UDC:004.63
Original scientific paper

CONVERTING RELATIONSHIPS TO XML NESTED STRUCTURES[1]

Angela C. Duta, Ken Barker, Reda Alhajj
Department of Computer Science, Universitiy of Calgary, CANADA
{duta,barker,alhajj}@cpsc.ucalgary.ca

Abstract: *Conversion of relational data to XML is a critical topic in the database area. This approach translates the rigid tabular structures of relational databases into hierarchical XML structures. Logical connections between bits of data depicted by relationships are represented more naturally by tree-like structures. Conv2XML and ConvRel are two algorithms for converting relational schema to XML Schema focusing on preserving the source relationships and their structural constraints. ConvRel translates each relationship individually into a nested XML structure. Conv2XML identifies complex nested structures capable of modelling all relationships existent in a relational database.*

Keywords: schema conversion, structural constraints, relationships, XML nested structure.

1. INTRODUCTION

Most data created over the last several decades has been stored using the relational model. Recently XML [11] is emerging as more prevalent in several areas including business. This is producing an increase in demand for tools to convert from relational databases to XML. Much work has investigated conversion techniques, either by translating the entire database or only data generated by a query. Some suggest mapping the database model to an XML structure either using *ad hoc* techniques or by using DTD or XML Schema to govern the process.

This paper details the conversion from the relational schema to XML Schema. In relational databases, relationships establish logical connections between tables. Participation and cardinality ratios associated with each relationship provide additional information about these connections. This information must be translated to XML so that the nested structures represent real data naturally. The resulted XML structure captures all connections between various parts of the relational data and presents data in a suitable way for Web publishing. 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`[3]. 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

[1] A preliminary version of this paper appeared in *Proceedings of the 6th International Conference on Enterprise Information Systems*, ICEIS2004, Porto, Portugal, April 4 -17, 2004.

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.

1.1. CONTRIBUTIONS

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 [8] 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. PAPER OVERVIEW

Following this introduction, Section 2 depicts an example that motivated us during this research. Sections 3 and 4 introduce ConvRel and Conv2XML that form the primary contribution of this paper followed by a practical example in Section 5 and the recent contributions in this area in Section 6. The paper concludes by stating conclusions and proposing future work

2. 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 through this paper 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.

3. THE *CONVREL* ALGORITHM

ConvRel analyzes each relationship to find a suitable XML *nested* and *compact* structure to represent it. 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* structure for a binary relationship is defined as a pair of outer element → inner element that (1) preserves the cardinality and participation ratios of the relationship and (2) captures data in a single XML root element. This qualitative factor is 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. This implies that there exists 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 → 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.

ConvRel converts each relationship to an XML structure using 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 → 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; → followed by the inner element represents the subordination in a nested structure.

3.1. 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 following: 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 *fk(p)* to represent the foreign key found in the child table that refers to the parent table[2]. The following functions are used to compute the space required for storing records in the XML file: size(x) and |X|. *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.

3.1.1. 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[3].

[2] 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.

[3] For simplicity in presentation, *Used Space 1* refers to the XML file length of Class 1, *Used Space 2* refers to Class 2, *etc.*

Used Space 1 =

$|P|*size(p)+$ //outer elements

$|C|*(size(c)-size(fk(p)))$ //inner elements

Used Space 2 =

$|C|*(size(c)-size(fk(p)))+$ //outer elements

$|C|*size(p)$ //inner elements

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 $|P| <= |C|$ 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.

Table 1: Possible classes for a 1:M relationship

Class	Structural constraints of the 1:M relationship			
	Parent (1;1):(1;1) Child	Parent (1;1):(0;1) Child	Parent (0;1):(1;1) Child	Parent (0;1):(0;1) Child
1	{Parent} → {Child}	{Parent} → {[Child]}	{Parent} → {Child} {[Child]}	{Parent} → {[Child]} {[Child]}
	Nested Compact	*Nested Compact*	*Not nested Mixed*	*Not nested Mixed*
2	{Child} → Parent	{Child} → Parent {[Parent]}	{Child} → [Parent]	{Child} → [Parent] {[Parent]}
	Nested Compact	*Not nested Mixed*	*Nested Compact*	*Not nested Mixed*
Preferred	**Class 1**	**Class 1**	**Class 2**	**Class 3**

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.

