

Automatic Generation of XML DTDs from Conceptual Database Schemas

Carsten Kleiner, Udo W. Lipeck
Universität Hannover, Institut für Informatik
Lange Laube 22, 30159 Hannover, Germany
{ck | ul}@informatik.uni-hannover.de

Abstract: *The goal of this article is to present an algorithm to simplify the automatic generation of descriptions of XML document structures. In particular we show how to obtain a DTD (document type definition) for data whose structure is described by a conceptual data model. An important objective of this translation is to preserve as much structural information from the conceptual schema as possible. This enables partial constraint checking by validating XML parsers and thus simplifies exchange of data between different databases, in particular the import of data in an XML document into another database schema. In detail we present translations of all constructs of an extended entity-relationship model to DTDs and integrate these into an algorithm. By basing the algorithm on conceptual schemas it is very general and may be customized for data in (object-)relational databases as well as data in databases of any other paradigm, e. g. native XML databases.*

Keywords: XML, automatic DTD generation, conceptual data model, EER model, data exchange

1 Introduction

1.1 Motivation

As many industrial and research publications in that area show, the extensible markup language (XML) is emerging as the standard language for data exchange. Among its great advantages are portability, extensibility and the possibility to add *semantics*, i. e. in particular structural constraints, to data within the document itself. Therefore one of the important tasks that XML will solve is the exchange of data and information between different partners. Since most important data nowadays resides in databases (be it relational, object-relational, object-

oriented or other types) it is important to automate the process of generating XML files containing information from a database. Of course one would like to preserve as much information as possible from the database to the XML file, especially constraints. Moreover for newly developed applications data also will have to be exchanged. Since this is likely to be done in XML, it is important to define the structure of these future XML documents in the modeling process already.

Requirements to be derived from the tasks mentioned in the preceding paragraph include the generation of XML documents from the contents of current databases as well as XML generation from data models for future applications and import of XML documents into a database. Since every¹ current database is an instantiation of a certain conceptual database schema and newly developed systems will be defined in terms of conceptual schemas, it is important to define rules and algorithms on how to generate XML files for the contents of systems described by a conceptual model. The conceptual model as sort of the smallest common subset of description of all desired areas should be chosen sufficiently general which is why the ER and EER models will be used in this article. Almost any important modeling construct that may be required is available in those models. The advantage of using a conceptual model in contrast to a certain type of database as in most other publications is flexibility: the transformations defined in this article may be adapted to any database system paradigm. The easiest will be an adaption to relational and object-relational systems where the translation of a conceptual schema to a logical schema is well defined, but even modern systems like XML-based databases may be used. For this type of database system the algorithm in this article may even be used in database design.

Moreover since at the moment of first writing the

¹at least theoretically

only agreed upon standard for describing XML document structures were DTDs those will be used. Limitations due to this choice will be pointed out and solutions will be mentioned that could be used as soon as more advanced languages like XML Schema will become widely used. In contrast to other publications we focus on preserving as many constraints as possible from the schema into the XML document. This is important since in that case a validating XML parser can be used to check the satisfaction of most of the constraints for a given document. This facilitates rather easy import routines and reduces the number of import errors due to constraint violations.

The remainder is organized as follows: after a brief review of recent publications on related topics we start with a motivating example in section 2.1. After that in section 2.2 we present possible translations of ER modeling constructs into DTDs focusing on fewest loss of information possible. In section 3 we integrate these translations into an algorithm which may be used to translate any given ER diagram into a DTD. A brief discussion of information loss by the proposed transformation is also given. Thereafter in section 4 we extend the translations to EER schemas where additional modeling constructs are available. Finally a short summary of results is given before closing with ideas for future research emerging during the development process.

1.2 Related Work

The connection of XML and databases has attracted many researchers in the last few years for its challenges. Work published on this topic can be roughly grouped into two categories: storing (and retrieving) XML files in databases and querying XML files stored in databases. The latter subject is not directly related to our approach since we assume data is stored in traditional databases and can therefore be queried by traditional query languages like SQL. If on the other hand special storage options for XML files are used, querying these files is an important issue. Literature in this area includes [19, 11, 21, 18, 5, 9, 6].

