On the Resolution of Representational Diversity

Joachim Hammer
Department of Computer Science
Stanford University
Stanford, CA 94305

Dennis McLeod
Department of Computer Science
University of Southern California
Los Angeles, CA 90089-0781

September 1995

1 Introduction

A major problem that arises frequently when attempting to support information sharing among autonomous heterogeneous database systems is the occurrence of *representational* differences that exist among related data objects in different component systems. A collection of cooperating but heterogeneous, autonomous component database systems may be termed a multidatabase system (MDBS) [LMR90], or loosely-coupled federated database system [SL90]. Such cooperating systems attempt to support operations for partial and controlled sharing of data with minimal affect on existing applications.

In this context, a key to supporting sharing involves folding of non-local data into the schema of an importing (local) component as gracefully as possible. Essential to achieving this is to support the resolution of semantic and representational (modeling) differences between the local and non-local perspectives. By resolving representational differences we mean two things: (1) determine as precisely as possible the relationships between sharable objects in different components, and (2) detect possible conflicts in their structural representations that pose problems when folding the non-local data into the local schema later on. In this chapter, we present an approach for resolving representational differences that preserves the autonomy of the participating databases.

2 Related Research

Research in the area of collaboration among heterogeneous database systems (HDBSs) began only a decade ago [FS83, SBD+81]. The term "heterogeneous databases" was originally used to distinguish work which included database model and conceptual schema heterogeneity from work in "distributed databases[1]" which addressed issues solely related to distribution [CP84]. Recently, there has been a resurgence in the area of heterogeneous database systems. This work may be characterized by the different levels of integration of the component database systems and by different levels of global (federation) services. In Mermaid [Tem87], for example, which is considered a tightly coupled HDBS, component database schemas are integrated into one centralized global schema with the option of defining different user views on the unified schema. While this approach supports pre-existing component databases, it falls short in terms of flexible sharing patterns. Due

[1]The term distributed database is used here as it has been mainly used in the literature, denoting a relatively tightly coupled, homogeneous system of logically centralized, but physically distributed component databases.

to the tight integration of conceptual schemas, no mechanism for dynamically resolving representational differences is needed.

The federated architecture proposed in [HM85], which is similar to the multidatabase architecture of [LA86], involves a loosely coupled collection of database systems, stressing autonomy and flexible sharing patterns through inter-component negotiation. Rather than using a single, static global schema, the loosely coupled architecture allows multiple import schemas, enabling data retrieval directly from the exporter and not indirectly through some central node as in the tightly coupled architecture. Examples of loosely coupled FDBSs are MRSDM [Lit85], Omnibase [REC+89], and Calida [JPSL+88]. Resolution of representational differences is performed manually.

Most of the early work on methodologies and mechanisms for integrating individual, user-oriented schemata focused on relational and semantic database models, and (partial) unification of multiple heterogeneous object-based databases is still in its infancy. In their 1986 survey paper, Batini et al. [BLN86] investigate twelve integration methodologies and compared them on the basis of five commonly accepted integration activities. However, most of the approaches examined in their survey do not directly address the problem of resolving representational differences. Kim et al. [KCGS93] provide the most up-to-date enumeration and classification of techniques for resolving representational conflicts arising within the context of multidatabase schema integration.

One common approach to conflict resolution is to reason about the meaning and resemblance of heterogeneous objects in terms of their structural representation [LNE89]. However, one can argue that any set of structural characteristics does not sufficiently describe the real-world meaning of an object, and thus their comparison can lead to unintended correspondences or fail to detect important ones. Other promising methodologies that were developed include heuristics to determine the similarity of objects based on the percentage of occurrences of common attributes [HR90, NEL86, SLCN88]. More accurate techniques use classification for choosing a possible relationship between classes [SSG+91]. Whereas most of these methods primarily utilize schema knowledge, techniques utilizing semantic knowledge (based on real-world experience) have also been investigated. Sheth et al. [SK93] introduce the concept of semantic proximity in order to formally specify various degrees of semantic similarities among related objects in different application domains based on the real-world context in which these objects are used. Fankhauser et al. [FN92] present an approach that utilizes fuzzy and possibly incomplete real world knowledge for resolving semantic and representational discrepancies. In their methodology, class definitions, or more generally schemas, are disambiguated by matching unknown terms with concepts in an interconnected knowledge base. A similar methodology is suggested by Heiler and Ventrone [VH93] that uses so-called enterprise models to provide an application context for describing unknown terms in the schemas of the participating components. [[Shall we mention Carnot here, too]]

