Semantic Integration of Heterogeneous Databases in Multidatabase System

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Computer Science & Engineering

By:
Sugandha Lohar
(800832024)

Under the supervision of:
Ms. Shalini Batra
Assistant Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

June 2010

Certificate

I hereby certify that the work which is being presented in the thesis entitled, **"Semantic Integration of Heterogeneous Databases in Multidatabase System"**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in Computer Science and Engineering submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Ms. Shalini Batra* and refers other researcher's works which are duly listed in the reference section.

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

Sugandha L.

(Sugandha Lohar)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Ms. Shalini Batra)
Assistant Professor
Computer Science and Engineering Department
Thapar University
Patiala

Countersigned by

02/02/10

(Dr. Rajesh Bhatia)
Head
Computer Science & Engineering Department
Thapar University
Patiala

2/2/10

(Dr. R.K.Sharma)
Dean (Academic Affairs)
Thapar University
Patiala

Acknowledgement

I would like to start by praising GOD ALMIGHTY for providing me with faith, self confidence, ability and strength to complete this thesis.

I would like to place on record my deepest gratitude to Ms. Shalini Batra, Assistant Professor, Thapar University, Patiala for her valuable advice and guidance in carrying out the thesis. I appreciate her unlimited help and patience with her students. I will always remain indebted to her for the moral support, encouragement and the infectious zeal and enthusiasm for work that she imbibed upon me.

With great pleasure and acknowledgement, I extend my profound thanks to Dr. Rajesh Bhatia, Assistant Professor and Head (CSED), Thapar University, Patiala who has been a constant source of inspiration for me throughout this work.

I am also thankful to all the staff members of the Department for their full cooperation and help.

Thanks are also due to my friends Shirin, Rashmi, Sushila, Tikka, Yogesh and Prashant for boosting me with their constant encouragement and confidence and bearing me, day and night throughout the thesis work.

Lastly, I am thankful to my parents for their blessing and moral support without which this thesis would not have been possible.

Sugandha Lohar

Abstract

Establishing semantic interoperability among heterogeneous information sources is becoming a critical issue in the database community with the increase in both the amount of data and the number of data producers. One of the challenging problems in the multidatabase systems is to find the most viable solution to the problem of interoperability of distributed heterogeneous autonomous local component databases. This has resulted in the creation of a global schema over set of these local component database schemas to provide a uniform representation of local schemas. Despite the critical importance, current approaches to semantic interoperability of heterogeneous databases have not been sufficiently effective.

Interoperability not only has to resolve the differences in data structures, it also has to deal with semantic heterogeneity. Taking semantically heterogeneous databases as the prototypical situation, the presented work describes how ontology (in the traditional metaphysical sense) can contribute to delivering a more efficient and effective process of matching by providing a framework for the analysis, and so the basis for a methodology.

An algorithm is proposed which detects the semantically related classes from heterogeneous database and a set of integrating operators are defined to integrate local schemas based on the semantic relevance of their classes. It provides a model independent representation of virtual classes of the global schema.

Table of Contents

List of Figures

List of Tables

CHAPTER 1
INTRODUCTION

Due to the explosion of the internet and intranets and the ongoing advances in the World Wide Web, network and database technologies have changed from centralized to distributed information systems. Most common problem faced by many organizations these days is the uniform and scalable access to multiple disparate information sources and repositories, including databases, knowledge bases, file systems, digital libraries and information retrieval systems. As these information sources get updated and change constantly, it becomes difficult for users to navigate, collect, evaluate and process data in this open and dynamic universe [1]. Decision makers often need information from multiple information sources but are unable to get and fuse such information in a timely fashion. This is due to the unpredictable state of networks and connection at information sources and due to the heterogeneous and evolving nature of information sources.

In recent years, there has been considerable interest in Multidatabase Systems (MDS) which attempt to logically integrate a number of independent distributed DBMSs while allowing the local DBMSs to maintain complete control of their operations. Thus a multidatabase system is a distributed DBMS in which each site maintains complete autonomy

Semantic heterogeneity or semantic conflict is the main source of problems in multidatabase design. Semantic heterogeneity is often present in multidatabase systems because of the lack of global schema definition. For last few years a significant quantity of multidatabase research has focused on resolving the problem of semantic heterogeneity or semantic conflicts.

Ontologies have been suggested as a way to solve the problem of information heterogeneity by providing formal, explicit definitions of data and reasoning ability over related concepts. Ontology mapping basically focuses on finding semantic correspondences between similar elements of different ontologies, thus allowing applications to agree on the terms that they use when communicating. It facilitates

communication by providing precise notions that can be used to compose messages (queries, statements) about the domain [2].

An approach using ontology mapping for detecting and integrating semantically related classes in an object-oriented multidatabase system is proposed in the thesis work done. This process is a vital one in the realization of multidatabase systems, as the semantic contents of local databases must be exploited before the integration process takes place.

To create multidatabase without semantic conflicts, a significant amount of research has focused on schema integration both at logical and the conceptual level However, in practice, semantic conflicts exist not only at the logical or conceptual level, but also at the instance or run-time level.

1.1 Need for Data Processing

One of the reasons for the original development of database systems was to overcome the difficulties of supporting shared access to several files created by different application programs. File based approach has many difficulties like:

- Separated and Isolated Data,
- Duplication of data
- Data Dependence
- Difficulty in representing data from the user's view
- Data Inflexibility
- Incompatible file formats

To overcome these difficulties database management system (DBMS) were created to manage these autonomous files as a single centralized collection of data. A DBMS is a computerized record keeping system whose overall purpose is to maintain data and make it available on demand [1]. This had several advantages over an autonomous files approach, namely it could be used to reduce data duplication, avoid data inconsistency, allow sharing of data, increase security and maintain integrity of the data. It was successful and sufficient to meet user requirements for several years. However, today's user data processing requirements and capabilities have changed and new applications often involve accessing and maintaining data from several pre-existing databases and software packages, which are typically located on autonomous software and hardware

platforms distributed over the many sites of a large computer network which leads to heterogeneity and legacy problems, initiating a need for timely and efficient solution.
One solution to above problem is to physically integrate all databases into one to provide location, replication and heterogeneity transparency but this approach will not be feasible for large databases. Another approach could be logical integration of all data needed by an application thus giving users an illusion of a globally integrated database, shielded from most types of heterogeneity without the data migrating to a new database and without requiring the users to know either the location or the characteristics of the different local databases and their corresponding DBMSs. The latter approach thus forms Multidatabase System.

1.2 Introduction to Multidatabase System

A multidatabase system provides integrated access to heterogeneous, autonomous local databases. It resides unobtrusively on top of existing database systems and presents the illusion of a single database to its users. Unlike the term distributed-database, a multidatabase system always implies the integration of heterogeneous database systems.
The basic requirements assumptions for a multi database system include the following

- The local DBMSs have independent meta-data and exist before joining the multi database system.
- The local DBMS should participate in the multi database system with little or no modification.
- The local DBMS retains autonomy with full control over local data and processes.

As shown in figure 1.1 there are federated and unfederated (where there are no local users) MDBS. A federated system is a cross between a distributed and a centralized DBMS, it is distributed system for global users and a centralized system for local users. Federated systems differ from distributed database management systems in the level of local autonomy provided. It could further be classified as loosely coupled and tightly coupled Federated MDBS where loosely coupled MDBS does not have global conceptual schema.

To describe links between global and local data descriptions and who is responsible for creating and maintaining these links, a tightly coupled MDBS maintains global schemas and these schemas are usually created and maintained by the system DBA.

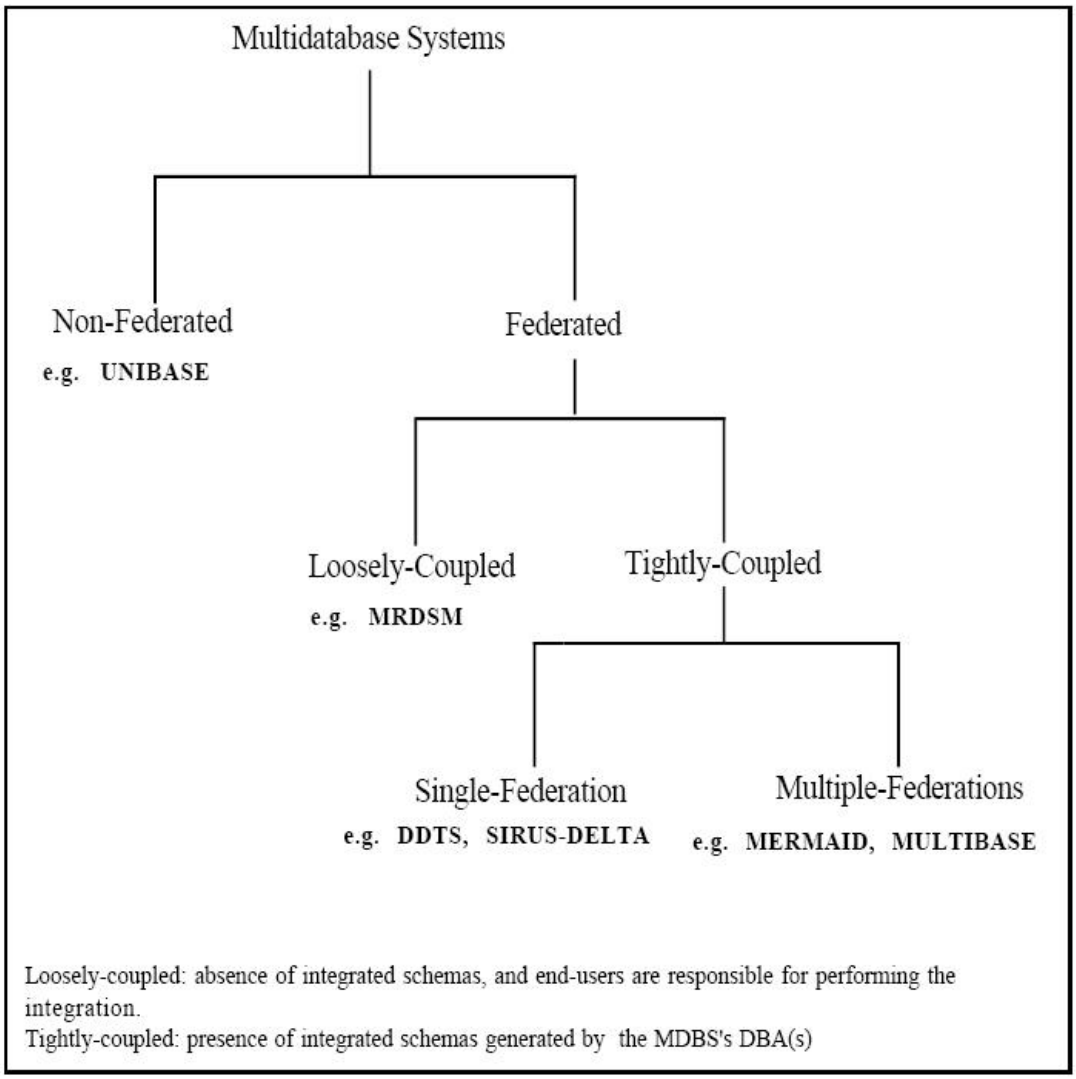

Figure 1.1: Taxonomy of Multidatabase System Architecture [3]

A Loosely coupled MDB system on the other hand, provides either importation or negotiation technique or a powerful query language. It is a site's DBA or end user's responsibility to perform the importation or negotiation facility while in latter case it is end user's responsibility to create a valid MDB queries. Figure 1.2 shows the reference

architecture for a tightly coupled federated multidatabase system. In the literature, multidatabase systems are also called federated database systems, multidatabase language systems and interoperable systems.

Two important dimensions of database integration that give the concept of coupling the local DBMSs to build a multidatabase are schema versus instance and physical versus virtual integration. Schema integration refers to the integration of meta-data at the time the multidatabase is designed, while instance integration refers to the process of merging query results (from several local databases) at run time.

