8-15-1997

Managing Subjectivity in Data Integration

Tania Nield
Infograte, Inc., neild@ece.nwu.edu

larry Henschen
Northwestern University, henschen@ece.nwu.edu

Managing Subjectivity in Data Integration

Tania Neild[**]
neild@ece.nwu.edu
Infograte, Inc. [**]
835 W. Diversey Pky Suite 10
Chicago, IL 60614
(773) 477- 0047

Larry Henschen
henschen@ece.nwu.edu
Northwestern University
Department of ECE
Evanston, IL 60201
(847) 491-3338

Abstract

The success of an integrated system is highly dependent on the successful development of a global data model. Any integrated system, a data warehouse, a mediator, or a composite system, must be able to handle multiple interpretations of a concept's symbols, extensions, and intensions. This paper discusses the underpinning subjective nature of data integration and illustrates the resulting semantic problems that occur. Annotations are introduced to extend the current models' ability to represent the semantic nuances of the underlying databases.

Introduction

While data within an organization has become increasingly dispersed, the need for a consolidated information view has remained important. Consequently, the research and development of integrated systems have becoming increasingly important. Such efforts have focused on either data warehouses which allow companies to physically consolidate data or on mediation systems which allow companies to virtually link distributed data. In either case, the database administrators (DBAs) of the various underlying data sources must understand and relate the data across the various systems. A global model is typically developed to represent the relationships, entities/objects, business rules, and integrity constraints in and between the various systems. Then each DBA determines how the data from their system interacts with the global model. Once the appropriate global model is developed and the underlying systems are related to it, the data may be physically or virtually consolidated.

Because success of an integrated system depends of the successful development of a global data model, integration methods, knowledge representations, and interoperability problems have been the focus of much information systems research. The knowledge representations include the Entity-Relationship (E-R) model, the Star Schema, and Object Oriented (OO) Model, while the interoperability problems vary from naming, scaling, formatting, computational, and granularity conflicts to identification inconsistencies and constraint mismatches. Although many models are able to manage syntactical problems, few models are able to completely handle the semantic heterogeneity. This paper discusses the subjective nature of data integration to explain why many of the semantic problems occur regardless of the knowledge representation. Annotations are introduced as a representational tool to explicitly model the subjective and semantic discrepancies between the databases.

Concepts

In order to model a body of information, there must be some way to organize similar things together. Each knowledge representation has a different term for organizing similar things. The E-R model classifies things into entities, and the OO model categorizes things into concepts. For simplicity and clarity, this paper will use OO terminology as it applies to all knowledge representations.

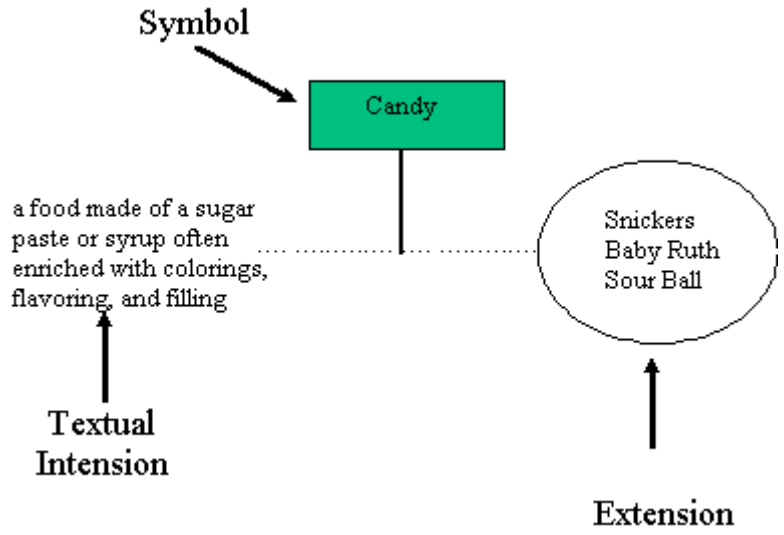

Figure 1.0 a: Concept with Text Intension

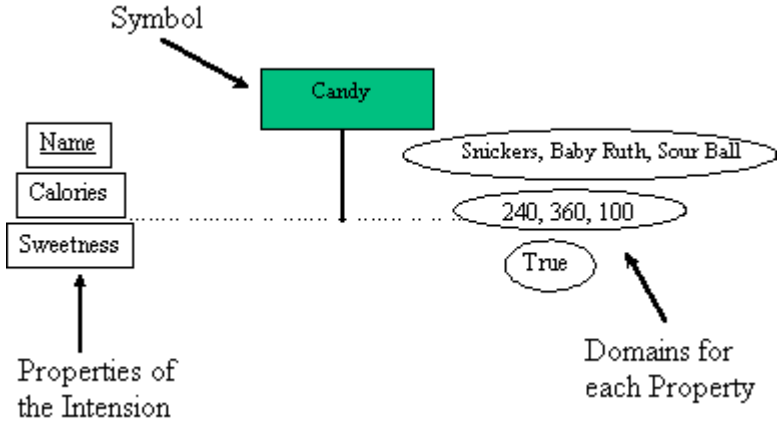

Figure 1.0 b: A Concept with Properties

A concept can be generally thought of as a recognition device that categorizes objects with similar properties together. A concept can be subdivided into an *intension*, an *extension*, and a *symbol*. The intension is the concept's definition. The extension is the set of all objects to which the concept's intension applies. The symbol is a word or picture that is used to denote the concept. [MO95] Figures 1.0 a and b illustrate a Candy concept.