Most work published so far on the connection of databases and XML documents focuses on storing arbitrary XML files in relational databases. Proposals on how to store a given XML file in a relational database can be found in [21, 14, 9]. In addition to storing [17] proposes database constraints to capture semantic information from a given XML file or DTD. Since XML documents may also be considered as a new structured data type storing XML files in object-relational systems has several advantages as described in [16, 1, 5]. The reverse direction

of retrieving data from databases is only considered under querying issues in those articles as well as in [20] which in addition covers performance issues on how to query XML files stored in relations. The first article that defined a presumably reversible mapping between DTDs and relational database systems is [13]. But in contrast to our approach only basic relational features are considered. Constraints like primary key, foreign key and cardinality constraints and object-relational features are not discussed there. Also mappings from object-relational databases to XML are mentioned in [2] as so-called XML-Taggers in the XPERANTO system. There only basic features of object-relational systems are included for querying purposes. Integrity constraints and more advanced features are not considered. Similarly [4] proposes a way to transform UML class diagrams to XML DTDs. In that work no comprehensive algorithm is given and associations in UML are only weakly considered since it is proposed to use XLinks which were still in the development stage at that time. Finally in [10] a detailed formal analysis on the decidability of satisfiability of integrity constraints in XML documents is given.

2 Example and General Ideas

In this section we start with a motivating example which illustrates the translation of a well-known ER schema to a DTD. In the second part translation rules from this example are derived and ideas for translating additional modeling constructs are developed.

2.1 Example

In this section the translation is illustrated by applying it to a well-known ER schema of a company database from [7] which is shown in figure 1. The DTD generated by our algorithm can be seen in figure 2. Even though the DTD is obtained by applying the algorithm from section 3 the example will be presented independent of the algorithm in order to motivate the general ideas.

At first, there is a root element *Company* that represents the entire company database. Since entity *department* is the only entity that is not functionally dependent on another entity it is defined as subelement *Department* of the root element. Also the only *n:m*-relationship in the schema appears as direct subelement *WorksOn* of the root element of the document. The structure of *Department* is given by subelement *Location* for the multivalued attribute *locations* and by subelements for the two functional relationships. Their XML cardinality (either * or +) can be derived from the

