Making Designer Schemas with Colors

Nuwee Wiwatwattana
U of Michigan
nuwee@eecs.umich.edu

H. V. Jagadish[*]
U of Michigan
jag@eecs.umich.edu

Laks V. S. Lakshmanan
U of British Columbia
laks@cs.ubc.ca

Divesh Srivastava
AT&T Labs–Research
divesh@research.att.com

Abstract

XML schema design has two opposing goals: elimination of update anomalies requires that the schema be as normalized as possible; yet higher query performance and simpler query expression are often obtained through the use of schemas that permit redundancy. In this paper, we show that the recently proposed MCT data model, which extends XML by adding colors, can be used to address this dichotomy effectively. Specifically, we formalize the intuition of anomaly avoidance in MCT using notions of node normal *and* edge normal *forms, and the goal of efficient query processing using notions of* association recoverability *and* direct recoverability. *We develop algorithms for transforming design specifications given as ER diagrams into MCT schemas that are in a node or edge normal form and satisfy association or direct recoverability. Experimental results using a wide variety of ER diagrams validate the benefits of our design methodology.*

1 Motivation

As XML has evolved from a document markup language to a widely-used format for exchange of structured and semistructured data, managing large amounts of XML data has become increasingly important. Since schema gives meaning to data, and determines the validity of queries and updates against the data, recently XML schema design issues have been investigated [2, 3, 15]. Fundamentally, XML schema design has two opposing goals: elimination of update anomalies requires that the schema be as normalized as possible; yet simpler query specification and higher query performance may both usually be obtained through the use of schemas that permit redundancy.[1]

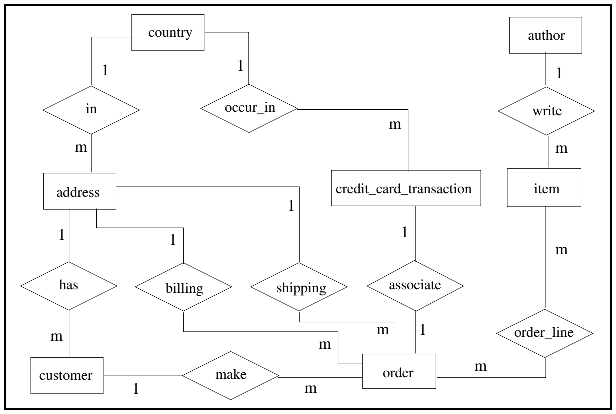

Figure 1. ER diagram of TPC-W benchmark. Attributes suppressed for brevity.

Consider, for example, the ER diagram of the TPC-W benchmark in Figure 1, where attributes are suppressed but can be readily imagined. There is a straightforward way of transforming such an ER diagram to an XML schema (see Figure 2), where entity types from the ER diagram (see Figure 1) are made children of the schema root, relationship types are made children of one of their participating entity types, and all remaining associations are captured through id/idrefs attribute values (indicated in the figure using directed edges). In such a normalized XML schema, update anomalies are avoided at the expense of poor query performance. Queries like "*Q1: list the orders placed by customers having addresses in Japan*" require an XQuery expression involving multiple (id/idrefs) value-based joins, and can be expensive to evaluate. One might argue that this schema design is needlessly *shallow*, as it fails to leverage the nested (tree) structure permitted by XML. Does the tradeoff between opposing design goals still exist when we exploit XML's nesting?

To appreciate this issue, consider, the XML schema of Figure 3. In an XML database that conforms to this schema, an `order` element would be nested under the `customer` element who made the order, which would be nested under the customer's `address` element, which would be nested under the corresponding `country` element. Given the 1:n relationships inherent between relevant entity types in the

---

[*] Supported in part by NSF under Grants No. IIS-0219513 and IIS-0438909.

[1] Ideally, schema design should be completely divorced from physical implementation and hence should be separated from performance concerns. Realistically, physical implementation mimics the given schema, but seeks higher performance only through adding indices, materialized views, and other such auxiliary structures. So the choice of schema has a huge impact on performance.

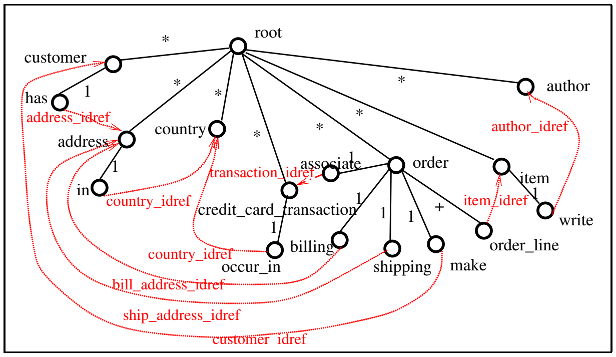

Figure 2. Shallow XML schema from TPC-W.

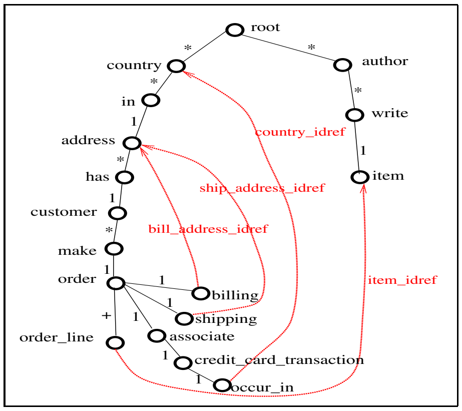

Figure 3. Anomaly-free XML schema from TPC-W.

ER diagram of Figure 1, no data would be duplicated in this XML database, and hence update anomalies would be avoided. Further, our previous query Q1 has both a simple expression:[2]

```
/country[@name = 'Japan']//order
```

and an efficient evaluation strategy using structural joins, which have been shown to be much more efficient than value-based joins [1, 7].

The single-tree structure of XML, however, imposes many limitations. For example, in the XML schema of Figure 3, the `billing` association between `order` and `address` is encoded as attribute values (indicated using a directed edge in the figure) in corresponding elements. This makes many other queries more cumbersome to express and expensive to evaluate. Consider "*Q2: list the orders placed with billing addresses in Japan*":

```
for $o in //order,
  $a in /country[@name='Japan']//address
where $o/billing/@bill_address_idref=$a/@id
return $o
```

A different XML schema design to the one in Figure 3, also without update anomalies, could have nested an `order` under the `address` based on the billing address association. This would have simplified the expression of Q2, but at the expense of Q1.

