U-Filter: A Lightweight XML View Update Checker

Ling Wang, Elke A. Rundensteiner and Murali Mani
Worcester Polytechnic Institute, Worcester, MA 01609, USA
{lingw|rundenst|mmani}@cs.wpi.edu

1 Introduction

Both XML-relational systems and native XML systems support creating XML wrapper views and querying against them. However, update operations against such virtual XML views in most cases are not supported yet.

Two problems concerning updating XML views need to be tackled. First, *update translatability* concerns whether the given update to the view can be achieved by updates on the base data without any view-side-effect [1, 6, 8]. This base data storage typically may be a relational database or a native XML document. Second, we need to devise an appropriate *translation strategy*, namely, assuming the view update is indeed translatable, how to map the updates on the XML view into the equivalent SQL updates or XML document updates on the base data.

The second issue, the translation strategy, has been studied in recent works [3, 4, 10]. Under the assumption that the given update is translatable, [3, 4] propose an approach to convert the XML into the relational view update problem. [10] studies the performance of executing translated updates. Our work here is *orthogonal* to these works by addressing new challenges related to the decision of translation existence when no particular restrictions have been placed on the defined view for the update translatability study. That is, in general, conflicts in schema and data are possible and a view cannot always be guaranteed to be revert-able [11], nor well-nested [3, 4] — as assumed by these prior works.

This update translatability issue is important in terms of both correctness and performance. Without translatability checking, blindly translating an XML view update into relational updates can be dangerous. Such blind translation may result in *view side effects*. To identify this, the view before and after the update would have to be compared as done in [9]. To adjust for such an error, the view update would have to be rejected and the database would have to be recovered for example by rolling back. This would be rather time consuming, depending on the size of the database. By performing an update translatability analysis, such ill-behaved updates could instead be identified early on and rejected. Thus it would clearly be less costly.

Based on the notion of data provenance (lineage) – the description of the origins of each piece of data in a view, recent works [5, 7] indicate a loose connection between the concept of provenance and the view update problem. However, these works do not answer the questions important to update translatability such as (i) whether the provenance is the correct translation and (ii) if it is not, whether there *exists* at least one (other) correct translation?

In this paper, we propose a general framework called U-Filter to assess the translatability of an update over an *arbitrary* XML view of a relational database,i.e., a view for which various schema level and data level conflicts potentially exist. U-Filter represents a practical approach that could be applied by any existing view update system for analyzing the translatability of a given view update before translation of it is attempted.

2 Examples of Translatability Cases

Fig. 1 shows a running example of a relational schema and sample data of a book database. User-specific XML wrapper views (Fig. 2) can be defined on top of it. Fig. 3 shows several examples of view updates using an XQuery "like" update syntax [10].

Example 1 *In Fig. 3,* u_1 *inserts a new book element into BookView. We notice that the title of the new book is empty and the price is "0.00". However, the underlying*

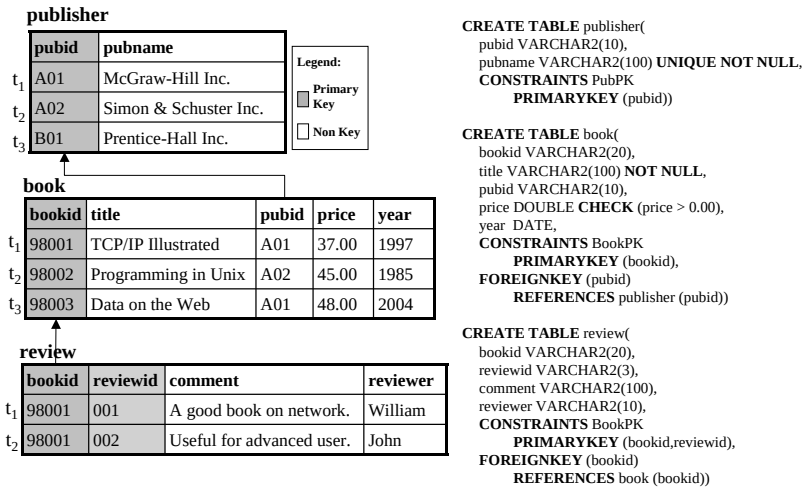

Figure 1: Relational Database of Running Example

```
<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>
```

Figure 2: View Definition over the Relational Database in Fig. 1

u_1

```
FOR $root IN document("BookView.xml")
UPDATE $root {
  INSERT
    <book>
      <bookid>"98004"<bookid>
      <title> </title>
      <price> 0.00 </price>
      <publisher> … </publisher>
    </book> }
```

u_2

```
FOR $root IN document("BookView.xml"),
    $book IN $root/book
WHERE $book/bookid/text() = " 98001"
UPDATE $root {
    DELETE $book/publisher}
```

u_3

```
FOR $book IN
document("BookView.xml")/book
WHERE
$book/title/text() = "DB2 Universal Database"
UPDATE $book {
  INSERT
    <review>
      <reviewid>001</reviewid>
      <comment>
        Easy read and useful.
      </comment>
    </review>}
```

u_4

```
FOR $root IN document("BookView.xml")
UPDATE $root {
  INSERT
    <book>
      <bookid>"98001"<bookid>
      <title>" Operating Systems "</title>
      <price> 20.00 </price>
      <publisher>
        <pubid>A01</pubid>
        <pubname>
          McGraw-Hill Inc.
        </pubname>
      </publisher>
    </book> }
```

Figure 3: Updates over View in Fig. 2

relational schema has the constraints that the title of book tuples is NOT NULL, while the price of the book tuple should be a positive number. Thus, u_1 is not translatable since it directly conflicts with the check constraints from the relational schema.