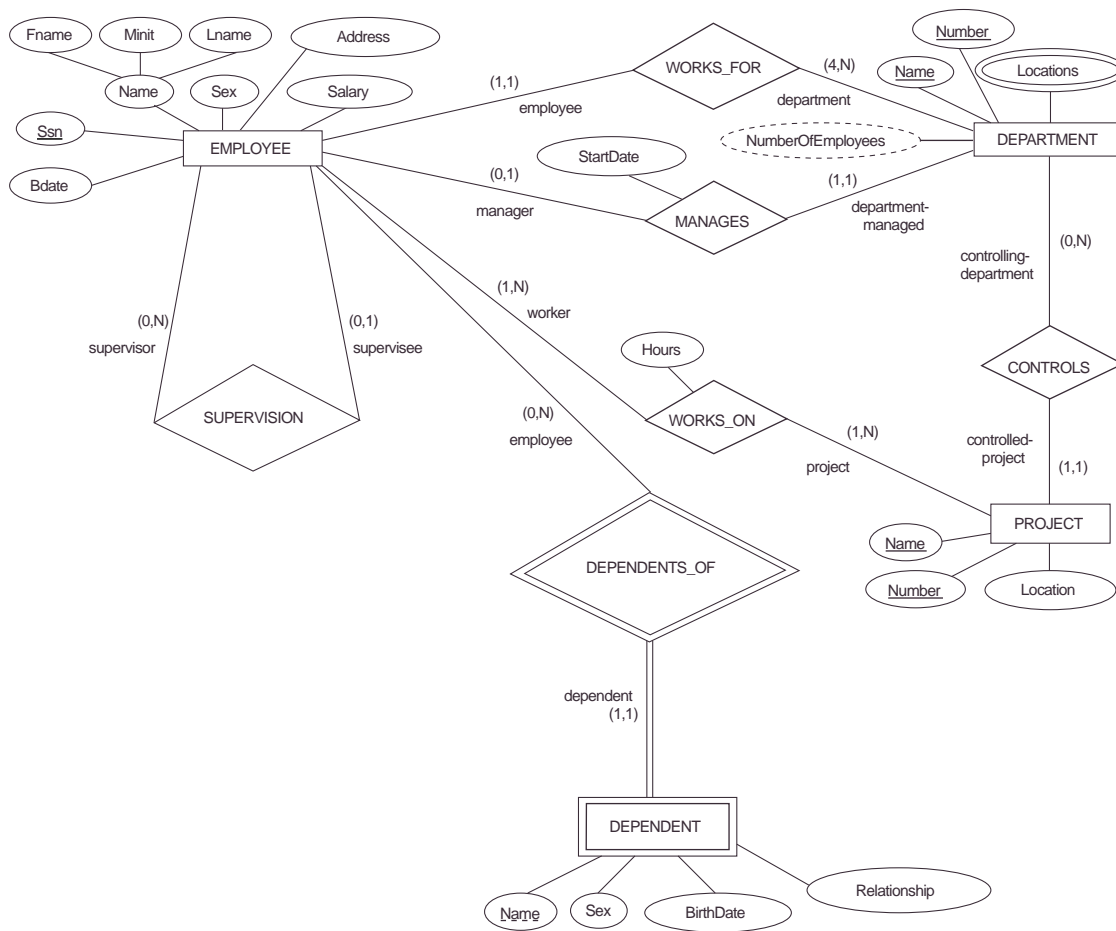


Figure 1. Example ER schema of a company database ([7]), schema name *Company*

participation of the opposite side in that relationship. The 1:1-relationship *manages* is modeled by an IDREF attribute *Manager* (i. e. the role name) for the opposite side and an attribute for the relationship attribute *StartDate*. Subelements of the relationship elements *Controls* and *WorksFor* introduce the elements for the dependent entities *project* and *employee*. The modeling support of cardinality constraints in DTDs is very poor: minimum cardinality (e. g. 4 in *WorksFor*) may be enforced by including as many subelements in the DTD, the last one with an additional +, if no maximum is required; maximum cardinalities can only be expressed by including additional optional subelements up to the desired maximum cardinality. This is very complicated and would not be necessary if using XML Schema since there one could

utilize the *minOccurs* and *maxOccurs* attributes to specify cardinality constraints. In the example we enforce the minimum cardinality by stating *WorksFor* four times.

Features illustrated by the definition of *Employee* are the subelement *Name* for a composite attribute (whose internal structure is omitted for space reasons), the modeling of weak entity types similar to regular 1:n-relationships and modeling relationships to the entity itself. According to the type of *supervision* we use a subelement as before. The difference is in modeling the structure of subelement *Supervision*: we do not use any further subelements in order to avoid redundant information in the XML file since entity *employee* is already taken care of. Thus we use an IDREF attribute to point to the particular supervisees. Since we have subelements of Su-

whereas IDs are globally (i. e. document-wide) unique can be overcome by using the entity name as prefix to the key value in the document instance. Similarly we can use composite strings for composite keys². Of course this is still not an equivalent translation since key datatypes are not preserved and a proper retransformation from DTD to conceptual schema will have to rely on such additional conventions but it is probably the best we can do in the presence of DTDs. If using XML Schema later one will be able to specify key attributes adequately by using the key element clause. Each **composite attribute** *A* in the ER schema will be translated into a subelement *A* of *E*. The subelement *A* itself can be defined in much the same way as we developed *E*: simple attributes of the composite attribute are modeled as XML attributes whereas composite attributes within *A* are translated into nested subelements of *A*. This process terminates once all attributes are simple (usually this is the case in the first nesting step already). Finally **multivalued attributes** *M* in the schema are translated as subelements with cardinality *. The contents of the subelement itself can be constructed like the entity *E* itself: simple ER attributes occur as XML attributes and complex attributes as subelements as described above.

Each **weak entity type** *W* of the ER schema with owner entity type *E* is modeled by a subelement *W* of the identifying relationship between *E* and *W*. This relationship is itself modeled as subelement of *E* with cardinality * (or + if each instance of *E* is guaranteed to have at least one instance of *W* attached; this can be derived from the participation constraint of *E* in the identifying relationship). The identifying relationship itself is modeled as regular 1:*n*-relationships which are described below. All attributes of *W* are modeled in the same way as attributes of strong entities were. Observe that opposed to the ER to relational translation we do not need to create a composite key of *E* and *W* for *W* here since this is implicitly done by using *W* as a subelement of *E*. The partial key of *W* (if existent) is not sufficient as ID attribute since it is only locally unique but not globally as required for IDs. Therefore it may be modeled as required CDATA attribute without enforcing the partial key or by combining the key of *E* with the partial key to obtain a globally unique identifier.

A **1:1-relationship** between entity types *S* and *T* where both participations are total should be translated by merging the corresponding XML elements *S* and *T* into one element. Attributes of the relationship are added to the XML element type. If the relationship is partial on one side *S* and total on the other *T*, the relationship is modeled by inserting an IDREF attribute into element *T* with the name of the role of *S*. All relationship attributes are also

inserted into element *T* in the same way as if they were attributes of *T*. If the relationship is partial on both sides we may use the subelement approach on either element but we have to include the subelement with the optional cardinality specification since it is not guaranteed that a relationship between the entities holds.