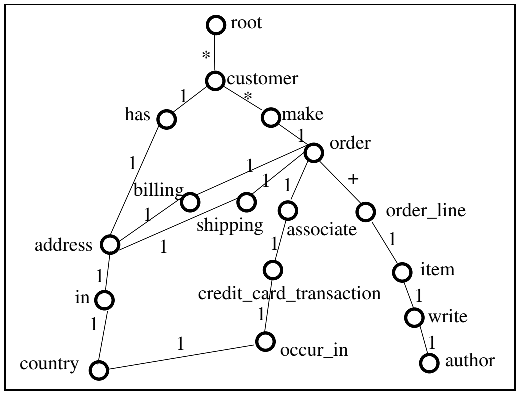

Figure 4. Deep XML schema from TPC-W with data redundancy. The XML tree will be obtained by traversing this graph from the root, and permitting multiple occurrences of elements.

It is, of course, feasible to design XML schemas that allow easy expression, using XPath and XQuery, of a large variety of queries, as well as their efficient evaluation, but this is at the expense of extreme data redundancy. Figure 4 is an example of such a schema: many queries are captured using XPath, but there can be a great deal of redundancy in the representation of various types of `address`, `country`, `item`, and `author` elements. These examples suggest the desirable goal of a schema that is beneficial for both updates and queries might be elusive for the XML model. It is clear that the single-tree structure of XML does not suffice for meeting this goal. But what about XML-like models, such as the recently proposed MCT (multi-colored trees) logical model for XML databases [13], which extends the XML model by supporting multiple tree structures (each in a different color) overlaid on the same set of XML data elements? In this paper, we investigate the following foundational question for XML/MCT databases:

> Given an ER diagram, is it possible to design MCT schemas where (i) update anomalies can be avoided, and (ii) all associations explicitly encoded in the ER diagram can be expressed using structural (XPath-like) expressions?

We show, surprisingly, that this is indeed possible, and we propose a novel schema design methodology for achieving the twin goals of update anomaly avoidance and ease of query expression plus evaluation efficiency, making use of the MCT data model. Our starting point is the design specification, expressed in the form of an ER diagram. In particular, for the TPC-W ER diagram of Figure 1, Figure 5 shows an MCT schema that achieves this goal. We make the following contributions in this paper:

- We formally define the types of associations between

---

[2] We use XPath and XQuery for stating XML queries.

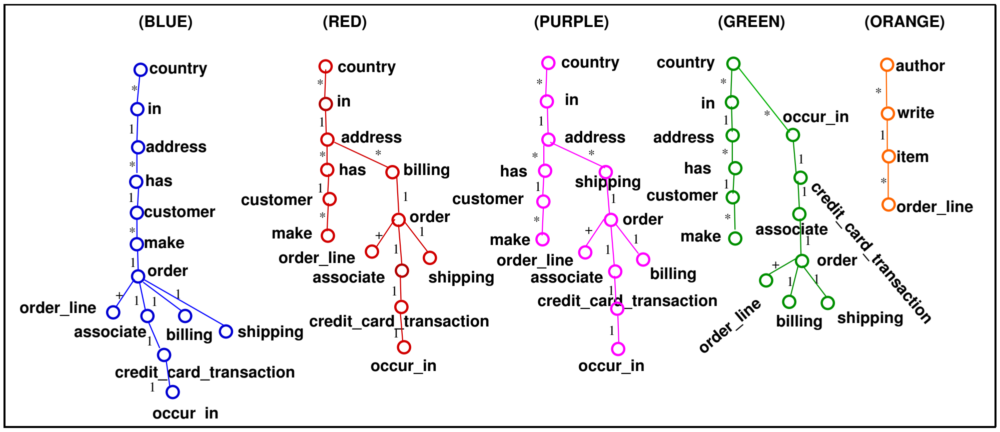

Figure 5. An MCT Schema for the ER Graph of the TPC-W schema in Figure 1. Nodes/edges shown repeated per color for clarity, but are represented/stored once.

data items encoded in an ER diagram that need to be expressed using structural (XPath-like) expressions.

- We define a number of desirable properties for the target logical MCT schema to satisfy:
  - *association recoverability*: no explicit value-based comparisons should be needed to recover any associations between data items encoded in the ER diagram, and *direct recoverability*: an "aggressive" version of association recoverability, which says that certain binary associations need to be captured by a single XPath axis step; these formalize the goal of expression ease and efficient evaluation of queries.
  - *node normalization*: no entity or relationship should be present more than once in any colored tree in an MCT/XML database, and *edge normalization*: no association should be present in more than one color; these formalize the goal of update anomaly-avoidance.
  - *instance independence*: the number of colors should be independent of the database instance, and *color frugality*: this reduces the cost/overhead that comes with maintaining multiple colors.
- We develop algorithms for translating an ER diagram into an MCT schema satisfying the various desirable properties. Not all combinations of properties are achievable at once. We formally show which combinations are feasible, and show that our algorithms achieve those goals.
- Experimental results using a wide variety of ER diagrams validate the benefits of our design methodology.

2 Preliminaries

2.1 ER Diagram and ER Graph

The entity-relationship (ER) model, first introduced by Chen [9], is widely used for conceptual modeling in database design. Over time, many different flavors of ER have emerged. For concreteness, the version we reference in this paper is from Elmasri and Navathe [10].

Specifically, we consider simplified ER diagrams, that contain only entity types, binary relationship types between distinct entity or relationship types, and atomic attributes. Arbitrary ER diagrams can be translated into such simplified ER diagrams by applying simple transformations [20].

For our translation purposes, we will find it convenient to regard a simplified ER Diagram as an undirected graph, called the *ER graph*, with one node corresponding to each entity and each relationship type, and an edge between a pair of nodes whenever they are adjacent in the ER diagram. Node labels and edge labels in the ER graph carry the desired semantic information from the ER diagram.

2.2 MCT: Data Model and Query Language

The multi-colored trees (MCT) logical data model [13] is an evolutionary extension of the XML data model of [11], and enhances it in two significant ways:

- Each data node has an additional property, referred to as a *color*, and nodes can have one or more colors from a finite set of colors.
- An MCT database consists of one or more colored trees $\mathcal{T}_i, 1 \leq i \leq c$, where c is the number of colors.

Essentially, a single colored tree is just like an XML tree. Allowing for multiple colored trees permits richer semantic structure to be added over the individual nodes in the database. In an MCT database, a node belongs to exactly one rooted tree for each of its colors.

MCT databases can be naturally queried using extensions of XPath [5] and XQuery [6]. In particular, each axis step in a path expression needs to be augmented with a color, identifying the colored tree in which the navigation is desired. We refer to this as the multi-colored version of XPath. This suffices for the purpose of this paper. For more details, see [13].

