

Preserving Composition in XML Object Relational Storage

Eric Pardede, J. Wenny Rahayu
Dept. of Comp. Science & Comp. Engineering
La Trobe University
Bundoora VIC 3086, Australia
{ekpardede, wenny}@cs.latrobe.edu.au

David Taniar
School of Business Systems
Monash University
Clayton VIC 3800, Australia
David.Taniar@infotech.monash.edu.au

Abstract

XML data can be stored in different types of databases including Object-Relational Databases (ORDB). Using ORDB, we get the benefit of relational maturity and the richness of object-oriented modeling. One modeling concept that can be captured is composition hierarchy, which is a special type of relationship that shows an exclusive existence-dependent "part-of" relationship. This type of relationship frequently occurs in XML data, yet very often when the data is stored in a database repository, the "part-of" relationship is either flattened or split into an entirely separate table.

In this paper we propose a model to preserve composition type in XML data into ORDB using the concept of row types. We use the Semantic Network diagram to represent the composition hierarchy in XML data. The composition hierarchy is divided into three types, namely single row composition, multi rows composition, and multi level composition. Each of these composition types will then be transformed into storage in an ORDB environment.

1. Introduction

XML (eXtensible Markup Language) is a document description metalanguage that is used to represent large scale data and documents in the World Wide Web [12]. For that reason alone, XML requires efficient database storage to keep its data.

Object-Relational Database (ORDB) is increasingly popular as the database storage for XML Data [6]. Its popularity relates to its ability to capture the object-oriented modeling semantic and the maturity of relational implementation.

Many works have been proposed to map the DTD and XML Schema into Object-Relational Schema [6, 7, 13]. The mapping includes different data structures and relationships. One type of relationship is *composition*. It is not the same with aggregation. While aggregation is identified as a relationship in which a composite object ("whole") consists of other component objects ("parts")

[11], composition has two additional constraints as follows.

- *Non-shareable*. In XML documents, we encounter many cases when a document can only be the part of one and only one other document. It represents the non-shareable constraint. Treating composition as usual aggregation enables other documents to own the "part" document and thus, violate the conceptual semantic.
- *Existence-dependent*. In XML documents, we also find cases where a document can only exist with another document. It represents the existence-dependent constraint. Treating composition as aggregation will enable a "part" document to remain in existence even though the "whole" document has been removed.

The reasons above have motivated us to differentiate the composition and the aggregation. This work will focus on the preservation of the composition hierarchy since other works have focused on aggregation.

In ORDB, composition can be implemented as the attribute of row type. It is the constructed data type that contains a sequence of attribute names and their data types [5, 8]. Row type attributes will be fully dependent and exclusive to the object that owns them.

It is the aim of this paper to propose models for preserving a composition in XML into ORDB, and in particular we introduce the use of row type. The work is performed by two mapping steps. First is the mapping from the conceptual model using Semantic Network Diagram to logical model using XML Schema. We extend the algorithm in [3] by proposing a mapping method for composition hierarchy. Second, a logical model is mapped into physical implementation using SQL in ORDB.

2. Background

In this section we briefly show some existing works on the implementation of composition hierarchy for XML Data. We also provide a brief knowledge foundation on the semantic network model before we use it in our proposed method.

2.1. Existing Implementation of Composition Relationship in XML

Some works have tried to preserve design modeling and store XML data into database based on relational model including traditional relational database and ORDB. However, there is no work that utilizes row data types that was introduced by SQL3 [5, 10] and is enriched in SQL4 [8, 10]. Very often when the XML data is stored in a database repository, the "part-of" relationship is either flattened or split into an entirely separate table.

[4] presents a simple mapping of XML data into relational tables. In this work, they treated XML documents as graphs with edges and leaves. The edges represent the relationship while the leaves represent the values. The composition in this work is mapped into separate flat tables by using composite keys. It has then diminished the composition semantic.

[1] proposes comprehensive mapping from DTD into OO Schema and implementing the results into tables. Nevertheless, in the implementation stage again the composition type is flattened.

