Use of On-line Taxonomies in Creating Global Schemas

Mohammad Shafique Al-Shishtawi and Muhammad Shafique
King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia

ABSTRACT: Two approaches for integrating heterogeneous database schemas based on the concept of a global schema are surveyed. The first approach relies heavily on manual resolution of various incompatibilities, particularly semantic, that exist among different local database schemas. The second handles this resolution automatically. In this work, usage of an on-line general taxonomy as a tool for resolving various incompatibilities among component schemas is introduced. This tool is a part of an on-going research for developing a new methodology for integrating heterogeneous database schemas using an on-line general taxonomy. Many advantages offered by the tool are discussed.

Keywords: Schema integration, semantic resolution, heterogeneous databases, federated databases, taxonomy

1. Introduction

Dependency on computer support to facilitate organizations' as well as individuals' daily work has become the norm. One of the major facilities computers offer is information systems. Potentially, these systems are based on different data models, managed by different incompatible DBMSs and run under different environments. Such systems are referred to as heterogeneous database systems. Frequently, users may need to access heterogeneous databases to acquire information. Should these databases satisfy user requests, they must be integrated. [1]

There are three different approaches for integrating heterogeneous databases: total integration, partial integration and interoperability [3]. One way of achieving total integration is via creating a global schema [1]. This work introduces a new methodology of integrating local schemas. It starts where the procedure discussed in [2] does and applies the concept of identifying semantically similar entities found in [1] to automate the process of semantic resolution. Object-oriented concepts are used as a structural framework in the new methodology. The new methodology combines the advantages of the work upon which it is based and avoids major disadvantages.

Section 2 presents basic concepts fundamental to this work. Section 3 summarizes a procedure for building a global schema. In section 4, an automatic way of identifying semantically similar entities is described. Section 5 introduces the new hybrid methodology. The last section, section 6, concludes the work.

2. Basic Concepts

2.1. Integration Process

As mentioned above, users may need to access information stored in autonomous databases. To achieve this, these databases have to be integrated. There are several problems, however, raised by autonomy. The major ones are: [1]

Different data models may be used to model different databases, which may result in completely different conceptualization of information.

Changes at any level in individual databases may take place without taking permission or even notifying database administrators (DBA) of other databases.

Participation in the global database takes place without any modification to the local databases. In addition, getting involved in different global activities is totally up to the local DBA.

Different kinds of incompatibilities (summarized in the next section) may exist among data entities. [3]

One approach for integrating heterogeneous databases is based on creating a permanent global schema. A global schema is basically another schema on top of the local schemas. It provides an integrated view to global users. It does not, however, provide any control over local DBAs decisions (an implication of autonomy). [1]

In order to create the global schema, local DBAs (which are assumed to have comprehensive knowledge about their own databases) are consulted by the global DBA to solve the syntactic and semantic conflicts among entities in local databases. When the global schema is completed, each node should copy it locally in order to achieve integration. This approach suffers from several disadvantages: [1]

The global schema can be huge in size. Nodes having limited storage may not be able to participate in the global system.

Resolving semantic and syntactic differences the way described is very hectic, time consuming and labor intensive.

Local database schemes may be changed arbitrarily and these changes must be reflected on the global schema.

2.2. Incompatibilities

Incompatibilities may occur in the structural representation of data. Moreover, semantic incompatibilities and data inconsistencies are also very probable. These incompatibilities occur due to: [3]

Real world entities are perceived differently by different individuals.

Individual differences in data modeling skills of schema designers.

Variations in the semantic richness of data models used for different schema definitions.

Semantic incompatibilities may take several forms. They include various conflicts. For example, naming conflicts, identity conflicts, type conflicts, key conflicts, behavioral conflicts and scaling conflicts. Moreover, data missing from some schemas while existing in the others is another sort of semantic incompatibility. Furthermore, inter-schema properties and different abstraction levels may cause semantic incompatibility. Details about semantic incompatibility may be found in [2] and [3].