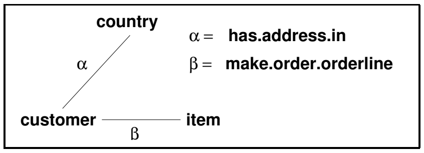

Figure 6. An Example Association Graph.

2.3 MCT: Schema

Informally, a multi-colored XML schema is a set of XML schemas, one for each color, along with possible *inter-color integrity constraints* (ICICs). Formally, an *MCT schema* is a tuple $(N, E_1, \ldots, E_c, \mathcal{C})$, where:

- N is a set of labeled nodes as in an XML schema,
- c is the number of colors,
- each E_i is a set of edges that define an ordered labeled graph on N, and
- $\mathcal{C}$ is a set of ICICs, where each ICIC is a tuple of k edges of the form $(e_{i_1}, e_{i_2}, \ldots, e_{i_k})$, where $k \leq c$ (the number of colors), $i_p \neq i_q$, whenever $p \neq q$, where each edge e_{i_j} is between the same pair of nodes, say $u, v \in N$, but in distinct colors.

Intuitively, an ICIC $(e_{i_1}, \ldots, e_{i_k})$ says that in any valid database instance either the edge between the nodes u and v must be present in all colors $i_1, \ldots, i_k$, or it must be absent in all colors. For example, in Figure 5, there is an edge between nodes `order` and `shipping` in each of the colors blue, red, purple, and green. An ICIC on the corresponding tuple of edges signifies that the association between an `order` element and a corresponding `shipping` element should either be reflected in all four colors or in none at all. Otherwise, there would be an inconsistency between the information encoded in these colored trees. In contrast, there is just one edge between the nodes `make` and `order`, which is present in blue. Accordingly, for this edge there is no associated inter-color integrity constraint.

3 MCT: Desirable Properties, Problem

3.1 Associations, Association Recoverability

A key goal in good schema design is ease of query expression and efficiency of evaluation. But this raises the question: *which queries should be expressible easily and evaluated efficiently?* In this section, we make this precise, based on the notions of *associations*, *association recoverability*, and *direct recoverability*.

Since we are interested in good MCT database schema design, we need to start with an initial design specification, for which we use the time-tested ER model. In the ER model, a pair of entity or relationship types is said to be *associated*, if there is a path between them in the ER graph. More generally, an *association* is any connected sub-graph of the transitive closure of the ER graph. Intuitively, an association graph defines a semantically meaningful tuple of entity/relationship types, with association graph edge labels capturing the path traversed in the ER graph. For example, Figure 6 shows an association graph from (the ER graph associated with) Figure 1. The edge (`customer`, `country`) is labeled `has.address.in`, identifying the ER graph path corresponding to this edge.

When translating an ER graph into an XML or MCT schema, a crucial consideration is the cost to query for associations present in the ER graph. In the XML/MCT model, if associations between elements are captured via values (i.e., id/idrefs), then recovering these associations will use expensive value-based joins. On the other hand, if associations are captured via structural links, then we can recover them via structural joins, which have been shown to be much more efficient [1, 7], and are also easier to express.

Thus, a key desirable property of a (XML or MCT) schema is that every association between data items encoded in the ER graph should be expressible as structural navigation using XPath, possibly extended with colors, as in [13]; no value-based comparisons should be needed! We refer to this property as *association recoverability*.

A major expressive (and computational) benefit of XPath is the use of an ancestor-descendant structural relationship without specification (or retrieval) of any intervening elements. To obtain the full benefit of exploiting XML structure to capture semantic associations, we propose an "aggressive" version of association recoverability. We call *direct recoverability*, the property that an association in the ER diagram can be specified as a *single parent-child or ancestor-descendant axis step (in some single color)* in the multi-colored version of XPath [13]. Not every association is eligible for direct recoverability. We define an *eligible association* between node types u and v as one that is:

1. *Binary* – By its very definition, direct recoverability is impossible for non-binary associations: they cannot be expressed in a single XPath axis step.
2. 1:n – It is easy to observe that an m:n relationship can only be directly recoverable if the schema permits explicit node redundancy, which would risk update anomalies. Note that m:n relationships can arise in many ways, including a single m:n path between u and v, or by a composition of multiple 1:n paths between u and v.

3.2 Node and Edge Normal Forms

A second key goal in good schema design is avoidance of update anomalies. In this section, we make this precise, based on the notions of *node* and *edge normal forms*.

Intuitively, we avoid update anomalies in a single color if no entity or relationship can be present more than once in that color. In the absence of any (semantic) integrity constraints on the data instance, this requires that the schema in that color be represented as a tree. The reason is that if an element is shared by more than one parent element in the schema in any one color, we cannot ensure that in the instance the same node will not be represented more than once in a color. If an MCT schema, confined to each of the single colors present in it, is representable as a tree, then the MCT schema is said to be in *node normal form*. Sometimes, we may have integrity constraints requiring certain associations to be disjoint. E.g., suppose `name` is shared by the parents `author` and `publisher` in a schema in one color. If we have the integrity constraint that author names and publisher names are disjoint, then even though this schema is not a tree in one color, in any valid instance that satisfies this integrity constraint, we are guaranteed to have node normal form satisfied. We do not explore such integrity constraints further in this paper. Node normal form guarantees that information in XML elements is not stored redundantly.

Similarly, we avoid update anomalies across colors if no edge in the ER graph (i.e., binary association) is present in more than one color. If this is the case for all edges in the ER graph, then the MCT schema is said to be in *edge normal form*. If an edge appears in multiple colored MCT trees, then an inter-color integrity constraint (ICIC) is required to manage this edge redundancy. An edge normal MCT schema has an empty set of ICICs.

Note that the node and edge normal forms are *independent*, in that being in one form does not imply being in the other. For example, the MCT schema in Figure 5 is in node normal form (no node is represented more than once in any color) but not in edge normal form (some edges do appear in multiple colors). Similarly, the XML schema in Figure 4 is in edge normal form (since it has only one color), but not in node normal form (in fact, there is a lot of redundancy).

3.3 Problem Statement

We have introduced desirable properties for XML or MCT schemas: association recoverability (AR), direct recoverability (DR), node normal form (NN), and edge normal form (EN), the first two related to the schema design goal of ease of query expression and efficiency of evaluation, and the last two to the avoidance of update anomalies.