[12] also works in the mapping of DTD into relational tables. They develop an algorithm and a prototype that convert the XML documents to tuples, translate the semi structured XML queries to SQL queries, and then convert the results to XML data. This work enlists limitation of the usage of relational database for XML documents. One of those highlights the limitations of implementing composition in pure relational table since this database is unable to have set-valued attributes.

[6] proposes mapping from XML Schema into OO/OR Schema. The work is based on previous works on mapping to Relational Schema. It compares how the mapping into relational schema can be changed into mapping to object-relational schema. The result is that it does not cover the unique properties of an object-relational model such as different types of relationships including composition.

[7] proposes mapping from DTD into Object-Relational Schema. The main contribution of the work is the use of a hybrid database where the users can select certain attributes to be stored in ORDB and others to be stored as it is (as XML data). It does not show mapping for different kinds of relationship and data structures.

Finally [13] proposes the mapping of aggregation type relationship of XML Schema to ORDB. The work correctly identifies a different type of aggregation that can be captured in XML Schema. However, the ORDB implementation still diminishes the "part-of" relationship meaning since the XML data is stored separately in tables, with cluster or nested tables.

We find that the existing works either have not preserved the composition or have preserved it as a usual aggregation. Therefore, in this work we are going to propose complete mapping to preserve the composition hierarchy from the conceptual level to the implementation.

2.2. Semantic Network: an overview

In this paper we use the Semantic Network Method [3] to model the conceptual level of XML Documents structures. The diagram can be viewed as a richer alternative of W3C XML Data Model [14]. The semantic network Method is divided into the semantic level and the schema level. The semantic level develops a specific diagram from the XML document structures. The schema level maps the diagram into syntax and structure formalism such as XML Schema.

The semantic network diagram is divided into four major components: nodes, directed edges, labels, and constraints. In Figure 1, there are 5 nodes: A, B, X, Y, and Z. The first two are complex nodes while the rest are basic nodes. There are four directed edges representing the semantic relationships between the objects. We have different labels corresponding to each edge. For example "p" indicates in-property relationship and "a" represents aggregation relationship. Finally, there are constraints added in nodes or edges such as uniqueness, cardinality, adhesion, ordering, etc. This diagram can show the conceptual design of the XML documents more completely than XML data model or UML [2].

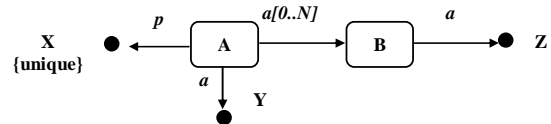


Figure 1. Semantic Network Diagram

In the schema level the semantic network method maps the four components of the diagram into XML Schema, which is mainly concerned with element/attribute declarations and simple/complex type definitions [3].

In the next section we show how we extend the schema level mapping to capture the composition relationship and then we follow through to the implementation of the XML Schema in ORDB.

3. Proposed Method

In this section we propose the mapping method of composition relationship in XML structures into ORDB using Row Type. For the semantic network diagram, we use label "c" to represent composition since previous works have not differentiated composition from aggregation. In the first step we map the "part" component

as the complex type inside the “whole” component. This process has disabled other components to own this particular “part” component.

For the second step, we directly map the outer complex type as the table and the inner complex type as the row type attribute. For multiple-row we use multiset data types standardized in SQL4 [8]. Another type of collection is also possible depending on the requirements for ordering and duplication semantic.

3.1. Implementation of Single Row Composition

Mapping Method. If the “part” component of the composition is single, we can use the single row (see Fig.2). The state network diagram as the conceptual model is mapped into XML schema and then we implement in ORDB tables. The symbol “c” in the diagram is used for composition type.

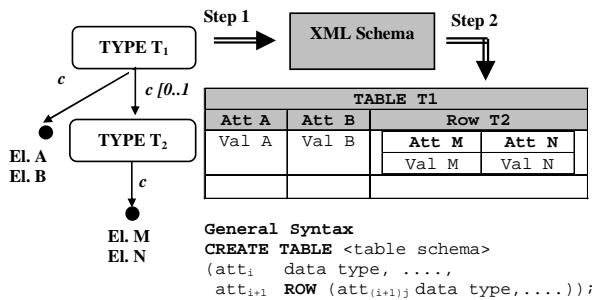


Figure 2. Composition for Single Row

Rule 1:

Step 1: For two types namely T₁ and T₂ with elements and/or attributes (A,B) and (M,N) respectively, if T₁ is composed by at most one T₂, implement T₁ as an outer complex type and T₂ as an inner complex type.

