

# SemQL: A Semantic Query Language for Multidatabase Systems

Jeong-Oog Lee and Doo-Kwon Baik

Software System Lab., Dept. of Computer Science & Engineering, Korea University,  
1, 5-ka, Anam-dong, Sungbuk-gu, 136-701, SEOUL, KOREA  
Phone: +82-2-925-3706

E-mail: {ljo, baik}@swsys2.korea.ac.kr

## ABSTRACT

An essential prerequisite to achieving interoperability in multidatabase systems is to be able to identify semantically equivalent or related data items in component databases. Another problem in multidatabase systems is allowing users to handle information from different databases that refer to the same real-world entity. In this paper, we provide semantic networks so that multidatabase systems can detect and resolve semantic heterogeneities among component databases. And we provide a semantic query language, SemQL, to capture the concepts about what users want. It enables users to issue queries to a large number of autonomous databases without prior knowledge of their schemas.

## Keywords

Semantic query language, multidatabase, semantic network, WordNet

## 1. INTRODUCTION

In conventional database systems, information was created, stored, and used independently to meet specific requirements. As the amount of information grew rapidly, it became harder to manage it efficiently in one database systems. Also, in large knowledge bases such as Web, many questions often can be answered using integrated information than using a single source. Therefore the need for integrating disjointed databases became increasing. A multidatabase system provides integrated access to heterogeneous, autonomous component databases in a distributed system. An essential prerequisite to achieving interoperability in multidatabase systems is to be able to identify semantically equivalent or related data items in component databases.

Another problem in multidatabase systems is allowing users to handle information from different databases that refer to the same real-world entity. That is, multidatabase systems let users focus on

specifying what they want, rather than thinking about how to obtain the answers, which frees them from the tremendous tasks of finding the relevant component databases, interacting with each database using a particular interface, and integrating results from multiple databases.

While there is a significant amount of researches discussing schema differences, work on semantic issues in the multidatabase is scarce [1]. Because only schema considerations do not suffice to detect semantic heterogeneity, additional knowledge has to be considered in order to gain semantic knowledge. We use linguistic knowledge in WordNet for integrating information. WordNet is an on-line lexical dictionary and organized by semantic relations such as synonymy, antonymy, hyponymy, and meronymy [6]. The noun portion of WordNet is designed around the concept of synset which is a set of closely related synonyms representing a word meaning.

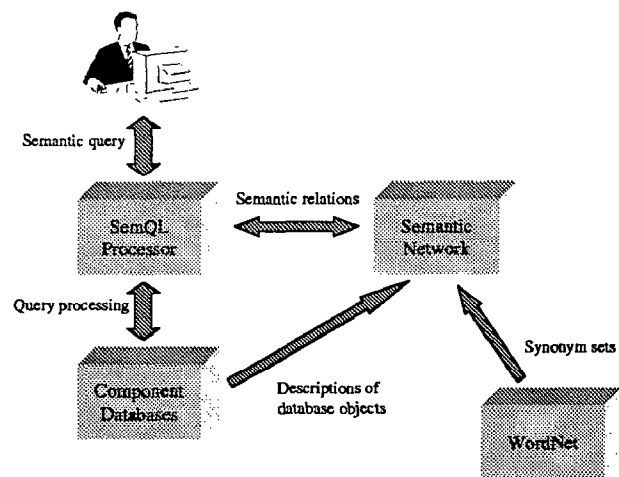


Figure 1. An outline of our approach for information integration

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. CIKM '99 11/99 Kansas City, MO, USA © 1999 ACM 1-58113-075-9/99/0010...\$5.00

Figure 1 shows an outline of our approach. Using the synsets in WordNet and the descriptions of database objects, we construct a semantic network which provides semantic relations among database objects. With this semantic network, we can detect and resolve semantic heterogeneities among component databases.

Also, we provide a semantic query language, SemQL, to capture the concepts about what users want, which enables users to issue queries to a large number of autonomous databases without prior knowledge of their schemas.

The rest of this paper is organized as follows. In section 2, we first review related work in multidatabase systems. In section 3, we address the problem of semantic integration. We classify the types of schema conflicts and discuss each type of the conflicts. After we introduce WordNet briefly, we present the process of constructing a semantic network. In section 4, we discuss the need of semantic query language and introduce SemQL, and describe the procedure of semantic query processing using the semantic network with some example scenarios. Finally, section 5 offers our conclusions.

## 2. RELATED WORK

Early researches on semantic heterogeneities in multidatabase systems focused on procedures to merge individual component database schemas into a single global schema. A global schema multidatabase supports a single, integrated global view to the user and provides simple and effective paradigm. However, creating and maintaining the global schema is difficult. The amount of knowledge required about local schemas and how to identify and resolve heterogeneity among the local schemas is a major problem with this approach. Also, a small change to a local schema may require huge changes to the global schema. Therefore, these static approaches are not adequate for large scale interoperable database systems and dynamic environments where databases change over time, and where new databases can be added autonomously, even though they might provide satisfactory support for small or static systems.