Each regular **binary 1:n-relationship** *R* with entity *S* on the 1 side and *T* on the *n* side of *R* can be represented by a subelement *R* of *S* with cardinality * or + depending on the participation of *T* in *R*. If the relationship *R* carries additional attributes the subelement *R* should be attached with all attributes of *R* as described above. Moreover the element *R* consists of a single subelement *T* representing the instance of *T* participating in *R*. The case where *S* and *T* are the same entity type has to be treated differently. Since all instances of *S* will already be included we should not use a subelement of *R* for the *n* side but rather an attribute of type IDREF. This will then point to the instance of *T* participating in *R*. The element *R* itself is defined as an empty element. Integrity problems arising from using IDREF are mentioned in the following paragraph.

Binary n:m-relationships *R* are mapped to top-level elements *R* in the DTD. The element *R* itself consists of two attributes which are defined as IDREF. These references will point to a pair of elements of *R* in the XML document later. By using IDREF XML documents with minimal redundancy are obtained since each tuple occurs only once and is only referenced at other places. The problem with DTDs is that IDREFs are untyped: there is no way to guarantee that in a given XML document the pointers will really point to elements of the desired element type. If an element of a different element type occurs in the document the XML parser will accept that also. Consequently we lose some constraint information by using IDREFs. If using XML Schema these problems are overcome since the construct `keyref` is available which can be used to model foreign key constraints. Alternatively without XML Schema one could use two subelements of *R* which are XPointers themselves. In a concrete XML document they would point to an element in the current document of the matching XML element type which has the ID attribute as defined by the current pair to be modeled from relationship *R*. Nevertheless the integrity problems would be even bigger in that case since a valid XPointer could even point to a completely different element and the document becomes conceptually more complex. Thus we have decided to use the IDREF version. Attributes of relationship *R* are attached to the element *R* either as attributes or as subelements depending on their complexity as described above for attributes of entities.

²As they are CDATA the datatype of the key is not reflected anyway.

Finally **n-ary relationships** of any cardinality³ are modeled similarly to binary $n:m$ -relationships: a top-level element is introduced for each relationship. It consists of n IDREF attributes or XPointer subelements for each of the participating n entity types. Attributes of the relationship are added as described in the preceding paragraph. The same remarks as in the binary case about information loss and usage of XML Schema apply.

This concludes the description on how to translate ER modeling constructs into a DTD since we have given translations for all possible constructs. In section 3.1 we will describe how to put these together in order to obtain a single DTD for a given ER schema. This is necessary since the order of the translation operations as well as the order in which the different parts of the DTD are put together have yet to be defined and will prove relevant.

3 Algorithmic Description of Generation and Example

In this section we will unite the proposed translations from conceptual schema to DTD into an algorithm to be applied to any given conceptual schema. The algorithm may be validated by e. g. applying it to the sample ER schema in section 2.1 to generate a DTD. Moreover we will comment on the quality of the ER to DTD transformation by analyzing how much information is lost if a retransformation to ER is performed.

3.1 Algorithmic Description

The complete algorithm in pseudo-code is depicted in figure 3. As root element we use an element corresponding to the whole database represented by the schema name. Elements for entities at higher levels are added as direct subelements of the root with the objective to preserve as many constraints from the schema as possible. These elements include elements for entities that are not functionally dependent on other entities (i. e. most general) as well as relationships of higher cardinality and arity. Finally in the rare case that certain entities occur only in partial relationships we have to include elements for them at this level since they might otherwise be missed by our algorithm.

In step 3 we start with the detailed contents definition of the elements from the previous step. Whenever we need

³This translation should be used, if they have to be preserved as modeled. In most cases we prefer to translate n -ary relationships into n identifying relationships to an artificial weak entity type as described in chapter 4 of [7].

new elements in the process they are appended to the list of elements in need of detailed definition. The process for each element includes a straight-forward mapping of entity attributes as described in (a), (b) and (e). Subelements are included for relationships to weak entity types (step (c)) and other functionally dependent entities (step (d)) such that a validating XML parser will be able to verify these constraints on a particular document.

In step (f) 1:1-relationships with total participation of the current entity and partial participation of the opposite side are realized by adding all the relationship attributes to the current element as well as an IDREF attribute for the opposite entity type. If on the other hand both participations are total (step (g)) we merge the opposite side entity and the relationship attributes into the current element by attaching all required attributes as XML attributes or subelements as described in the previous section⁴.

