Path-based MXML Storage and Querying

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
  - A path-based approaches is presented
- Context Representation
- Multidimensional XPath (MXPath)
- MXPath to SQL conversion algorithm
- 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 users/applications
  - XML queries are translated to SQL queries
  - SQL queries are executed
  - Results are translated back to XML and returned to the user/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***: an expression which 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

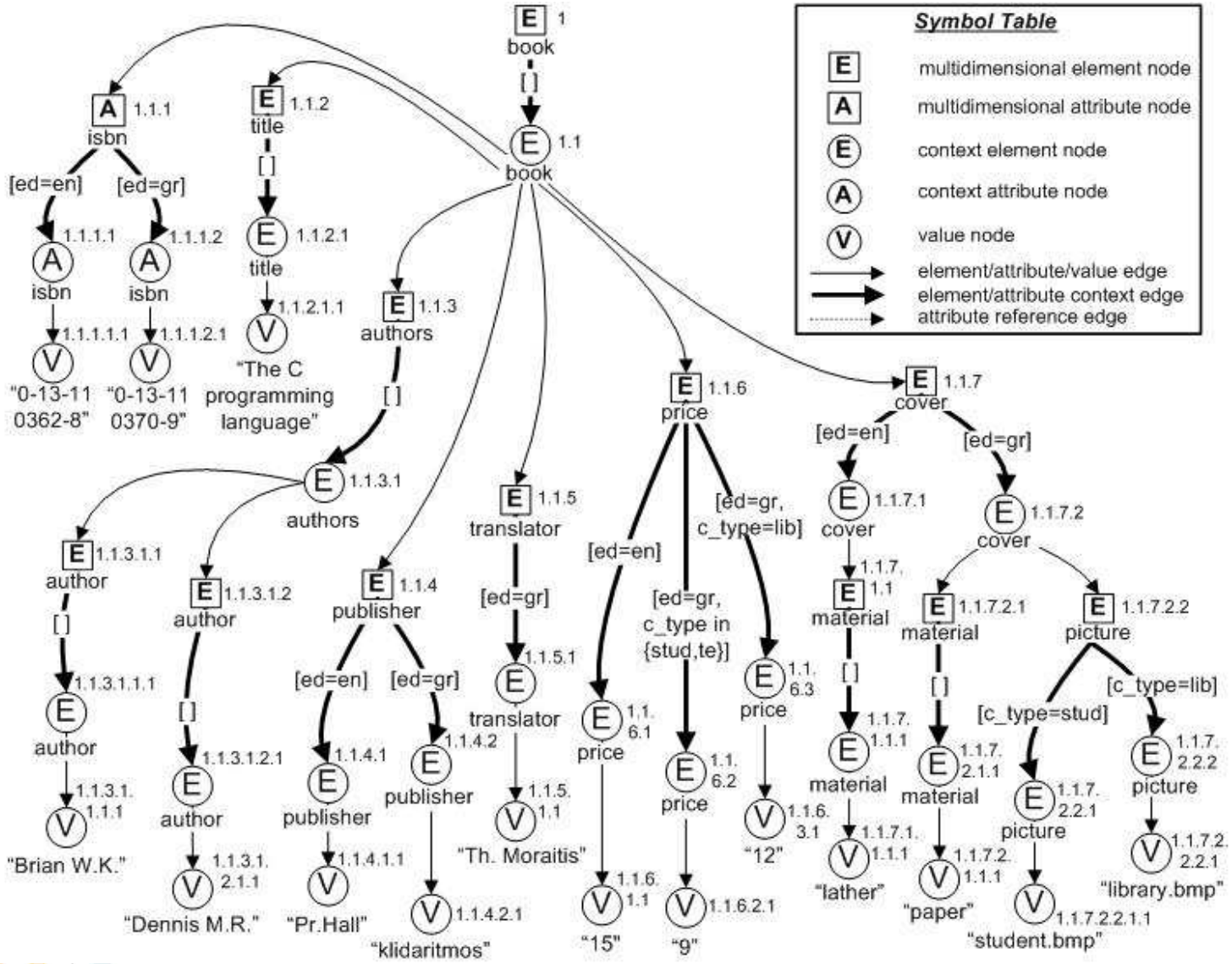
MXML – Example

```
<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 (defined by a context specifier) only within the boundaries of a single multidimensional element/attribute.