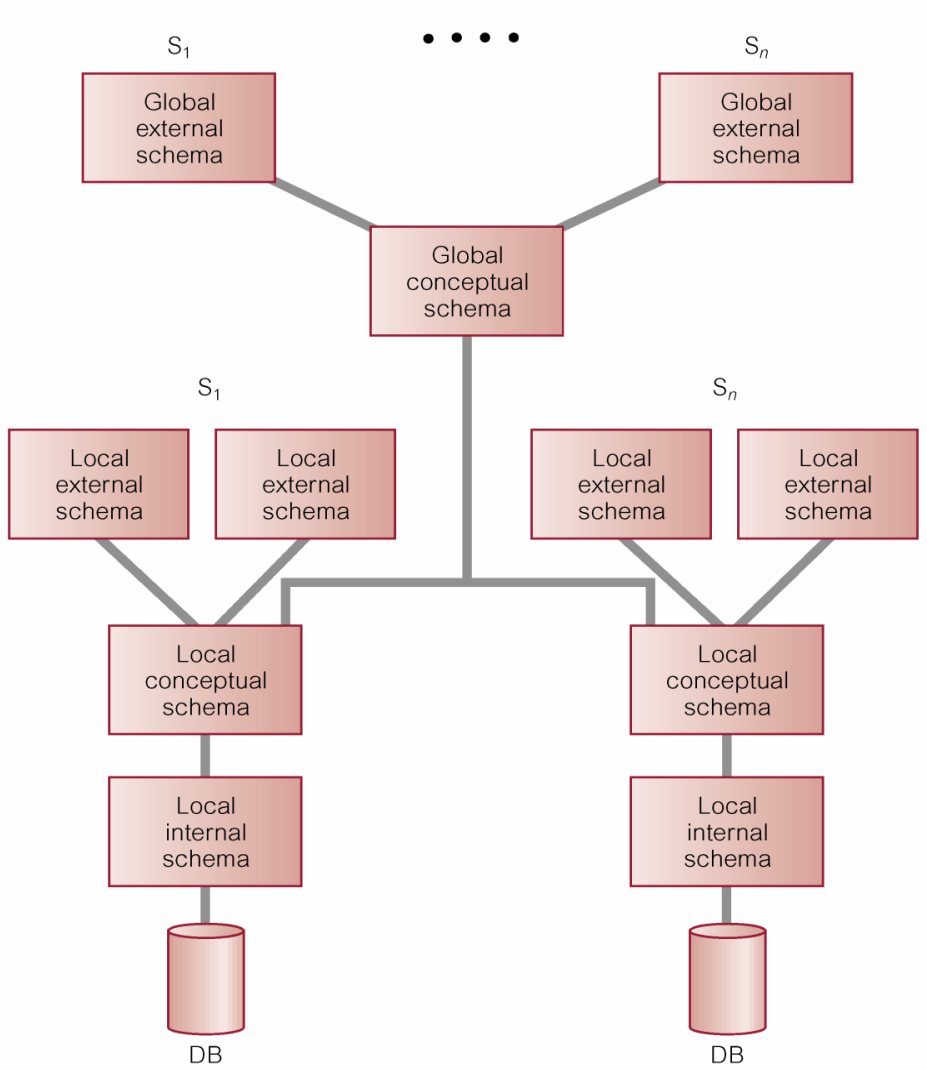

Figure 1.2: Reference architecture for a Tightly Coupled FMDBS [4]

In addition, a database can be integrated physically or virtually. Physical integration leaves no atomicity to local systems while virtual integration gives local system full control of their own operations.

The distinction between different types of multidatabase systems can be seen clearly in the taxonomy [2]. Bright's classification scheme is shown as a continuum in Figure 1.3.

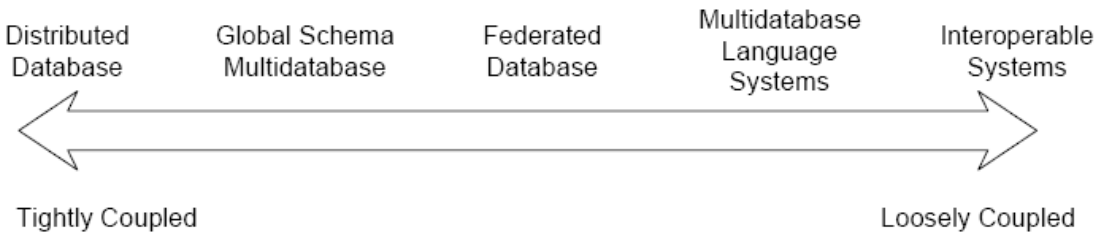

Figure 1.3: Different types of multidatabase systems [1].

At one extreme is the distributed database system, which is the most tightly integrated, while the interoperable system is on the other extreme. A distributed database is a homogeneous database system that is physically integrated at the schema level. In this case the problem of semantic conflicts has been solved at the stage of global schema construction. Any new node added to the distributed system must follow the predefined terms in the global schemata. Unlike distributed database, the semantic heterogeneity present in existing local database systems causes more problems in designing all other systems illustrated in Figure 1.3.

Henceforth, the multidatabase system referred here excludes the distributed database but includes global schema multidatabase, federated database multidatabase language systems, and interoperable systems. Therefore a working definition of multidatabase system is one in which there are local autonomous databases which require an integrated view for the display of query results submitted by end users.

1.2.1 Schema Integration

In MDBS where a global schema or several global schemas are maintained, schema integration has to be performed. Schema integration is the process where information from several local schemas is merged into a single global schema. It provides global users with distribution, local autonomy and heterogeneity transparency in their

interoperable access to the component databases. During schema integration it is necessary to identify local objects that are related to each other in some respect and define their corresponding global objects. There are several conflicts or heterogeneity which makes schema integration a difficult task.

1.2.2 Semantic Heterogeneity

Even though the concept of multidatabase systems has been around for several decades, the development of multidatabase systems still faces many obstacles. One obstacle is the semantic heterogeneity or semantic conflict between different database systems.

Semantic heterogeneity refers to the fact that data present in different systems may be subjected to different interpretations even when the corresponding database schemas are identical [5].

The main source of these conflicts is the different data abstraction/representation mechanism used by different designers. Even worse is the presence of the many data modeling methodologies that exist in today's computing environment. Database design methodologies have evolved from those based on hierarchical models to the currently dominant object oriented database models. However, not every organization changes its database system whenever a "better" system is available in fact many large organizations are still using database systems that are more than 25 years old [6]. Database systems designed of different database models, different query languages, different DBMSs, different version of same DBMS, or different operating systems and network protocols increases the complexity of the schema integration problem.

In the schema integration, semantic relevance can be detected by classifying these different concepts for example, when the term salary is used in a local database, a multidatabase designer would have several questions like the term may represent monthly or annual salary or the unit of measurement could be either U.S. dollar or Indian Rupees or this term may represent an entity or may be an attribute. Without precisely knowing the relevant context information, the multidatabase system cannot be properly designed. Thus the dimensions should be identified for the component schema by categorizing or classifying all possible sources of semantic conflicts. Some commonly identified sources from the literature are discussed below.

1.2.2.1 Naming

Naming refers to the semantic relationship between object, attribute, or instance name. These relationships include synonyms, homonyms and unrelated term.

- **Synonyms:** Synonyms are terms that represent similar concepts. For example, the name of a company may be reported differently in different systems like “IBM” versus “International Business Machine” versus “I.B.M”.
- **Homonyms:** When the same name is used for two different concepts. For example, “school” may be used to represent a university in one database while it may represent a play school in another database.
- **Unrelated terms:** Unrelated terms are those that are not related to one another directly, that is, they are neither synonyms nor homonyms. For example, the two terms “school” and “ID” are not directly related. These are different concepts and can be related to each other only through externally defined business rules.

1.2.2.2 Structure

Structural conflicts arise as a result of a different choice of modeling constructs for integrity constraints [7]. These conflicts include: type differences, abstraction differences, dependency conflicts, modeling conflicts, and cardinality conflicts, among others [1].

- **Type differences:** Type differences arise when the same concept is represented by different modeling constructs in different schemas. For example, the concept “skill” may be represented as an entity in one database while being represented as an attribute in another database.
- **Abstraction differences:** Abstraction is a process of simplifying complex things [7]. Consider a relational database table as an example; the selection of attributes would be an abstraction process. In this case, abstraction differences refer to different sets of attributes for the same concept selected by different database designers. For instance, in one database, a “student” table may have three attributes. While in another database, a table may contain more than three attributes but represent exactly the same meaning.

- **Cardinality conflicts:** When a group of concepts are related among themselves through different dependencies in different local schemas. For example, the one-to-many relationship between “managers” and “employees” may apply in one organization whereas the relationship may be one-to- one in another organization.
- **Key conflicts:** Different keys are assigned to the same concept in different schemas. For example, “Reg_ID” and “SSN” could both be utilized as the primary key for the entity “employee”.
- **Integrity conflicts:** When different updating rules or computation functions are placed on the same concept, behavioral conflicts arise. For example, in one database, a client record cannot exist without the assignment of an employee in one database, while in a second database this referential integrity constraint need not be enforced.

1.2.2.3 Scaling and Type Conflicts

The concept of instance and Scaling and type conflicts overlap but are not encompassed by context differences [9].

- **Data type differences:** Data type differences arise when different data types are used for the same concept. For example, the data type for “Regn_No.” could be an integer in one database while the same attribute could be defined as a string in another database.
- **Scale differences:** Even when the same data type is used in different databases, the same value could still mean the same thing. The commonly seen attribute price, for instance, has a numerical value, which could be Pounds in one database but may represent Rupees in another.

1.2.2.4 Instance Conflicts

- **Instance identification conflicts:** The problem of instance identification conflicts is very similar to the naming conflict. Naming conflicts exist in the level of schema integration. The instance identification conflicts, as the name suggested, exist in the instance level. Suppose that there is an object (instance name) “US” in one database that represents the United States. In

a second database, an object name “USA” could represent the same concept.

- **Data value conflicts:** Data value conflicts arise when the same object instances have different values. Assuming that there are two object instances both called “Phone_No” in two different databases and that these objects both represent the same person’s phone number. In this case the domain, naming and structure of these two objects are identical. However, the value of the phone numbers stored in these two object instances could be different. This difference at the instance level is known as a data value conflict.

All the identified semantic conflicts from above are summarized in Table 1.1. The naming and structural conflicts focus on the method of schema integration. The third and fourth dimensions (domain and identification) indicate that the building of a multidatabase system requires more than just schema integration when considering the challenge of integration at the instance level [10]. Conflicts at the instance level often require convergence rules or assertions to assist in the process of resolution.

Semantic Conflicts		Identifiable Components
Schema Conflicts	Naming	Synonyms, Homonyms
	Structural	Type, Cardinality, Key, Integrity,
Scaling and Type Conflicts		Data Type, Data Scale
Instance Conflicts		Instance Identification, Data values

Table 1.1: Categorization of semantic conflicts in databases [1].

1.3 Introduction to Ontology

The term "ontology" comes from the field of philosophy that is concerned with the study of being or existence. In philosophy, one can talk about ontology as a theory of the nature of existence. In computer and information science, ontology is a technical term denoting an artifact that is designed for a purpose, which is to enable the modeling of knowledge about some domain, real or imagined. Ontology defines a set of representational primitives with which to model a domain of knowledge or discourse. The representational primitives are typically classes (or sets), attributes (or properties), and relationships (or relations among class members). In the context of database systems, ontology can be viewed as a level of abstraction of data models, analogous to hierarchical and relational models, but intended for modeling knowledge about individuals, their attributes, and their relationships to other individuals. Ontologies are typically specified in languages that allow abstraction away from data structures and implementation strategies; in practice, the languages of ontologies are closer in expressive power to first-order logic than languages used to model databases. For this reason, ontologies are said to be at the "semantic" level, whereas database schema are models of data at the "logical" or "physical" level. Due to their independence from lower level data models, ontologies are used for integrating heterogeneous databases, enabling interoperability among disparate systems, and specifying interfaces to independent, knowledge-based services.

1.3.1 Applications of ontology

The concept of ontology has been widely employed by several research communities. In the artificial intelligence (AI) community, ontologies have been used to capture domain knowledge for knowledge-based systems. The knowledge is typically represented in the knowledge-based system’s representation language using the vocabulary provided by ontology. In the distributed artificial intelligence (DAI) community, which includes research on distributed problem solving (DPS) and multiagent systems (MAS), ontologies have been accepted as an effective means to facilitate collaboration and communication among agents [11]. The need for ontologies has also been addressed in the information retrieval area to facilitate semantic information searching. Other areas,

such as natural language processing (NLP), utilize ontologies to facilitate natural language generation and interpretation [10]. The database community is not an exception. In particular, research on distributed, heterogeneous databases has begun to exploit ontologies in order to support semantic interoperability, cross database search etc.

1.3.2 Ontology Mapping for Databases

Ontology mapping is the problem of finding semantic correspondences between similar elements of different ontologies [11]. As discussed above semantic heterogeneity occurs whenever two contexts do not share the same interpretation of information For example, as shown in Figure 1.4, Swoogle returns 346 documents when searching for spring. The top ranked results show that the same term has many different meanings, e.g. one spring means the season and the other spring means the ground water.

Figure 1.4: Swoogle snapshot

When the greater organization requires integrated data, problems emerge because database systems cannot “talk” or query one another directly. A mediator is required to

coordinate the communication and/or data exchange process just like as shown in Figure 1.5 where people try to say same thing in different ways.

Figure 1.5: People trying to say same thing in different way [12].

To solve such semantic heterogeneity among different information systems many approaches such as using synonym sets, term networks, concept lattices, features and constraints have been proposed [10]. However these approaches are not sufficient to solve the problem of semantic heterogeneity in many areas.

Ontologies are a key component to solve the problem of semantic heterogeneity, and thus enable semantic interoperability between different applications and services. Given the reality of multiple ontologies over many domains, ontology mapping that aims to find semantic correspondences between similar elements of different ontologies has been the subject of research in various communities [2].

Ontology mapping is “a necessary precondition to establish interoperability between agents or services using different ontologies” [13]. That is, the mapping between ontologies provides the means for agents and services to either translate Ontology mapping is also widely used to support data integration and information transformation.

For example, Figure 1.6 illustrates a simple scenario where data are structured in different formats in two data sources, D1 and D2, which are associated with ontologies O1 and O2 respectively. To integrate instances from D1 to D2, the mapping relation m between O1 and O2 is needed.

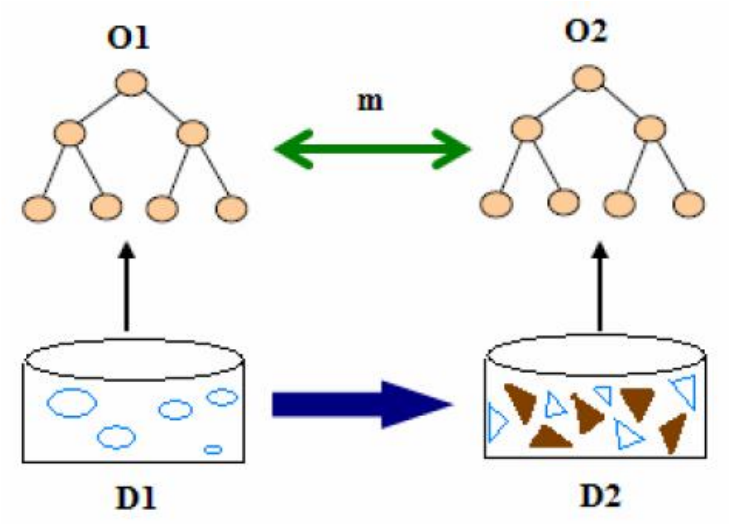

Figure 1.6: Ontology mapping in supporting Data Integration [12].

