From Entity Relationship to XML Schema: A Graph-Theoretic Approach

Massimo Franceschet, Donatella Gubiani, Angelo Montanari, and Carla Piazza

Department of Mathematics and Computer Science, University of Udine
Via delle Scienze 206, 33100 Udine, Italy

Abstract. We propose a mapping from the Enhanced Entity Relationship conceptual model to the W3C XML Schema Language with the following properties: information and integrity constraints are preserved, no redundance is introduced, different hierarchical views of the conceptual information are available, the resulting XML structure is highly connected, and the design is reversible. We investigate two different ways to nest the XML structure: a maximum connectivity nesting, that minimizes the number of schema constraints used in the mapping of the conceptual schema reducing the validation overhead, and a maximum depth nesting, that keeps low the number of (expensive) join operations that are necessary to reconstruct the information at query time using the mapped schema. We propose a graph-theoretic linear-time algorithm to find a maximum connectivity nesting and show that finding a maximum depth nesting is NP-complete. We complement our investigation with an implementation of the devised translation and we embed the implemented module in a software framework for the conceptual and logical design of spatio-temporal databases.

1 Introduction

The inventors of XML intended to create a document format for web pages and other narrative documents to be read by people. Despite these intentions, the most common applications of XML today involve the storage and exchange of data for use by computer applications. An XML database is a data persistence software that allows one to store data in XML format. Two major classes of XML database exist: *XML-enabled databases*, which map XML data to a traditional database (such as a relational database), and *native XML databases*, which define a logical model for an XML document and stores and retrieves documents according to that method.

The design of a database follows a consolidated methodology comprising conceptual, logical, and physical modeling of the data. This paper is a contribution toward the development of design methodologies and tools for native XML databases. We adopt, in the spirit of [1], the well-understood Entity Relationship model, extended with specialization (ER for short), as the conceptual model for native XML databases. Specialization is particularly relevant in the design of semi-structured data. Moreover, we choose W3C XML Schema Language

Z. Bellahsène et al. (Eds.): XSym 2009, LNCS 5679, pp. 165–179, 2009.

(XML Schema, hereinafter), as the schema language for XML. The major alternative is Document Type Definition (DTD), but DTD is strictly less expressive than XML Schema; in particular, it lacks expressive means to specify integrity constraints, which are fundamental in database design. The contributions of this paper are precisely the following:

1. we propose a mapping from ER to XML Schema with the following properties: information and integrity constraints of the ER model are preserved, no redundance is introduced, different hierarchical views of the conceptual information are permitted, the resulting structure is highly connected, and the design is reversible;
2. we give a graph-theoretic interpretation of the structure nesting problem, that is, the problem of finding the best way to nest the elements corresponding to entities and relationships of the ER schema. We propose a linear-time algorithm to find a maximum connectivity nesting forest, that is, a nesting forest with the highest number of edges, or, equivalently, with the lowest number of trees. This is the forest that minimizes the number of schema constraints used in the mapping of the conceptual schema and hence that reduces the validation overhead to the minimum. Moreover, we show that the problem of finding a maximum depth nesting forest, that is, a nesting forest with the largest value for the summation of node depths, is NP-complete. Such a forest minimizes the number of (expensive) join operations that are necessary to reconstruct the information at query time using the mapped schema and thus reduces the query evaluation time;
3. we implement the devised mapping and embed it into ChronoGeoGraph [2], a software framework for the conceptual and logical design of spatio-temporal XML and relational databases.

The outline of the papers follows. Section 2 contains the basic mapping from ER to XML Schema. The structure nesting problem is investigated in Section 3. Section 4 summarizes related work and proposes future research directions.

2 The Basic Mapping

This section describes the target schema language and provides the mapping of the basic elements of an ER diagram into the target schema language.

2.1 The Target Schema Language

The XML data model is *hierarchical* and *semistructured*. XML elements may be simple elements containing character data or they may nest other child elements, generating a hierarchical tree-like structure. Moreover, elements of the same type may have different structures, e.g., some child elements may be absent or repeated an arbitrary number of times. By contrast, the relational data model is *flat* and *structured*: table attributes must be atomic and tables have a rigid schema. We will deeply exploit the hierarchical and semistructured nature of the XML data model in the proposed encoding of the ER conceptual model.