Multidatabase languages are an attempt to resolve some of the problems associated with a global schema. With these systems, no global schema is maintained. This approach puts the integration responsibility on users by providing functionality beyond standard SQL to allow users to specify integration information as part of the query. Multidatabase language approach eliminates problems of global schema creation and maintenance, but presents a more complex global interface to the user.

In several researches [4][9], new approaches have been developed for integrating of information using new technological developments such as agent technology, domain ontologies, intelligent mediator, and high-level query languages, in dynamic and open environments. The InfoSleuth project at MCC [9] is broadening the focus of database research to meet the challenge presented by the World Wide Web. It extends the capabilities of the Carnot technologies into dynamically changing environments, where the identities of the resources to be used may be unknown at the time the application is developed. InfoSleuth, therefore, rigidly observe the autonomy of its resources, and does not depend on their presence. Information-gathering tasks are thus defined generically, and their results are sensitive to the available resources. In the SIMS project [4], each SIMS agent contains a detailed model of its domain of expertise and models of the information sources that are available to it. Given an information request, an agent selects an appropriate set of information sources, generates a plan to retrieve and process the data, uses knowledge about the information sources to reformulate the plan for efficiency, and then executes it. These approaches were designed to support flexibility and openness. A common assumption of

these dynamic approaches is that the users know pre-existing knowledge for integrating information, which might be a burden to the users.

Recent advances in online dictionaries and thesauruses make it possible to apply linguistic theory in an automated fashion, which enable users to perform integrating information more comfortably. In [2], the Summary Schemas Model (SSM) is proposed as an extension to multidatabase systems to aid in semantic identification. The SSM uses a global data structure to abstract the information available in a multidatabase system. The abstracted form allows users to use their own terms (imprecise queries) when accessing data rather than being forced to use system-specified terms. The system uses the global data structure to match the user's terms to the semantically closest available system terms. However, this approach tends to centralize the search within a single logical index thereby introducing performance limitations for large networks.

As linguistic theories evolved in recent decades, linguists became increasingly explicit about the information a lexicon must contain in order for the phonological, syntactic, and lexical components to work together in the everyday production and comprehension of linguistic messages [6]. WordNet is an electronic lexical system developed at Princeton University. Several approaches [3][5] uses WordNet as knowledge about the semantic contents of images to improve retrieval effectiveness. In particular, [3] uses WordNet for query and database expansion.

## 3. SEMANTIC NETWORKS

A multidatabase system can provide a uniform interface to a multitude of component databases. Consider the knowledge a multidatabase system would need to answer the following question:

"Find those professor whose salary is over \$50,000."

To answer this question, the system must have knowledge of several kinds:

- It must know where to find the relevant information on the component databases (access knowledge).
- It must know which entities, attributes, or values in the component databases meet the semantics in the query (semantic knowledge).

To acquire this knowledge, we must cope with semantic conflicts. We develop semantic networks, which specify the relations among entities, attributes, and value domains in component databases. The basic idea is to use synsets in WordNet to provide mapping mechanism. Once semantic network is constructed, we can detect and resolve semantic heterogeneity based on it.

### 3.1 Classification of Semantic Heterogeneity

A multidatabase system provides integrated access to heterogeneous, autonomous component databases in a distributed system. In order to gain integrated access to the multidatabase, semantic heterogeneities have to be detected and resolved. Semantic heterogeneities include differences in the way the real world is modeled in the databases, particularly in the schemas of the databases [7].

Since a database is defined by its schema and data, semantic heterogeneities can be classified into schema conflict and data conflict [10]. Schema conflicts mainly result from the use of

different structures for the same information and the use of different names for the same structures. Data conflicts are due to inconsistent data in the absence of schema conflicts. Figure 2 shows an example to illustrate semantic heterogeneities.

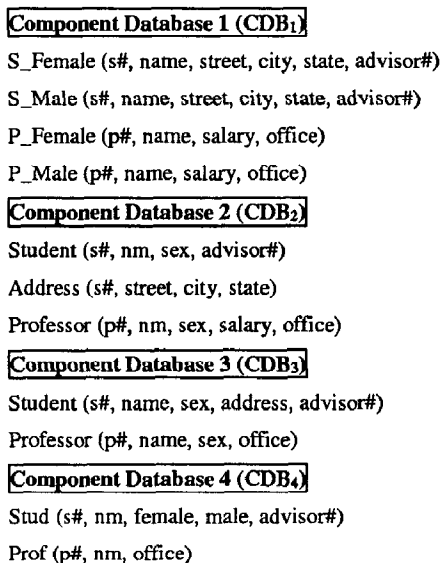


Figure 2. An example database schemas

As our focus is only on the schema conflicts, we assume that data conflicts such as different representations for the same data are already conformed. Focusing on schema conflicts, we discuss the types of conflicts which are considered in this paper as follows.

