Designing Information-Preserving Mapping Schemes for XML

Denilson Barbosa

Juliana Freire **Alberto O. Mendelzon**

VLDB 2005

Motivation

- An XML-to-relational mapping scheme consists of a procedure for shredding XML documents into relational databases, a procedure for publishing the databases back as documents, and constraints the databases must satisfy
- The focus to date has been mostly on the performance of queries (see e.g., (Krishnamurthy et al. [2003]) for a survey) and updates (Tatarinov et al. [2001, 2002])
- We need to understand the properties of a mapping scheme (in any domain) to determine its suitability for a given application
  - Well studied for traditional data models (Hull [1986], Abiteboul and Hull [1988], Miller et al. [1993])
  - We are only starting in the XML context [XSYM'04], (Bohannon et al. [2005])

Information Preservation – Goals

Answering queries:

- Requires reconstructing every fragment of the document: *losslessness* [XSYM'04]
- Previous methods (possibly with simple extensions) suffice

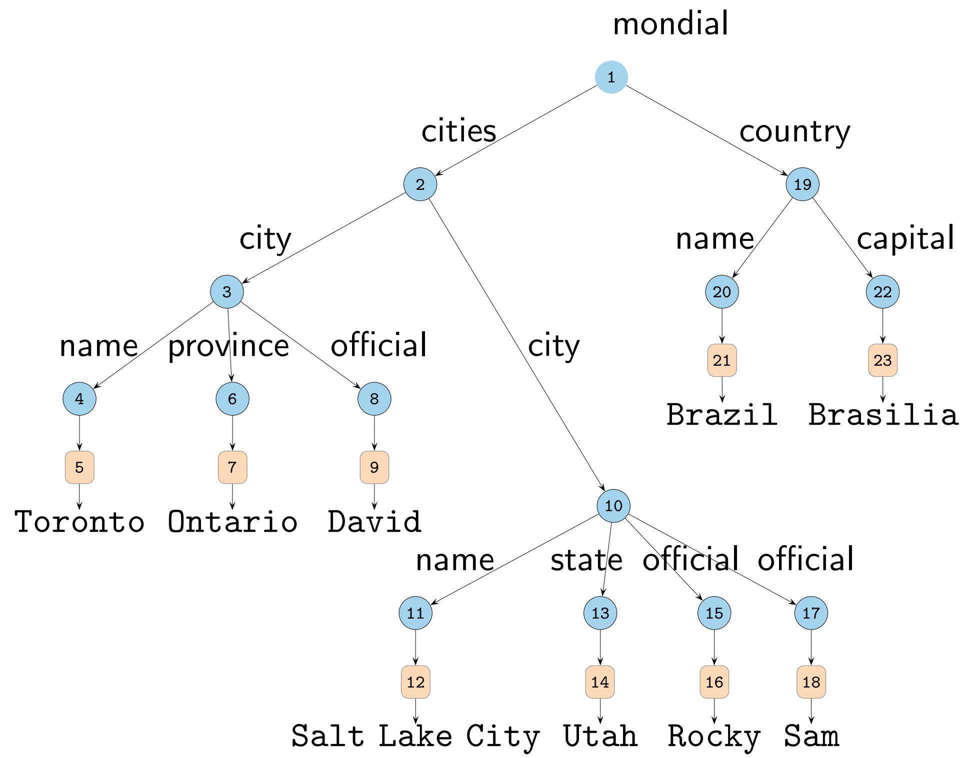
Processing updates, preserving document validity:

- Requires that the resulting database "represents" a valid document and that every valid document can be represented by some database: *validation* [XSYM'04]
- Losslessness alone is not enough
- Problem: checking whether the update is *permissible*

Example

Consider the following DTD and a valid document:

$$\begin{aligned}
&mondial \leftarrow cities, country*\\
&cities \leftarrow city*\\
&city \leftarrow name, (province|state), official+\\
&country \leftarrow name, capital\\
&name \leftarrow \#PCDATA\\
&province \leftarrow \#PCDATA\\
&state \leftarrow \#PCDATA\\
&official \leftarrow \#PCDATA\\
&capital \leftarrow \#PCDATA
\end{aligned}$$



Example – cont'd.

