBMS, we first decompose the query statement into blocks. Next, each block is transformed into an extended relational algebra expression which is then optimized using well known logical rule-based optimizing techniques. In section 5.2.3, the virtual tables in the relational expression are further simplified. This allows further logical optimization. The optimized relational algebra expression (which corresponds to a QEP as well) still assumes the global view as a centralized database. The next step (section 5.2.4) is to decompose the query using derivation information in the Knowledge Base. Multiple query execution paths are possible. Hence, the application of the existing optimization strategies to select the optimal QEP is incorporated. An important aspect is the acquisition of cost parameters for component databases. Existing approaches and a proposal of a new approach for obtaining cost parameters in our architecture is suggested. The existing optimization techniques for multidatabases do not consider exploiting semantic knowledge for query optimizing. Techniques using semantic knowledge for multidatabase query optimizing are presented. Finally, generation of subqueries along with QEP for Subquery Processor module are discussed. Proposed future work includes investigating into techniques for obtaining cost parameters for component databases and enumerating all possible query optimization techniques based on semantic knowledge.

6. A FRAMEWORK FOR THE INTELLIGENT WEB

In this chapter, we will consider a framework for Internet computing and communication. This framework applies some of the concepts developed in the previous chapters into a Web environment. Some discussions into techniques and technologies presented in this chapter require further research for incorporating into the framework. We decided to discuss this framework as a separate chapter rather than integrate it to the concluding chapter due to the significant impact the framework has on the issue of intelligent information access on the Internet. It is clear that the Web (note that we use the terms Web, Net and Internet interchangeably) is influencing every aspect of society and deploying the framework presented aims at achieving the *Intelligent Web* - providing ubiquitous, intelligent access to information and services on the Web – a highly coveted and desirable goal.

Internet has become the chief medium of communication and interaction in the new economy. The Internet has driven all aspects of society to communicate and share information through its medium. We have seen business applications (e-commerce), communication channels (e-mail, newsgroups, etc.) and various other information and services moving on-line. The fact that it is accessible to a wide range of people worldwide and a convenient and cheaper means for communication have driven its exponential growth. It is quite probable that the Internet will affect almost all aspects of society worldwide in the near future.

Internet provides access to large amounts of heterogeneous information sources. Traditionally, large amounts of data and information have been stored in database systems and accessed via query languages in either centralized DBMSs or distributed environments (distributed databases and multidatabases). The Internet also provides access to large amounts of heterogeneous data. Hence, it is logical for database researchers to investigate into approaches to store and access this information. There are some significant differences between data stored in database systems (whether centralized or distributed) from data and information present on the Internet:

1. Structured vs. Unstructured: In general, databases systems provide a schema in a single data model. The schema describes the data content in a database. Also, it provides a structure to the stored data and information so as to easily access this information. In the Internet environment, no such schema exists. The information present in the Internet is not known. There exists little or no structure at all for information on the Web. Data and information are loosely coupled, distributed via links and change dynamically without prior notification.
2. Access methods: In database systems, data is stored in a certain format and structure with indexes for fast retrieval. Also, easy-to-use query languages (such as SQL) are provided to the user to access data and information easily and efficiently. For information of the Web, there does not exist any query language and the only means of access is with the use of Web addresses (also called Uniform Resource Locators – URLs for short). Data is accessed either through links or directly inputting the address.

3. Homogeneous vs. Heterogeneous: Data and information in a database system is uniform according to specification of the schema. That is, in general, data types, either system generated or user generated, are specified and query results conform to these types. Internet is a heterogeneous environment. Information is presented in different languages, formats, and contexts without any relations between them.
4. Centralized vs. distributed: In database systems, there is some point of centralized control, either global site (in distributed and multidatabase systems) or system catalogs (in centralized databases), which contain schema and relevant information. The Web is a truly distributed environment. There is no centralized control.

Intelligent ubiquitous access to data, information and services on the Web is a highly desirable goal. Recently, we have seen a number of approaches, addressing the issues mentioned above, in database and information retrieval community. We discuss some of the proposed approaches in section 6.1 including their benefits and limitations. In section 6.2, we discuss our approach. A framework that provides ubiquitous intelligent access to data and information on the Net is proposed. The advantages of the framework from proposed approaches are illustrated. A look into the future Internet applications based on this framework is provided. In section 6.2.3, we investigate into some future research issues and areas that need to be focused to achieve ubiquitous intelligent access to information sources on the Internet in the context of the framework presented.

6.1 Related Work

Recently, we have seen a surge of efforts in trying to access information and data from the Web. From these efforts, we have classified three major approaches which are detailed below:

1. Wrapper Approach: This approach was introduced with the TSIMMIS project ([23] [44], [49], [86]) at Stanford University. In this approach, wrappers are built over data sources to provide structure to the data sources (in TSIMMIS the structure obtained is a schema in Object Exchange Model - OEM). Mediators are used to query different data sources via wrappers (in TSIMMIS, OEM-QL is a query language used to query the OEM schema). Many efforts following a similar approach, proposing either manual ([8], [48]) or semi-automated/fully automated ([3], [5], [43], [60], [108]) wrapper generation has been discussed in literature.
2. Query System Approach: This approach tries to provide high-level declarative query facilities to the Web. Adoption of SQL-like query language, with extension, for Web-based queries is presented in [57]. Other efforts include [47], [61], [77], [78] and others.
3. Keyword Searching: This is the most popular approach to retrieving information on the Web. Users use keywords as input parameters to search relevant items on the web. Search engines (such as Altavista [4], Lycos [75] and others) facilitate searches through maintaining indexes of interesting keywords. Usually, the indexes are constructed and maintained by robots that occasionally scan the Web.

Some major issues that must be addressed by every approach for intelligent information retrieval and communication on the Web include: (i.) extendibility; (ii.)

flexibility; (iii.) ability to handle dynamically changing data sources; (iv.) provide easy-to-use query facility; and (v.) provide, preferably, semantically enhanced intelligent information extraction mechanisms.

The first approach (i.e. Wrapper Approach) tries to provide a well-known database interface with a schema and query language to the Internet. Although, this would be a favorable goal, there are certain drawbacks of this architecture in a Web environment. Firstly, the wrapper approach is most certainly faces the limitations on the extendibility issue. Wrapping every Web data source, which is loosely structured, is impossible and impractical. Also, in most cases the data sources change dynamically which requires in some cases changes in wrappers' mapping. Although certain methodologies that automate these processes have been described in literature, we feel that wrapper-based approach is not feasible as a general solution for providing ubiquitous intelligent access to any Web source available; rather, used in integrating and querying certain types of known Web data sources.

The second approach (i.e. Query System Approach) has an advantage where the data sources need not be wrapped. Dynamic changes of Web pages are handled easily. Hence, arbitrary queries can be programmed on any Web page. However, a major limitation of this approach is that a "starting point" (and complex programming) is usually required. For instance, given that the user needs information about company A and (s)he knows the company's web address (i.e. URL), the user can program a query to obtain the required information from the Web site of the company. However, this

approach lags in the fourth aspect (i.e. semantically enhanced intelligent information extraction mechanisms – semantic heterogeneity resolution), which means to find the initial Web page of the company (i.e. starting point) (such as which company's sell product P, rather than find the products sold by company A given the homepage of company A). Thus, we feel that this approach also has limited scope in providing ubiquitous intelligent access to information on the Internet.

The third approach (i.e. Keyword Searching) is the most promising and widely used approach for information extraction on the Internet today. However, it is well known that this approach is still naïve in terms of semantic and intelligent means of retrieving information from the Web. In this chapter, we provide a framework for ubiquitous, intelligent information extraction from the Web by applying some of techniques developed in the previous chapters.

6.2 Our Work

In order to provide ubiquitous intelligent access to information and services on the Internet, we propose a framework for the Web. This framework extends the current architecture of the Internet providing intelligent means of access to data and information. We discovered the problem of semantic heterogeneity recurring impeding intelligent access to heterogeneous data sources on the Web similar to integration of heterogeneous databases. Our proposal considers methods discussed previously in chapter 3, to the Web environment. The proposed framework is discussed in section 6.2.1. In addition, a communications paradigm (for e-commerce applications) using the framework is

illustrated in this section. A retrospective look into future applications using our framework is presented in section 6.2.2. Finally, some future research issues to enable the proposed *Intelligent Web* are discussed in section 6.2.3.

6.2.1 Framework for the Internet

In this section, we present a framework for the Internet that enables intelligent ubiquitous data access on the Web. We try to extend the third approach (i.e. Keyword Searching) to provide intelligent access to data on the Web. This is because we feel that this approach is the only solution, among the three approaches, that can successfully overcome the extendibility problem. The overall view of the proposed framework is presented in figure 29. It is important to note that in this framework, the search engines are extended to become *Information Brokers* (IB) rather than just searching keywords. We apply our approach to resolving semantic heterogeneity (discussed in chapter 3) for storage of information at the Information Broker nodes. A Knowledge Base with the use of domain specific ontologies is proposed as a means of storing information. The keywords are mapped on to the ontologies. Of course, it is unreasonable to expect all information of the Web to be mapped into a single ontology. Hence, sets of Information Brokers considering different application domains for information on the Internet are proposed. Note that there may be overlap between different application domains. Thus, IBs collaborate within themselves to provide ubiquitous intelligent access to data and information. The architecture is extendable with the addition of different information brokers and domain ontologies. Also, the semantic heterogeneity is resolved through the use of ontologies. Robots, similar to indexing in a search engine scan and map