Finally in step 4 elements for $n:m$ - or k -ary relationships ($k > 2$) are defined as direct subelements of the root element of the document. The references to the participating entities are realized by IDREFs which can only enforce the existence of matching XML elements but not correctness of the desired element type. This can only be overcome by using a different technique for specifying XML documents such as XML Schema (see also section 5.2). Attributes of these relationships are added in the same way as attributes of entities described above.

3.2 Quality of Transformation and Reversibility

As usual when transforming between models of different semantic richness one should analyze the loss of information incurred by the transformation. The easiest way to do so is by trying to retransform into the original model and find what information can not be reconstructed. Many parts of the conceptual schema may be reconstructed from the DTD but also with several exceptions.

A major restriction is that participation of entities in relationships modeled by IDREF attributes cannot be reconstructed. This is a major restriction since several relationships could not be set up properly in a reconstruction process, e. g. the relationship *Works_On* can be recon-

⁴In the situation that two entities have a 1:1-relationship and occur on the n side of 1: n -relationships to a third entity the algorithm should in step h1 use subelements for both dependent entities. In the case that participation in the 1:1-relationship is total on both sides these entities will not be merged into one element in step (g) since they are not new at that point. The 1:1-relationship will be modeled by IDREFs in step (f) later and thus each entity has its own element. Thus the case where the 1: n -relationships fall together and the 1:1-relationship is between an entity type and itself is also treated correctly.

1. define <SchemaName> to be the root element
2. subelements with cardinality * of root element <SchemaName> are:
 - (a) elements for all entities not occurring on the n -side of any 1: n -relationship
 - (b) elements for all binary $n:m$ -relationships and all k -ary relationships with $k > 2$
 - (c) elements for all entities occurring only on n sides and only partially in relationships
3. while not all entity elements used are defined in the DTD define the next element by (favor entities having identifying relationships to weak entities):
 - (a) introducing a subelement for each composite attribute
 - (b) introducing a subelement with cardinality * or + for each multivalued attribute
 - (c) defining a subelement for each relationship with weak entity types (cardinality as in the ER schema)
 - (d) introducing a subelement for each 1: n -relationship where the current element is on the 1-side with cardinality * or +
 - (e) defining XML attributes for all simple-type attributes of the current element (key attributes as ID, others as CDATA)
 - (f) defining XML attributes for all attributes of 1:1-relationships where the current element participates totally and the other side is partial; define an IDREF attribute for the opposite side of such a relationship; following the same process, if the opposite side entity is a previously seen entity, even if participation is total
 - (g) merging the opposite side entity and all relationship attributes into this element for all 1:1-relationships with total participation on both sides, if the opposite side entity is a new entity
 - (h) for each relationship element obtained from (c) or (d):
 - h1. defining a subelement for the opposite side of the relationship, if it is a new entity
 - h2. defining an IDREF attribute for the opposite side of the relationship, if it is a previously seen entity
 - h3. defining attributes for all relationship attributes as above with entity attributes
4. for all relationship elements from $n:m$ - or k -ary relationships with $k > 2$:
 - (a) introducing an IDREF attribute for all entities participating in this relationship
 - (b) defining attributes for all relationship attributes as above with entity attributes

Figure 3. Algorithm to generate a DTD from an ER schema

structured as binary $n:m$ -relationship but in general it cannot be determined between which entities it holds. To overcome this for a particular XML document one may either obtain the entity information from the key attribute of the XML element pointed to by the reference⁵ or use special annotations maintaining the name of the entity participating in a relationship. Using a special XML element `OppositeEntity` with a single attribute `EntityName` could be used for this purpose. These problems are due to the fact that IDREFs are untyped in XML and therefore consistency cannot be fully enforced even if using annotations. Thus this problem occurs wherever IDREF is used; in XML Schema as described above no such references are necessary and therefore the problem disappears.

Also relationship attributes of 1:1-relationships that were included in the entity element of the total participa-

tion side cannot be distinguished from regular attributes of that entity without additional annotations (e. g. attribute `StartDate` of `manages` in the example in section 2.1). Weak entities (e. g. `dependent`) cannot be reconstructed as weak entities (just as regular entities) since they were modeled in the same way. Role names from the ER schema were not included in the DTD and can therefore not be obtained from DTD or document. Finally composite keys (e. g. combination of attributes `name` and `number` as key of entity `department`) cannot be broken up into their components because they were transformed into a single attribute and there is no way to distinguish them from other attributes. All these problems can be overcome by using special annotations in the XML document if reconstruction of the ER schema may be desired or by using the advanced modeling features of XML Schema as described above.