Step 2: For two complex types namely T₁ and T₂ with elements and/or attributes (A,B) and (M,N) respectively, if T₁ is composed by at most one T₂, implement T₂ as a single row attribute of table T₁.

Transformation result is $Table T_1 (A, B, Row T_2 (M, N))$.

Example 1:

Type AUTHOR is composed by at most one type ADDRESS (see Fig.3). The composition type will be mapped into a single row attribute in ORDB table.

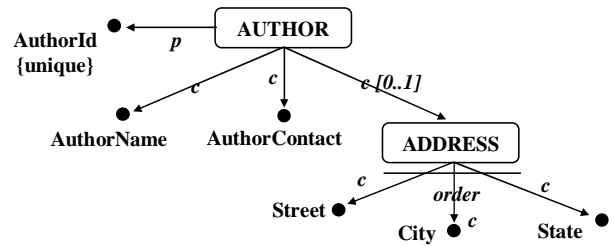


Figure 3. AUTHOR-ADDRESS Composition

Transformation into XML Schema:

```
<xsd:complexType name = "AUTHOR_Type">
<xsd:element name = "AuthorName" type =
"xsd:string"/>
<xsd:element name = "Contact" type =
"xsd:integer"/>
<xsd:element name = "ADDRESS" minOccurs = "0"/>
  <xsd:complexType>
  <xsd:sequence>
    <xsd:element name = "Street" type =
"xsd:string" minOccurs = "0"/>
    <!--and other elements -->
  </xsd:sequence>
  </xsd:complexType>
<xsd:attribute name = "AuthorId" type =
"xsd:string" use = "required"/>
</xsd:complexType>

<key name="AUTHOR_authorId">
  <selector xpath = "AUTHOR">
  <field xpath="@AuthorId"/></key>
```

Transformation into ORDB:

```
CREATE TABLE Author
(AuthorID CHARACTER VARYING(20)
CONSTRAINT Author_author_id_pk PRIMARY KEY,
AuthorName CHARACTER VARYING(2),
Contact NUMBER,
Address ROW (Street CHARACTER_VARYING(50),
City CHARACTER_VARYING(30),
State CHARACTER_VARYING(5)));
```

Analysis. The proposed method from the conceptual to the implementation level has captured the real semantic of composition hierarchy. The first contribution is in the transformation of semantic network diagram to XML Schema. We come up with XML Schema where the “part” component is defined as complex type inside the “whole” type element. By doing this we avoid other element type to share the particular “part” component (non-shareable constraint). We also ensure that on removal of the “whole” type, we remove all “part” components that are defined inside it (existence-dependent constraint).

```
<xsd:complexType name = "C_Type">...
<xsd:element name = "E_Name" minOccurs = "0"/>
  <xsd:complexType>...
</xsd:complexType>
</xsd:complexType>
```

The second contribution is the transformation of the XML Schema to the ORDB in the form of Row Type

attribute. The mapping is straightforward. The outer complex type in XML Schema is mapped into the main table with its attributes and elements as the table attributes. The inner complex type in the XML Schema is also mapped as an attribute of the table with row data type. We use the SQL syntax **TABLE (...ROW())**.

In example 1, the *address* data has composition relationship to the *author*. It means that the *address* can only be attached with the *author* data and its existence is dependent on the *author* data. Using existing mapping, the *address* will be mapped as ordinary type outside the *author*. In our proposed method, we define *address* data inside the *author* to ensure the composition constraint.

3.2. Implementation of Multiple Row Composition

Mapping Method. If the “whole” component can have more than one “part” component of the same type, we use multiple row (see Fig.4).