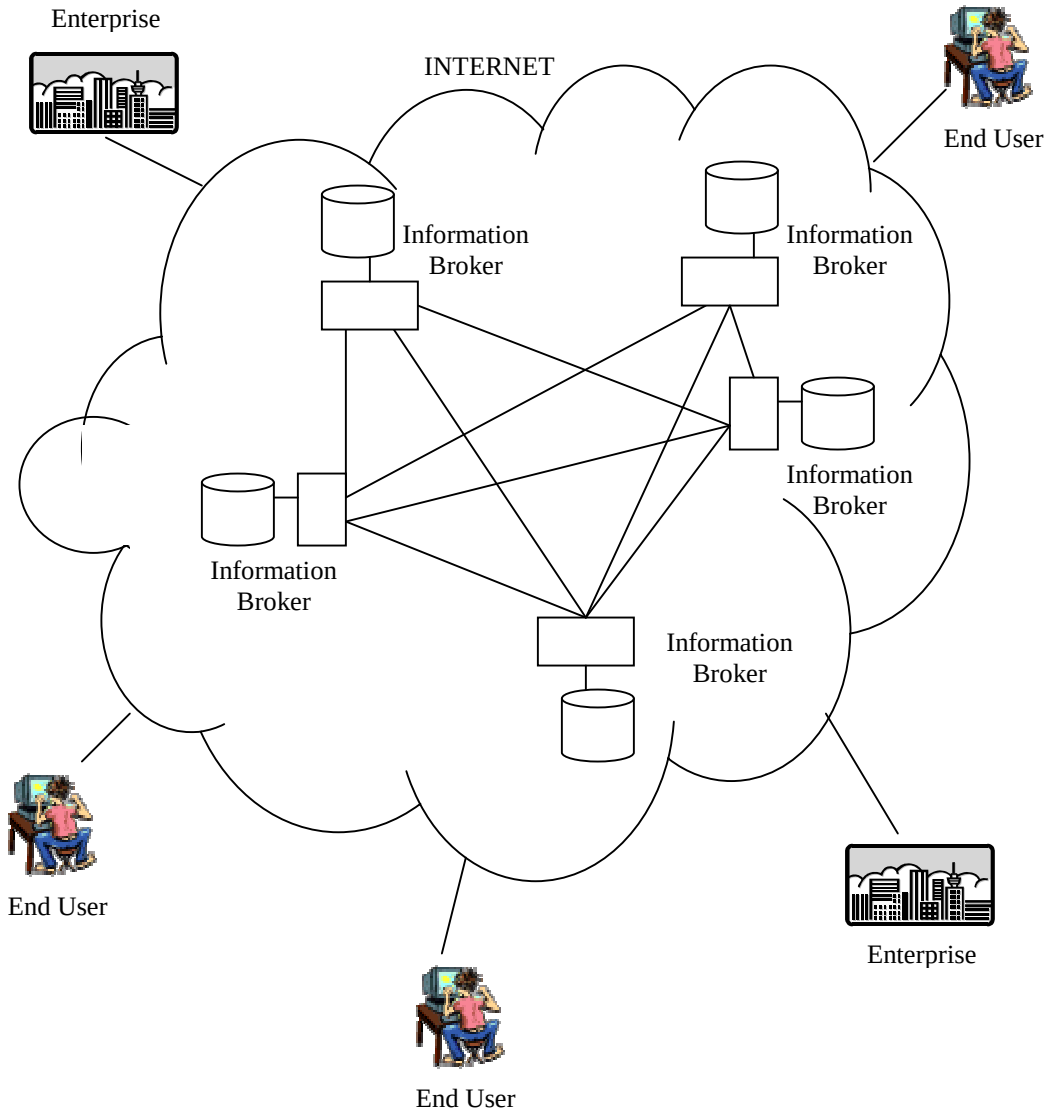


Figure 29. Overall View of Proposed Framework for the Internet

information to the ontology. Humans may also intervene in the process. Also, information extraction methodologies [29] are used in populating the Knowledge Base and instance mappings of the ontology. An important aspect is that *enterprises* map their services to the ontology of the Information Brokers by themselves. Enterprises (such as governments', companies, etc.), which define the services they provide, would like to publish the existence of their services on the Web. Hence, a very desirable place to

publish their services would be Information Brokers where most of traffic on the Web is directed. This relieves the Information Brokers from the enormous task of mapping services to the ontology. These ontologies with their mappings provide a great resource for all users of the Internet. All information needs can be passed through these nodes providing intelligent data access. In the next part of this section, we propose a model for e-commerce applications using the framework described before.

We define e-commerce applications broadly where a user requests for some service on the Internet. By service, we mean a well-defined requirement such as buying an airline ticket, transferring the title of car, renewing a passport, etc. rather than a general searching such as search on “cars” keyword. The model for specifying and fulfilling services is provided in figure 30. The user’s request is translated into an

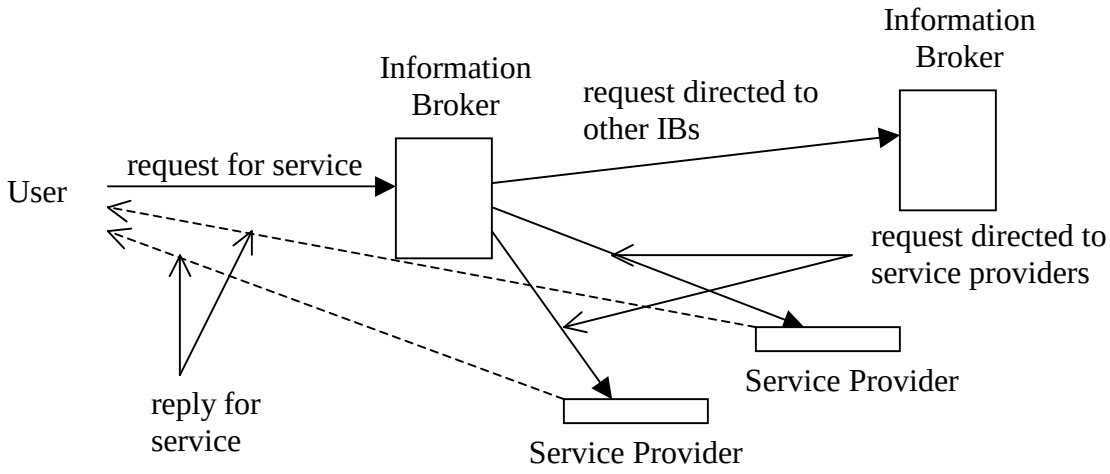


Figure 30. A Model for E-Commerce Applications on the Web

ontological form of request (via the browser or client program at the user’s computer) and transmitted to an Information Broker. The Information Broker searches the ontology for relevant matches. The matched service providers are informed of the request, which in

turn replies to the user of their service along with specifications for fulfilling the request. In addition, the Information Broker forwards the request to other Information Brokers that it feels contains relevant service provider information for the request.

Now let us consider a simple example of an e-commerce application in this framework.

Example 29: A user would like to buy an airline ticket to fly from 'Miami' to 'Los Angeles' on 'July 14, 2000'. This request is converted to the appropriate format and transmitted to an Information Broker. The Information Broker contains services from different airlines that sell tickets within the US, say 'American Airlines' and 'United Airlines' in this example. The request to buy the ticket from 'Miami' to 'Los Angeles' on 'July 14, 2000' is forwarded to American Airlines and United Airlines Web sites. Also, this request may be forwarded to other Information Brokers as deemed required. The American Airlines service provider and United Airlines service provider (i.e. Web sites) replies to the request to the user's site. The reply may include the fares, flight numbers, information needed to buy the ticket such as (credit card etc.), encryption information (such as encryption keys for security purposes etc.) and other relevant information to the request. The users next can choose his/her preference to which airline (s)he would prefer and buy the airline ticket. Next, the user's selected response is (if needed encrypted as specified) and transmitted to the service provider to purchase the ticket. If the user is a frequent flyer and purchases tickets frequently, (s)he can store the service provider information at the client so that the next time, (s)he needs a ticket, a request to the Information Broker can be avoided by directly contacting the Airline Service Provider.

The above example is a simple e-commerce application, however, provides some significant insights into the framework and model proposed. This example illustrates the issue of (i.) extensibility – through the use of multiple Information Brokers; (ii.) flexibility – different service providers may require different customized responses to certain services (such as American Airlines requiring social-security number for ticket purchasing while this information is not required by United Airlines). These aspects are incorporated the response agents of every service provider enabling flexibility; (iii.) handle dynamic changes. For instance, a change of the service is easily incorporated at the service site, transparent to the user or IBs; (iv.) intelligent access – through the use of ontologies and their mappings; and (iv.) easier query facility. This is an area for future research, where techniques providing easier means to specify a user's request and services in an ontological form are needed.

In the general case, this framework has significant implications on intelligent access to services and information on the Web. The next section looks at a hypothetical scenario, which illustrates the model's implications on future Internet applications.

6.2.2 A Futuristic View of E-Commerce Applications

In this section, we describe an extended example of a scenario of future web applications on the Internet. Let us consider our Web Application (which may be an enhanced Web browser). The user has the following requests:

1. Register for courses at Florida International University (FIU). The URL of FIU is known.
2. Buy an airline ticket to travel from Miami to Seattle. This is a general request where the user tries to obtain the cheapest ticket.
3. Renew the tag of user's vehicle from Florida Department of Motor Vehicle (FDMV). The user does not know URL of Florida Department of Motor Vehicle.
4. Buy automobile insurance for six months from AAA Auto Club South.

The user specifies his/her requests using the Web Application, which translates the requests in terms of ontological format and transmits the request as described below. Each of the requests are integrated into a mobile agent and transmitted to the appropriate destinations.

- Request 1: In request 1, the URL of FIU is known. Hence, the agent for request 1 will travel to the FIU's web site. The service provider of FIU matches the request with the appropriate service which in turn spawns a reply agent that specifies the required information for registering to courses at FIU (such as student id, course reference numbers, pin of student, encryption information – keys etc.) and transmits it to the user. The user next inputs all the required information (in a pre-defined form of the reply agent) and re-transmits the agent to register for the courses.
- Request 2: The second request agent is transmitted to an Information Broker (as this is a general request without prior knowledge by the user about airlines he/she prefers etc.). The request and its response will be directed to the Information Brokers similar to the method described in example 29.

- Request 3: In this request the URL of the Florida Department of Motor Vehicle is not known but the user is aware that the service is provided by FDMV. This request is firstly transmitted to the Information Broker (to find the web site of FDMV) which either directs the request to FDMV's web site or to another Information Broker capable of finding the FDMV web site. At the web site, the service provider of FDMV matches the request with the service required. Similar to request 1, the reply agent containing the necessary information (such as a form which requires VIN# of vehicle, payment information, etc.) to renew the tag is sent to the user. The user next fills the necessary information required for tag renewal and retransmits an agent to FDMV service provider to renew the user's tag.
- Request 4: Due to the fact that the user does not know URL of AAA Auto Club South, similar to request 3, the request agent is transmitted to an Information Broker. A similar communication and transmission pattern to request 3 occurs in fulfilling request 4.

Note that once the requests are satisfied with responses, the user fills the required information and an agent is transmitted to the service provider directly to fulfill users requirements. The user is capable of saving the service information for later use (so as to not re-apply the effort in finding the services).

The communication and transmission of requests and reply agents is transparent to the user, which is performed in an automated way (a highly desirable goal since many users' spend much time and effort surfing the Web to find the required information and services manually in the current framework on the Internet. Usually, it is by serendipity

that the user finds the required information). The proposed framework is an extension of the current search engines of the Internet, thus providing a smooth transition to the Information Broker framework. The framework and model for e-commerce applications is flexible, extendable and fulfills the requirements of intelligent ubiquitous information access and computing on the Internet. In summary, this proposal is a transition from the current Web to the *Intelligent Web*. As with any endeavor, there are certain technical hurdles to overcome, in order to achieve the Intelligent Web. However, we feel that if the focus of research and technology directions aims in achieving this goal (i.e. this framework), Intelligent Web is a near possibility. These research issues and technology directions are discussed briefly in the next section.