- **Entity versus entity structure conflicts (EESC)**

These conflicts occur when CDBs use different numbers of entities to represent the same information. For example, CDB<sub>1</sub> uses two entities, **S\_Female** and **S\_Male**, for general information on student, while CDB<sub>3</sub> uses only one entity, **Student**.

- **Entity versus attribute structure conflicts (EASC)**

This type of conflicts occurs if an attribute of some CDBs is represented as an entity in others. In the example database schemas, CDB<sub>3</sub> uses an attribute, **address**, in the entity **Student** to represent the student's address, while CDB<sub>2</sub> represents the same information in the entity **Address**.

- **Entity versus value structure conflicts (EVSC)**

These conflicts occur when the attribute values in some CDBs are semantically related to the entities in other CDBs. For example, CDB<sub>1</sub> uses two entities, **S\_Female** and **S\_Male**, for female and male students, respectively, while the same information for sex is represented as values of an attribute **sex** in CDB<sub>3</sub>.

- **Attribute versus attribute structure conflicts (AASC)**

These conflicts occur when CDBs use different numbers of attributes to represent the same information. For example, CDB<sub>1</sub> uses three attributes, **street**, **city** and **state**, for information on address, while CDB<sub>3</sub> uses one attribute, **address**.

- **Attribute versus value structure conflicts (AVSC)**

These conflicts occur when the attribute values in some CDBs are semantically related to the attributes in others. In the example

database schemas, CDB<sub>4</sub> uses two attributes, **female** and **male**, for information on sex, while the same information is represented as values of an attribute **sex** in CDB<sub>3</sub>.

- **Entity versus entity name conflicts (EENC)**

These conflicts arise due to different names assigned to entities in different CDBs. For example, the entity for student is called **Student** in CDB<sub>3</sub> and **Stud** in CDB<sub>4</sub>.

- **Attribute versus attribute name conflicts (AANC)**

Attribute name conflicts are similar to the entity name conflicts. An example of this conflict type occurs in the example database schemas where the attribute for student's name is called **name** in CDB<sub>3</sub> and **nm** in CDB<sub>4</sub>.

### 3.2 WordNet as an on-line Lexical Dictionary

Since the word "word" is commonly used to refer both to the utterance and to its associate concept, in order to reduce ambiguity, "word form" will be used to refer to the physical utterance and "word meaning" to refer to the lexicalized concept that a form can be used to express. Then the starting point for the lexical semantics can be said to be the mapping between forms and meanings [6].

WordNet is the product of a research project at Princeton University which has attempted to model the lexical knowledge of a native speaker of English. The system has power of both an on-line thesaurus and an on-line dictionary, and much more. Information in WordNet is organized around logical groupings called synsets. Each synset consists of a list of synonymous word forms and semantic pointers that describe relationships between the current synset and other synsets. The semantic pointers can be of a number of different types including, Synonymy, Antonymy, Hyponymy and Meronymy,

Mappings between forms and meanings are many:many-some forms have several different meanings, and some meanings can be expressed by several different forms. Two difficult problems of information integration, polysemy and synonymy, can be viewed as complementary aspects of this mapping. That is to say, polysemy and synonymy are problems that arise in the course of gaining access to information in multidatabase systems.

The case of 'customers' and 'clients' is an example of synonymy, while the word form 'client' has many different meanings, which shows an example of polysemy. The following figure 3 shows several different meanings for 'client' in WordNet.

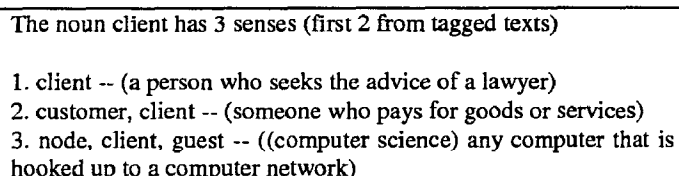


Figure 3. Different meanings for 'client' in WordNet.

### 3.3 Constructing Semantic Networks

As semantically alike entities and attributes might be abbreviated differently in different databases, we use not the names of entities and attributes but the words in WordNet, which describe the concepts of the entities and attributes. A semantic network

provides mappings between words in WordNet and words provided by the local DBAs using synsets in WordNet.

The overall process of constructing a semantic network as follows.

**STEP 1.** At each CDB, the local DBAs must make descriptions of entities and attributes. As it is not possible for any system to capture semantics without human interaction, creating a semantic network requires some initial input from CDB administrators. For simplicity of our approach, the local DBAs use only single noun, compound noun, or noun phrase in making descriptions.