Representing XML Schema using its own syntax requires substantial space and the reader (and sometimes the developer as well) gets lost in the implementation details. We embed ER schemas into a more succinct target schema language for XML documents (*XSL* for short) whose expressive power lies between DTD and XML Schema. XSL allows sequences and choices of elements as in DTD. XSL extends DTD with the following three constructs:

- *occurrence constraints.* These constrain the minimum and maximum number of occurrences of an item. The minimum constraint is a natural number and the maximum constraint is a natural number or the constant N denoting a finite unbounded natural number. The notation is `item[x,y]`, where x is the minimum constraint, y is the maximum constraint, and item is a single element, a sequence, or a choice. When both x and y are equal to 1, the occurrence constraint may be omitted;
- *key constraints.* If A is an element and KA is a child element or attribute of A, then the notation `KEY(A.KA)` means that KA is a key for element A. Keys composed of more than one attribute are allowed;
- *foreign key constraints.* If A is an element with key KA, B is an element, and FKA is a child element or attribute of B, then `KEYREF(B.FKA --> A.KA)` means that FKA is a foreign key of B referring to the key KA of A.

Recall that DTD allows only the specification of [0,1] occurrence constraints (denoted by ?), [0,N] occurrence constraints (denoted by *), and [1,N] occurrence constraints (denoted by +). Moreover, DTD offers a limited key/foreign key mechanism by using ID-type and IDREF-type attributes. However, the ID/IDREF mechanism is too simple for our goals, for instance it is not possible to restrict the scope of uniqueness for ID attributes to a fragment of the entire document. Also, only individual attributes can be used as keys.

The mapping of XSL into W3C XML Schema is achieved as follows: sequence and choice constructs correspond to *sequence* and *choice* schema elements; occurrence constraints are implemented with *minOccurs* and *maxOccurs* schema attributes; key and foreign key constraints are captured by *key* and *keyref* schema elements, respectively. A full example of XSL definition in given in Figure 2.

2.2 Mapping ER Elements

An ER schema contains entities and relationships between entities [3]. Both can have attributes, which can be either simple, composed, or multi-valued. Some entities are weak and are identified by proprietary entities through identifying relationships. Moreover, some entities may be specialized into more specific entities. Specializations may be partial or total, disjoint or overlapping. Relationships may involve two or more entities; each entity participates in a relationship with a minimum and a maximum participation constraint. Integrity constraints associated with an ER schema comprise multi-valued attribute occurrence constraints, relationship participation and cardinality ratio constraints, specialization constraints (sub-entity inclusion, partial/total, disjoint/overlapping constraints), as well as key and foreign key constraints. We refer to integrity constraints successfully represented in the target schema language as *internal constraints*, whereas *external constraints* are those constraints that cannot be captured in the target schema language due to lack of expressive power and must be validated using an additional schema validator.

The mapping we propose has the following properties:

- it preserves all the information and as much as possible of integrity constraints of the original conceptual schema. An extension to the standard XML Schema validator has been implemented in order to capture the constraints that are missed in the translation due to lack of expressiveness of the target schema language;
- it does not include redundancy in the mapped schema: the original conceptual information is represented only once in the logical XML design;
- it allows different hierarchical views of the conceptual information; this permits to adapt the structure of the logical schema taking into consideration the typical (most frequent) transactions of the database management system;
- it achieves maximum connectivity in the nesting structure used to embed the elements of the conceptual design. As we will show in Section 3, this amounts to minimize the number of schema constraints used in the mapped schema and hence the validation overhead;
- it allows to reverse the design: from the logical XML schema it is possible to go back to the conceptual ER model.

Entities. Each entity is mapped to an element with the same name. Entity attributes are mapped to child elements. The encoding of composed and multi-valued attributes takes advantage of the flexibility of the XML data model: composed attributes are translated by embedding the sub-attribute elements into the composed attribute element; multi-valued attributes are encoded using suitable occurrence constraints. An example is given in Figure 2. As opposed to the relational mapping, no restructuring of the schema is necessary.

Relationships. Each binary relationship has two (left and right) participation (or cardinality) constraints of the form (x, y), where x is a natural number and stands for the minimum participation constraint, and y is a positive natural number or the special character N that represents a finite and unbounded number and stands for the maximum participation (or cardinality ratio) constraint. Typically, x is either 0 or 1, and y is either 1 or N. Hence, we have $2^4 = 16$ typical cases.

Let us consider two entities A, with key KA, and B, with key KB, and a binary relation R between A and B with left participation constraint (x_1, y_1) and right participation constraint (x_2, y_2). We denote such a case with the notation $A \overset{(x_1,y_1)}{\longleftrightarrow} R \overset{(x_2,y_2)}{\longleftrightarrow} B$. The encodings for all the typical cases are given in the following:

1. $A \overset{(0,1)}{\longleftrightarrow} R \overset{(0,1)}{\longleftrightarrow} B$. We have the two possible mappings shown below:

```
A(KA, R[0,1])                        B(KB, R[0,1])
 R(KB)                                R(KA)
B(KB)                                A(KA)
KEY(A.KA), KEY(B.KB), KEY(R.KB) KEY(B.KB), KEY(A.KA), KEY(R.KA)
KEYREF(R.KB --> B.KB)                KEYREF(R.KA --> A.KA)
```

   The two mappings are equivalent in terms of number of used constraints. Notice that the constraint KEY(R.KB) is used to capture the right maximum participation constraint in the left mapping: it forces the elements KB of R to be unique, that is, each B element is assigned to at most one A element by the relation R. Similarly for the constraint KEY(R.KA) in the right solution.

