Updating XML Views

Ling Wang, Elke A. Rundensteiner and Murali Mani

Worcester Polytechnic Institute
Worcester, MA 01609, USA
{lingw|rundenst|mmani}@cs.wpi.edu

Abstract

In this proposal we study the XML view update problem from two aspects. First, we propose a theory and a checking algorithm to decide whether a correct translation exists for a given view update. Then we develop an update translation mechanism to generate the correct translations over the base data storage when it is mappable. Our work can be used by both commercial database or XML data management systems as advanced features in supporting XML view based application.

1 Motivation

With the growing popularity of XML, it has become the primary data model for views. Both XML-relational systems such as [10, 19] and native XML systems such as [16] support creating XML wrapper views and querying against them. However, update operations against such virtual XML views in most cases are not supported yet.

XML view update problem is more complex than that of pure relational view update [2, 12, 14]. Not only do all the problems in the relational context still exist in XML semantics, but we also have to address the new update issues introduced by the XML hierarchical data model and its flexible update language.

Two problems concerning updating XML views need to be tackled. First, *update translatability* concerns whether some updates on the base data storage, which typically may be a relational database or a native XML document, can be made to effect the given update to the view without causing any view-side-effect. Second, we need to devise an appropriate *translation strategy*. That is, assuming the view update is indeed translatable, how to map the updates on the XML view into the equivalent tuple-based SQL updates or XML document updates on the base data.

This dissertation proposes to explore both aspects of the view update problem. First, a light-weight update translatability checking framework is proposed based on a set of well-established theory and practical reasoning algorithms. Our work in this direction can be applied by any existing XML view update system in industry and academia for analyzing the translatability of a given update statement before translation of it is attempted. Thereafter we aim to find a correct update translation over the base data storage when it is mappable. Work in this direction can be used by both commercial database or XML management systems as advanced feature in supporting XML view-based applications.

2 State-of-Art in View Updating

Update Translatability. An abstract formulation of the update translatability problem is given by the *view complementary theory* in [2, 12]. It uses the invariance of the complement of a view, namely *database side-effect free*, to decide the translatability of a given update. However, by requiring the *database side-effect free* property, the complementary theory is too restrictive to be practical. In [14], the authors relax the criteria for a correct translation as only requiring *view side-effect free*. Based on the notion of a *clean source*, it presents an approach in the relational context for determining the existence of update translations by performing a syntax analysis of the view definition.

The nested hierarchical structure of XML views and the flexible update operations on the view place new challenges on the update translatability issue for XML views. Therefore the previous works in the relational context need to be advanced.

Recent works [8, 9, 13] indicate a loose connection between *data provenance* [8, 9] or *lineage* [13] and the view update problem. The distinction between "why provenance" and "where provenance" is used to guide the view update process to find an appropriate update

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the VLDB copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Very Large Data Base Endowment. To copy otherwise, or to republish, requires a fee and/or special permission from the Endowment.

Proceedings of the 31st VLDB Conference, Trondheim, Norway, 2005

publisher

	pubid	pubname
t_1	A01	McGraw-Hill Inc.
t_2	A02	Simon & Schuster Inc.
t_3	B01	Prentice-Hall Inc.

Legend: Primary Key / Non Key

book

	bookid	title	pubid	price	year
t_1	98001	TCP/IP Illustrated	A01	37.00	1997
t_2	98002	Programming in Unix	A02	45.00	1985
t_3	98003	Data on the Web	A01	48.00	2004