6.2.3 Future Research Issues and Technology Directions

A significant challenge to developing Information Brokers is to consider design and development of ontologies for general-domains on the Internet. We have seen a project named WebKB ([30], [121]) at Carnegie Mellon University which aims at developing such ontologies. However, we feel that further research projects and efforts are required in this direction. Methodologies to easily translate any user request into an ontological form are needed. Research into linguistics considers this issue. Standardization of interfaces among Information Brokers and between Service Providers needs to be evolved. Currently industry standards such as Extensible Markup Language (XML) for data transmitting, CORBA for distributed application development, agent communication languages (such as Knowledge Interchange Format - KIF and Knowledge Query Manipulation Language - KQML) and Java as a preferable language

for application development on the Web are emerging technologies that can be utilized in implementing the framework. These technologies need to be investigated in the context of the framework presented. In terms of technological advances supporting e-commerce applications, we have seen many directions: (i.) the concept of *e-money* for transactions of the Internet is being pursued in the industry; (ii.) new encryption and security mechanisms of transmission of data over the Net is a very active research area; (iii.) electronic signatures for e-commerce transactions have been legalized in the US recently; (iv.) emergence of XML-based frameworks such as eCo [114] for e-commerce; and others. We feel that it is important to view and utilize these emerging technologies and techniques within the context of the framework discussed in order to provide intelligent computing and communication paradigm over the Internet. This is certainly a fruitful endeavor for the future research projects.

6.3 Summary

A highly desirable goal of the Internet today is to have intelligent access to information and services on the Net. The currently proposed and existing approaches in the database area in terms of query languages and wrappers fail to accomplish ubiquitous, intelligent access to information on the Internet in certain areas such as extendibility and intelligent access. The most widely used approach for information access on the Web is keyword searching. This methodology is criticized for their failure to provide intelligent access to information. Hence, in this chapter, we extended the current Internet framework consisting of centralized search engines to a set of collaborating Information Brokers. Information Brokers contain a Knowledge Base and domain specific ontologies.

Information and services on the Web are mapped to the ontologies. This brings about intelligent access to information on the Web. The service providers on the Internet map their services into the Information Brokers as a means of publishing the existence of their services. A model for e-commerce applications is proposed which utilizes the proposed framework. This method is flexible, extendable and provides intelligent access to information and services. A futuristic view of e-commerce applications using this model was described. Also, some research issues and technology directions that can be incorporated to achieve the goal of an *Intelligent Web* were briefly discussed.

7. CONCLUSION

In this chapter, we summarize the contributions of this thesis, including some future work. At the end of each chapter, a summary of contributions and future work has been discussed in detailed. Hence, in this chapter, we briefly mention the contributions and the chapters that discuss these areas. It is recommended for the reader to refer to the summary at the end of each chapter for detail discussion.

The main contributions of this thesis work include:

1. Architecture for multidatabase systems: A flexible, scalable, extendable and easy-to-develop architecture for developing a multidatabase system, utilizing Semantic Binary Object-oriented Data Model (Sem-ODM) and Semantic SQL query language was presented in chapter 2. The use of Sem-ODM for accessing heterogeneous data sources provided expressive data modeling capabilities to heterogeneous distributed data sources capturing the meaning of the information integrated. Semantic SQL query language provided an easier well-known query facility for heterogeneous data access.

- Semantic heterogeneity resolution methodology: A major impediment for the ubiquitous use of multidatabase technology is the difficulty in resolving semantic heterogeneity between data sources. Previous approaches to semantic heterogeneity based on heuristic approaches leads to incorrect integration and querying. A semi-automated complete, correct and unambiguous methodology based on extents of meta-data constructs to resolve semantic heterogeneity of heterogeneous information systems was presented in chapter 3.

- Schematic heterogeneity resolution techniques: In resolving schema-level conflicts (see chapter 4), we provided the following contributions: (i.) developed a language called SemOSQL/M to define global Sem-ODM views over component schemas; (ii.) enumerated of all possible schema-level conflicts and their resolutions in defining global Sem-ODM views; (iii.) designed Knowledge Bases to store and retrieve semantic knowledge and meta-data; and (iv.) developed a tool that assists in creating global views.
- A superior integration methodology: The integration methodology incorporates semantic knowledge used for semantic heterogeneity resolution with schema-level conflict resolutions. This approach has the following advantages: (i.) ideal solution to heterogeneous data access problem by providing global views for users' accessing multiple heterogeneous data sources in an expressive data model and query language (similar to centralized database systems); (ii.) avoids the overhead in maintaining a single global schema; (iii.) completeness and correctness of the queries is preserved through non-heuristic approaches to semantic heterogeneity resolution; (iv.) automated semantic heterogeneity resolution and easier global view definition. This is a significant improvement from existing approaches; (v.) a step-wise process to acquire semantic knowledge rather than at the beginning of integration cycle and (vi.) ability to handle dynamic changes of the underlying data sources. This integration methodology is discussed in chapters 3 and 4.
- Query processing and optimization: The plethora of well-defined existing methodologies to process and optimize relational SQL queries were exploited by transforming the Semantic SQL query based on Sem-ODM schema to a SQL query

based on virtual tables (discussed in section 5.2.1). A strategy to obtain local cost parameters from component data sources is suggested. Techniques for optimizing Semantic SQL queries exploiting semantic knowledge acquired during integration were presented. Query processing and optimizing of Semantic SQL queries in a multidatabase environment are discussed in chapter 5.

- A framework for intelligent computing and communication on the Web: A highly desirable goal of the current Web-based environment is intelligent access to information and services on the Web. A framework applying the semantic heterogeneity resolution techniques (developed in chapter 3) to provide a framework for the Internet in achieving intelligent ubiquitous computing and communication was presented in chapter 6.

Future areas of work include: (i.) ontology development for general problem domains and application in heterogeneous database environments; (ii.) extended rules for identification of semantic relations; (iii.) deployment of query optimizing techniques based on semantic knowledge in multidatabase environments and empirical testing on performance gains; (iv.) investigation of different innovative techniques for query optimizing using semantic knowledge; (v.) research into ontological based service-specification and query techniques on the Internet; (vi.) investigation of application of current technologies in the framework proposed for intelligent access to information on the Web; and (vii.) development of easier human-computer interaction modes for easy specification of user's requests (as well as services) in ontological formats – that is, capturing the meaning (semantics) of the user's request in a flexible and easier manner.

LIST OF REFERENCES

[1] Abiteboul S. and Bonner A., "Objects and Views". In *Proceedings of the ACM International Conference of Management of Data* (*SIGMOD'91*), 1991, pp.238-247.

[2] Access web site: http://www.microsoft.com/office/access/default.htm

[3] Adelberg B., "NoDoSE – A Tool for Semi-automatically Extracting Structured and Semistructured Data from Text Documents". In *Proceedings of the ACM International Conference of Management of Data* (*SIGMOD'98*), 1998, pp.283-294.

[4] Altavista Search Engine: http://www.altavista.com

[5] Ashish N. and Knoblock C., "Wrapper Generation for Semi-structured Internet Sources". In *SIGMOD Record*, Vol. 26, No. 4, 1997, pp.8-15.

[6] Aslan G. and McLeod D., "Semantic Heterogeneity Resolution in Federated Databases by Metadata Implantation and Stepwise Evolution". In *VLDB Journal*, Vol. 8, No. 2, 1999, pp.120-132.

[7] Athauda R., "Heterogeneity Resolution in MSemODB", Technical Report 2000-02, School of Computer Science, Florida International University, 2000.

[8] Atzeni P. and Mecca G., "Cut and Paste". In *Proceedings of the Symposium on Principles of Database Systems* (*PODS '97*), 1997, pp.144-153.

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

[10] Bayardo R.J., Bohrer W., Brice R., Cichocki A., Fowler J., Helal A., Kashyap V., Ksiezyk T., Martin G., Nodine M., Rashid M., Rusinkiewicz M., Shea R., Unnikrishnan C., Unruh A. and Woelk D., "InfoSleuth: Agent-Based Semantic Integration of Information in Open Dynamic Environments". In *Proceedings of the ACM International Conference on Management of Data* (*SIGMOD '97*), 1997, pp.195-206.

[11] Bernstein P.A., Goodman N., Wong E., Reeve C.L. and Rothnie J.B., "Query Processing in a System for Distributed Databases (SDD-1)". In *ACM Transactions on Database Systems* (*TODS*), Vol. 6, No. 4, 1981, pp.602-625.

[12] Beryoza D., "Dynamic Data Retrieval on the World Wide Web". Unpublished Ph.D. Thesis, School of Computer Science, Florida International University.

[13] Blakeley J., “OQL[C++]: Extending C++ with an Object Query Capability”. In *Modern Database Systems: The Object Model, Interoperability, and Beyond*, 1995, ACM Press, pp.69-88.

[14] Bouzeghoub M. and Métais E., “Semantic Modeling of Object Oriented Databases”. In *Proceedings of the International Conference on Very Large Data Bases* (*VLDB ’91*), 1991, pp.3-14.

[15] Breitbart Y. and Reyes T., “Overview of ADDS System”. In *Modern Database Systems: The Object Model, Interoperability and Beyond*, ACM Press, 1995, pp.683-701.

[16] Breitbart Y. and Tieman L., “ADDS – Heterogeneous Distributed Database System”. In *Proceedings of the International Seminar on Distributed Data Sharing Systems*, 1984, pp.7-24.

[17] Breitbart Y., Olson P. and Thompson G., “Database Integration in a Distributed Heterogeneous Database System”. In *Proceedings of the IEEE International Conference on Data Engineering* (*ICDE ’86*), 1986, pp.301-310.

[18] Bright M.W., Hurson A.R. and Pakzad S., “Automated Resolution of Semantic Heterogeneity in Multidatabases”. In *ACM Transactions on Database Systems* (*TODS*), Vol. 19, No. 2, 1994, pp.212-253.

[19] Brill D., Templeton M. and Yu C.T., “Distributed Query Processing Strategies in Mermaid, A Frontend to Data Management Systems”. In *Proceeding of the IEEE International Conference on Data Engineering* (*ICDE ‘84*), 1984, pp.211-218.

[20] Brown P., “Implementing the Spirit of SQL-99”. In *Proceedings of the ACM International Conference of Management of Data* (*SIGMOD ’99*), 1999, pp.515-518.

[21] Bunge M. A., *Treatise on Basic Philosophy: Vol. 3: Ontology I: The Furniture of the World*, Reidel, Boston, 1977.

[22] Bunge M.A., *Treatise on Basic Philosphy: Vol. 4: Ontology II: A World of Systems*, Reidel, Biston, 1979.

[23] Chawathe S., Garcia-Molina H., Hammer J., Ireland K., Papakonstantinou Y., Ullman J. and Widom J., “The TSIMMIS Project: Integration of Heterogeneous Information Sources”. In *Proceedings of IPSJ Conference*, 1994, pp.7-18.

[24] Chen A.L.P., “Outerjoin Optimization in Multidatabase Systems”. In *Proceedings of the International Symposium on Databases in Parallel and Distributed Systems* (*DPDS ‘90*), 1990, pp.211-218.

[25] Chung C.W., “DATAPLEX: An Access to Heterogeneous Distributed Databases”. In *Communications of the ACM* (*CACM*), Vol. 33, No. 1, 1990, pp.70-80.