2. $A \overset{(0,1)}{\longleftrightarrow} R \overset{(0,N)}{\longleftrightarrow} B$. The preferred mapping is shown on the left below. The solution on the right uses an extra constraint to capture the left maximum cardinality.

```
A(KA, R[0,1])                       B(KB, R[0,N])
 R(KB)                               R(KA)
B(KB)                               A(KA)
KEY(A.KA), KEY(B.KB)                KEY(B.KB), KEY(A.KA), KEY(R.KA)
KEYREF(R.KB --> B.KB)               KEYREF(R.KA --> A.KA)
```

3. $A \overset{(0,1)}{\longleftrightarrow} R \overset{(1,1)}{\longleftrightarrow} B$. The suggested view is the left one shown below. The element A is fully embedded into element B; hence no foreign key constraint is necessary and the right minimum cardinality holds. The right maximum cardinality is captured by KEY(B.KB). The solution given on the right uses an additional key constraint for the left maximum cardinality as well as an extra foreign key constraint; moreover, it looses the chance to nest the resulting structure.

```
A(KA, R[0,1])                      B(KB, R[1,1])
 R(B)                               R(KA)
   B(KB)                           A(KA)
KEY(A.KA), KEY(B.KB)               KEY(B.KB), KEY(A.KA), KEY(R.KA)
                                   KEYREF(R.KA --> A.KA)
```

4. $A \overset{(0,1)}{\longleftrightarrow} R \overset{(1,N)}{\longleftrightarrow} B$. The suggested mapping is given on the left below. The constraint KEY(R.KA) is used to capture the left maximum cardinality. The opposite solution misses the encoding of the right minimum cardinality, which must be dealt with as an external constraint (added with clause CHECK).

```
B(KB, R[1,N])                            A(KA, R[0,1])
 R(KA)                                    R(KB)
A(KA)                                    B(KB)
KEY(B.KB), KEY(A.KA), KEY(R.KA)          KEY(A.KA), KEY(B.KB)
KEYREF(R.KA --> A.KA)                    KEYREF(R.KB --> B.KB)
                                         CHECK("Right min card")
```

5. $A \overset{(0,N)}{\longleftrightarrow} R \overset{(0,N)}{\longleftrightarrow} B$. We have two symmetrical views using the same number of constraints:

```
A(KA, R[0,N])                 B(KB, R[0,N])
 R(KB)                         R(KA)
B(KB)                         A(KA)
KEY(A.KA), KEY(B.KB)          KEY(B.KB), KEY(A.KA)
KEYREF(R.KB --> B.KB)         KEYREF(R.KA --> A.KA)
```

6. $A \overset{(0,N)}{\longleftrightarrow} R \overset{(1,1)}{\longleftrightarrow} B$. The best solution is the left one below that uses the full nesting of elements. The opposite embedding, on the right, spends an extra keyref constraint and does not achieve element nesting.

```
A(KA, R[0,N])                B(KB, R[1,1])
 R(B)                         R(KA)
   B(KB)                     A(KA)
KEY(A.KA), KEY(B.KB)         KEY(B.KB), KEY(A.KA)
                             KEYREF(R.KA --> A.KA)
```

7. $A \overset{(0,N)}{\longleftrightarrow} R \overset{(1,N)}{\longleftrightarrow} B$. The mapping on the left is the one we propose. The opposite embedding looses the right minimum participation constraint.

```
B(KB, R[1,N])                 A(KA, R[0,N])
 R(KA)                         R(KB)
A(KA)                         B(KB)
KEY(B.KB), KEY(A.KA)          KEY(A.KA), KEY(B.KB)
KEYREF(R.KA --> A.KA)         KEYREF(R.KB --> B.KB)
                              CHECK("Right min card")
```

8. $A \overset{(1,1)}{\longleftrightarrow} R \overset{(1,1)}{\longleftrightarrow} B$. Two symmetrical views are possible:

```
A(KA, R[1,1])                B(KB, R[1,1])
 R(B)                         R(A)
   B(KB)                        A(KA)
KEY(A.KA), KEY(B.KB)         KEY(B.KB), KEY(A.KA)
```