In this paper, we characterize which combinations of these desirable properties are achievable and under what conditions, for both single color XML and multi-colored MCT schemas. Further, we develop algorithms for translating ER graphs into MCT schemas with various feasible combinations of desirable properties, while reducing the number of colors used.

In any implementation of MCT, there will be some cost associated with each color. In our own implementation, this cost is low, but non-zero. Therefore, *color frugality* is desirable, while satisfying other properties.

4 Translation from ER to Single Color XML

4.1 Mapping Entity and Relationship Types

Recall that we consider simplified ER diagrams, which have only entity types, binary relationship types, and atomic attributes. In translating such an ER diagram to an XML schema, it is natural to map ER entity and relationship types to XML elements, and ER atomic attributes to XML attributes.

To determine the structural relationships between the elements, the main idea is to construct a schema by traversing the ER graph, covering all nodes. We first preprocess the ER graph and orient the directionality of edges incident on relationship nodes, based on the cardinality of participation of the entity nodes:[3]

- If an entity of type E can participate in multiple relationships of type R, then the edge (E, R) is oriented from node E to node R.
- If an entity of type E can participate in only one relationship of type R, then the edge between E and R can go in either direction.

Any valid instance of the XML schema generated is required to be a tree. The way edges are oriented above helps us ensure that in an instance of the schema, each element will have at most one parent. If an edge were oriented the other way, a single element e of type E could have multiple parent elements r of type R.

Our first observation is that within the framework of (single color) XML, it is not always possible to achieve both node normal form and association recoverability. As an example, consider the TPC-W ER diagram in Figure 1, and focus on the two entity types `order` and `item` with a many-many relationship `order_line`. It is clear that, if we design any XML schema that captures these associations using structure, it is forced to represent either the `order` nodes or the `item` nodes multiple times.

4.2 Mapping Constraints

We consider three main types of constraints in the ER diagram. *Key constraints* are orthogonal to the translation. They merely contribute to keys of the appropriate element types in XML.

ER diagrams have *cardinality constraints*, which dictate what is the min/max number of occurrences of an entity in a relationship. XML schema also permits constraints that require a min/max number of child elements of a given type for a parent element of a given type. However, in the opposite direction, it has just one implicit "cardinality" constraint – every element other than the root must have one parent element. To the extent possible, the XML schema

---

[3] Note that a higher-order relationship type treats lower-order relationship types as its "entities", so the above procedure suffices even in the presence of higher-order relationship types in the ER graph.

cardinality constraint can be aligned with the cardinality constraints specified in the ER diagram. However, there may be mismatches, both in terms of too many and in terms of too few cardinality constraints, given that they are systematic in XML and completely arbitrary in the ER diagram.

ER diagrams also have *participation constraints*, which dictate whether or not all entities must participate in relationships. If the ER diagram has a participation constraint from a parent to a child in XML, this merely says that the parent node has at least one child node of the specified type, which is easily captured in XML schema using a minimum cardinality constraint on (child) element occurrence. If such a participation constraint does not exist, the parent node may have no child, which is easy to capture.

If the ER diagram does not have a participation constraint between a node in XML and its parent, this means that the node could occur without its parent. We must explicitly allow for this, expecting instances not just rooted at A say, but also allowing instances rooted at B, which typically is a child of A. Manipulating such heterogeneous sets of instances is precisely what XML systems do, so we do not have any additional issues to resolve.

4.3 NN and AR

The following theorem gives necessary and sufficient conditions for an ER graph to be translatable into a single color XML schema while achieving the twin goals of node normal form (NN) and association recoverability (AR).

Note that apart from key constraints, cardinality, and participation constraints, no other constraints are assumed to be available for the purpose of this result.

Theorem 4.1 (NN and AR for XML) : *Let G be an arbitrary ER graph. Then G can be translated into an equivalent single color XML schema satisfying both AR and NN iff G satisfies the following conditions: (i) G is a forest; (ii) G does not contain any many-many relationship types or any k-ary relationship types, $k > 2$; and (iii) No entity type in G is on the "many" side of more than one one-many relationship type.*

The key intuitions for the "only if" direction of the proof are as follows. If G is not a forest, then some entity or relationship type has multiple parents, and an instance can be created that would either violate the requirement of the XML database being a tree/forest (if we want to retain AR) or be forced to represent some associations as values (if we want to retain NN). If G contains a many-many relationship then, as discussed previously, a similar choice manifests itself. Suppose G contains a 3-ary relationship type R between entity types E_1, E_2 and E_3. Even if this is a 1:1:1 relationship type, it still poses a problem for a single color XML schema. The issue is that, while R is one-one from each of (E_1, E_2) into E_3, (E_2, E_3) into E_1 and (E_1, E_3) into E_2, the cardinality constraints from E_1 to E_2, for example, could still be many-many, reducing the problem to the previous case. Finally, even when all relationship types are binary and (at most) one-many, if some entity type is on the many side of more than one one-many relationship, it would need to have multiple parents (if we want to retain AR), or at least one of the associations would have to be represented using values (to preserve NN).

For the "if" direction of the proof, it is straightforward to find an orientation of all of G's edges in such a way that all the one-many and one-one relationships are always traversed correctly, i.e., from the "one" side to the "many" side, thus achieving AR. Because of the implicit functional dependency from the "many" side to the "one" side, no instance of any node type will be redundantly represented in a database instance, thus achieving NN.

The theorem shows that the class of ER graphs which can be mapped to a single color XML schema while preserving both AR and NN is rather limited. NN, without AR, can be achieved by covering all nodes of the ER graph with a forest of trees with no overlapping edges. Associations not captured in structure would need to be modeled using explicit id/idref values. Note that edge normal form is not an issue for a single color.

5 Translation from ER to Multi-Color MCT

A key result is that, unlike for single color XML, an MCT schema can indeed concurrently satisfy the desirable properties of NN and AR for arbitrary ER graphs. But not all properties are simultaneously achievable. In particular, there is a fundamental tension between edge normal form (EN) and complete direct recoverability (DR).