Many real world cases also demonstrate the need for ontology mapping to support schema/data integration. For example, a web marketplace such as Amazon may need to combine products from multiple vendors' catalogs into its own. A researcher may want to merge his/her bookmarks with those of his/her peers etc.

Ontology mapping can thus be used to define upper ontology which can be used as a framework for analysis of databases.

1.4 Thesis Outline

The thesis is divided into following six chapters.

Chapter 1: Introduction to Ontology, Multidatabase System and the type of semantic conflicts present to integrate schemas is discussed.

Chapter 2: Literature Review section gives brief survey of research going on in the area of semantic integration of heterogeneous databases in multidatabase system is discussed.

Chapter 3: After going through literature review, the problem statement has been identified and defined in this chapter

Chapter 4: The general concepts used in the thesis work are explained. It provides the details of integrating operators used for integration of databases.

Chapter 5: Algorithm for calculating the semantic relatedness and integrating databases to generate global schema is proposed. Results of implementation of algorithm are given.

Chapter 6: Conclusion of the work done and scope for future work is proposed in this chapter.

CHAPTER-2
LITERATURE REVIEW

Semantic integration has been a long-standing challenge for the database community. It has received steady attention over the past two decades, and has now become a prominent area of database research [1]. To provide a uniform interface and high level location transparency for the users to retrieve data in a heterogeneous database system, a global schema is usually created by integrating schemas of the component databases. This problem has been studied since the early 1980s [7, 14, 15].

A variety of approaches to schema integration have been proposed in [16, 17, 18]. The integration process requires establishing semantic correspondences or matches between the component schemas and then using the matches to merge schema elements [7, 14]. The multidatabase system's designers usually transform the local schemas into a common data model, compare local schemas to exploit their semantic correspondences and identify their schematic differences, merge the local schemas into a global schema (or a set of global schemas), and define the mappings between the global schema and the local schemas.

Irrespective of the approach used in the realization of a multidatabase system, it is vital to compare local schema contents to detect types of schematic conflicts that may exist among these local schemas as a natural consequence of local autonomy and evolution over time [3]. A lot of research is going on in the area of semantic heterogeneity detection and reconciliation.

The work of Kashyap and Sheth [19] proposes a model for capturing semantic similarities between objects (called semantic proximity) with which they represent semantic similarity between objects. Semantic similarity between objects is represented as SemPro(O_1,O_2)={Context, Abstraction, (D_1,D_2),(S_1,S_2)}. First component, Context represents the context in which objects are being compared, second component Abstraction is the mapping used to relate the domains of O_1 and O_2, the third component enumerates the domain definitions of O_1 and O_2, and finally the fourth component

enumerates the state of objects O_1 and O_2. The authors also provide taxonomy of schematic conflicts and introduce the concept of schema correspondences to capture structural similarities between objects. It is not clear in their work how the semantic similarity between two objects is inferred (manually, semi-automatic or automatic), and how the contexts are defined and built. Also the SemPro() function deals with two objects at a time. This is impractical seems when dealing with a very large number of objects.

Chen [20] describes an algorithm that integrates schemas semi-automatically. This algorithm assumes that inter-schema correspondences are provided in the form of assertions by the integrator. With this type of assertion it is possible to express a semantic relationship between the classes' parent and brother for example from a database with the class uncle from another database. This work does not take inheritance into consideration.

Li and Clifton [21] present a procedure that uses a classifier to categorize attributes according to their field specifications and data values. They train a neural network to recognize similar attributes. The list of discriminators they use includes data type, length, format specifications, existence of constraints (primary keys, foreign keys, candidate keys, and so on), disallowing null values and access restrictions. The focus of the work is to identify similar attributes by grouping them into different groups, where ultimately similar attributes will end in the same group. Their work utilizes only the information that is typically exists in schema definitions and database dictionaries and these are not enough to capture real world semantics of schema objects as it is widely understood in the database community. Also, their work deals with a very limited set of semantic conflicts. Finally, their works do not address mapping values between similar attributes that have incompatible domains.

Reddy et al. [22] has identified various types of problems like naming conflicts, type conflicts, key conflicts etc. that arise during the schema integration process. The methodology used for the creation of an integrated schema from a given set of local database schema involves acquisition of semantic knowledge pertinent to the objects of a local objects schema. During knowledge acquisition process, for each property of a local

object, parameters that contribute to the semantic meaning of the property are identified (such as meta-properties) and their values (meta-values) are captured. Further, concepts such as object equivalence class and property equivalence class are utilized to facilitate the creation of the integrated schema.

Ontology research is another discipline that deals with semantic heterogeneity in structured data. Prior related research on ontology can be considered from three different points of view. One perspective is related to building ontologies. This aspect of research discusses, e.g., “how to build a taxonomy tree” or “how good an ontology conforms to the conceptualization of a community”. Building a proper ontology in terms of its explication (i.e., how an ontology reveals implicit assumptions) and its accordance with the conceptualization of the community is an important research issue [2].
Another perspective considers the representation of ontologies and reasoning based on ontologies. Topics such as “how to represent an ontological definition” or “what features are important in representing ontologies for reasoning” are addressed.
The last perspective concerns semantic integration of database schemas. This topic addresses issues such as “how formal ontologies can help to solve heterogeneity problems “, “what kind of heterogeneity problem can it solve”, “how do we relate formal ontologies with schemas” or “how do formal ontologies interact with existing system architectures” [23].

Wache et al. in [13] give a survey on ontology-based information integration systems. According to the way of exploiting ontologies in information integration, they distinguish three main ontology based approaches: single, multiple and hybrid. Single ontology approach uses one global ontology to which all information sources are linked by relations expressed via mappings that identify the correspondence between each information source and the ontology. In multiple ontologies approaches, each information source is described by its own ontology and inter-ontology mappings are used to express the relationships between the ontologies. The hybrid approaches combine the two previous approaches. Each information source has its own ontology and the semantic of the domain of interest as a whole is described by a global reference ontology. In these

approaches there are two types of mappings between an information source and its local ontology and mappings between local ontologies and the global ontology.

Bergamaschi et al. [24] introduced an approach to integrate schemas by extracting similarity relations from schema definitions of component databases, not directly from ontology. The approach is a semiautomatic relation extraction based on schema definitions and needs supervision of an expert. Based on the extracted relations they introduce an algorithm to integrate schema definitions into a global homogeneous schema.

After going through the research done in the area of semantic heterogeneity and schema integration in multidatabase systems various semantic conflicts and challenges in schema integration have been studied. Using ontology mapping for databases as framework, where an approach is followed to form upper level ontology by defining mapping between different ontologies is used to detect semantically related classes and to integrate the local schemas by integrating operators to generate global schema.

CHAPTER 3
PROBLEM STATEMENT

3.1 Problem Definition

A multidatabase system (MDBS) is a database system that resides on top of existing local autonomous component databases systems and presents a single database to its users. MDBS usually maintains a single global database schema, which is integration of all component database schemas. As the amount of data and the number of data producers are growing, it is becoming increasingly difficult to establish semantic interoperability among heterogeneous information sources and performing semantic schema integration in multidatabase system. Interoperability has to resolve the differences in data structures, and deal with semantic heterogeneity. Syntax defines the structure of the schema items while semantics refer to the meaning of data. In this work focus is on the part of semantics related to the meanings of the terms used as identifiers in schema definitions.

Taking semantically heterogeneous databases as the prototypical situation, it has been described how ontology (in the traditional metaphysical sense) can contribute to delivering a more efficient and effective process of matching by providing a framework for the analysis, and so the basis for a methodology. The objective is to use object-oriented approach to integrate schemas of distributed heterogeneous autonomous local component database schemas into a global schema. The resulting global schema provides a uniform interface and high level of location transparency for retrieval of data from the local component databases. To provide a model independent representation of virtual classes of the global schema, a set of integration operators are defined to integrate local schemas based on the semantic relevance of their classes.

3.2 Methodology

The step-by-step methodology followed to perform semantic integration in heterogeneous multidatabase system.

- Study of Heterogeneous Multidatabase System.
- A thorough analysis of various semantic heterogeneity conflicts and semantic integration techniques.
- Study of Ontology and its application to resolve conflicts in heterogeneous databases
- Design and Implementation of Algorithm to generate virtual global schema by detecting and integrating semantically related classes from a heterogeneous local component schemas.

CHAPTER 4
INTEGRATION OF HETEROGENEOUS DATABASES

4.1 Multidatabase System

A multidatabase system is a system that supports the uniform access to a set of autonomous databases called local or component databases. It allows its users to access and update data from multiple, pre-existing, autonomous, and probably heterogeneous databases [10]. The multidatabase system automatically performs query decomposition, together with data model and access language transformations, to convert a global query into an appropriate set of sub-queries for the local databases. If the query produces a response, the sub-results coming from individual databases are then transformed and composed into the format of the multidatabase system response language. Distribution, heterogeneity, and the effects of local autonomy of local information sources are transparent and therefore users of the multidatabase are given the illusion of logically integrated information. Preferably, local database administrators retain full control over their local information and operations; this protects an organization's existing investment in hardware, software and user training [25]. The schema contains the format, structure, and organization of the data in a system and multidatabase system maintains only the global schema and the component database system actually maintains all user data. Multidatabase systems permit data sharing among users and application programs which lead to easier application program development and their construction does not disturb the operation of existing application programs.

The architectures of multidatabase systems are divided into two broad categories namely tightly coupled and loosely coupled multidatabase systems. In a tightly coupled multidatabase system, data is accessed using global schemas, which are created and maintained by the multidatabase system administrators. In a loosely coupled multidatabase system, users are responsible for creating their federated schemas, either by using import/export schemas or by using a powerful query language [3].

In general, multidatabase systems realization techniques are broadly divided into techniques that support users with a powerful query language that enables them to access data that is stored in different local databases [14]. The second alternative of realizing a

multidatabase system is through schema integration [2, 3, 17]. Using this technique (Figure 4.1), the multidatabase system's designers usually transform the local schemas into a common data model, compare local schemas to exploit their semantic correspondences and identify their schematic differences, merge the local schemas into a global schema or a set of global schemas, and define the mappings between the global schema and the local schemas.

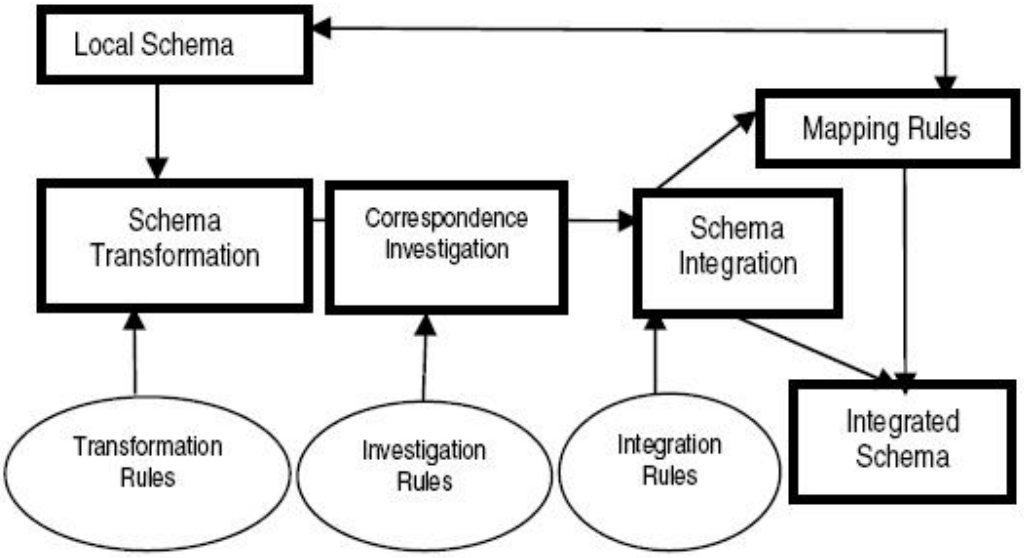

Figure 4.1: **Generic framework for integrating heterogeneous local databases [27].**

Schemas transformation is a process to transform heterogeneous database schemas into the global common data model and also to support query expressed in the global common data manipulation language have to be decomposed and translated into the local query language and also query produces a response, the sub-results coming from individual local schemas are then to be transformed and composed into the format of multidatabase system response language. A schema transformation should only be applied to restructure a schema if the resulting schema is equivalent to the original schema. Correspondence assertions mean correspondence between tables and classes, correspondence between columns and attributes. Based on these assertions, integration rules are designed, the work of [28, 29], show different method of using a set of primitive integration operators to do the integration. Various restructuring operators are proposed to rename or restructure schema objects of component schemas to resolve conflicts between them. The mapping information is important for the decomposition of a global query against global

schema and for this mapping strategy between the global schema and component schemas is necessary i.e. mapping for attributes and object instances among a virtual class and its constituent classes (classes in the component schemas).