9. $A \overset{(1,1)}{\longleftrightarrow} R \overset{(1,N)}{\longleftrightarrow} B$. The preferred mapping is the one given on the left below. The opposite embedding fails to capture the right minimum cardinality.

```
B(KB, R[1,N])                A(KA, R[1,1])
 R(A)                         R(KB)
   A(KA)                     B(KB)
KEY(B.KB), KEY(A.KA)         KEY(A.KA), KEY(B.KB)
                             KEYREF(R.KB --> B.KB)
                             CHECK("Right min card")
```

10. $A \overset{(1,N)}{\longleftrightarrow} R \overset{(1,N)}{\longleftrightarrow} B$. Two symmetrical mapping are possible:

```
A(KA, R[1,N])                        B(KB, R[1,N])
 R(KB)                               R(KA)
B(KB)                                A(KA)
KEY(A.KA), KEY(B.KB)                 KEY(B.KB), KEY(A.KA)
KEYREF(R.KB --> B.KB)                KEYREF(R.KA --> A.KA)
CHECK("Right min card")              CHECK("Left min card")
```

Notice the use of an external constraint in both solutions to check the minimum participation constraint: this is the only case in the mapping of relationships in which we have to resort to external constraints in the preferred mapping. One may be tempted to add, in the left case, the foreign key `KEYREF(B.KB --> R.KB)` to check the missing constraint. The foreign key would force each B instance to appear under an A instance and hence each B instance would be associated with at least one A instance. Unfortunately, such a foreign key is allowed in XML Schema only if R.KB is a key, which is not possible since a B instance may be associated with more than one A instance and hence there may exist repeated B instances under A. An alternative bi-directional solution is the one that pairs the two described mappings. Such a solution captures all integrity constraints specified at conceptual level. It imposes, however, the verification of an additional *inverse relationship constraint*, namely, if an instance x of A is inside an instance y of B, then y must be inside x in the inverse relationship. Such a constraint is not expressible in XML Schema.

The other six cases are the inverse of some of the above-described solutions. For instance, $A \overset{(0,N)}{\longleftrightarrow} R \overset{(0,1)}{\longleftrightarrow} B$ is the inverse of case 2. We have implemented a similar strategy to map relationships of higher arity and relationships with non-typical participation constraints, e.g., the constraint (2,10).

It is interesting to notice that, thanks to its hierarchical nature, the XML logical model allows to capture more constraints specified at conceptual level than the relational logical model. Indeed, for *all* relationships with one participation constraint equal to (1,N), the minimum participation constraint is lost when mapping the ER model into the relational model; furthermore, some constraints in specialization are also missed [3].

Weak entities and identifying relationships. A weak entity always participates in the identifying relationship with participation constraint equal to (1,1). Hence, depending on the form of the second participation constraint, one of the cases discussed above applies. The key of the weak entity is obtained by composing its partial key with the key of the owner entity and the owner key in the weak entity must match the corresponding key in the owner entity. For instance, suppose we have $A \overset{(0,N)}{\longleftrightarrow} R \overset{(1,1)}{\longleftrightarrow} B$ where B is weak and owned by A. The translation is:

```
A(KA, R[0,N])
 R(B)
   B(KB, KA)
KEY(A.KA), KEY(B.KB, B.KA)
CHECK(B.KA = A.KA)
```

The external constraint `CHECK(B.KA = A.KA)` cannot be avoided. Indeed, suppose we remove the owner key KA from the weak entity B. We need to set a key composed by the pair KA of A and KB of B that now lie at different nesting levels. If we point the selector of the key schema element at the level of entity A, then the field pointing to KB is invalid since it selects more than one node; on the other hand, if we point the selector at the level of entity B, then the field referring to KA is also not valid, since it must use the parent axis to ascend the tree, but such an axis is not admitted in the XPath subset supported by W3C XML Schema[1].

Specialization. The mapping fully exploits the hierarchical nature of the XML data model. Let us consider an entity A with key KA that specializes in two entities B with attributes attB and C with attributes attC. If the specialization is partial-overlapping, then the mapping is as follows:

```
A(KA, B[0,1], C[0,1])
  B(attB)
  C(attC)
KEY(A.KA)
```

Both B and C elements are embedded inside A element. Neither key nor foreign key constraints are necessary. The partial-overlapping constraint is captured by using the occurrence specifiers: an A element may contain any subset of {B, C}. If the specialization is total-overlapping, the regular expression in the first clause must be replaced with `(B, C[0,1]) | C`: any *non-empty* subset of {B, C} is permitted. In case of partial-disjoint specialization the regular expression becomes `(B | C)[0,1]`: either B or C or none of them are included. Finally, a total-disjoint specialization in encoded with the regular expression `(B | C)`: either B or C is present.