{Parent}
{Child}
{keyref from Child to Parent}

Figure 1: Class 3 for a flat conversion using keyref

3.1.2. 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.

3.1.3. 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).

Table2: Possible Classes for a M:N relationship. Attributes of the child relation are represented by X.

Class	Structural constraints of the M:M relationship			
	A(1;M):(1;N) B	A(1;M):(0;N)B		A(0;M):(0;N)B
1	{A} → {Child} → X → B	{A} → {[Child]} → X → B	{Parent B} → {Child} → X → A {[A]}	{A} → {[Child]} → X → B {[B]}
	Nested *Compact*	*Nested* *Compact*	*Not nested* *Mixed*	*Not nested* *Mixed*
2	{Child} → X → A → B	{Child} → X → A → [B]		{Child} → X → [A] → [B]
	Nested *Compact*	*Nested* *Compact*		*Nested* *Compact*
4	{A} → {Child} → X {B} {keyref from Child to B}	{A} → {[Child]} → X {B} {keyref from Child to B}	{B} → {Child} → X {A} {keyref from Child to A}	{A} → {[Child]} → X {B} {keyref from Child to B}
	Hybrid *Compact*	*Hybrid* *Compact*	*Hybrid* *Compact*	*Hybrid* *Compact*

Preferred	**Long parent** → **{Child}** → **X** → **Short parent**	**Partial participation parent** → **{[Child]}** → **X** → **Total participation parent**	**Class2**

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.

Used Space 1 =

$|A|*size(a)+$ //outer elements from Parent A

$|C|*(size(c)-size(fk(a))-size(fk(b)))+$ //inner elements from Child are without foreign keys

$|C|*size(b)$ //inner elements from Parent B

Similarly,

Used Space 1' = $|C|*size(a)+|B|*size(b)+$

$|C|*(size(C)-size(fk(a))-size(fk(b)))$

Assume: Used Space 1 < Used Space 1', then $(|C|-|B|)*size(b) < (|C|-|A|)*size(a)$.

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.

Class 2 is also nested and compact but allows higher redundancy thereby making the XML file larger.

Used Space 2 =

$|C|*(size(c)-size(fk(a))-size(fk(b)))+$ //outer elements from Child are without foreign keys

$|C|*size(a)+$ //inner elements from Parent A

$|C|*size(b)+$ //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,

Used Space 2 - Used Space 1 = $(|C|-|A|)*size(a) >= 0$

Used Space 2 - Used Space 1' = $(|C|-|B|)*size(b) >= 0$

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.

ConvRel guarantees that there is a way to transform each relationship or table from a relational schema to an XML Schema (see [2] for more details). Table 3 summarizes the XML structures for each type of relationship.

4. THE *CONV2XML* ALGORITHM

Table 3: Relationship conversion to XML. PR = partial participation relation. TR = total participation relation

Relationship	XML nested structure	Preferred class
(1;1):(1;1)	Parent → Child	Class 1
(1;1):(0;1)	PR → [TR]	Class 1 or 2
(0;1):(0;1)	Grouping	Class 3
(1;1):(1;M)	Parent → {Child}	Class 1
(1;1):(0;M)	Parent (PR) → {[Child (TR)]}	Class 1
(0;1):(1;M)	Child (PR) → [Parent (TR)]	Class 2
(0;1):(0;M)	Grouping	Class 3
(1;M):(1;N)	Longest Parent → {Intermediate relation} → Shortest Parent	Class 1
(0;M):(1;N)	PR → {[Intermediate relation]} → TR	Class 1
(0;M):(0;N)	Intermediate relation → [Parent A] → [Parent B]	Class 2

The ConvRel algorithm includes only a single relationship at a time. In a real relational database each table is connected to several other tables in a complex structure. In this section we discuss the influence this has on creating a nested structure for the entire database.