[26] Codd E.F., “A Relational Data Model for Large Shared Data Banks”. In *Communications of the ACM* (*CACM*), Vol. 13, No. 6, 1970, pp.377-387.

[27] Codd E.F., “Extending the Database Relational Model to Capture More Meaning”. In *ACM Transactions on Database Systems* (*TODS*), Vol. 4, No. 4, 1979, pp.397-434.

[28] Collet C., Huhns M.N. and Shen W.-M., “Resource Integration Using a Large Knowledge Base in Carnot”. In *IEEE Computer*, Vol. 24, No. 12, 1991, pp.55–62.

[29] Cowie J. and Lehnert W., “Information Extraction”. In *Communications of the ACM* (*CACM*), Vol. 39, No. 1, 1996, pp.80-91.

[30] Craven M., DiPasquo D., Freitag D., McCallum A., Mitchell T., Nigam K. and Slattery S., “Learning to Extract Symbolic Knowledge from the World Wide Web”. In Technical Report, CMU-CS-98-122, School of Computer Science, Carnegie Mellon University, 1998.

[31] Czejdo B., Rusinkiewicz M. and Embley D., “An Approach to Schema Integration and Query Formulation in Federated Database Systems”. In *Proceedings of the IEEE International Conference on Data Engineering* (*ICDE ’87*), 1987, pp.477-484.

[32] Date C. J., *An Introduction to Database Systems*, 7th edition, Addison-Wesley, 1999.

[33] Dayal U and Hwang H., “View Definition and Generalization for Database Integration in a Multidatabase System”. In *IEEE Transactions on Software Engineering* (*TSE*), Vol. 10, No. 6, pp.628-645.

[34] Dayal U., “Processing Queries Over Generalization Hierarchies in a Multidatabase System”. In *Proceedings of the International Conference on Very Large Data Bases* (*VLDB ’83*), 1983, pp.342-353.

[35] Dayal U., “Query Processing in Multidatabase System”. In *Query Processing in Database Systems*, Springer, 1985, pp.81-108.

[36] Dogac A., Dengu C., Kilic E., Ozhan G., Ozcan F., Nural S., Evrendilek C., Halici U., Arpinar B., Koksal P., Kesim N. and Mancuhan S., "METU Interoperable Database System". In *SIGMOD Record,* Vol. 24, No. 3, 1995, pp.56-61.

[37] Dogac A., Halici U., Kilic E., Ozhan G., Ozcan F., Nural S., Dengu C., Mancuhan S., Arpinar B., Koksal and Evrendilek C., "METU Interoperable Database System". In *Proceedings of the ACM International Conference on Management of Data* (*SIGMOD '96*), 1995, pp.552-552.

[38] Du W., Krishnamurthy R. and Shan M.-C., "Query Optimization in Heterogeneous DBMS". In *Proceedings of the International Conference on Very Large Databases* (*VLDB '92*), 1992, pp.277-291.

[39] Egyhazy C.J., Triantis K. P. and Bhasker B., "A Query Processing Algorithm for a System of Heterogeneous Distributed Databases". In *Distributed and Parallel Databases*, Vol. 4, No. 1, 1996, pp.49-79.

[40] Elmasri R. and Navathe S. B., *Fundamentals of Database Systems*, 2nd edition, Addison-Wesley, 1994.

[41] Elmasri R. and Navathe, "Object Integration in Logical Database Design". In *Proceedings of the IEEE International Conference on Data Engineering* (*ICDE '84*), 1984, pp.426-433.

[42] Elmasri R., Hevner A. and Weeldreyer, "The Category Concept: An Extension to the Entity-Relationship Model". In *Data and Knowledge Engineering* (*DKE*), Vol.1, No.1, 1985, pp.75-116.

[43] Embley D.W., Campbell D.M., Jiang Y. S., Liddle S.W., Ng Y.-K., Quass D. and Smith R.D., "Conceptual-Model-Based Data Extraction from Multiple-Record Web Pages". In *Data and Knowledge Engineering* (*DKE*), Vol. 31, No. 3, 1999, pp.227-251.

[44] Garcia-Molina H., Hammer J., Ireland K., Papakonstantinou Y., Ullman J., and Widom J., "Integrating and Accessing Heterogeneous Information Sources in TSIMMIS". In *Proceedings of the AAAI Symposium on Information Gathering*, 1995, pp.61-64.

[45] Garcia-Molina H., Ullman J. D. and Widom J., *Database System Implementation*, Prentice-Hall, 1999.

[46] Gu H., Perl Y., Geller J., Halper M., Liu L.-M. and Cimino J.J., "Representing the UMLS as an Object-oriented Database: Modeling Issues and Advantages". In *Journal of the American Medical Informatics Association* (*JAMIA*), Vol. 7, No. 1, 2000, pp.66-80.

[47] Guan T., Liu M. and Saxton L.V., "Structure-Based Queries over the World Wide Web". In *Proceedings of the International Conference on Conceptual Modeling* (*ER '98*), 1998, pp.107-120.

[48] Gupta A., Harinarayan V. and Rajaraman A., "Virtual Database Technology". In *SIGMOD Record*, Vol. 26, No. 4, 1997, pp.57-61.

[49] Hammer J., Garcia-Molina H., Ireland K., Papakonstantinou Y., Ullman J., and Widom J., "Information Translation, Mediation, and Mosaic-Based Browsing in the TSIMMIS System". In *Proceedings of the ACM International Conference on Management of Data* (*SIGMOD '95*), 1995, pp.483 - 483.

[50] Hayne S. and Ram S., "Multi-User View Integration System (MUVIS): An Expert System for View Integration". In *Proceedings of the IEEE International Conference on Data Engineering* (*ICDE '88*), 1988, pp.402-409.

[51] Kashyap V. and Sheth A., "Semantic and Schematic Similarities Between Database Objects: A Context-Based Approach". In *VLDB Journal*, Vol. 5, No. 4, 1996, pp.276-304.

[52] Kelley W., Gala S., Kim W., Reyes T. and Graham B., "Schema Architecture of the UniQSL/M Multidatabase System". In *Modern Database Systems: The Object Model, Interoperability and Beyond*, ACM Press, 1995, pp.621-648.

[53] Kent W., "Solving Domain Mismatch and Schema Mismatch Problems with an Object-Oriented Database Programming Language". In *Proceedings of the International Conference on Very Large Data Bases* (*VLDB '91*), 1991, pp.147-160.

[54] Kim W. and Seu J., "Classifying Schematic and Data Heterogeneity in Multidatabase Systems". In *IEEE Computer*, Vol. 24, No. 12, 1991, pp.12-18.

[55] Kim W., Choi I., Gala S.K., Scheevel M., "On Resolving Schematic Heterogeneity in Multidatabase Systems". In *Distributed and Parallel Databases*, Vol.1, No. 3, 1993, pp.251-279.

[56] Koh, J.L. and Chen L.P., "Integration of Heterogeneous Object Schemas". In *Proceedings of the International Conference on Entity Relationship Approach* (*ER '93*), 1993, pp.297-314.

[57] Konopnicki D. and Shmueli O., "W3QS: A Query System for the World-Wide Web". In *Proceedings of the International Conference on Very Large Databases* (*VLDB '95*), 1995, pp.54-65.

[58] Krieger D., T. Andrews, "C++ Bindings to an Object Database". In *Modern Database Systems: The Object Model, Interoperability, and Beyond*, 1995, ACM Press, pp.89-107.

[59] Krishnamurthy R., Litwin W. and Kent W., "Languages Features for Interoperable Databases with Schematic Discrepancies". In *Proceedings of the ACM International Conference on Management of Data* (*SIGMOD '91*), 1991, pp.40-49.

[60] Kushmerick N., Weld D.S. and Doorenbos R.B., "Wrapper Induction for Information Extraction". In *Proceedings of the International Joint Conference on Artificial Intelligence* (*IJCAI '97*), Vol. 1, 1997, pp.729-737.

[61] Lakshmanan L.V.S., Sadri F. and Subramanian I.N., "A Declarative Language for Querying and Restructuring the Web". In *Proceedings of the International Workshop on Research Issues in Data Engineering* (*RIDE '96*), 1996, pp.12-21.

[62] Lakshmanan L.V.S., Sadri F. and Subramanian I.N., "SchemaSQL – A Language for Interoperability in Relational Multi-database Systems". In *Proceedings of the International Conference on Very Large Databases* (*VLDB '96*), 1996, pp.239-250.

[63] Landers T. and Rosenberg R., "An Overview of Multibase". In *Proceedings of the International Symposium on Distributed Data Bases*, 1982, pp.153-184.

[64] Larson J.A., "Bridging the Gap Between Network and Relational Database Management Systems". In *IEEE Computer*, Vol. 16, No. 9, 1983, pp. 82-92.

[65] Larson J.A., Navathe S.B. and Elmasri R., "A Theory of Attribute Equivalence in Databases with Application to Schema Integration". In *IEEE Transactions on Software Engineering* (*TSE*), Vol. 15, No. 4, 1989, pp.449-463.

[66] Lee C. and Chen C.-J., "Query Optimization in Multidatabase Systems Considering Schema Conflicts". In *IEEE Transactions on Knowledge and Data Engineering* (*TKDE*), Vol. 9, No. 6, 1997, pp.941-955.

[67] Legaria C.G. and Rosenthal A., "Outerjoin Simplification and Reordering for Query Optimization". In *ACM Transactions on Database Systems* (*TODS*), Vol. 22, No. 1, 1997, pp.43-74.

[68] Levy A.Y., "Obtaining Complete Answers from Incomplete Databases". In *Proceedings of the International Conference on Very Large Data Bases* (*VLDB '96*), 1996, pp.402-412.

[69] Levy A.Y., Rajaraman A., Ordille J.J., "Querying Heterogeneous Information Sources Using Source Descriptions". In *Proceedings of the International Conference on Very Large Databases* (*VLDB '96*), 1996, pp.251-262.

[70] Li W.-S. and Clifton C., "Semantic Integration in Heterogeneous Databases Using Neural Networks". In *Proceedings of the International Conference on Very Large Data Bases* (*VLDB '94*), 1994, pp.1- 12.

[71] Litwin W. and Abdellatif A., "Multidatabase Interoperability". In *IEEE Computer*, Vol. 19, No. 12, 1986, pp.10-18.

[72] Litwin W., Mark L. and Roussopoulos N., "Interoperability of Multiple Autonomous Databases". In *ACM Computing Surveys*, Vol.22, No.3, 1990, pp.267-293.

[73] Lu H., Ooi B.-C. and Goh C.-H., "On Global Multidatabase Query Optimization". In *SIGMOD Record*, Vol. 21, No. 4, 1992, pp.6-11.

[74] Lu X., "A Semantic Wrapper Used in Heterogeneous Database Systems". Master's Thesis, School of Computer Science, Florida International University, 2000.