- ***Inherited Context***: Is the context, which is inherited from the ancestor nodes 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 (1/5)

- 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 (2/5)

Path-based Approach

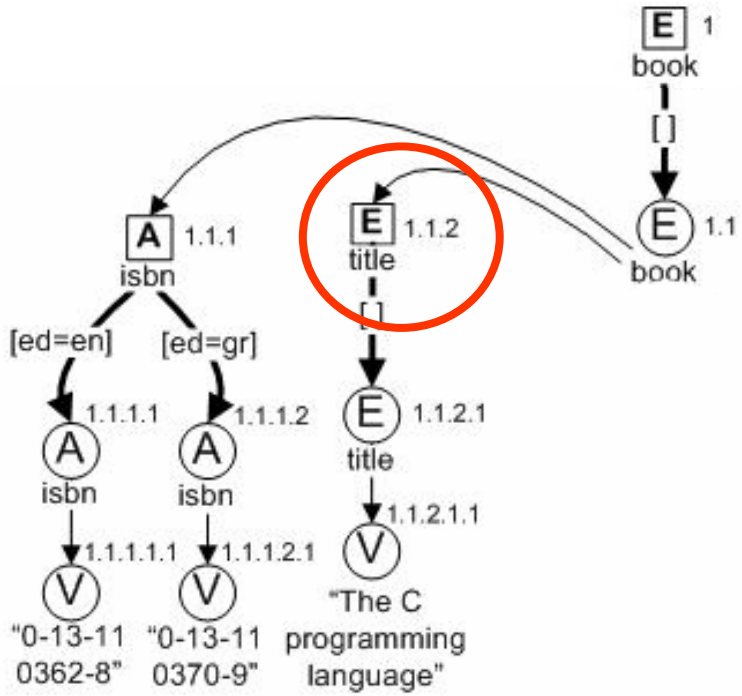
- MXML nodes are divided into groups, according to their types. Each group is stored in a separate table named after the type of the nodes (elements, attributes and value nodes).
- There is a *Path Table,* which stores all possible paths of the MXML tree.
- For node indexing, it is used a dotted format of Dewey-labeling schema.

MXML Storage (3/5)

Path-based Approach

Dewey-labeling schema

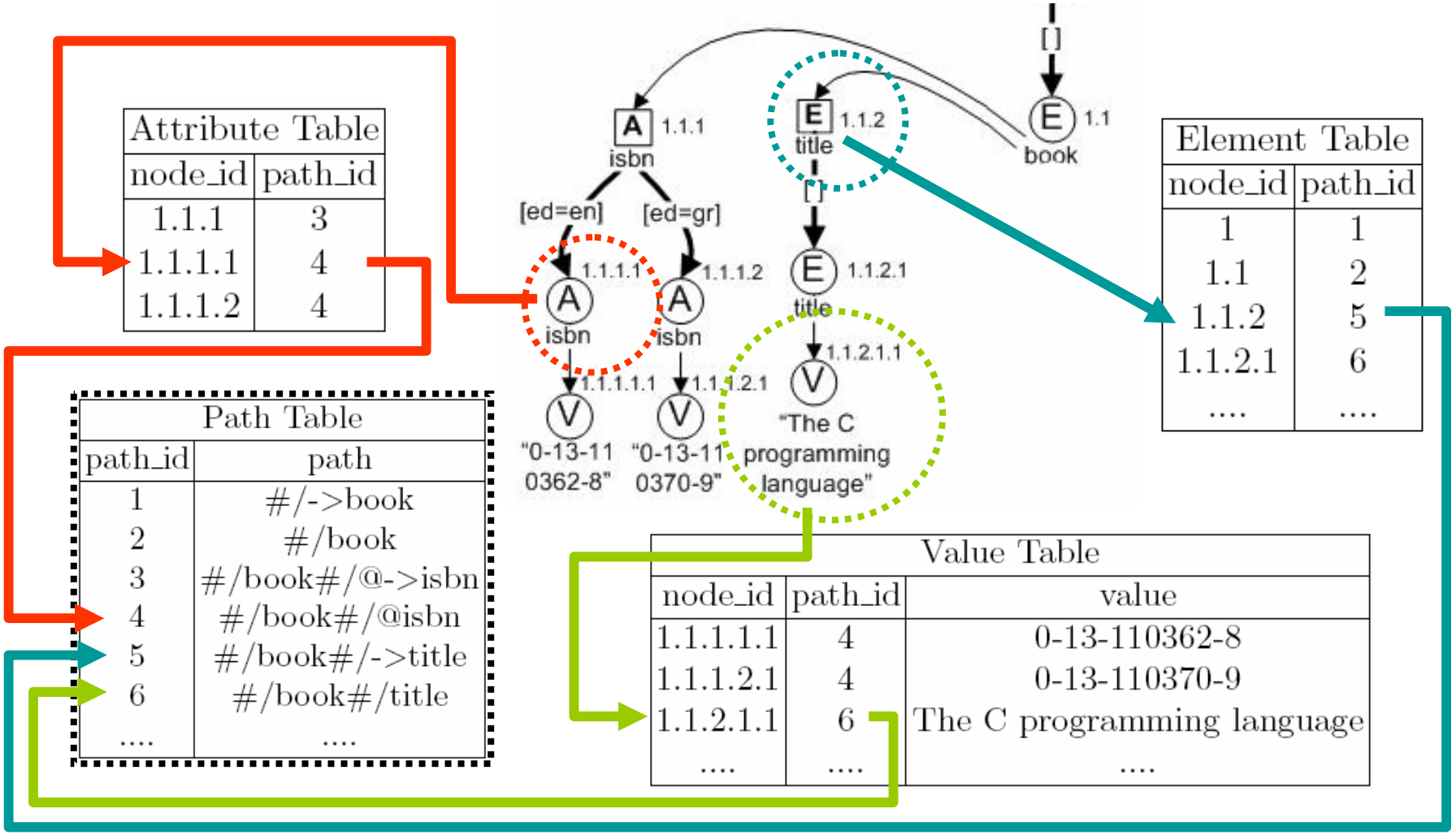
- Used for indexing the nodes of MXML tree.
- A label is a dotted string $a_1.a_2.a_3 \ldots a_n$.
- For each a_i (i=1…n), *i* represents the depth and a_i the position number of a node among its siblings.



Ex. Node "1.1.2" is the 2nd child of node "1.1" and is placed at the 3rd level of the MXML tree.

MXML Storage (4/5)

Path-based Approach



Attribute Table

node_id	path_id
1.1.1	3
1.1.1.1	4
1.1.1.2	4

Element Table

node_id	path_id
1	1
1.1	2
1.1.2	5
1.1.2.1	6
....	....

Path Table

path_id	path
1	#/->book
2	#/book
3	#/book#/@->isbn
4	#/book#/@isbn
5	#/book#/->title
6	#/book#/title
....	....

Value Table

node_id	path_id	value
1.1.1.1.1	4	0-13-110362-8
1.1.1.2.1	4	0-13-110370-9
1.1.2.1.1	6	The C programming language
....	....	....

MXML Storage (5/5)

Path representation

Path Table	
path_id	path
1	#/->book
2	#/book
3	#/book#/@->isbn
4	#/book#/@isbn
5	#/book#/->title
6	#/book#/title
....	....

Ex.