Regardless of the approach used to realize a multidatabase system exploring the local databases to determine their respective semantic content is vital and is considered the most difficult task in the realization of a multidatabase system [3]. The benefits of a multidatabase system cannot be exploited unless the correspondences between local schema elements are clearly identified. As much of the semantic of the database schemas and application programs are in the mind of database designers and users, this makes the process of comparing several databases, to discover their semantic relevance, a very complicated task that depends on the human integrator. The World Wide Web is adding an additional dimension to multidatabase systems realization; resources that can join a multidatabase system are increasing in number and size [5]. Creating and maintaining the global schema, which requires the use of database integration techniques, is a critical issue in multidatabase systems. Major issue in schema integration is associated with combining diverse schemas of the different databases into a coherent global view by reconciling any structure or semantics conflicts between the local component databases [22].

4.1.1 Object-Oriented Data Model

Object orientation perceives real world entities as sets of objects, where an object is an abstract representation of a real world entity that has a unique identity, embedded properties and the ability to interact with other objects and with itself via messages [28]. Object-oriented data model is chosen as common data model to construct a global conceptual model by set of integration operators. The object-oriented model has grown in popularity because of its simplicity in representing complicated concepts in an organized manner or we can say due to its semantic richness and availability. It can also be said that in object-oriented data model there is only one concept objects as opposed to the two concepts entities and relationship of the entity relationship model. Using two concepts may cause problems in the process of schema integration and schema transformation because the same real world object may be modeled as entity in one schema but as a

relationship in another schema [29]. As shown in Figure 4.2 objects represents single concept where each component database participating in multidatabase are represented as an object with a declared interface and hidden implementation. Multidatabase system is seen as a set of interacting objects whose interaction is achieved through message passing. Heterogeneity is supported because details are hidden by encapsulating an object's implementation. Autonomy is supported because the constituent objects are allowed to change their implementation freely as long as they keep their interfaces unchanged.

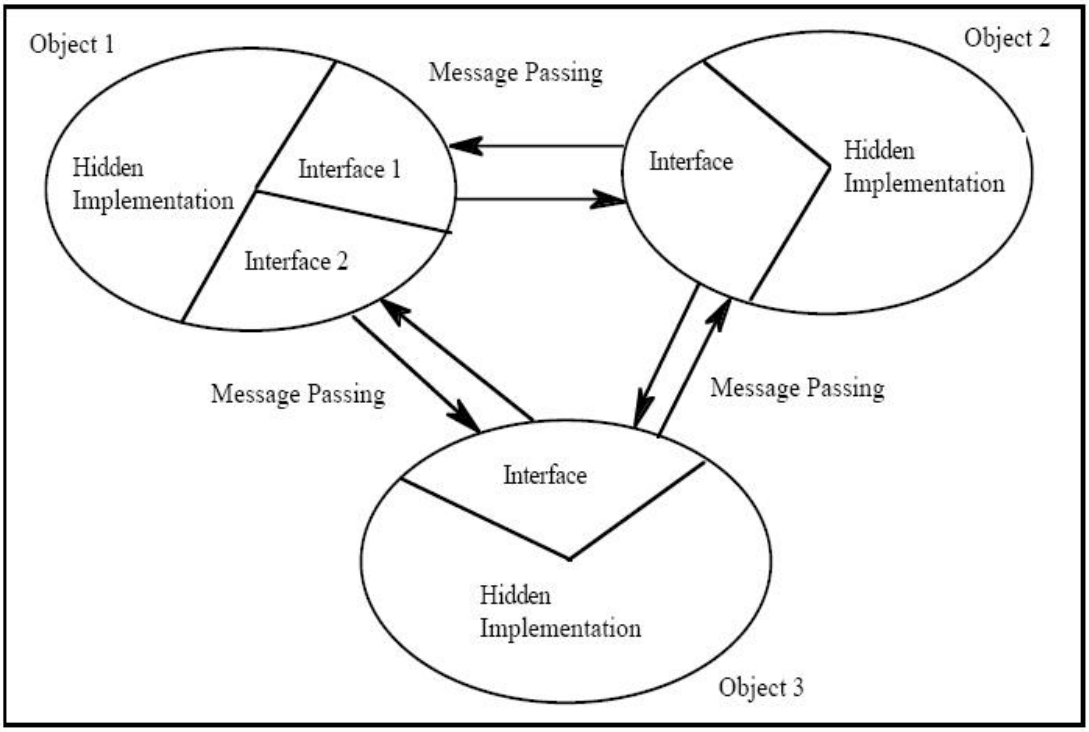

Figure 4.2: A Multidatabase System based on distributed object architecture [3].

Using encapsulation, methods, and inheritance, it is possible to define almost any type of data representation. Consequently, object-oriented models have been used not only to model the data, but also to model the data models. That is, object-oriented models have seen increased use as canonical models into which all other data models are transformed into and compared. The object oriented model has very high expressiveness and is able to represent all common modeling techniques present in other data model. In general, tightly

coupled systems integrate the diverse sources of intelligence through a global conceptual scheme, normally denominated canonical model, providing a uniform vision of the diverse component at a high level [28]. The use of a canonical model hides the structural differences between the different components and gives to the user the illusion to be accessing a simple centralized database. The global schema proposed in an object-oriented data model is a virtual schema because no actual data are stored for this schema. Thus, the classes in the global schema are called virtual classes and the objects associated with the classes are called virtual objects.

4.1.2 Heterogeneity in Multidatabase System

The system heterogeneity concerns the differences in the low level architectural platforms local component databases, such as hardware configuration, operating systems and communication facilities. At these level, different systems techniques is used by different DBMSs such as query processing strategy, concurrency control mechanism and transaction management facilities. The physical heterogeneity/syntactic heterogeneity refers to the differences in the data representation. It may concern the encoding of same data values into integers or real of different types. Next, it may concern different implementation of the data model, e.g., of tables in different relational systems. The formats of manipulation procedures, e.g., of SQL queries, also often differ. Finally, the data models themselves can differ. Data of interest may be in a relational database, and in an object-oriented database.

The Schematic and semantic heterogeneity refers to the discrepancies between data names, values, and the conceptual structures. Data heterogeneity refers to difference among local definitions, such as attribute types, formats, or precision [30]. Schematic heterogeneity means, all component databases have different data model structure where equivalent and related data concepts are present and having conflicting structural representation. It can also be said that all local component databases model having same real world concept but structured in a different manner. Semantic heterogeneity refers to differences or similarities in the meaning of local data. For example, two schema elements in two local data sources can have the same intended meaning, but different names. Thus, during integration, it should be realized that these two elements actually

refer to the same concept. Alternatively, two schema elements in two data sources might be named identically, while their intended meanings are incompatible. Hence, these elements should be treated as different things during integration. It is vital to compare local schema contents to detect type of schematic conflicts that may exist among these local schemas. Schematic heterogeneity arises when information that is represented as data under one schema, is represented within the schema in another, for example as relation or class names. Semantic relationships between the local databases determine how local classes should be integrated in the global schema. Various types of semantic relationships are possible between different schema objects. These can be defined in many ways, such as in terms of the real world objects they represent or with respect to their intended meaning and use [8]. Schema objects in different databases are either semantically related or semantically incompatible. Semantically related schema objects represent the same real world concept. Semantically incompatible objects, on the other hand, are objects that cannot be semantically related and therefore cannot be merged into a global object when building a multidatabase.

4.2 Ontology Based Approach for Database Integration

For the purposes of database integration, the traditional philosophical (metaphysical) notion of ontology is useful where it is the set of things whose existence is acknowledged by a particular theory or system of thought [13]. This view was summarized by Quine, who claimed that the question ontology asks can be stated in three words ‘What is there?’ and the answer in one ‘everything’.

From the perspective of database integration, each database can be regarded as a theory that acknowledges the existence of a set of objects its ontology [11]. Some care needs to be taken to distinguish this traditional metaphysical use of the word ontology from one that has recently developed in Computer Science where an ontology is regarded as a specification of a conceptualisation [23] and has been applied to a wide range of things, including dictionaries.

Ontology provides a framework and suggests a process for the analysis needed for semantic matching. This process focuses on the semantics of the database, identifying

semantic divergence and it aims to purge this divergence to produce an ontological model.

Explicit and formal definition of semantics of the terms guided many researchers to apply formal ontologies as a potential solution of semantic heterogeneity. A formal ontology consists of logical axioms that convey the meaning of terms for a particular community [23]. Consensus on ontological definitions among members of a community is an important difference between ontologies and conceptual models. While conceptual models are application-dependent, ontologies are only based on people's understanding. Formal ontology is considered more than schema definitions in databases. Schemas are mainly concerned with organizing data in databases; formal ontologies are concerned merely with the understanding of the members of the community and helps to reduce ambiguity in communication. It is important to note that schema definitions are based on the ontology definition and vice versa, they convey part of the knowledge about ontology of a community.

4.2.1 Role of Ontology for Global Schema Generation

There are mainly two trends for using ontologies in resolving semantic heterogeneity. One uses ontologies for translating queries, or their results [11]. This approach is suitable whenever schemas are subject to frequent changes such as DTDs in XML data, when many data sources are involved, or the number of involved data sources changes frequently such as data sources in the Internet. Limitation of this approach is high processing cost, since for every query, ontologies must be processed to derive required mappings also human supervision to validate extracted similarities is not possible, due to the need for immediate action and lack of human supervision makes this approach less reliable.

Another approach uses ontologies for the generation of global schemas [23]. It is suitable whenever the schemas are not subject to frequent changes. In this approach, database schemas commit to the ontology of a community. This is done by relating every term in the schema delineations to a definition in the ontology of the community. In Figure 4.3 two databases DB_{p1} and DB_{p2} commit to an Ontology p by referring to the terms defined

in the Ontology p. Such relation can be established either by hard links or by using the same terms as they are defined in the ontology.

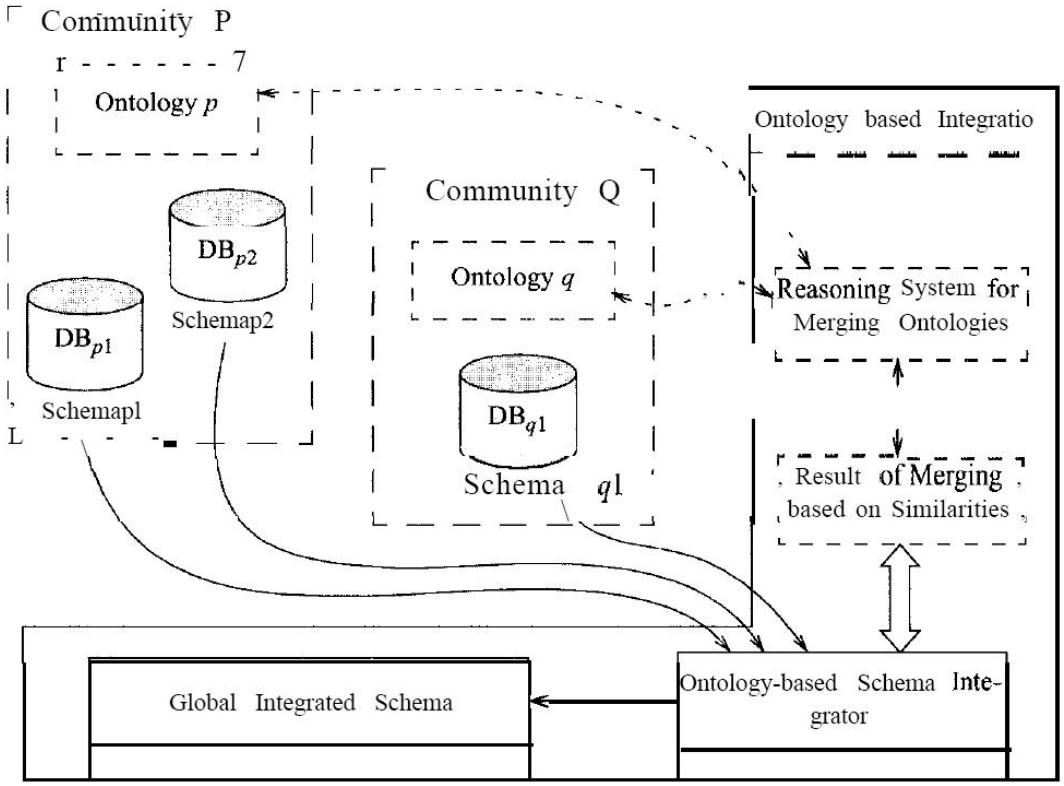

Figure 4.3: Global schema generation based on a common ontology produced by integration of ontologies [23].

In order to find out whether and how elements from different schemas are related, we use similarity relations are used. Global schema can be generated by a schema integrator based on the result of merging ontologies. Thus, with ontological approach not only suggest a global schema, but also try to find all the possible meaningful mappings between the generated global schema and the component schemas for instance, the schema integrator suggests a class in the global schema as well as the mapping of this class and its attributes into one or more classes in the underlying local schemas. As the number of underlying databases and the communities increase, the number of derived mappings increases, even though many of them may not be used by the applications. A statistical analysis or human supervision to maintain only the valid mappings and schema items can help to overcome this disadvantage of increase of unrelated mappings.

4.2.2 Merging Ontologies by Means of Similarity Relations

Merging ontologies is based on finding similarities or differences between schema definitions. Based on the four concepts of similarities mentioned below, similarity relation between two coherent schema definitions can be identified. Let RWS () be a function that represent real world semantics of an ontology.

