MXML Storage and the Problem of Manipulation of Context

Nikolaos Fousteris¹, Manolis Gergatsoulis¹, Yannis Stavrakas²

¹ Department of Archives and Library Science, Ionian University
Ioannou Theotoki 72,49100 Corfu, Greece.
{nfouster,manolis}@ionio.gr

² Institute for the Management of Information Systems (IMIS),
R. C. Athena,
G. Mpakou 17, 11524, Athens, Greece.
yannis@inmis.gr
Introduction & Motivation

- The problem of storing and querying XML data using relational databases has been considered a lot.

- Multidimensional XML is an extension of XML and it is used for representing data that assume different facets, having different values or structure, under different contexts.

- We expand the problem of storing and querying XML to multidimensional XML data.
Outline

- XML Storage
- Multidimensional XML (MXML)
  - Fundamental concepts
  - MXML example and graphical representation
- MXML Storage
  - Two storing approaches are presented
- Context Representation
- Multidimensional XPath (MXPath)
- Context Comparison
- Summary & Future work
XML Storage (1/2)

- Includes techniques to store XML data in Relational Databases

- XML applications (internet applications) are able to exploit the advantages of the RDBMS technology

- Operations over XML data, are transformed to operations over the Relational Schema
XML Storage (2/2)

- **Methodology**
  - A Relational Schema is chosen for storing XML data
  - XML queries are produced by applications
  - XML queries are translated to SQL queries
  - SQL queries are executed
  - Results are translated back to XML and returned to the application

- **Techniques**
  - Schema Based
  - Schema Oblivious
Multidimensional XML (MXML)
Fundamental Concepts (1/3)

- MXML is an extension of XML

- In MXML data assume different facets, having different value or structure, under different contexts according to a number of dimensions which may be applied to elements and attributes
**Dimension**: is a **variable**. Assigning different values for each dimension it is possible to construct different environments for MXML data.

**World**: represents an **environment** under which data obtain a meaning and is determined by assigning to every dimension a single value.

**Context Specifier**: specifies a **set of worlds** (context) under which a facet of an MXML element or attribute, is the holding facet of this element or attribute.
Multidimensional elements/attributes are elements/attributes that have different facets under different contexts.

Each multidimensional element/attribute contains one or more facets, called Context element/attributes.
MXML Graphical Representation

Symbol Table
- root node
- multidimensional element node
- multidimensional attribute node
- context element node
- context attribute node
- value node
- element/attribute/value edge
- element/attribute context edge

Diagram showing the graphical representation of MXML nodes and attributes.
Explicit Context: Is the true context only within the boundaries of a single multidimensional element/attribute.

Inherited Context: Is the context, which is inherited from a ancestor node to a descendant node in the MXML graph.

Inherited Context Coverage: It constraints the inherited context of a node, so as to contain only the worlds under which the node has access to some value node.
MXML Storage (1/2)

- MXML storage includes techniques that store MXML data in Relational Databases.

- Applications using MXML storage are able to exploit the advantages of the RDBMS technology.

- MXML additional features (context, different types of MXML nodes/edges etc.) should be considered.
MXML Storage (2/2)

- **Naive approach**
  Uses a single table (Node Table), to store all information contained in a MXML document. Each row of the table represents a MXML node of the MXML graph.

- **Type Approach**
  MXML nodes are divided into groups according to their type. Each group is stored in a separate table named after the type of the nodes.
Naive Approach

Node Table:
Stores each MXML node in a row.

<table>
<thead>
<tr>
<th>node_id</th>
<th>parent_id</th>
<th>ordinal</th>
<th>tag</th>
<th>value</th>
<th>type</th>
<th>explicit_context</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>book</td>
<td>-</td>
<td>CE</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>isbn</td>
<td>-</td>
<td>MA</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>isbn</td>
<td>-</td>
<td>CA</td>
<td>[ed=en]</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>1</td>
<td>-</td>
<td>0-13-110362-8</td>
<td>VN</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>2</td>
<td>isbn</td>
<td>-</td>
<td>CA</td>
<td>[ed=gr]</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>1</td>
<td>-</td>
<td>0-13-110370-9</td>
<td>VN</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>2</td>
<td>title</td>
<td>-</td>
<td>ME</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>1</td>
<td>title</td>
<td>-</td>
<td>CE</td>
<td>[ ]</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>1</td>
<td>-</td>
<td>The C progr. lang.</td>
<td>VN</td>
<td>-</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
<td>....</td>
<td>....</td>
<td>....</td>
<td>....</td>
<td>....</td>
</tr>
<tr>
<td>43</td>
<td>42</td>
<td>1</td>
<td>picture</td>
<td>-</td>
<td>CE</td>
<td>[c_type=stud]</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
<td>....</td>
<td>....</td>
<td>....</td>
<td>....</td>
<td>....</td>
</tr>
</tbody>
</table>
Type Approach

Type Tables:

Store each MXML node in a row of a specific table according to node’s type.
Comparison

- **Naive approach**
  - is straightforward
  - appear many NULL values
  - queries involve a large number of self-joins of the Node Table

- **Type Approach**
  - avoids NULL values
  - reduces the size of the tables involved in joins (performance)
Question

How can we represent in a Relational Database the set of worlds which are contained in a context specifier, for each MXML node?
**Possible Worlds Table:**
Assigns a unique ID to each possible combination of dimension values (world).

