MXML Storage and the Problem of Manipulation of Context

Nikolaos Fousteris [1] , Manolis Gergatsoulis [1] , Yannis Stavrakas [2]

[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

MXML – Fundamental Concepts (2/3)

- ***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

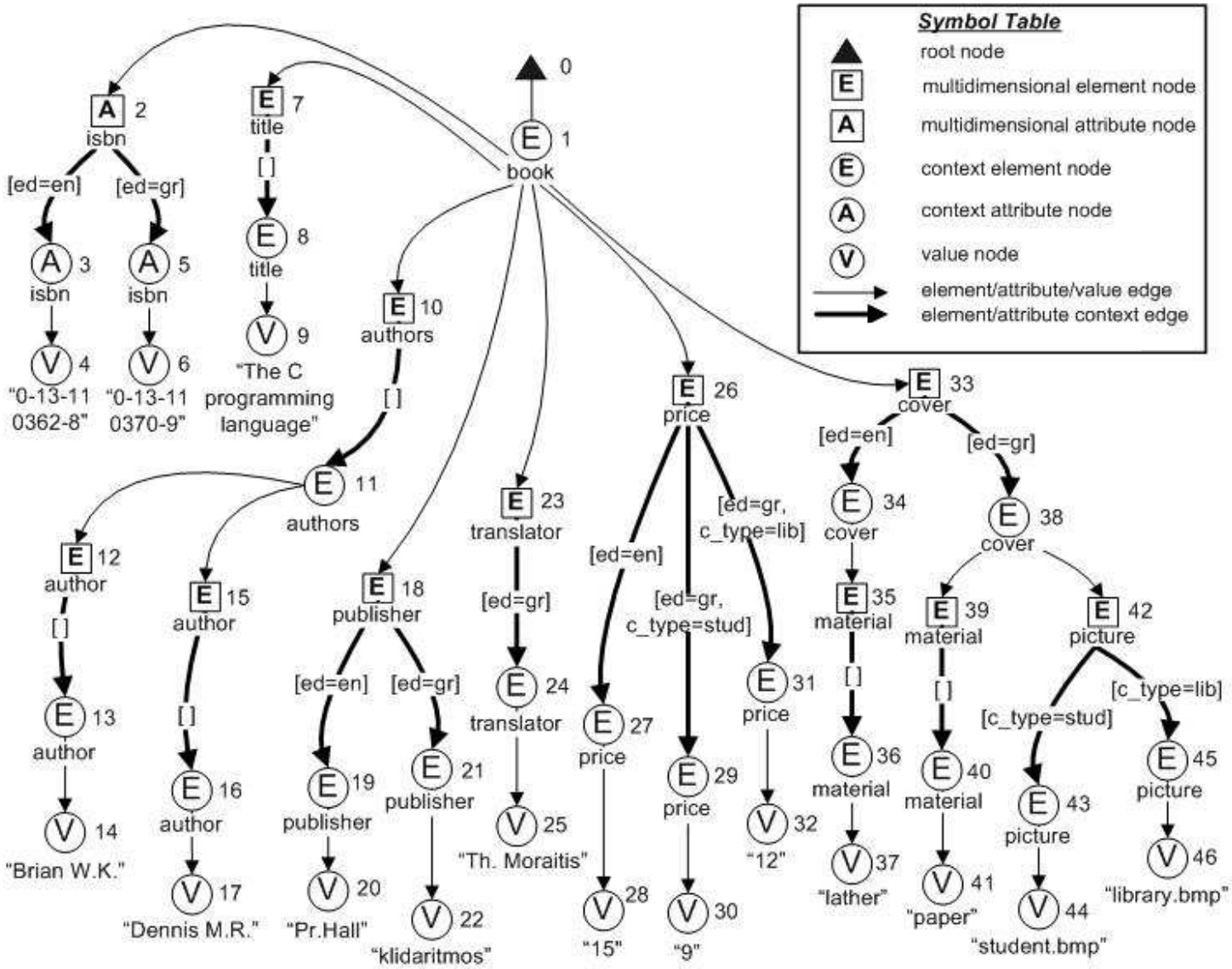
MXML – Example

```
<book isbn=[edition=english]"0-13-110362-8"[/]
                [edition=greek]"0-13-110370-9"[/]>
  <title>The C programming language</title>
  <authors>
      <author>Brian W. Kernighan</author>
      <author>Dennis M. Ritchie</author>
  </authors>
  <@publisher>
  [edition = english] <publisher>Prentice Hall</publisher>[/]
  [edition = greek] <publisher>Klidarithmos</publisher>[/]
  </@publisher>
  <@translator>
  [edition = greek] <translator>Thomas Moraitis</translator>[/]
  </@translator>
  <@price>
     .......
```

