Identifying, Classifying, and Resolving Semantic Conflicts in Distributed Heterogeneous Databases: A Case Study

Magdi Kamel
Naval Postgraduate School

During the past three decades, most medium and large organizations have seen a dramatic proliferation of databases and their associated database management systems. While these databases are useful in supporting their different activities, organizations soon discover the need to access and share data across these systems. Under current technology, this integration is usually not possible due to the heterogeneity of the database management systems, the platforms that these systems reside on, and the semantics of the different databases.

In this paper we address the issue of semantic conflicts identification and resolution in distributed heterogeneous databases. Using a real world example of heterogeneous databases obtained from various Department of Defense functional applications, we develop a framework for identifying and classifying semantic heterogeneity for databases represented in the Enhanced Entity Relationship Model. We also propose solutions for resolving each type of semantic conflict presented in the classification framework.

Most medium and large size organizations have over time developed many disparate and incompatible database systems. While these systems are useful in supporting their different activities, organizations soon discover the need to access and share data across these incompatible systems.

Under current technology, this integration is usually not possible due to the heterogeneity of these systems. This heterogeneity exists at three basic levels (Bertino et al., 1989). The first is the platform level. Database systems reside on different hardware, use different operating systems, and communicate with other systems using different communications protocols. The second level of heterogeneity is the database management system level. Data is managed by a variety of database management systems based on different data models and languages (e.g., file systems, navigational database systems, relational database systems, etc.). Finally, the third level of heterogeneity is that of semantics. Since different databases have been designed independently, semantic conflicts are likely to be present. This includes schema conflicts (e.g., name, type) and data conflicts (e.g., inconsistencies).

Several approaches have been proposed to address the issues of integrating heterogeneous database systems. A common requirement for these approaches is a semantically rich integrating model to represent, identify, and resolve the conflicts of the different component database systems. In this paper we examine the issues of identifying and resolving semantic conflicts when integrating incompatible databases using the Enhanced Entity Relationship (EER) Model.

This article is organized as follows. Section 2 presents a real world scenario of heterogeneous databases drawing on specifications obtained from various Department of Defense functional applications. Section 3 reviews an approach, known as the federated database approach, to integrating these heterogeneous databases and transforms the databases of Section 2 into the EER Model. Section 4 develops a classification framework of semantic heterogeneity and illustrates each type of semantic conflict with an example. Section 5 discusses possible means of resolving each type of semantic conflict identified in the classification framework. The proposed
solutions are applied to the individual schemas of our scenario to create a common global schema which represents the integration of data from the underlying databases. Finally, Section 6 concludes the paper with a summary and directions for future research. Appendix A provides an overview of the EER Model and a description of the diagrammatic convention used in the throughout.

**Heterogeneous Database Scenario**

Personnel information for the Department of Defense is currently stored in a variety of separate and diverse databases. A great wealth of data is available, but is maintained by different organizations, using different database management systems (DBMS), design philosophies, and hardware platforms. This section describes three actual administrative databases currently in use by various organizations within the Department of Defense to maintain information on commissioned officer personnel.

**Active Duty Military Inventory (ADMI) Database**

The ADMI database is maintained by the Defense Manpower Data Center, and includes data on all active duty military personnel, both officer and enlisted. It stores basic information of interest to the personnel administration function, such as name, rank, social security number, date of birth, sex, race, etc. It also keeps data on marital status, number of dependents, and whether a member’s spouse is also a member of the military. In addition to these facts, the ADMI database contains an extensive number of statistical elements concerning a member’s status on original entry to military service.

**Officer Personnel Information Systems (OPINS)**

The OPINS database is maintained by the Bureau of Naval Personnel to track commissioned officer assignment, promotion, and qualification status. OPINS stores similar common personnel information to that in the ADMI database, such as name, rank, sex, etc. The data reflects important differences in the OPINS area of interest, however. It contains relatively detailed data about an officer’s educational history, both civilian and military, as well as the military qualifications resulting from that training.

**Inactive Manpower and Personnel Management Information System (IMAPMIS)**

The IMAPMIS database is maintained by the Naval Reserve Force as an integrated repository of information on all officer and enlisted members of the Naval Reserve. Like the ADMI database, and the OPINS, the IMAPMIS maintains the essential administrative data needed by the personnel function (name, rank, pay entry base date, etc.). It also stores a wide variety of unique information specific to the Naval Reserve manpower management process. This includes reserve unit affiliation, last paid drill, total credited drills, and retirement points accumulated.

The three databases (ADMI, OPINS, and IMAPMIS) were all developed and implemented at different times, by different organizations, and for different purposes. It is not, therefore, surprising that in spite of a considerable overlap, each database has a different conceptual emphasis. A review of the Defense Manpower Data Center ADMI database reveals a very different area of interest than, for instance, that of OPINS. The Defense Manpower Data Center is concerned with issues such as total military end-strength, allotment of personnel resources to budgetary program elements, etc. OPINS, on the other hand, being a service-specific database, captures a very different set of data for a given officer, including present and past assignments by billet and promotion year group. There is a large overlap in the area of basic information (name, rank, SSN, etc.), but it is obvious that the designers of OPINS were interested in a different view of the commissioned officer than that presented by ADMI. IMAPMIS data overlaps both ADMI and OPINS, and additionally captures information of specific interest to the personnel management of the Reserve force, such as Reserve unit affiliation, and last credited drill period.

Besides varying areas of informational interest, the three database design efforts employed very different naming conventions. ADMI largely employs plain language labels for data elements which are easily understood while OPINS uses much more service-specific language. IMAPMIS follows the OPINS terminology closely, but is described in a particular DBMS language, and the entity and attribute names are awkward and not always easily matched to their corresponding elements in ADMI and OPINS. This results in a great deal of semantic heterogeneity, since it becomes an important issue to resolve whether each designer means the same thing when an attribute is called UNIT, for instance.

**Integration Strategy**

Frequently, queries that span the three databases need to be answered. For example, we may like to retrieve all available data for a given value of a key attribute, such as Social Security Number. Obviously, a query against any one of the databases cannot ensure retrieving all available data on a specific officer. Information on all officers may not exist in a single database. To guarantee no loss of any information already available, we must somehow formulate a global query which will be processed against a global schema that represents the integration of the three databases. When the query returns the requested information the further problem of conflicting data remains. Due to differences in update times, data entry errors, etc., even identical attributes for the same officer may contain different data values.

To allow queries that span several databases, a federated