review

	bookid	reviewid	comment	reviewer
t_1	98001	001	A good book on network.	William
t_2	98001	002	Useful for advanced user.	John

```
CREATE TABLE publisher(
  pubid VARCHAR2(10),
  pubname VARCHAR2(100) UNIQUE NOT NULL,
  CONSTRAINTS PubPK
       PRIMARYKEY (pubid))

CREATE TABLE book(
  bookid VARCHAR2(20),
  title VARCHAR2(100) NOT NULL,
  pubid VARCHAR2(10),
  price DOUBLE CHECK (price > 0.00),
  year  DATE,
  CONSTRAINTS BookPK
       PRIMARYKEY (bookid),
  FOREIGNKEY (pubid)
       REFERENCES publisher (pubid))

CREATE TABLE review(
  bookid VARCHAR2(20),
  reviewid VARCHAR2(3),
  comment VARCHAR2(100),
  reviewer VARCHAR2(10),
  CONSTRAINTS BookPK
       PRIMARYKEY (bookid,reviewid),
  FOREIGNKEY (bookid)
       REFERENCES book (bookid))
```

Figure 1: Relational Database of Running Example

```
<DB>
  <publisher>
    <row>
      <pubid>A01</pubid>
      <pubname> McGraw-Hill Inc. </pubname>
    </row>  ...
  </publisher>
  <book>
    <row>
      <bookid>98001</bookid>
      <title>TCP/IP Illustrated</title>
      <pubid>A01</pubid>
      <price>37.00</price>
      <year>1997</year>
    </row>  ...
  </book>
  <review>
    <row>
      <bookid>98001</bookid>
      <reviewid>001</reviewid>
      <comment>A good book on network.</comment>
      <reviewer>William</reviewer>
    </row>  ...
  </review>
<DB>
```

Figure 2: Default XML View of Database in Fig. 1

```
<BookView>
FOR $book IN document("default.xml")/book/row,
      $publisher IN document("default.xml")/publisher/row
WHERE ($book/pubid = $publisher/pubid)
AND ($book/price<50.00) AND ($book/year > 1990)
RETURN {
   <book>
      $book/bookid, $book/title, $book/price,
      <publisher>
         $publisher/pubid, $publisher/pubname
      </publisher>,
      FOR $review IN document("default.xml")/review/row
      WHERE ($book/bookid = $review/bookid)
      RETURN{
       <review>
            $review/reviewid, $review/comment
      </review>}
   </book>},
FOR $publisher IN document("default.xml")/publisher/row
RETURN{
      <publisher>
         $publisher/pubid, $publisher/pubname
      </publisher>}
</BookView>
```

(a)

```
<BookView>

   <book>
      <bookid>98001</bookid>
      <title>TCP/IP Illustrated</title>
      <price>37.00</price>
      <publisher>
         <pubid>A01</pubid>
         <pubname> McGraw-Hill Inc. </pubname>
      </publisher>
      <review>
         <reviewid> 001 </reviewid>
         <comment>
            A good book on network.
         </comment>
      </review >
      <review>
         <reviewid> 002 </reviewid>
         <comment>
            Useful for advanced user.
         </comment>
      </review >
   </book>
<book>
   <bookid>98003</bookid>
   <title>Data on the Web</title>
   <price>48.00</price>
   <publisher>
      <pubid>A01</pubid>
      <pubname> McGraw-Hill Inc. </pubname>
   </publisher>
</book>

<publisher>
   <pubid>A01</pubid>
   <pubname> McGraw-Hill Inc. </pubname>
</publisher>
<publisher>
   <pubid>A02</pubid>
   <pubname> Simon & Schuster Inc </pubname>
</publisher>
<publisher>
   <pubid>B01</pubid>
   <pubname> Simon & Schuster Inc </pubname>
</publisher>

<BookView>
```

(b)

Figure 3: XQuery Views over Relational Database in Fig. 1

translation. Their work has several similarities with ours, e.g., try to find the data trace (provenance) at the query syntax level. However, we utilize this data trace or provenance for a different purpose. The question that [8, 9] tries to answer is: given two equivalent queries that are rewritings of each other, when are the provenances guaranteed to be identical? Instead, we use the provenance to find a correct translation, if one exists, for a given update query.