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



MXML – Fundamental Concepts (3/3)

- ***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)

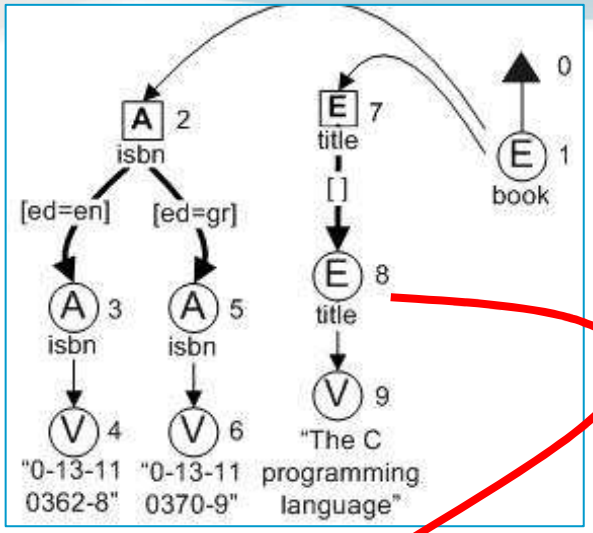
- **Naive approach**

  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



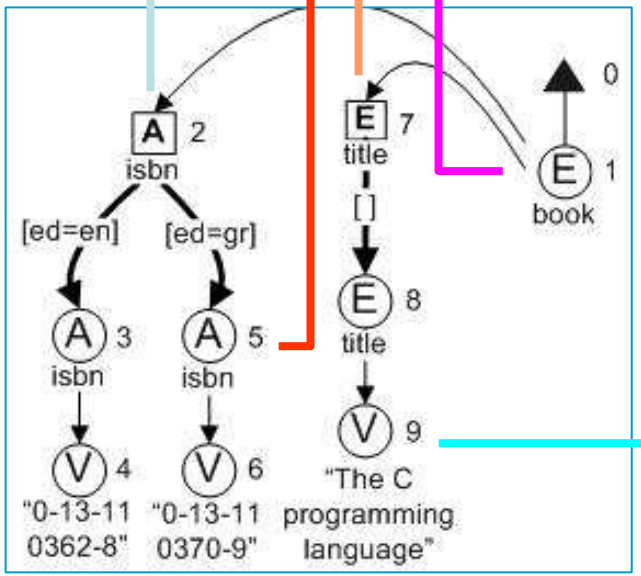
Node Table:

Stores each MXML node in a row.

Node Table

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

Type Approach



ME Table

node_id	parent_id	ordinal	tag
7	1	2	title
10	1	3	authors
....	....	....	....

CE Table

node_id	parent_id	ordinal	tag	explicit_ context
1	0	1	book	-
8	7	1	title	[ ]
....	....	....	....	....
19	18	1	publisher	[ed=en]
21	18	2	publisher	[ed=gr]
....	....	....	....	....

MA Table

node_id	parent_id	ordinal	tag
2	1	1	isbn

CA Table

node_d	parent_id	ordinal	tag	explicit_context
3	2	1	isbn	[ed=en]
5	2	2	isbn	[ed=gr]

Value Table

node_id	parent_id	value
4	3	0-13-110362-8
6	5	0-13-110362-9
9	8	The C programming language
....	....	....

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)

Context Representation (1/6)

Question

How can we represent in a Relational Database the set of worlds which are contained in a context specifier, for each MXML node?

Context Representation (1/6)

Naive Representation of Context

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.