[75] Lycos Search Engine: http://www.lycos.com

[76] McLoed D. and Si A., "The Design and Experimental Evaluation of an Information Discovery Mechanism for Networks of Autonomous Database Systems". In *Proceedings of the IEEE International Conference in Data Engineering* (*ICDE '95*), 1995, pp.15-24.

[77] Mendelzon A.O. and Milo T., "Formal Model of Web Queries". In *Proceedings of the Symposium on Principles of Database Systems* (*PODS '97*), 1997, pp.134-143.

[78] Mendelzon A.O., Mihaila G.A. and Milo T., "Querying the World Wide Web". In *Proceedings of the International Conference on Parallel and Distributed Information Systems* (*PDIS '96*), 1996, pp.80-91.

[79] Meng W. and Yu C., "Query Processing in Multidatabase Systems". In *Modern Database Systems: The Object Model, Interoperability and Beyond*, ACM Press, 1995, pp.551-572.

[80] Meng W., Yu C.T. and Kim W., “A Theory of Translation From Relational Queries to Hierarchical Queries”. In *IEEE Transactions on Knowledge and Data Engineering* (*TKDE*), Vol. 7, No. 2, 1995, pp.228-245.

[81] *Microsoft ODBC 2.0 Programmer’s Reference and SDK Guide*, Microsoft Press, 1994.

[82] Motro A., “Integrity = Validity + Completeness”. In *ACM Transactions on Database Systems* (*TODS*), Vol. 14, No. 4, 1989, pp.480-502.

[83] Navathe S., Elmasri R. and Larson J., “Integrating User Views in Database Design”. In *IEEE Computer*, Vol.19, No. 1, 1986, pp.50-62.

[84] OMG’s web site: http://www.omg.org/

[85] Oracle database web site: http://www.oracle.com/database/

[86] Papakonstantinou Y., Garcia-Molina H. and Widom J., “Object Exchange Across Heterogeneous Information Sources”. In *Proceedings of the IEEE International Conference on Data Engineering* (*ICDE ‘95*), 1995, pp.251-260.

[87] Quillian M.R., “Semantic Memory”. In *Semantic Information Processing*, ed. by, Marvin Minsky, MIT Press, 1968, pp.227-270.

[88] Ramakrishnan R. and Gehrke J., *Database Management Systems*, Second Edition, McGraw-Hill, 2000.

[89] Rishe N., “A File Structure for Semantic Databases". In *Information Systems*, Vol. 16, 1991, pp.375-385.

[90] Rishe N., “A Methodology and Tool for Top-down Relational Database Design”. In *Data and Knowledge Engineering* (*DKE*), Vol. 10, pp.259-291, 1993.

[91] Rishe N., “Interval-based approach to lexicographic representation and compression of numeric data". In *Data and Knowledge Engineering* (*DKE*), Vol. 8, 1992, pp.339-351.

[92] Rishe N., A. Vaschillo, D. Vasilevsky, A. Shaposhnikov, S.-C. Chen, “A Benchmarking Technique for DBMS`s with Advanced Data Models”. To appear in the *Symposium on Advances in Databases and Information Systems* (*ADBIS-DASFAA 2000*), September 5-8, 2000.

[93] Rishe N., Athauda R., Yuan J. and Chen S.C., "Knowledge Management for Database Interoperability". Submitted to the *International Conference on Information Reuse and Integration* (*IRI 2000*), November 1 - 3, 2000.

[94] Rishe N., *Database Design: The Semantic Modeling Approach*, McGraw-Hill, 1992.

[95] Rishe N., Sun W., Barton D., Deng Y., Orji C., Alexopoulos M., Loureiro L., Ordonez C., Sanchez M. and Shaposhnikov A., "Florida International University High Performance Database Research Center". In *SIGMOD Record*, Vol. 24, No. 3, 1995, pp.71-76.

[96] Rishe N., Yuan J., Athauda R., Lu X. and Ma X., "SemWrap: A Semantic Wrapper over Relational Databases, with Substantial Size Reduction of User's SQL Queries". In *Proceedings of the International Conference on Extending Database Technology - Software Demonstrations Track* (*EDBT 2000*), 2000, pp.13-14.

[97] Rishe N., Yuan J., Athauda R., Lu X., Ma X., Vaschillo A., Shaposhnikov A., Vasilevsky D. and Chen S.C., "SemanticAccess: Semantic Interface for Querying Databases". To appear in the *International Conference on Very Large Databases* (*VLDB 2000*), September 10-14, 2000.

[98] Rishe N., "Semantic SQL". Internal Document, High-performance Database Research Center, School of Computer Science, Florida International University, 1998.

[99] Roantree M., Murphy J. and Hasselbring W., "The OASIS Multidatabase Prototype". In *SIGMOD Record*, Vol. 28, No. 1, 1999, pp.97-103.

[100] Roth M.T., Arya M., Haas L., Carey M., Cody W., Fagin R., Schwarz P., Thomas J. and Wimmers E., "The Garlic Project". In *Proceedings of the ACM International Conference on Management of Data* (*SIGMOD '96*), 1996, pp.557-557.

[101] Sciore E., Siegel M. and Rosenthal A., "Using Semantic Values to Facilitate Interoperability Among Heterogeneous Information Systems". In *ACM Transactions of Database Systems* (*TODS*), Vol. 19, No. 2, 1994, pp.254-290.

[102] Shan M., "Pegasus Architecture and Design Principles". In *Proceedings of the ACM International Conference on Management of Data* (*SIGMOD '93*), 1993, pp.422-425.

[103] Shan M.-C., Ahmed R., Davis J., Du W. and Kent W., "Pegasus: A Heterogeneous Information Management System". In *Modern Database Systems: The Object Model, Interoperability and Beyond*, ACM Press, 1995, pp.664-682.

[104] Shaposhnikov A., "Algorithms for Efficient Transaction Management and Consistent Queries in Client-Server Semantic Object-oriented Parallel Databases". Ph.D. Thesis, School of Computer Science, Florida International University, 1998.

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

[106] Sheth A.P., Larson J.A. Cornelio A. and Navathe S., "A Tool for Integrating Conceptual Schemas and User Views". In *Proceedings of the IEEE International Conference on Data Engineering* (*ICDE '88*), 1988, pp.176-183.

[107] Shipman D. W., "The Functional Data Model and the Data Language DAPLEX". In *ACM Transactions on Database Systems* (*TODS*), Vol. 6, No. 1, 1981, pp.140-173.

[108] Soderland S., "Learning to Extract Text-Based Information from the World Wide Web". In *Proceedings of the International Conference on Knowledge Discovery and Data Mining* (*KDD '97*), 1997, pp.251-254.

[109] SQL Server web site: http://www.microsoft.com/sql/default.htm

[110] SQL-92. ANSI Standard SQL language, 1992.

[111] Storey V.C., Chiang R.H.L., Dey D., Goldstein R.D. and Sundaresan S., "Database Design with Common Sense Business Reasoning and Learning". In *ACM Transactions on Database Systems* (*TODS*), Vol. 22, No. 4, 1997, pp.471-512.

[112] Storey V.C., Ullrich H. and Sundaresan S., "An Ontology for Database Design Automation". In *Proceedings of the International Conference on Conceptual Modeling* (*ER '97*), 1997, pp.2-15.

[113] Subramanian D.K. and Subramanian K., "Query Optimization in Multidatabase Systems". In *Distributed and Parallel Databases*, Vol. 6, No. 2, 1998, pp.183-210.

[114] Tanenbaum J., Chowdhry T. and Hughes K., "eCo System: An Internet Commerce Architecture". In *IEEE Computer*, Vol. 30, No. 5, 1997, pp.48-55.

[115] Tseng F.S.C., Chen A.L.P. and Yang W.-P., "Answering Heterogeneous Database Queries with Degrees of Uncertainty". In *Distributed and Parallel Databases*, Vol. 1, No. 3, 1993, pp.281-302.

[116] Ullman J. D. and Widom J., *A First Course in Database Systems*, Prentice-Hall, 1997.

[117] Vaschillo A., "A Semantic Paradigm for Intelligent Data Access". Ph.D. Thesis, School of Computer Science, Florida International University, 2000.

[118] Wand Y. and Storey V., "An Ontological Analysis of the Relationship Construct in Conceptual Modeling". In *ACM Transactions on Database Systems* (*TODS*), Vol. 24, No. 4, 1999, pp.494-528.

[119] Wand Y. and Wang R., "Anchoring Data Quality Dimensions in Ontological Foundations". In *Communications of the ACM* (*CACM*), Vol. 39, No. 11, 1996, pp.86-95.

[120] Wand Y. and Weber R., "An Ontological Model of an Information System". In *IEEE Transactions on Software Engineering* (*TSE*), Vol. 16, No. 11, 1990, pp.1282-1292.

[121] WebKB project's homepage: http://www.cs.cmu.edu/~WebKB/

[122] Weeldreyer, "Structural Aspects of the Entity-Category-Relationship Model". In Technical Report HR-80-250, Honeywell Computer Sciences Center, 1980, pp.17-38.

[123] Winston P.H., *Artificial Intelligence*, Second Edition, Addison-Wesley, 1984.

[124] Yu C.T. and Chang C.C., "Distributed Query Processing". In *ACM Computing Surveys*, Vol.16, No.4, 1984, pp.399-433.