Other elements of the ER schema that can be reconstructed even though it is not obvious include: multi-

⁵whose value consists of entity name and key value as described above

valued attributes (these are subelements of cardinality * that are empty elements themselves and have no IDREF-attributes), composite attributes (subelements of cardinality 1 that are empty elements themselves and have no IDREF-attributes) and 1:n-relationships with cardinality constraints like *Works_For* which are detected as repeated subelements with additional subelements for the opposite side entity.

Altogether we can say that even without additional annotations an important part of the ER schema can be reconstructed from the DTD. If complete retransformation is desired utilization of additional annotations or vocabulary is required. Because of the flexibility of XML this is no major problem even though constraint checking has to be done explicitly with these annotations and cannot be performed implicitly by a validating parser.

4 Extending the Generation to EER schemas

Since the constructs of the traditional ER model were not sufficient for advanced application modeling numerous extensions have been proposed. The enhanced ER model (or EER model for short) as used in this article has been described in detail in [7] and uses extensions to the ER model similar to the ones presented in [22] and [12]. The additional constructs available are specialization and generalization, both with partial or total participation and disjoint or overlapping, as well as categories or union types. For our purposes we need not differentiate between specialization and generalization since we work on the finished schema where both have the same meaning. The only difference is in the modeling process which is not considered here.

The first and probably most common form of specialization is a **disjoint specialization with total participation**. We use XML elements for both super- and subclass entities where the elements of the subclasses contain a subelement representing the superclass. This mapping is possible since every object belongs to exactly one subclass and the superclass is abstract. Only subclass elements are directly included in the DTD. This enforces the abstract superclass specification (which could not be done in the modeling in [4]) and enables reconstruction of the specialization from the DTD since only in this case the same structured subelement occurs in different elements. For these and all other constructs in this section we assume that the other translations to DTDs are as described in section 2.2 on mapping the ER schema.

For a **disjoint, partial specialization** we need to in-

clude superclass elements in the DTD directly as well since it may be instantiated. Thus we need elements for all entities and include the superclass element directly in the DTD with an optional subelement to be chosen from all possible subclasses. That means, if we have a superclass G with possible subclasses S_1, \dots, S_n in the current specialization we include the term `<!ELEMENT G (S1|. . .|SN)?>` as description of G . This mapping is similar to [4] and is obviously reversible.

Overlapping specializations can be treated in the same way independent of being total or partial. We introduce elements for both superclass G as well as all subclasses S_1, \dots, S_n in the specialization. The element of the superclass is directly included in the DTD with optional subelements for any possible subclass, i. e. the term `<!ELEMENT G (S1?|. . .|SN?)>` is used. Since multiple subclasses are possible we need to include the option for multiple subelements. A possible total restriction in the EER schema is not enforced in this mapping, i. e. even though the specialization was defined to be total a given XML element could include an element of the superclass only. This could be solved by introducing a complicated expression that guarantees that at least one of the possible subelements has to be present. This addition may be useful in certain applications but is not included in our general mapping for simplicity reasons. In [4] this approach was also used but only in the case of two subclasses where it is feasible. Because of the unique way to specify this kind of specialization the mapping is reversible.

A **category with total participation** is modeled similar to a disjoint, total specialization: we have superclasses C_1, \dots, C_n which are categorized by a subclass S . Therefore we need to introduce elements for all superclasses directly into the DTD this time. They consist of a mandatory subelement representing the category S . If on the other hand the **category is partial** we can use the same modeling approach but with an optional subelement S representing the subclass S since not every superclass must belong to this subclass. The mapping of categories is reversible but in the result they can not be distinguished from disjoint specializations with the same participation. This is due to the fact that in this approach no semantic information about super- or subclasses is included. We think that it is nevertheless acceptable because of the rare use of categories; in case retransformation is required the semantic information about super- and subclasses would have to be added.

Finally a minor restriction has to be mentioned: all specializations and categories are assumed to be user-defined. That means that no special support for attribute or predicate defined specializations is included. We think never-

theless that these additions would be possible: the discriminating attributes have to be modeled as XML elements (in contrast to the mapping defined in section 2.2) with one element for each possible discriminating attribute value. The number of distinct values will probably not be too large since for each a different subclass is included in the EER schema. The discriminated entities are included as subelements of the elements corresponding to the discriminating attribute. We do not include this mapping into our algorithm to keep it simple and because the benefits gained by this complication are questionable.