Data represented in different local schemas may be quantitatively incompatible. Reasons include: use of different accuracy levels, asynchronous updates and lack of security. Detailed description of data incompatibility may be found in [3].

2.3. Semantic Knowledge Acquisition Process

In order to integrate different schemes, relationships among entities in different schemes has to be established. This, however, is hampered by different incompatibilities discussed above. To solve this problem, these incompatibilities must be resolved. This, in turn, requires collecting and incorporating missing or implied semantic knowledge for all entities present in different schemes. [3]

The process of semantic knowledge acquisition can either be done manually or be automated. The manual process is hectic and labor intensive. The automatic process, on the other hand, is much easier. As complete semantic knowledge is acquired and stored, the problem of identifying semantically similar entities, possibly represented differently, is greatly simplified. [1]

2.4. Linguistic Concepts

It is possible to identify the semantic relationships among terms using linguistic theory. In order to utilize the linguistic theory automatically in a system, one may need the help of good user interfaces, and on-line dictionaries and thesauruses. The user interface should effectively guide the user, with the help of on-line dictionaries, to determine the semantic relationships among system entities. Basically, these entities are referred to by various linguistic terms. These terms are called access terms. Therefore, the problem reduces to determining the semantic relationships among these access terms. [1]

2.4.1. Semantic Relationships

Based on the semantic content, one may unify words in a taxonomy. Two semantic relationships are of particular importance: hypernymy and synonymy. Hypernyms are more general words. That is, words with broader meaning. Hyponyms, on the other hand, are words with more specific meaning (opposite of hypernyms.) Synonyms can either be strong or weak. Strong synonyms are equivalent from the semantic point of view. They can be substituted for each other in a sentence without changing its meaning. Weak synonyms can also be substituted for each other but with some change in the meaning of the sentence. [1]

2.4.2. Imprecision

Frequently, systems will contain information related to user requests but does not precisely match them. This imprecision is a result of users and system designers perceiving the real world differently. In order to deal with that properly, it is necessary to quantify the similarity between two terms. [1]

A measure used to define the degree of similarity between two terms is called the semantic distance metric (SDM). If all pairs of terms are connected by some combination of hypernym-hyponym and synonym links in a taxonomy structured as a tree, the SDM is the weighted count of the links in the path between any two terms. The path is supposed to be the shortest path between the two terms. Well known algorithms can be used to find the shortest path. [1]

In our work, two terms are considered semantically similar if the SDM between them has a value of one. Higher values simply indicate that that the terms are less similar.

3. Building a Global Schema

3.1. Translating Local Schemas into Component Schemas

The first step toward integration is to translate local schemas of local databases into schemas represented in a common data model (CDM). CDM schemas are referred to as *component schemas* [4]. Component schemas are needed for several reasons: [2]

- Homogenize data models
- Present an integrated view of information about different entities
- Store the implicit/explicit semantic knowledge underlying assumptions obtained during the knowledge acquisition process described above

Four parameters for *each* component schema have to be derived from the corresponding local database. These parameters are: [3]

1. The set of all distinguishable entities (objects) defined in a local database
2. The matrix specifying the relationships among the set of local objects
3. The semantic knowledge collected for each property for each object
4. The mapping knowledge which is required to construct instances of a set of objects from data stored in the local database

3.2. Deriving the Global Schema

After deriving local object schemas, it is possible to derive the global schema. This requires deriving four parameters from the corresponding parameters derived for each component schema. More precisely, the following four global parameters should be derived: [3]

1. The set of objects in the global schema (global objects) and their properties
2. The matrix determining the relationships among global objects. There are three steps involved:
   - Automatic derivation based on the matrices derived for local schemas
   - Manual derivation of missing relationships
   - Consistency checking of the global schema
3. The semantic knowledge of global objects
4. The mapping knowledge from the global schema to various component schemas

4. Automatic Identification of Semantically Similar Entities

The objective is to automatically identify semantically similar access terms used in databases, i.e., the names used in a database schema. To achieve that, an abstract view of access terms of all local databases has to be created and structured in a hierarchy. At the lowest level of the hierarchy are *entry-level* terms that correspond to access terms existing in local databases. At each higher level in the hierarchy, some of the lower level entries are abstracted to form what is called a summary schema. Summary schemas are not exact representation of their children. They retain, however, *most* of the semantics contained in the children. This structure was introduced to facilitate the processing of the *summary schemas model (SSM)* described in [1].

