Integrating Heterogeneous Database Schemas Using as On-line Taxonomy: A Methodology for Creating a Global Schema

by

Mohammad Shafique Ahmad Al-Shishtawi

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER SCIENCE

April, 1997

Integrating Heterogeneous Database Schemas Using an On-line Taxonomy: A Methodology for Creating a Global Schema

BY

Mohammad Shafique Ahmad Al-Shishtawi

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER SCIENCE

April 1997

KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS
DHAHRAN 31261, SAUDI ARABIA

COLLEGE OF GRADUATE STUDIES

This thesis, written by

MOHAMMAD SHAFIQUE AHMAD AL-SHISHTAWI

under the direction of his Thesis Advisor and approved by his Thesis Committee, has been presented to and accepted by the Dean of the College of Graduate Studies, in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE.

Thesis Committee

Muhammad Shafiq 7 June, 97
Dr. Muhammad Shafique (Chairman)

7 June, 97
Dr. Abdullah Al-Sukairi (Member)

Jun 7, 1997
Dr. Mostafa Aref (Member)

٩٧/٦/٧ هـ
Dr. Muhammad Al-Mulhem (Member)

Department Chairman

Dean, College of Graduate Studies

9/6/97
Date

كلية الدراسات العليا
COLLEGE OF GRADUATE STUDIES

Dedicated to

my parents

and

my brother

whose prayers, motivation and

moral support have brought about

the accomplishment of this work

ACKNOWLEDGMENT

Praise to Allah, the almighty God, for his support, blessings and disguise throughout this work. Acknowledgment is due to King Fahd University of Petroleum and Minerals (KFUPM) for all the support extended during this research.

I would like to express my gratitude to my thesis committee chairman Dr. Muhammad Shafique for his advice and guidance. I would like also to thank my thesis committee members Dr. Abdullah Al-Sukairi, Dr. Mostafa Aref and Dr. Muhammad Al-Mulhem for their significant suggestions and continuous support.

I would like to express my deep appreciation for the faculty and staff members of the College of Computer Science and Engineering for everything they did. I would like also to acknowledge all the faculty and teachers who taught me during my school and academic years.

Thanks are due to the Information Technology Center of KFUPM for providing all necessary computing facilities. Special thanks for all of my friends and colleagues for their help and cooperation.

Last and not the least, I would like to express my sincere gratitude and appreciation for my parents and my brother for their motivation, support and sacrifices.

CONTENTS

LIST OF FIGURES

THESIS ABSTRACT

Full Name: Mohammad Shafique Ahmad Al-Shishtawi

Title: Integrating Heterogeneous Database Schemas Using an On-line Taxonomy: A Methodology for Creating a Global Schema

Major Field: Information and Computer Science

Date of Degree: April 1997

In the process of developing heterogeneous database systems, one of the main tasks is integrating various component database schemas. Several methodologies for integrating component schemas can be found in the literature. These methodologies are criticized for being very human labor intensive, for demanding extensive global DBA knowledge, for the difficulty in determining semantically similar objects, and for the difficulty in maintaining the global schema.

In this thesis, a new methodology for integrating heterogeneous database schemas is developed. The new methodology avoids the disadvantages that exist in the work upon which it is based. In addition, it offers its own advantages. First, missing relationships among access terms do exist naturally in an on-line general lexicon taxonomy and the need to manually derive them no longer exists. Second, semantic knowledge of global objects is derived automatically. Finally, automatic knowledge mapping is achieved.

Master of Science Degree

King Fahd University of Petroleum and Minerals
Dhahran 31261, Saudi Arabia

April 1997

خلاصة الرسالة

اسم الطالب الكامل: محمد شفيق أحمد الشـــشتاوى

عنوان الدراســة: دمج قواعد البيانات غير المتجانسة باستخدام تصنيف للكلمــات: طريقــة منهجية لإنشاء مخطط شامل

التخــصـــــص: معلومات وعلوم الحاسب الآلي

تاريــخ الشهــادة: إبريل ١٩٩٧ م

إن أحد المهام الرئيسية في عملية إنشاء نظم قواعد البيانــات غــير المتجانســة هــي دمــج مخططات قواعد البيانات المكونة للنظام. يوجد عدة طرق منهجية للقيام بهذا الدمج تم نشرها في أبحاث سابقة. وقد وجه نقد لهذه الطرق لشدة اعتمادها على العنصر البشــري، ولكونــها تتطلب من المدير العام لقواعد البيانات معرفة شاملة بها، ولصعوبة تحديد الذوات (الأجســام) المتشابهة في المعنى وأيضا لصعوبة صيانة المخطط الشامل لقاعدة البيانات.

في هذه الرسالة، تم تطوير طريقة منهجية جديدة لدمج مخططات قواعد البيانات غير المتجانسة. تتفادى الطريقة الجديدة عيوب الطرق السابقة التي بنيــت علــى أســسها هــذه الطريقة، كما أنها تأتى بمميزات جديدة. أولا، وجود العلاقات فيما بين "عبــارات الوصــول" بشكل طبيعي في تصنيف الكلمات ولا حاجة لاستنتاجها يدويا. ثانيا، يتم اســتنتاج المعرفــة بمعاني الذوات في النظام الشامل ذاتيا. وأخيرا، يتم مقابلة المعرفة ذاتيا.

درجة الماجستير في العلوم

جامعة الملك فهد للبترول والمعادن

الظهران ٣١٢٦١، المملكة العربية السعودية

إبريل ١٩٩٧ م

CHAPTER ONE

INTRODUCTION

Starting at late sixties, database management systems (DBMS) began taking over older techniques of file-based data processing. A major reason for adopting the DBMSs approach is that they provide the capability of defining integrated views of related data for various database applications. Advantages gained are many, among them are: elimination of duplications, avoiding the multiple updates problem and minimizing inter-application inconsistencies. [1]

Over the years, database systems (DBS) have spread and become a major support of organizations' and individuals' daily work. Various databases were developed to satisfy specific requirements of certain users. Such autonomous (potentially heterogeneous) database systems are unable to satisfy current users requirements characterized as being more sophisticated and involving more than one database. To meet recent user requirements, integrating these heterogeneous databases (HDB) becomes a necessity. [3]

As the popularity of databases increase, so does the heterogeneity among them [8]. Heterogeneity has a wide range of possibilities. It could range from just simple

structural differences to major differences in design and/or modeling, for example [13, 18]. Existence of heterogeneity implies structural and semantic conflicts. Such conflicts hamper the process of database integration. [17]

Research in the area of heterogeneous database systems (HDBS) can be traced back to late seventies. It can be divided into three phases: [8]

1. Late seventies: Establishing the feasibility of integrating heterogeneous DBMSs (HDBMS)
2. Early eighties: Schema integration research and other operational issues
3. Late eighties: Global transaction management

The interest of this thesis is in schema integration. Particularly, it is concerned with developing a new schema integration methodology. The work is based mainly on the schema integration methodology described in [17]. Research described in [3, 14, 15 and 18] provide several concepts of particular importance for the development of the new methodology. These concepts include: identification of semantically similar data in different local databases [3], effect of object orientation on designing and implementing federated database systems (FDBS) [15], a reference system architecture for FDBSs [18], and classifying semantic, syntactic and data conflicts [14]. The work is related to semantic research since integration is achievable only if structural and semantic conflicts are resolved [3, 17].

1.1. Work Motivation

It has been mentioned in the literature that completely automating the process of schema integration is not possible! The difficulty is believed to be in discovering the relationships among access terms of various schemas. The main reason given for this difficulty is the necessity of capturing the semantics of the schemas. [18] This thesis agrees with this point of view to a certain extent. However, major steps toward making the problem much simpler can be taken.

The view in this thesis is based on the fact that relationships among access terms representing real world objects do exist naturally; they don't have to be discover again. Hence, if these objects (or at least the ones involved in various schemas) and their attributes can be represented as well as relationships among them, the problem reduces to only mapping various schema objects to their corresponding representations. It is really not necessary to capture *all* the semantics involved. That is, in this representation, it is necessary to capture only needed objects, needed object attributes and needed relationships.

1.2. Objectives

Integrating local database schemas by creating a global schema has been described in the literature. Several integration methodologies have been described. Some of the major problems associated with these methodologies are: [3]

- They are very human labor intensive

- Extensive knowledge is required by the global DBA about schemas to integrate, how to integrate them and about global users requirements.
- Determining semantically similar objects for the purpose of unifying them is difficult.
- Global schema creation and maintenance is difficult and time consuming.

The methodology developed in this research tries to avoid or alleviate these problems. The development of the methodology has progressed with the following objectives in mind:

1. Make the process less human labor intensive. That is, partially automate the process.
2. Distribute the work and effort required in a better way over both local and global DBAs.
3. Provide an easier way of determining semantically similar schema objects.
4. Provide an easier way to create and maintain the global schema.

1.3. Thesis Organization

The thesis is divided into five chapters. Chapter 1 presents an overview of the work. Chapter 2 provides the necessary background and reviews basic concepts. It is divided into three main sections: section 2.1 presents a general classification and a schema architecture of HDBSs, section 2.2 addresses major issues in the context of schema integration, section 2.3 summarizes the schema integration methodology developed by

Reddy et al. [17] Work related to the research is also presented in the section. Chapter 3 discusses major issues concerning semantically similar objects.

Chapter 4 provides details of the contributions of this research. The chapter presents an overview of the developed methodology in section 4.1. Section 4.2 describes the extensions introduced to the basic building block of the methodology, the general lexicon taxonomy. Section 4.4 discusses the details of the methodology phases. Sections 4.3 and 4.5 provide two examples to illustrates the way the methodology works. The methodology is analyzed and compared against other methodologies in section 4.6. Section 4.7 gives general implementation guidelines of the methodology. Chapter 5 describes a case study where the developed methodology is applied. Finally, chapter 6 concludes the work and gives future directions.

CHAPTER TWO

BACKGROUND AND OVERVIEW

2.1. General Classification and Schema Architecture of HDBSs

The survey by Sheth and Larson [18] describes two main issues: a classification of multidatabase systems (MDBS) and a comprehensive five-level schema architecture for heterogeneous database systems. The survey tries to unify conflicting terminology used in heterogeneous database research. The classification and terminology present in the survey are adopted in this research. The five-level schema architecture will be adapted to suit the undergoing research. The rest of this section elaborates the two issues.

2.1.1. Classification

Figure 1 shows an autonomy-based classification of MDBSs (adopted from [18]). An MDBS, by definition, is one that supports operations on several DBSs each of which is managed by a potentially different DBMS. Each of these DBSs is called a

component DBS. If each component DBS is managed by the same DBMS, the MDBS is considered homogeneous. If this condition is not fulfilled, the MDBS is considered heterogeneous. [18]

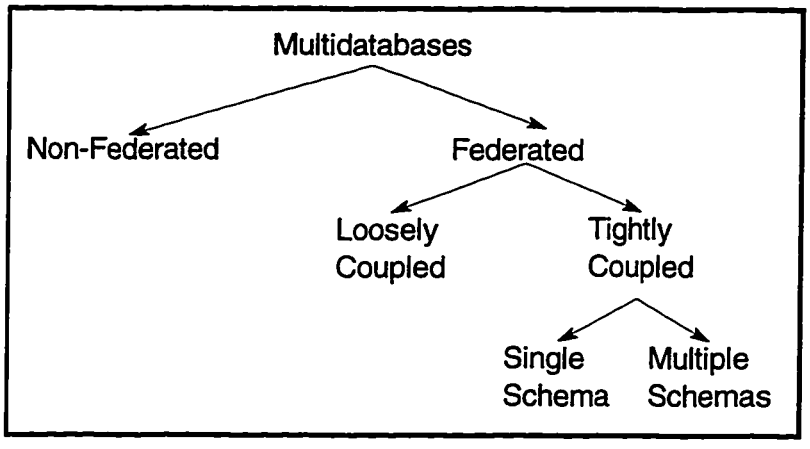

Figure 1: Autonomy-based Classification of MDBSs

MDBSs can either be federated or non-federated. A federated DBS (FDBS) consists of autonomous component DBSs each of which provides controlled sharing of its resources. No centralized control is exhibited over any of the component DBSs. Generally speaking, most FDBSs are heterogeneous in nature. Hence, FDBSs can also be referred to as heterogeneous database systems. [18]

FDBSs can be further classified as loosely coupled and tightly coupled systems. Loosely coupled systems are created and maintained by their users. Loosely coupled systems can support several federated (global) schemas. Tightly coupled systems, on the contrary, are created and maintained by the federation (global) database administrator (DBA). Depending on the number of federations (one or more), tightly coupled systems are divided into single schema and multiple schema FDBSs, respectively. [18]

2.1.2. General Schema Architecture

Figure 2 presents the five-level schema architecture described by Sheth and Larson. At the very bottom of the figure lie the various databases. At the next higher level comes the local schemas which represent the conceptual schema of the corresponding databases. Component schemas are derived by translating each local schema to a common data model (see section 2.3.1.1 for more details). Each component schema may have zero or more export schemas. Each export schema is a subset of the corresponding component schema. This subset is the part of the component schema that is made available for global use. Export schemas may also include access control information with respect to the data managed by the component schema. On top of the export schemas come the global schemas. Each global schema represents an integration of several export schemas. On top of the global schemas, external schemas may be defined. External schemas are schemas defined for use by different users and/or applications or classes of them. They are used for reasons of customization, specifying additional integrity constraints and specifying access control information with respect to the data managed by the FDBSs. [18]

Reddy et al. [17] build their methodology on the basis of the four-level schema architecture shown in Figure 3. Despite terminology inconsistencies between the two