5 Conclusion and Future Work

5.1 Summary

In this article we have presented rules on how to translate constructs from the entity-relationship model into a DTD. We tried to preserve as much information from the schema to the DTD in order to be able to use validating XML parsers for partial constraint checking. The translations for the single constructs were integrated into an algorithm which can be applied to any given ER schema. The translation preserves almost all semantic information since it is fully reversible if using additional XML annotations. We presented a comprehensive example illustrating the algorithm as well. In addition an extension of the translation to the additional modeling constructs of the enhanced ER model (EER model) was presented. Altogether this will facilitate automatic DTD generation for exchange of most data stored in databases today and in future applications.

5.2 Future Work

Several areas for future work and research remain: we are currently working on the implementation of the automatic DTD generation for relational database schemas by using the algorithm given above and reverse engineering approaches that help to reconstruct ER schemas from relational schemas including foreign key constraints. The next step will be the extension to object-relational databases and the EER model. In that area the translation of conceptual schema to logical and physical schema is not as clearly defined yet as in the relational case.

Moreover we will experiment with the use of a validating XML parser for constraint checking. This will be even more important, once more advanced languages for describing XML document structures such as XML Schema become widely used. If all constraints can be transferred to the XML document then an automatic data import into a given database schema should be possible.

Many newly developed applications will be modeled by object-oriented techniques using UML diagrams. Therefore an important addition will be the extension of the translations from EER schemas to UML class diagrams (as were used in [4]. This task should not be too difficult though since most features of class diagrams are also available in EER diagrams and we have already compared some of the features in section 4. Closely related would be extensions to specialized EER models for certain domains such as spatial or spatio-temporal data. This would lead to sublanguages of XML for special datatypes, which was the ultimate motivation for this work (see e. g. [15] for work on exchanging spatial data in XML format that conforms to the OpenGIS specification).

Finally the use of our technique on databases of different paradigms such as object-oriented or native XML should deserve further investigation. Especially the latter where our algorithm could even be used in database design and definition would be a very interesting area for future experiments.

References

- [1] S. Banerjee, V. Krishnamurthy, M. Krishnaprasad, R. Murthy: Oracle 8i - The XML Enabled Data Management System. In *Sixteenth International Conference on Data Engineering (ICDE'00)*, 28 February - 3 March 2000, San Diego, IEEE Computer Society Press, Los Alamitos, CA, 2000, 561-568.
- [2] M. Carey, D. Florescu, Z. Ives, Y. Lu, J. Shanmugasundaram, E. Shekita, S. Subramanian: XPERANTO: Publishing Object-Relational Data as XML. In D. Suciu, G. Vossen (eds.), *Proceedings of the Third International Workshop on the Web and Databases, WebDB 2000, Dallas, Texas, USA, May 18-19, 2000*, 105-110.
- [3] P. P. Chen: The Entity-Relationship Model — Towards a Unified View of Data. *ACM Transactions on Database Systems 1:1 (1976)*, 9-36.
- [4] R. Conrad, D. Scheffner, J. C. Freytag: XML Conceptual Modeling Using UML. In A. Laender, S. Liddle, V. Storey (eds.), *Conceptual Modeling - ER 2000 - 19th Int. Conference on Conceptual Modeling, Salt Lake City, Utah, USA, October 9-12, 2000*, LNCS 1920, Springer-Verlag, Berlin, 2000, 558-571.
- [5] J. Cheng, J. Xu: XML and DB2. In *Sixteenth International Conference on Data Engineering (ICDE'00)*, 28 February - 3 March 2000, San Diego, IEEE Computer Society Press, Los Alamitos, CA, 2000, 569-576.
- [6] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, D. Maier, D. Suciu: Querying XML Data. *Data Engineering 22:3 (1999)*, 10-18.