**Explicit Context Table:**
Represents the explicit context (set of worlds) for a MXML node.

**Inherited Context Coverage Table:**
Assigns an inherited context coverage (set of worlds) to a MXML node.
Naive Representation of Context

Possible Worlds Table

<table>
<thead>
<tr>
<th>world_id</th>
<th>edition</th>
<th>customer_type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>gr</td>
<td>stud</td>
</tr>
<tr>
<td>2</td>
<td>gr</td>
<td>lib</td>
</tr>
<tr>
<td>3</td>
<td>en</td>
<td>stud</td>
</tr>
<tr>
<td>4</td>
<td>en</td>
<td>lib</td>
</tr>
</tbody>
</table>

Explicit Context Table

<table>
<thead>
<tr>
<th>node_id</th>
<th>world_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
</tr>
</tbody>
</table>

Inherited Coverage Table

<table>
<thead>
<tr>
<th>node_id</th>
<th>world_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
</tr>
</tbody>
</table>
Problems of Naive Representation of Context

- It is needed one row for each possible world in the Possible Words Table
- More than one entries in the Explicit Context Table or the Inherited Coverage Table are required to represent the context of one MXML node
- SQL queries derived from MXML queries contain joins with the Possible Words Table
Ordered-Based Representation of Context

**Basic idea**: Total ordering of worlds based on:

- Total ordering of dimensions
- Total ordering of dimension values

For $k$ dimensions with each dimension $i$ having $z_i$ possible values, we may have $n = z_1 * z_2 * \ldots * z_k$ possible ordered worlds.

Each world is assigned a *unique integer value* between 1 and $n$ ($w_1$ to $w_n$).
Context Representation (3/7)

Ordered-Based Representation of Context

- Dimensions Ordering
- Dimension Values Ordering
- Possible Worlds Ordering
World Vector:
- A binary number representing a context specifier. The position of every bit corresponds to the position of a world in the total ordering of all possible worlds.

- Each bit of the world vector has two possible values: 1 if the corresponding world exists in context specifier or 0 if it does not.

<table>
<thead>
<tr>
<th>binary digit for W1</th>
<th>......</th>
<th>binary digit for Wn</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 or 0: world exists or not</td>
<td>......</td>
<td>n=possible worlds number</td>
</tr>
</tbody>
</table>

possible worlds ordering
Ordered-Based
Representation of Context

**Context Representation (5/7)**

![Diagram of ordered-based context representation]

Finding position “i” of a world (belonging to a context specifier) in the world vector

Ex: node 27 \(^{\text{ed=en}}\) => world_vector = 0011, positions 3 (w3) and 4 (w4)

ordered worlds:

\[
\begin{align*}
\text{w1} &= (\text{gr,stud}) \\
\text{w2} &= (\text{gr,lib}) \\
\text{w3} &= (\text{en,stud}) \\
\text{w4} &= (\text{en,lib})
\end{align*}
\]
Finding worlds (belonging to a context specifier) from the position of the “1” bit values in a world vector.

Ex: world_vector of node 27 = 0011

(en,stud) (en,lib) = “ed=en”
Context Representation (7/7)

Explicit Context Table:
Assigns an explicit context (expressed in binary format according to world vector representation) to a MXML node.

Inherited Context Coverage Table:
Assigns an inherited context coverage (expressed in binary format according to world vector representation) to a MXML node.
Multidimensional XPath (MXPath) (1/2)

MXPath:

- An extension of XPath able to easily express context-aware queries on MXML data.
- Both explicit context (ec) and inherited context coverage (icc) are used to navigate over multidimensional elements and attributes.
- Conditions on the explicit context at any point of the path are allowed.
- Both multidimensional and context nodes can be returned.
**Multidimensional XPath (MXPath) (2/2)**

**MXPath example:**

\[ \text{icc() } \geq \text{ “-”} \],/child::book

/child::cover[ec() \geq \text{ “ed=gr”}]/child->picture

Query in English:

*Find the (multidimensional) sub-element picture of element cover of the greek edition of the book.*

cover[ec() \geq \text{ “ed=gr”}] is an *explicit context qualifier*. The function \( \text{ec()} \) returns the explicit context of a node. The above qualifier says that the ec of the node cover must be superset of the context described by the context specifier [ed=gr].
Context Comparison (1/2)

MXPath query example:
[icc() >= “-”]/child::book
/child::cover[ec() >= “ed=gr”]/child->picture

Basic idea

Using expression [ec() >= “ed=gr”], we need to compare the context specifier “ed=gr” with the context specifiers, which are stored in the Relational Database in order to evaluate MXML query.

How can we do this using the Ordered-Based representation?
Context Comparison (2/2)

Let $Q_1$ (stored), $Q_2$ (query) context specifiers and $G(Q_1), G(Q_2)$ the binary world vectors of $Q_1, Q_2$

Comparing $Q_1$ with $Q_2$:

- $Q_1 = Q_2 \iff G(Q_1) = G(Q_2)$ equivalently
- $Q_1 = Q_2 \iff (G(Q_1) \text{ XOR } G(Q_2)) = 0$
- $Q_1 \neq Q_2 \iff \neg (G(Q_1) = G(Q_2))$
- $Q_1 \geq Q_2 \iff (G(Q_1) \text{ AND } G(Q_2)) = G(Q_2)$
- $Q_1 > Q_2 \iff ((G(Q_1) \text{ AND } G(Q_2)) = G(Q_2)) \text{ AND } (G(Q_1) \neq G(Q_2))$
- $Q_1 \leq Q_2$ if $Q_2 \geq Q_1$ and $Q_1 < Q_2$ if $Q_2 > Q_1$

Note: These rules help on transforming MXML queries to SQL queries
Summary

- MXML data representation
- Storing MXML in Relational DB
  (2 relational schemas were presented)
- MXML querying using MXPath & Query transformation including context representation

Future work

- Algorithm construction and evaluation for query transformation
- Use of alternative indexing techniques for improving relational schema and query performance
References


Thank you..