ABSTRACT
This paper addresses the problem of efficiently transforming relational schemas into nested-based XML schemas. The proposed conversion algorithm considers preserving the structural constraints, cardinality and participation ratios, by analyzing each relation in relationship to others. Several candidate XML structures are proposed for each type of relationship according to its structural constraints from which the optimum one is chosen in terms of compact nested XML structure and determining the smallest XML data file.

Categories and Subject Descriptors
H.2.1 [Database Management]: Logical Design

General Terms
Design

1. INTRODUCTION
1.1 Hypothesis and Methodology
XML promises to be a more natural and flexible approach to storing real data. It also overcomes the data exchange problem between incompatible database systems. Making XML the preferred data storage technique implies solving several problems. One problem is how to make relational data available in XML.

This paper proposes a technique to transform relationships found in relational databases into nested XML structures by focusing on their structural constraints, cardinality and participation ratios. The source relational database must be in the third normal form, accept primary and foreign keys, and define other constraints such as unique and null. The third normal form of the relational database ensures a correct starting point for our approach; it eliminates possible problems generated by previous insertions, deletions, and updates. The normalized relational overcomes the lack of a standard normalized XML schema. The relational metadata of the source database is the key factor in generating appropriate XML Schema.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SAC ’04, March 14-17, 2004, Nicosia, Cyprus.  
Copyright 2004 ACM 1-58113-812-1/03/04...$5.00.

This paper focuses on generating an XML Schema for each relational data source that is most representative from the following perspectives: (1) preserving structural constraints (cardinality and participation) of the relationships from the input database; (2) representing the flat relational structures in a compact nested XML structure; (3) choosing the smallest nested representation for the XML data file; and (4) modeling multiple relationships that involve a set of relations.

We determine the cardinality of a relationship based on the source RDBMS’ metadata rather than on the data stored in the relations. Some approaches [2] have considered incorporating cardinality by analyzing only the actual data stored in the relations. Although the data stored in a database at some point may satisfy a constraint, its current satisfaction does not guarantee the constraint is correct or necessary. Deriving cardinality and other constraint information from the source metadata is more efficient and accurate because it is not data-dependent.

1.2 Contribution
This paper proposes a conversion method from the relational model to XML that uses the advantages from both the relational and XML models. The differences from other approaches are:

- the starting point: is formed by the relational schema which provides much useful information to generate the XML Schema. Most past approaches do not use the relational schema but try to extract the same information by analyzing the data stored in the database at a particular moment. Using the relational schema excludes errors or incorrect assumptions made based on the extraction method or the particular state of the stored data.

- the source relationships: our approach includes all relationship types (1:1, 1:M, and M:N) to the XML model.

- the XML nested structure has been used in other approaches that use non-normalized relations. Our approach uses normalized relations to convert to XML nested structures, thereby exploiting the advantage of well-formed relational design.

- the use of the XML Schema: forces the XML data to be correct according to the original relational schema. Past approaches generate an XML data file and based on it, a DTD or XML Schema. Our approach reverses this by considering the model first and then ensures the XML data file conforms to the model.
2. EXTRACTION OF THE RELATIONAL SCHEMA

To determine the structural constraints the relational database must be in third normal form as this excludes duplicates. We determine the structural constraints of each relationship by extracting the information from database metadata. To do this we need to define a set of rules to use for identifying cardinality and participation ratios in binary relationships. Consider the following situation: A and B are two relations with a foreign key relationship between them.

Definition 1 Parent/child relation: The child relation, also named the referencing relation [1], is the relation that contains the foreign key attribute. The child relation refers to one or more parent relations. In this approach the child and the parent are considered to be different relations.

Definition 2 Cardinality ratio of the parent relation: If A is the parent and B is the child relation, the cardinality ratio of A is always 1.

Explanation: The relational schema can only directly capture relationships of type 1:1 or 1:M between two relations. The relationship M:N is modeled using an intermediate relation that is connected to the other two relations through two relationships of type 1:M.

Definition 3 Cardinality ratio of the child relation: If the foreign key attribute has a unique constraint set on it either as a unique index or as a primary key, then the cardinality ratio of the child relation is 1 (thus, the relationship is of type 1:1). Conversely, if there is no unique constraint then it is a 1:M relationship.

Definition 4 Participation of the child relation: If the foreign key attribute has a not null constraint set on it, then the child relation participates totally in the relationship. Otherwise it participates partially in the relationship.