4.1. Taxonomy

The SSM requires a general hierarchical taxonomy with disambiguated definition of words where homographs and polysemy are resolved [1]. The taxonomy should be on-line and should combine information found in traditional dictionaries and thesauruses. It makes use of three types of semantic links: synonyms, hypernyms, and hyponym. Hypernym and hyponym links are reciprocal and construct the hierarchy. Synonym links, on the other hand, are symmetric and travel across the hierarchy between subtrees and leaf nodes. Building such a taxonomy is an achievement by itself. Therefore, it is sufficient to make use of any existing taxonomy (or a combination of them) and augment them if necessary. [1]

4.2. Creating a Hierarchy of Summary Schemas

As mentioned above, a summary schema is a group of access terms abstracted from existing terms in local databases. The terms used in a summary schema are hypernyms of the access terms. To simplify the matter, it is assumed that database schema access terms are the leaf nodes and internal nodes are summary schemas. Each internal node summarizes the schemas of its children. The whole hierarchy is five levels high (top level is level one) which makes it easier to traverse. [1] Figure 1 below shows an example of a summary schema.

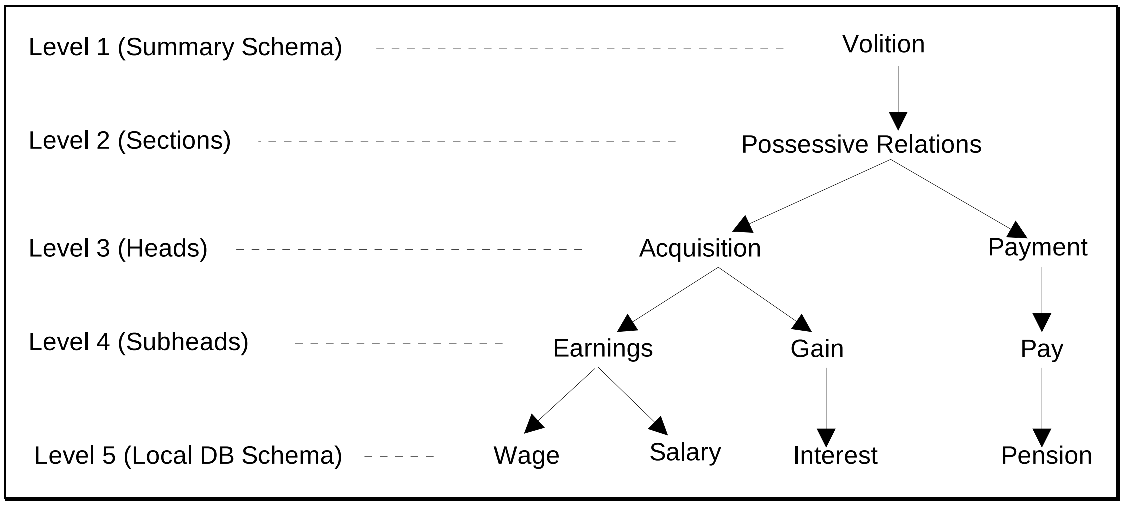

Figure 1: A Sample Summary Schema

Initially, local DBAs have to do some effort mapping all local access terms to entry-level terms in the taxonomy. They are given the capability of automatically looking up terms in the taxonomy where they can select the proper definition of a term. This capability simplifies much the effort they have to make. The remaining effort for constructing (and later maintaining) the hierarchy is simply automatic. [1]

For each entry-level in the hierarchy, a “subhead” is determined via a hypernym link. Subheads contain, in their list of synonyms, pointers to various entry-level nodes. Similarly, hypernym links connect higher levels to their next lower level. List of hyponym links at each higher level, will point to hyponym terms. Hypernyms, as such, represent the summary schemas and hyponyms are more concrete or accurate terms. [1]

5. The Integration Methodology

This section introduces a new methodology for integrating heterogeneous database component schemas. The shaded area of Figure 2 below shows the scope of this work. The approach combines three main ideas:

1. Integrating heterogeneous database component schemas based on building a global schema structure.
2. Identifying semantically similar entities with the aid of an existing on-line taxonomy.
3. The general framework of object oriented systems.