Consider this (lossless) mapping scheme:

$$
\begin{aligned}
&mondial \leftarrow cities, country* \\
&cities \leftarrow city* \\
&city \leftarrow name, (province|state), official+ \\
&country \leftarrow name, capital \\
&name \leftarrow \#PCDATA \\
&province \leftarrow \#PCDATA \\
&state \leftarrow \#PCDATA \\
&official \leftarrow \#PCDATA \\
&capital \leftarrow \#PCDATA
\end{aligned}
$$

```
city (cityId, name, ord, province, state)
official (officialId, cityId, name, ord)
country (countryId, name, capital, ord)
```

```
city (1, 'Toronto', 1, 'Ontario', NULL)
city (4, 'Salt Lake City', 2, NULL, 'Utah')
official (2, 1, 'David', 1)
official (5, 4, 'Rocky', 1)
official (6, 4, 'Sam', 2)
country (7, 'Brazil', 'Brasilia', 1)
```

- **Problems:**

```
UPDATE city SET province='Utah'
WHERE name='Salt Lake City'
```

Legal SQL update

```
update
delete //city[name='Toronto']/official[last()]
```

Cannot be checked statically

Checking for Permissible Updates

Using a mapping scheme that is only lossless:

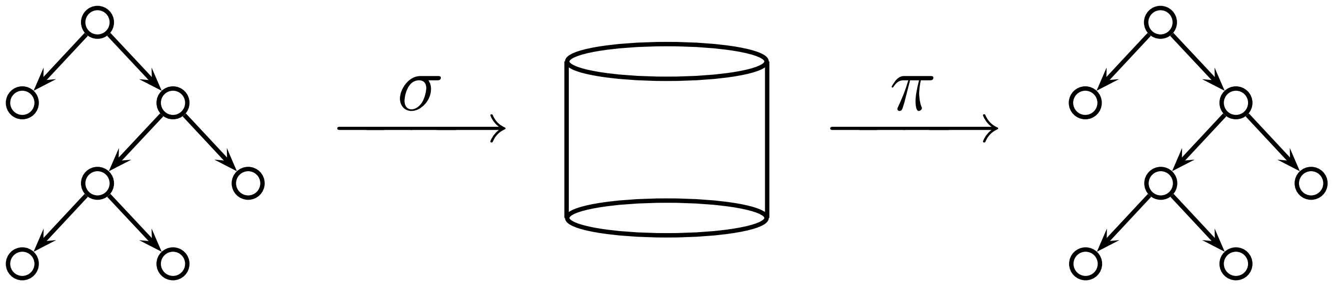
- *Publish* the portions of the database affected by the update, and validate the result
  - Potentially expensive operation; large fragments of the document may have to be reconstructed
- *Build* a (incremental) validator into the DBMS
  - In-DBMS validation is expensive (Nicola and John [2003]) and incremental validation requires maintaining considerable auxiliary information [ICDE'04],(Balmin et al. [2004])
  - Requires a new component whose functionality overlaps with the DBMS constraint checking mechanism

Outline

1. Motivation
2. Information-Preserving Mapping Schemes
   - Losslessness
   - Validation
3. Designing Information-Preserving Mapping Schemes
4. LILO
   - Mapping scheme transformations
5. Conclusion