Update Translation Strategy. Works in this direction are different from above. They assume all the view updates considered are translatable and focus on solving the ambiguity issue and finding the "best" translation. The update translation strategy has been studied for the Select-Project-Join views on relational databases [17, 18, 1]. These works have been further extended for object-based views in [4], when the view is anchored in a *pivot relation* and updates are specified only in those *well-nested* relations.

The update translation in XML views has also been explored to some degree in recent works such as [6, 7, 22] and commercial database systems [3, 11, 21]. Under the assumption that the given update is translatable, [6, 7] propose an update translation strategy for converting the XML view update into a relational view update. The main result of [22] is a proposal of an XQuery update grammar. It also studies the execution performance of translated updates.

However, none of these works consider any of the following basic questions: (i) what is the search space for possible optimized translations? (ii) Which is the most suitable one and how to identify it?

3 A Running Example

We now use an XML view defined over a relational database as an example to briefly illustrate that updates through XML views can be problematic.

Fig. 1 shows a relational schema and sample data, which contains a list of publishers as well as their published books and corresponding reviews. Recent XML systems [10, 15] use a basic XML view, called *default XML view*, to define one-to-one relational-to-XML mappings (Fig. 2). On top of this default XML view, a *view query* defines user-specific XML wrapper views (Fig. 3a). Let's first consider several update operations.

Example 1 *To delete only the title of a book is not valid since the title of the book relation is NOT NULL.*

Example 2 *To delete the publisher of the first book from the BookView is not translatable. The reason is there is a foreign key from book relation to the publisher relation in the underlying relational database. When the publisher is deleted, the corresponding book tuple has to be either also deleted, or the pubid of the book is replaced with NULL, depending on the deletion policy defined by the foreign key constraints. However, neither of these two are correct because they both would cause the corresponding book to no longer appear in the view. We thus say that this update is not translatable since it causes a view side effect in the form of an unintended view deletion.*

Example 3 *Inserting a new book into BookView with "bookid=98003" is not translatable since a book with the same bookid already exists in the book relation.*

An update over a given view can be problematic for different reasons, such as violating constraints, causing view side effects or conflicting with the underlying data as shown by the three examples above. An update thus needs to be carefully checked before the translation starts to avoid translation costs, or even a faulty translation.

Example 4 *Deleting the review of a certain book from the BookView is translatable. We can either delete the review tuple or replace the bookid with NULL. Both are correct translations.*

The update translation system needs to (i) explore all the possible translations that satisfy some well-established criteria, and (ii) choose the one that is "as close as possible" to the original database state. That is, we would like to minimize the effect of the view update on the database.

4 Dissertation Objectives

In this proposal, we focus on the translatability and translation strategies of updates through the XML views, which wrap either relational or XML data. Fig. 4 shows the system framework of our proposed XML view updating system. The proposed techniques will be implemented and tested in our XML view management system *Rainbow* [26]. In particular, we focus on the following objectives.

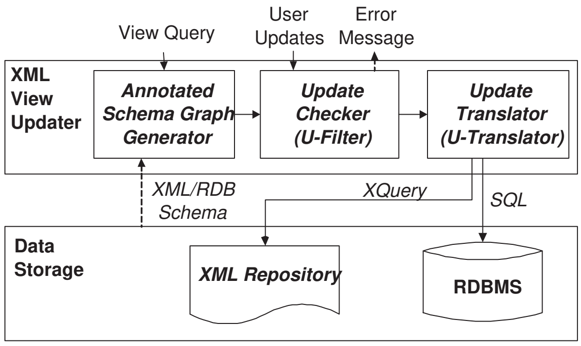

Figure 4: The Framework of XML View Updating System

- *Resolve the mismatch between the XML hierarchical view model and the base data model.* The nested structure imposed by an XML view may be in conflict with the hierarchy explicitly or implicitly defined by the underlying base data model. In other words, if the base is relational, the constraints of the relational schema imply the base hierarchical structure. If the base is an XML document, its schema expresses the hierarchy. In either case, the base hierarchy can possibly conflict with the view hierarchy. This mismatch will affect the translatability of the view updates. In particular, the challenge arises from the fact that the XML view does not determine a unique relational database schema or XML document schema underneath, and so assumptions about the specific nature of the base data storage cannot be built into the view-update algorithm. Example 2 indicates the update translation problem caused by this mismatch.