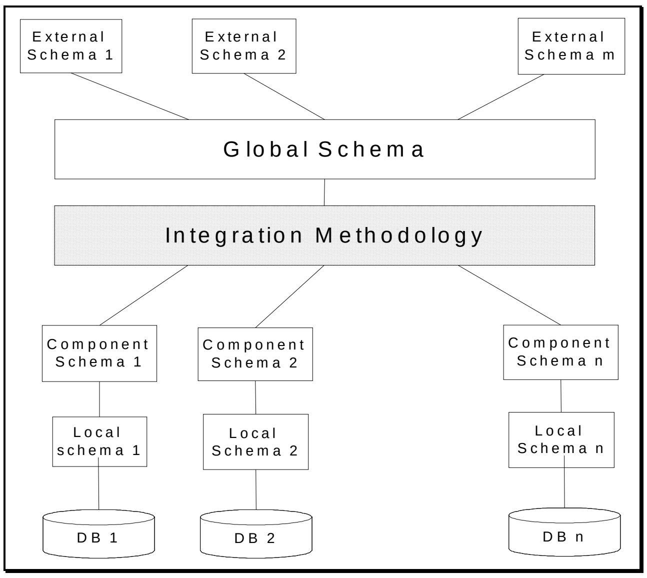

Figure 2: Current Work Fills in the Shaded Area

By putting together these ideas, the new methodology automates the major steps that used to be done manually. These steps have been described briefly in section 3.2.

In section 3 above, it is mentioned that local DBAs have to create four different parameters. With the proposed methodology, local DBAs have to do much simpler work. They have to determine various objects in each database, map them to the corresponding terms in the on-line taxonomy and fill in the missing or implied semantic knowledge. In addition, this task is also simplified by the availability of on-line dictionaries and thesauruses that can be referenced before defining objects in the local object schema. This should reduce semantic incompatibilities to minimum.

After each local DBA translates his database into the CDM form, he has to map each object (represented by a term in the CDM schema) to the corresponding entry in the taxonomy.

Starting with all objects of all local databases, the global DBA has to extract various sets of equivalent objects. Each set of equivalent objects will be represented by only one object in the global schema.

Determination of equivalent objects is simple. If two objects were mapped to two different entries in the taxonomy and it happens that these two entries are synonyms, then these two objects belong to one equivalent set of objects.

5.1. The Taxonomy: A New Look

The basic building block of the new approach is the existence of an on-line general taxonomy. The taxonomy will be used by all DBAs as a pool of their knowledge about their databases. It will serve as a tool for resolving various incompatibilities among objects participating in the federation.

Describing the taxonomy as “general” means that each object used within the databases participating in the federation will have a corresponding entry in the taxonomy. Moreover, needed properties and meta-properties of the objects will also have corresponding entries in the taxonomy. Other objects in the world do not really have to exist in the taxonomy.

The on-line taxonomy described in section 4.1 will be used. It has to be extended, however, to suite the new approach. One major extension is to consider the taxonomy hierarchy as a hierarchy of classes rather than just simple hierarchy of terms. Each term is represented as a class in the hierarchy. Since the taxonomy is general, there will be classes for all objects, properties and meta properties. Figure 3 below shows the general form of an entry (a class) in the taxonomy.

Each class represents a level of abstraction of some real world object. Some of the classes are linked to other classes via cross links. Such links denote that the linked classes are synonyms.

Each class will contain various class variables. These variables correspond to properties of the objects represented by this class. In each class, there will be class methods that are used to achieve compatibility among compatible classes.