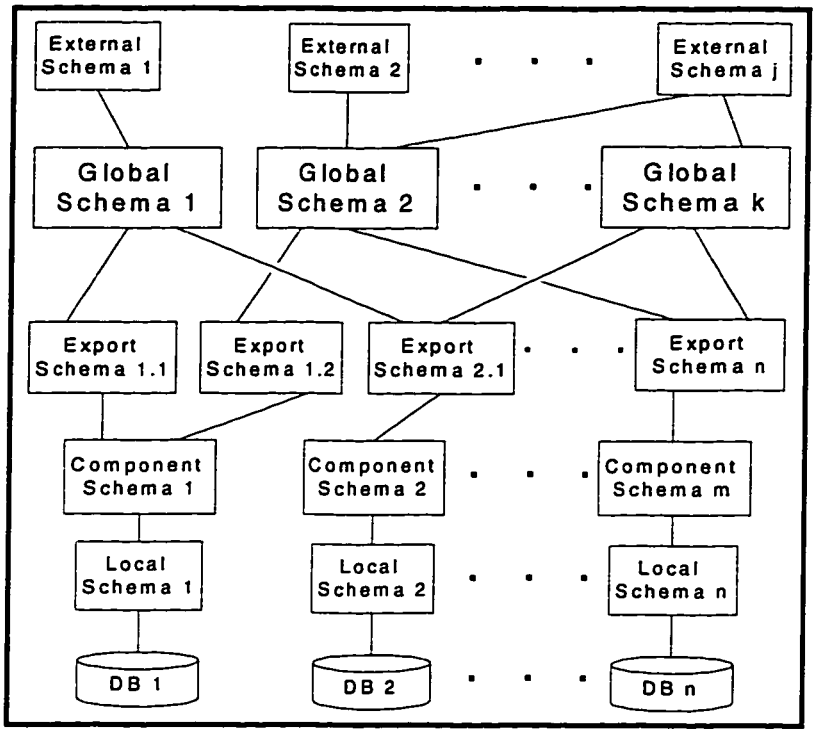

Figure 2: Five-level Schema Architecture of HDBSs

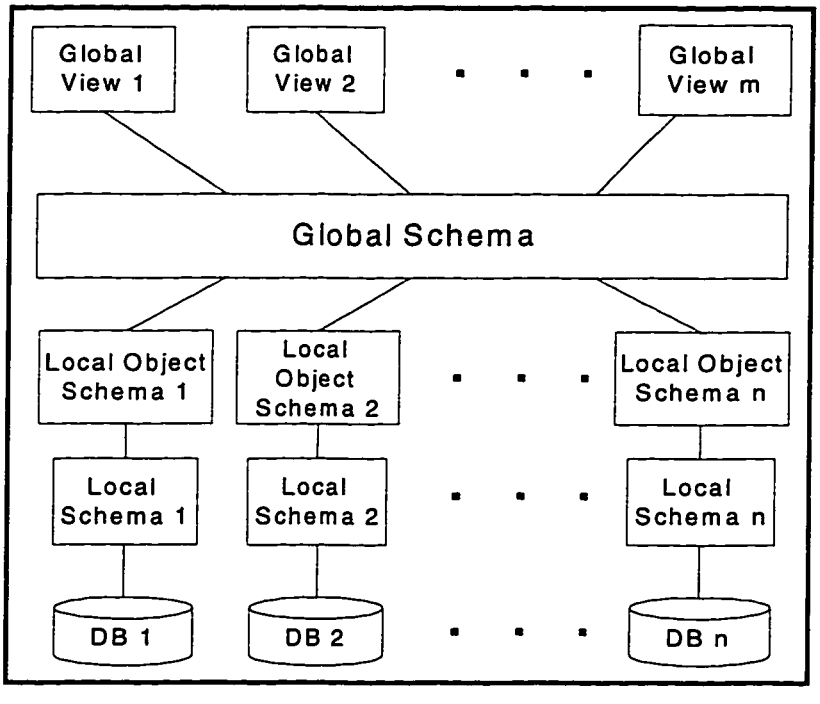

Figure 3: Four-level Schema Architecture of HDBSs

architectures, the four-level schema architecture is clearly a subset of the (more) comprehensive five-level architecture. Local Object Schemas and Global Views in Figure 3 correspond to Component Schemas and External Schemas in Figure 2 respectively. The missing level (export schema level) does not affect the generality and applicability of the methodology developed by Reddy et al. As a matter of fact, export schemas can be considered redundant as far as integration methodology development is concerned. The reason is that export schemas can simply be considered as a subset of the corresponding component schema. [18]

2.1.3. Research Schema Architecture

As mentioned above, the methodology developed in this research is based on that developed in [17]. For this reason and to avoid terminology confusions, the terminology used in Sheth and Larson [18] are applied to the four-level architecture proposed by Reddy et al. [17]. Figure 4 shows the resulting schema architecture that is used through out this research. To avoid any confusion with the terminology used in Figure 2 and Figure 3, the various levels of Figure 4 are described below.

At the bottom of Figure 4 are component databases. Component databases are simply the local databases. The first level of schemas, local schemas, lies above the component databases. A local schema represents the conceptual schema of the corresponding component database. Potentially, local schemas are expressed in different data models. [18]

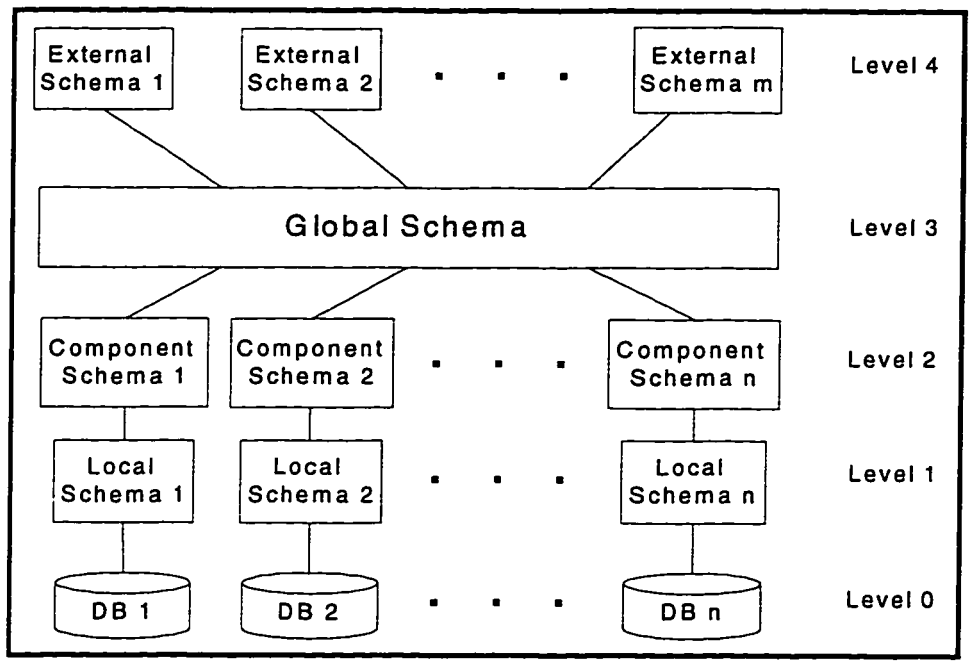

Figure 4: Research Schema Architecture

Corresponding to each local schema is a component schema. Each component schema is an equivalent translation of the corresponding local schema. Component schemas are all expressed in the same data model, called the common data model (CDM). Object oriented data model has been suggested for use as a CDM for the capabilities it provides. [10, 13, 15, 18]

At the third level comes the global schema. It represents the result of integrating the underlying component schemas. External schemas are defined on the fourth level. Eternal schemas are used for reasons of customization, defining more integrity constraints and specifying global access control constraints. [18]

2.2. Schema Integration

Schema integration is one of the major tasks which arise in the context of developing HDBSs. It is defined as the process of integrating existing or proposed component database schemas into a single federated (usually called global) schema. [1, 15, 18] This section is devoted for discussing main issues related to this process.

2.2.1. Design Approaches

There are two major design approaches for integrating HDBSs. Each approach corresponds to one class of HDBSs. Both approaches produce global schemas. The following two sections describe the two major approaches.

2.2.1.1. Temporary Global Schema Approach

Multidatabase languages are similar to standard structured query language (SQL). They possess, however, additional capabilities. These capabilities include: [3]

- Defining global schemas as views
- Handling various structured and semantic conflicts (see section 2.2.3). Several semantics of the same data can be defined by some multidatabase languages.
- Manipulating different data representations
- Transforming source data to the integrated representation
- Making implicit decisions in interpreting user requests and provide default functions

The temporary global schema approach is usually used for creating global schemas for loosely coupled HDBSs. Loosely coupled systems are characterized by integrating large number of very autonomous databases where users are mostly interested in creating on-the-fly global schemas, fill them with data and then destroy them after use. In this regard, temporary global schema approach is more suitable than the permanent global schema approach discussed in the next section. The temporary global schema approach offers the following advantages: [3, 18]

- The global DBA has much less overhead with respect to integration. It is the responsibility of the user to specify the relations and mappings among component schema objects

- Multiple semantics are supported since different users may interpret and map schema objects differently. This is useful when user needs cannot be anticipated
- Much less development time, maintenance efforts and storage requirements for global schemas

2.2.1.2. Permanent Global Schema Approach

Global schemas for tightly coupled systems are created on the basis of this approach. Many methodologies and procedures are described in the literature. The common feature among them is the activity of resolving syntactic (structural) and semantic conflicts among component databases involved in the integration. The result of this activity is the creation of one (or more) homogenized and integrated schema which represents a summary of the information contained in the component schemas. [1, 3, 17, 18]

Even though this approach provides for data independence where duplication, heterogeneity and location information are hidden, it suffers from serious disadvantages: [3]

- It is very human labor intensive
- Extensive knowledge is required by the global DBA about component schemas involved in the integration, how to integrate them and about global users requirements

- Global schema maintenance against any possible modification in the autonomous component schemas is very difficult
- Global schemas can be huge in size such that users with limited storage may not be able to participate in the federation
- Determining semantically similar objects and then resolving the wide range of possible conflicts among them is a difficult and time consuming task

2.2.2. Major Problems of Schema Integration

The autonomous nature of database development is the major cause of database heterogeneity [3, 13, 18]. The more heterogeneous the participating databases are, the more difficult it is to integrate them. Several main problems can be identified: [3]

- 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 may take 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 conflicts (described in section 2.2.3) may exist among data objects. [17]

2.2.3. Schema and Data Conflicts

Even though the feasibility of integrating HDBMSs was established since two decades, HDBs are not yet popular. The basic problem seems to be the lack of a comprehensive understanding of schema conflicts (both syntactic and semantic) and data heterogeneity. Despite the many methodologies proposed for schema integration, a practical solution for homogenizing various types of heterogeneity and conflicts is still absent. In this section, the issue of schema and data conflicts is discussed. [1, 8, 14]

2.2.3.1. Definition

Any two syntactic representations of the same concept are said to have a conflict if the representations can be described as: [1]

1. Equivalent representation: Different but equivalent modeling constructs are used. The notion of equivalence may be any of the following three types:
    a) Behavioral equivalence: Two representations are equivalent in this sense if for every instantiation of one representation, there exists a corresponding instantiation of the other that provides the same set of answers to any given query.
    b) Mapping equivalence: Two representations are equivalent if for every instance of the first representation there exists a corresponding instance of the other representation.

c) Transformational equivalence: One representation is equivalent to another in this sense if applying a set of transformations to the first representation will yield the second. The transformations applied must be atomic and, hence, equivalence preserving.

2. Compatible representation: No contradiction exists neither in the modeling constructs used, in the designer's perception nor in the integrity constraints. The representations are neither identical nor equivalent though
3. Incompatible representation: Representations are contradictory

If the two concepts, on the other hand, have identical syntactic representations across component schemas, then no conflicts exist. The phrase "identical syntactic representations" means that exactly the same modeling constructs are used to represent the concept. This case rarely happens though, possibly due to any of the problems mentioned in section 2.2.2.

2.2.3.2. Schema Integration and Conflicts

In order to integrate HDB schemas, various kinds of heterogeneity and conflicts must be resolved. In addition, resolving these conflicts should not increase the processing time and response time significantly. [16] To achieve that, identification of common concepts is necessary. However, due to several reasons (mentioned below), common concepts are mostly represented by syntactically different representations that may be semantically conflicting as well. A better (thorough) understanding of conflicts and causes of heterogeneity is needed to successfully resolve the conflicts and, hence,

carry out the integration. [14, 17] The following, summarizes the major causes of conflicts: [1, 17]

- Real world objects 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.

2.2.3.3. Classification of Conflicts

Two kinds of conflicts pertain to integrating HDB schemas. Schema conflicts, the first kind of conflicts, refers to possible syntactic and semantic (see section 2.2.3.4 for various forms of semantic conflicts) differences among component schemas. Schema conflicts occur basically due to two reasons: [14, 17]

1. Using different syntactic constructs to represent the same information. Syntactic conflicts occur due to this reason.
2. Associating different specifications (semantic information) with the same concept. This causes semantic conflicts to occur.

The second kind of conflicts is data conflicts. It refers to contradictory data. Data contradiction (inconsistency) my be one of two types: [14]

1. Wrong data: occurs when integrity constraints imposed implicitly or explicitly on the data are violated.
2. Data conflicts based on schema conflicts: occur when schema conflicts take place.