**STEP 2.** The descriptions are decomposed into unit nouns. A unit noun means single noun, compound noun, or noun phrase that must be found in WordNet. For example, as compound noun, 'identification number', can be found in WordNet, it is treated as a unit noun in our approach. Figure 4 describes an algorithm for decomposing the descriptions into unit nouns.

```

For each  $D_i$ 
If  $|D_i| = 1$  Then  $U_i = \{<n_1>\} = \{<w_1>\}$ 
Else If  $|D_i| = 2$  Then
    Input  $w_1 \cdot w_2$  into WordNet
    If  $w_1 \cdot w_2$  found in WordNet Then  $U_i = \{<n_1>\} = \{<w_1 \cdot w_2>\}$ 
    Else  $U_i = \{<n_1>, <n_2>\} = \{<w_1>, <w_2>\}$ 
Else
    Decompose  $D_i$  into  $w_1 \cdot w_2$  and  $w_3$ 
    If  $w_1 \cdot w_2$  found in WordNet Then  $U_i = \{<n_1>, <n_2>\} = \{<w_1 \cdot w_2>, <w_3>\}$ 
    Else
        Decompose  $D_i$  into  $w_1$  and  $w_2 \cdot w_3$ 
        If  $w_2 \cdot w_3$  found in WordNet Then  $U_i = \{<n_1>, <n_2>\} = \{<w_1>, <w_2 \cdot w_3>\}$ 

```

$D_i = \{w_1, w_2, \dots\}$  : The set of words in  $i$ 'th description  
 $|D_i|$  : The number of words in  $D_i$   
 $U_i$  : The set of unit nouns of  $D_i$   
 $w_i \cdot w_j$  : Concatenation of  $w_i$  and  $w_j$

**Figure 4.** An algorithm for decomposing descriptions into unit nouns

In the algorithm, we assume that the number of words in a description has at most three. In the case of a description having more than three words, the algorithm might be expanded easily. Examination of a few examples shows how the algorithm works.

**Example 1:** Let us suppose that the description for  $CDB_3$ .Student.name is *student's name*. As the *student* in the description is the name of the entity **Student**, the number of words in the description,  $|D_{name}|$  is 1. As we can find the word *name* in WordNet, the set of unit nouns of  $D_{name}$ ,  $U_{name}$ , is  $\{<name>\}$ .

**Example 2:** Let us suppose that the description for  $CDB_3$ .Student.s# is *student's identification number*. In a similar way as Example 1, the *student* in description is the name of the entity **Student**. Therefore, the number of words in the description,  $|D_{s\#}|$  is 2. And as the words *identification number* can be found in WordNet, the set of unit nouns of  $D_{s\#}$ ,  $U_{s\#}$ , is  $\{<identification number>\}$ . That is, the number of unit nouns in  $U_{s\#}$  is 1.

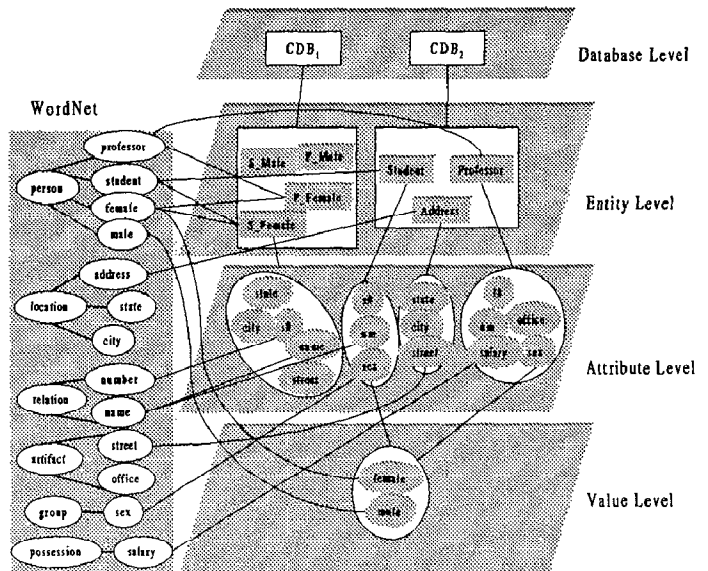
**Example 3:** Let us suppose that the description for  $CDB_1$ .S\_Female is *female student*. In this case, the compound word, *female student*, can not be found in WordNet. Therefore, the set of unit nouns  $U_{S\_Female}$  is  $\{<female>, <student>\}$  and  $|U_{S\_Female}|$  is 2.