- [7] R. Elmasri, S. B. Navathe: *Fundamentals of Database Systems (Third Edition)*, 3rd edition. World Student Series, Addison-Wesley, Reading, MA, 2000.
- [8] R. Elmasri, J. Weeldreyer, A. Hermes: The Category Concept: An Extension to the Entity-Relationship Model. *Data & Knowledge Engineering 1 (1985)*, 75–116.
- [9] D. Florescu, D. Kossmann: Storing and Querying XML Data using an RDMBS. *Data Engineering 22:3 (1999)*, 27–34.
- [10] W. Fan, L. Libkin: On XML Integrity Constraints in the Presence of DTDs. to appear in: *Proceedings of PODS 2001*.
- [11] T. Fiebig, G. Moerkotte: Evaluating Queries on Structure with eXtended Access Support Relations. In D. Suciu, G. Vossen (eds.), *Proceedings of the Third International Workshop on the Web and Databases, WebDB 2000, Dallas, Texas, USA, May 18-19, 2000*, 41–46.
- [12] M. Gogolla, U. Hohenstein: Towards a Semantic View of an Extended Entity-Relationship Model. *ACM Transactions on Database Systems 16:3 (1991)*, 369–416.
- [13] G. Kappel, E. Kapsammer, S. Rausch-Schott, W. Retschitzegger: X-Ray - Towards Integrating XML and Relational Database Systems. In A. Laender, S. Liddle, V. Storey (eds.), *Conceptual Modeling - ER 2000 - 19th Int. Conference on Conceptual Modeling, Salt Lake City, Utah, USA, October 9-12, 2000*, LNCS 1920, Springer-Verlag, Berlin, 2000, 339–353.
- [14] C.-C. Kanne, G. Moerkotte: Efficient Storage of XML Data. In *Sixteenth International Conference on Data Engineering (ICDE'00), 28 February - 3 March 2000, San Diego*, IEEE Computer Society Press, Los Alamitos, CA, 2000, 198.
- [15] C. Kleiner, U. W. Lipeck: Web-Enabling Geographic Data with Object-Relational Databases. In A. Heuer, F. Leymann, D. Priebe (eds.), *Datenbanksysteme in Büro, Technik und Wissenschaft - 9. GI-Fachtagung Oldenburg, 7.-9. März 2001*, Informatik aktuell, Springer-Verlag, Berlin, 2001, 127–143.
- [16] M. Klettke, H. Meyer: XML and Object-Relational Database Systems - Enhancing Structural Mappings Based on Statistics. In D. Suciu, G. Vossen (eds.), *Proceedings of the Third International Workshop on the Web and Databases, WebDB 2000, Dallas, Texas, USA, May 18-19, 2000*, 63–68.
- [17] D. Lee, W. W. Chu: Constraints-Preserving Transformation from XML DTD to Relational Schema. In A. Laender, S. Liddle, V. Storey (eds.), *Conceptual Modeling - ER 2000 - 19th Int. Conference on Conceptual Modeling, Salt Lake City, Utah, USA, October 9-12, 2000*, LNCS 1920, Springer-Verlag, Berlin, 2000, 323–338.
- [18] J. McHugh, J. Widom: Query Optimization for XML. In M. Atkinson, M. Orłowska (eds.), *VLDB'99 - Proceedings of the 25th International Conference on Very Large Data Bases, Edinburgh, Sept 7-10, 1999*, Morgan Kaufmann Publishers, San Francisco, 1999, 314 – 326.
- [19] N. Shinagawa, H. Kitagawa, Y. Ishikawa: X2QL: An eXtensible XML Query Language Supporting User-Defined Foreign Functions. In J. Stuller, J. Pokorny, B. Thalheim, Y. Masunaga (eds.), *Current Issues in Databases and Information Systems - East European Conference on Advances in Databases and Information Systems, Prague, Czech Republic, September 5-9, 2000*, LNCS 1884, Springer-Verlag, Berlin, 2000, 251–264.
- [20] J. Shanmugasundaram, E. J. Shekita, R. Barr, M. J. Carey, B. G. Lindsay, H. Pirahesh, B. Reinwald: Efficiently Publishing Relational Data as XML Documents. In A. E. Abbadi, M. L. Brodie, S. Chakravarthy, U. Dayal, N. Kamel, G. Schlageter, K. Y. Whang (eds.), *VLDB 2000 - Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, Morgan Kaufmann Publishers, 2000.
- [21] J. Shanmugasundaram, K. Tuftte, G. He, C. Zhang, D. DeWitt, J. Naughton: Relational Databases for Querying XML Documents: Limitations and Opportunities. In M. Atkinson, M. Orłowska (eds.), *VLDB'99 - Proceedings of the 25th International Conference on Very Large Data Bases, Edinburgh, Sept 7-10, 1999*, Morgan Kaufmann Publishers, San Francisco, 1999, 302 – 314.
- [22] T. J. Teorey, D. Yang, J. P. Fry: A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model. *ACM Computing Surveys 18:2 (1986)*, 197–222.