Data conflicts may occur due to several reasons: [14, 17]

- **Different accuracy levels**: The same data may be stored in different databases with different accuracy levels (e.g. cm and meter).
- **Different units**: Objects may be stored in different units in various schemes (e.g. cm and inch)
- **Different expressions**: The same data is represented by different expressions (e.g. CD-ROM and Compact Disk Read Only Memory)
- **Asynchronous updates**: Local databases are autonomous and updates may not be reflected on some of them at a point of time or even at all.
- **Lack of security**: Unauthorized users may change local databases due to lack of security.

Kim and Seo [14] propose a practical and complete classification of schema conflicts based on the relational model. In the rest of this section, this classification is generalized to include other models too.

Figure 5 shows the general classification of schema conflicts. Kim and Seo base their classification on relational model constructs like tables and attributes. Trying to be more general, the classification presented here is based on *Information Units (IU)* that can be at any level of granularity of any data model. For example, attributes, objects, functions and facts are different possible IUs. Schema conflicts could occur between the same two IUs. This kind of conflicts can either be one-to-one IU conflicts (e.g. one attribute conflicting with another) or many-to-many IU conflicts (e.g. two objects conflicting with three other objects representing the same information). Both

one-to-one and many-to-many IU conflicts can be syntactic conflicts or semantic conflicts as illustrated in Figure 5 .

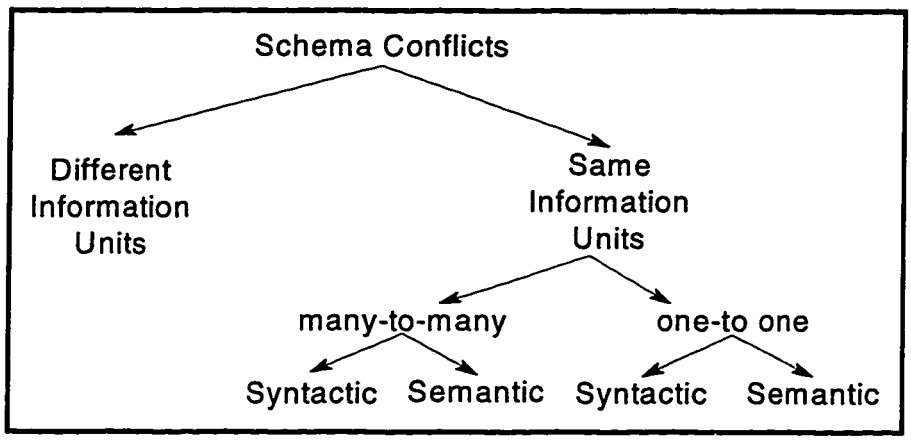

Figure 5: Classification of Schema Conflicts

Schema conflicts could occur between two different IUs. For example, it could occur between an attribute and a table. This kind of conflicts can be regarded as several m-to-n conflicts. It is clear that m-to-n conflicts are definitely conflicting in syntax. They could also conflict in semantics. Compound conflicts could also occur. These conflicts can simply be considered as several conflicts of the types discussed above. The following section provides more elaboration on semantic conflicts.

2.2.3.4. Forms of Semantic Conflicts

Many semantic conflict forms are possible. Here are some of the most obvious ones. [17]

- **Naming conflicts**: Objects in local databases are named autonomously. Conflicts may result in different concepts being named by one name (homonyms) or one concept named differently (synonyms).

- **Identity conflicts**: Different objects in different databases are used to represent the same concept. [3]
- **Type conflicts**: Different structures are used to represent the same concept in various schemes.
- **Key conflicts**: Different keys in different schemas are assigned to the same concept.
- **Behavioral conflicts**: Refer to various insertion/ deletion policies associated with the same objects in different schemes.
- **Missing data**: The same concept in different schemes may be represented by different attributes/properties. Essentially, each representation of the concept will be missing some information existing in the other.
- **Level of abstraction**: Objects in some schemas may be more detailed than others.
- **Identification of related concepts**: Concepts in local schemas may be related even though they are not the same. Properties relating these concepts have to be identified.

2.3. Schema Integration Methodologies

In this section, some of the latest mature work that has been described in the literature is summarized. This work provides important concepts for this research.

2.3.1. Reddy's Methodology

In this section, the methodology developed by Reddy et al. [17] is summarized.

2.3.1.1. 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* [10]. Adopting a CDM is necessary for several reasons: [17]

- Homogenize data models used in different local schemas and achieve uniformity in modeling constructs.
- Present an integrated view of information about each entity since it could be distributed over local schemas.
- Store the explicit semantic knowledge as well as the implicit semantic knowledge and 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. The first parameter is the set of all distinguishable entities (objects) defined in a local database. Each object in the set has some properties that characterize it. [17]

The second parameter is the matrix specifying the relationships among the set of local objects. This matrix need only to be generated by each local DBA based on his local database only. The relation between the instances of any two objects may belong to any of the following categories: [17]

- Equivalence relation (cannot exist in a single database schema)
- Super class/ subclass relation
- Overlapping relation
- Disjoint relation

The third parameter is the semantic knowledge collected for each property for each object. This knowledge is captured via the set of meta-properties and their corresponding meta-values. Meta-properties provide the semantic meaning to property values. [17]

The last parameter is the mapping knowledge which is required to construct instances of a set of objects from data stored in the local database. It provides details about how an object in a component schema is mapped to its storage representation in the corresponding local database. The procedure for deriving this parameter is data model dependent. [17]

2.3.1.2. 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. [17]

The set of objects in the global schema is constructed from the various sets of objects of local schemas derived previously. First, groups of equivalent objects must be identified. Each group is then replaced by a single object in the global schema. Equivalent objects have *equivalent real world state*. Two objects are said to have

equivalent real world states if they represent the same sets of instances of some real world entity. The two objects, though, need not provide identical information. [17]

The next step after deriving global schema objects, is to derive their properties. Similar to the equivalence among objects, two properties may also be equivalent. If two properties describe the same characteristic of some real world object, they are said to be equivalent and should be represented by a single property in the global schema. In general, a group of, say, *m* properties of an object may be equivalent to *n* properties of another equivalent object. This leads to the possibility that a concept modeled by several properties of an object may be equivalent to another object. This type incompatibility is a result of differences in abstraction levels. [17]

The second parameter to derive is the matrix determining the relationships among global objects. Detailed derivation procedure of this matrix is described in [17]. The three steps involved are:

1. Automatic derivation based on the matrices derived for local schemas: The relationships between two objects can be found easily from these matrices if the two objects belong to the same schema. If not, then the relationship between them is defined as the sum of their corresponding entries in the corresponding relationship matrices. If conflicts occur, they have to be resolved manually as they arise.
2. Manual derivation of missing relationships: This is simply done by the global DBA in case some relationships could not be derived in the previous step.

3. Consistency checking of the global schema: Particularly, two inconsistencies are determined and resolved in this step. The first occurs when it can be implied from the global matrix that one object is a subclass of another object, yet they are disjoint. The second inconsistency occurs when cycles in subclass relationships can be determined from the global matrix.

The third parameter to derive is the semantic knowledge of global objects. As objects are characterized by their properties, the semantic meaning of global properties must be determined. The semantic meaning of a property is captured via its meta-properties. Meta-properties of a global property are derived from the meta-properties of the equivalent property set from which the global property was derived [17]

Object integration is achieved by the last parameter, the mapping knowledge from the global schema to various component schemas. If two equivalent objects from two different local object schemas are to be integrated, they have to be made compatible. They require making their equivalent properties compatible. This is achieved by identifying ways the equivalent properties of the objects are incompatible and then making them compatible. Making two properties compatible is achieved via transforming the meta-values of the meta-properties of one incompatible property to the other. Another way of integrating two objects is via generalization. This, however, is applicable only to objects that are overlapping (but not contained). [17]

2.3.2. Related Semantic Work

In this section, work related to this research is summarized.

2.3.2.1. Semantic Knowledge Acquisition Process

In order to integrate different schemas, relationships among access terms in different schemas has to be established. This, however, is hampered by different incompatibilities discussed above. To solve this problem, these conflicts must be resolved. This, in turn, requires collecting and incorporating *missing and/or implied* semantic knowledge for all objects present in different schemas. [17]

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 objects, possibly represented differently, is greatly simplified. [1]

2.3.2.2. Linguistic Concepts

It is possible to identify the semantic relationships among terms using linguistic theories. 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 objects. Basically, these objects 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. [3]

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. [3]

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. [3]

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. [3]

CHAPTER THREE

ISSUES ON SEMANTICALLY SIMILAR OBJECTS

3.1. Automatic Identification of Semantically Similar Objects

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 is 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. The summary schemas model (SSM) offers many advantages discussed in section 3.1.3. [3, 4, 12]

3.1.1. Taxonomy

The SSM requires a general hierarchical taxonomy with disambiguated definition of words where homographs and polysemy are resolved [3]. 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 links. 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 it if necessary. [3]

It is necessary to distinguish between two types of taxonomies: full taxonomy and operational taxonomy. The full taxonomy contains the following information: terms, their disambiguated definition and the semantic relationship among definitions. Each entry in the full taxonomy has the following "fields": [3]

- Term name
- Term definition
- A pointer to a hypernym (not available for the root)
- A list of pointers to hyponyms (not available for leafs)
- A list of pointers to synonyms (not available for non leafs)

The full taxonomy does require a specialized structure. It must be available on-line to aid database designers and users understand precisely the meaning of different terms. It is not required for the automated processing of the SSM. [3]

The operational taxonomy is similar to the full taxonomy except that is contains less data. It requires much less storage when compared to the full taxonomy. It is only

used by the system for its automated processing. Each entry in the operational taxonomy has the following "fields":

- Term ID
- A pointer to a hypernym (not available for the root)
- A list of pointers to hyponyms (not available for leafs)
- A list of pointers to synonyms (not available for non leafs)

Note that the operational taxonomy is one level shorter then the full taxonomy. The leafs of the full taxonomy do not appear in the operational taxonomy. [3]

3.1.2. 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. [3] Figure 6 shows an example of a summary schema.

3.1.3. Merits and Demerits of the SSM

Using the SSM together with a multidatabase language to create a temporary global schema offers many advantages. When compared to the approach that creates a permanent global schema, the following advantages reveal: [3]

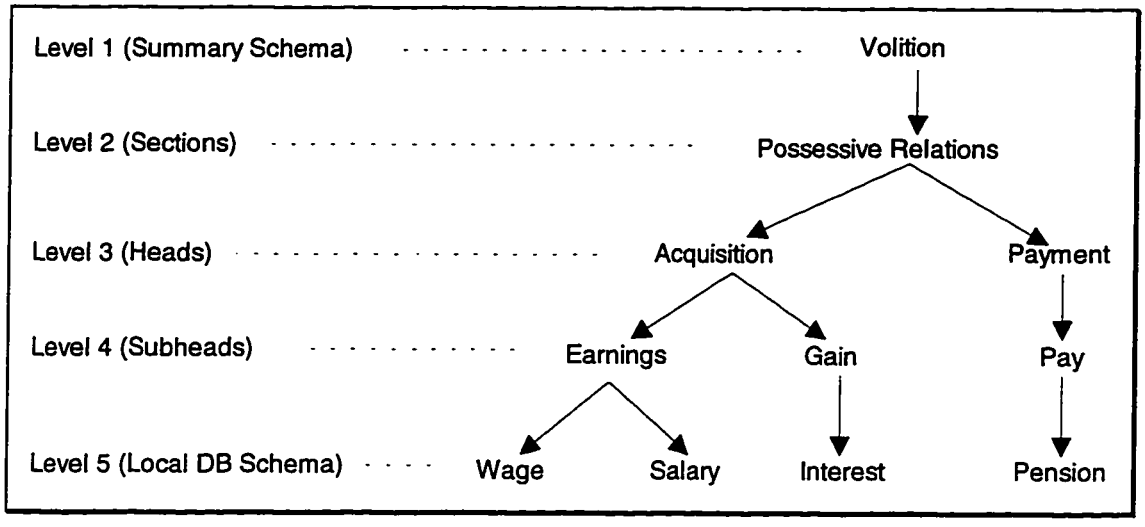

Figure 6: A Sample Summary Schema

- It has a smaller data structure that is easier to create, maintain and store.
- Users can submit queries using terms meaningful to them. They don't have to worry about terms used in other databases.
- The extra processing overhead is relatively small.
- Integration efforts are shared among local DBAs, users and, to a much lesser extent, global DBAs.
- Nodes with small storage capacity and processing power can participate in the global database if this approach is adopted as they don't need much resources.