**Example 4:** Put the case that a description for an attribute A is *office phone number*. When *office phone number* is left branching ( $\langle office \text{ phone} \rangle \langle number \rangle$ ), the  $\langle office \text{ phone} \rangle$  can not be found in WordNet, while it is right branching ( $\langle office \rangle \langle phone \text{ number} \rangle$ ), both  $\langle office \rangle$  and  $\langle phone \text{ number} \rangle$  can be found in WordNet. Therefore, the set of unit nouns  $U_A$  is  $\{<office>, <phone \text{ number}>\}$ .

The result for this decomposing process is a description table (DT). A description table consists of Rec#, Object\_Type, Object\_Name, Descriptions, and Unit\_nouns.

**STEP 3.** After making a description table, the CDB administrators must cope with synonymy and polysemy. The noun portion of WordNet is designed around the concept of synset which is a set of closely related synonyms representing a word meaning. To identify unit nouns related by synonymy automatically, our approach uses synsets in WordNet. However, to acquire correct meaning of a unit noun, the CDB administrators must cope with its polysemy manually. For example, when the local DBAs input a unit noun 'client' into WordNet, and the result is as figure 3, he/she must choose one among many different meanings. This is more or less a burden to the local DBAs, but the rest of the creation and maintenance is automatic.

**STEP 4.** Given the sets of unit nouns of the two component databases ( $CDB_1$  and  $CDB_2$ ),  $U_{CDB_1}$  and  $U_{CDB_2}$ , each unit noun in  $U_{CDB_1}$  is compared with unit nouns in  $U_{CDB_2}$ , respectively, using synsets in WordNet. The output for this comparison process is a semantic network. Figure 5 shows a partial semantic network for schemas of figure 2.



**Figure 5.** A partial semantic network for schemas of figure 2.

## 4. SEMANTIC QUERY PROCESSING

Seen from a semantic perspective, the process of database design proceeds from the real world to the data. The designer develops his own conceptualization of the real world and turns his conceptualization into a database design. This has led designers to develop different, often incompatible, schemas for the same information. Therefore, users needing to combine information from several databases are faced with the problem of locating and integrating relevant information.

One possible solution to the user's problem is simply allowing the user to access the relevant databases. But, often a user may not know the relevant databases exist. Even though the user knows about the location of the relevant databases, the user probably does not know all of the schemas of the relevant databases. Therefore, this solution requires the user to learn all the schemas, which is an unacceptable burden to the users.

Another type of approach is a global schema multidatabase, which supports a single, integrated global view to the user. However, creating and maintaining the global schema is difficult. Also, users need to learn the global schema.

More efficient and effective approach is allowing users to issue queries to a large number of autonomous databases with his/her own terms. It frees users from learning schema. We propose SemQL as a semantic query language for users to issue queries using not schema but concepts that the users know.

### 4.1 SemQL as a Semantic Query Language

The SemQL is similar to SQL except that it has no FROM clause. The basic form of the SemQL is formed of the two clauses SELECT and WHERE and has the following form:

```
SELECT <concept list>
WHERE <condition>
```

Here <concept list> is a list of concepts whose values are to be retrieved by the query. The <condition> is a conditional search expression that identifies the tuples to be retrieved by the query. Figure 6 shows only part of syntax for SemQL.

```
<concept list> ::= <concept-expr> | <concept-expr>, <concept list>
<concept-expr> ::= <entity-concept-name>.<attribute-concept-name>
<condition> ::= <condition> AND <comparison> |
               <condition> OR <comparison> |
               NOT <condition> |
               <comparison>
<comparison> ::= <value> <comp-op> <value>
<comp-op> ::= = | != | > | < | >= | <=
<val> ::= <concept-expr> | <constant>
```

Figure 6. A part of syntax for SemQL

The SemQL clauses specify not the entity or attribute names in component database schemas but concept names about what users want. For example, suppose a user wants to find those professor

whose salary is over \$50,000. We assume that the user is familiar with SQL, but knows neither of the component database schemas. Then the user might issue a query in SemQL using concepts that he/she knows;

```
SELECT professor.name
WHERE professor.salary > $50,000
```

### 4.2 Semantic Query Processing Procedure

The overall procedure of semantic query processing is shown in figure 7. The SemQL Processor consists of Query Parser, Resource Finder, Mapping Generator, Sub-query Generator, Query Distributor and Integrator.