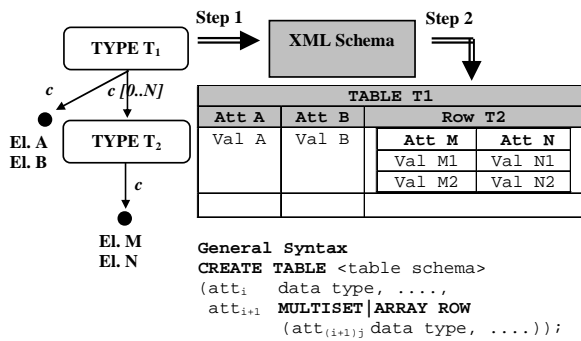


Figure 4. Composition for Multiple Row

Rule 2:

Step 1: For two types namely T_1 and T_2 with elements/attributes (A,B) and (M,N) respectively, if T_1 can be composed by more than one T_2 , implement T_1 as a complex type and T_2 as an inner complex type with maxOccurs constraint = unbound.

Step 2: For two complex types namely T_1 and T_2 with elements/attributes (A,B) and (M,N), if T_1 can be composed by more than one T_2 , implement T_2 as a multiple row attribute of table T_1 . Transformation result is $Table T_1 (A, B, Row T_2 (M, N))$

Example 2:

Type BOOK can be composed by more than one type EDITION (see Fig.5). The composition type will be mapped into a multiple-row attribute in ORDB table.

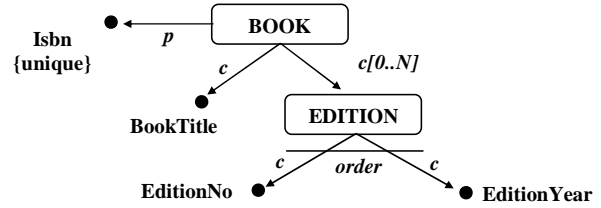


Figure 5. BOOK-EDITION Composition

Transformation into XML Schema:

```

<xsd:complexType name = "BOOK_Type">
  <xsd:element name = "BookTitle" type =
  "xsd:string"/>
  <xsd:element name = "EDITION" minOccurs = "0"
  maxOccurs= "unbounded"/>
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name = "EditionNo" type =
      "xsd:integer" minOccurs = "0"/>
      <!--and other elements -->
    </xsd:sequence>
  </xsd:complexType>
  <xsd:attribute name = "ISBN" type =
  "xsd:string" use = "required"/>
</xsd:complexType>

```

```

<key name="BOOK_Isbn">
  <selector xpath = "BOOK">
  <field xpath="@ISBN"/></key>

```

Transformation into ORDB:

```

CREATE TABLE Book
(Isbn CHARACTER VARYING(5)
CONSTRAINT book_isbn_pk PRIMARY KEY,
BookTitle CHARACTER VARYING(50),
Edition MULTISSET (ROW (EditionNo NUMBER,
EditionYear NUMBER)));

```

Analysis. If we can have more than one “part” component of the same type for one “whole” component, we use multiple row attribute. Our mapping method has also satisfied the two composition constraints.

For the first step we map the “whole” and “part” component as the outer and the inner complex type respectively in the XML schema. To preserve the multiple feature, we use the XML Schema syntax **maxOccurs= “unbounded”**.

```

<xsd:complexType name = "C_Type">...
<xsd:element name = "e_name" minOccurs = "0"
maxOccurs= "unbounded"/>
<xsd:complexType>... </xsd:complexType>
</xsd:complexType>

```

In the second step, we map the outer complex type as the table in ORDB and the inner complex type as the row attribute. To preserve the multiple feature we implement the row as a collection with this syntax **TABLE (...MULTISSET (ROW()))**.

Example 2 shows that *edition* data is firstly mapped into inner complex type and *book* data as outer complex type. On the second transformation, we implement *edition*

type as a collection of row attribute in table BOOK. If we remove a book details, all details of the book edition will also be removed.

3.3. Implementation of Multi-Level Row Composition

Mapping Method. In ORDB, supported by SQL4, we can have a row inside another row attribute. The data structure will be more complex than the flat relation in conventional model, but sometimes it resembles the real world problem.