The major disadvantage of the SSM approach has to do with the interpretation correctness of terms used in user queries and access terms. In fact, this interpretation depends on issues like: precise usage of terms by both designers and users, and the suitability of the dictionaries and thesauruses used. [3]

Initially, local DBAs have to do some minor effort in mapping all local access terms to entry-level (level 5 in Figure 6) terms in the taxonomy. They are given the capability of automatically looking up terms in the full 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. [3]

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. [3]

CHAPTER FOUR

THE INTEGRATION METHODOLOGY

4.1. Overview of the Integration Methodology

This section introduces a **Ta**xonomy **B**ased **Da**tabase **Sch**ema **In**tegrating Methodology (TAB DASHING). The shaded area of Figure 7 shows the scope of this work. The approach combines three main ideas:

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

By putting together these ideas, TAB DASHING automates the major steps that used to be done manually. These steps have been described briefly in section 2.3.1.2. In short, TAB DASHING offers the following advantages:

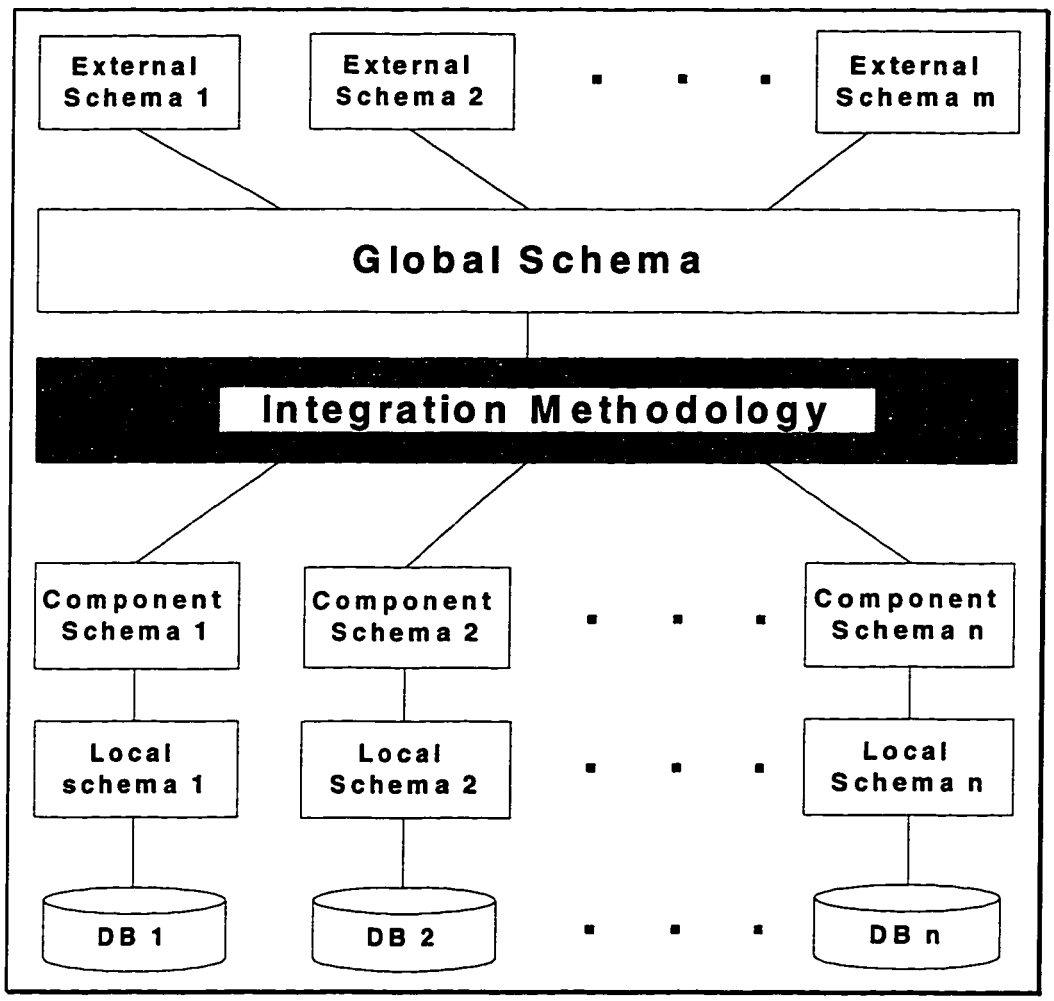

Figure 7: Thesis Work Fills in the Shaded Area

1. The need to derive missing relationships among access terms manually no longer exists. These relations exist naturally in the global taxonomy.
2. Manual derivation of semantic knowledge of global objects is now automatic.
3. Automatic knowledge mapping is achieved.

In section 2.3 above, it is mentioned that local DBAs have to create local object schemas from the corresponding local databases. Local DBAs had to create four different parameters. With the developed methodology, local DBAs have to do much simpler work. They only 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 component schema. This should reduce semantic conflicts 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.

Having that done, the rest of the work is carried out automatically within the taxonomy.

4.2. The Taxonomy: A New Look

The basic building block of the new approach is the existence of an on-line general taxonomy of words. The taxonomy will be used by all DBAs as a pool of their knowledge about their databases. It will serve as a "resolver" of various conflicts 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.

Restricting the domain of the taxonomy by making it "domain specific" will surely give the integrators some relief since the number of terms will be reduced significantly. The problem, however, is that databases frequently contain access terms that are not from the domain. These access terms will cause a problem while integration since they will not be available in a domain specific taxonomy.

The on-line taxonomy described in 3.1.1 will be used. It has to be extended, however, to suite the new approach. The taxonomy is extended to be a hierarchy of classes rather than just a 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 8 shows the general format of an entry (a class) in the taxonomy.

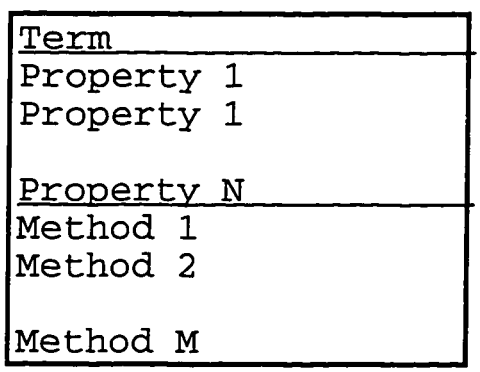

Figure 8: A Class in the Taxonomy

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 represent synonym terms.

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).

4.3. Example I: An Illustrative Example

The way semantic resolution is done and compatibility is achieved are best illustrated by an example. The example is kept simple such that it is easy to understand. Consider a merger of two companies A and B. The executives of the merged companies are interested in the data stored in the Personnel Database of company A (PDBA) and the Personnel Database of company B (PDBB) collectively. The two databases are relational. The two local DBAs are willing to integrate these databases. PDBA has a an entity named "Emp" and PDBB contains an entity named "Staff-Member". Figure 9 illustrates these two objects.

Applying the phases TAB DASHING, the DBAs will be able to identify the following two objects: "Emp" from PDBA and "Staff-Member" from PDBB.

```
Database Name: PDBA
Entity Name: Emp
Attribute 1: EName: Char (30)
Attribute 2: Wage

Database Name: PDBB
Entity Name: Staff-Member
Attribute 1: SName: Char (30)
Attribute 2: Salary
```

Figure 9: Two Sample Relations

Figure 10 shows the parts of the taxonomy significant to this example. Figure 11 shows some classes of the taxonomy hierarchy. The first object, "Emp", can be mapped to "Employee" in the taxonomy. The local DBA of PDBB will be able to map the second object, "Staff-Member", to "Staff" in the taxonomy. Note that if both DBAs have mapped the two terms to "Employee", for example, the problem would be simpler.

Depending on the semantic distance metric (SDM) described in section 2.3.2.2, the degree of semantic similarity between these two objects can be determined. The SDM between "Employee" and "Staff" is low indicating their similarities (see section 2.4.2). Actually, these two objects are synonyms according to the taxonomy and their SDM has a value of one, the lowest possible value. Moreover, the synonym link between the two objects indicates an equivalence relation between them. Hence, their properties (attributes) have to be equivalent, or at least compatible.

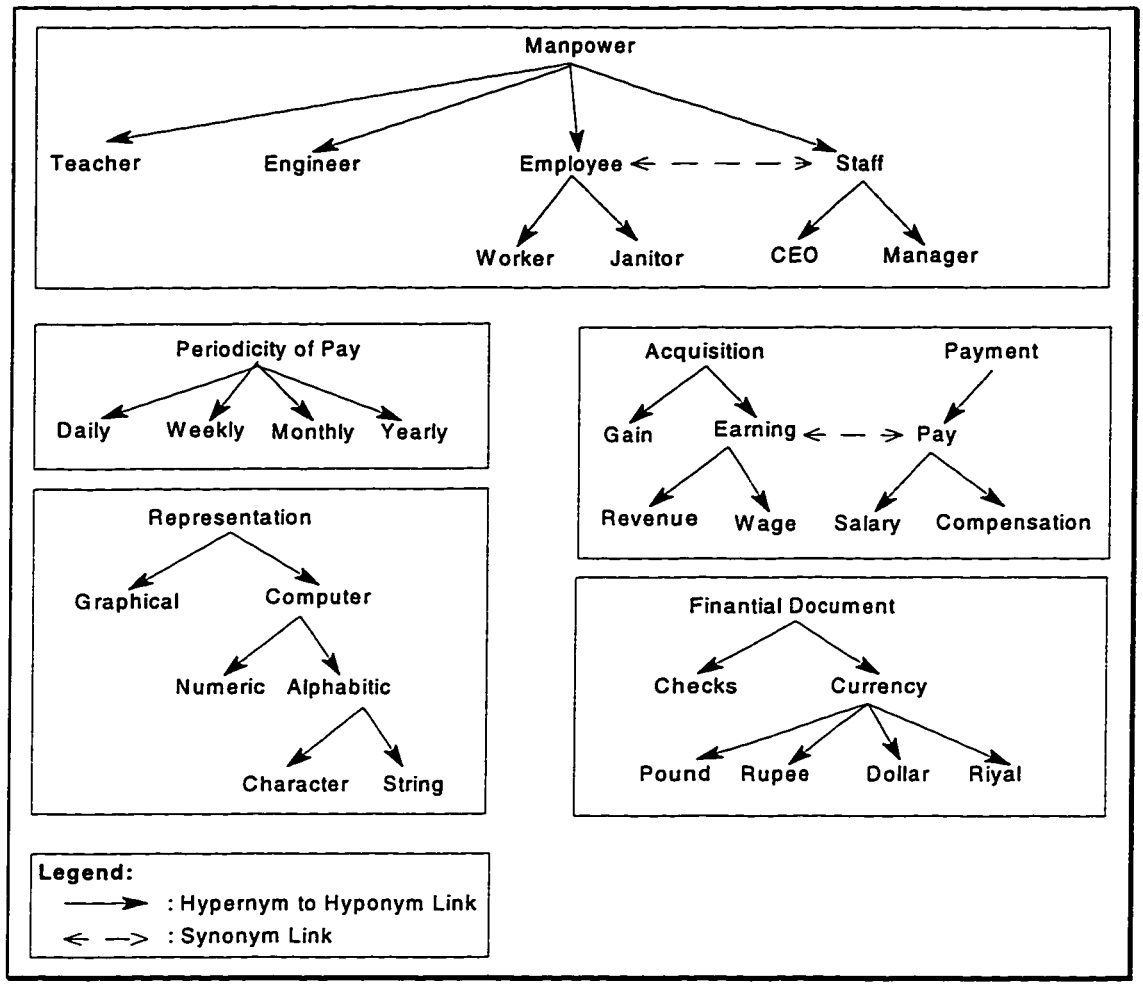

Figure 10: Parts of the taxonomy for example I

Employee
Name: *Pointer to Name class*
Wage: *Pointer to Wage class*

Staff
Name: *Pointer to Name class*
Salary: *Pointer to Salary class*

Wage
Value
Currency: *Pointer to Currency class*
Periodicity-of-pay: Pointer to ...
Method: Wage-to-Salary;

Salary
Value
Currency: *Pointer to Currency class*
Periodicity-of-pay: Pointer to ...

Figure 11: Sample Classes of the Taxonomy Hierarchy

In the example at hand, one property of "Emp" (namely, "EName") is equivalent to its corresponding property of "Staff-Member" ("SName"). The second property, on the other hand, of "Emp" ("Wage") is only compatible with "Salary", the second property of "Staff-Member".

The equivalence of "EName" and "SName" can be easily determined from their mapping. Both of them are mapped to Name in the taxonomy which means that they have the same properties (i.e. meta-properties). That is not enough, however, to guarantee their equivalence. As mentioned before, the values of these meta-properties have to be equal in order to guarantee the equivalence of the properties. In fact, this is the case with "EName" and "SName". To further illustrate, one meta-property of the property Name is its computer representation. Both "EName" and "SName" are represented as characters of length 30 each.

Another issue of concern in this example is the compatibility of the two properties "Wage" and "Salary". This compatibility can be determined from the taxonomy. In this example, it is assumed that wages are paid in terms of Riyals (the meta-property Currency) per day (the meta-property Periodicity-of-pay). Salaries, on the other hand, are paid in terms of Riyals per month. Here, the meta-property Periodicity-of-pay has two different values.

Compatibility between "Wage" and "Salary" can be achieved via methods defined in the corresponding classes in the taxonomy hierarchy. A method in the "Wage" class, for example, will multiply a wage by 30 to get the salary per month.

Figure 12 shows a method for this purpose. The method is defined in the class "Wage". It acts on the attribute Value of the "Wage" class. Whenever this method is called, the required calculations are performed and the computed result is returned.

```
Method: Wage-to-Salary;
     RETURN Wage.Value × 30
end;
```

Figure 12: A Sample Method for Achieving Compatibility

4.4. TAB DASHING Details

In this section, TAB DASHING will be described in detail. In order to make the methodology easy to follow and simple to apply, it is divided into several phases. "Like every other methodology, we do not present an algorithm in the sense of stepwise action rather as a way of thinking", says Eder and Frank [7].Only phases 1 and 2 may be curried out in parallel. The rest of the phases have to be applied in sequence.

4.4.1. Phase 1: Selecting a Taxonomy

The importance of using a lexicon taxonomy in the methodology is that it captures various semantic relationships among the words people use to express ideas. In earlier methodologies, researchers started by trying to build these semantic relationships among access terms (words) used in local schemas, for example [10, 17], for the sake of identifying semantically similar access terms and resolving conflicts among them. Researchers who developed these methodologies assumed the existence of a human

integrator (mostly the global DBA) who can find (or at least confirm) these relationships and solve the conflicts. The problems of such approach were discussed in section 2.2.1.2.

To simplify the process, the idea of using a general lexicon taxonomy was introduced in the context of developing the SSM (see section 3.1 for a brief discussion) [3, 4, 12]. Building on the same idea, I suggest using the taxonomy hierarchy as the basic building block of TAB DASHING.