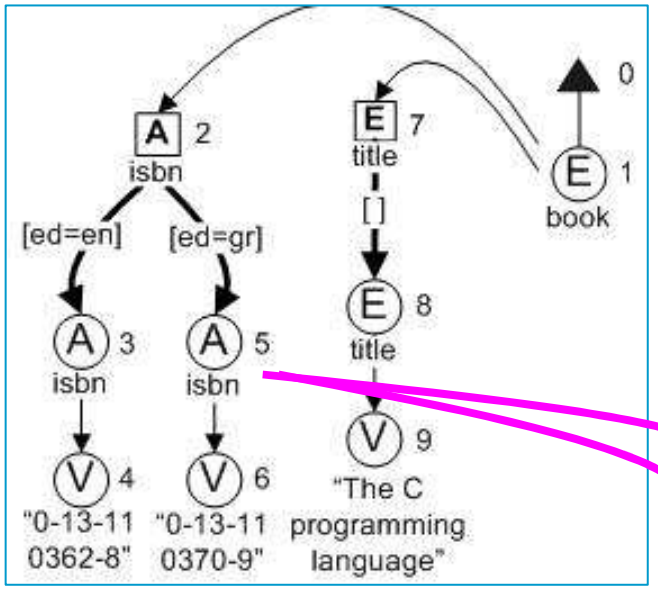
Possible Worlds Table		
world_id	edition	customer_type
1	gr	stud
2	gr	lib
3	en	stud
4	en	lib

Explicit Context Table	
node_id	world_id
1	1
1	2
1	3
1	4
....	....
5	1
5	2
6	1
6	2
6	3
6	4
....	....

Inherited Coverage Table	
node_id	world_id
1	1
1	2
1	3
1	4
....	....
5	1
5	2
6	1
6	2
....	....

Context Representation (2/6)

Naive Representation of Context



Possible Worlds Table		
world_id	edition	customer_type
1	gr	stud
2	gr	lib
3	en	stud
4	en	lib

Explicit Context Table	
node_id	world_id
1	1
1	2
1	3
1	4
....	....
5	1
5	2
6	1
6	2
6	3
6	4
....	....

Inherited Coverage Table	
node_id	world_id
1	1
1	2
1	3
1	4
....	....
5	1
5	2
6	1
6	2
....	....

Context Representation (1/7)

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

Context Representation (2/7)

Ordered-Based Representation of Context

Basic idea: Total ordering of worlds based on:

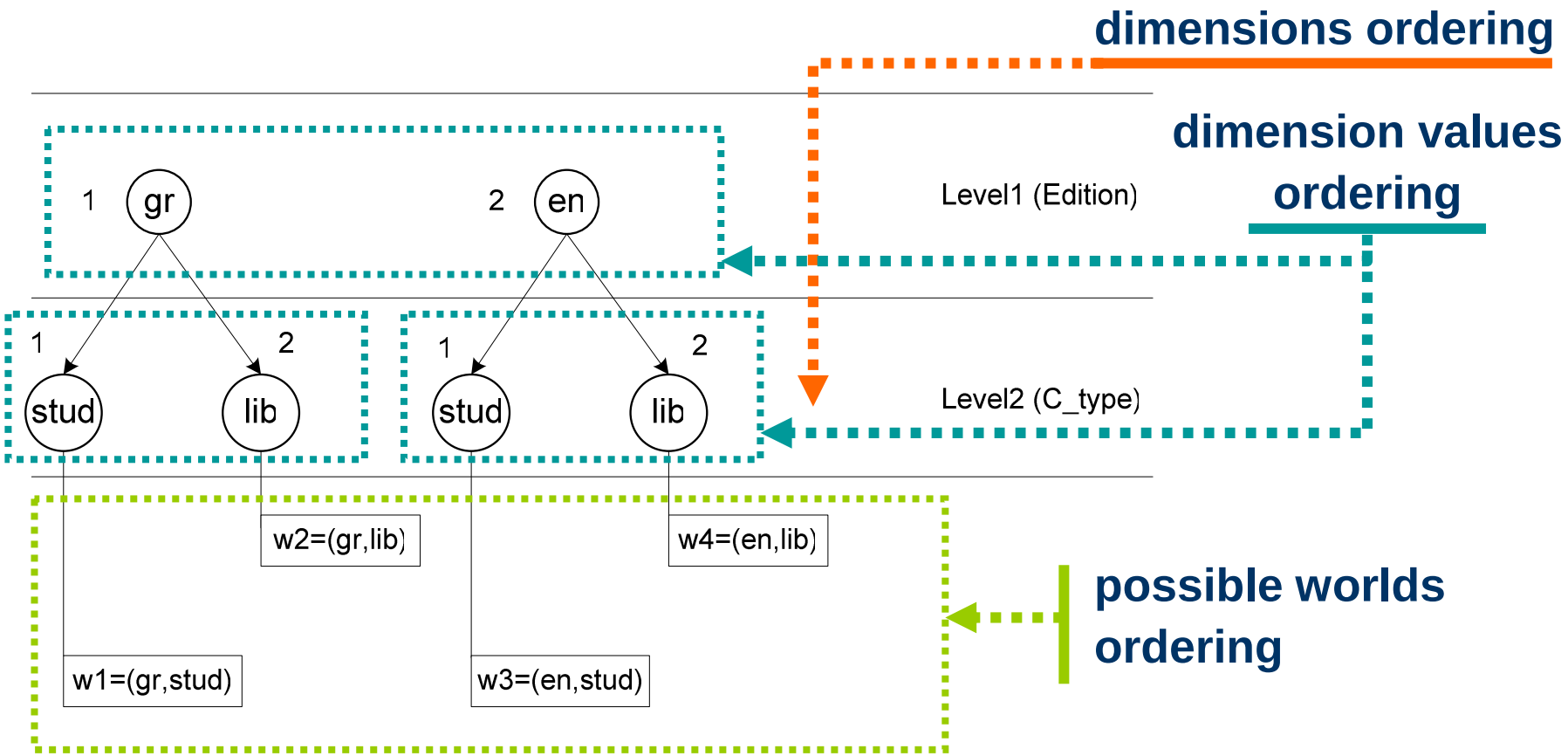
- Total ordering of dimensions
- Total ordering of dimension values

For k dimensions with each dimension i having zi possible values, we may have n=z1*z2*….*zk possible ordered worlds.

Each world is assigned a unique integer value between 1 and n (w1 to wn).

Context Representation (3/7)

Ordered-Based Representation of Context



Context Representation (4/7)

Ordered-Based Representation of Context

World Vector:

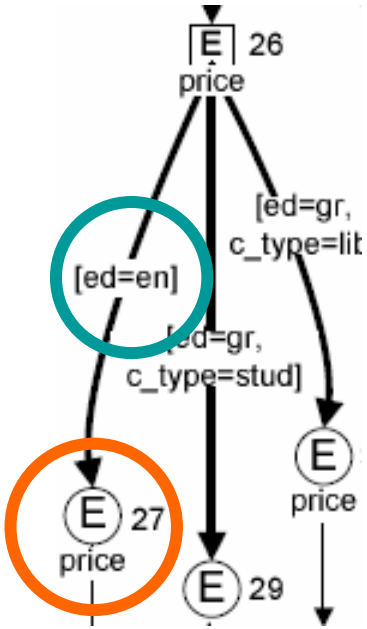
- A binary number representing a context specifier.