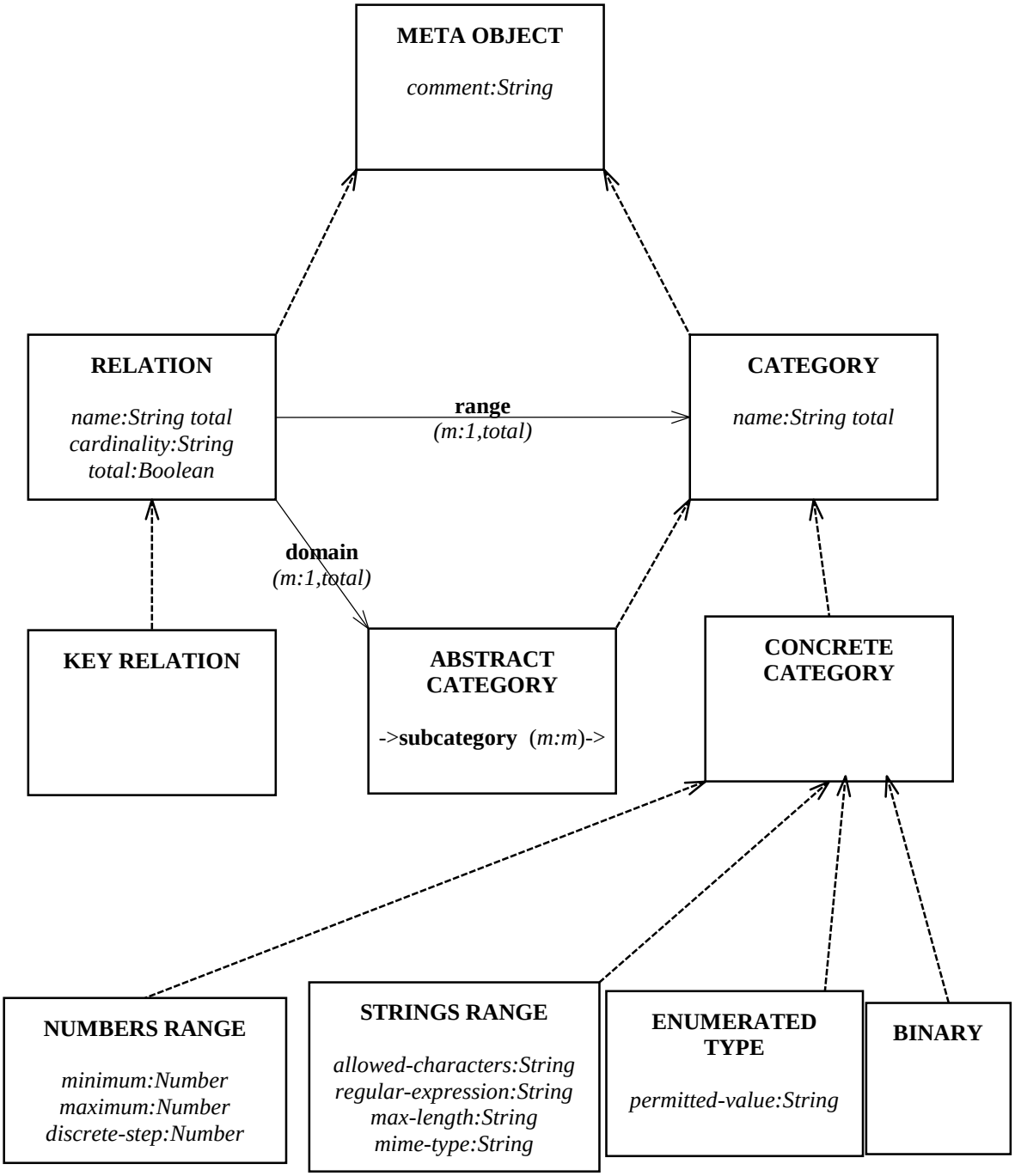
[125] Yu C.T., Chang C.C., Templeton M., Brill D. and Lund E., "Query Processing in a Fragmented Relational Distributed System: Mermaid". In *IEEE Transactions on Software Engineering* (*TSE*), Vol.11, No. 8, 1985, pp.795-810.

[126] Yu C.T., Zhang Y., Meng W., Kim W., Wang G., Pham T. and Dao S., "Translation of Object-Oriented Queries to Relational Queries". In *Proceedings of the IEEE International Conference on Data Engineering* (*ICDE '95*), 1995, pp.90-97.

[127] Zhu Q. and Larson P.A., "A Query Sampling Method for Estimating Local Cost Parameters in a Multidatabase System". In *Proceedings of the IEEE International Conference on Data Engineering* (*ICDE '94*), 1994, pp.144-153.

[128] Zhu Q. and Larson P.-A., “Solving Local Cost Estimation Problem for Global Query Optimization in Multidatabase Systems”. In *Distributed and Parallel Databases*, Vol. 6, No. 4, 1998, pp.373-420.

APPENDIX 1



Subschema representing Sem-ODM Meta-Schema

META-OBJECT — category (A catalog of meta objects)

comment — attribute of *META-OBJECT*, range: *String* (*m:1*) (Comment about meta object)

RELATION — subcategory of *META-OBJECT* (A catalog of relations)

name — attribute of *RELATION*, range: *String* (*m:1*,*total*) (Name of relation)

cardinality — attribute of *RELATION*, range: *String* (*m:1*) (Cardinality of relation)

total — attribute of *RELATION*, range: *Boolean* (*m:1*) (Totality of relation)

CATEGORY — subcategory of *META-OBJECT* (A catalog of categories)

name — attribute of *CATEGORY*, range: *String* (*m:1*,*total*) (Name of category)

ABSTRACT-CATEGORY — subcategory of *CATEGORY*

KEY-RELATION — subcategory of *RELATION* (A catalog of key relations)

CONCRETE-CATEGORY — subcategory of *CATEGORY* (A catalog of concrete categories)

STRINGS-RANGE — subcategory of *CONCRETE-CATEGORY* (A catalog of strings)

allowed-characters — attribute of *STRINGS-RANGE*, range: *String* (*m:1*) (Allowed characters of the strings range)

regular-expression — attribute of *STRINGS-RANGE*, range: *String* (*m:1*) (Regular expression)

max-length — attribute of *STRINGS-RANGE*, range: *String* (*m:1*) (Maximum length of strings range)

mime-type — attribute of *STRINGS-RANGE*, range: *String* (*m:1*) (Mime type)

NUMBERS-RANGE — subcategory of *CONCRETE-CATEGORY* (A catalog of number ranges)

minimum — attribute of *NUMBERS-RANGE*, range: *Number* (*m:1*) (Minimum of

numbers range)

maximum — attribute of *NUMBERS-RANGE*, range: *Number* (*m:1*) (Maximum of numbers range)

discrete-step — attribute of *NUMBERS-RANGE*, range: *Number* (*m:1*) (Discrete step in numbers range)

ENUMERATED-TYPE — subcategory of *CONCRETE-CATEGORY* (A catalog of enumerated types)

permitted-value — attribute of *ENUMERATED-TYPE*, range: *String* (*m:1*) (Permitted value)

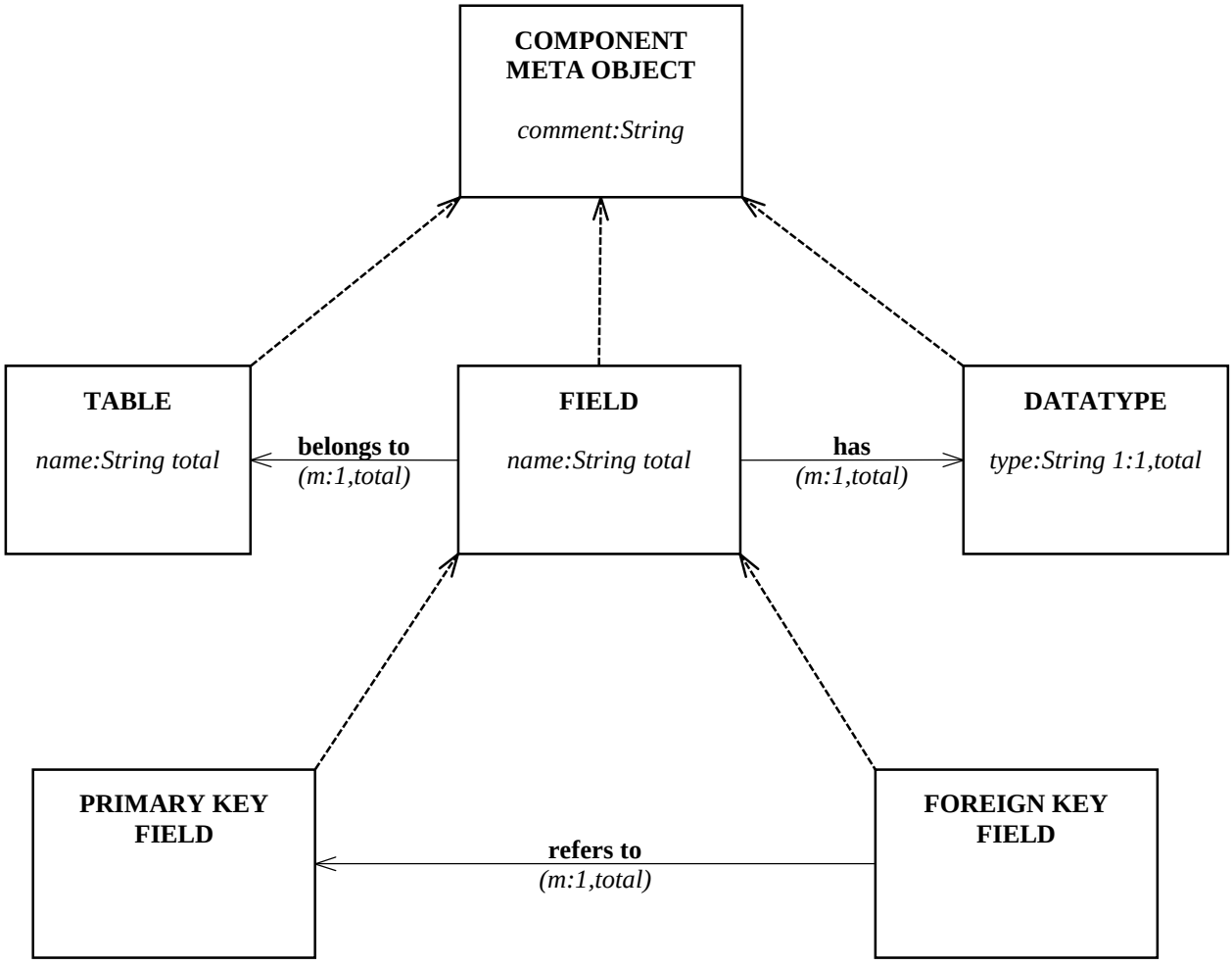
BINARY— subcategory of *CONCRETE-CATEGORY* (A catalog of binary types)

subcategory — relation from *ABSTRACT-CATEGORY* to *ABSTRACT-CATEGORY* (*m:m*) (Subcategory relation)

domain — relation from *RELATION* to *ABSTRACT-CATEGORY* (*m:1,total*) (Domain of relation)

range — relation from *RELATION* to *CATEGORY* (*m:1,total*) (Range of relation)

APPENDIX 2



Subschema representing relational meta-schema

COMPONENT-META-OBJECT — category (A catalog on component meta objects)

comment — attribute of *COMPONENT-META-OBJECT*, range: *String* (*m:1*) (Comment on component meta object)

TABLE — subcategory of *COMPONENT-META-OBJECT* (A catalog of tables)

name — attribute of *TABLE*, range: *String* (*m:1,total*) (Name of table)

FIELD — subcategory of *COMPONENT-META-OBJECT* (A catalog of fields)

name — attribute of *FIELD*, range: *String* (*m:1,total*) (Name of field)

DATATYPE — subcategory of *COMPONENT-META-OBJECT* (A catalog of datatypes)

type — attribute of *DATATYPE*, range: *String* (*1:1*,*total*) (Type of datatype)

PRIMARY-KEY-FIELD — subcategory of *FIELD* (A catalog of primary key fields)