- **Disjoint definition:** This level has the lowest degree of similarity. Two ontology p and q of different community are said to be disjoint if RWS(p) ∩ RWS(q) = Ø for instance, narrow street and high way truck.
- **Overlapping definition:** Two ontologies p and q are said to overlap if RWS(p) ∩ RWS(q) ≠ Ø i.e. the conjunction of two class definitions cannot be proven to be false.
- **Inclusion definitions**: Ontology p is included by another ontology q if RWS(p) is subset of RWS(q). For instance, “man” is a subconcept of “person”.
- **Equal definitions:** This level has the highest degree of similarity. Two ontologies p and q are said to be equal if RWS(p) = RWS(q) for instance, “vehicle” and “transportation facility” are equal if they have the same definition.

Deriving similarities between ontologies requires common references in two ontologies and a reasoning system for matching. The common references can be provided by a higher level ontology such as the library of ontologies on the Ontolingua [31] site or by a thesauruses such as WordNet [32]. Finding similarities can also be done by experts familiar with both communities or by a hybrid semiautomatic method.

The similarity relations are used to merge two ontologies. The above discussed similarity relation definitions in the respective communities are considered for merging process and explicitly establish similarity relations

- If two definitions are equal, the result of merging is a unique definition which is referred to by both original terms. That is, different terms in the local schema definitions can refer to the same concept for example synonym terms such as “Person” and “Resident” as shown in Figure 4.4
- If definition C_i specializes C_j then the subconcept or subrelation similarity will be explicitly established between them (e.g., “Student” and “Person” in Figure 4.5).

- If a definition C_i overlaps with C_j then an additional new concept or relation will be declared as the conjunction of the two definitions. Although the conjunction of the two definitions may not be proven false, we may not have any instance of such a concept, practically. Yet, if instances of such overlapping concepts exists, the relevance of the new overlapping concept needs supervision of an expert e.g., concepts "Staff' and "Student" or "Lecturer" and "Graduate student" in Figure 4.5. Deciding whether such a concept is relevant for the application domain or not should be done by an expert. Assigning a term to the new concept or relation can be done automatically or by an expert, as well.

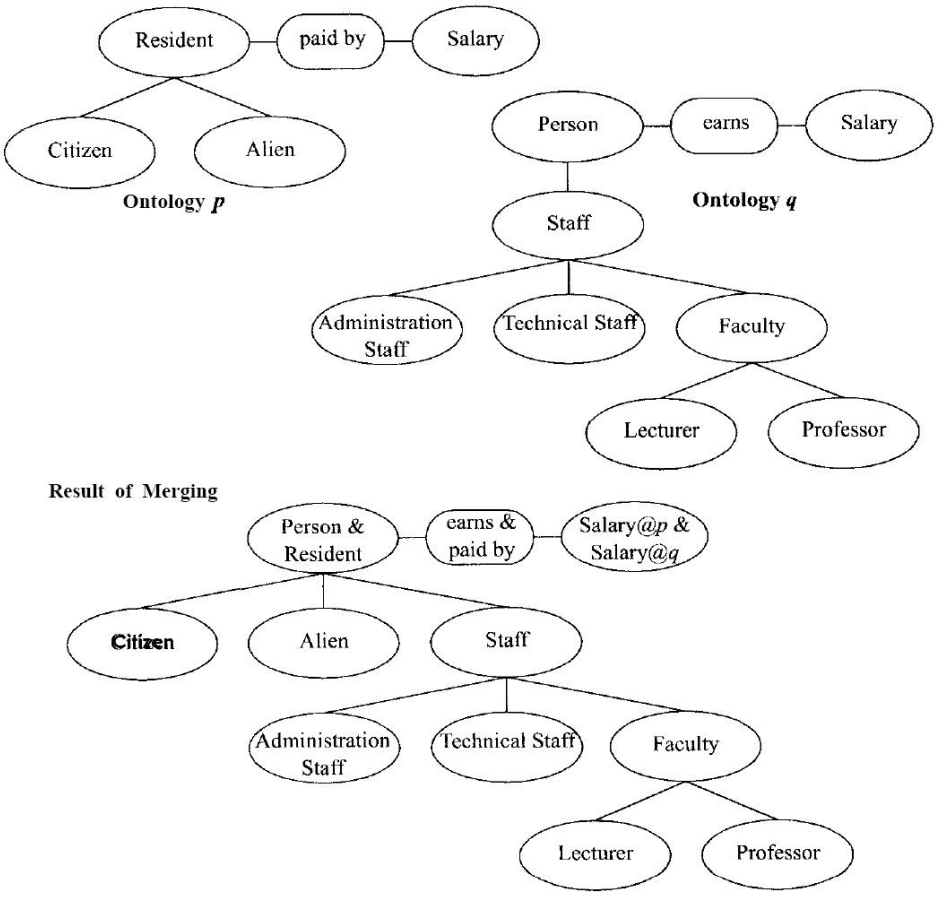

Figure 4.4: Merging Ontologies using the concept of equality [23]

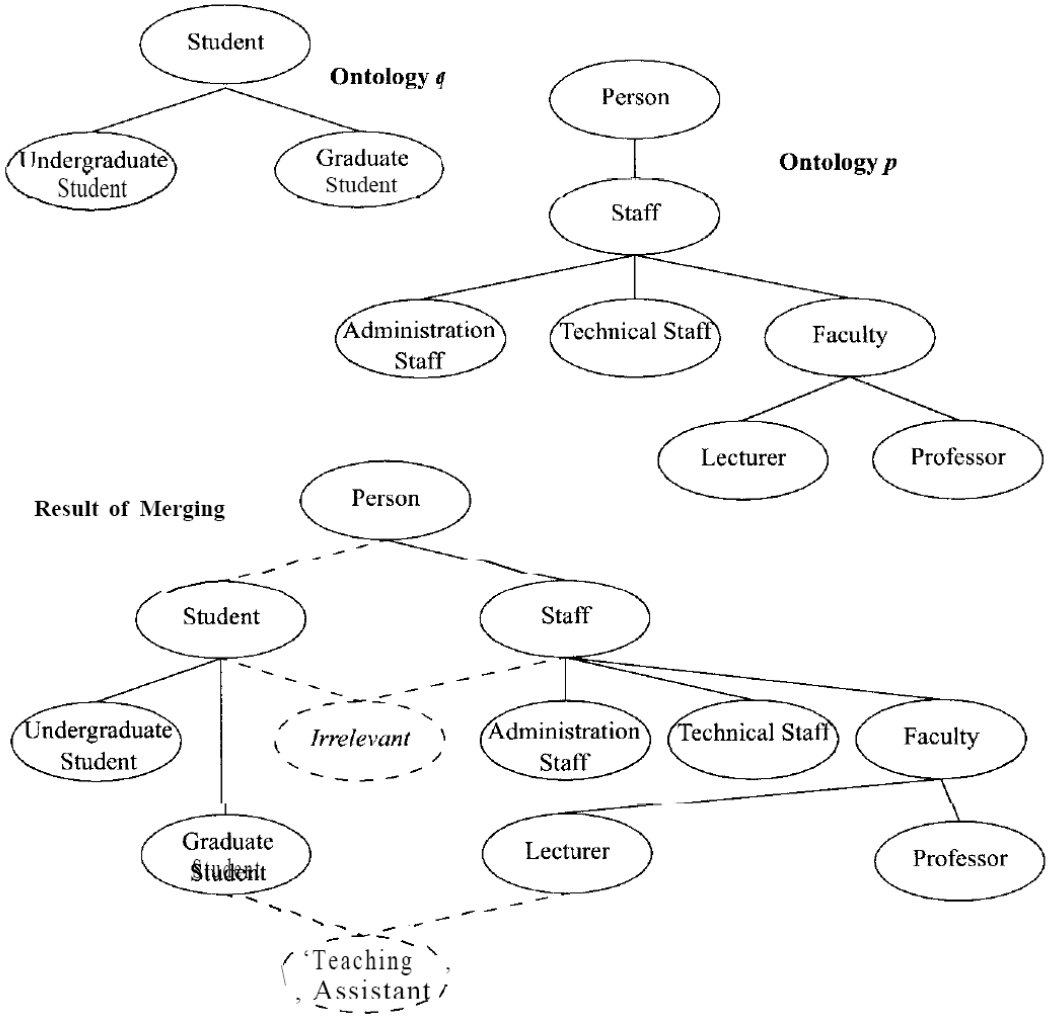

Figure 4.5: Merging ontologies using the Inclusion relation [23]

4.3 Proposed Integration Approach

Using above ontology approach as a framework and considering the similarity concepts and semantic heterogeneities discussed, an integration approach using integrating operators for global schema generation is proposed.

4.3.1 Deriving Semantically Related classes

From the above similarity concept used for ontologies, semantic relationship are derived through which classes can be related to each other by determining how local classes should be integrated in the global schema. Considering LDB_1, LDB_2, LDB_3…….LDB_n as n local or component schemas and LC_1, LC_2, LC_3……..LC_n as n local classes in each

component schema. Let add() be a function that returns the expanded or added arguments in global schema and type() be a function that returns the set of classes.

- **Equivalent Classes:** A class LC_1 of local schema LDB_1 is said to be equivalence to class LC_2 of local schema LDB_2 if it represents same real world concept and have same schema name regardless of difference in number of attributes and data types. For example, class LC_1-Student (roll-no, name, age, class, result) and class LC_2-Student (regn-no, name, fathers_name, birth-date, class, result) are considered equivalent even though LC_1-Student has five attributes and LC_2-Student has six attributes and result of LC_1-Student is in form of CGPA and marks of LC_2-Student is in the form of percentage. The extents of LC_1-Student and LC_2-Student could be identical ($LC_1=LC_2$), disjoint ($LC_1 \cap LC_2 = \emptyset$) or overlapping ($LC_1 \cap LC_2 \neq \emptyset$).
- **Synonymous Classes:** These Classes have the same definition as equivalent classes except that the names of class LC_1 and LC_2 are synonymous and have different name.
- **Homonymous Classes:** Two classes are said to be homonymous if they represent different real world concepts but have the same name. These classes cannot be merged into a common class in the global schema as it is semantically heterogeneous.
- **Containment of Classes:** A class LC_1 of local schema LDB_1 is said to be contained in a class LC_2 of local schema LDB_2 if type (LC_2) is subset of type (LC_1) and add (LC_1) is a subset of add(LC_2). LC_1 is a subclass of LC_2. For instance, LDB_1-Employee (number, name, job, birth-date, salary, address) is a sub class of LDB_2-Person (no, name, birth-date, address). The difference in the name of the key can be seen (number in LDB_1-Employee and no in LDB_2-Person, still, in a unified global representation LDB_1-Employee is a subclass of LDB_2-Person).

Semantic relationship is defined at the attribute level many attributes are said to correspond to each other and therefore can be represented as one set of global attribute. For example consider two classes LC_1 and LC_2 of schema LDB_1 and LDB_2 respectively

where class LDB_1-Student (roll-no, name, age, class, result) and class LDB_2-Student (regn-no, name, fathers_name, dob, class, result) has semantically related attribute where:
roll-no = regn-no
name = name
age = map(dob)
class =class
result = map(result)
Here map(result) is a mapping function that converts percentage into CGPA for result and map(dob) converts it into age. Thus it provides mapping to generate global set of attributes.
User defined global schema can also be generated based on the user's perspective whether he/she wants the result in CGPA format or percentage format or whether father's name should be included or not in the set of attribute or age should be represented in DOB format or directly as age.

4.3.2 Integrating Operators

The integration operators are applied to set of local classes to generate virtual classes. Thus global schema or integrated schema constitutes of virtual classes constructed from the set of local classes of component databases participating in multidatabase system.
The selection of integration operator depends on semantic relevance present between the local classes or on the basis of the user's perspective. Following are the set of class integration operators to generate global class for an integrated schema in multidatabase.

- **Generalize Operator:** The generalize operator creates a common superclass of $Class_1$ from LDB_1 and $Class_2$ from LDB_2 (as shown in Figure 4.6). It integrates local classes by generating a common superclass to them. Before integration using generalize operator, local classes have to be related with equivalence, synonymy or containment semantic relationships.

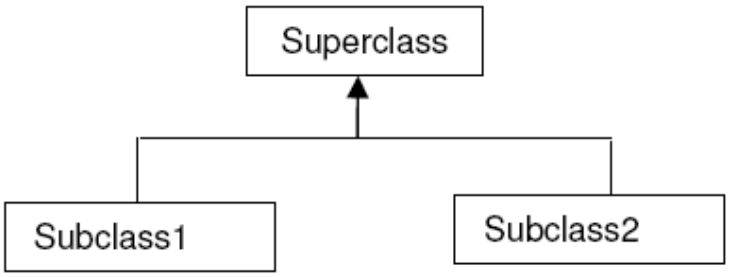

Figure 4.6: Generalization Concept

From n local databases LDB_1, LDB_2, LDB_3……LDB_n with n local classes C1, C2, C3……Cn, the global class can be generated using generalize operator which derives common attributes from all the local classes and by taking union of all the objects of the local class. Thus attributes of virtual class (VC_a) and virtual objects (VC_O) of virtual class is represented as follows

$$VC_a = C1_a \cap C2_a \cap \ldots\ldots\ldots \cap Cn_a$$

$$VC_O = C1_O \cup C2_O \cup \ldots\ldots\ldots \cup Cn_O$$

For instance, as shown in Figure 4.5, Global class is generated by integrating common attributes of the local classes bus, car, plane and boat.