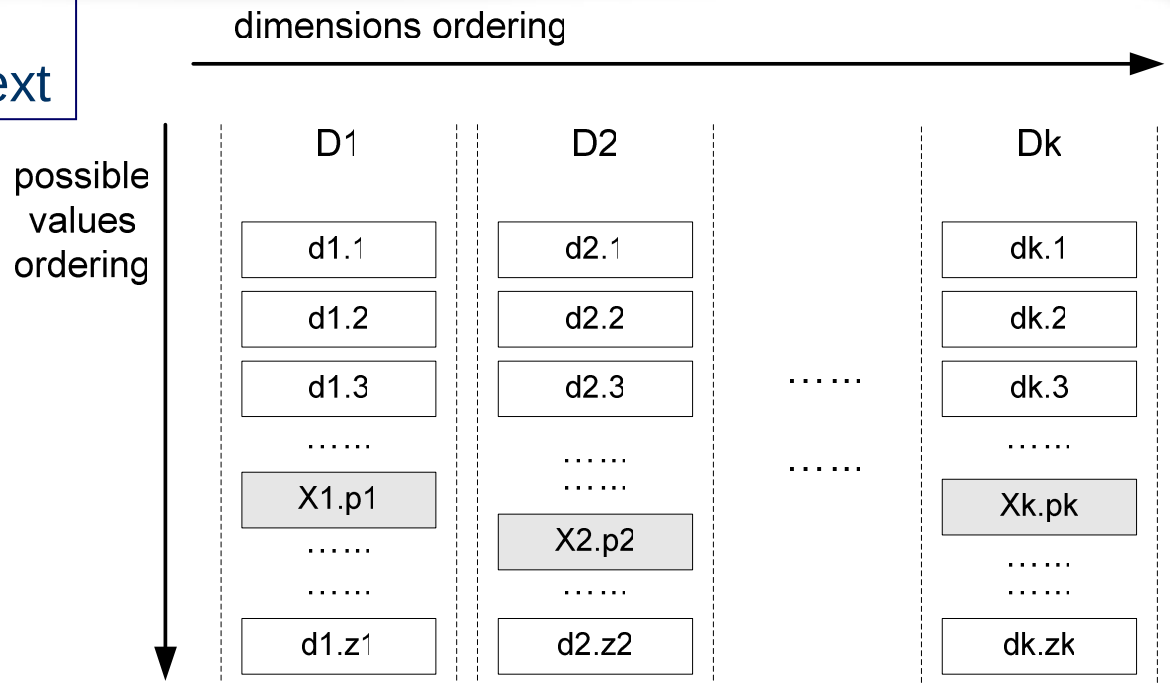
The position of every bit corresponds to the position of a world in the total ordering of all possible worlds.

- 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)

Context Representation (5/7)

Ordered-Based
Representation of Context





world wi = (D1=X1.p1 , D2=X2.p2 ,…., Dk=Xk.pk)

$i = pk + (p(k-1) -1)*zk + (p(k-2) -1)*zk*z(k-1) +\ldots.+ (p1 -1)*zk*z(k-1)*\ldots.*z2$

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

Ex: node 27 "ed=en" => world_vector = 0011 ,positions 3 (w3) and 4 (w4)

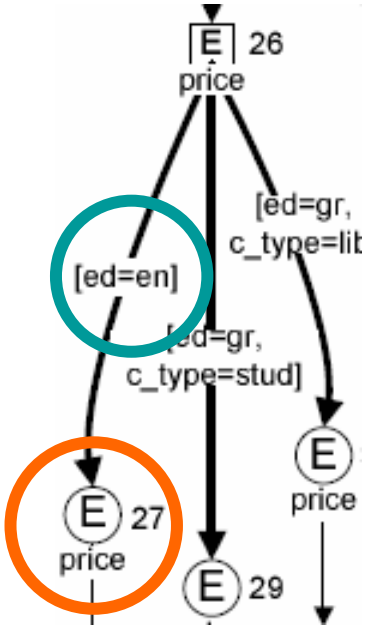
ordered worlds:

w1=(gr,stud)	w2=(gr,lib)	w3=(en,stud)	w4=(en,lib)

DBIS database & information systems group ionian university

Context Representation (6/7)

Ordered-Based
Representation of Context



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)

Ordered-Based
Representation of Context

Explicit Context Table:

Assigns an **explicit context** (expressed in binary format according to world vector representation) to a MXML node.

Explicit Context Table	
node_id	world_vector
1	1111
2	1111
3	1100
....	....
31	0100
....	....
43	1010
....	....

Inherited Context Coverage Table:

Assigns an **inherited context coverage** (expressed in binary format according to world vector representation) to a MXML node.

Inherited Coverage Table	
node_id	world_vector
1	1111
2	1111
3	1100
4	1100
5	0011
6	0011
....	....