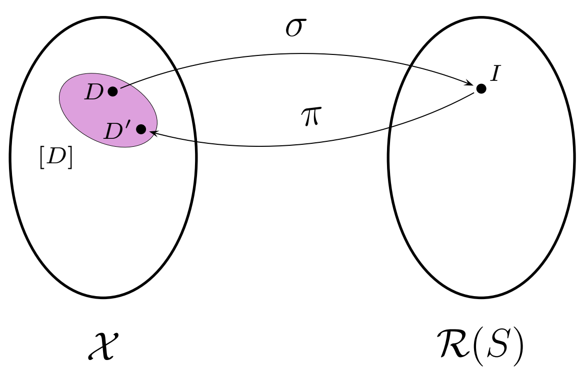
Information-Preserving Mapping Schemes

- A **mapping scheme** is a triple $\mu = (\sigma, \pi, S)$

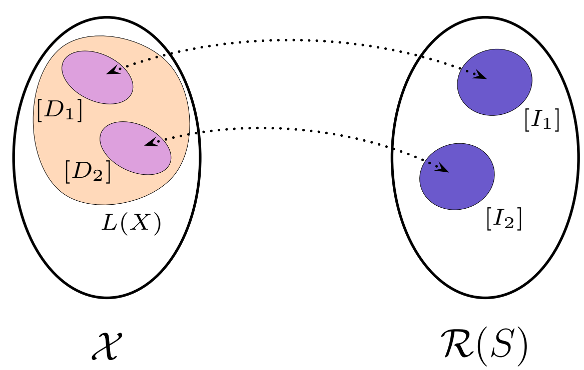


- A **class** of mapping schemes is defined by the languages for writing σ, π, and the constraints in S.
- The $\mathcal{XDS}$ class of mapping schemes [XSYM'04]
  - Mapping language: XQuery augment with mapping expressions
  - Relational constraints: boolean queries in Datalog$^{\neg}$
  - Publishing language: SilkRoute – XQuery over "canonical" XML views of the databases
  - Powerful by design

Information-Preserving Mapping Schemes



lossless mapping scheme



lossless and validating mapping scheme

$\mathcal{X}$: all XML documents

$\mathcal{R}(S)$: all legal instances of S

$L(X)$: all valid documents w.r.t. X

$[\cdot]$: equivalence class

- $\mu = (\sigma, \pi, S)$ is lossless **iff** $\pi(\sigma(\cdot))$ is the identity on equivalence classes of documents
- $\mu = (\sigma, \pi, S)$ is lossless *and* validating **iff** σ and π are bijective and $\sigma = \pi^{-1}$ (up to equivalence)
- $\mu = (\sigma, \pi, S)$ is lossless and validating **iff** $X \equiv S$

Losslessness and validation are undecidable for $\mathcal{XDS}$ mapping schemes [XSYM'04]

Designing Mapping Schemes

$$\begin{tikzcd}
X \arrow[r, shift left, "\sigma"] & S \arrow[l, shift left, "\pi"] \arrow[r, shift left, "\alpha_1"] & S_1 \arrow[l, shift left, "\beta_1"] \arrow[r, shift left, "\alpha_2"] & \cdots \arrow[l, shift left, "\beta_2"] \arrow[r, shift left, "\alpha_k"] & S_k \arrow[l, shift left, "\beta_k"] \\
 & \mu_0 & \mu_1 & \cdots & \mu_k
\end{tikzcd}$$

- Goal: designing a mapping scheme $\mu_k = (\sigma_k, \pi_k, S_k)$ that is both lossless and validating
- Framework for designing lossless and validating mapping schemes in $\mathcal{XDS}$:
  - Start with μ_0 that is *known* to be lossless and validating
  - Apply *equivalence-preserving* transformations between μ_i and μ_{i+1}
  - In the paper: rewriting $\mu = (\sigma, \pi, S)$ in $\mathcal{XDS}$ and α_i, β_i in wrec-ILOG$^\neg$ into $\mu' = (\sigma', \pi', S')$ in $\mathcal{XDS}$

LILO – Initial Mapping Scheme

Initial mapping scheme in LILO: Edge^{++} is both lossless and validating [XSYM'04]

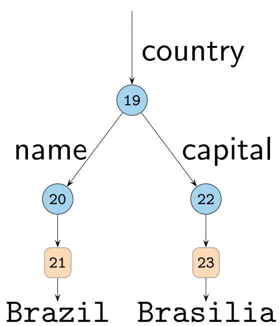
- Relational Schema:
  - `Edge`, `FLC`, `ILS`, `Value`: document structure and content
  - `Type`: element types
  - `Transition`: transition functions of all content models in the DTD
- Constraints:
  - Structural Constraints ensure the database represents a well-formed XML document; e.g., the database encodes a tree, the ordering of siblings is consistent, etc.
  - Validating Constraints ensure that the content of every element is *valid*; i.e., spells a word accepted by an appropriate DFA
- Each validation constraint is implemented by a recursive Datalog$^{\neg}$ program

LILO Transformations – Example

Goal: replace a validating constraint by equivalent constraints that are easier to enforce

Example: enforcing the rule $country \leftarrow name, capital$

- Initial Edge^{++} mapping (S_0):