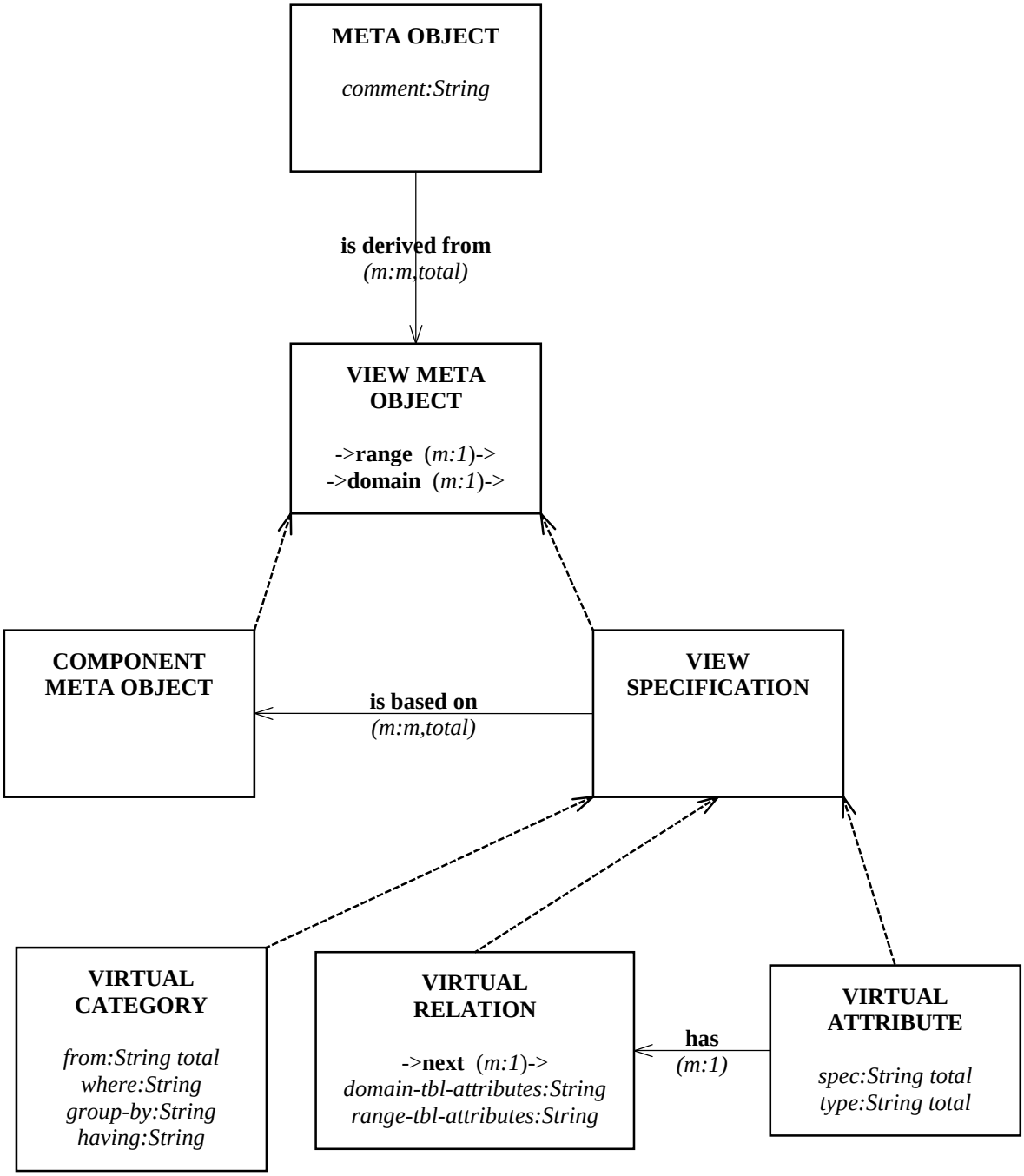
FOREIGN-KEY-FIELD — subcategory of *FIELD* (A catalog of foreign key fields)

belongs-to — relation from *FIELD* to *TABLE* (*m:1*,*total*) (Field belongs to a table)

has — relation from *FIELD* to *DATATYPE* (*m:1*,*total*) (Field has a datatype)

refers-to — relation from *FOREIGN-KEY-FIELD* to *PRIMARY-KEY-FIELD* (*m:1*,*total*) (Foreign key field refers to primary key field)

APPENDIX 3



Subschema representing mapping between Sem-ODM and relational schemas

META-OBJECT — category (A catalog of meta objects)

comment — attribute of *META-OBJECT*, range: *String* (*m:1*) (Comment about meta

objects)

VIEW-META-OBJECT — category (A catalog of view-meta-objects)

COMPONENT-META-OBJECT — subcategory of *VIEW-META-OBJECT* (A catalog of component meta objects)

VIEW-SPECIFICATION — subcategory of *VIEW-META-OBJECT* (A catalog of view specifications)

VIRTUAL-CATEGORY — subcategory of *VIEW-SPECIFICATION* (A catalog of virtual categories)

from — attribute of *VIRTUAL-CATEGORY*, range: *String* (*m:1*,*total*) (A catalog of from clauses)

where — attribute of *VIRTUAL-CATEGORY*, range: *String* (*m:1*) (A catalog of where clauses)

group-by — attribute of *VIRTUAL-CATEGORY*, range: *String* (*m:1*) (A catalog of group by clauses)

having — attribute of *VIRTUAL-CATEGORY*, range: *String* (*m:1*) (A catalog of having clauses)

VIRTUAL-RELATION — subcategory of *VIEW-SPECIFICATION* (A catalog of virtual relations)

domain-tbl-attributes — attribute of *VIRTUAL-RELATION*, range: *String* (*m:1*) (A catalog of domain attributes (of join condition))

range-tbl-attributes — attribute of *VIRTUAL-RELATION*, range: *String* (*m:1*) (A catalog of range attributes (of join condition))

VIRTUAL-ATTRIBUTE — subcategory of *VIEW-SPECIFICATION* (A catalog of

virtual attributes)

spec — attribute of *VIRTUAL-ATTRIBUTE*, range: *String* (*m:1,total*) (A catalog of specifications for attributes)

type — attribute of *VIRTUAL-ATTRIBUTE*, range: *String* (*m:1,total*) (A catalog of types of specifications)

range — relation from *VIEW-META-OBJECT* to *VIEW-META-OBJECT* (*m:1*) (Range of view meta object)

domain — relation from *VIEW-META-OBJECT* to *VIEW-META-OBJECT* (*m:1*) (Domain of view meta object)

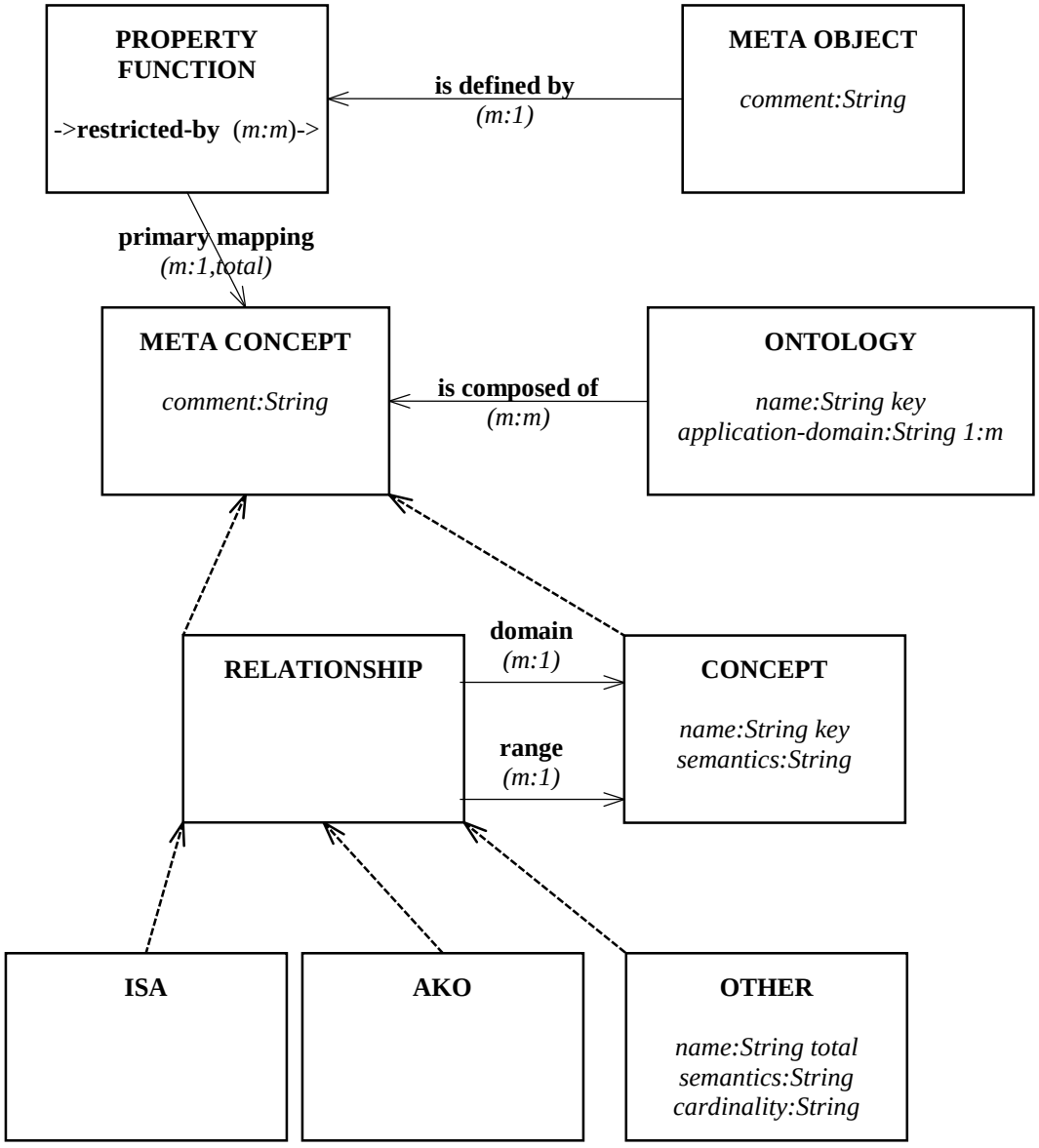
next — relation from *VIRTUAL-RELATION* to *VIRTUAL-RELATION* (*m:1*)

is-derived-from — relation from *META-OBJECT* to *VIEW-META-OBJECT* (*m:m,total*) (Meta object is derived from view meta object)

is-based-on — relation from *VIEW-SPECIFICATION* to *COMPONENT-META-OBJECT* (*m:m,total*) (View specification is based on component meta objects)

has — relation from *VIRTUAL-ATTRIBUTE* to *VIRTUAL-RELATION* (*m:1*) (Virtual attribute may have a join condition (e.g. specifying multi-valued attributes))

APPENDIX 4



Sub-schema representing ontologies and mapping to Sem-ODM schema

META-OBJECT — category (A catalog of meta objects)

comment — attribute of *META-OBJECT*, range: *String* (*m:1*) (Comment about meta object)

PROPERTY-FUNCTION — category

META-CONCEPT — category (A catalog of meta concepts)

comment — attribute of *META-CONCEPT*, range: *String* (*m:1*) (Comment of meta concept)

CONCEPT — subcategory of *META-CONCEPT*

name — attribute of *CONCEPT*, range: *String* (*key*) (Name of concept)

semantics — attribute of *CONCEPT*, range: *String* (*m:1*) (Meaning of concept in English)