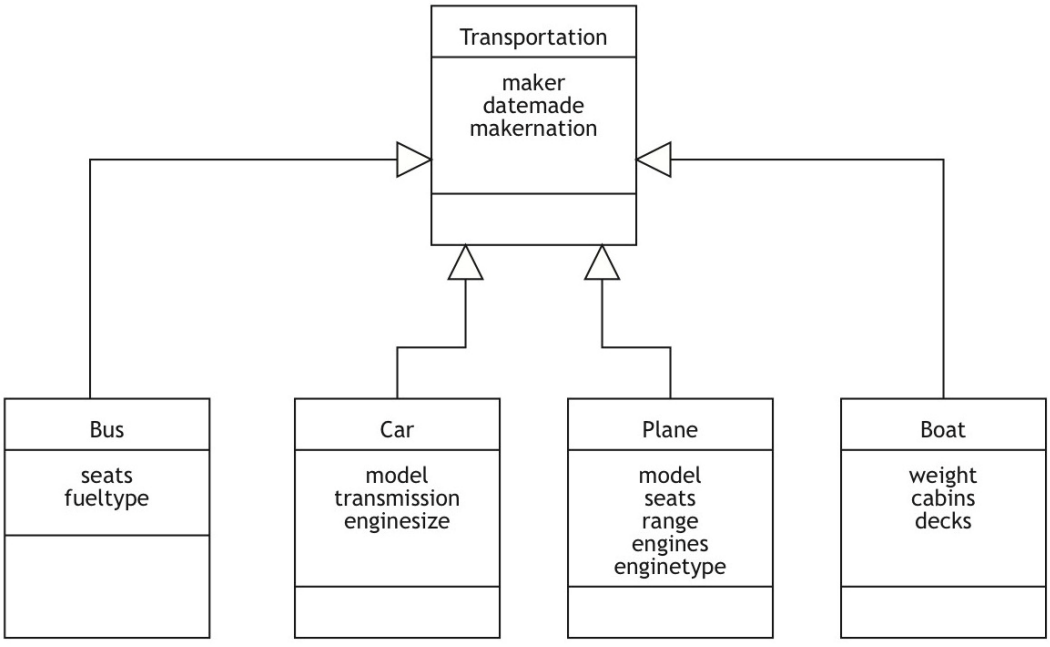

Figure 4.7: Generalisation of local classes.

❖ **Specialization Operator:** It is reverse of generalization. The specialization operator creates a common subclass of Class1 from LDB_1 and Class2 from LDB_2 (as shown in Figure 4.8). It integrates local classes by generating a common subclass to them. Before integration using specialize operator, local classes have to be related with equivalence, synonymy or containment semantic relationships.

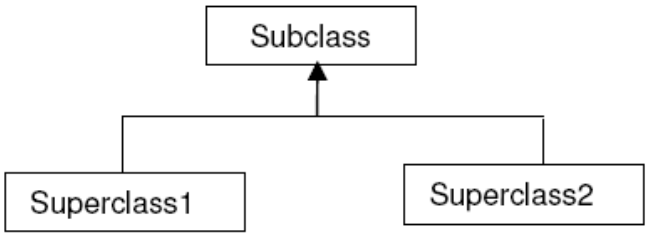

Figure 4.8: Specialization Concept.

From n local databases LDB_1, LDB_2, LDB_3……LDB_n with n local classes C1, C2, C3……Cn, the global class can be generated using specialize operator which performs union of attributes from all the local classes and by taking intersection of all the objects of the local class. Thus attributes of virtual class (VC_a) and virtual objects (VC_O) of virtual class is represented as follows

$$VC_a = C1_a \cup C2_a \cup \ldots\ldots\ldots U\ Cn_a$$

$$VC_O = C1_O \cap C2_O \cap\ \ldots\ldots\ldots \cap Cn_O$$

For instance, as shown in Figure 4.9, Global class Tutor is generated by taking union of local classes student and employee.

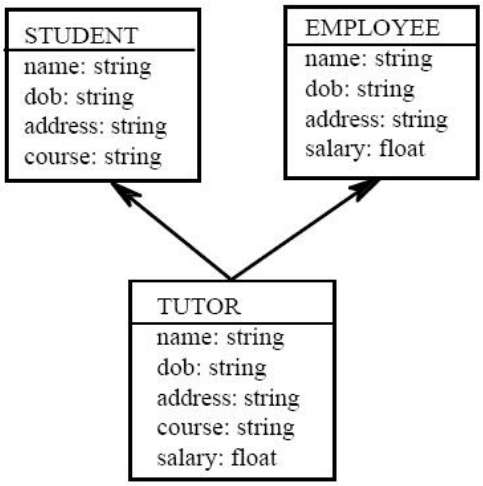

Figure 4.9: Specialization of local classes [3].

- **Combine Operator:** The Union operator integrates C1 from LDB_1 and C2 from LDB_2 into a virtual class as new class. Only new class will appear in the Global Schema. It merges equivalent or synonymous local classes into a single virtual class. The type of the virtual class is the union of all attributes that belong to C1, C2,.....…..Cn, taking into consideration that equivalent local attributes appear once in virtual class regardless of their schematic conflicts.

  Thus if Considering n local databases LDB_1, LDB_2, LDB_3……LDB_n with n local classes C1, C2, C3……Cn, the global class can be generated using Combine operator by taking union of all the attributes of the local class. Thus attributes of virtual class (VC_a) and virtual objects (VC_O) of virtual class is represented as follows

$$VC_a = C1_a \cup C2_a \cup \ldots\ldots\ldots \cup Cn_a$$

$$VC_O = C1_O \cup C2_O \cup \ldots\ldots\ldots \cup Cn_O$$

  For instance as shown in Figure 4.10 Global class Person is generated by taking union of all the objects and the attributes of the local schemas Citizen, Foreigner, Lecturer and Prof

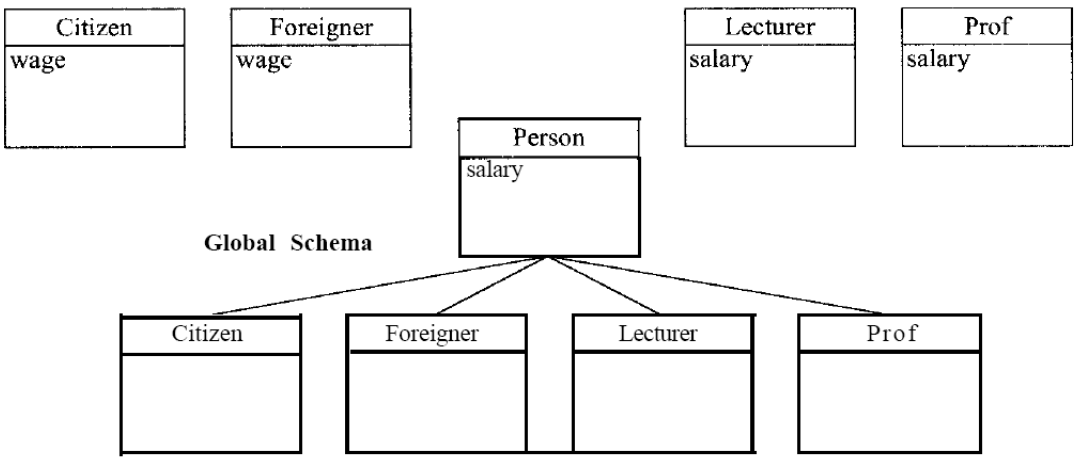

Figure 4.10: Combine operator to merge local classes.

- **Import Operator:** Import operator is responsible for importing local classes to the global level without merging them with classes from other local schemas. It provides direct mapping.

To import class C1 from LDB_1, the attributes of virtual class (VC_a) and virtual objects (VC_O) of virtual class is represented as follows

$$VC_a = VC1_a$$

$$VC_O = VC1_O$$

- **Aggregate Operator:** The aggregate operator is an overloaded operator and can be applied to both objects and attributes of the local classes. It integrates local class by aggregating them into higher level aggregate class.

  For n local databases LDB_1, LDB_2, LDB_3……LDB_n with n local classes C1, C2, C3……Cn, attributes of virtual class (VC_a) are specified by user as an argument to the operator. The virtual objects (VC_O) of virtual class is simply the union of all the other objects of local class and it is represented as follows

  $$VC_O = C1_O \cup C2_O \cup \ldots\ldots\ldots \cup Cn_O$$

  For instance as shown in Figure 4.11 Product class in global schema contains three attribute where price1 is from DB1 and price2 is from DB2. This class is useful if to compare prices of both the databases.

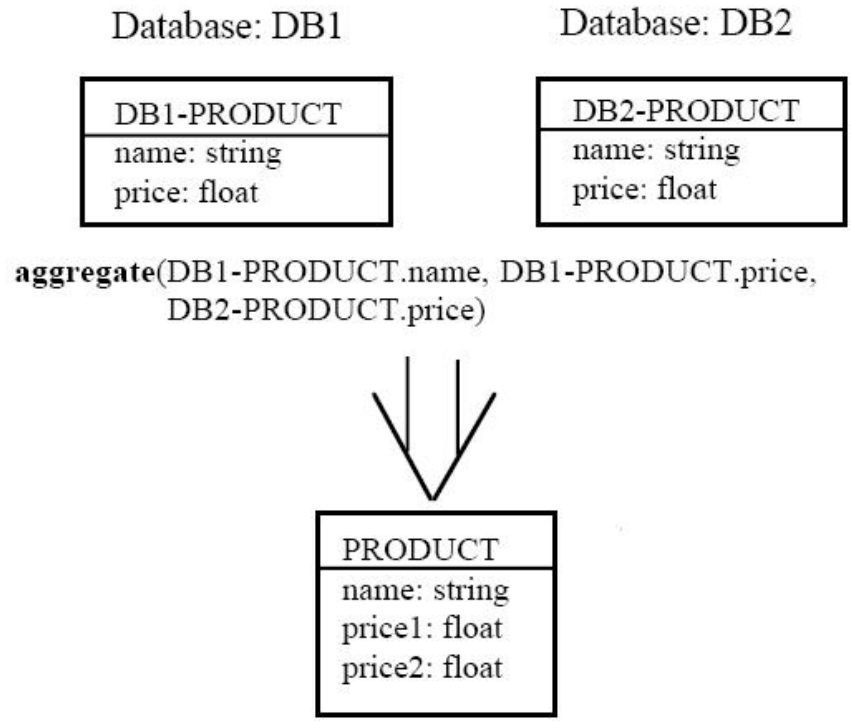

Figure 4.11: Aggregate operator to generate global class [3].

From the above discussion, an algorithm is designed which give step by step process to generate global schema containing virtual classes by detecting semantically related classes in heterogeneous databases and by using integrating operators discussed above to integrate semantically relevant classes.

The implementation of Algorithm for integrating operator is done in C++. Algorithm and the implementation is discussed in next chapter.

CHAPTER 5
PROPOSED ALGORITHM AND RESULTS

To perform semantic integration of heterogeneous databases an algorithm is proposed which identifies semantically related classes and integrates them using the integrating operators. The approach used for integrating databases is based on the concept of integrating ontologies.

5.1 Algorithm

Let LC_1, LC_2,……..LC_n be local classes of the component database schemas LDB_1, LDB_2……..LDB_n where n is the number of the local databases participating in Multidatabase System. A global schema is a set of virtual classes VC_1, VC_2……VC_n generated by integrating these local classes where n is number of classes in global schema. Only semantically related classes from local databases can be integrated to generate a global class.

5.1.2 Detecting Semantic Relatedness

To find semantic relatedness between classes of heterogeneous local database following steps are performed.

A. A database is created of maximum possible names of a class. For example, if a university database is considered then each local database will contain student class with different names such as stud, student etc. A database of these names is created and each name is assigned a weight between 0 and 1 on the basis of their frequency of usage. For example student is used more commonly as compared to stud so student is assigned less weight as compared to stud.
B. The name of the classes from each local database is related by assigning weights on the basis of their semantic relatedness. If the name is completely semantically related then they are assigned weight 0. As the relatedness decreases, the weight increases. It ranges in between 0 and 1. If the name of the classes is totally heterogeneous then they are assigned with weight 1. Heterogeneous or unrelated classes are not included for integration process.

Semantic relevance of name of local class LC_i class is thus given by equation (a)

$$N_i = Wt(LC_i) \text{ and } i=1,2,\ldots n \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{(a)}$$

where N_i is measure of semantic relevance of name of the class LC_i, Wt is a function that returns corresponding wt. of the class of local database LC_i obtained from the database of relevant terms. Here 'i' represents the i^{th} local database and n is the number of component databases.

C. Next step is to find the semantic relatedness between the attributes of the class of each local class. Considering a general template of a class, mapping between the attributes is found, for *e.g.* if we have employee class with salary in dollars in one local database and employee class with salary in rupees in another local database, a mapping function needs to be defined. A conversion function can be defined for converting the dollars into rupees.

i) Thus to find semantic relevance between attributes, two things need to be evaluated. First is to find the semantic relevance of the name of the attribute with that of template class. The name of the corresponding attribute could be either similar, synonymous or a substring. For *e.g.* Location in one database can be Loc in another database could be Place or in third database it could be Location itself.

ii) Second is to see if the domain of the attribute is similar to the domain of the attribute of the generic template of the class considered. Two domains are similar if a mapping from the first domain to second one can be defined. The template could be any class from one of the local database or a default template could be designed.

If Domain (Att_j of LC_i) = Domain (Att_j of LC_k) then

$$Att_j \text{ of } LC_i = map(Att_j \text{ of } LC_k) \ldots\ldots\ldots\ldots\text{(b)}$$

where Attj is the j^{th} attribute of local class LC_i and j=1, 2, ….., m_i

Att_j is the j^{th} attribute of local class LC_k and j=1, 2, ……, m_k

i, k-represents i^{th} and k^{th} local database respectively and

m is the number of corresponding attributes.

map() is a function that defines mapping between the attributes and LC_k is considered as a generic template class.

iii) Semantic relatedness of attributes of class can thus be calculated as follows:

$$A_i = 1 - \frac{\sum corresponding\ attributes\ between\ classes\ LC_k and\ LC_i}{\sum No.of\ attributes\ in\ LC_k + \sum No.of\ attributes\ in\ LC_i} \ldots (c)$$

where i, k-represents i^{th} and k^{th} local database respectively.