It is suggested in [3] to select an existing taxonomy (or a combination of existing ones) rather than building one from scratch. Section 3.1.1 describes the taxonomy. Section 4.2 shows how it looks in this research. Building a taxonomy from scratch requires a significant effort. Moreover, the existing taxonomies are suitable and sufficient for the purposes of TAB DASHING. Furthermore, definitions and semantic relationships in existing taxonomies are tested and verified for its usefulness as opposed to a newly built one.[3]

Customizing and/or augmenting the selected taxonomy is not out of question. As a matter of fact, if only some of the features required are met in the taxonomy, the global DBA or even some/all local DBAs should consider augmenting the most suitable taxonomy.

Selecting a taxonomy can be done by the global DBA alone or by some/all local DBAs together with the global DBA. The main features they should look for while selecting one are listed below: [3]

- A general lexicon taxonomy

- The entries in the taxonomy should be disambiguated
- The hypernyms hierarchy should be simple
- Hypernyms should be semantically intuitive
- The taxonomy should have a limited number of synonym cross-references

4.4.2. *Phase 2: Translating Local Schemas to CDM Component Schemas*

Integrating local schemas in their original (potentially heterogeneous) data models is a difficult task. It becomes easier if the problem is reduced to integrating schemas that share a common data model (CDM). Reasons making it necessary to adopt a CDM are mentioned in section 2.3.1.1. To achieve this, each local schema is translated to an equivalent schema represented in the CDM. [6]

Since each local DBA is assumed to be knowledgeable of the database s(he) administrates, s(he) would be the best person to carry out the translation. Help of the on-line taxonomy can be sought in the process of translation. The on-line taxonomy is a browsable version of the taxonomy hierarchy that provides the DBAs with the precise meaning of each term in the hierarchy. It is suggested (but not necessary) to use the terms found in the taxonomy instead of the original terms of the local schemas. This should simplify carrying out the next phase. This situation is much the same as a person trying to translate an article to an equivalent article in another language. He would seek the help of a proper dictionary and/or thesauruses, and use the words s(he) discovers.

As an example, suppose there is an access term "Emp" in some local schema. The local DBA, by assumption, well understands the meaning of "Emp". Now, the local DBA wants to translate the local schema to a corresponding component schema. Of course, s(he) can still use the same access term "Emp" or any other term. If a proper term is used, however, the next phase will be simpler to carry out. To select the proper term s(he) would want to know which term in the taxonomy is most suitable. For that s(he) will need to browse the taxonomy and select the term that has the best definition. The local DBA might find terms like "Employee" and "Staff" prime candidates. Chung and Mah [6] mention a few guidelines for translating schemas in relational, network, hierarchical and object oriented models to the unified model, an object oriented extension of the relational model.[6]

4.4.2.1. The CDM Selected for Use with the Methodology

Two main features DBAs should look for when selecting a CDM are, first, it should be *rich enough* in semantics such that it can model the semantics of the data models of various local schemas. In other words, the CDM should subsume other data models; second, it should be *simple enough* such that creation and maintenance of the global schema is easy. [6, 11, 15] Rich data models are ones with more data modeling constructs, properties and constraints while simple data models are ones with fewer such constructs [10]. The two features, hence, are contradictory. The object oriented data model, however, seems to be accepted as one that satisfies both features. More details about main concepts and issues of the model may be found in Bertino and Martino [2].

Some researches proposed to use the object oriented data model or an extension of it (see [5] for an example) as the CDM. Others suggested extending the relational model with object oriented features [6]. It seems that object orientation offers the necessary features needed for this kind of research. Therefore, we will consider the object oriented data model (or any of its extensions) as a suitable CDM for schemas to be integrated by TAB DASHING.

4.4.2.2. Translation from Different Data Models to CDM

Local databases are potentially modeled using different data models. Some of the widely used data models are the relational, network, hierarchical and object-oriented data models. In order to translate local schemas modeled using a data model different from the CDM, it is necessary to follow some kind of translation guidelines. These guidelines when applied to a schema in some data model (source data model) should result in an equivalent schema in the target data model (the CDM.)

Fortunately, earlier research has produced such guidelines. Chung and Mah [6] present guidelines for converting relational, network, hierarchical and object-oriented data models to the unified data model, a relational data model with object-oriented features.

Chung and Mah [6] claim that the unified model and the object-oriented model are essentially the same except that the object-oriented model has more features. They also claim that converting the extra features to the available ones in the unified data model is "quite straightforward." Hence, it is quite reasonable to accept the given

guidelines as valid ones for translating local schemas with their data model being relational, network, hierarchical or object-oriented to the CDM (the object-oriented data model.)

Guidelines for translating other data models to the object-oriented data model can be developed if not already available. This kink of work, however, is beyond the scope of this research.

4.4.3. Phase 3: Mapping Access Terms

This phase is also carried out by each local DBA. Each local DBA has to, first, identify all objects in the translated schema, the component schema. Each of these objects is represented by an access term in the component schema (see section 3.1). As mentioned in phase 2, each local DBA is assumed to be well knowledgeable of the database s(he) administrates.

In the second step, each local DBA has to determine, precisely, which taxonomy term matches the meaning of each access term. This task is much simplified by the availability of the on-line taxonomy that can be referred to even before identifying access terms (see phase 2 above) in component schemas.

The last step is now straightforward. Each access term is mapped to the corresponding taxonomy term. This is made possible by making the taxonomy structure globally available. That is, it can be accessed by all local DBAs as well as the global DBA. When an access term is mapped to a taxonomy term, the taxonomy

term should be the most specific term in the taxonomy. That is, the term that lies as deep as possible in the taxonomy structure.

To illustrate, assume that one of the objects in the component schema is named "Emp". The first step the local DBA should do is to identify this object (as well as others). As the meaning of each access term ("Emp" in this example) is well understood by the local DBA, the second step aims at finding the taxonomy term whose meaning corresponds to the meaning of "Emp". Let us assume that the taxonomy term "Employee" is found to be the best corresponding term in the taxonomy. In the last step, the access term "Emp" is mapped to the taxonomy term "Employee".

Term Matching

Terms in component schemas are mapped to the taxonomy terms on the basis of there semantics. If the semantics of a component schema term matches the semantics of a taxonomy term then the component schema term should be mapped to this taxonomy term.

One way to compare the semantics of two terms is to first consider the definition of the two terms (the one the local DBA knows versus the one in the taxonomy). Second, the properties of the terms are compared against each other. An ideal situation will occur when the definitions are matching and the properties are identical in number and semantics. The ideal situation is not very likely in most cases. A more realistic situation would be one in which definitions and only some properties are matching. In this case, terms having more properties in common are ones with

closer meaning. Resolving the conflicts among properties with different semantics can be considered (see section 2.2.3.3).

This matching method might not be the best to apply. It is sufficient, however, for the purposes of developing TAB DASHING. Better matching techniques can be considered. If one is found, it would add to the power of TAB DASHING.

4.4.4. Phase 4: Augmenting Semantic Knowledge

Section 2.3.2.1 addresses the issue of explicitly storing semantic knowledge missing or implied about each object. The collected knowledge will be stored in the taxonomy term corresponding to the access term representing the object.

Automating the process of semantic knowledge acquisition was also addressed. As each local DBA maps an access term to a taxonomy term, s(he) can be prompted to fill in the missing and/or implied semantics. As the semantics are captured via instance variables of each taxonomy term (a class), it is obvious that valueless instance variables are the ones with missing semantics. Local DBAs will be prompted to fill in the information for these variables.

4.4.5. Phase 5: Determining Equivalent Objects and Creating the Global Schema

In a single local schema, there will be no equivalent objects. Otherwise, the minimality condition of database design is violated. Equivalent objects might exist, however, among component schemas. As mentioned previously, it is essential to determine such objects to achieve integration (see section 3.1). This task is quite

involved and difficult in other methodologies. In this methodology, however, the global DBA has a much easier and straightforward job.

If two access terms were mapped (in phase 3) to the same taxonomy term, then it is obvious that the corresponding objects belong to one equivalent set of objects. Assume, for example, that the access terms "Emp" and "Staff-Member" from databases A and B respectively were both mapped to the taxonomy term "Employee". The global DBA can safely say that "Emp" and "Staff-Member" are equivalent objects.

Similarly, if access terms were mapped to different taxonomy terms that share synonym links among them, then the corresponding objects belong to one equivalent set of objects. Assume that "Staff-Member" in the previous example was mapped to the taxonomy term "Staff" that shares a synonym link with "Employee" (see Figure 10). In this case too, the global DBA can safely declare the equivalence between "Emp" and "Staff-Member". In any other case, the global DBA may assume that the objects are not semantically similar.

For the sake of constructing the global schema, the global DBA has to select a representative access term from the taxonomy for each set of equivalent access terms. In the case where all access terms are mapped to one taxonomy term, the global DBA may only select that taxonomy term as an access term in the global schema. If access terms were mapped to synonymous taxonomy terms, the global DBA may select any of these synonyms as an access term in the global schema. The resulting global schema is an object oriented schema. In the previous example, if the two access terms

were mapped to "Employee", "Employee" is the term that will be selected as the access term in the global schema that corresponds to "Emp" and "Staff-Member". On the other hand, where "Emp" and "Staff-Member" are mapped to "Employee" and "Staff" respectively, either "Employee" or "Staff" may be selected as the corresponding access term in the global schema.

For unique access terms (ones that have no equivalents), the corresponding taxonomy term may be selected as a representative access term in the global schema. Assume, for example, a unique access term "Eng" was mapped to "Engineer". Then, "Engineer" will be used as the representative access term in the global schema.

Up to this point the need for the on-line taxonomy no longer exists. The global schema has just been created and only one more phase, described next, remains.

4.4.6. Phase 6: Defining Compatibility Methods

Section 2.2.3.3 describes different kinds of conflicts that may exist between two access terms from two different component schemas. The way these conflicts are resolved in this methodology is via compatibility methods. A compatibility method is simply a method defined in a taxonomy term (class) that is compatible, but conflicting with, another taxonomy term. A compatibility method is needed for each different conflict between two terms. Compatibility methods will reside in various classes of the global schema created in phase 5.

According to the way synonyms are perceived in this research, synonymous terms refer to semantically similar objects. That is these synonyms should be

compatible. Synonyms, however, might have conflicts among them. Therefore, compatibility methods will have to be defined in the taxonomy term that need to be made compatible with another synonymous taxonomy term. We call this kind of compatibility methods *Horizontal Compatibility Methods* (HCM).

Another type of compatibility methods is the *Vertical Compatibility Methods* (VCM). VCMs are defined to resolve conflicts between two taxonomy terms one is a super class of the other. This situation will arise in cases where one of the properties of the super class is redefined in a subclass and the property is given the same name. Redefining a property this way is never needed unless there is a semantic difference (semantic conflict) between the super class property and the subclass property.

Basically, there is no difference between HCMs and VCMs except that the former apply to synonymous terms while the later apply to hypernym/hyponym terms. Examples of HCMs and VCMs may be found in sections 4.3 and 4.5.

Compatibility methods will have to be invoked only when data retrieval takes place. In this case, it is possible to have these methods invoked automatically. Example II in section 4.5 illustrates the idea.

Compatibility methods can either be defined by the global DBA or by individual local DBAs. In either case, however, definition cannot be done before phase 5 is complete and global schema access terms are known.

4.5. Example II: A Complete Example

To illustrate the power of TAB DASHING and to facilitate its analysis, we take the very same schemas, schema objects and properties presented in Reddy et al. [17] and apply TAB DASHING to it. We will be able to get a result equivalent of that in [17]. To simplify building the example, schemas, objects, and object properties and meta-properties found in [17] 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 [17]:

{Emp, Employee}

{Dept, Division}

{Visiting-Prof, V-Prof}

Figure 13 shows the super class/ subclass information found in [17]:

Phase 1 of TAB DASHING aims at selecting a taxonomy. Figure 14 shows the parts of the selected taxonomy significant to this example.

In phase 2, translation of local schemas to CDM component schemas takes place. The result of translating schema 1, schema 2 and schema 3 to the CDM (Object Oriented Data Model in TAB DASHING) is shown in Figure 15.

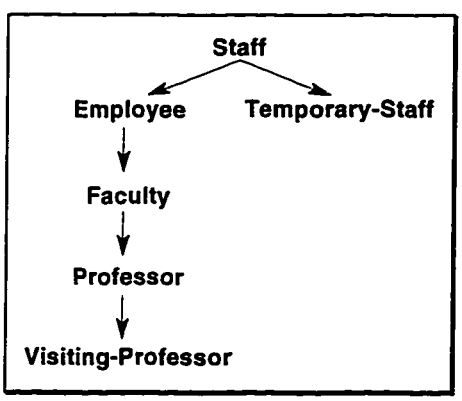

Figure 13: Super class/Subclass Information in Reddy et al.