For simplicity we discuss only two 1:1 relationships between three tables. First, each relationship is converted separately to a nested XML structure using the ConvRel algorithm. An XML structure is then created that combines the two previously found to obtain a nested structure, if possible. This implies that we must identify the cases when two nested structures combined generate a valid nested structure.

Table 4: Two relationships with a common table. T = total participation. P = partial participation

Case	Participation ratios in the relationship A:B			Participation ratios in the relationship B:C			Relationship A:B:C XML representation
	A	B	XML representation	B	C	XML representation	
1	T	T	A (parent) → B (child)	T	T	B (parent) → C (child)	A → B → C
2	T	T	A (parent) → B (child) *Changes to:* B (child) → A (parent)	T	P	C (partial) → [B] (total)	C → [B] → A
3	P	T	A (partial) → [B] (total)	T	P	C (partial) → [B] (total)	A → [B] C keyref from C to B
4	T	T	A (parent) → B (child)	P	P	B C keyref from C to B	A → B C keyref from C to B

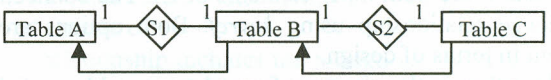

Figure 2: Two relationships between three tables

Consider the situation depicted in Figure 2 where Table A is the parent in the first relationship and Table B is the parent in the second relationship. Table 4 details four cases for this situation. For each case the resulted structure must capture and preserve the source functional dependencies. Case 1 from Table 4 is the nicest case where each relationship and their combination can be modelled in a nested XML structure. In Case 2 from Table 4 the total relationship 1:1 between two relations A and B as described above can be represented as Parent → Child or Child → Parent. Both are nested and compact XML structures and generate the same length of XML file. We have chosen the first one, as this is most similar to the relational model. However, if the relations A and B are also involved in other relationships that require some changes to properly model them, then modelling A:B as Child → Parent is the preferred approach. Case 3 does not allow a nested structure to model both relationships, although there is a nested structure for each relationship when considered separately. In this case one of the relationships is represented in XML using *<keyref>*. Case 4 depicts a relationship with partial participation of both relations. In this case, the relationship in which at least one relation has a total participation is modelled separately. The third relation (from the partial relationship – i.e. "C" in Case 4 of Table 4) is added to the structure and references are used to reconstruct the partial relationship.

It is important to analyse the tables' involvement in more relationships, which is accomplished by Conv2XML. Conv2XML uses a graph representation that combines all structures discovered by ConvRel. The vertices are tables and the edges represent connections between tables so the inner element is the head and the outer element is the tail

of the arc. Note that the arcs are not necessarily created following the orientation of the relationships.

Two categories of edges exist in the directed graph: (1) full edges representing links that are modelled as nested structures, and (2) dotted edges representing relationships that are modelled using *keyref*. The last type of arc is drawn from the child to the parent table.

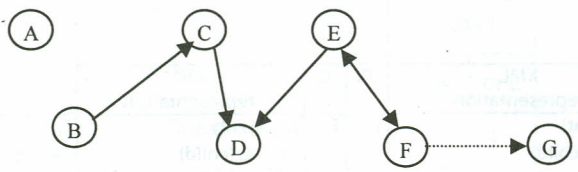

Figure 3: A directed graph representing the links between tables

In Figure 3, A is an isolated node, so it represents a table with no relationships. The edge F→G represents a loose connection because it can only be modelled using *keyref*. The edges B→C, C→D, E→D, and the bi-directional edge EF are full edges that represent relationships identified by the algorithm that can be modelled with nested structures. This analysis is done for each relationship separately, so there are situations when not all full edges are incorporated in a nested structure. In the example from Figure 3, D is the inner element of two different elements (C and E) so it is impossible to model with an XML nested structure without introducing enormous amounts of redundancy. The two possible options are: (1) a nested structure for B→C→D, another one for E and F either as E→F or as F→E, and a *keyref* for E→D; (2) a nested structure for B→C, and a second one for E→D and E→F (D and F are both inner elements of E). The connection between C and D is modelled as a loose relationship using *keyref*. Both options are valid and they are considered equivalent in terms of design.

