Semistructured Data and its Conceptual Models

Murali Mani
Worcester Polytechnic Institute, USA

Antonio Badia
University of Louisville, USA

INTRODUCTION

Semistructured data is data with no predefined schema, or a very flexible schema. Because such data does not fit well in traditional databases, new data models have been developed to deal with it. XML is the most well known of these new data models.

Traditional database systems have adapted to handle XML data by extending data types, query languages, indexing methods, and optimization techniques to the nature of XML; also, brand new database systems have been developed. However, the first step in developing a database is to create a conceptual model that can be used as the starting point for the design process (Davis, 1993). Despite the fact that existing conceptual models, most notably, Entity-Relationship (E-R) models (Chen, 1976), are inadequate for semistructured data because relatively little research is devoted to adapting conceptual models to the characteristics of this type of data (Badia, 2002; Elmasri et al., 2002). In this paper, we will review the main characteristics of XML and E-R, show the mismatch between them, and propose extensions to E-R that address those mismatches.

BACKGROUND

In this section, we review the basic ideas of conceptual models and XML in order to fix some vocabulary for the rest of the paper.

CONCEPTUAL MODELS

We briefly review the main characteristics of E-R models (Chen, 1976; Thalheim, 2000). There are other conceptual models, like Unified Modeling Language (UML) (Rumbaugh, Jacobson & Booch, 1999) and Object-Role-Modeling (ORM) (Halpin & Bloesch, 1999); we will not include them here for lack of space, although we will make a few comments later on.

An Entity-Relationship (E-R) (Chen, 1976; Thalheim, 2000) model is based on three basic concepts: entity types, attributes, and relationships. E-R models are usually depicted in E-R diagrams; an example is given in Figure 1. Entity types are depicted as rectangles with a name inside, attributes as ovals, and relationships as lines with a diamond shape on them.

Entity types represent things either real or conceptual. They denote sets of objects, not particular objects; in this respect, they are close to classes in object-oriented models. The set of objects modeled by an entity type is called its extension. Particular objects are called entities.

Relationships are connections among entity types. Relationships may involve any number of entity types; those involving two entity types (called binary relationships) are the most common. However, n-ary relationships (involving n > 2 entity types) are also possible. In particular, relationships relating one entity to itself are allowed. For example, a relationship Manager-of may relate the entity Employee to itself. To distinguish the ways in which Employee participates in this relationship, roles are added to the entity (manager and managee, in this example). Relationships are fundamental in an E-R model in that they carry very important information in the form of constraints: participation constraint tells us whether all objects in the extension of an entity are involved in the relationship or whether some may not be. For example, entities Department and Employee have a relationship works-for between them. If all employees work for some department, then participation of Employee in Works-for is total. However, if there can be employees which are not assigned to a particular department, then participation is partial. Cardinality constraint tells us how many times an object in the entity’s extension may be involved in a relationship and allows us to classify binary relationship as one-to-one, one-to-many, or many-to-many. There are several notations to state constraints in an E-R diagram. The one chosen here associates with each entity type E and relationship R a pair of numbers (min, max), where min represents the minimum number of times an entity in E appears in R (thus, min represents the participation constraint by being 0 for partial and 1 for total), and max represents the maximum number of times an entity in E appears in R (thus, max represents the cardinality constraint by being 1 for one-to-relationships and greater than
for many-to relationships. The latter case is traditionally represented by using the letters \( M \) or \( N \). Thus, the (1,1) by Employee and Works-for indicates that all employees work for exactly one department; the (0,M) by Department and Manages indicates that not all departments manage projects, but those that do may manage more than one, and so on.

Entity types and relationships may have attributes, which are properties with a value. Attributes convey characteristics or descriptive information about the entity type or relationship to which they belong. Attributes may be simple or composite (made up of simpler parts, like the attribute Address of entity Department in the example, which is made up of parts named street, city, and zip), single or multivalued (capable of having one or several values for a particular entity; multivalued attributes are displayed as dual ovals, like locations in the example above, meaning that some department may have multiple locations), and primitive or derived (a derived attribute value is computable from other information in the model). A key attribute is an attribute whose value is guaranteed to exist and be different for each entity in the entity type. Therefore, the attribute is enough to point out a particular entity. All entity types are assumed to have at least one key attribute.

A contentious issue is the semantics of entity type’s attributes, in particular, whether attributes are required (i.e., every entity of the type must have a value for each attribute of the type) or optional (i.e., some entities of the type may or may not have values for some attributes). Different authors take different views on this issue, some even arguing that it is a mistake to consider attributes optional (Bodart et al., 2001). Since this has an impact when transforming E-R models into different data models, we will point out how to deal with each view.

Some E-R models admit weak entities, entities with no key attributes; these entities are connected by a one-to-many relationship to a regular entity, called the strong entity. What characterizes a weak entity is its lack of clear identity (reflected in the lack of a key) and its dependence for existence on the strong entity. As a typical example, an entity loan may have an associated weak entity payment. Clearly, a loan is associated with several payments (hence, the one-to-many relationship), and if a loan ceases to exist (say it is paid off), then the associated payments also cease to exist.

Many proposals for additional features have been made over the years. The most successful one is the addition of class hierarchies by introducing IS-A (class/subclass) relations between entities. This addition, obviously motivated by the success of object-oriented methods for analysis, allows the designer to recognize commonalities among entities; usually this means shared attributes exist. Shared attributes are removed and put together in a new entity (class) which is a generalization of the others, and a class-subclass relationship is created. As in object-oriented approaches, inheritance of attributes is assumed. In Figure 1, entity type Employee has two subtypes, Hourly-employee and Salaried-Employee. The IS-A relationship is indicated by a downward triangle instead of a diamond in the line joining the involved entity types. The IS-A relationship can be annotated to distinguish several situations: whether the subclasses are disjoint or not and whether the subclasses together cover the superclass (i.e., every entity of the superclass must also belong to one of the subclasses) or not. Note that