XPath: /book//picture

SQL wildcard

SQL1:

select.. from.. where path LIKE '/book%/picture'

case1: '/book/cover/picture' (match) correct

case2: '/booklet/cover/picture' (match) error

SQL2:

select.. from.. where path LIKE '#/book#%/picture'

Cases like *case2* above could not happen.

Context Representation (1/5)

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 (2/5)

Ordered-Based Representation of Context

Basic idea: Total ordering of worlds based on:

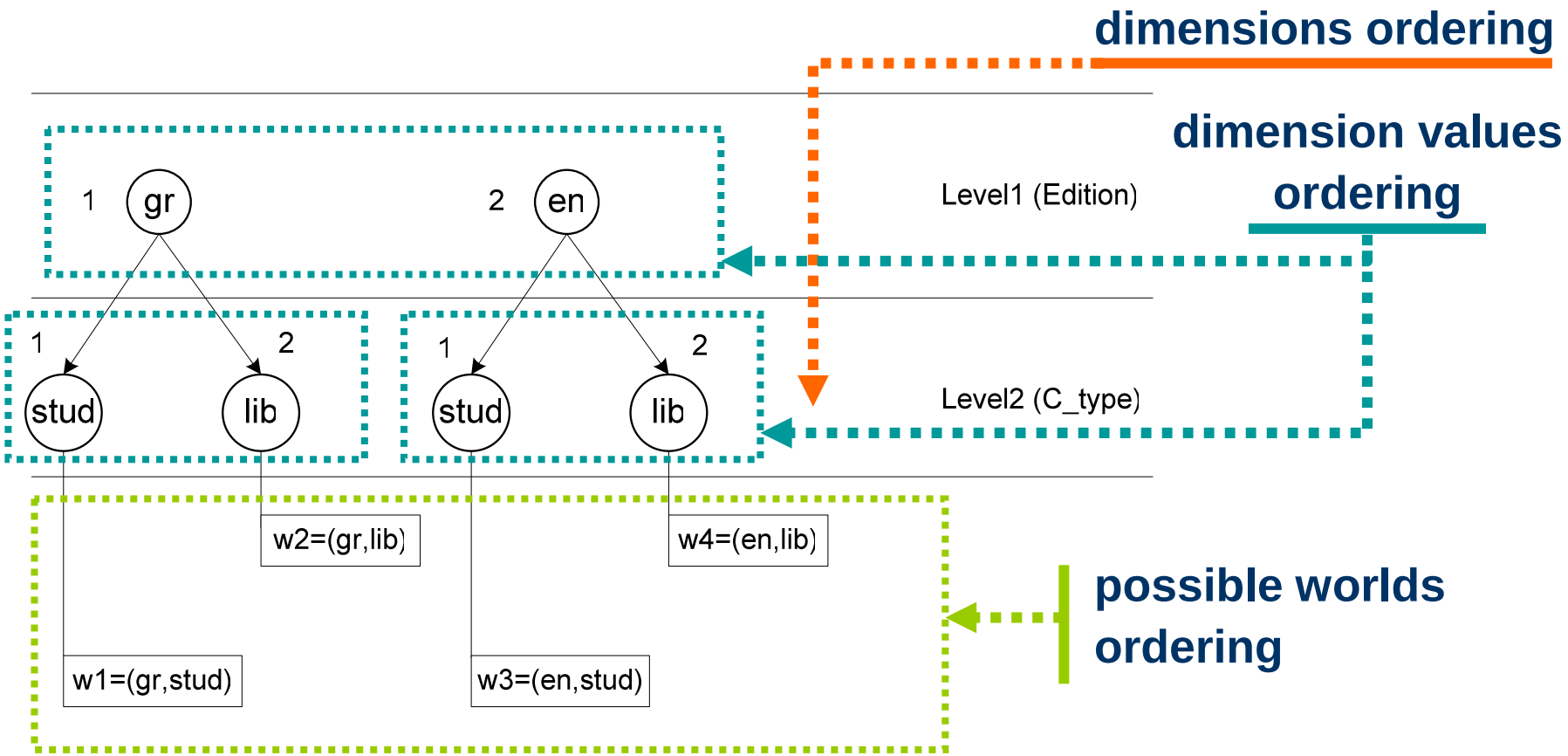
- Total ordering of dimensions
- Total ordering of dimension values

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).