The ConvRel algorithm is thereby transformed to a problem of discovering trees in a directed graph. Identifying a tree in a directed graph is efficiently solved with the depth-first algorithm [1]. The depth-first algorithm is applied to full edges only as those could generate conflicts. In the example from Figure 3, element F has a loose connection to G, but this does not influence the decision of how to model other full connections from F.

The only change to ConvRel applies to a total relationship of type 1:1 [5]. The determining factor in identifying the orientation for the outer element → inner element of the relationship is its similarity to the relational Parent → Child orientation. The Child → Parent modelling for this relationship type is equivalent to the Parent → Child orientation in terms of the nested and compact structure and the length of the XML file [4]. Thus, the conversion algorithm of a relationship to a nested structure is altered in the following way. If Table A participates in a total 1:1 relationship with Table B and also in other relationships with other tables, then the graph will have a bi-directional edge between A and B, which allows this relationship to be modelled in connection with other relationships. If Tables A and B are involved only in this total 1:1 relationship then the relationship is modelled using the similarity to the relational orientation Parent → Child. This change in the ConvRel algorithm creates an additional bi-directional type of edge in the directed graph.

In summary, the conversion algorithm from RDBMS to XML Conv2XML includes the following steps:

1. Determine the 1:1, 1:M, and M:N relationships found in the relational DB.

2. Convert each relationship separately to a nested structure using ConvRel.
3. Construct the adjacency matrix associated to a directed graph of the database.
4. Identify the trees in the directed graph.
5. Construct the XML nested-based Schema.
    a. Create the XML Schema root.
    b. Create XML complex types for each relation, excluding the foreign keys attributes for relations represented in a nested structure.
    c. Create XML complex elements for each XML complex type and set *minOccurs* and *maxOccurs* values according to the participation and cardinality ratios, respectively.
    d. Create the primary and unique keys using *key* and *unique*, including only attributes that have not been eliminated at Step 5.b.
    e. Create foreign keys in the XML Schema root using *keyref* for the relationships not represented as nested structures.

To ensure the XML Schema has a single root, an arbitrary root is created using the name of the database. This root incorporates all elements from the structure in the same way the database contains all tables, relationships, constraints, and indexes from the database. The database in the relational model and the root element from XML has similar functions. Thus, the root element in XML Schema contains definitions for the following elements: (1) elements that are roots of the trees identified by applying the depth-first algorithm; (2) isolated elements that are not connected to any other elements; (3) semi-isolated elements that are connected to other elements through *keyref* references and are not part of any other nested structure.

The child table in a relationship includes the foreign key field, which is a column taken from the parent table. In a nested structure these foreign keys are not required and should not appear as they cause problems in update or delete operations due to referential integrity constraint enforcement. Thus, the foreign key field is eliminated from the child table when it is transformed to an XML element if the relationship between the parent and child table is represented as a nested structure.

5. EXAMPLE

This section applies ConvRel and Conv2XML described previously to the database from Figure 4. The relationship model does not realise M:N relationships physically between two tables but uses an intermediate table to create the original M:N relationships (i.e. workson table). Once the M:N relationships are identified they are represented in the directed graph along with other relationships.

STRUCTURAL CONSTRAINTS CONSIDERED:

EMPLOYEE(1,1) - CHILD(0,M)
BRANCH(0,1) - DEPARTMENT(1,M)
DEPARTMENT(0,1) - PROJECT(0,1)
BUILDING(1,1) - ROOM(1,M)
EMPLOYEE(1,1) - WORKSON(0,M)
PROJECT(1,1) - WORKSON(1,M)

Figure 4: Employees Database

The identified directed graph is then mapped to the adjacency matrix (see Figure 5). The depth-first algorithm is used to identify the trees in the graph. Each tree is represented as a complex nested structure. `Keyrefs` are used to reconstruct links broken by graph splitting when the XML Schema is identified. For each table, a complex type is created using the name concatenated with "Type". The name of the element is the same as the corresponding relation. The XML data file groups elements of the same data type (previous records from the same table) in an outer element with the table name concatenated with an "s". The child table in a relationship includes the foreign key field, which is a column taken from the parent table. In a nested structure these foreign keys are not required and should not appear as they cause problems in update or delete operations due to enforcing the referential integrity constraint.