D. From equation (a) and (c), total relatedness(R_i) of each class can be calculated using following equation

$$R_i = \alpha \cdot (N_i + A_i) \ldots\ldots (d)$$

Where α is multiplication factor to keep R_i in the range 0-1

E. A threshold value(T) is assumed between 0 and 1 and the total relatedness is compared with this value

If ($R_i < T$) then

The class is considered to be semantically related

else

It is treated as semantically heterogeneous and is not included in integration process.

5.1.2 Integration Method ( )

Using integrating operators, virtual global class is created to integrate semantically related classes. Steps followed for integration are:

A. Input the relevant local classes LC_1, LC_2,……………………………LC_n of local databases LDB_1, LDB_2,……………………………LDB_n respectively.

B. Select the Integrating operator to be used for integration

a. If Integrating operator == Generalize then

i. Global _attribute _list = Intersect($att\_arr_1[]$, $att\_arr_2[]$….$att\_arr_n[]$)

ii. Global_class = Union(LC_1,LC_2…….LC_n)

$att\_arr[]_i$ is the attribute list of local class LC_i where i=1,2,…….,n

b. Else if Integrating operator == Specialize then

i. Global _attribute _list = Union($att\_arr_1[]$, $att\_arr_2[]$….$att\_arr_n[]$)

ii. Global_class = Intersect(LC_1,LC_2…….LC_n)

$att\_arr[]_i$ is the attribute list of local class LC_i where i=1,2,…….,n

c. Else if Integrating operator == Combine then

i. Global _attribute _list = Union(att_arr$_1$[], att_arr$_2$[]….att_arr$_n$[])

ii. Global_class = Union(LC_1,LC_2…….LC_n)

att_arr[]$_i$ is the attribute list of local class LC_i where i=1,2,……..,n

d. Else if Integrating operator == Import then

i. Input the Local database to be imported

ii. Global _attribute _list = attribute list of selected local database.

iii. Global_class = local class of selected local database.

e. Else if Integrating operator == Aggregate then

i. Global _attribute _list = argument list given by user

ii. Global_class = Union(LC_1,LC_2…….LC_n)

C. Display the Virtual Global schema generated.

5.1.3 Global Schema Generation Algorithm

Input: Set of Local classes of heterogeneous component database

Output: A Virtual Global Schema

Steps:

// To detect semantic relevance of local classes

i. **for** local class $LC_i \in$ local database LDB_i where i=1 to n

a. calculate R_i;

b. **if**($R_i < T$)

LC_i is semantically related;

Else

LC_i is semantically heterogeneous;

// Integration of semantically related classes

ii. Call Integration Method() ;

5.2 Results and Discussion

Using the above algorithm a program in C++ is implemented to perform integration of semantically related databases. To show how semantic relevance is calculated, a Hospital scenario is considered where semantic relevance for class doctor is found.

5.2.1 Implementation of Semantic Relevance Detection Algorithm

Consider following classes Medical_Employee, Doc and Doctor of three local Databases LDB_1, LDB_2, and LDB_3 respectively.

Medical_Employee					
Emp_no	**Name**	**Sex**	**Designation**	**Place**	**Income**
D00105	Aditya Sharma	M	Surgeon	Banglore	50,000
D001010	Shirin Chandna	F	Gynecologist	Delhi	35,000
N40059	Geeta Rao	F	Nurse	Delhi	12,000
D20054	Rashmi Gupta	F	Cardiologist	Mumbai	51,000
N40159	S. Venkatavi	F	Head Nurse	Banglore	15,000

Table 5.1: Class Medical_Employee of Local Database LDB_1

Doc						
SSN	**Fname**	**Lname**	**Loc**	**Sex**	**Income**	**Phone_No**
U0012342	Rajat	Narang	Michigan	M	$8,000	3345278901
E9803421	Sadie	Hewgill	NY	F	$6,000	4267893232
U0022342	Prashant	Malviya	Chicago	M	$8,000	8322696601
E7803421	Kurson	Hwang	NY	F	$7,500	8866232311
U0012342	Peter	Fleming	Michigan	M	$4,500	3345278901

Doctor							
ID	**Name**	**Location**	**DOB**	**Sex**	**Salary**	**email**	**Contact_no**
ID5004	Rajneet Sandhu	Mumbai	26/05/83	F	450K	rsandhu@gmail.com	9944553251
ID6004	Paritosh Kalla	Pune	26/03/78	M	500K	paritosh@fms.com	9832567899
ID4600	Shreya Awasthi	Delhi	2/12/1980	F	320K	shreya1@aiims.org	8843298023

ID7632	Shikhar Oberoi	Noida	23/6/72	M	600K	soberoi@pgi.org	9423156783
ID9604	Vishal Kapoor	Gurgaon	15/08/82	M	550K	vishkap@yahoo.com	8097623411

Table 5.2: Class Doc of Local Database LDB_2

Table 5.3: Class Doctor of Local Database LDB_3

To find the semantic relevance of the class, firstly the Name of each local schema class is assigned a weight using Table 5.4 which shows relevant terms for Hospital Database. The weight is assumed according to the generalized usage of the term, for instance to create a database of Doctor in a hospital, database designers generally use the term "Doctor" or substring of Doctor such as "Doc". These terms are assigned less weight. Other term like Medical_employee is assigned more weight because it can be used for other employees in hospital, for eg. A nurse or a head nurse is also called as Medical_employee or term like Dr. is also used by PhD scholars.

Relevant Terms of a Hospital Database		
General Name	**Related Terms**	**Weight**
Doctor	Doctor	0
	Medical_employee	0.7
	General_practitioner	0.25
	Employee	0.8
	Surgeon	0.5
	Medical_practitioner	0.5
	Doc	0.25
	Dr.	0.75
Nurse	Nurse	0
	Sister	0.25
	Employee	0.6
	Medical_employee	0.7
	Medical_staff	0.6

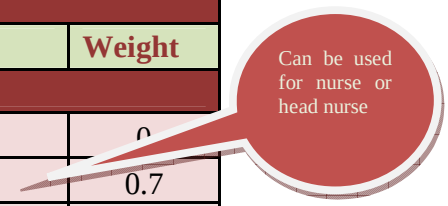

Table 5.4: Relevant Terms used in a Hospital Database

Considering class Doctor of Local Database LDB_3 as a generic template class, $\alpha = 0.4$ and Threshold T=0.4.

For class Medical_employee (Emp_no, Name, Sex, Designation, Place, Income) of Local Database LDB_1,

N_1 = Wt(Medical_employee)=0.7

Here no. of attributes that can be mapped to attributes of generic template class

ID = Emp_no

Name = Name

Sex = Sex

Location = Place

Salary = Income

Thus using equation (c) A_1 is calculated as follows

$A_1 = 1-(5/(8+6)) = 0.65$

Next is to calculate total semantic relatedness R_1 which is

$R_1 = 0.4 (0.7 + 0.65) = 0.54$

For Local Database LDB_2 Doc (SSN, Fname, Lname, Loc, Sex, Income, Phone_no)

N_2 = Wt(Doc)=0.25

Here No. of attributes that can be mapped to attributes of generic template class

ID = SSN

Name = map (Fname, Lname)

Sex = Sex

Location = Loc

Salary = map(Income)

Contact_no = Phone_no

Using map function Fname, Lname can be mapped as a single Name by concatenating the two names with space. Similarly Income in dollars can be converted to Rupees using map function.

Thus using equation (c) A_2 is calculated as follows

$A_2 = 1-(7/(8+7)) = 0.54$

Next is to calculate total semantic relatedness R_2 which is

$R_2 = 0.4 (0.25 + 0.54) = 0.316$

Since class doctor of Local database LDB_3 is considered as the template class so total semantic relevance $R_3 = 0$.

Now if we compare the total semantic relevance of all the three class with threshold value it is found that

i. $R_1 > T$
    a. Medical_Employee is Semantically Heterogeneous so it is not included in Integration Process
ii. $R_2 < T$
    a. Doc is semantically related to Doctor and is included for Integration Process
iii. R_3 is used as template class so it is included in Integration Process.

5.2.2 Programming Results

Next step is to generate global schema using Integrating operators. A Program in C++ is made to generate global schema. Using the above relevant classes Doc and Doctor, a global schema class Doctor is created.

Following is the output of the Program:

Initially it inputs the local Class Doc of Local database LDB_1 and local class doctor of Local Database LDB_2. Figure 5.1 and Figure 5.2 shows the snapshot of the program output which inputs the attributes and the class names of the local class participating in integration process.

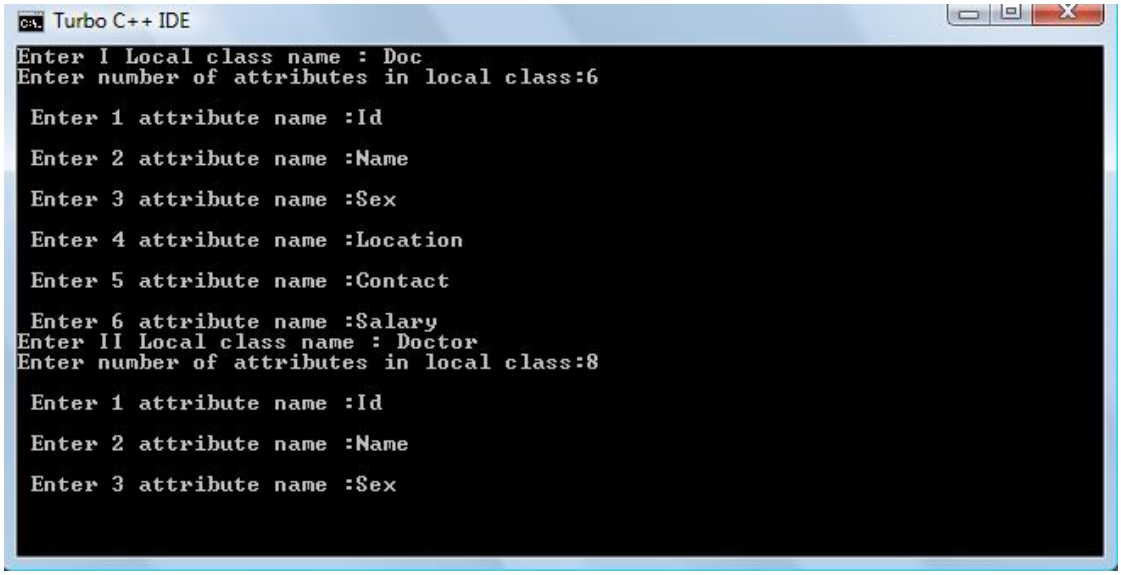

Figure 5.1: Snapshot 1 of Program Output

Figure 5.2: Snapshot 2 of Program Output

- **Import Operator:** Figure 5.3 shows the snapshot of the program if import operator is selected as the integrating operator. It imports class Doctor of local database LDB_2

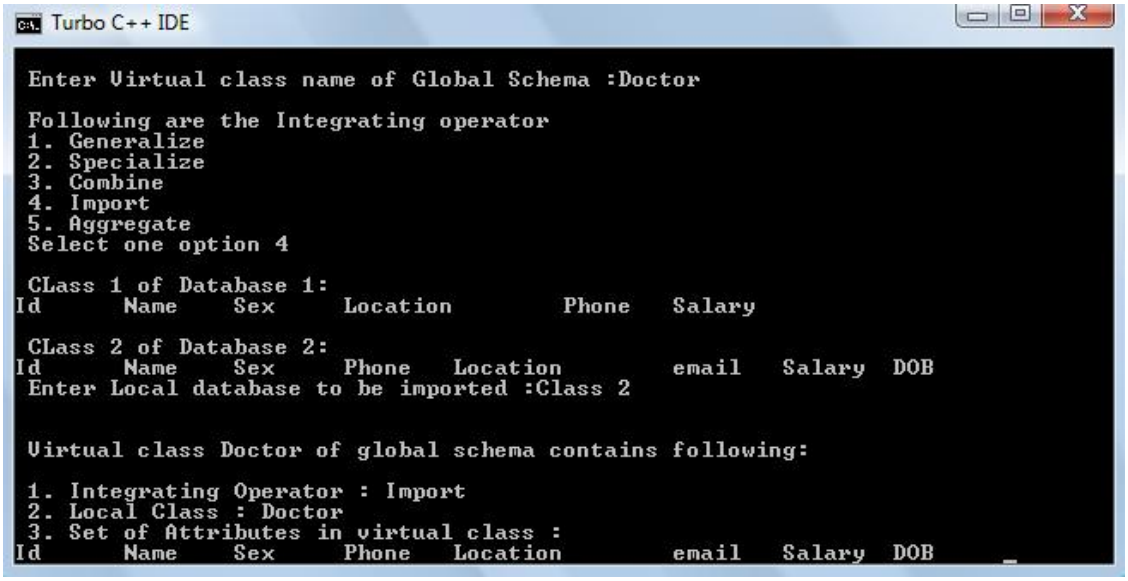

Figure 5.3: Snapshot 3 of Program Output

- **Generalize Operator:** Figure 5.4 shows the snapshot of the program if Generalize operator is selected as the integrating operator.