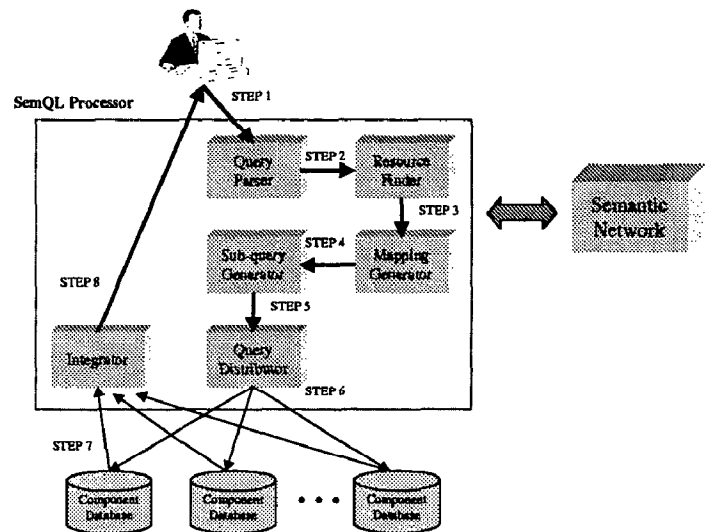


Figure 7. The overall procedure of semantic query processing

**STEP 1.** The users issue a semantic query with his/her own concepts to retrieve equivalent or related data items.

**STEP 2.** The Query Parser parses the query and extracts entity, attribute and value concepts from the query.

**STEP 3.** The Resource Finder identifies the relevant component databases where the concepts exist using the semantic network.

**STEP 4.** The Mapping Generator generates the mappings between concepts in original query and representations in component databases.

**STEP 5.** The Sub-query Generator re-formulates the original query into multiple sub-queries for each component database schema according to the mappings. In this step, looking up the semantic network, Sub-query Generator adds FROM clause to the sub-query.

**STEP 6.** The Query Distributor submits the sub-queries to the component databases.

**STEP 7.** The component databases receive the sub-query and execute it. And then return the result tuples to the SemQL Processor.

**STEP 8.** The Integrator merges the intermediate results from various component databases and presents the integrated results to the users.

In this section, we introduce some example query scenarios to demonstrate the procedure of semantic query process of our approach. Through the example scenarios, we will also explain how the semantic conflicts can be detected and resolved using semantic networks. The following query that retrieves those professor whose salary is over \$50,000 is provided as the first example query. We assume that the user who issues the query only knows the concepts about what he/she want. That is, the user does not know the detailed schema structure for each component database.

QUERY: Find those professor whose salary is over \$50,000.

The user can issue the query as follows to the SemQL Processor (STEP 1).

```
SELECT professor.name
WHERE professor.salary > $50,000
```

The Query Parser parses the query and extracts concepts from the query (STEP 2). The extracted concept list is {professor, name, salary}. And then, looking up the semantic network, the Resource Finder finds the relevant component databases where all the concepts, 'professor', 'name', 'salary', exist (STEP 3). CDB<sub>1</sub> and CDB<sub>2</sub> may be identified as the relevant component databases.

The Mapping Generator identifies the mappings between concepts in original query and representations in CDB<sub>1</sub> and CDB<sub>2</sub>, respectively (STEP 4). Figure 8 shows the results for mapping procedure. We use Semantic Diagram to represent the mappings. Appendix contains an explanation of the Semantic Diagram.

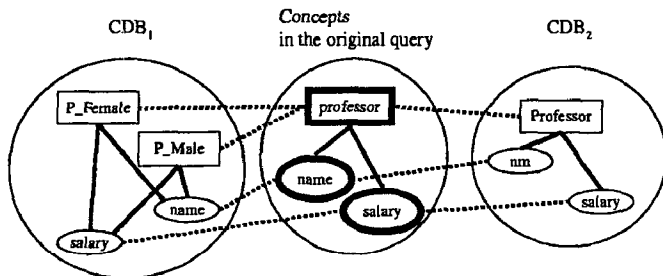


Figure 8. The mappings between concepts and database objects for the first example scenario

Depending on the mappings, we can detect two types of semantic conflicts, EESC and AANC. CDB<sub>1</sub> uses two entities, P\_Female and P\_Male, for professor, while CDB<sub>2</sub> uses only one entity, Professor. This is the case for the type of EESC. The type of AANC occurs where the attribute for professor's name is called name in CDB<sub>1</sub> and nm in CDB<sub>2</sub>.

After mapping procedure, the Sub-query Generator generates sub-query for CDB<sub>1</sub> (STEP 5);

```
SELECT name
FROM P_Male
WHERE salary > $50,000
UNION
```

```
SELECT name
FROM P_Female
WHERE salary > $50,000
```

and for CDB<sub>2</sub>;

```
SELECT nm
FROM Professor
WHERE salary > $50,000
```

And then the Query Distributor submits the two sub-queries to CDB<sub>1</sub> and CDB<sub>2</sub>, respectively (STEP 6). The two component databases, CDB<sub>1</sub> and CDB<sub>2</sub>, return the result tuples of the sub-queries to the SemQL Processor (STEP 7). The Integrator merges the results from the two CDBs and presents the integrated results to the users (STEP 8).

The second example query is to find those female students who live in Seoul.