The generalization to the case of n sub-entities is immediate in all cases except the total-overlapping case. Let $a_1, \ldots, a_n$ be the sub-entities of a total-overlapping specialization. We indicate with $\rho(a_1, \ldots, a_n)$ the regular expression allowing all non-empty subsets of sub-entities. Such an expression can be recursively defined as follows:

$$\rho(a_1, \ldots, a_n) = \begin{cases} a_1 & \text{if} \quad n = 1 \\ (a_1, a_2[0,1], \ldots, a_n[0,1]) \mid \rho(a_2, \ldots, a_n) & \text{if} \quad n > 1 \end{cases}$$

[1] See the official W3C XML Schema Language specification at the W3C site `http://www.w3.org/TR/xmlschema-1/#coss-identity-constraint`. The authors of [4] seem to have repeatedly missed this point.

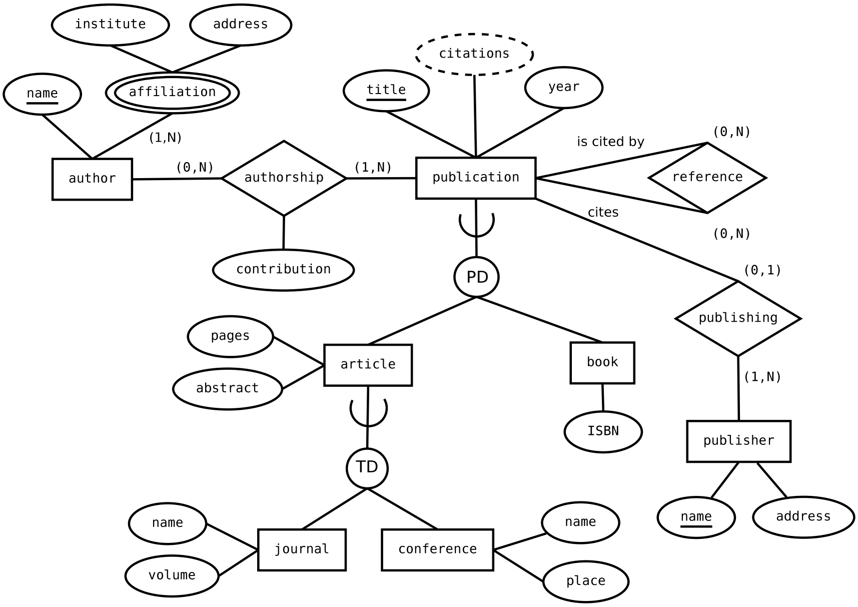

Fig. 1. A citation-enhanced bibliographic database

The size of the expression $\rho(a_1, \ldots, a_n)$ is $n \cdot (n + 1)/2$. Furthermore, the regular expression is deterministic, in the sense that its standard automata-theoretic translation is a deterministic automaton. This is relevant since both DTD and XML Schema content models must be deterministic.

As a full example of the mapping, consider the ER schema in Figure 1. It describes a citation-enhanced bibliography, a typical semi-structured data instance. The mapped schema is shown in Figure 2. The XML Schema version is available at ChronoGeoGraph web site [2].

3 Nesting the Structure

Nesting the XML structure has two advantages. The first advantage is the reduction of the number of constraints inserted in the mapped schema and hence of the validation overhead. The second advantage is the decrease of the (expensive) join operations needed to reconstruct the information at query time. Indeed, highly nested XML documents can be better exploited by tree-traversing XML query languages like XPath. We illustrate these points with the following example. Suppose we want to model a one-to-one relationship direction (`dir`) between entities manager (`man`), with attributes `ssn` and `name` and key `ssn`, and department `dep`, with attributes `name` and `address` and key `name`. A manager

```
publication(title, year, citations, reference[0,N],
            authorship[1,N], (article | book)[0,1])
  reference(title)
  authorship(name, contribution)
  article(pages, abstract, (journal | conference))
    journal(name, volume)
    conference(name, place)
  book(ISBM)
publisher(name, address, publishing[1,N])
  publishing(title)
author(name, affiliation[1,N])
  affiliation(institute, address)

KEY(publication.title), KEY(publisher.name)
KEY(author.name), KEY(publishing.title)
KEYREF(reference.title --> publication.title)
KEYREF(authorship.name --> author.name)
KEYREF(publishing.title --> publication.title)
```

Fig. 2. The mapping of the citation-enhanced bibliographic database

directs exactly one department and a department is directed by exactly one manager. In the following, we propose a flat mapping (on the left) and a nested mapping (on the right) of the corresponding ER fragment:

```
man(ssn, name, dir)                  man(ssn, name, dir)
  dir(name)                           dir(dep)
dep(name, address)                     dep(name, address)
KEY(man.ssn)                         KEY(man.ssn)
KEY(dep.name)                        KEY(dep.name)
KEY(dir.name)
KEYREF(dep.name  --> dir.name)
KEYREF(dir.name --> dep.name)
```

Notice that nesting saves three constraints over five (one key and two foreign keys). Furthermore, suppose we want to retrieve the address of the department directed by William Strunk. A first version of this query written in XPath and working over the flat schema follows:

```
/dep[name = /man[name = "William Strunk"]/dir/name]/address
```

The query joins, in the outer filter, the name of the current department and the name of the department directed by William Strunk. This amounts to jump from the current node to the tree root. An alternative XPath query tailored for the nested schema is given in the following:

```
/man[name = "William Strunk"]/dir/dep/address
```

This version of the query fluently traverses the tree without jumps. The same happens if the query is written in XQuery. We expect that the second version of the query is processed more efficiently by most XML query processors.