Precise definition of each term in the taxonomy must exist in the corresponding class. This will aid local DBAs in determining the classes that best represent their objects.

As each term represents a real world entity, the term will possess properties that correspond to properties of the real world entity it represents. The properties of a term are modeled as properties (instance variables) of the corresponding class in the hierarchy. Some of the properties may be inherited from the term's ancestors (hypernyms).

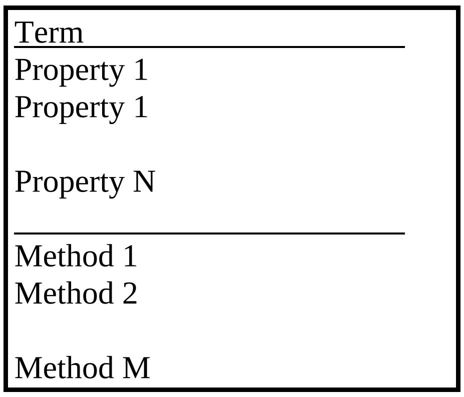

Figure 3: A Class in the Taxonomy

5.2. Capturing Semantic Knowledge

In our work, it is assumed that all needed properties of a term (a real world entity) will be modeled in the corresponding class. This assumption is reasonable since the number of properties is finite for any entity. Having this assumption, properties of any object in some component schema will be a subset of the properties of the corresponding class.

Methods can also be included in the classes. Methods of a class may be its own or inherited from an ancestor. These methods can be used to achieve compatibility among incompatible properties.

The problem of identifying semantically similar entities is now simple. Our work considers two entities to be semantically similar if there exist a synonym link between them in the taxonomy hierarchy. It is not necessary to first check the equivalence among the properties of the two objects as suggested in section 3. If two objects are found to be equivalent this way, then they must be representing the same real world state (see 3.2). If they do, then their sets of properties must be equivalent. Some or all of them might be incompatible, however.

5.3. Achieving Semantic Mapping among Objects

If two properties are equivalent but incompatible, then they must be made compatible. Remember that two properties are compatible if and only if their meta properties are equal. Moreover, two meta properties are equal if and only if their meta values are equal. If the two properties are incompatible due to some meta property, then a transformation has to be defined to make the two meta values compatible. In the new methodology, this can easily be done via methods defined in the classes representing the meta properties.

5.4. An Example

To illustrate the power of this tool and to facilitate its analysis, we take the very same schemas, schema objects and properties presented in Reddy et al. [3] use them with this tool. We will be able to get a result equivalent of that in [3]. To simplify building the example, schemas, objects, and object properties and meta-properties found in [3] are mentioned below:

Local Schemas and Objects
Local Schema 1 = {Emp, Faculty, Dept}
Local Schema 2 = {Employee, Visiting-Prof, Professor}
Local Schema 3 = {Temporary-Staff, V-Prof, Division}

Objects and Properties
Emp = {Emp-Id, Emp-Sal, Dept-Name}
Faculty = {Fac-Id, Fac-Pay, Fac-Specialization}
Dept = {Dept-Budget}
Employee = {E-Id, E-Sal, Div-Name, E-Rank}
Temporary-Staff = {TS-Id, Working-Hours, Wages}
Staff = {Staff-Id, Staff-Sal}

Properties and Meta-Properties
Emp-Sal = {Currency, Periodicity-of-Pay}
Dept-Budget = {Currency, Periodicity-of-Grant}

The following objects were found to be equivalent in [3]:
{Emp, Employee}
{Dept, Division}
{Visiting-Prof, V-Prof}

Figure 4 shows the super class/ subclass information found in [3].

First, a taxonomy should be selected. Figure 5 shows the parts of the selected taxonomy significant to this example.

Second, translation of local schemas to CDM component schemas takes place. The result of translating schema 1, schema 2 and schema 3 to the CDM is shown in Figure 6.