Rule 3:

Step 1: For three components namely T_1 , T_2 and T_3 , with elements and/or attributes (A,B), (M,N), and (X, Y) respectively, if T_1 is composed by T_2 and T_2 is composed by T_3 , implement the T_2 as inner complex type of T_1 and T_3 as inner complex type of T_2 .

Step 2: For three complex types namely T_1 , T_2 and T_3 , with elements and/or attributes (A,B), (M,N), and (X, Y) respectively, if T_1 is composed by T_2 and T_2 is composed by T_3 , implement the last two types as a multi-level row attribute of T_1 . Transformation result is Table T_1 (A, B, $\begin{matrix} 1/n \\ 0 \end{matrix}$ Row T_2 (M, N, $\begin{matrix} 1/n \\ 0 \end{matrix}$ Row T_3 (X, Y)))

Example3:

Continuing example 2, now type EDITION can be composed by at most one type PRICE (see Fig.7). The composition type will be mapped into a multi-level row attribute in ORDB table.

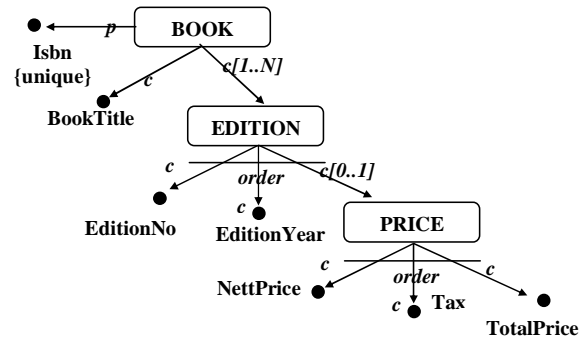


Figure. 7. BOOK-EDITION-PRICE Composition

Transformation into XML Schema:

```

<!--see previous example for key and unique constraint -->
<xsd:complexType name = "BOOK_Type">
  <xsd:element name = "BookTitle" type = "xsd:string"/>
  <xsd:element name = "EDITION" minOccurs = "0" maxOccurs = "unbounded"/>
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name = "EditionNo" type = "xsd:integer" minOccurs = "0"/>
      <!--and other elements -->
      <xsd:element name = "Price" type = "xsd:string" minOccurs = "0"/>
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name = "NettPrice" type = "xsd:float" minOccurs = "0"/>
          <!--and other elements -->
        </xsd:sequence>
      </xsd:complexType>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:attribute name = "Isbn" type = "xsd:string" use = "required"/>
</xsd:complexType>

```

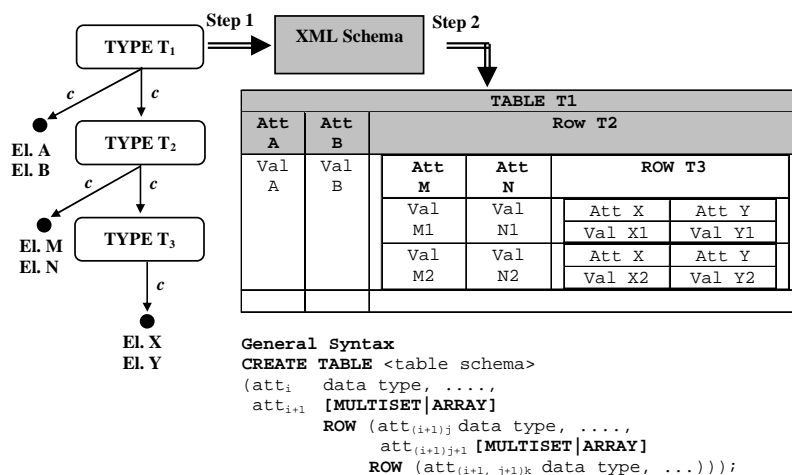


Figure. 6. Composition for Multi-level Row

Transformation into ORDB:

```
CREATE TABLE Book
(Isbn CHARACTER VARYING(5) CONSTRAINT
book_isbn_pk PRIMARY KEY,
BookTitle CHARACTER VARYING(50),
Edition MULTISSET
(ROW (EditionNo NUMBER,
EditionYear NUMBER,
Price ROW (NetPrice DECIMAL(3,2),
Tax DECIMAL(3,2),
TotalPrice DECIMAL(3,2)))));
```

Analysis. The third mapping type is actually the combination of two of the previous mapping methods. In this case, one “part” component can actually be composed by other “part” component. Like in the previous two mapping types, our method has satisfied the composition constraints.