5.1 Satisfying NN, EN and AR

Consider the TPC-W ER diagram of Figure 1. It is easy to see that this cannot be translated to single color XML, while preserving AR and NN. The reasons are many, including the many-many relationship type `order_line` between `order` and `item`, and the fact that entity type `order` is on the many side of multiple one-many relationship types, `billing`, `shipping`, `make`. However, it is possible to "cover" this ER graph using multiple colored trees in a database independent way, such that each colored tree satisfies node normal form (NN). One such covering (and resulting MCT schema) is shown in Figure 5, which uses five colors. Since *each* edge in the ER graph is present in at least one color, arbitrary association graphs can be traversed using the multi-colored version of XPath. Not all such coverings are in edge normal form (EN); in particular, the MCT schema of Figure 5 is not. To obtain a covering which is also in EN, care needs to be taken not to traverse the same edge of the ER graph in multiple colors.

```
Algorithm MC ($G$)
  1. If an entity of type $E$ in $G$ can participate in multiple relationships of type $R$, then the edge $(E, R)$ is oriented from node $E$ to node $R$. Leave the remaining edges in $G$ undirected. Mark all nodes unprocessed.
  2. Choose an unprocessed node from a strongly connected component (SCC) of $G$ with no incoming directed edges from other SCCs, and apply a new color to it. If no such node exists, stop. The selected node is called the current start node, and the color applied is called the current color. The set of current roots (of the colored forest under construction) is the singleton set comprising the current start node.
  3. Construct a tree of the current color rooted at the start node by performing a depth-first traversal, following colorable directed edges in the correct (from the one side to the many side) direction, and choosing the orientation for colorable undirected edges when traversed. All nodes and edges traversed have the current color applied (in addition to any colors they may already have). An edge is said to be colorable, if it is not colored, and either (i) the node at the other end does not already have the current color, or, (ii) if it has the current color, it is a current root but not the current start node. A node is marked processed when it has no outgoing or undirected uncolored edges incident, and the edge is either directed or not incident on the current start node.
  4. Find another unprocessed node, if any, from an SCC of $G$ with no incoming directed edges from other SCCs, and if there are other incident edges at least one of them colorable. Call it the start node, add it to the set of current roots and repeat from step 3. If a colorable edge is traversed to reach a current root node $u$, remove $u$ from the set of current roots.
  5. Repeat from step 2.
```

Figure 7. Algorithm `MC`.

If such a covering is done in a careless way, one can end up with a huge number of colors, and pay the cost/overhead of dealing with them. We now provide an algorithm, `MC`, in Figure 7, that takes a simplified ER graph (as described in Section 2.1), and produces a multi-colored MCT schema that is normalized (both NN and EN) and is fully association recoverable (AR).

Intuitively, Algorithm `MC` works by choosing as a start node (in step 2) a candidate for a root of a new color (since a color may create a forest, multiple roots are possible in a color). It then traverses the ER graph (in step 3) looking to add colorable edges in the correct (from the one side to the many side) direction to the current colored tree, possibly orienting any one-one edges encountered. If no new colorable edges can be added to the current tree, a new root is picked (in step 4), if possible, in the same color. If this is not possible, a root is picked in a different color (step 5). In the presence of one-one edges (which are initially not oriented), it is possible that a "root" of a color is encountered during a traversal from a different "root" of the same color. In this case (step 4), the trees are merged.

Since the input graph is finite, no edge is colored twice, and a node is marked processed once all (outgoing and undirected) edges incident on it are colored, the algorithm will eventually terminate. When the algorithm terminates, it is easy to establish that every edge has been colored. In each color, we have a forest of trees, so it is an MCT schema. Each entity/relationship type appears exactly once in the ER Graph, and hence at most once in any color of the MCT schema, so we have a node normal form. By construction, each edge is colored exactly once, so we have edge normal form. Every direct association in the ER diagram is captured as an edge in some color in the MCT schema, so we have association recoverability. As a consequence, we have the following result:

Theorem 5.1 (NN, EN, and AR for MCT) : *Let G be a simplified ER graph. Then Algorithm* `MC` *translates* G *into an equivalent MCT schema satisfying NN, EN and AR.*

5.2 Satisfying NN, AR, DR

The MCT schema produced by Algorithm `MC` satisfies many desirable properties, but the resulting MCT schema does not necessarily satisfy the aggressive version of AR, namely, direct recoverability (DR). Recall that DR is the ability to use a single ancestor-descendant axis step in a single color to retrieve eligible associations.

As a toy example, consider an ER graph with seven nodes $E_1, R_2, E_3, R_4, E_5, R_6, E_7$, where R_2 (resp., R_4, R_6) is a one-many relationship type from entity type E_1 to E_3 (resp., from E_5 to E_3, and from E_3 to E_7). On this ER graph, Algorithm `MC` would produce an MCT schema with two colors. There are many choices of MCT schemas, depending on the first start node that Algorithm `MC` chooses, two of which are: (i) $\{\{E_1, R_2, E_3, R_6, E_7\}, \{E_5, R_4, E_3\}\}$, or (ii) $\{\{E_1, R_2, E_3\}, \{E_5, R_4, E_3, R_6, E_7\}\}$. But no matter which MCT schema is picked, the EN property guarantees that some eligible binary association will not be directly recoverable: in case (i), this is (E_5, E_7), while in case (ii), this is (E_1, E_7).

A desirable objective would be to identify MCT schemas where *every* eligible association is directly recoverable, possibly at the expense of edge normal form, and uses few colors. In the above toy example, such an MCT schema does exist in two colors, namely, $\{\{E_1, R_2, E_3, R_6, E_7\}, \{E_5, R_4, E_3, R_6, E_7\}\}$. This can be obtained by starting from an MCT schema produced by Algorithm `MC`, and adding as many edges as possible to each colored tree, thereby giving up the edge normal form.

This approach, which we refer to as `MCMR` (minimal color, maximal recoverable), is a useful heuristic. But it does not always provide complete direct recoverability. To illustrate this, consider a second toy ER graph (also with seven nodes) $E_1, R_2, E_3, R_4, E_5, R_6, E_7$, where R_2 (resp., R_6) is a one-many relationship type from entity type E_3 to E_1 (resp., E_5 to E_7), and R_4 is a one-one relationship type between E_3 and E_5. In this case, an MCT schema produced by Algorithm `MC` has a single color tree, possibly rooted at E_3 with two branches $\{\{E_3, R_2, E_1\}, \{E_3, R_4, E_5, R_6, E_7\}\}$. However, this MCT schema does not support direct recoverability, in particular, of the eligible association (E_5, E_1). It is easy to verify that an MCT schema needs to have two colors to support complete direct recoverability on this ER graph, *which cannot be obtained by any MCMR-style approach.*

One way of obtaining an MCT schema with complete direct recoverability is to directly leverage Algorithm `MC`. Essentially, there are many different MCT schemas that can be produced by Algorithm `MC`, depending on the choices of start node and colorable edges chosen at each step. We use Algorithm `DUMC` to denote the approach of taking the *disjoint union* of these MCT schemas. By reasoning over the structure of eligible associations in ER graphs, we can establish that the MCT schema produced by Algorithm `DUMC` satisfies NN, AR and DR. However, the number of colors is not necessarily minimized, among all such MCT schemas.