- *Handle flexible XML view updates.* Compared to the fixed tuple-based update in the relational view update scenario, updates can be specified on any XML view element. New issues thus arise. As an example, XML views can be very complex and potentially contain data duplications, although the relational database can be normalized and the XML document conforms certain schema. The flexible granularity of XML updates thus could touch part of the duplication, while leaving others untouched. Such an update would not be translatable without any side effect.

- *Support efficient order sensitive update translation.* The user can insert a new element into or delete an existing element from certain position of the XML view. This order-sensitive update translation is a problem specific for XML views, because both relational and the object-oriented data model thus far have not covered ordered models. We plan to support order sensitive updates not only for translatability analysis but also for efficient order handling techniques beyond the general order sensitive XQuery processing.

5 Update Translatability Issue

In this direction, we propose a fundamental theory and practical algorithms to determine whether a given update over the XML view is indeed translatable.

5.1 Theoretical Foundation

We first need a theory that characterizes precisely the conditions under which a mapping from XML view updates into updates on the base data storage is correct, be it XML or relational model.

For this purpose, we propose the concept of *source* for a given XML view element, which characterizes the *search space* of all potential correct update mappings. Then based on the idea of the *clean source*, we propose a *clean source theory* as criterion to determine whether a given view update mapping is correct.

This theory is a natural extension of the earlier work from the relational context [14], but now emphasizes the new challenges defined by the XML data model. These challenges are caused by two facts. (i) The potential conflicts between the hierarchies of XML views and the underlying relational or XML database exist. (ii) Duplicates are rather common in XML views which is a major reason of causing update not translatable.

We expect our clean source theory to serve as a solid theoretical foundation for developing practical algorithms towards update translatability checking.

5.2 Light-weight XML View Update Checker

We propose a lightweight view update checking framework called *U-Filter*. To check whether a view update is translatable, U-Filter first performs two steps of schema-level (and thus very inexpensive) checking. Only when necessary, more expensive checking requiring the base data to be accessed is employed.

The first step, called *update validation*, identifies whether the given view update is valid according to the *view schema*. This view schema can either be predefined or be inferred from the view definition query and the base relational schema knowledge. Given that a lot of work has already been done in the literature on schema validation [5, 20], here we focus on two questions closely related with the view update issue: (i) How to extract the view schema from the view query and the relational schema? (ii) Which constraints should the validation procedure consider? The problem in Example 1 is identified by this first step.

In the second step, called *schema-driven translatability reasoning*, any update determined to be valid by Step 1 is further examined. Here the potential view side effects are checked, which can be caused by different reasons, including foreign key constraints conflicting with the view structure or base data duplication in the view. This compile-time check only utilizes the view query and the relational schema. Example 2 is identified here.

Updates identified as translatable by the previous two steps could potentially still conflict with the base data. For instance, in Example 3, u_3 survives the first two steps, but is found not to be translatable. In our third step, the run-time *data-driven translatability checking*, such conflicts will be identified by issuing *probe queries*. This check can only be resolved by examining actual base data. This is typically rather expensive. Hence it is practical to employ this only at last, when the prior check steps have been considered and the update passes these filters.

Moreover, for an order-sensitive XML view specified using a FLWOR expression, an order-sensitive update can delete an existing view element in a certain position or insert a new view element into a certain position of the view. This update can cause new data-related update translatability issues in terms of the update position itself being defined. Special order-specific probe query are utilized to verify the legacy of the deleting or inserting position.