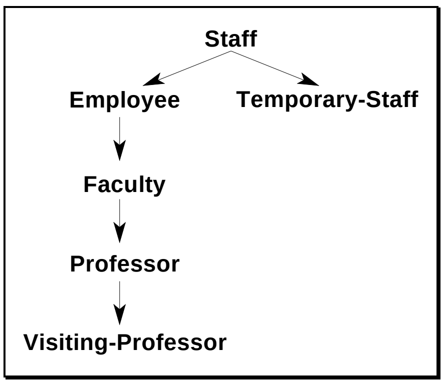

Figure 4: Super class/Subclass Information in Reddy et al.

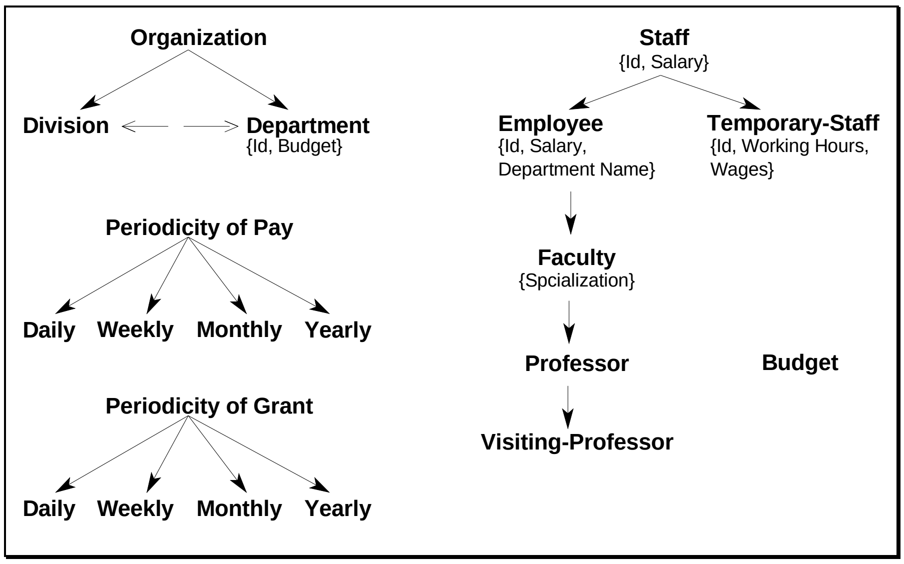

Figure 5: Parts of the Taxonomy for the Example.