Edge$_0$

pid	*eid*	*label*
1	19	country
19	20	name
19	22	capital

ILS$_0$

left	*right*
20	22

FLC$_0$

pid	*first*	*last*
19	20	22

Value$_0$

eid	*value*
20	Brazil
22	Brasilia

Type$_0$

eid	*type*
19	t_1

Transition$_0$

type	*from*	*label*	*to*	*acc*
t_1	q_0	name	q_1	no
t_1	q_1	capital	q_2	yes

- Validation constraint: recursive Datalog$^{\neg}$ program

LILO Transformations – Example

Step 1: *inline* the name and capital elements

S_0

Edge$_0$

pid	*eid*	*label*
1	19	country
19	20	name
19	22	capital

FLC$_0$

pid	*first*	*last*
19	20	22

ILS$_0$

left	*right*
20	22

Value$_0$

eid	*value*
20	Brazil
22	Brasilia

S_1

Country$_1$

country	*name*	*capital*
19	20	22

Value$_1$

eid	*value*
20	Brazil
22	Brasilia

Validation constraints:

- *name* and *capital* are unique in **Country$_1$**
- FKs: *name* and *capital* in **Country$_1$** refer to *value* in **Value$_1$**

$$\alpha_1 : \mathcal{R}(S_0) \rightarrow \mathcal{R}(S_1)$$

$\mathsf{Diff}(e) :- \mathsf{Edge}_0(\_, e, \texttt{'country'})$

$\mathsf{Diff}(e) :- \mathsf{Edge}_0(\_, e, \texttt{'capital'})$

$\mathsf{Diff}(e) :- \mathsf{Edge}_0(\_, c, \texttt{'country'}), \mathsf{Edge}_0(c, e, \texttt{'name'})$

$\mathsf{Country}_1(e, n, c) :- \mathsf{Edge}_0(e, n, \texttt{'name'}), \mathsf{Edge}_0(e, c, \texttt{'capital'})$

$\mathsf{Edge}_1(e, c, l) :- \mathsf{Edge}_0(e, c, l), \neg\mathsf{Diff}(e)$

$\mathsf{FLC}_1(p, f, l) :- \mathsf{FLC}_0(p, f, l), \neg\mathsf{Diff}(p)$

$\mathsf{ILS}_1(l, r) :- \mathsf{ILS}_0(l, r), \neg\mathsf{Diff}(l)$

$\mathsf{Value}_1(e, v) :- \mathsf{Value}_0(e, v)$

$$\beta_1 : \mathcal{R}(S_1) \rightarrow \mathcal{R}(S_0)$$

$\mathsf{Edge}_0(e, c, l) :- \mathsf{Edge}_1(e, c, l)$

$\mathsf{Edge}_0(e, c, l) :- \mathsf{Edge}_1(e, \_, \texttt{'country'}), \mathsf{Country}(c, \_, \_),$
$l = \texttt{'country'}$

$\mathsf{Edge}_0(e, c, l) :- \mathsf{Country}_1(e, c, \_), l = \texttt{'name'}$

$\mathsf{Edge}_0(e, c, l) :- \mathsf{Country}_1(e, \_, c), l = \texttt{'capital'}$

$\mathsf{FLC}_0(p, f, l) :- \mathsf{FLC}_1(p, f, l)$

$\mathsf{FLC}_0(p, f, l) :- \mathsf{Country}(p, f, l)$

$\mathsf{ILS}_0(l, r) :- \mathsf{ILS}_1(l, r)$

$\mathsf{ILS}_0(l, r) :- \mathsf{Country}_1(\_, l, r)$

$\mathsf{Value}_0(e, v) :- \mathsf{Value}_1(e, v)$

LILO Transformations – Example

Step 2: *inline* the values of the name and capital elements

S_1

Country$_1$

country	*name*	*capital*
19	20	22

Value$_1$

eid	*value*
20	`Brazil`
22	`Brasilia`

S_2

Country$_2$

country	*name*	*capital*
19	`Brazil`	`Brasilia`

Validation constraints:

- *name* and *capital* are not null in **Country$_2$**

$$\alpha_2 : \mathcal{R}(S_1) \rightarrow \mathcal{R}(S_2)$$

$$\begin{aligned}
\text{Diff}(e) &:- \text{Country}_1(p, e, _), \text{Value}_1(e, _) \\
\text{Diff}(e) &:- \text{Country}_1(p, _, e), \text{Value}_1(e, _) \\
\text{Edge}_2(e, c, l) &:- \text{Edge}_1(e, c, l) \\
\text{FLC}_2(p, f, l) &:- \text{FLC}_1(p, f, l) \\
\text{ILS}_2(l, r) &:- \text{ILS}_1(l, r) \\
\text{Country}_2(e, n, c) &:- \text{Country}_1(e, v_1, v_2), \\
&\quad\ \text{Value}_1(v_1, n), \text{Value}_1(v_2, c) \\
\text{Value}_2(e, v) &:- \text{Value}_1(e, v), \neg\text{Diff}(e)
\end{aligned}$$

$$\beta_2 : \mathcal{R}(S_2) \rightarrow \mathcal{R}(S_1)$$

$$\begin{aligned}
\text{Edge}_1(e, c, l) &:- \text{Edge}_2(e, c, l) \\
\text{FLC}_1(p, f, l) &:- \text{FLC}_2(p, f, l) \\
\text{ILS}_1(l, r) &:- \text{ILS}_2(l, r) \\
\text{PName}(*, e, n) &:- \text{Country}_2(e, n, _) \\
\text{PCapital}(*, e, c) &:- \text{Country}_2(e, _, c) \\
\text{Country}_1(e, n, c) &:- \text{PName}(n, e, _), \text{PCapital}(c, e, _) \\
\text{Value}_1(e, v) &:- \text{Value}_2(e, v) \\
\text{Value}_1(e, v) &:- \text{PName}(e, v, _) \\
\text{Value}_1(e, v) &:- \text{PCapital}(e, _, v)
\end{aligned}$$

LILO Transformations

Each transformation *changes* the way documents are stored, *simplifying* the validation constraints

- *Inlining* element ids or element values
  - Ex.: $country \leftarrow name, capital$ becomes $country \leftarrow capital$
- *Nesting* the contents of elements within their parents
  - Ex.: $mondial \leftarrow cities, country*$ and $cities \leftarrow city*$ become $mondial \leftarrow city*, country*$ and we skip (resp. reinsert) the $cities$ element in σ (resp. π)
- *Outlining*: split the contents of some elements into several relations
  - Ex.: $mondial \leftarrow city*, country*$ becomes $mondial^1 \leftarrow city*$ and $mondial^2 \leftarrow country*$

- Applicable to the vast majority (over 88%) of the XML schemas used in practice

Conclusion

- Vast literature on XML to relational mappings, but the focus to date has been on efficiency, not information preservation
  - Initial work in the XML setting [XSYM'04], (Bohannon et al. [2005])
- Framework for designing lossless and validating mapping schemes in $\mathcal{XDS}$
  - Mechanical, powerful, extensible
  - Results in efficient relational configurations
  - Guarantees both losslessness and validation, by design
  - Exploits the existing RDBMS constraint checking infrastructure

Conclusion

- Previous methods (with straightforward extensions) can guarantee losslessness (but not validation)
  - Numbering schemes capturing both element identity and ordering fully preserve the structure of the documents (Bohannon et al. [2002], Deutsch et al. [1999], Florescu and Kossmann [1999], Shanmugasundaram et al. [1999])
  - Some of LILO's transformations can be viewed as extending those in previous methods with validation constraints
- Schema-aware methods have been shown to provide better query and update performance. Similar effect on LILO compared to Edge^{++} (on XMark):
  - LILO is up two times (83% on average) faster for insertions and 45% faster (36% on average) for deletions when compare to Edge^{++}
- Cost based approaches (Bohannon et al. [2002], Zheng et al. [2003]) rely on *hypothetical* workload execution costs which might be inaccurate [SIGMOD'05]

Conclusion

- Several techniques have been proposed for translating other XML Schema constraints into relational ones in mapping schemes
  - Keys (Davidson et al. [2003]), foreign-keys (Chen et al. [2003]), cardinality constraints (Bohannon et al. [2002], Lee and Chu [2000]), ID/IDREF attributes [ICDE'04], and type specialization [XSYM'04]
- However, to the best of our knowledge, no work has addressed the problem of mapping the *element validity* constraint
- Future work includes defining more transformations; compiling the mapping scheme transformations; thorough experimental study; combining LILO with cost-based methods

Thank you.