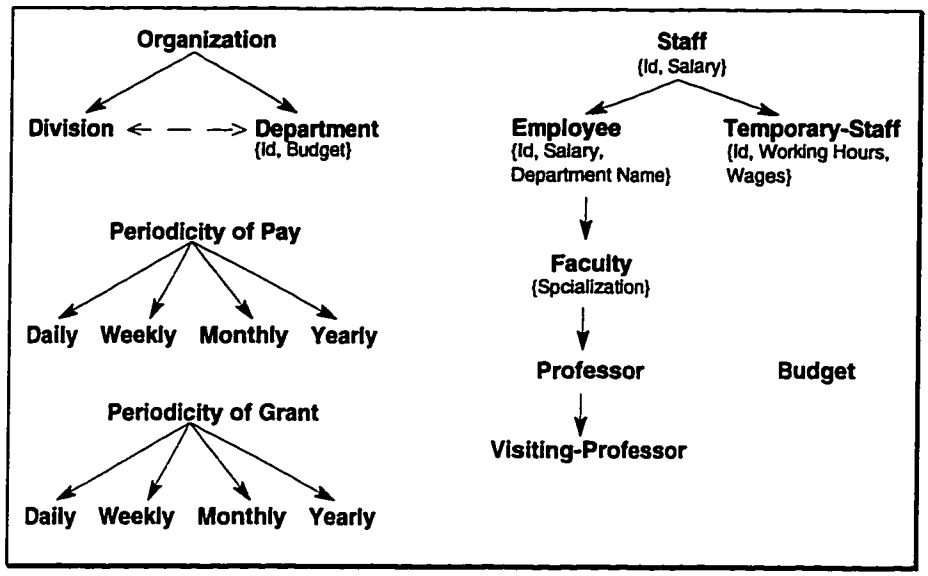

Figure 14: Parts of the Taxonomy for Example II

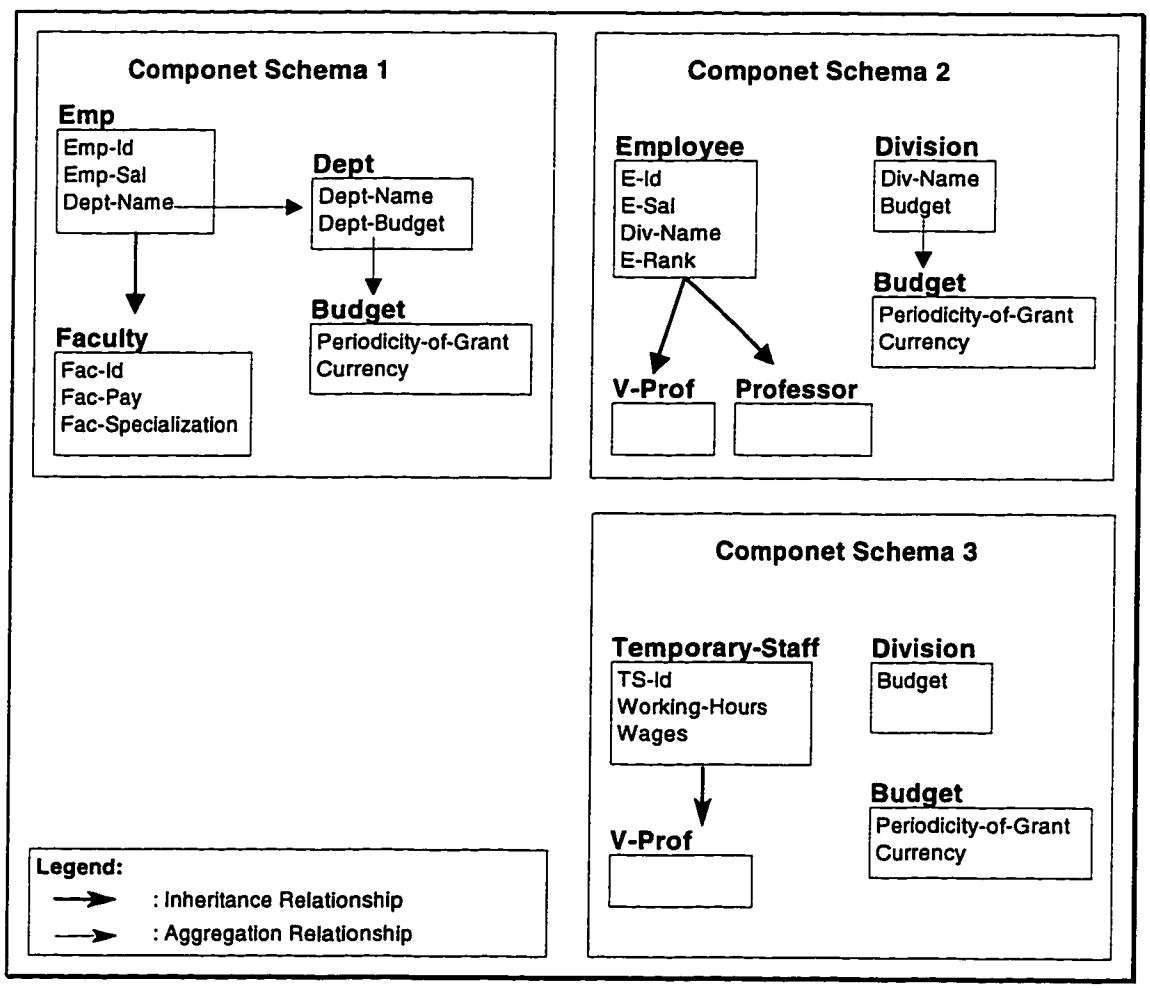

Figure 15: Translated Schemas

In phase 3, local DBAs 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"

In phase 4, missing or implied semantic knowledge is 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".

In phase 5, the global DBA determines 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.

In phase 6, 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 have to be defined in "Division" class. The method has to 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 16 shows an outline of the method.