Multidimensional XPath (MXPath) (1/2)

MXPath:

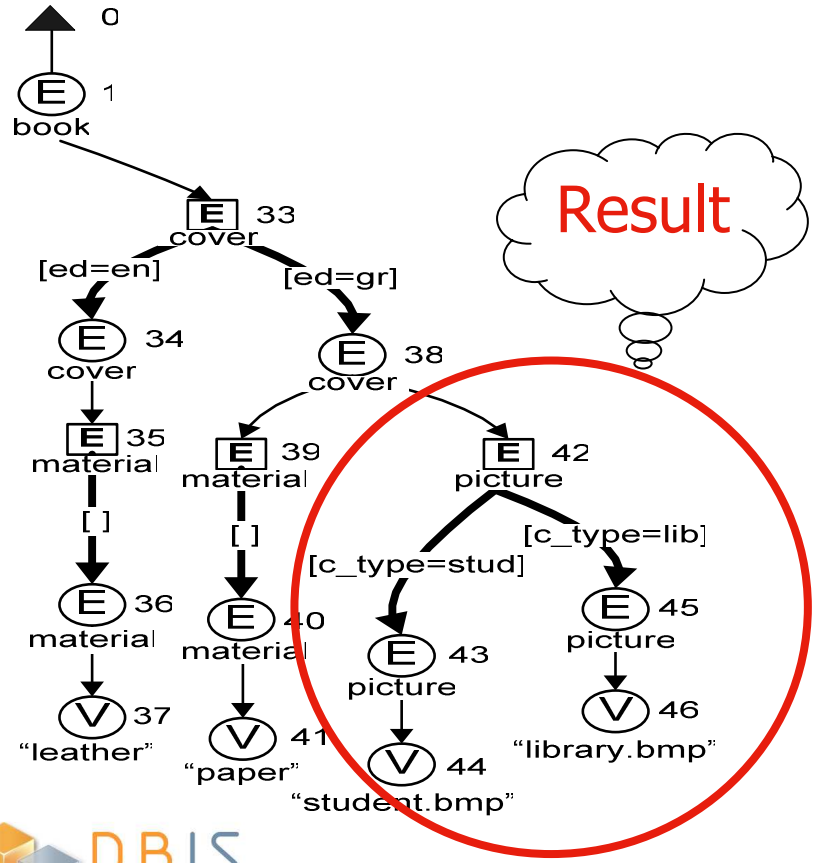
- An extension of XPath able to easily express context-aware queries on MXML data.
- Both explicit context (**ec**) and inherited context coverage (**icc**) are used to navigate over multidimensional elements and attributes.
- Conditions on the explicit context at any point of the path are allowed.
-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

Multidimensional XPath (MXPath) (2/2)

MXPath example:

[icc() >= “-”],/child::book

/child::cover[ec() >= “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() >= “ed=gr”]

is an *explicit context qualifier*. The function 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].

DBIS database & information systems group ionian university

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 Q1(stored),Q2(query) context specifiers and G(Q1),G(Q2) the binary **world vectors** of Q1,Q2

Comparing Q1 with Q2:

Q1=Q2 **<=>** G(Q1)=G(Q2) equivalently

Q1=Q2 **<=>** (G(Q1) XOR G(Q2))=0

Q1!=Q2 **<=>** NOT(G(Q1)=G(Q2))

Q1≥Q2 **<=>** (G(Q1) AND G(Q2))=G(Q2)

Q1>Q2 **<=>** ((G(Q1) AND G(Q2))=G(Q2)) AND (G(Q1)≠G(Q2))

Q1≤Q2 if Q2≥Q1 and Q1<Q2 if Q2>Q1

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

1. N. Fousteris, Y. Stavrakas, and M. Gergatsoulis. *Multidimensional XPath*. In *Proc. of iiWAS 2008* , pp. 162-169. ACM, 2008 .
2. Y. Stavrakas, and M. Gergatsoulis. *Multidimensional Semistructured Data: Representing Context-Dependent Information on the Web*. In *Proc. of CAiSE'02*, pp. 183-199, Springer 2002.
3. I. Tatarinov, S. Viglas, K. S. Beyer, J. Shanmugasundaram, E. J. Shekita, and C. Zhang. *Storing and querying ordered XML using a relational database system*. In *Proc. of the 2002 ACM SIGMOD Int.Conf. on Management of Data*, pp. 204-215. ACM, 2002.

Thank you..