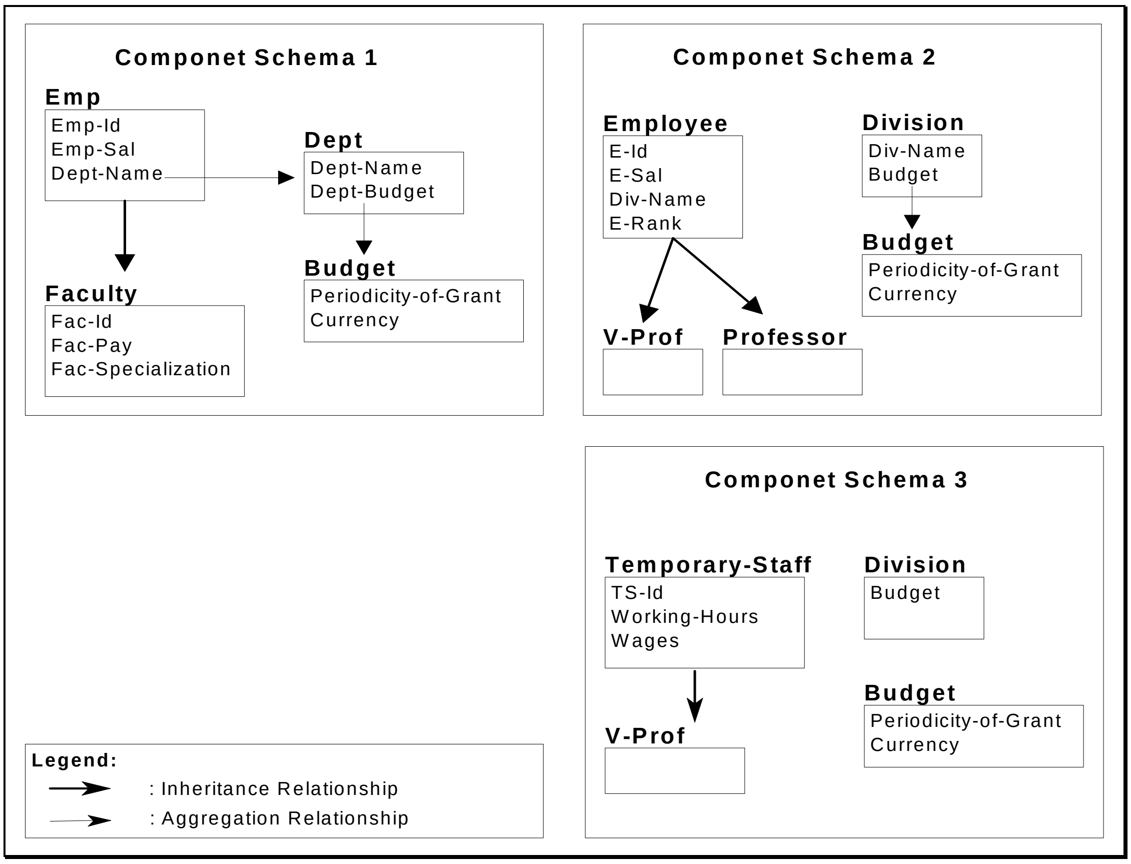

Figure 6: Translated Schemas

Local DBAs in the next step will identify component schema access terms and find proper taxonomy terms corresponding to them. The access terms identified in each component schema and the corresponding taxonomy terms to which they are mapped are mentioned below:

"Component Schema 1.Emp" mapped to "Employee"
"Component Schema 1.Faculty" mapped to "Faculty"
"Component Schema 1.Dept" mapped to "Department"

"Component Schema 2.Employee" mapped to "Employee"
"Component Schema 2.Visiting-Prof" mapped to "Visiting Professor"
"Component Schema 2.Professor" mapped to "Professor"
"Component Schema 2.Div" mapped to "Division"

"Component Schema 3.Temporary-Staff" mapped to "Temporary Staff"
"Component Schema 3.V-Prof" mapped to "Visiting Professor"
"Component Schema 3.Division" mapped to "Division"

Missing or implied semantic knowledge is then explicitly stored. Examples of this knowledge include: "Currency" and "Periodicity of Pay" of the "Salary" property of "Employee", and "Currency" and "Periodicity of Grant" of the "Budget" property of "Department".

The global DBA can then determine equivalent objects. The global DBA should be able to determine the same sets of equivalent objects mentioned above. Since "Component Schema 1.Emp" and "Component Schema 2.Employee" are both mapped to the same taxonomy term, "Employee", their equivalence is obvious. Similarly, "Component Schema 2.Visiting-Prof" and "Component Schema 3.V-Prof" are obviously equivalent. "Component Schema 1.Dept", "Component Schema 2.Div" and "Component Schema 3.Division" are also equivalent since the access terms are either mapped to the same taxonomy term or to synonymous taxonomy terms.

The global DBA should now select an access term for each set of equivalent access terms. The following may be selected as representative access terms:
"Employee" represents "Component Schema 1.Emp" and
"Component Schema 2.Employee"
"Department" represents "Component Schema 1.Dept",
"Component Schema 2.Div" and
"Component Schema 3.Division"
"Visiting Professor" represents "Component Schema 2.Visiting-Prof" and
"Component Schema 3.V-Prof"

"Component Schema 1.Faculty" and "Component Schema 3.Temporary-Staff" are the only two access terms without equivalents. Their representative access terms are "Faculty" and "Temporary Staff" respectively.

Finally, compatibility methods are to be defined. In this example, it is assumed that "Department.Budget" is granted five times a year in Rupees while "Division.Budget" is granted once a year in terms of Dollars. Here, the meta-properties "Periodicity of Grant" and "Currency" have different meta-values. To achieve compatibility, an HCM (Horizontal Compatibility Method) have to be defined in "Division" class. The method has two formal parameters that are assigned proper values when used. In this example, it is assumed that \$1 = 30 Rupees. Hence, "PoG" and "Rate" will be assigned the values: 1/5 and 30 respectively. Figure 7 shows an outline of the method.