The method shown in Figure 16 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 14) a VCM 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 17. In Figure 17, "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 x PoG x Rate
end;
```

Figure 16: HCM Method Resolving the Budget Conflict

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

Figure 17: VCM Method Converting Employee Salary to Faculty Salary

4.6. Methodology Analysis

This methodology is developed to overcome a few problems and achieve some objectives (see section 1.2). The problems were found in methodologies developed earlier and the objectives were determined to avoid them and offer new features. In this section, we compare TAB DASHING to earlier ones, and highlight the new features and extra overhead.

The first (and main) feature of TAB DASHING is the availability of relationships among access terms in a pre selected taxonomy. The need to build these relationships is no longer present. Earlier methodologies try to build these relationships every time from scratch depending on the existence of a human to find out these relationships and resolve possible conflicts. Some earlier researchers went a bit further [3], where help of an existing taxonomy is sought to build these relationships.

The second feature is the ability to automatically identify semantically similar objects. Thanks to synonym links in the taxonomy. In earlier methodologies, a human (usually the global DBA) had to find out himself different sets of equivalent objects. Any conflict arose, had to be resolved manually. Conflicts are resolved automatically in TAB DASHING via compatibility methods. Reddy et al. [17] resolved the conflicts via transformation maps.

Automatic acquisition of semantic knowledge is the third feature. Previously, both local and global DBAs had to provide missing and/or implied semantic knowledge of objects. In TAB DASHING, there is no need to do this since this

knowledge is already present. In some cases a need might arise to augment the taxonomy with some more information. Augmentation, if takes place, will have to be done only once. Moreover, there should not be too much missing information from the taxonomy.

As TAB DASHING adopts object oriented concepts, it was necessary to generalize the classification of schema conflicts to suit the methodology (see section 2.2.3.3). Generalizing the classification of schema conflicts can be considered as a "by product" of developing the methodology.

The methodology is much less human labor intensive. Compare the amount and nature of work required from the DBAs using this methodology to that required by users of the methodology developed in [17], for example.

The methodology requires minimal knowledge by the global DBA of different local schemas. This is a result of distributing the integration tasks over local DBAs and the global DBA giving each a fair share of work and reducing the responsibility domain of the global DBA. In earlier methodologies, the global DBA is overwhelmed by doing most (if not all) of the integration tasks with no (or very little) local DBA assistance.

Earlier methodologies produce very rigged global schemas that are difficult to create, in the first place, and hard to modify (maintain) afterwards. This methodology, on the other hand, makes it easy to create and maintain the global schema.

Easy maintenance of the global schema is facilitated by a very important "by product" of this approach is its dynamic nature. At any moment, a new local schema

can join the federation without disturbing any of the global users. In addition, local schemas already participating in the federation can be dynamically modified (maintained). Thanks to the possibility of using the taxonomy dynamically.

The price to achieve this easiness and simplicity of the methodology is paid in terms of extra overhead. First, the taxonomy should contain more information when compared to the taxonomy used in the SSM (see section 3.1). Second, compatibility methods have to be developed to resolve various conflicts. The number of compatibility methods that have to be developed is a factor of the number of conflicts among objects in component schemas being integrated. Finally, the number of aggregation relationships [2] is potentially large since any property of any object could itself be an object.

4.7. General Implementation Guidelines

In order to make use of the methodology in actual integration problems, it has to be realized as a computerized system. That is, the methodology has to be implemented. Implementation of the methodology is neither trivial nor impossible! It can be considered as a medium/large size project. As usual, the best way to handle projects of this size is to *divide* it into several implementation phases and *conquer* each one independently.

Dividing the methodology into six phases gives a clue for dividing the project. The project can be broken into six implementation phases each of which is concerned

with providing the necessary support for carrying out the corresponding phase of the methodology. Literally speaking, the following are the six implementation phases:

Implementation Phase 1

This implementation phase should take care of making the taxonomy available for the DBAs. Basically, there are two options: either build the taxonomy from scratch or take one off-shelf. Section 4.4.1 discusses the advantages and disadvantages of each option.

Selecting a taxonomy off-shelf might require some augmentation/customization. Main augmentation/customization activities include: transforming each term to a class (section 4.2), defining different synonym links and adding missing/implied properties of each term.

Implementation Phase 2

Local schemas not in the CDM have to be translated to the CDM. Since the CDM used in this research is the object-oriented data model, the concern here is for translation from different data models to the object-oriented data model. The literature presents guidelines and techniques to carry out such translations (see [6] for example). This implementation phase is concerned with realizing these guidelines in the computerized system such that the users will be guided by the system to do the translation.

As a start, it is sufficient to consider the most commonly used data models: the relational, network and hierarchical data models. Other data models can be included

later on. Obviously, the more data models supported by the system, the stronger the system is.

Since translating local schemas may be considered a pre-integration activity, an alternative approach can be followed. The implementation phase can be restricted only to accept component schemas as input to the system. That is, the translation of local schemas will be done by the local DBAs *some how*, and the result of that will be input to the system. The system, in this case, has to have the ability to recognize different input elements and reconstruct component schemas internally.

Implementation Phase 3

This implementation phase is concerned with extracting various access terms from each component schema and then defining a correspondence (mapping) between each access term and a term in the taxonomy. This implementation phase simply provides an interface for each local DBA. It facilitates browsing of the taxonomy terms in order to find the proper taxonomy term that corresponds to each access term then define this correspondence.

Implementation Phase 4

The issue of missing/implied semantic knowledge is addressed in section 2.3.2.1. The missing semantics can be realized as properties without values. In this implementation phase, all such properties will have to be presented to the integrators so that they can augment the missing semantics.

Automating this process is straightforward. While each access term is mapped (in implementation phase 3) to the proper taxonomy term, the integrators can be prompted to fill the missing semantics. This kind of interaction between implementation phases 3 and 4 has to be considered prior to carrying out any of them; otherwise, it might become difficult (or even impossible) to combine both of them after completion.

Implementation Phase 5

As mentioned in section 4.4.5 equivalent terms are those mapped to the same taxonomy term or those mapped to different taxonomy terms that share synonym links among them. This implementation phase is concerned with finding out these terms. These terms represent equivalent objects in the global system. Creating the global schema and presenting it to the DBAs is another task in this implementation phase.

Implementation Phase 6

In order to define HCMs, synonymous terms that have access terms mapped to them have to be first identified. Proper HCMs can then be defined. VCMs are needed in a different situation. Wherever a property of a super class is redefined in a subclass and the property is given the same name, then there is a need to define a VCM. In this case, it is necessary to build a technique to find such properties and inform the integrators about them.

It should be the responsibility of the system to provide the integrators with the proper tools (text editor, compiler, ...etc.) to write the methods. Then, it should place the method in the proper class.

Some of these implementation phases are sufficiently large such that they can be further divided into rather smaller phases. For example, implementation phases 2, 5 and 6 seem to be too large to be handled without dividing them further.

CHAPTER FIVE

CASE STUDY: INTEGRATION OF TWO STUDENT DATABASE SCHEMAS

In this case study, two student database schemas will be integrated using TAB DASHING. The schemas are used in two different schools to keep track of necessary information about the students of each school. The two (local) schemas are relational and are shown in Figure 18 and Figure 19 respectively.

Students (Student Id, Parent Id, Name, Grade, SchCode, Joining Date, Birth Date, Admission-Status)

SchoolCode (SchCode, Description)

Parent (Parent Id, Name, NatCode, POBox, Postal Code, OPhone, HPhone, Notes)

TNationality (NatCode, Nationality, AbbrNat, Country, AbbrCountry)

GradeCode (Grade, Description)

Figure 18: Local Schema 1

Student (Parent-Id, Join-Date, Id, Name, B-Loc, B-Date, Nation, Sex, Religion, Lang, Curr-Sch, Last-Sch, Parent, Relation, Job, Job-Loc, J-Phone, H-Phone, Address, Level, Year, Class, Status)

School (Id, Name, Sex, Lang, Address, Manager, Vice-Man, Tel)

Sex (Id, Name)

Level (Id, Name)

Lang (Id, Name)

Year (Id, Name)

Nation (Num, Nation)

Figure 19: Local Schema 2

Phase 1 of the methodology suggests selecting a taxonomy. The taxonomy that will be used for this case study is "Roget's Thesaurus" [9]. Figure 20 shows the necessary parts of the taxonomy that will be used during integration.

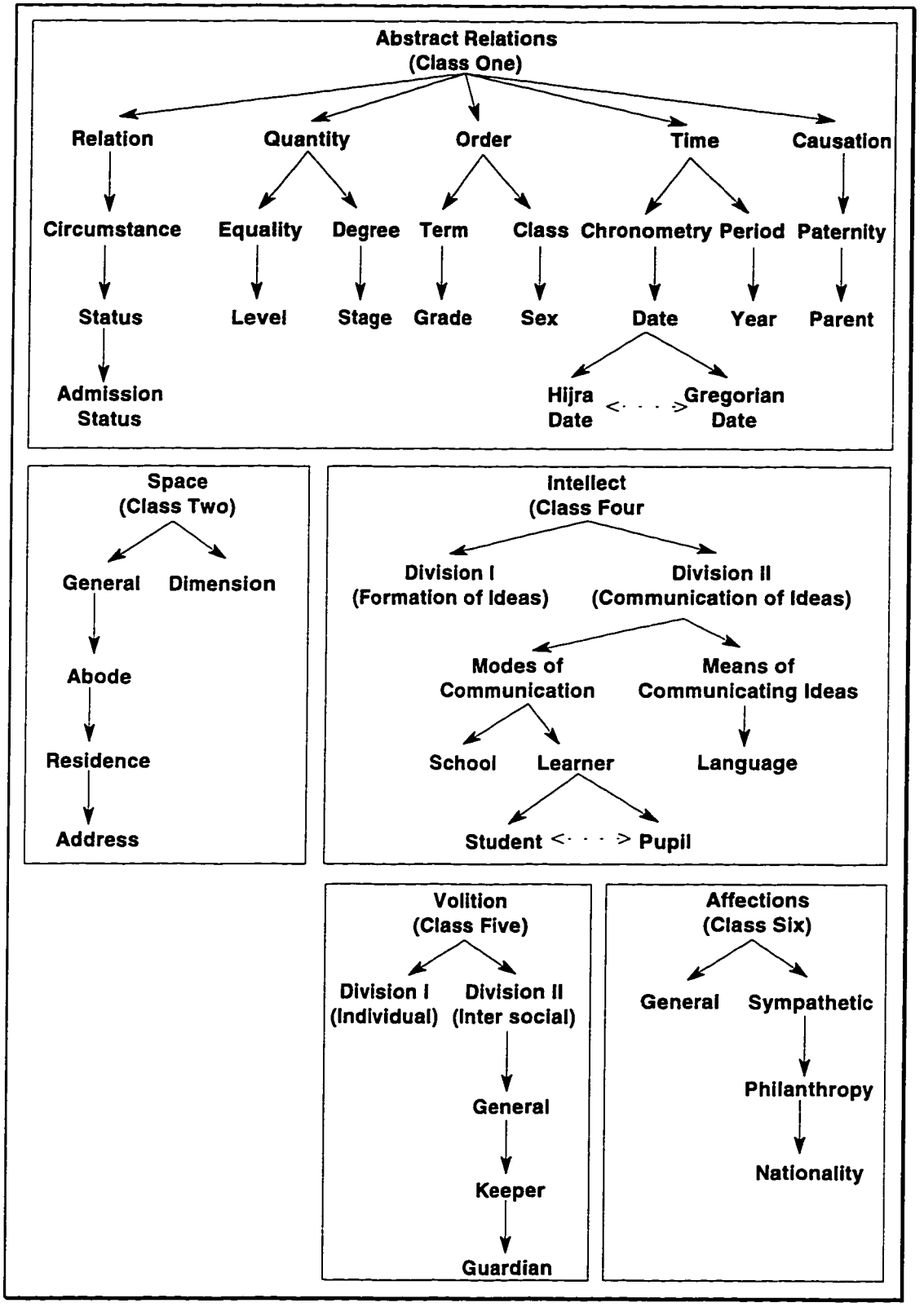

Figure 20: Parts of Roget's Taxonomy

Figure 20 shows the result of translating the first local schema to the CDM, "Component Schema 1". Looking at "Component Schema 1" it can be seen that the local DBA has taken the advice of using taxonomy terms instead of the original terms found in the local schema. For example, he used "Nationality" instead of "TNationality". Figure 22, on the other hand, shows that the local DBA of the second database preferred to use the same terms of his local schema.

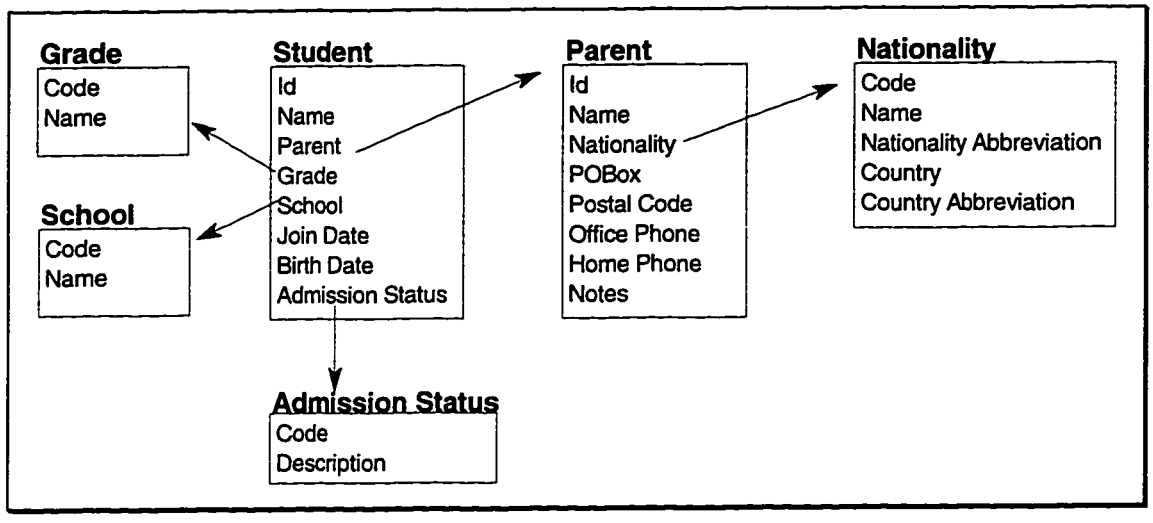

Figure 21: The CDM of Local Schema 1

In phase 3, the two local DBAs have to identify component schema access terms and map them to the proper taxonomy terms. Figure 23 shows the access terms of the two component schemas and the taxonomy terms to which they are mapped.

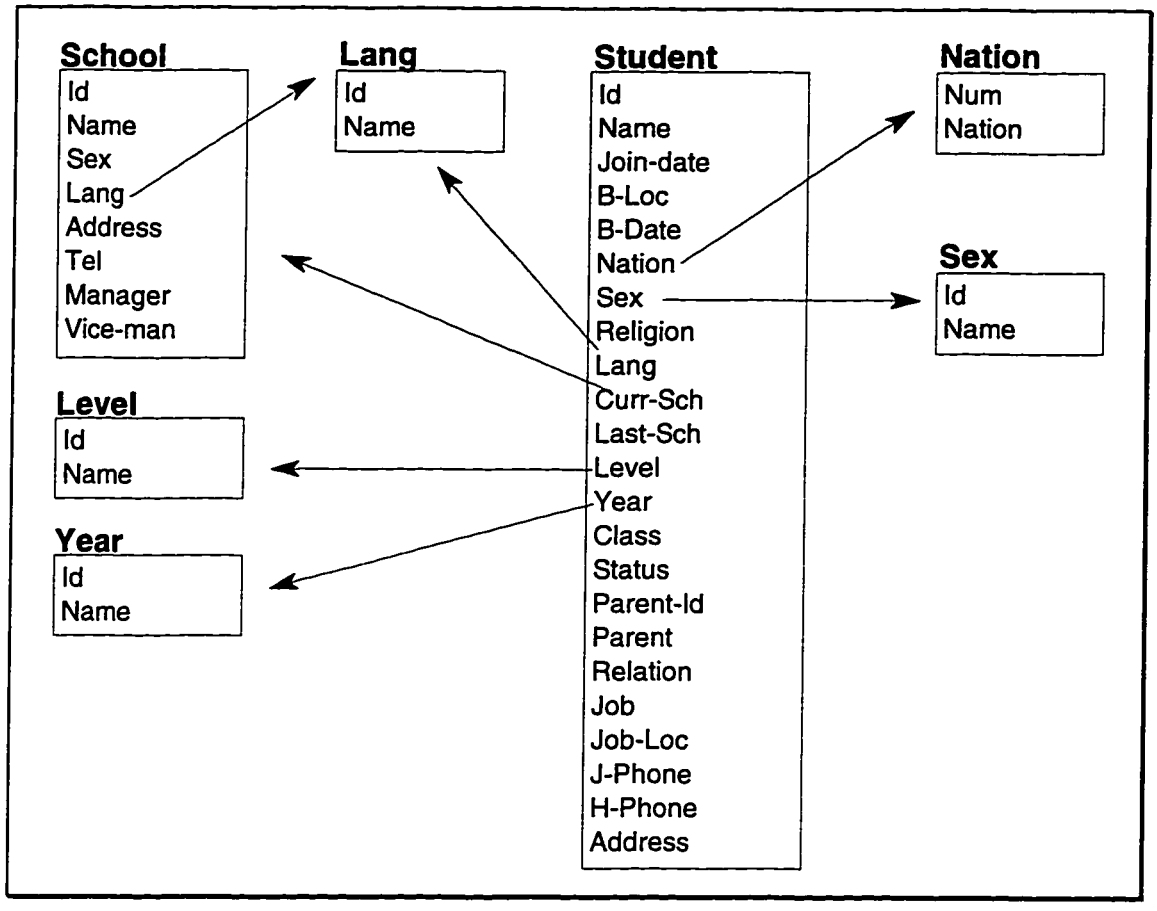

Figure 22: The CDM of Local Schema 2

Component Schema 1	
Access Term	**Taxonomy Term**
Student	Student
Parent	Guardian
School	School
Nationality	Nationality
Grade	Grade
Admission Status	Admission Status

Component Schema 2	
Access Term	**Taxonomy Term**
Student	Pupil
School	School
Nation	Nationality
Level	Stage
Year	Year
Sex	Sex
Lang	Language

Figure 23: Access Term Mapping

Comparing the properties of the objects of the component schemas against those of the taxonomy terms, we find that some properties are missing from some objects. Figure 24 shows the taxonomy terms and their properties that are needed through out this case study. For example, "Component Schema 1.School" has only two properties: "Code" and "Name". The taxonomy term "School" contains additional properties like "Phone" and "Address".

Some semantic knowledge is not explicitly stored in "Component Schema 1". This knowledge has to appear explicitly in order to carry out the integration. For example, "Component Schema 1.Grade" should have two more properties explicitly stored. These properties are: "Stage" and "Year". "Stage" represents the degree the student is seeking (e.g. Elementary, Intermediate, High) while "Year" represents the year he is attending (First, Second, ...etc.).

Starting from phase 5, the global DBA takes over. In phase 5, equivalent objects will be determined. Figure 25 shows equivalent objects in the two component schemas, the representative access term in the global schema and the reason the objects were found to be equivalent.

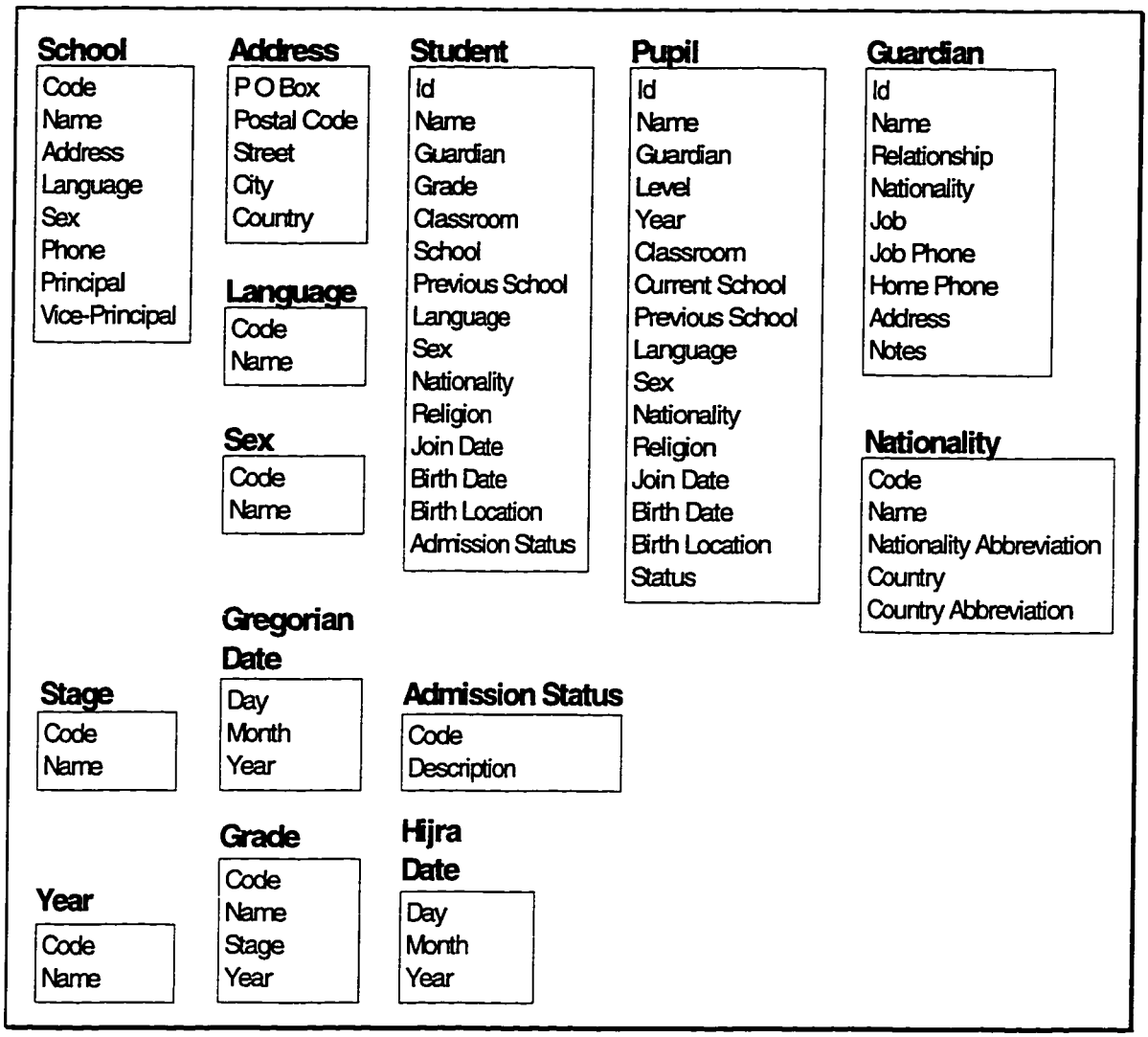

Figure 24: Class Properties

Component Schema 1	Component Schema 2	Global Schema Access Term	Reason of Equivalence (1* or 2**)
Student	Pupil	Student	1
School	School	School	2
Nationality	Nationality	Nationality	2

1*: Component schema terms are mapped to synonymous taxonomy terms
2**: Component schema terms are mapped to the same taxonomy term

Figure 25: Equivalent Objects in Component Schemas

Access terms in component schemas that have no equivalents are represented in the global schema by the same taxonomy terms to which they were mapped. In "Component Schema 1", "Guardian" and "Admission Status" are the only access terms that have no equivalents. In "Component Schema 2", "Year", "Sex" and "Language" have no equivalents too. The global schema is shown in Figure 26.

Defining compatibility methods, phase 6, is needed to achieve compatibility between two pairs of objects: between "Student" and "Pupil", and between "Hijra Date" and "Gregorian Date". The properties of "Student", "Pupil", "Gregorian Date" and "Hijra Date" (after augmenting possible missing/implied semantics) are shown in Figure 24. In Figure 27, two HCMs are shown. The methods will be used to achieve compatibility between the two object pairs. The methods will be defined in "Student" and "Gregorian Date" classes of the global schema described earlier.

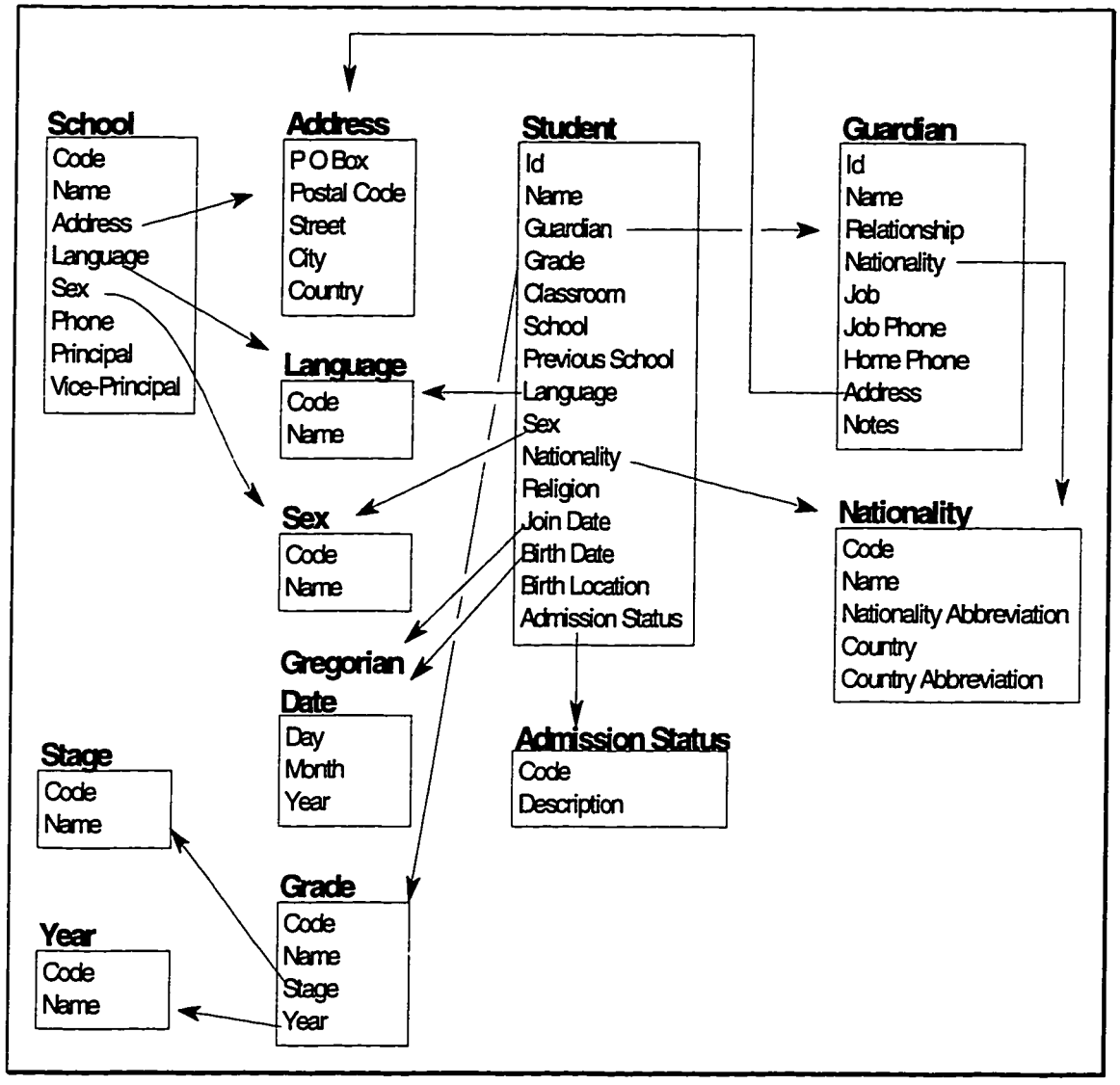

Figure 26: The Global Schema