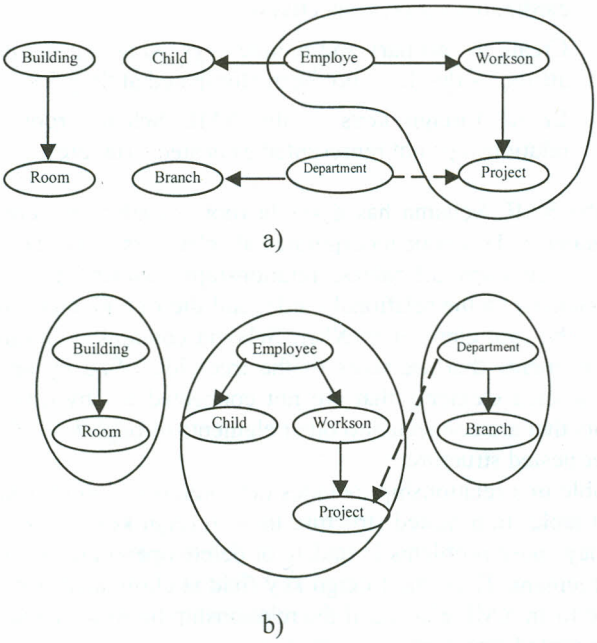

Figure 5: a) Directed graph between tables with the M:N relationship identified. Arrows are directed from the outer element to the inner element. B) Trees obtained from the directed graph. Continue lines are for nested structures. Dashed links represent relationships converted in XML as `keyref` references.

After the elements are created additional constraints (i.e. primary and unique keys) are included in the XML Schema. For inner elements of a nested structure the structural constraints of the former relationships are represented with `minOccurs` and `maxOccurs` restrictions as in Figure 6. The tree root elements have `maxOccurs` equal to `"unbound"`, regardless of the relation's cardinality in the database. This ensures that the tree roots are not inner elements of any other element, except the XML Schema root element. If the eliminated foreign key column is also part of the primary key of the child table as in the example then the primary key in the XML Schema contains the rest of the primary attributes and the constraint is still preserved (a detailed example is presented in [2]).

```
<element name="employees">
 <complexType>
  <sequence minOccurs="0" maxOccurs="unbounded">
<element name="employee" type="r:employeeType" />
  </sequence>
 </complexType>
</element>

<complexType name="employeeType">
 <sequence>
  <element name="employeeid" type="integer"/>
                    <element name="name" type="string"/>
  <element name="childs">
   <complexType>
    <sequence minOccurs="0" maxOccurs="unbounded">
     <element name="child" type="r:childType"/>
    </sequence>
   </complexType>
  </element>
 </sequence>
</complexType>
```

Figure 6: The XML nested structure between Employee and Child elements

In conclusion, ConvRel and Conv2XML are two algorithms for conversion of relationships into XML nested structures focused on preserving their structural constraints. ConvRel translates each relationship individually into a nested XML structure. Conv2XML considers the implications of relationship interconnections in a relational database.

6. RELATED WORK

One of the first approaches to make relational data accessible in XML data files is DB2XML [10]. Either the entire database or a portion is selected through queries for transformation to XML. SilkRoute [6] is considered a general and dynamic tool for exporting relational or object-relational data to XML. It is efficient as it combines the power of the database query engine and features of the XML-QL query language.

Lee *et al.* [8] propose 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 output is an unflattened element-oriented or attribute-oriented DTD. The unflatten process is applied to a single table at a time and 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 is a flat XML structure. Unfortunately, the *nest* factor used in NeT relies on the relational schema and also the actual data stored in the database, which leads to inconsistent results so it is somewhat unreliable. Lee *et al.* [7, 9] have extended the nesting approach to multiple tables, using the CoT algorithm (Constraints-based Translation). It is one of the first approaches that deal 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. A drawback of this approach is that it includes in a nested structure only one child relation. If there are more child relations for a particular parent table, these relationships are represented using IDREF.