The participation constraint of the parent relation is not captured in the relational model’s structure and must be reentered using the information from the Entity - Relationship Diagram manually.

3. XML SCHEMA DEFINITION (XSD)

XML Schema is a definition language for XML documents that is substantially more verbose than DTD with new features [5] available including: (1) is written in XML; (2) supports element and attribute data types similar to the database data types; (3) supports standard or new user-defined simple and complex data types; (4) supports use of namespaces; and (5) is extendable for future changes.

XSD offers a set of features very similar to relational concepts. The element minOccurs has two roles in XML Schema. First, it replaces the null/not null constraint of the fields from tables. A column that accepts null values is converted into an element that has minOccurs="0". The default value, minOccurs="1", is used for non-null. Secondly, minOccurs replaces the participation values when modeling relationships between tables, while maxOccurs replaces the cardinality ratios. Relationships from the relational model can be represented in XML Schema using key and keyref tags on elements and attributes (not just attributes as in DTDs). Key is similar to a primary key, while keyref is a foreign key. These XSD features are incorporated in our approach.

4. MOTIVATING EXAMPLE

If a conversion generates a nested XML structure the key question is which table becomes the outer element and which the subelement? One option is to select the parent table as the outer element and the child table as the inner element. Another is to use the child table as the outer element and the parent table as the subelement.

Consider the 1:M relationship between tables Category and Products. Some XML users might prefer the Category table to become the top element as the Products table contains details of the categories stored. Thus, products should be included as details in Category element in the XML Schema. This structure is a Parent → Child, as each category element contains details about all the products from that category. Other XML users might prefer to have the product description first and included in it the description of the category from which it comes. Intuitively it is more natural to think of a product first and then identify the category to which it belongs. This structure is a Child → Parent one.

Possible XML structures are classified and encoded as follows:

- Class 1 designs the Parent → Child nested structure;
- Class 2 designs the Child → Parent nested structure;
- Class 3 designs the XML flat structure using keyref references;
- Class 4 designs additional Parent → Child nested structures for the M:N relationships modeled as a combination between a nested structure and a keyref reference. The nested structure models the link between one parent and the intermediate relation and the keyref reference models the link between the second parent and the child.

5. THE CONVREL ALGORITHM

The order of the metrics considered in our approach is: (1) structural constraints of the relationships from the relational model; (2) nested structure; (3) compact structure; (4) length of the generated XML file; and (5) similarity to the relational structure Parent → Child.

Using these criteria we determine the ways in which the relations and the relationships can be transformed into XML structures; discuss their advantages and disadvantages; and finally identify the best strategy in terms of the nestable property. The type of each relationship and the structural constraints of the participant tables are the decisive factors in defining the candidate XML class space. Within this space the balance of the criteria are applied.

A nested XML structure is a tree-like structure which represents a relationship. Thus, all records of the participating tables are included inside a single outer element. This is a qualitative factor represented by the following three possible values: (1) nested: a complete nested XML structure; (2) hybrid: a nested XML structure that includes keyref references; (3) not nested: a flat XML structure or structure where some data of a specific element is included in the nested structure but not all.

A compact structure is one that uses the minimum number of XML schema elements to represent a relationship. Thus, it allows a single complex element definition for each table. Conversely, a mixed structure allows data from a table to be dispersed
throughout the XML file in two or more distinct representations. A compact structure is a nested or a grouped-based structure.

A frequent reason for converting relational data to XML is to transfer information between two incompatible database systems. This requires the XML data file to be as short as possible. Even more, if data is stored in the XML format the length of the file has a dramatic influence on the performance of queries, updates, and delete operations. The length of the XML data file is considered only when more than one class remains after they are filtered by the nestable and compactness property preferring the shorter file. For these classes the estimated length of the XML data files is computed using the structure defined in the XML Schema.

If there is still more than one candidate class, the preferred class is the one that has the Parent \(\rightarrow\) Child orientation. In other words, between similar structures we prefer arbitrarily the parent relation to be the outer element and the child be the inner element.

If none of the candidate classes remains in the search space after filtering with the nestable property then the relational structure is converted to XML using keyref to replace the foreign key relationship directly.