Classifications

In order to integrate heterogeneous data sources, the objects from the various data sources must be classified into concepts. Because the concepts at the global level are likely to vary from the concepts at the individual database level, the re-classification of concepts is not a simple task. Multiple interpretations of symbols, extensions, intensions and the overall model complicate this process.

Symbols

There are can be many symbols for the same intension. A concept with multiple symbols and the same intensions is called a synonym. Consider the two sales databases where one database has a table called Client, and the other system has a table called Customer. While each table may list the organizations that

purchase goods and services from the company, the DBAs of the individual tables may not realize these tables each contain the same type of objects.

A symbol can have multiple intensions. A concept with the same symbol and different intensions is called a homonym. For example, there may be two databases that both have a table called Candy. One table may include the values {Sour Balls, Candy Canes} because its Candy intension is "crystallized sugar". The other table may contain a list of measurements because its Candy intension is "any of the various units of weight used in India, Burma, or Ceylon usually equal to between 500 and 600 pounds". So even though two DBAs may both be referring to the Candy symbol, they are each actually referring to different concepts.

Symbols are improperly used to help determine extensions. Given a universal concept with the symbol Candy and the intension, "a confection", a DBA may exclude a French Fries object and include Sour Ball object into its extension. However, given the concept with the symbol Fast Food and the same intension, the same DBA may include French Fries from the extension and exclude Sour Balls from the extension. A DBA's associations with the symbol changed the extension. With several DBAs and their different associations with the symbols, this problem is magnified. The connotations of a symbol often lead to an object being included in an extension where it does not match the intension.

Extensions

Two or more concepts can have the same extension. For example, there may be two properties, Date and Mileage, in two different databases which both have the values '19960130' and '19961228'. Consequently, the DBAs may think they are referring to the same concept because the values of the objects are the same, but they are actually referring to different concepts.

Similarly, two or more concepts can have different extensions for the same concept. Consider a Mileage property in two different databases. If they have distinct non-overlapping data sets, then the DBAs may think they are referring to different concepts when they are actually referring to the same concepts.

Intensions

The intension of a concept is often ambiguous. Consider the alternate Candy intension "a fun, tasty, and caloric food". One DBA might think that the French Fries object is a Candy object because she considers French Fries a fun, tasty, and caloric food. Additionally, the DBA may not think the Sour Ball object is a Candy object because she does not consider Sour Balls tasty. On the other hand, another DBA may believe that Sour Balls are a tasty Candy object and that French Fries are not a fun Candy object. Because the intension of the Candy concept was subjectively defined, the DBAs associated different and even conflicting values to the same concept. A concept's intension can be hard to define and can result in multiple interpretations and multiple extensions.

Many modelers avoid creating a well-defined and unambiguous intension by replacing the textual intension with a set of properties and by replacing the extension with a set of domain values for each property. Once a set of properties for a universal concept is established, the domain for the property is a set of values from the underlying databases. Figure 1.0 b represents the Candy concept with properties and domains. In this figure, Snickers, Baby Ruths, and Sour Balls are the domain values for the Name property. While the properties and domains are required to create a complete and structured model, they are not a substitution for the intension.

Model

Even when intensions are well defined and DBAs are able to accurately relate their data objects to global concepts, the resulting model can be complicated and cumbersome to interpret. For example, consider two university databases with a 'teacher' concept defined as 'a person who teaches a class'. Both DBAs will link

their individual database teacher objects to global concept. Now assume that Database 1 (DB1) has an implicit constraint that states all teachers are animal biology teachers. In order to capture this knowledge, the teacher concept could be sub-typed by subject taught. If DB1 has another implicit constraint that states all animal biology teachers own a pet, then the relationship between animal and teacher or animal and animal biology teacher could be created. The process of sub-typing or new concept-relationship development may quickly continue and result in a model that is elaborate and difficult to maintain. Additionally, subjective discrepancies are increasingly possible at each stage.

Although a well defined and manageable model may be created, users may not know or agree with the developed concepts. In mediated systems, this is particularly difficult because users may access the integrated data through their individual data source's model. This problem is illustrated through a mediated query in the integrated system for the two university databases noted above. Assume a user asks about teachers. Because knowledge about subject and pet-ownership is implicit in DB1 and not available in DB2, the users have no method to distinguish when they wish to ask about all teachers, animal biology teachers, or pet owners. When the granularity and implicit constraints of the underlying systems will often differ from the global model, the DBAs will be forced to decide what distinctions the users will want at the individual database level. If a DBA believes a user will care about a given implicit constraint or finer concept granularity, a local view will have to be created in the individual database model or the user will have to issue all queried directly to the global model.