QUERY: Find those female students who live in Seoul.

The query can be posed as follows (STEP 1):

```
SELECT student.name
WHERE student.sex = 'female' AND student.city = 'Seoul'
```

The Query Parser parses the query and extracts concepts from the query – {student, name, sex, female, city} (STEP 2). And then, The Resource Finder identifies the relevant component databases, CDB<sub>1</sub>, CDB<sub>2</sub> and CDB<sub>3</sub>, which possess all the concepts (STEP 3). The mappings between concepts in original query and representations in the relevant CDBs are generated (STEP 4) and shown in figure 9.

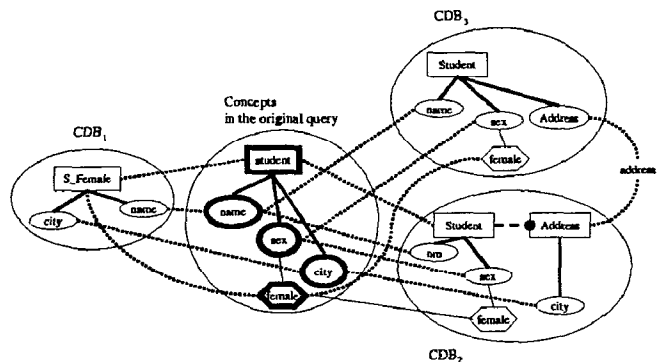


Figure 9. The mappings between concepts and database objects for the second example scenario

In this example scenario, CDB<sub>1</sub> uses three attributes, street, city, and state, for information on address, while CDB<sub>3</sub> uses one attribute, address. This is the case of AASC. The type of EASC exists where CDB<sub>3</sub> uses an attribute, address, in the Student entity to represent the student's address, and CDB<sub>2</sub> represents the same information in the Address entity. As CDB<sub>1</sub> uses the S\_Female entity for female students, and the same information for

sex is represented as values of the **sex** attribute in CDB<sub>3</sub>, the type of EVSC also occurs in the example scenario.

Now, the Sub-query Generator re-formulates the original query into three sub-queries for CDB<sub>1</sub>, CDB<sub>2</sub> and CDB<sub>3</sub> according to the mappings (STEP 5). Thus, the sub-query for CDB<sub>1</sub> might be;

```
SELECT name
FROM S_Female
WHERE city = 'Seoul'
```

The original query might be re-formulated for CDB<sub>2</sub>;

```
SELECT Student.nm
FROM Student, Address
WHERE Student.s# = Address.s# AND Address.city = 'Seoul'
```

and for CDB<sub>3</sub>;

```
SELECT name
FROM Student
WHERE sex = 'female' AND Address LIKE '%Seoul%'
```

After the Sub-query Generator re-formulates the original query into sub-queries, the Query Distributor sends them to CDB<sub>1</sub>, CDB<sub>2</sub> and CDB<sub>3</sub>, respectively (STEP 6). The three component databases, CDB<sub>1</sub>, CDB<sub>2</sub> and CDB<sub>3</sub>, return the result tuples of the sub-queries to the SemQL Processor (STEP 7). Finally, the Integrator merges the results from the three CDBs and presents the integrated results to the users (STEP 8).

In the third example scenario, we assume that only two component databases, CDB<sub>3</sub> and CDB<sub>4</sub> exist in the multidatabase system. The query is as follows.

QUERY: Find all of the female students.

The query in SemQL might be;

```
SELECT student.name
WHERE sex='female'
```

Then, SemQL Processor identifies CDB<sub>3</sub> and CDB<sub>4</sub> as the relevant component databases. The mappings between concepts and objects are shown in figure 10.

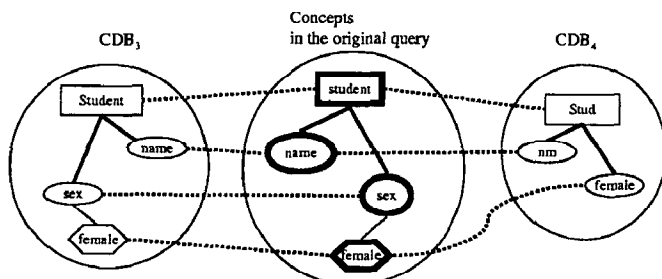


Figure 10. The mappings between concepts and database objects for the second example scenario

In CDB<sub>3</sub>, the information on sex is represented as values of the attribute **sex**, while CDB<sub>4</sub> uses two attributes, **female** and **male**, for the same information. This is the case of AVSC. CDB<sub>3</sub> uses **Student** entity for students and **name** attribute for student's name, while the entity for students is called **Stud** and the attribute for student's name is called **nm** in CDB<sub>4</sub>. It shows the types of EENC and AANC.