In the mapping proposed in Section 2, nesting is achieved by using specializations and total functional relationships, which are relationships such that one of the participating entities has a participation constraint equal to (1,1). While the nesting structure of specialization is uniquely determined, this is not always the case with the nesting structure induced by total functional relationships. Indeed, it may happen that some entity can be nested in more than one other entity, generating a *nesting confluence*. Moreover, *nesting loops* can occur. Both nesting confluences and nesting loops must be broken to obtain a hierarchical nesting structure. This can be done, however, in different ways. Hence, the problem of finding the *best* nesting structure arises.

In the following, we formalize the nesting problem in graph theory. Let S be an ER schema and $G = (V, E)$ be a directed graph such that the nodes in V are the entities of S that participate in some total functional relationship and $(A, B) \in E$ whenever there is a total functional relationship R relating A and B such that B participates in R with cardinality constraint (1,1). Hence, the direction of the graph edges indicates the entity nesting structure, that is, $(A, B) \in E$ whenever entity A contains entity B. We call G the *nesting graph* of S. A nesting confluence corresponds to a node in the graph with more than one predecessor and a nesting loop is a graph cycle. A spanning forest is a subgraph G' of G such that: (i) G' and G share the same node set; (ii) each node in G' has at most one predecessor; (iii) G' has no cycles. Notice that each spanning forest is a valid nesting solution since it contains neither confluences nor cycles. In general, however, a graph has (exponentially) many spanning forests. We are ready to define the following two nesting problems:

The maximum connectivity nesting problem (MCNP). Given a nesting graph G for an ER schema, find a maximum connectivity spanning forest (MCSF), that is, a spanning forest with the maximum number of edges, or, equivalently, with the minimum number of trees;

The maximum depth nesting problem (MDNP). Given a nesting graph G for an ER schema, find a maximum depth spanning forest (MDSF), that is, a spanning forest with the maximum sum of node depths.

Notice that both problems always admit a solution which is not necessarily unique. The MCNP finds a forest that minimizes the number of schema constraints when the forest is used to nest the entities. Indeed, as shown above, each nesting edge reduces the number of constraints in the mapped schema and hence the larger is the number of edges in the nesting graph, the lower is the number of constraints in the resulting schema. On the other hand, the MDNP finds a forest that minimizes the number of join operations that are necessary to reconstruct the information at query time. Indeed, the deeper is a node in the nesting forest, the larger is the number of nodes belonging to the nesting path containing that node, and the lower is the chance of requiring a join operation in a query involving that entity.

The reader might wonder if a spanning forest with the maximum connectivity is also a spanning forest with the maximum depth. The answer in negative, as shown in the example depicted in Figure 3.

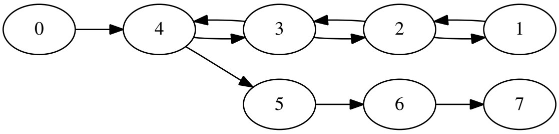

Fig. 3. A nesting graph: a MCSF is obtained by removing edges (1,2), (2,3), and (3,4). It is composed of one tree, 7 edges, and the sum of node depths is 19. A MDSF is the simple path from node 1 to node 7 plus the node 0. It comprises 2 trees, 6 edges, and the sum of node depths is 21. Notice that in this case both solutions are unique.

In the following, we describe an efficient algorithm MCSF that finds a maximum connectivity spanning forest of a nesting graph G. A root node in a directed graph is a node with no incoming edges. Given a node v in a graph G, a reachability tree of v is a directed tree rooted at v containing a path for each node reachable from v in G. Notice that at least one reachability tree exists for any node v, that is, the shortest-path tree from v. The algorithm MCSF works as follows:

1. compute the graph G' of the strongly connected components (SCCs) of G;
2. let $C_1, \dots, C_k$ be the root nodes in G'. For i from 1 to k, compute the reachability tree $T(C_i)$ rooted at some node in C_i in the graph G and remove all nodes in $T(C_i)$ and all edges involving nodes in $T(C_i)$ from G;
3. output the forest obtained by taking the disjoint union of all trees $T(C_i)$ for i from 1 to k.