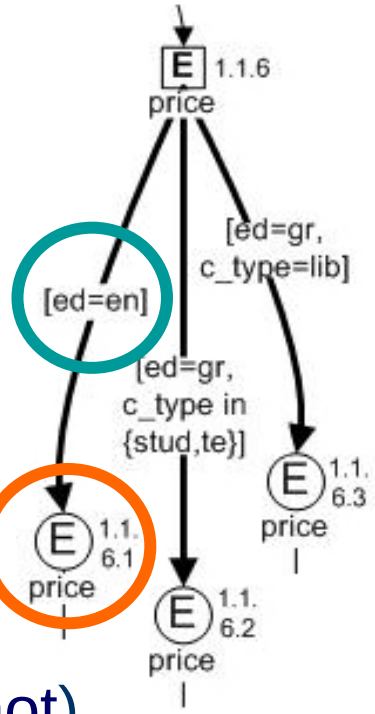
Context Representation (3/5)

Ordered-Based Representation of Context



Context Representation (4/5)

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)

binary digit for W1	……	binary digit for Wi 1 or 0: world exists or not	……	binary digit for Wn n=possible worlds number

possible worlds ordering

Ex: world_vector of the expl. context of node 1.1.6.1 = 0011

Context Representation (5/5)

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.

EC_Table	
node_id	world vector
....	....
1.1.7.2	000111
1.1.7.2.1	111111
1.1.7.2.1.1	111111
1.1.7.2.2	111111
1.1.7.2.2.1	100100
....	....

Inherited Context Coverage Table:

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

ICC_Table	
node_id	world vector
....	....
1.1.7.2	000111
1.1.7.2.1	000111
1.1.7.2.1.1	000111
1.1.7.2.2	000101
1.1.7.2.2.1	000100
....	....

Multidimensional XPath (MXPath) (1/2)

MXPath:

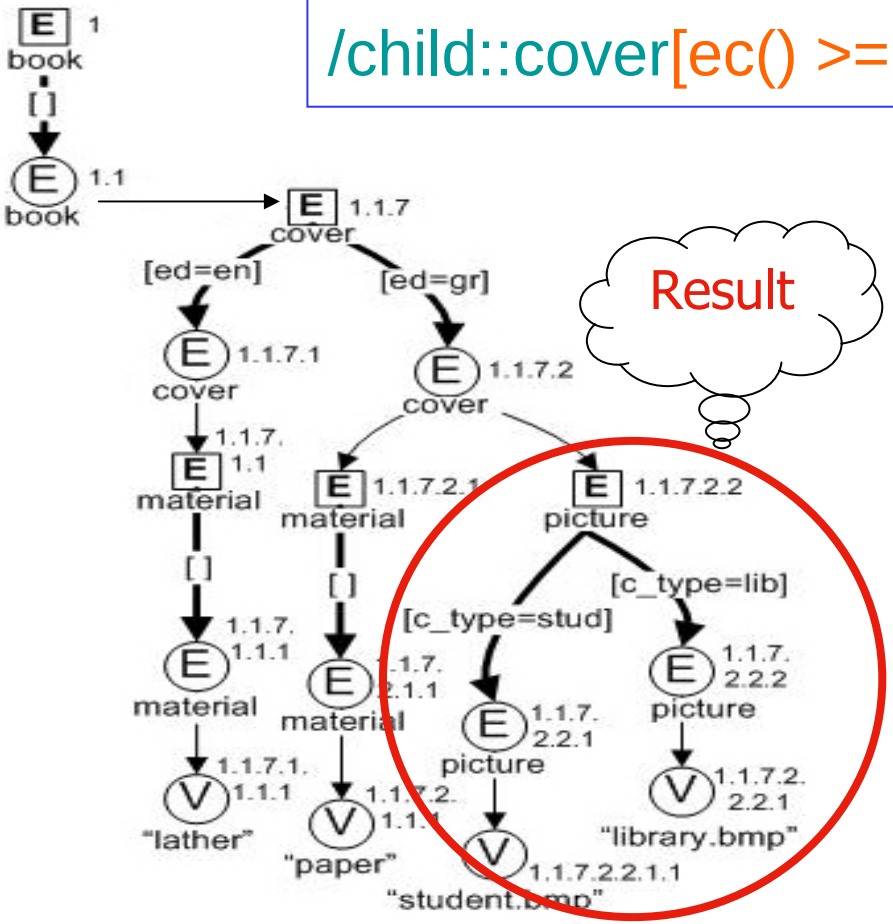
- An extension of XPath able to easily express context-aware queries on MXML data.