According to the mappings, the sub-query for CDB<sub>3</sub> might be generated as follows.

```
SELECT name
FROM Student
WHERE sex = 'female'
```

and the original query might be re-formulated for CDB<sub>4</sub>;

```
SELECT nm
FROM Stud
WHERE female = 'YES'
```

## 5. CONCLUSIONS

In order to integrate disjointed information sources into a multidatabase system, we must consider two views of the integration problem. First, in the system's view, multidatabase systems can identify semantically equivalent or related data items in component databases. Different conceptualizations of the real world have led designers to develop different, often incompatible, schemas for the same information. Second, in the user's view, multidatabase systems can allow users to handle information from different databases more easily. That is, they let users focus on specifying what they want, rather than finding the relevant sources, interacting with each source using a particular interface, and integrating results from various sources.

In this paper, focusing on the two views, we have suggested a method for integrating information in distributed component databases. As only structural similarity is not adequate for capturing semantics, we use synonym sets in WordNet as background knowledge of our approach. From the word meanings between synonym sets in WordNet and descriptions of database objects, semantic networks can be created. With the semantic relations presented in semantic networks, semantic heterogeneities among component databases can be detected and resolved. Also, we provide a semantic query language, SemQL, to capture the concepts about what users want. SemQL allows users to issue queries with his/her own concepts, which frees them from learning various component database schemas. It gives users more convenient and efficient access mechanism.

## 6. REFERENCES

- [1] Amit Sheth, Vipul Kashyap, "So Far(Schematically) yet So Near(Semantically)", Proceedings, IFIP WG 2.6 Conference on Semantics of Interoperable Database Systems(Data Semantics 5), 1993, pp.283-312.
- [2] A. R. Hurson, M. W. Bright, "Global Information Access for Microcomputers", Journal of Mini and Micro Computer Applications, 1991.
- [3] Aslandogan, Y. A., C. Thier, C. T. Yu, J. Zou and N. Rishe, "Using semantic contents and WordNet in image retrieval",

Proceedings of the 20<sup>th</sup> Annual ACM SIGIR Conference on Research and Development in Information Retrieval, 1997.

- [4] Craig A. Knoblock, Yigal Arens, and Chun-Nan Hsu, "Cooperating Agents for Information Retrieval", Proceedings of the second International Conference on Cooperative Information Systems, 1994.
- [5] Eugene J. G. And Neil C. R., "Natural-Language Retrieval of Images Based on Descriptive Captions", In ACM Transactions on Information Systems 14(3), July, 1996.
- [6] G. A. Miller, R. Beckwith, C. Fellbaum, D.Gross, and K. Miller, "Five Papers on WordNet", CSL Reort 43, Cognitive Systems Laboratory, Princeton Univ., 1990.
- [7] M. Garcia-Solaco, F. Saltor, and M. Castellanos, "Semantic Heterogeneity in Multidatabase Systems", Object-Oriented Multidatabase Systems: A Solution for Advanced Applications, Edited by Orman A. Bukhres, Ahmed K. Elmagarmid, Prentice Hall Inc., 1996, pp.129-202.
- [8] M. Lauer, M. Dras, "A Probabilistic Model of Compound Nouns", Proceedings of the 7<sup>th</sup> Australian Joint Conference on AI, 1994.
- [9] R. Bayardo, W. Bohrer, et al, "InfoSleuth: agent-based semantic integration of information in open and dynamic environments", ACM SIGMOD Record, Vol. 26, No. 2, June, 1997.
- [10]W. Kim, J. Seo, "Classifying Schematic and Data Heterogeneity in Multidatabase Systems", IEEE Computer, Vol. 24, No. 12, 1992, pp.12-18.

### Appendix: Semantic Diagram

Figure 1 shows a number of different diagrammatic notations for representing Semantic Diagram. Semantic Diagram represents the mappings between user's concepts and database objects by means

of the graphical notation. The notations are similar to the notations of E-R Diagram.

Entities such as **S\_Female**, **Student** and **Prof** are shown in rectangular boxes. Entity-concpets are distinguished by being placed in double rectangular boxes. Attributes are shown in ovals, and each attribute is attached to its entity by a thick straight line. Similarly, Attribute-concepts are shown in double ellipses. Values are represented in hexagons and attached to attributes by a thin straight line. Value-concepts are represented in double hexagons. Referenced relations correspond to relations that are referenced by a foreign key. Similar connection is used to connect a database object to a concept.





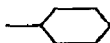



| Symbol   | Meaning             |
|--|---------------------|
|   | Entity              |
|   | Entity Concept      |
|    | Attribute           |
|    | Attribute Concept   |
|    | Value               |
|  | Value Concept       |
|  | Similar Connection  |
|  | Referenced Relation |

Figure 1. Summary of Semantic Diagram notation