RELATIONSHIP — subcategory of *META-CONCEPT* (A catalog of relationships)

domain — relation from *RELATIONSHIP* to *CONCEPT* (*m:1, total*) (A relationship has a domain concept)

range — relation from *RELATIONSHIP* to *CONCEPT* (*m:1, total*) (A relationship has a range concept)

ISA — subcategory of *RELATIONSHIP* (A catalog of ISA relationships)

AKO — subcategory of *RELATIONSHIP* (A catalog of a-kind-of relationships)

OTHER — subcategory of *RELATIONSHIP* (A catalog of other (not ISA or AKO) relationships)

name — attribute of *OTHER*, range: *String* (*m:1,total*) (Name of relationship)

semantics — attribute of *OTHER*, range: *String* (*m:1*) (Meaning of relationship)

cardinality — attribute of *OTHER*, range: *String* (*m:1*) (Cardinality of relationship)

ONTOLOGY — category (A catalog of ontologes)

name — attribute of *ONTOLOGY*, range: *String* (*key*) (Name of ontology)

application-domain — attribute of *ONTOLOGY*, range: *String* (*1:m*) (Application domains of ontology)

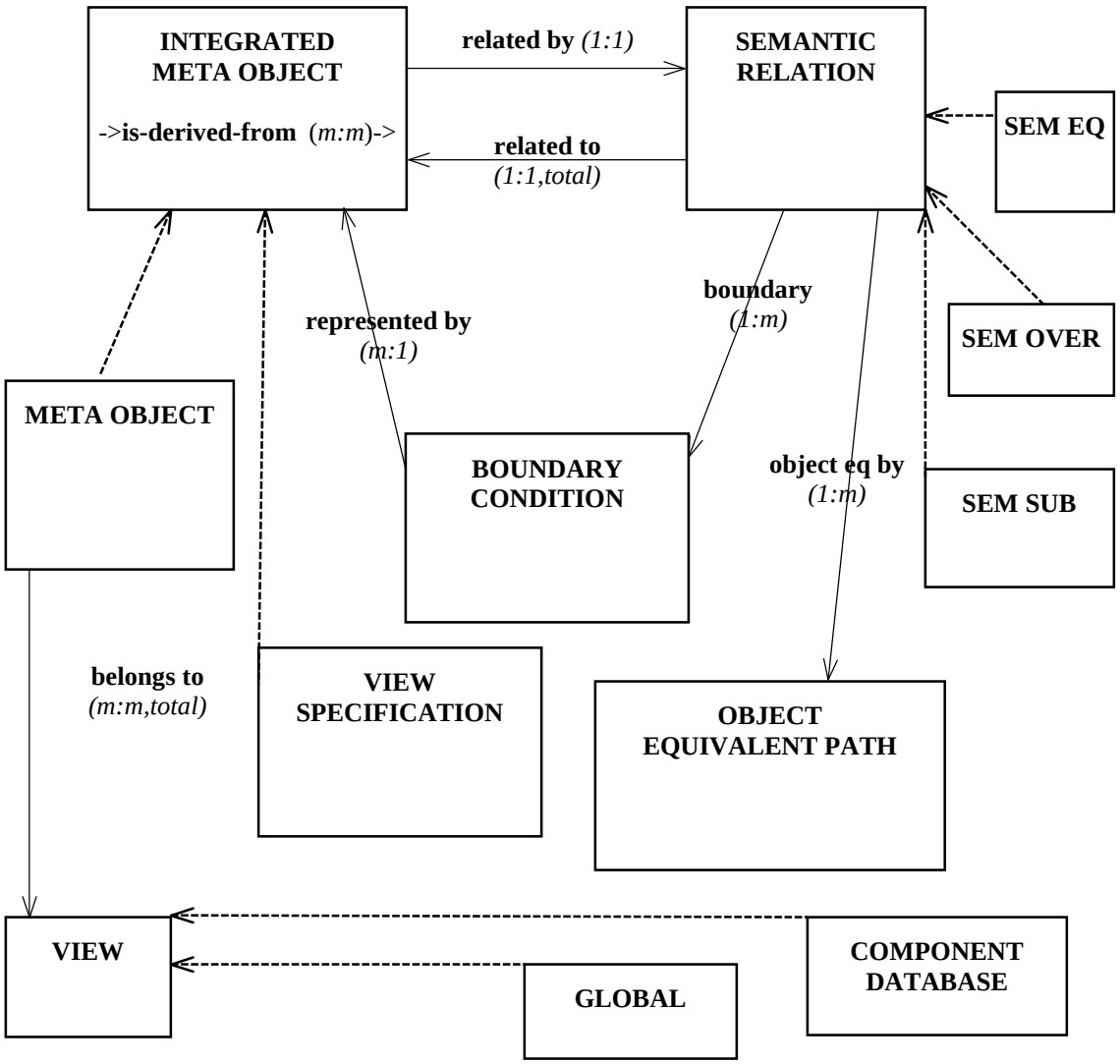
restricted-by — relation from *PROPERTY-FUNCTION* to *PROPERTY-FUNCTION* (*m:m*) (Property function may have restrictions specified by other property functions)

is-defined-by — relation from *META-OBJECT* to *PROPERTY-FUNCTION* (*m:1*) (Meta object is defined by property functions)

primary-mapping — relation from *PROPERTY-FUNCTION* to *META-CONCEPT* (*m:1,total*) (Property function has a primary mapping to a meta concept)

is-composed-of — relation from *ONTOLOGY* to *META-CONCEPT* (*m:m*) (Ontology is composed of a set of meta-concepts)

APPENDIX 5



Subschema of knowledge base at global site

INTEGRATED-META-OBJECT — category (A catalog of integrated meta objects)

META-OBJECT — subcategory of *INTEGRATED-META-OBJECT* (A catalog of meta objects (global view or component database))

VIEW-SPECIFICATION — subcategory of *INTEGRATED-META-OBJECT* (A catalog of view specifications)

SEMANTIC-RELATION — category (A catalog of semantic relations)

COMPONENT-DATABASE — subcategory of *VIEW* (A catalog of compoent databases)

SEM-EQ — subcategory of *SEMANTIC-RELATION* (A catalog of semantically equivalent relations)

SEM-OVER — subcategory of *SEMANTIC-RELATION* (A catalog of semantically overlapping relations)

SEM-SUB — subcategory of *SEMANTIC-RELATION* (A catalog of semantically subset relations)

BOUNDARY-CONDITION — category (A catalog of boundary conditions)

OBJECT-EQUIVALENT-PATH — category (A catalog of object equivalent paths)

VIEW — category (A catalog of views)

GLOBAL — subcategory of *VIEW* (A catalog of global views)

related-by — relation from *INTEGRATED-META-OBJECT* to *SEMANTIC-RELATION* (*1:1*) (Integrated meta object is related by semantic relation)

related-to — relation from *SEMANTIC-RELATION* to *INTEGRATED-META-OBJECT* (*1:1,total*) (Semantic relation is related to an integrated meta object)

boundary — relation from *SEMANTIC-RELATION* to *BOUNDARY-CONDITION* (*1:m*) (Semantic relation contains boundary conditions)

represented-by — relation from *BOUNDARY-CONDITION* to *INTEGRATED-META-OBJECT* (*m:1*) (Boundary condition is represented by an integrated meta object)

object-eq-by — relation from *SEMANTIC-RELATION* to *OBJECT-EQUIVALENT-PATH* (*1:m*) (Semantic relation contains object equivalent paths)

is-derived-from — relation from *INTEGRATED-META-OBJECT* to *INTEGRATED-META-OBJECT* (*m:m*) (Integrated meta object is derived from another integrated meta object)

belongs-to — relation from *META-OBJECT* to *VIEW* (*m:m,total*) (Meta object belongs to a global or component view)

VITA

RUKSHAN INDIKA ATHAUDA

April 24, 1978	Born, Colombo, Sri Lanka
1994	Diploma in Computer Science Institute of Technological Studies Colombo, Sri Lanka
1996	B.S., Computer Science Florida International University Miami, Florida
1998	M.S., Computer Science Florida International University Miami, Florida
2000	Doctorate in Computer Science Florida International University Miami, Florida
1996-2000	Undergraduate Research Assistant Department of Physics Florida International University Miami, Florida
1996-1999	Lab Instructor School of Computer Science Florida International University Miami, Florida
1996-2000	Graduate Research Assistant High-performance Database Research Center School of Computer Science Florida International University Miami, Florida
2000	Graduate Teaching Certificate The Academy for the Art of Teaching Florida International University Miami, Florida

CONFERENCE PUBLICATIONS

Rishe N., Athauda R., Yuan J. and Chen S.C., “Knowledge Management for Database Interoperability”. Submitted to the *International Conference on Information Reuse and Integration*, Honolulu, Hawaii, November 1 - 3, 2000.

Rishe N., Athauda R.I., Yuan J. and Chen S.C., “Semantic relations: The key to integrating and query processing in heterogeneous databases”. To appear in *The 4th World Multiconference on Systemics, Cybernetics and Informatics*, Orlando, Florida, July 23 – 26, 2000.

Rishe N., Yuan J., Athauda R., Lu X., Ma X., Vaschillo A., Shaposhnikov A., Vasilevsky D. and Chen S.C., “SemanticAccess: Semantic Interface for Querying Databases”. To appear in *The International Conference on Very Large Data Bases (VLDB 2000)*, September 10-14, 2000.

Rishe N., Yuan J., Athauda R., Lu X. and Ma X., “SemWrap: A semantic wrapper over relational databases, with substantial size reduction of user's SQL queries”. In the *Proceedings of the 7th Extending Database Technology (EDBT 2000) - Software Demonstrations Track*, March 27-31, 2000.

Athauda R., “Heterogeneity Resolution in MSemODB”, Technical Report 2000-02, School of Computer Science, Florida International University, 2000.

Rishe N., Barton B., Prabakaran N., Gutierrez M., Martinez M., Athauda R., Gonzalez A. and Graham S., "Landsat Viewer: A Tool to create Color Composite Images of Landsat Thematic Mapper Data". In *Proceedings of the International Conference on Geospatial Information in Agriculture and Forestry*, June 1-3, 1998.

Rishe N., Barton D., Prabakaran N., Gutierrez M., Alvarez E., Athauda R., Rodriguez J., Gonzalez A., "Landsat Data Visualizing via the Internet". In *Proceedings of the International Symposium on Spectral Sensing Research*, December 13-19, 1997.

Prabakaran N., Rishe N. and Athauda R., "Tracking Hurricane Paths". In *Proceedings of Image Registration Workshop*, NASA GFSC, November 20-12, 1997.

JOURNAL PUBLICATIONS

Branly R.M., Athauda R.I., Fillingim M.O. and Van Hamme W.,"Light Curve Solutions for Eclipsing Binaries in NGC 188". In *Astrophysics and Space Science* 235 (1): 149-160, January 1996.