- Both explicit context (**ec**) and inherited context coverage (**icc**) are used to navigate over multidimensional elements and attributes.
- Conditions on the explicit context at any point of the path are allowed.
-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

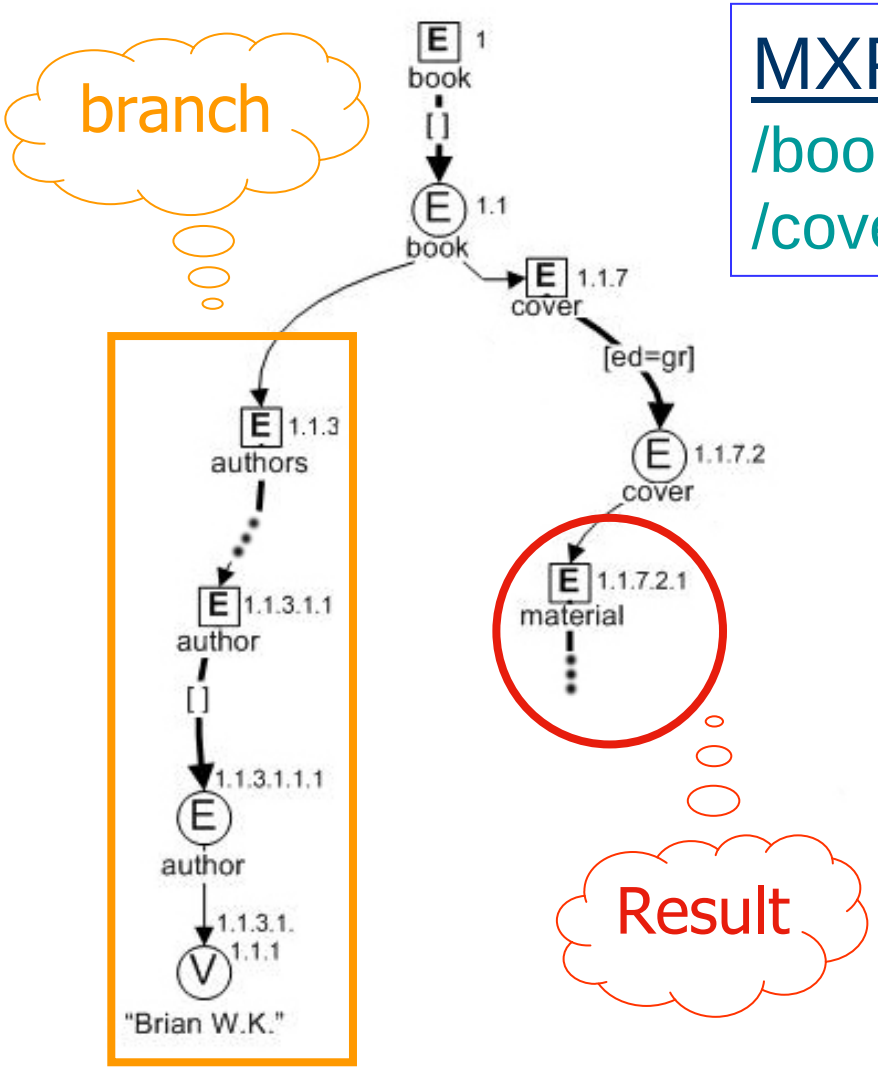
is an *explicit context qualifier*. The function ec() returns the explicit context of a node. The above qualifier says that the ec of the node cover must be superset of the context described by the context specifier [ed=gr].