The ConvRel algorithm that converts each relationship to an XML structure has the following steps:

1. Determine the relationships from the RDB.
2. For each relationship determine the inner and outer elements as follows:
   a. Determine the candidate XML classes based on the type of relationship and structural constraint ratios for the tables under consideration.
   b. If more than one candidate class is possible, choose the one with a nested and compact structure; if no class is left, transform the tables into separate elements and restore the relationship using keyref.
   c. If there is more than one candidate class with a nested and compact structure, then determine the length of the generated XML file and choose the one with the lowest value.
   d. If two or more classes have equal length then we choose arbitrarily the one with the Parent \(\rightarrow\) Child orientation.
3. Tables not involved in any relationship are transformed into isolated elements.

ConvRel analyzes each relationship to find a suitable XML nested structure to represent it. If no XML nested structure is found then the ConvRel converts each table separately and reconstructs their relationship using <keyref>. Thus, all tables and relationships from RDBMS are translated into XML.

Representing XML Schema using its own syntax requires substantial space and the reader gets lost in the implementation details instead of the current idea. Several notations are used to represent the XML structure graphically: [] for repetitive element; [ ] for optional element; \(\rightarrow\) followed by the inner element represents the subordination in a nested structure.

### 6. Identification of the optimum XML nested structure for each relationship

The structural constraints are represented as Parent Table (child participation; parent cardinality):(parent participation; child cardinality) Child Table. For example consider the relationship Parent (1,1):(0,M) Child. The relationship is of type 1:M where the parent relation participates partially. The child table has a total participation in the relationship, so its participation value is 1. The representation chosen for the structural constraints ratios is the one that is closest to the XML style and is interpreted as follows: each record from Parent corresponds to a minimum of 0, maximum of M records in Child. Conversely, each record in Child corresponds to a minimum of 1 and a maximum of 1 records in Parent.

Consider p a record in the parent table. We use \(\text{fk}(p)\) to represent the foreign key found in the child table that refers to the parent table. The following functions are used to compute the space required for storing records in the XML file: \(\text{size}(x)\) and \(|X|\). \(\text{Size}(x)\) applies to a record or an attribute and returns the number of bytes of its parameter. \(|X|\) applies to a table and returns its cardinality.

#### 6.1 The 1:M relationship

The first relationship type considered is a 1:M relationship between two arbitrary tables where one has the role of the parent (P) and the other of the child (C). Table 1 details the changes that the nested structures in Classes 1 and 2 require to represent the participation ratios associated with the relationship. For a (1:1):(1:M) relationship both Classes 1 and 2 generate nested and compact structures. Therefore, estimates for the XML files’ length must be computed.

\[
\text{Used Space 1} = \begin{cases} 
|P| \times \text{size}(p) + & \text{for outer elements} \\
|C| \times (\text{size}(c) - \text{size}(\text{fk}(p))) & \text{for inner elements} 
\end{cases}
\]

\[
\text{Used Space 2} = \begin{cases} 
|C| \times (\text{size}(c) - \text{size}(\text{fk}(p))) + & \text{for outer elements} \\
|C| \times \text{size}(p) & \text{for inner elements} 
\end{cases}
\]

Since for the (1:1):(1:M) case Classes 1 and 2 are compact and nested structures, the shorter one must be identified. It is clear that Used Space 1 <= Used Space 2 as \(|P| \leq|C|\) in a total 1:M relationship. The values of Used Space for Classes 1 and 2 are equal if it is a total 1:1 relationship, but not for a 1:M. Therefore, we conclude that Class 1 is the preferred representation for a (1,1):(1,M) relationship.

Class 1 in a (1,1):(0,M) relationship depicted in Table 1 is the only compact and nested structure as Class 2 includes parent records that do not participate in the relationship and have no child records. Conversely, Class 2 in a (0,1):(1,M) relationship is the only compact and nested structure. If both tables participate partially in the relationship, then no nested structure can represent it. Thus a flat conversion with the keyref element is used for relationship representation. Class 3, the flat structure, for all cases is illustrated in Figure 1.

---

1 For simplicity in presentation we consider that a foreign key refers to a primary key, but it can refer to any other unique candidate key.

2 For simplicity in presentation, Used Space 1 refers to the XML file length of Class 1. Used Space 2 refers to Class 2, etc.
6.2 The 1:1 relationship
The 1:1 relationship is a special case of the 1:M relationship where each parent record has at most one child record. The total relationship 1:1 determines both Classes 1 and 2 to be nested and compact. In addition to the 1:M relationship, they also have the same XML file length. Thus, the similarity to the relational Parent→Child orientation makes Class 1 preferable. The other three cases of partial 1:1 relationships are similar to those from Table 1 but allow at most one record outside of the nested structure because of the cardinality ratio constraint.