Theorem 5.2 (NN, AR, DR for MCT) : *Let G be a simplified ER graph. Then Algorithm* `DUMC` *translates G into an equivalent MCT schema satisfying NN, AR, and DR.*

6 Experimental Evaluation

We would like to understand how the techniques presented in this paper perform in practice. An immediate issue we face with experimental evaluation is that there is no standard benchmark through which to measure the quality of database design. We adopted a two-pronged strategy: [4]

1. ER Collection: We collected ER diagrams from database textbooks, CASE tools examples, and online sites. One of them is a real-world schema from the Database Derby Contest [17]. For each diagram in this collection, we generated multiple schemas, one for each strategy discussed in this paper. To compare these schemas, we need a benchmark query set whose performance we can measure (except for the Database Derby schema which came with a query set). Since most ER diagram sources do not come with associated query workloads, these query dependent metrics are hard to obtain. To obtain a suggestion of what these may be – we generated a query workload for each ER diagram, based on emulating the XMark [16] set of queries through identifying correspondences between schema elements. We obtained the XMark update query workload from the UpdateX project at the University of Florida [19].

None of these ER diagrams come with associated data sets. So we cannot actually load a data set and run the queries defined above. Rather than manufacture synthetic data sets for this purpose, we restrict our study to an analysis of complexity metrics such as numbers of value joins, color crossings, groupings by value, and duplicate eliminations. The expectation is that these expensive operations predict the overall query performance.

We recognize that the nature of workload may be quite different in various application contexts, and further that our emulation is subjective. However, given the absence of an available workload, we felt this was a better approach than using a workload designed to our liking. We supplement this extensive strategy with the other more intensive strategy described next.

2. TPC-W: The TPC-W benchmark is well accepted, and has a fairly complex schema. For this benchmark, we have both a data set and a query workload, and so are able to conduct more in-depth investigation. We derived an ER diagram (Figure 1) to describe the TPC-W schema, and automatically generated multiple schemas from this ER diagram following the techniques developed in this paper. We implemented each of these on the native XML database TIMBER [12]. We present not just the absolute performance numbers for the TPC-W queries, but also the query complexity surrogates used in the ER diagram analysis. The two subsections that follow describe experiments with each of the above data sets in reverse order.

In each case, we compared several different schemas, both single color and multi-colored.

DEEP: Single color association recoverable but not node normalized (see Figure 4).

SHALLOW: Single color node normalized but not association recoverable (see Figure 2).

AF: Single color node normalized but attempts to maximize both direct and indirect association recoverability (see Figure 3).

The multi-colored schemas we designed are all (except UNDR) both node normalized and association recoverable, but differ as follows:

EN: Edge normalized, but does poorly on direct association recoverability (according to Algorithm `MC`).

DR: Maximizes direct association recoverability, but is not edge normalized (see Figure 5).

MCMR: Minimal color maximal recoverable is local color minimal, and tries to maximize direct recoverability subject to this color minimality. It is not edge normal.

---

[4] Due to space constraints, interested readers are invited to our web supplement [20] for details about ER diagrams and queries used.

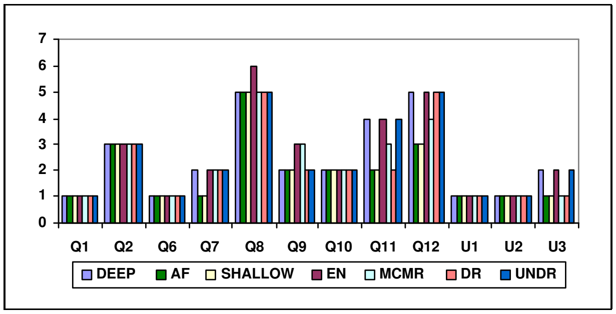

Figure 8. Number of structural joins for TPC-W queries

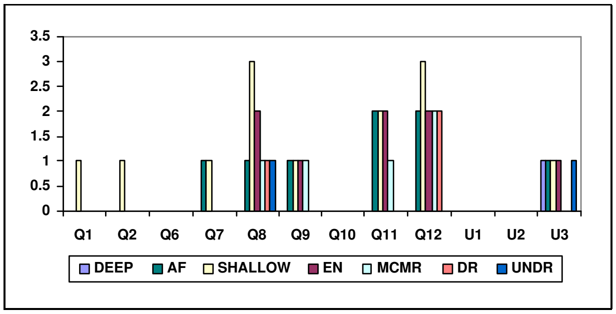

Figure 9. Number of value joins/color crossings for TPC-W queries

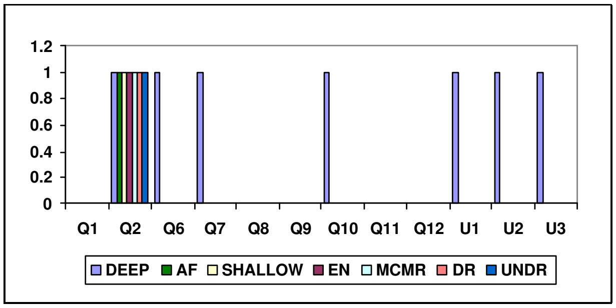

Figure 10. Number of duplicate eliminations/duplicate updates/group by values/ for TPC-W queries

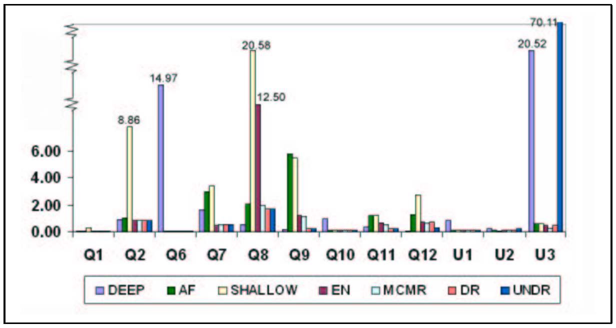

Figure 11. Performance in seconds for TPC-W queries

UNDR: Un-normalized direct recoverable is a multi-colored schema in which direct recoverability, without color crossings, has been selectively increased at the cost of node normalization.

6.1 TPC-W Experiments

From each XML schema, we obtained an XML data file using the ToXgene [4] data generator, orchestrated to contain equivalent content to produce equivalent query results. The experiments were performed on a single processor Pentium III 866MHz equipped with 512 MBytes of main memory, 30 GBytes of disk storage and Windows 2000, running the TIMBER XML database [12]. TIMBER's buffer pool size was set to 256 MBytes. The data page size was 8 KBytes.