A different approach uses behavior to solve domain and schema mismatch problems [Ken91]. Domain and schema mismatch are two important semantic integration problems for interoperating heterogeneous databases. The domain mismatch problem generally arises when some commonly understood concept, for example money, is represented differently in different databases (i.e., U.S. dollars vs. English pounds). Schema mismatch arises when similar concepts are expressed differently in the schema (i.e., a relationship that is being modeled as one-to-one in one schema and one-to-many in another). Kent [Ken91] proposes to use an object-oriented database programming language to express mappings between these common concepts that allow a user to view them in some integrated way. It remains to be seen if a language that is sophisticated enough to meet all of the requirements given by Kent in his solution can be developed in the near future.

Figure 1: Example 1. Semantic heterogeneity

3 Heterogeneity in a Collaborative Sharing Environment

In order to support the sharing of information among a collection of autonomous, heterogeneous databases we must overcome many types of *heterogeneity* that exist at various levels of abstraction (e.g., hardware heterogeneity, operating system heterogeneity) making it difficult for components to cooperate. Heterogeneity is a natural consequence of the independent creation and evolution of autonomous databases which are tailored to the requirements of the application system they serve and there is not always an agreement regarding the clear definition of the problem. In this section, we will focus on a specific kind of heterogeneity that occurs at the database system level, called *representational heterogeneity*. By representational heterogeneity we mean variations in the meaning in which data is specified and structured in different components. Before we can present a solution to resolving representational heterogeneity, it is useful to examine the different kinds of heterogeneities that may occur. In order to clarify the concepts that we employ in this work, we are using several sample database schemas taken from the scientific community. In specific, our example scenario depicted in Figure 1 draws from several existing gene banks (e.g., Brookhaven Protein Databank [BKW$^+$77], the human genome repository [Cou88]) that store information about structure and sequence of macromolecules in humans as part of a large-scale genomic decoding initiative by the NSF [Fre91].

3.1 A Spectrum of Representational Heterogeneities

Within the context of a multidatabase system with heterogeneous, autonomous components, we can identify a spectrum of representational heterogeneity based on the following levels of abstraction:

1. Meta-data language (conceptual database model):
   The components may use different collections of and techniques for combining the structures, constraints, and operations used to describe data. Different data models provide different

Figure 2: Example 2. Semantic heterogeneity

structural primitives (e.g., records in the relational data model versus objects in the functional data model) and operations for accessing and manipulating data (e.g., QUEL versus OSQL[2]). Note, even when two database systems support the same data model, differences in their data definition languages may still contribute to semantic heterogeneity.

2. Meta-data specification (conceptual schema):
   While the components share a common meta-data language (conceptual database model), they may have independent specifications of their data (varied conceptual schemas). For example, this refers to the different schemas in Figure 1 modeling similar information on genomic data in four different ways.

3. Object comparability:
   The components may agree upon a conceptual schema, or more generally agree upon common sub-parts of their schemas; however, there may be differences in the manner in which information facts are represented [Ken89]. This variety of heterogeneity also relates how information objects are identified, and to the interpretation of atomic data values as denotations of information modeled in a database (e.g., naming, missing information). In Figure 2, the types **Amino Acid Chains** in Schema A and **Amino Acid Sequences** in Schema B probably represent the same kind of information despite a difference in their type names. Both types have associated with them information regarding the length of an amino acid chain or sequence as well as information on who processed a particular sample. However, Schema A does not contain information on the exact structure of a chain whereas Schema B does (missing information).

4. Low-level data format:
   While the components agree at the model, schema, and object comparability levels, they may utilize different low-level representation techniques for atomic data values (e.g., units of

[2]Object flavored dialect of SQL, used as a DDL/DML in function-based database systems such as IRIS [FBC+87].