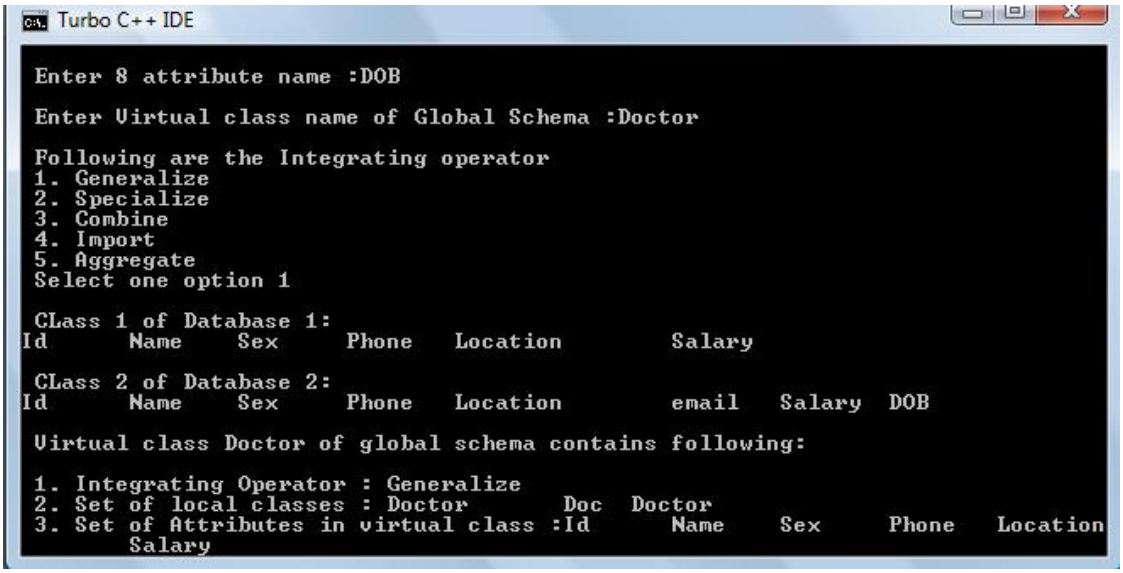

Figure 5.4: Snapshot 4 of Program Output

- **Specialize Operator:** Figure 5.5 shows the snapshot of the program if Specialize operator is selected as the integrating operator.

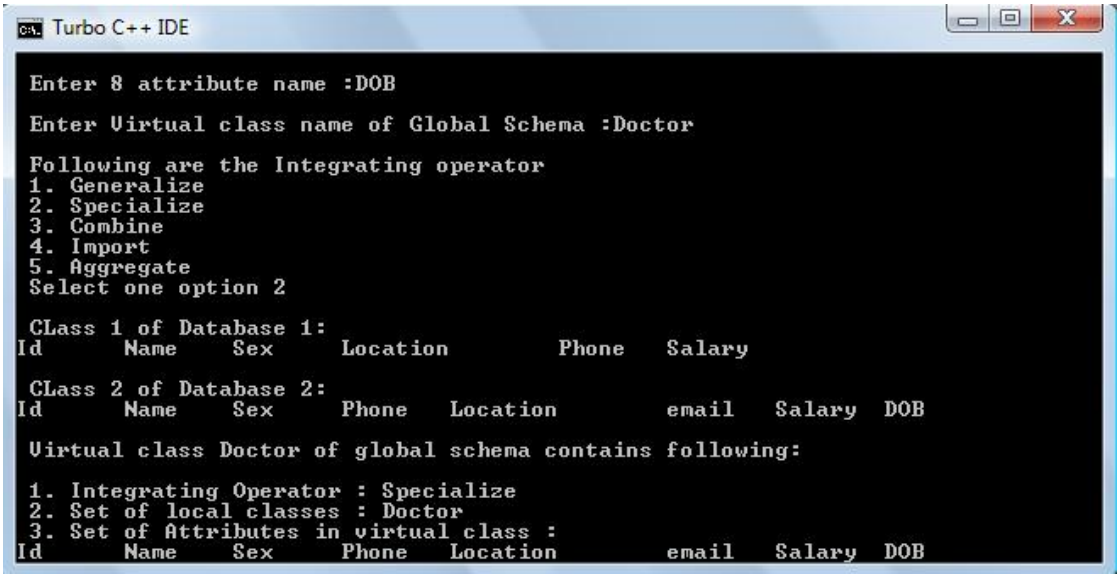

Figure 5.5: Snapshot 5 of Program Output

- **Combine Operator:** Figure 5.6 shows the snapshot of the program if Combine operator is selected as the integrating operator.

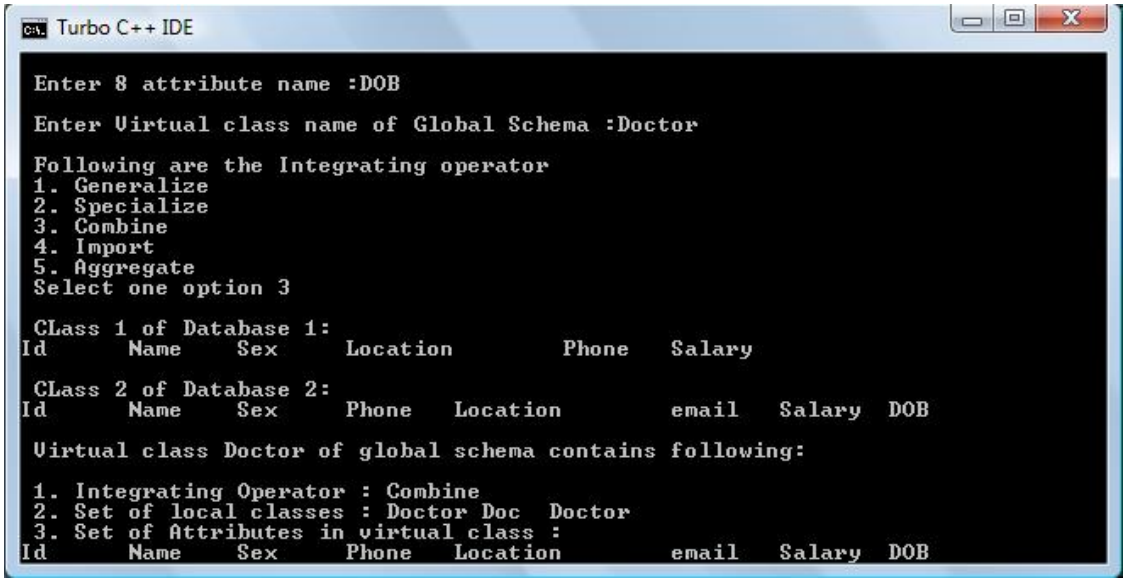

Figure 5.6: Snapshot 6 of Program Output

CHAPTER 6
CONCLUSIONS AND FUTURE SCOPE

6.1 Conclusions

In the presented work, various semantic problems of schema integration in creating a global schema for a multidatabase system are studied. To solve such problem and create global schema for federated database systems, the concept of ontology is used as a framework. Similar to the approach used for ontologies, that forms upper level ontology by defining mapping between them is used in creating a global schema from heterogeneous databases.

The semantic relevance between classes of different local databases is found using similarity concepts as used in merging ontologies. The semantically related classes are then integrated using integrating operators discussed that generates global database.

An algorithm to detect semantically related classes and database integration using integrating operators for creating global database as group of homogeneous virtual classes from heterogeneous local classes is proposed. This approach is useful in sense that global schema can be generated dynamically as per the user's perspective.

6.2 Future Scope

In the presented work, only part of the algorithm is automated, an automated tool can be designed based on the algorithm proposed to generate Global schema.

Maintaining global schema or to reflect changes in the global schema with the evolution of the local schema with time, could be used for future research.

REFERENCES

[1] Te-Wei Wang and Kenneth E. Murphy, "Semantic Heterogeneity in Multidatabase Systems: A Review and a Proposed Meta-Data Structure", Journal of Database Management, Vol. 15, No. 4, pp. 71-87,2004

[2] Natalya F. Noy, "Semantic Integration: A Survey Of Ontology-Based Approaches", SIGMOD Record, Vol. 33, No. 4, pp. 65-70, 2004

[3] R. Duwairi, "Views for interoperability in a heterogeneous object-oriented multidatabase system", Ph.D. thesis, University of Wales, Cardiff, UK, 1997.

[4] M.W. Bright, A.R. Hurson, and Simin H. Pakzad, "A Taxonomv and Current Issues in Multidatabase Svstems", IEEE, 1992.

[5] AnHai Doan and Alon Y. Halevy, "Semantic-Integration Research in the Database Community: A Brief Survey", American Association for Artificial Intelligence, pp.83-94, 2005.

[6] Miller, Yu and Nilakanta, "Integration of relational databases and record-based legacy systems for populating data warehouses", Paper presented at the 35th Annual Hawaii International Conference on System Sciences (HICSS'02), Hawaii, 2002.

[7] Batini, C., Lenzerini, M., and Navathe, S., "A Comparative Analysis of Methodologies for Database Schema Integration." ACM Computing Survey, pp. 323–364, 1986.

[8] Naiman and Ouksel, "A classification of semantic conflicts in heterogeneous database systems", Journal of Organizational Computing, pp. 167-193, 1995.

[9] Madnick, "Integrating information from global system: Dealing with the on- and off- ramps of the information superhighway", Journal of Organizational Computing, pp.69-82, 1995.

[10] W. Kim, and J. Seo, "Classifying Schematic and Data Heterogeneity in Multidatabase Systems", IEEE, 1991.

[11] Xiaomeng Su, "Semantic Enrichment for Ontology Mapping", Ph.D. thesis, Norwegian University of Science and Technology, 2004.

[12] Ming Mao, “Ontology Mapping: Towards Semantic Interoperability in Distributed and Heterogeneous Environments”, PhD. thesis, University of Pittsburgh, 2008.

[13] Wache, H., et al., “Ontology-Based Integration of Information - A Survey of Existing Approaches.” IJCAI-2001 Workshop on Ontologies and Information Sharing, Seattle, USA, pp. 108-117, 2001

[14] Pottinger, R. A., and Bernstein, P. A., “Merging Models Based on Given Correspondences”, International Conference on Very Large Databases (VLDB), San Francisco, 2003.

[15] Parent, C., and Spaccapietra, S, “Issues and Approaches of Database Integration”, Communications of the ACM, pp. 166–178, 1998.

[16] Calvanese D., et al, “Knowledge representation approach to information integration”. Proceedings of the Workshop on AI and Information Integration (AAAI’98), pp. 58-65, 1999.

[17] Castano S., Antonellis V., and Capitani V., “Global viewing of heterogeneous data sources”. IEEE Transactions on Knowledge and Data Engineering, Vol. 13, No. 2, pp. 277-297, 2001.

[18] Rahm E and Bemstien P., “A survey of approaches to automatic schema matching”. The VLBD Journal, Vol. 10, pp. 334-350, 2001.

[19] Kashyap V. and Sheth A., “Semantic and schematic similarities between database objects: a contextbased approach”. The VLDB Joumal, 5: pp. 276-304, 1996.

[20] Chen Y., “Integrating heterogeneous object oriented schemas”, Journal of Information Science and Engineering, pp. 555-591, 2000.

[21] Li W. S. and Clifton C. “Semantic integration in heterogeneous databases using neural networks”. Proceeding of the VLDB Conference, Santiago, Chile, pp. 1 - 12, 1994.

[22] Reddy, et al., “A Methodology for Integration of Heterogeneous Databases”, IEEE Transactions on Knowledge and Data Engineering. Vol. 6, 1994.

[23] F. Hakimpour, A. Geppert, “Resolving Semantic Heterogeneity in Schema Integration: an Ontology Based Approach”, ACM, pp. 297-308, 2001.

[24] S. Bergamaschi, S. Castano, S. De Capitani di Vimercati, S. Montanari, and M. Vincini., “An intelligent approach to information integration.” Formal Ontology in Information Systems, pp. 253-267, 1998

[25] Cheng Hian Goh, "Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Systems”, Ph.D. thesis, Massachusetts Institute of Technology,1997.

[26] Dey D., Sarkar S. and De P., “A distance-based approach to entity reconciliation problem in heterogeneous databases”. IEEE Transactions on Knowledge and Data Engineering, Vol 14, No. 3, pp. 567-496, 2002.

[27] M. G. Ali, “Object-Oriented Approach for Integration of Heterogeneous Databases in a Multidatabase System and Local Schemas Modifications Propagation”, International Journal of Computer Science and Information Security Vol. 6, pp. 55-60, 2009.

[28] Ching-Ming Chao, “Schema Integration between Object-Oriented Databases” Tomkang Journal of Science and Engineering, 2001.

[29] Soon M. Chung and Pyeong S. Mah, “Schema Integration for Multidatabase Using the Unified Relational and Object-Oriented Model”, ACM, 1995.

[30] Kim W. and Sea J., “Classifying schematic and data heterogeneity in multidatabase systems”, IEEE Computer, 24(12), pages 12-18, 1991

[31] Adam Farquhar, Richard Fikes, and James Rice, “The ontolingua server: a tool for collaborative ontology construction”, International Journal of Human-Computer Studies, pp. 707-727, 1997.

[32] P.R.S. Visser, et al., “Resolving ontological heterogeneity in the kraft project”, In 10th International Conference and Workshop on Database and Expert Systems Applications DEXA ’99, University of Florence, Italy, 1999.

LIST OF PUBLICATIONS

Papers Communicated

Sugandha Lohar and Shalini Batra, "Ontological Approach to Integrate Semantically Related Databases", communicated to 'CiiT International Journal of Data Mining and Knowledge Engineering', June 2010.

Sugandha Lohar and Shalini Batra, "Semantic Integration of Heterogeneous Databases in Multidatabase System", communicated to 'Journal of Information Technology', June 2010.