The top portion of Table 1 presents the storage requirements of 7 schemas. All node normalized MCT schemas have the same number of elements, attributes and content nodes. EN and MCMR, which have only 2 colors require only slightly more storage than SHALLOW and AF. Storage requirements are higher as more direct associations are covered, by DR, UNDR, and DEEP, respectively. Violating node normalization costs a great deal more in storage than violating edge normalization.

We have 16 queries in the workload (Q1-Q13 and U1-U3), 3 of which are update queries. 4 of these 16 queries were indifferent to choice of schema. Results for the remaining 12 queries are presented here. For each of these queries, we measured the query performance and also several metrics with respect to query expression.

Execution time in seconds of 7 schemas are listed in Table 1. The parentheses in number of results column indicates the duplicate results returned by DEEP (and UNDR, if applicable). SHALLOW requires value joins to recover associations, and hence its performance suffers. DEEP, which is direct association recoverable, always has an outstanding performance, but only if the data redundancy is not an issue, as in Q6, Q7, Q10, U1-U3. With multiple colors, wherever we have direct association recoverability, performance is similar to that of DEEP. Thus, DR and UNDR perform much closer to DEEP than EN and MCMR. Q12 is a query that makes use of the un-normalized structure in UNDR to win over DR. In Q11, DR and UNDR do even better than DEEP due to smaller database size.

In update queries, multi-colored schemas may internally pay the price for color integrity preservation if they are not edge normalized even if they are node normalized. However, this cost is lower than that of a value join or un-normalized constraint maintenance. For example, for U3 involving a single element update, MCMR and DR are less costly than AF, SHALLOW, and EN, in which value joins or color crossings are needed. DEEP and UNDR are expensive here because of the data redundancy. MCMR is faster

Query	Num. Results	DEEP	AF	SHALLOW	EN	MCMR	DR	UNDR
Num.Elements		6,084,002	2,642,111	2,642,111	2,642,111	2,642,111	2,642,111	4,732,855
Num. Attributes		2,177,280	958,148	958,148	958,148	958,148	958,148	1,087,748
Num. Content Nodes		1,729,440	725,806	725,806	725,806	725,806	725,806	829,486
Data Mbytes		1337.99	583.25	583.49	609.94	642.03	747.49	820.57
Num. Colors		1	1	1	2	2	5	5
Q1	1	0.05	0.04	0.30	0.04	0.04	0.04	0.05
Q2	378	0.87	1.05	8.86	0.83	0.83	0.82	0.82
Q6	315(9825)	14.97	0.05	0.05	0.05	0.05	0.04	0.04
Q7	2004(2536)	1.66	2.99	3.41	0.48	0.59	0.59	0.59
Q8	36	0.58	2.10	20.58	12.50	1.94	1.73	1.72
Q9	18	0.16	5.85	5.49	1.18	1.16	0.21	0.21
Q10	1(3)	0.99	0.09	0.08	0.10	0.10	0.10	0.10
Q11	1	0.36	1.21	1.18	0.68	0.50	0.21	0.20
Q12	1	0.04	1.25	2.77	0.73	0.70	0.74	0.32
U1	10(67)	0.85	0.10	0.10	0.10	0.10	0.10	0.12
U2	2(4,6)	0.20	0.08	0.04	0.09	0.08	0.08	0.27
U3	1(6,6)	20.52	0.63	0.62	0.49	0.28	0.48	70.11

Table 1. TPC-W Data Statistics and Query Processing Time in Seconds. The first letter of the query label indicates query type: Q=Read-only, U=Update. The results column indicates the number of results produced for a read-only query, and the number of elements updated for an update query. In parentheses are number of duplicate results for DEEP (and UNDR, if applicable).

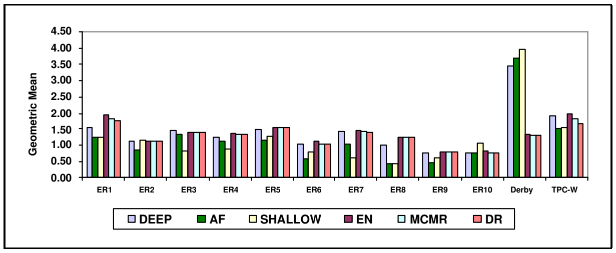

Figure 12. Number of structural joins for ER collection

than DR because it has fewer colors.

Figures 8-10 show 3 query characteristics gathered from the XQuery expressions corresponding to the query run. (We actually gathered many more statistics, but report only these as the most interesting.) The time taken to evaluate a query appears to be almost proportional to the number of value joins or color crossings, with an added amount if there is grouping or duplicate elimination required. There is little correlation between the time to evaluate a query and the number of structural joins. Beyond this, there is little surprise in these figures: schemes with direct association recoverability minimize the number of value joins and color crossings, thus explaining their good performance.

6.2 Collections of ER Diagrams

We took our collection of 11 distinct ER diagrams, ranging in size from 10-30 (entity and relationship type) nodes. For each of these, we generated the six different schemas described above (we excluded UNDR since there were too many subjective ways in which to unnormalize each schema), for a total of 66 different schemas. The maximum number of colors used was 7. The Database Derby came

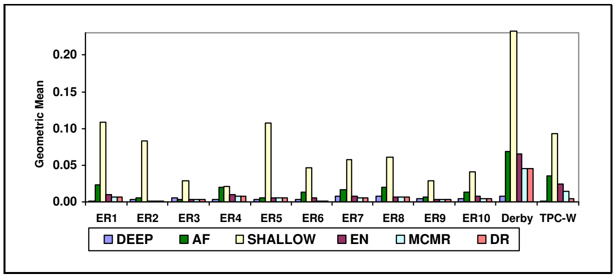

Figure 13. Number of value joins/color crossings for ER collection

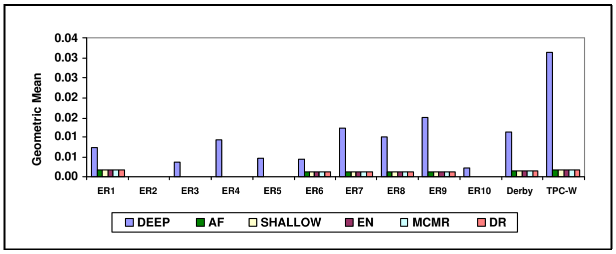

Figure 14. Number of duplicate eliminations/duplicate updates/group by values/ for ER collection

with 20 queries, 8 of which are update queries. For each of 28 queries from the XMark [16] benchmark, 8 of which are update queries, we wrote an equivalent query against each of the 66 different schemas. We computed several metrics for each of these ($28 \times 60 + 20 \times 6 =$) 1800 queries. In Figures 12-14 we report values for the three primary metrics identified in the context of the TPC-W study.