5.3 Research Progress

As preliminary work thus far, we have proposed and proven the general update translatability theory for XML views published over a relational database [24]. An initial study of the update translatability of XML views over the relational database in the “round-trip” case is done in [23]. The round-trip case is characterized by a pair of reversible lossless mappings for (i) loading the XML documents into the relational database, and (ii) extracting an XML view identical to the original XML document back out of it. We prove that any *valid* update operation over such XML views, given a valid pair of round-trip mappings, is always translatable.

We will continue to explore the update translatability checking approaches for both relational databases and XML documents. We aim to develop a complete theory and establish practical algorithms for the update translatability issue.

6 Update Translation Strategies

Now assuming the view update is indeed translatable, the question remains how best to translate the updates on the XML view into the equivalent tuple-based SQL updates or XML document updates on the base data. This requires understanding the ways in which individual view update requests may be satisfied by updates over the underlying data storage. In some cases, there will be precisely one way to perform the database update that results in the desired view update. However, in most other cases, the new view state may correspond to many database states. Consequently, the question of choosing a view update translator arises. Of these database states, we would like to choose one that is “as close as possible” under some measure to the original database state, namely, to minimize the effect of the view update on the database [4, 17].

6.1 Challenging Issues

A set of criteria must first be established to distinguish between a "good" and a "bad" translation. The five validity criteria from relational view updates [17, 18, 1, 4] are syntactically based, namely, they are based on the schema knowledge of the view and the underlying data storage. The *semantic* consideration [1, 18], namely to consider the "real" world meaning of the view, are not addressed by these criteria. In the XML view update context, we need to modify these criteria.

Conceptually an enumeration of all possible valid translations of each view update on the view should be examined. For practical reasons, we do not want to instantiate this enumeration, we merely use it to define the space of alternatives. We identify a search space of correct update translations based on the syntax of a view definition.

When the underlying data storage is a relational database, the mismatch between the two update languages (XQuery FLWU updates on the view versus SQL queries on the base) must be dealt with. Further, new optimization techniques on generating efficient base updates, which consider the performance impacts imposed by for example using some partially materialized index, need to be designed.

Order functions deserve special attention in XML view update study. Special probe queries over the XML base will be employed, which extract information to generate a proper order code or position in the XML document for the newly inserted view element.

6.2 Research Progress

We have accomplished the order sensitive query optimization when an XML view is defined over the relational database [25]. We present a general approach for supporting order-sensitive XQuery-to-SQL translation that works irrespective of the chosen XML-to-relational data mapping and the selected order-encoding method. We will continue to explore the update translation approaches for XML views published over both relational databases and XML documents. We aim to develop a reliable and efficient algorithm for the update translation.

7 Conclusions

In this proposal, we tackle the problem of updating XML views in the context of both XML and relational data model underneath. We propose to develop a full-fledged XML view update system, which first determine whether an update is mappable and then find a correct update translation over the base data storage when it is mappable. We expect this work to be used by both commercial database or XML data management systems as advanced features for view based applications.

References

[1] A. M. Keller. The Role of Semantics in Translating View Updates. *IEEE Transactions on Computers*, 19(1):63–73, 1986.

[2] F. Bancilhon and N. Spyratos. Update Semantics of Relational Views. In *ACM Transactions on Database Systems*, pages 557–575, Dec 1981.

[3] S. Banerjee, V. Krishnamurthy, M. Krishnaprasad, and R. Murthy. Oracle8i - The XML Enabled Data Management System. In *ICDE*, pages 561–568, 2000.

[4] T. Barsalou, N. Siambela, A. M. Keller, and G. Wiederhold. Updating Relational Databases through Object-Based Views. In *SIGMOD*, pages 248–257, 1991.

[5] M. Benedikt, C. Y. Chan, W. Fan, and R. Rastogi. DTD-Directed Publishing with Attribute Translation Grammars. In *VLDB*, pages 838–849, 2002.

[6] V. P. Braganholo, S. B. Davidson, and C. A. Heuser. On the Updatability of XML Views over Relational Databases. In *WEBDB*, pages 31–36, 2003.

