

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|rundenst|manijiangm}@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 (*delete nodeID*), where *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 $u_1 = \{\text{delete CI1.PS1.S2}\}$ over the XML view (Fig. 1) deletes the student ‘Mike Fisher’. We can delete *Student.t2* 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 $u_2 = \{\text{delete CI1.C1}\}$. 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: $T_1 = \{\text{delete Professor.t1}\}$, $T_2 = \{\text{delete Course.t1}\}$ and $T_3 = \{\text{delete Professor.t1, delete Course.t1}\}$. 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 $u_3 = \{\text{delete CI1}\}$, it is easy to see that $T_1 = \{\text{delete Course.t1}\}$ will achieve the update without causing any view side effect for the same reason as Example 1. On other hand, $T_2 = \{\text{delete Professor.t1}\}$ will cause a side effect since *CI2* would disappear. For update $u_4 = \{\text{delete CI3}\}$, we find that $T'_1 = \{\text{delete Course.t3}\}$ is a correct translation for the same reason as Example 1. $T'_2 = \{\text{delete Professor.t2}\}$ 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