Figure 1: Class 3 for a flat conversion using keyref

6.3 The M:N relationship
A M:N relationship is physically implemented in the relational model using an intermediate relation, thus splitting the relationship into two 1:M relationships. The intermediate relation captures the primary attributes of the other two tables creating its own primary key from the original tables. As the M:N relationship is richer than the previous ones discussed, the conversion of this relational structure to XML can be accomplished in several ways (see Table 2). The M:N relationship allows two structures for Class 1, one with each of the parents as the outer element. Consider Class 1 the one where Parent A is the outer element and Class 1’ with Parent B the outer element. Class 1’ is detailed in Table 2 for the A (1:M):(0:N) B relationship as both classes must be discussed separately. The same distinction must be made for Classes 4 and 4’. Classes 4 and 4’ are derivations of Classes 1 and 1’ in the M:N relationships when one parent and its intermediate relation are captured in the nested structure. Further, the second parent is connected to the intermediate relation through keyref references. Class 3 is a flat structure formed by the two parents and the child relations and the keyref references from a child to each of the parents and is not detailed for this relationship type (see Figure 1 for a general idea).

For the relationship A (1:M):(1:N) B, Classes 1 and 1’ are both nested and compact structures, but their corresponding XML data files have different length. This must be evaluated in each case and is dependant on the record length of each parent relation. Class 2 is also nested and compact but allows higher redundancy thereby making the XML file larger.

Used Space 1 =

\[ |A| \times \text{size}(a) + \]
\[ |C| \times (\text{size}(c) - \text{size}(fk(a)) - \text{size}(fk(b))) + \]
\[ |C| \times \text{size}(b) \]

//outer elements from Parent A

//inner elements from Child are without foreign keys

Similarly,

Used Space 1’ =

\[ |C| \times \text{size}(a) + |B| \times \text{size}(b) + \]
\[ |C| \times (\text{size}(c) - \text{size}(fk(a)) - \text{size}(fk(b))) \]

Assume: Used Space 1 < Used Space 1’, then

\[ |C| - |B| \times \text{size}(b) < |C| - |A| \times \text{size}(a) \]

If we consider that |B| is approximately equal to |A| in a total M:N relationship, then the class that requires less space is the one with the outer element being the longest parent and the inner parent the shortest one.

The A (1:M):(1:N) B relationship allows Class 2 to be a nested and compact structure too. Thus, there is more than one candidate class that is nested and compact.

\[
\text{Used Space 2} =
\begin{align*}
|C| \times \text{size}(c) - \text{size}(fk(a)) - \text{size}(fk(b)) + \\
\end{align*}
\]

//outer elements from Child are without foreign keys

\[
|C| \times \text{size}(a) + \\
\]

//inner elements from Parent A

\[
|C| \times \text{size}(b) + \\
\]

//inner elements from Parent B

The XML file from Class 2 is always larger than Classes 1 or 1’ because each parent record must participate in at least one association: |C| >= |A| and |C| >= |B|. Thus,

\[
\text{Used Space 2 - Used Space 1} = \text{(|C| - |A|)*size(a)} >= 0 \\
\text{Used Space 2 - Used Space 1'} = \text{(|C| - |B|)*size(b)} >= 0
\]

Classes 4, and 4’ are unsuitable for enforcing the total participation for both parent relations. The nested part of these structures between one parent and the intermediate relation enforces the participation for one half of the relationship. Since the other part of the M:N relationship is represented using references (keyref), the structure is less controllable. Thus, the nestable property has only the value Hybrid for Classes 4 and 4’.

In summary, for a total M:N relationship Classes 1 or 1’ are suitable, so the shorter one should be selected. In a (1:M):(0:N) relationship, the parent that is partially involved in the relationship must be the top element and the inner elements are the intermediate table and the totally involved parent. In a (0:M):(0:N) relationship, Class 2 is the only candidate with a compact and nested structure.