Example 2 *u_2 in Fig. 3 deletes the publisher of the first book. In the underlying relational database, there is a foreign key from book relation to publisher relation. So, when the publisher is deleted, the corresponding book tuple has to be either also deleted, or the pubid of the book needs to be 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 side-effect of the corresponding book to no longer appear in the view. We thus say that u_2 is not translatable since it causes a view side effect.*

Example 3 *The update u_3 in Fig. 3 inserts a review for the book "DB2 Universal Database", while this book is not in the view. And u_4 inserts a new book which conflicts with an existing book ($book.t_1$), since they both have "bookid=98001". Both u_3 and u_4 are not translatable.*

3 U-Filter: Our Approach for View Update Checking

The above examples illustrate that potential conflicts at both the schema or the data level can affect the translatability of a given view update. To address these factors we propose a lightweight view update checking framework called *U-Filter*. It generates an Annotated Schema Graph (ASG) to model the constraints from both the view query and the relational schema. ASG is then extensively used by 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 *update validation* step identifies whether the given view update is valid according to the *view schema*, which can be pre-defined [2] or be inferred from the view definition query and the base relational schema knowledge. The problem in Example 1 is identified by this step.

In the second step, called *schema-driven translatability reasoning*, any valid update from Step 1 is further examined. Here the potential view side effects are checked, which can be caused by different reasons such as (i) foreign key constraints conflicting with the view structure or (ii) base data duplication in the view. This compile-time check only utilizes the view query and the relational schema. Example 2 is identified to be not translatable here. Our earlier works [11, 12, 14] describe the theoretical foundation and practical algorithms for this step.

Updates that passed the previous two steps could potentially still conflict with the base data (Example 3). In our third step, the run-time *data-driven translatability checking*, such conflicts will be identified. This check can only be resolved by examining actual base data. This is typically rather expensive. Hence it is practical to employ this only after the prior check steps have already been considered and the update has successfully passed these filters.

Fig. 4 shows the overall framework of U-Filter. We present algorithms and optimizations for each step of U-Filter in [13]. It guarantee to filter out all XML updates that cannot be translated. The remaining updates are fed to the update translation engine, which then can generate the corresponding SQL update statements.

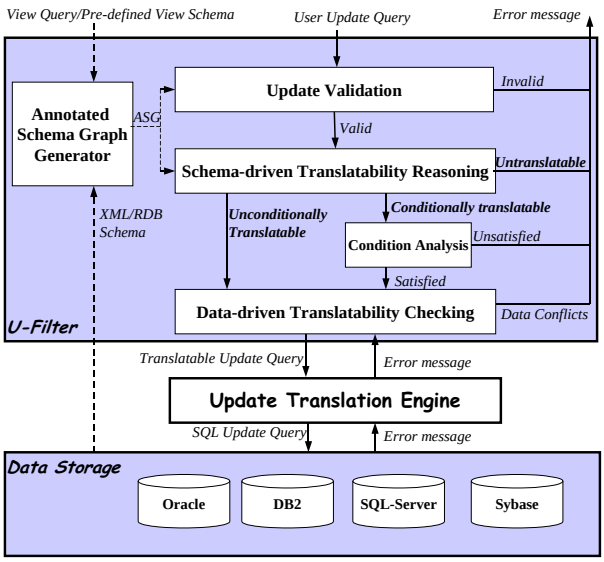

Figure 4: Framework of U-Filter

4 Conclusions

In this paper, we have proposed a lightweight framework, called *U-Filter*, that solves the full spectrum of the XML view update translatability problem. A three-step translatability checking process is used to guarantee that only translatable updates are fed into the actual translation system to obtain the corresponding SQL statements. Our solution is *practical* since it does not require any additional update capability from the relational database. Our solution is *efficient* since we perform schema-level (thus very inexpensive) checks first, while utilizing data-level checking only as the last step.

References

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

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

[3] 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.

[4] V. P. Braganholo, S. B. Davidson, and C. A. Heuser. From XML View Updates to Relational View Updates: Old Solution to a New Problem. In *VLDB*, pages 276–287, 2004.

[5] P. Buneman, S. Khanna, and W.-C. Tan. Why and Where: A Characterization of Data Provenance. In *ICDT*, pages 316–331, 2001.

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

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

[8] 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.

[9] 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.

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

[11] 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.

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

[13] L. Wang, E. A. Rundensteiner, and M. Mani. U-Filter: A Full-fledged XML-to-Relational Update Translatability Checking Framework. Technical Report WPI-CS-TR-05-11, Computer Science Department, WPI, 2005.

[14] L. Wang, E. A. Rundensteiner, and M. Mani. Updating XML Views Published Over Relational Databases: Towards the Existence of a Correct Update Mapping. In *DKE Journal*, in press, 2005.