In the first step, we map the “whole” component as the outer complex type and the “part” component as the inner complex type in the XML schema. Inside the inner complex type we can have another inner complex type.

```
<xsd:complexType name = "C_Type">...
<xsd:complexType>...
<xsd:complexType>...
<xsd:complexType>
<xsd:complexType>
</xsd:complexType>
```

In the second step, we map the outer complex type as the table in ORDB and the inner complex type as the row attribute. Most inner complex type will be implemented as row attribute of another row. We will use the syntax **TABLE(...ROW(...ROW(...))**.

In example 3 the *price* data is firstly mapped into inner complex type inside another inner type, *edition*. For the second mapping, the *price* type is implemented as *price* row attribute inside *edition* row attribute.

4. Conclusion

In this paper we have shown how the composition relationship in XML data using semantic network-based conceptual model can be preserved in the implementation using Object Relational Database (ORDB). The composition hierarchy is implemented as Row data type.

In logical level we propose the mapping of Semantic Network Diagram into XML Schema. In implementation level, we propose the usage of single-row, multiple-row,

and multi-level rows for different cardinality and different levels in the composition hierarchy.

Unlike other works in transformation, our proposed implementation preserves the “part” component as the part of the “whole” component. By doing this, the implementation maintains the semantic stated in the conceptual level. In addition, using row type, we have utilized the rich facility in ORDB.

5. References

- [1] R. Bourret, “Mapping DTDs to Databases”, in <http://www.xml.com/pub/a/2001/05/09/dtdtodbs.html>, 2001
- [2] S. Conrad, D. Scheffner, and J. Freytag, “XML Conceptual Modeling using UML”, *ER 2000*, Springer-Verlag, 2000, 558-571
- [3] L. Feng, E. Chang, and T. Dillon, “A Semantic Network-Based Design Methodology for XML Documents”, *ACM TOIS* 20(4), 2002, 390-421
- [4] D. Florescu and D. Kossmann, “Storing and Querying XML Data using an RDMBS”, *IEEE Data Engineering Bulletin* 22(3), 1999, 27-34
- [5] P. Fortier, *SQL3 Implementing the SQL Foundation Standard*, McGraw Hill, 1999
- [6] W-S. Han, K-H. Lee, and B S. Lee, “An XML Storage System for Object-Oriented/Object-Relational DBMSs”, *Journal of Object Technology* 2(1), 2003, 113-126
- [7] M. Klettke and H. Meyer, “XML and Object-Relational Database Systems - Enhancing Structural Mappings Based on Statistics”, *WebDB 2000*, Springer-Verlag, 2000 151-170
- [9] J. Melton, (ed.), *Database Language SQL – Part 2 Foundation*. ISO-ANSI WD 9072-2, International Organization for Standardization, Working Group WG3 (August 2002)
- [10] E. Pardede, J.W. Rahayu and D. Taniar, “New SQL Standard for Object-Relational Database Applications”, *SIIT 2003*, IEEE, 191-203
- [11] J. Rumbaugh, et al, *Object-Oriented Modelling and Design*, Prentice Hall, 1991
- [12] J. Shanmugasundaram, K. Tufte, C. Zhang, G. He, D.J. DeWitt, and J.F. Naughton. “Relational Databases for Querying XML Documents: Limitations and Opportunities”, *VLDB 1999*, Morgan-Kaufman, 1999, 302-314
- [13] N.D. Widjaya, D. Taniar, and J.W. Rahayu, "Aggregation Transformation of XML Schemas to Object-Relational Databases", *IICS 2003*, Springer-Verlag, 2003
- [14] W3C, “The XML data model”, available at <http://www.w3.org/XML/Data-model.html/>, 2000