Our approach extends the work done by Lee *et al.* [7, 9] in the area of conversion from relational to XML data by including additional elements in the analysis such as: (1) all possible combinations of relational structural constraint ratios; (2) M:N relationships conversion; (3) use of XML Schema instead of DTD which implies additional relational information be transferred in XML; (4) a nested structure can represent several relationships; and (5) algorithm formalisation and its implementation in an efficient tool.

7. CONCLUSION AND FUTURE WORK

This paper introduced a detailed method for representing relational information in a tree-like structure in XML. The algorithms use the advantages of the relational model, such as database normalization, relationships, cardinality and participation ratios, exactness of relational data types, and of the XML Schema, such as a more natural representation in nested structures. ConvRel analyzes each type of relationship and determines 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. Conv2XML is based on the depth-first algorithm that efficiently identifies tree structures in an oriented graph. Thus, the Entity-Relationship Diagram associated with the relational database is transformed so that it can model nested structures and is analysed from the perspective of a directed graph. The conversion algorithms presented in this paper have been implemented in Java version 1.3.1. It extracts the metadata of a DB2 database and based on additional user input for certain semantic cardinality ratios produces a nested XML Schema.

Additional future work includes incorporating the query metric and the XML structure evolution. The research community has not yet agreed upon a standard query method so it has not been included in our method. XML's ability to evolve and alter its structures by adding or subtracting elements, subelements, and attributes is an interesting feature that has not been adequately exploited yet.

REFERENCES

[1] Cormen, T. H. Introduction to Algorithms, Reading, The MIT Press, 2nd edition, 2001.

[2] Duta A. Conversion from Relational Schema to XML Nested-Based Schema - M.Sc. Thesis, University of Calgary, Department of Computer Science, August 2003.

[3] Duta, A., Barker, K., and Alhajj R. Conv2XML: Relational Schema Conversion to XML nested-based schema. In *The 6th International Conference on Enterprise Information Systems (ICEIS'04)*, Portugal, April 2004.

[4] Duta, A., Barker, K., and Alhajj R. ConvRel: Relationship conversion to XML nested structures. In *ACM Symposium on Applied Computing (SAC'04)*, Cyprus, March, 2004.

[5] Elmasri, R., and Navathe, S. B. Fundamentals of Database Systems, Addison-Wesley, 4th edition, 2003.

[6] Fernandez, M., Tan, W.-C., and Suciu, D. SilkRoute: Trading between Relations and XML. In *International World Wide Web Conference (WWW)*, Amsterdam, Netherlands, May 2000.

[7] Lee, D., Mani, M. and Chu, W. W. Effective Schema Conversions between XML and Relational Models. In *European Conference on Artificial Intelligence (ECAI), Knowledge Transformation Workshop (ECAI-OT)*, Lyon, France, July 2002.

[8] Lee, D., Mani, M., Chiu, F. and Chu, W. W. Nesting-based Relational-to-XML Schema Translation. In *International Workshop on the Web and Databases (WebDB)*, Santa Barbara, CA, USA, May 2001.

[9] Lee, D., Mani, M., Chiu, F. and Chu, W. W. NeT & CoT: Translating Relational Schemas to XML Schemas using Semantic Constraints. In *11th ACM Int'l Conf. on Information and Knowledge Management (CIKM)*, McLean, VA, USA, November 2002.

[10] Turau V. Making Legacy Data Accessible for XML Applications. [Internet] Available from <http://citeseer.nj.nec.com/turau99making.html>, 1999. [Accessed January 15th, 2004]

[11] World Wide Web Consortium. XML Schema Part 0, 1, and 2. [Internet] Available from <http://www.w3.org/TR>, 2001. [Accessed January 19th, 2004]

Received: 07 June 2004

Accepted: 08 December 2004