Annotations

In order to model the semantic nuances between various database models, additional logic can be added to explicitly represent the conditions concerning when, why, who, where, and how the individual databases relate to the global model. In reference to the previous teacher example, it would be simpler to indicate when the distinction between teacher, pet owner, and animal biologist is required. Perhaps queries from a certain application reference the attribute as a pet owner, perhaps queries which include references to a specialization concept are concerned with the teacher as animal biologist, and perhaps all users in the accounting department reference the concept as a general teacher. If these conditions could be expressed in the global model, then users could continue to issue queries through their original systems and DBAs would not have to create all of the various local views. Similarly, in the previous candy example it may be important to determine if a user finds a particular food a fun, tasty, and caloric. A user could be prompted regarding their taste for various foods or presented with qualifications for who or what was used to determine that a food was candy. In these cases, additional knowledge could be added to the model to represent the object qualifications.

Consider two employment models for the integration of fictitious Green and Blue company databases. In these databases, the Green Company notes all retirees as an employee of the company with Status = 'R' and no job title other than the implicit retiree title. Additionally, all other Green company employees must have a job title. In the Blue Company, employees are not tracked once they have retired. The Blue company is a flat organization and does not have explicit job titles; an employee's job title is implicitly 'a staff member of a given department'. Consider the query: Find the job titles for Rob Roy. If Rob Roy retired from the Green company and works in the Blue company's accounting department, then the model in Figure 2.0a will not return any information. Figure 2.0 is only able to model explicit knowledge and no explicit job titles are available for Rob. However, Figure 2.0b's annotations model the implicit job title knowledge and provide more detailed information that Rob is a retiree and accounting staff member.

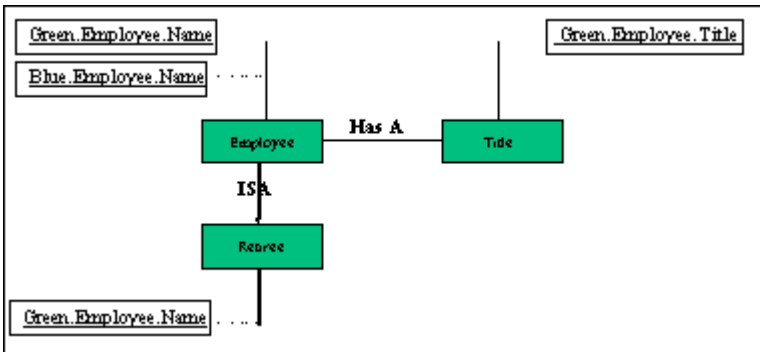

Figure 2.0 a: A Standard Model

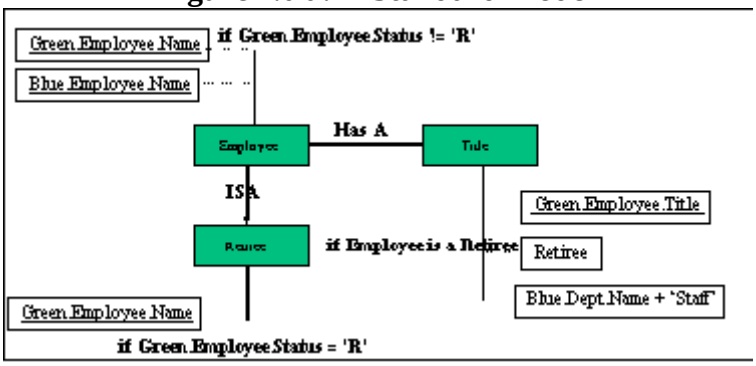

Figure 2.0 b: An Annotated Model

Although various languages such as HiLog, logic, or KIF could be used to express the implicit knowledge, a general format was used for simplicity. Also the language and interpretation of annotations can vary with the functionality of the annotations. Annotations could be divided into various types such as preconditions, postconditions, physical join operations, and permissions.

Summary

Due to the subjective nature of heterogeneous data integration, semantic problems often occur. The symbol for two or more concepts can be the same, unknown, or different because it is the intension only that uniquely determines the extension. Symbols are only a sign for the intension and as such they are not part of the intension and do not imply the extension. Likewise, an extension is simply a set of objects and does not imply the symbol or intension. The set of objects in an extension can not be used to determine if two concepts are related, and may only be used as a guide. Any model's intensions should be well documented, unambiguous, and visible to both the DBAs and users. Moreover, models that maintain additional logic to explicitly represent the conditional equivalence and implicit information will help manage and uncover the subjective complications. An annotated model facilitates integration by allowing a more flexible and comprehensive model to be built.

References- Available on Request from Neild