Conversion of Generalization Hierarchies and Union Types from Extended Entity-Relationship Model to an XML Logical Model

Rebeca Schroeder  
Informatics and Statistics Department,  
Federal University of Santa Catarina,  
Florianópolis/SC, Brazil 88040-900  
+55 48 33317566  
rebecks@inf.ufsc.br

Ronaldo dos Santos Mello  
Informatics and Statistics Department,  
Federal University of Santa Catarina,  
Florianópolis/SC, Brazil 88040-900  
+55 48 33317566  
ronaldo@inf.ufsc.br

ABSTRACT
This short paper proposes alternative rules for converting generalization/specialization hierarchies and union types, defined in the Extended Entity-Relationship model, to an XML logical model. Our approach considers all the possible constraints and constructs for generalization and union types, generating abstract schemas for the logical design of XML documents.

Categories and Subject Descriptors
H.2.1 [Logical Design]: Schema and subschema

General Terms
Design

Keywords
XML schemas, EER, Generalization/Specialization, Union Types

1. INTRODUCTION
Methodologies that translate conceptual models into XML data models are useful in several contexts. In XML database design, for example, a conceptual schema can be defined for representing an application domain and then translated into a logical schema that considers the XML data model. Database schemas can be translated into conceptual schemas through a reverse engineering process and then converted to XML documents. Besides, such methodologies can be used to design XML documents that will be interchanged among business partners in a same application domain.

In conceptual models such as EER (Extended Entity-Relationship) and UML (Unified Modeling Language), generalization/specialization and union types are considered fundamental and useful modeling constructs [2]. However, most of the approaches that translate conceptual models into an XML data model focus on the conversion of entities, classes and associative relationships and not on the conversion of generalizations and union types, as well as their related modeling constraints. Another problem with current proposals is that they generate model-dependent schemas, e.g., the conversion is directed to a specific XML schema language, like DTD [3] or XML Schema [1, 4].

This short paper is focused on the conversion of generalization/specialization and union types from the conceptual level to an XML logical level. As in conventional database design, we consider a three-level modeling for XML document design. In our approach, the conceptual level is represented by the EER model. The logical level is represented by an XML logical model, which is an abstract representation for different XML implementation models. The implementation level is represented by XML schema languages like DTD and XML Schema.

In this paper we present alternative rules for converting generalization/specialization and union types from EER model to the XML logical model. Only constructs of the XML data model which are common to most relevant XML schema languages are considered in our XML logical model. This logical model makes our conversion a portable approach regarding different XML schema languages that it can represent. Thus, we proceed to represent all the possible constraints and constructs for generalization and union types by XML elements, attributes and relationships.

2. CONVERSION OF GENERALIZATIONS
We define three rules to convert generalization hierarchies from EER schemas to XML logical schemas (see Figure 1). These rules are based on general strategies to convert EER generalizations to conventional database models defined in [2].

The first alternative is defined by Rule1, and the Figure 1(b) presents the XML logical schema generated from the generic EER hierarchy of Figure 1(a). In this rule, each EER superclass and subclass is explicitly represented by an XML complex element, and each element maintains their own attributes. Hierarchical relationships are established among the superclass and subclass elements to represent the subset relationship in a straightforward way. Disjointeness constraints are represented in the superclass’ element by one of two order constructs for XML complex elements: ordered or exclusive. Completeness constraints are represented by the cardinality of the target element in a source element. All constraints on generalization types can be converted...
by Rule1, except total and overlapping generalizations because there is no way to guarantee that at least one of the sub-elements will be present to represent the total constraint. The main advantage of this alternative is that it is flexible, specially when superclass or subclasses have relationships with other entities.

Figure 1. Conversion alternatives

The second alternative generates only XML elements for the subclasses, and its attributes are propagated into each subclass element. This alternative is defined by Rule2 and the target XML logical schema is shown in Figure 1(c). Total constraint is guaranteed because only elements for the subclasses are created. Therefore, this rule cannot be applied to partial generalizations. This alternative generates redundancy if a superclass instance is specialized in more than one subclass instance. Thus, it is not a suitable alternative for overlapping generalizations. Despite that, there are more limitations on Rule2. First, other relationships with the superclass must be repeated for each subclass element, creating undesirable redundancy. Second, this rule is not convenient when there are many attributes in the superclass. However, this is a simple and suitable way to represent total and disjoint generalizations.

The alternative defined by Rule3 generates only one XML element from a generalization hierarchy. The XML element is created to represent the superclass and its attributes, as well as the attributes of the subclasses, as shown in Figure 1(d). Subclass attributes are treated as optional, and a discriminating simple element type is added to the element to distinguish the subclasses’ instances. Completeness and disjointness constraints are represented by the cardinality of the simple element type. On applying this strategy, we assume that the distinction between subclasses is irrelevant to most superclass instances. This approach generates the simplest schema, and no hierarchical relationships are needed. However, it addresses some disadvantages, like the possibility of generating a large number of null values for the subclass attributes. Besides, all relationships that are related only to the subclasses should be converted to relationships with the superclass element.

3. CONVERSION OF UNION TYPES

Union types with total constraint may be converted as a total and disjoint generalization. Thus, applying the Rule1, elements created to represent the superclasses become sub-elements of the subclass element. In Rule2, only elements for superclasses are created, and in Rule3 an element to represent the subclass and all the attributes of the superclasses is created. However, the conversion of partial union types is more complex, and Rule1 and Rule3 cannot be applied because of the partial constraint of the union type.

The first possibility to convert partial union types is to apply Rule2. In this case subclass attributes must be optional for representing the partial constraint. The second one is to establish the relationship among subclass and superclasses through reference attributes. This strategy can represent all constraints of the union types, and can also be applied to other cases of multiple inheritance when the other rules do not work. However, reference attributes reduce the connectivity among elements of the schema and the complexity of the logical schema is increased.

4. CONVERSION OF MULTI-LEVEL HIERARCHIES

For converting multiple-level generalization hierarchies, we must perform a bottom-up analysis of the hierarchy, treating one generalization/union type at a time until we reach the root entity. For each step, it is necessary to analyze constraints over the hierarchy and EER associations related to the entities of the hierarchy to achieve the suitable conversion strategy.

5. CONCLUSION

In this short paper we present conversion rules for translating EER generalizations and union types into an XML logical model. We also discuss the application of rules on each EER composition involving these conceptual constructs. The XML logical schemas generated by the proposed rules are not tied to an XML implementation model, and it could be translated to the most common XML schema definition languages.

Different rules can be applied to a constraint case of generalization or union type, and it is possible to generate different XML logical schemas from a conceptual schema. We are currently developing a tool to generate logical schemas in an automatic way depending on the conceptual schema constraints, the applicability of the conversion rules proposed, and other criteria that could be established by an expert user. Besides the generalization and union types, other EER constructs are treated by specific conversion rules in this tool, implementing a complete conversion from the EER model to the XML logical model.

6. REFERENCES


