HUX: A Schema-centric Approach for Updating XML Views

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

Categories and Subject Descriptors
H.2 [Database Management]: Miscellaneous

General Terms
Algorithms

1. MOTIVATION

The problem of updating XML views is more complex than in the relational scenario due to its nested hierarchical structure. While several research projects [1, 4] began to explore this XML view updating problem, they typically provide no guarantee for avoiding view side effects. An update translatability analysis can be employed to reason about the potential view side effects before performing the update. Intuitively, such analysis could examine the actual base data [2, 3]. However, as indicated by [2], this data-centric search for a translation tends to be very expensive, even for relational view updates. Instead, in our HUX project, we have designed a comprehensive solution of exploring schema-knowledge to optimize this analysis.

Fig. 1(a) shows a running example of a relational database for a course registration system. A virtual XML view in Fig. 1(c) is defined by the view query in Fig. 1(b). The following examples illustrate cases of classifying updates as translatable or not translatable. Here we only use a delete primitive with the format \( \text{delete \ nodeID} \), where \text{nodeID} is the abbreviated identifier of the element to be deleted. For example, CI1.PS1 represents the first Professor-Student element of the first ClassInfo element.

Example 1. Update \( u_1 = \left\{ \text{delete CI1.PS1.S2} \right\} \) over the XML view (Fig. 1) deletes the student ‘Mike Fisher’. We can delete Student.S1 to achieve this without causing any view side effect. This can be concluded by looking at the schema of the view. From the view query (Fig. 1(b)), each student can only appear once in the view, namely, in the ClassInfo element that represents its course-professor-student relationship. Deleting any student element in the view can always be translated as deleting the student tuple without causing any side effect. The schema knowledge is sufficient to decide if an update is translatable.

Example 2. Consider the update \( u_2 = \left\{ \text{delete CI1.C1} \right\} \). The appearance of the view element CI1.C1 is determined by two tuples: Professor.t1 and Course.t1. There are three choices for achieving this update: \( T_1 = \left\{ \text{delete Professor.t1} \right\} \), \( T_2 = \left\{ \text{delete Course.t1} \right\} \) and \( T_3 = \left\{ \text{delete Professor.t1, delete Course.t1} \right\} \). All of three translations would cause a view side effect, namely, the whole ClassInfo element would disappear. This conclusion again can be made based on schema knowledge. From the view query, we see that any ClassInfo element must always have a pair of Professor and Course sub-elements. Deleting the course element would break this join condition and thus make the whole ClassInfo element disappear. The schema knowledge is sufficient to classify the update as untranslatable.

Example 3. For update \( u_3 = \left\{ \text{delete CI1} \right\} \), it is easy to see that \( T_1 = \left\{ \text{delete Course.t1} \right\} \) will achieve the update without causing any view side effect for the same reason as Example 1. On the other hand, \( T_2 = \left\{ \text{delete Professor.t1} \right\} \) will cause a side effect since CI2 would disappear. For update \( u_4 = \left\{ \text{delete CI3} \right\} \), we find that \( T_3 = \left\{ \text{delete Course.t1} \right\} \) is a correct translation for the same reason as Example 1. \( T_3 = \left\{ \text{delete Professor.t1} \right\} \) is a correct translation since CI3 is the only class Prof. Tim Merrett teaches. The difference here indicates that the schema knowledge itself is not sufficient for deciding translatability. The translatability depends on the actual base data.

2. HUX: HANDLING UPDATES IN XML

As we can see from the above examples, not only view updates can happen anywhere along the view hierarchy, but also side effects can appear anywhere in the view. The XML view side effect checking is thus more complex than in the relational case. A view update can be classified as translatable or untranslatable using either schema or data knowledge. In this paper, we aim to support updates of XML views by (i) extending the relational view update solution and (ii) utilizing schema knowledge as much as possible. For this, we propose our schema-centric XML view updating system named HUX (Handling Updates in XML). HUX bridges the XML and relational view update problem. One direction for handling updates over XML views may be to “convert” the XML view update problem to the equivalent relational view update problem (if possible). For this purpose, let us follow the approach from the...
FOR $p$ in document(Professor/ROW),
WHERE $p$.pid = $c$.pid
RETURN $<$/Professor/>

FOR $s$ in document(Student/ROW),
WHERE $s$.cid = $c$.cid
RETURN $<$/Student/>

Figure 1: The running example for the course registration system

Figure 2: Schema graph of the XML view

HUX is a schema-centric solution. Fig. 3 shows the HUX framework. The Schema-driven Translatability
Reasoning (STAR) process first filters out all untranslatable updates and classifies some updates as definitely translatable based purely on the schema. For updates that cannot be classified by the STAR process, the Schema-directed Data Checking (SDC) process examines a subspace of the data (guided by the schema knowledge) to definitely decide whether the update is or is not translatable. Untranslatable updates are directly rejected. Updates, that successfully pass the STAR or SDC process, are forwarded to the SQL Update Generator to produce the correct SQL update statements to be executed over the underlying relational database. HUX guarantees that the generated SQL updates are view side-effect-free. By requiring no extra side effect checking or roll back results in a major performance benefit. In our example, during the schema level check, update u1 is classified as translatable. We translate this update by deleting the corresponding tuple in the Student relation. Update u2 will be found to be untranslatable by the schema-level check. Updates u3 and u4 cannot be classified as translatable or untranslatable by the schema-level check. Therefore we proceed to the data-level check, where we find that u3 and u4 are both translatable, and the respective candidate translations are suggested.

3. CONTRIBUTIONS

We propose the first pure data-driven strategy for XML view updating, which guarantees that all updates are fully classified. We propose a schema-driven update translatability reasoning strategy, which uses schema knowledge to efficiently filter out untranslatable and identify translatable updates when possible. We then design an interleaved strategy that combines both schema and data knowledge into one update algorithm to perform a complete classification for XML view updates. We have implemented algorithms and optimization techniques in HUX (Details and proofs in [5]).

4. REFERENCES