Figure 4 illustrates an execution of the sketched algorithm.

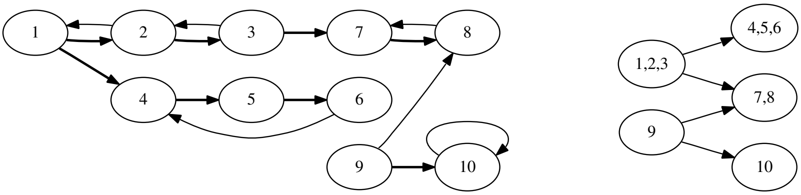

Fig. 4. A nesting graph (left) and its SCC graph (right). Starting the visit from the root component {1,2,3}, the resulting MCSF is shown in bold. A second MCSF is obtained by starting the visit from the root component {9}.

Theorem 1. *MCSF computes a maximum connectivity spanning forest of a nesting graph G in time linear in the size of G.*

Proof. As far as the complexity of algorithm MCSF is concerned, the only crucial point is the computation of the SCCs which can be performed in time $\Theta(|V|+|E|)$ exploiting a double depth-first search [5].

As for correctness, we start observing that the MCNP is equivalent to the problem of finding a spanning forest with minimum number of tree roots, since in a graph with n nodes a forest has $n-k$ edges if and only if it has k roots. We proceed by induction on the number of SCCs of G. In the base case, suppose G has one SCC. Then we can build a spanning forest having just one root, since each node reaches all the nodes of G. In this case our algorithm correctly computes a spanning forest having one root.

As for the inductive step, let us assume that we have proved the correctness of our algorithm on graphs having at most $r-1$ SCCs and let G be a graph with r SCCs. The SCC graph G' of G is an acyclic graph, hence there exists at least one node C in G' without outgoing edges, i.e., C is a SCC of G that does not reach any other SCC. We distinguish two cases: (1) in G' the node C has at least one incoming edge; (2) in G' the node C has no incoming edges. In case (1) a spanning forest of G having minimum number of roots has the same number of roots of a spanning forest of $G \setminus C$ having minimum number of roots. By inductive hypothesis our algorithm is correct on $G \setminus C$, i.e., it determines a spanning forest having the correct number of roots. Moreover, since C has at least one incoming edge in G', we have that C is not used by our algorithm as root node of G'. Hence, our algorithm determines on G a forest having the same number of roots of that determined on $G \setminus C$. This means that our algorithm is correct on G. In case (2), if a spanning forest of $G \setminus C$ having minimum number of roots has k roots, then a spanning forest of G having minimum number of roots has $k+1$ roots (and vice versa). In this case, since C does not reach and is not reached by other components of G, our algorithm determines one tree rooted at C and works on the remaining components as it works on $G \setminus C$. Hence, since by inductive hypothesis it is correct on $G \setminus C$, it is correct also on G. □

On the other hand, the MDNP is hard and, unless P = NP, there is no efficient algorithm that solves this problem.

Theorem 2. *The maximum depth nesting problem is NP-complete.*

Proof. (Sketch) We recall that an optimization problem is NP-complete if the (standard) decision problem associated with it is NP-complete [5]. The decision problem associated with our problem consists in deciding whether a graph has a spanning forest of depth k. In the rest of this proof we denote such a problem as DDNP.

Let $G=(V,E)$ be a directed graph and F be a spanning forest of G. Let the depth of F, denoted by S_F, be the sum of depths of nodes in F. We say that a spanning forest is a chain if it contains $|V|-1$ nodes having one outgoing edge and one leaf. We claim that: (1) $S_F \leq (|V|\cdot(|V|-1))/2$; (2) $S_F = (|V|\cdot(|V|-1))/2$ iff F is a chain. The idea behind the proof is as follows: to maximize the depth of a generic forest the nodes has to be pushed as deep as possible, leading to a chain of nodes whose depth is clearly $(|V| \cdot (|V|-1))/2$.

It is easy to see that DDNP is in NP: given a spanning forest F, its depth S_F can be computed in polynomial time. We show that DDNP is NP-hard by reducing the Hamiltonian path problem – the problem of deciding whether there exists a path that visits each node of a graph exactly once – to it. The above claim allows us to prove that G has an Hamiltonian path if and only if there G has a spanning forest of depth $(|V| \cdot (|V|-1))/2$. Indeed, if G has an Hamiltonian path H, then H is a spanning forest of G. Moreover, since an Hamiltonian path is a chain, we have $S_H = (|V| \cdot (|V|-1))/2$ (point (2) of the claim). On the other hand, if G has a spanning forest of depth $(|V| \cdot (|V|-1))/2$, then by point (2) of the claim G has a spanning forest which is a chain. A chain is nothing but an Hamiltonian path, hence our graph has an Hamiltonian path. □