### Table 1: Possible classes for a 1:M relationship

<table>
<thead>
<tr>
<th>Class</th>
<th>Structural constraints of the 1:M relationship</th>
</tr>
</thead>
<tbody>
<tr>
<td>P (1:1):(1:1) C</td>
<td>P (1:1):(0:1) C</td>
</tr>
<tr>
<td>Nested</td>
<td>Compact</td>
</tr>
</tbody>
</table>

7. RELATED WORK
Lee et al. [2] proposed an approach for creating nesting-based XML structures from flat relational schema. First, the Flat Translation (FT) converts each table into a flat element structure.
Secondly, the Nesting-based Translation (NeT) applies the *nest* operator to the flat structures. The input of this approach is the relational schema along with its data. The output is an unflattened element-oriented or attribute-oriented DTD. The unflattened process is applied to a single table at a time. For this reason, it can create nested structures only for non-normalized tables or for an intermediate (dependent) table, in normalized databases. The parent tables in normalized databases are not guaranteed to have repeatable values for any column; thus their translation using this approach will be a flat XML structure. The authors have conducted several experiments on FT and NeT and compared them with DTDs obtained by DB2XML. Based on these experimental results they conclude that their approach yields better results in terms of accuracy and length. Unfortunately, the *nest* factor used in NeT relies on the relational schema and also the actual data stored in the database, which makes this approach unreliable.

Lee et al. [3] [4] have extended the nesting approach to multiple tables, using the CoT algorithm (Constraints-based Translation). It is one of the first approaches that deals with relationships. The source database contains several interconnected tables and based on the cardinality of the binary relationships two types are identified 1:1 and 1:M. A directed IND (Inclusion Dependency) Graph of tables is created from which an empirical way to nest XML structures is identified.

Table 2: Possible Classes for a M:N relationship. Attributes of the child relation are represented by X.

<table>
<thead>
<tr>
<th>Class</th>
<th>Structural constraints of the M:M relationship</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A(1:M):(1:N) B</td>
</tr>
<tr>
<td>1</td>
<td>{A} → {Child}</td>
</tr>
<tr>
<td></td>
<td>→ X → X</td>
</tr>
<tr>
<td></td>
<td>→ B → A</td>
</tr>
<tr>
<td></td>
<td>Nested Compact</td>
</tr>
<tr>
<td></td>
<td>Nested Compact</td>
</tr>
<tr>
<td>2</td>
<td>{Child} X X A A B B</td>
</tr>
<tr>
<td></td>
<td>→ X → X</td>
</tr>
<tr>
<td></td>
<td>→ A → A</td>
</tr>
<tr>
<td></td>
<td>→ B → B</td>
</tr>
<tr>
<td></td>
<td>Nested Compact</td>
</tr>
<tr>
<td>4</td>
<td>{A} → {Child}</td>
</tr>
<tr>
<td></td>
<td>→ X → X</td>
</tr>
<tr>
<td></td>
<td>→ B → B</td>
</tr>
<tr>
<td></td>
<td>Hybrid Compact</td>
</tr>
<tr>
<td></td>
<td>Hybrid Compact</td>
</tr>
<tr>
<td></td>
<td>Hybrid Compact</td>
</tr>
</tbody>
</table>

Our approach extends the work by Lee et al. [3] [4] in the area of conversion from relational to XML data by including additional elements in the analysis such as: (1) more criteria in choosing the XML nested structure, including the impact of all possible combinations of relational structural constraint ratios; (2) M:N relationships conversion; (3) quantification and formalisation of the XML data files’ length; (4) use of XML Schema instead of DTD which implies additional relational information to be transferred in XML; and (5) algorithm formalisation and its implementation in an efficient tool.

8. CONCLUSION AND FUTURE WORK

The contribution of this paper is to analyze each type of relationship and determine a set of candidate XML structures capable of representing it. These candidate structures are filtered with criteria such as the nested and compact structure, and the length of XML data file. The result is an XML structure that best represents each relation and its participation in relationships. The conversion algorithm presented here has been implemented in Java Version 1.3.1. It extracts the metadata of a DB2 database and, based on additional user input for certain cardinality ratios, produces a nested XML Schema.

The approach presented in this paper has several limitations that must be considered for future research. We analysed each type of relationship separately. Relational databases have interconnected relationships for a set of relations that makes it impossible to model each relationship in a nested structure. This approach has dealt with binary relationships, but real relational databases can include more tables in each relationship. The conversion into XML nested structures of relationships that include three or more tables would be of interest. Query metrics should be included in the relational to XML conversion. It could have an important impact on the class selected as it determines the data retrieval speed. This is still an open issue and several query methods exist that use totally different approaches. Another important aspect is related to the XML flexibility. XML allows structures to be altered, either by adding or subtracting elements, subelements, and attributes. Possible changes determined by the XML structure evolution should be considered when translating the relational metadata.

9. REFERENCES