```
Method: HijraDate-to-GregorianDate();
     RETURN     f([Gregorian     Date].Day,     [Gregorian
Date].Month,
                [Gregorian Date].Year)
     /* f: a function that curries out the transformation
*/
end;


Method: Pupil-to-Student();
     Studnet.Id = Pupil.Id
     Studnet.Name = Pupil.Name
     Studnet.Custodian = Pupil.Custodian
     Studnet.Grade.Level = Pupil.Level
     Studnet.Grade.Year = Pupil.Year
     Studnet.Classroom = Pupil.Classroom
     Studnet.School = Pupil.[Current School]
     Studnet.[Previous School] = Pupil.[Previous School]
     Studnet.[Join Date] = Pupil.[Join Date]
     Studnet.[Birth Date] = Pupil.[Birth Date]
     Studnet.Sex = Pupil.Sex
     Studnet.Religion = Pupil.Religion
     Studnet.Nationality = Pupil.Nationality
     Studnet.Language = Pupil.Language
     Studnet.[Admission Status] = Pupil.Status
     RETURN
end;
```

Figure 27: Compatibility Methods

CHAPTER SIX

CONCLUSION AND FUTURE WORK

6.1. Conclusion

Integrating heterogeneous databases is gaining more interest nowadays. The reason is that a huge number of autonomous (potentially heterogeneous) database systems were developed to satisfy single user requirements. Some recent applications require data from several of these systems while maintaining their autonomy is still desirable. In order to build such global systems heterogeneous database systems (HDBS) must be built above the autonomous database systems.

Building HDBSs is not trivial. Several tasks are involved. Schema integration is a major task. It is defined as the process of integrating existing or proposed component database schemas into a single global schema.

There are two design approaches for integrating HDBSs. The first approach creates a temporary global schema while the second creates a permanent global schema. In this research, the second design approach is adopted. The underlying schema architecture is a four-level schema architecture with standard terminology.

The major problem of schema integration is resolving the wide range of possible conflicts. Conflicts could range from simple structural differences to major variations in semantics, design and/or modeling.

One of the objectives of this thesis is to automate identifying semantically similar objects such that positive steps toward automating conflict resolution among such objects can be taken. To achieve that, a pool of the knowledge available from different component databases as well as the DBAs have to be used. This knowledge pool is chosen to be an on-line general lexicon taxonomy.

In the taxonomy, each term is represented as a class with properties and methods pertaining to the object represented by the class. The taxonomy contains hypernym/hyponym hierarchical links as well as synonym cross links.

The whole point of the thesis is to develop a **Ta**xonomy **B**ased **Da**tabase **Sch**ema **In**tegration Methodology (TAB DASHING). TAB DASHING is developed to achieve certain objectives. Its start point is the local database schema. It then goes through six phases, the result of which is a global schema. Analysis of TAB DASHING shows that it overcomes some major drawbacks of earlier methodologies and provides extra features.

To illustrate the workings of TAB DASHING, an illustrative example followed by a more comprehensive one are provided. To show its applicability, a case study is conducted. General implementation guidelines of TAB DASHING are also provided. These guidelines should help realizing TAB DASHING as a computerized system.

In brief, this thesis provides two contributions. The first contribution is that it gives details of a new database schema integration methodology. The methodology, besides its other advantages, provides two new features. First, it helps identifying semantically similar objects automatically such that positive steps toward automatically resolving schema conflicts can be taken. Second, the methodology is dynamic in nature which facilitates global schema creation and maintenance.

The second contribution of the thesis is a generalization of the classification of schema conflicts. The generalized classification is based on *Information Units* that can be at any level of granularity of any data model.

6.2. Future Work

There are four tasks that have particular significance in the development of an HDBS. The four tasks are: schema translation, access control, negotiation and schema integration. TAB DASHING handles schema integration. It provides enough support for schema translation. Access control can be easily handled by a five-level schema architecture. The task that TAB DASHING still has room for is negotiation. In future, it is possible to extend TAB DASHING to have suitable protocols for message exchange during carrying out the negotiation task.

In section 4.4.3, the subject of term matching is addressed. The better matching techniques used, the stronger TAB DASHING becomes. Hence, developing better matching techniques and incorporating them in TAB DASHING is another future work possibility.

Implementation of the methodology is yet another direction for future work. General implementation guidelines are provided in section 4.7.

REFERENCES

1. Batini, C et al. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, Vol. 18, No. 4, 1986, pp. 323-364.

2. Bertino, E and Martino, L. Object-Oriented Database Management Systems: Concepts and Issues. Computer, April 1991, pp. 33-47.

3. Bright, M et al. Automated Resolution of Semantic Heterogeneity in Multidatabases. *ACM Transactions on Database Systems*, Vol. 19, No. 2, June 1994, pp. 212-253.

4. Bright and Hurson, A. Linguistic Support for Semantic Identification and Interpretation in Multidatabases. *Proceedings of the 1st International Workshop on Interoperability in Multidatabase Systems*. IEEE Computer Society Press, Los Alamitos, California, 1991, pp. 306-313.

5. Castellanos, M et al. A Canonical Model for the Interoperability among Object Oriented and Relational Databases. Ozsu, Dayal & Valduriez (eds.) *Distributed Object management*. Morgan Kaufmann 1992.

6. Chung, S and Mah, P. Schema Integration for Multidatabases Using the Unified Relational and Object-Oriented Model. *Proceedings of the ACM Computer Science Conference*, New York, 1995, pp. 205-215.

7. Eder, J and Frank, H. Schema Integration for Object Oriented Database Systems. *Software Systems Engineering, American Society of Mechanical Engineers, Petroleum Division*, Vol. 59, 1994, pp. 275-284.

8. Elmagarmid, A and Pu, C. Guest Editors's Introduction to the Special Issue on Heterogeneous Databases. . *ACM Computing Surveys*, Vol. 22, No. 3, September 1990, pp. 175-178.

9. The Everyman Roget's Thesaurus of English Words & Phrases, Peter Roget, Chancellor Press, Britain, 1986.

10. Gotthard, W et al. System-Guided View Integration for Object-Oriented Databases. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 4, No. 1, Feb. 1992, pp. 1-22.

11. Gracia-Solaco, M et al. A Structure Based Schema Integration Methodology. *Proceedings of the IEEE International Conference on Data Engineering*, 1995, pp. 505-512.

12. Hurson, A and Bright, M. Global Information Access for Microcomputers. *Journal of Mini Micro Computers Applications*, Vol. 10, No. 2, 1991a, pp. 73-81.

13. Kaul, M. View System: Integrating Heterogeneous Information Bases by Object-Oriented Views. *In Proceedings of the 6th International Conference on Data Engineering (Los Angeles, California)*, pp. 2-10.

14. Kim, W et al. Classifying schematic and data heterogeneity in multidatabase systems. *Computer*, Vol. 24, No. 12, Dec. 1991, pp. 12-18.

15. Pitoura, E et al. Object Orientation in Multidatabase Systems. *ACM Computing Surveys*, Vol. 27, No. 2, June 1995, pp. 144-195.

16. Ram, S. Guest Editor's Introduction: Heterogeneous Distributed Database Systems. *Computer*, Vol. 24, No. 12, Dec. 1991, pp. 7-9.

17. Reddy, M et al. A Methodology for Integration of Heterogeneous Databases. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 6, No. 6, December 1994, pp. 920-933.

18. Sheth, A et al. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, Vol. 22, No. 3, September 1990, pp. 183-236.

VITA

- Mohammad Shafique Ahmad Al-Shishtawi
- Born on 24 May 1970 in Egypt
- Joined King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia in September 1987
- Received Bachelor of Science degree in Computer Science from the department of Information and Computer Science, College of Computer Science and Engineering, KFUPM in January 1993
- Received Master of Science degree in Computer Science from KFUPM in April 1997