The number of structural joins appears, in most cases, to be traded off against the number of value joins and color crossings. However, there are cases, such as ER10, where

SHALLOW uses more structural joins and more value joins than other techniques, because it breaks down what is a single ancestor-descendant relationship in other schemas into two such relationships with an intervening value join.

Looking at the crucial metrics of value joins and color crossings, we find that SHALLOW requires the most, and DEEP the least. EN requires many more than MCMR and DR. In short, schemas with direct recoverability do much better on this metric. However, un-normalized schemas, such as DEEP, suffer from the cost of duplicate elimination. On balance, MCMR and DR appear to be comparable, and to have the least cost. Since MCMR requires less space than DR, we recommend MCMR for most situations.

While the actual performance numbers quoted are specific to our particular implementation, some general conclusions, applicable to most implementations of MCT, can be drawn. Association recoverability and direct recoverability permit important queries to be evaluated without requiring value joins. As long as such joins are more expensive than other operations (which is true even in relational implementations of XML (or MCT) with a node labeling scheme to enable fast structural joins), the schema design techniques proposed in this paper are of value.

7 Related Work

There has recently been interest in XML schema design. Arenas and Libkin [2, 3] were the first to propose a notion of normal form for XML, i.e., XNF, and provide an information theoretic rationalization for their definition. Schewe [15] studied normal forms for XML taking order into account, using counter-free functional dependencies. Pigazzo and Quintarelli [14] have recently developed an algorithm for designing a normalized XML schema starting from an ER diagram. It is worth noting that the schema produced by their algorithm suffers from the same kinds of limitations as those illustrated in the introduction (Figures 3 and 4). There is also consideration of keys and functional dependencies in the XML context (see, e.g., [8]), but this is orthogonal to our work.

8 Conclusions

We investigate the schema design tradeoff between the goal of query expression ease and evaluation efficiency, and the goal of update anomaly avoidance, for XML-like schemas. We demonstrate that the recently proposed MCT data model, which extends XML by adding colors, can be used to address this problem effectively. We develop algorithms for transforming design specifications given as ER diagrams into MCT schemas that satisfy the twin goals of update anomaly avoidance and query expression ease + evaluation efficiency. Experimental results using a wide variety of ER diagrams on the TIMBER system validate the benefits of our design methodology.

There are many avenues for future work. Often we will be aware of constraints that apply at the instance level, and knowledge of these constraints can be used to obtain better MCT schema designs. Examples include knowledge of tree instances in recursive ER diagrams (such as the section hierarchy in documents). Another interesting direction of future work is to understand how MCT models can be derived from analysis of XML data, in particular the id/idref values that need to encode associations in the XML model.

References

[1] S. Al-Khalifa, H. V. Jagadish, N. Koudas, J. M. Patel, D. Srivastava, and Y. Wu. Structural joins: A primitive for efficient XML query pattern matching. In *Proc. ICDE*, 2002.

[2] M. Arenas and L. Libkin. A normal form for XML documents. In *Proc. PODS*, 2002.

[3] M. Arenas and L. Libkin. An information-theoretic approach to normal forms for relational and XML data. In *Proc. PODS*, 2003.

[4] D. Barbosa, A. Mendelzon, J. Keenleyside and K. Lyons. ToXgene: A template-based data generator for XML. In *Proc. WebDB*, 2002.

[5] A. Berglund, S. Boag, D. Chamberlin, M. F. Fernandez, M. Kay, J. Robie, and J. Simeon. XML path language (XPath) 2.0. W3C Working Draft. Available from http://www.w3.org/TR/xpath20/.

[6] S. Boag, D. Chamberlin, M. F. Fernandez, D. Florescu, J. Robie, and J. Simeon. XQuery 1.0: An XML query language. W3C Working Draft. Available from http://www.w3.org/TR/xquery.

[7] N. Bruno, N. Koudas, and D. Srivastava. Holistic twig joins: Optimal XML pattern matching. In *Proc. SIGMOD*, 2002.

[8] P. Buneman, S. B. Davidson, W. Fan, C. S. Hara, W. C. Tan. Reasoning about keys for XML. In *Proc. DBPL*, 2001.

[9] P. P. Chen. The entity-relationship model: Toward a unified view of data. In *ACM TODS*, 1976.

[10] R. Elmasri, and S. B. Navathe. Fundamentals of database systems. Addison Wesley, 4th ed., 2003.

[11] M. F. Fernandez, A. Malhotra, J. Marsh, M. Nagy, and N. Walsh. XQuery 1.0 and XPath 2.0 data model. W3C Working Draft. Available from http://www.w3.org/TR/query-datamodel/.

[12] H. V. Jagadish, S. Al-Khalifa, A. Chapman, L. V. S. Lakshmanan, A. Nierman, S. Paparizos, J. M. Patel, D. Srivastava, N. Wiwatwattana, Y. Wu, and C. Yu. TIMBER: A native XML database. *The VLDB Journal*, 11(4):274–291, 2002.

[13] H. V. Jagadish, L. V. S. Lakshmanan, M. Scannapieco, D. Srivastava, and N. Wiwatwattana. Colorful XML: One hierarchy isn't enough. *Proc. SIGMOD*, 2004.

[14] P. Pigazzo and E. Quintarelli. An algorithm for generating XML schema from ER schemas (Extended Abstract). In *Proc. SEBD*, 2005.

[15] K.-D. Schewe. Redundancies, dependencies, and normal forms for XML databases. In *Proc. Australian DB Conference*, 2005.

[16] A. R. Schmidt, F. Waas, M. L. Kerste, D. Florescu, I. Manolescu, M. J. Carey, and R. Busse. The XML benchmark project. Technical Report INS-R0103, April 2001.

[17] The Database Derby. 4th E.R. Conference, Chicago, IL., USA. October 29, 1985.

[18] TPC-W, A Transactional Web E-Commerce Benchmark. Available at http://www.tpc.org/tpcw/.

[19] UpdateX, the XQuery Updates Project (GALAX). Available at http://www.cise.ufl.edu/research/mobility/.

[20] N. Wiwatwattana, H. V. Jagadish, L. V. S. Lakshmanan, and D. Srivastava. Making designer schemas with colors: web supplement. Available at http://www.eecs.umich.edu/db/timber/mct/index.html.