[7] V. P. Braganholo, S. B. Davidson, and C. A. Heuser. From XML view updates to relational view updates: old solutions to a new problem. In *VLDB*, pages 276–287, 2004.

[8] P. Buneman, S. Khanna, and W.-C. Tan. Data provenance: Some basic issues. In *Foundations of Software Technology and Theoretical Computer Science*, 2000.

[9] P. Buneman, S. Khanna, and W.-C. Tan. Why and where: A characterization of data provenance. In *ICDT*, 2001.

[10] M. J. Carey, J. Kiernan, J.Shanmugasundaram, E. J. Shekita, and S. N. Subramanian. XPERANTO: Middleware for Publishing Object-Relational Data as XML Documents. In *The VLDB Journal*, pages 646–648, 2000.

[11] J. M. Cheng and J. Xu. XML and DB2. In *ICDE*, pages 569–573, 2000.

[12] S. S. Cosmadakis and C. H. Papadimitriou. Updates of Relational Views. *Journal of the Association for Computing Machinery*, pages 742–760, Oct 1984.

[13] Y. Cui, J. Widom, and J. L. Wienner. Tracing the lineage of view data in a warehousing environment. In *ACM Transactions on Database Systems*, volume 25(2), pages 179–227, June 2000.

[14] U. Dayal and P. A. Bernstein. On the Correct Translation of Update Operations on Relational Views. In *ACM Transactions on Database Systems*, volume 7(3), pages 381–416, Sept 1982.

[15] M. F. Fernandez, A. Morishima, D. Suciu, and W. C. Tan. Publishing Relational Data in XML: the SilkRoute Approach. *IEEE Data Engineering Bulletin*, 24(2):12–19, 2001.

[16] H. Jagadish, S. Al-Khalifa, L. Lakshmanan, A. Nierman, S. Paparizos, J. Patel, D. Srivastava, and Y. Wu. Timber: A native xml database. In *VLDB*, 2002.

[17] A. M. Keller. Algorithms for Translating View Updates to Database Updates for View Involving Selections, Projections and Joins. In *Fourth ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, pages 154–163, 1985.

[18] A. M. Keller. Choosing a View Update Translator by Dialog at View Definition Time. In *VLDB*, pages 467–474, 1986.

[19] M. Fernandez et al. SilkRoute: A Framework for Publishing Relational Data in XML. *ACM Transactions on Database Systems*, 27(4):438–493, 2002.

[20] M. Murata, D. Lee, M. Mani, and K. Kawaguchi. Taxonomy of xml schema languages using formal language theory. In *ACM TOIT*, 2005.

[21] M. Rys. Bringing the Internet to Your Database: Using SQL Server 2000 and XML to Build Loosely-Coupled Systems. In *VLDB*, pages 465–472, 2001.

[22] I. Tatarinov, Z. G. Ives, A. Y. Halevy, and D. S. Weld. Updating XML. In *SIGMOD*, pages 413–424, May 2001.

[23] L. Wang, M. Mulchandani, and E. A. Rundensteiner. Updating XQuery Views Published over Relational Data: A Round-trip Case Study. In *XML Database Symposium*, pages 223–237, 2003.

[24] L. Wang and E. A. Rundensteiner. On the Updatability of XQuery Views Publised over Relational Data. In *ER*, pages 795–809, 2004.

[25] L. Wang, S. Wang, B. Murphy, and E. A. Rundensteiner. Order Sensitive XQuery Processing over Relational Sources: An Algebraic Approach. In *IDEAS*, 2005.

[26] X. Zhang, K. Dimitrova, L. Wang, M. EL-Sayed, B. Murphy, L. Ding, and E. A. Rundensteiner. RainbowII: Multi-XQuery Optimization Using Materialized XML Views. In *Demo Session Proceedings of SIGMOD*, page 671, 2003.