The method shown in Figure 7 will be invoked whenever the global data item "Department.Budget" is queried and the local data item "Division.Budget" has data to participate.

In this example, it is assumed that an "Employee" is paid his "Salary" in Rupees and a "Faculty" in Dollars. Since the "Salary" property is defined in both "Employee", the super class, and "Faculty", the subclass, (see Figure 5) a VCM (Vertical Compatibility Method) is needed to resolve the conflict. If interest is in the "Salary" of the "Faculty", the VCM should be defined in "Employee" as shown in Figure 8. In Figure 8, "Rate" is given the value 1/30 since \$1 is assumed to be equal to 30 Rupees.

```
Method: DivisionBudget-to-DepartmentBudget (PoG, Rate);
        RETURN Budget.value × PoG × Rate
end;
```

Figure 7: HCM Method Resolving the Budget Conflict

```
Method: EmployeeSalary-to-FacultySalary (Rate);
        RETURN Salary.value × Rate
end;
```

Figure 8: VCM Method Converting Employee Salary to Faculty Salary

6. Conclusion

In this work, various concepts of particular importance to the subject of integrating heterogeneous database schemas have been presented. Integrating heterogeneous database schemas using global schema approach has been briefly discussed. Identifying semantically similar entities and resolving conflicts among them has been illustrated. Using an on-line general taxonomy as a tool for resolving incompatibilities is discussed. The taxonomy is a “by-product” of the on-going research for developing a new methodology for integrating heterogeneous database schemas.

Acknowledgments

The authors would like to express their deep appreciation to King Fahd University of Petroleum and Minerals for providing the necessary facilities to accomplish this work.

References

[1] Bright, M, Hurson, A and Pakzad, S. Automated resolution of semantic heterogeneity in Multidatabases. ACM Transactions on Database Systems, Vol. 19, No. 2, June 1994, pp. 212-253.

[2] Pitoura, E, Bukhares, O and Elmagarmid, A. Object orientation in multidatabase systems. *ACM Computing Surveys*, Vol. 27, No. 2, June 1995, pp. 144-195.

[3] Reddy, M, Prasad, B, Reddy, P and Gupta, A. A methodology for integration of heterogeneous databases. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 6, No. 6, December 1994, pp. 920-933.

[4] Sheth, A and Larson, J. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, Vol. 22, No. 3, September 1990, pp. 183-236

Bibliography

[1] Batini, C and Navathe, S. A comparative analysis of methodologies for database schema integration. ACM Computing Surveys, Vol. 18, No. 4, 1986, pp. 323-364.

[2] Batini, C. A methodology for database schema integration in the entity relationship model. IEEE Transactions on Software Engineering, Vol. SE-10, No. 6, 1984.

[3] Bertino, E and Martino, L. Integration of heterogeneous database applications through an object-oriented interface. Information Systems, Vol. 14, No. 5, 1989.

[4] Frog, J. Converting Relational to Object-Oriented Databases. ACM SIGMOD RECORD, Vol. 26, No 1, 1997, pp. 53-58.

[5] Kim, W and Seo, J. Classifying schematic and data heterogeneity in multidatabase systems. Computer, Vol. 24, No. 12, Dec. 1991, pp. 12-18.

[6] Larson, J. A theory of attribute equivalence in databases with application to schema integration. IEEE Transactions on Software Engineering, Vol. 15, No. 4, April 1989.

[7] Reddy, M, Prasad, B and Reddy, P. A Model for resolving semantic incompatibilities in integrating heterogeneous databases. Proceedings of International Conference on Management of Data, December 1989.

[8] Reddy, M. Heterogeneous distributed database management systems: modeling and managing heterogeneous data. Ph.D. Thesis, School of Mathematical and Computational Information Sciences, University of Hyderabad, India, 1990.