4 Related and Future Work

There is a vast literature about the integration of XML with relational databases – see [6] for a general comparison of XML and relational concepts and for basic kinds of mappings between them. We found, however, that this literature in partly redundant (even when contributions come from the same authors) and the corresponding citation network is quite disconnected. Nevertheless, we identified three main research themes connected to our work. The research theme closest to the present contribution is that of mapping ER conceptual schemas into some XML schema language. We would like to mention a couple of contributions: Kleiner and Lipeck [7] present a mapping from ER with specialization into DTD that preserves information about the structure and as many constraints as possible. The expressiveness limitations of DTD, however, reduce the number of preserved constraints, complicate the mapping, and do not allow a full reversibility of the design. Elmasri et at. [1] design a system for generating user-customized hierarchical views of an ER schema, creating XML Schemas from such views and generating SQL queries to build XML instance documents from data in a relational database with the same conceptual schema. In particular, they describe an algorithm to eliminate graph cycles in the ER diagram subset selected by the user.

A second related research topic is the translation of relational logical schemas into some XML schema language. It is worth noticing, however, that, from the point of view of integrity preservation, converting relational logical schemas is easier than converting ER conceptual schemas. Indeed, as pointed out in Section 2, the relational model allows to specify fewer integrity constraints than the ER model. An informative contribution in this research line is [8], which includes a survey of different techniques of representing many-to-many relationships within XML and gives many references to related works. In particular, Duta et al. [9] propose algorithms for transforming relational schemas to XML Schema considering the following metrics in this order: constraint-preservation, nested structure, compact structure, length of generated XML file, similarity to the relational structure. The authors state that the incorporation of a query metric (as investigated in this paper) in the translation criteria would be desirable.

A third relevant research thread is the development of conceptual models for XML databases. Proposals include suitable extensions of the ER model, e.g., the ERX model [10], models based on UML [11] and ORM [12], and hierarchical models, like ORA-SS [13]. Besides proposing a new conceptual model tailored for XML, most contributions in this research line give automatic procedures to map the conceptual model to some XML schema language. See [14] for a survey comparing many conceptual models for XML including more references.

Future work comprises the investigation of polynomial-time approximation algorithms for the maximum depth nesting problem and the integration of the algorithms in the translation module. Moreover, we intend to test the translation, validation, and query performance on a realistic case study.

References

1. Elmasri, R., Li, Q., Fu, J., Wu, Y.C., Hojabri, B., Ande, S.: Conceptual modeling for customized XML schemas. Data and Knowledge Engineering 54(1), 57–76 (2005)
2. Gubiani, D., Montanari, A.: ChronoGeoGraph: an expressive spatio-temporal conceptual model. In: SEBD, pp. 160–171 (2007), http://dbms.dimi.uniud.it/cgg/
3. Elmasri, R., Navathe, S.B.: Fundamentals of Database Systems, 5th edn. Addison-Wesley, Reading (2007)
4. Liu, C., Vincent, M.W., Liu, J.: Constraint preserving transformation from relational schema to XML Schema. World Wide Web 9(1), 93–110 (2006)
5. Cormen, T.H., Stein, C., Rivest, R.L., Leiserson, C.E.: Introduction to Algorithms. McGraw-Hill Higher Education, New York (2001)
6. Kappel, G., Kapsammer, E., Retschitzegger, W.: Integrating XML and relational database systems. World Wide Web 7(4), 343–384 (2004)
7. Kleiner, C., Lipeck, U.W.: Automatic generation of XML DTDs from conceptual database schemas. In: GI Jahrestagung (1), pp. 396–405 (2001)
8. Link, S., Trinh, T.: Know your limits: Enhanced XML modeling with cardinality constraints. In: ER, pp. 19–30 (2007)
9. Duta, A.C., Barker, K., Alhajj, R.: Conv2XML: Relational schema conversion to XML nested-based schema. In: ICEIS, pp. 210–215 (2004)
10. Psaila, G.: ERX: A conceptual model for XML documents. In: SAC, pp. 898–903 (2000)
11. Combi, C., Oliboni, B.: Conceptual modeling of XML data. In: SAC, pp. 467–473 (2006)
12. Bird, L., Goodchild, A., Halpin, T.A.: Object role modelling and XML-Schema. In: Laender, A.H.F., Liddle, S.W., Storey, V.C. (eds.) ER 2000. LNCS, vol. 1920, pp. 309–322. Springer, Heidelberg (2000)
13. Dobbie, G., Xiaoying, W., Ling, T., Lee, M.: Designing semistructured databases using ORA-SS model. In: WISE (2001)
14. Necasky, M.: Conceptual modeling for XML: A survey. In: DATESO (2006)