MXPath to SQL conversion algorithm (1/4)

Methodology:

- Give a MXPath query “Q”, we construct the Multidimensional Tree Pattern “G” of Q.
- We divide “Q” in sub-paths according to segmentation rules, which define the appropriate segmentation points in “G”.
- The sub-paths of “Q” and the predicates are used as input for the MXPath to SQL conversion algorithm.

MXPath to SQL conversion algorithm (2/4)



MXPath example:

/book[authors[author="Brian W.K."]]
/cover[ec()="ed=gr"]/->material

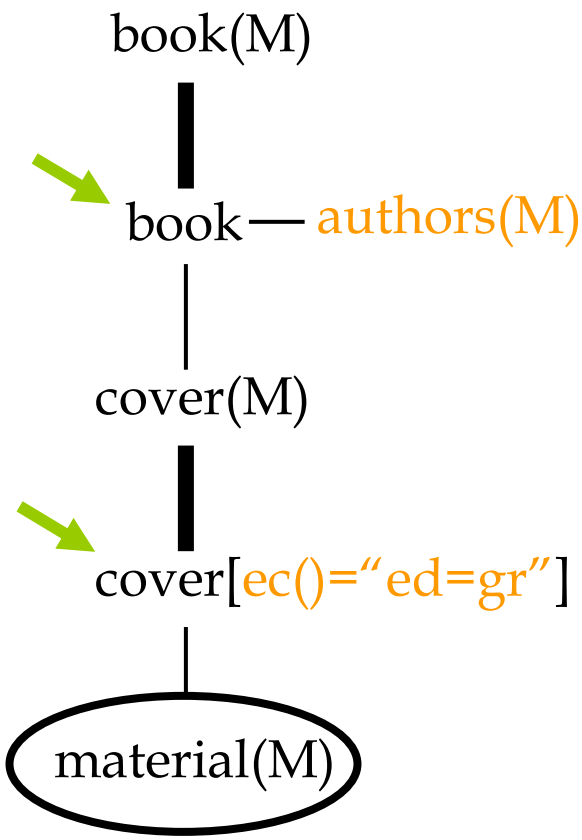
Query in English:

Find the (multidimensional) sub-element material of element cover of the greek edition of the book which has an author named "Brian W.K.".

Notice that the predicate [authors[author="Brian W.K."]] is a branch.

MXPath to SQL conversion algorithm (3/4)

Multidimensional Tree Pattern



MXPath query "Q":

/book[authors[author="Brian W.K."]]
/cover[ec()="ed=gr"]/->material

Segmentation (Q):

Segmentation

branch (authors[author="Brian W.K."]):

Sub_Path_1: #/book#/authors#/author

Value_1="Brian W.K."

MXPath to SQL conversion algorithm (4/4)

MXPath: /book[authors[author="Brian W.K."]]/cover[ec()="ed=gr"]/->material

SQL:

```
Select a3.node_id
 From Element_Table a1, Element_Table a2,
        Element_Table a3, Path_Table p1,
        Path_Table p2, Path_Table p3,
        EC_Table ec1
 Where p1.path Like '#/book' and
  a1.path_id = p1.path_id and
  a1.node_id = ANY (
  Select SUBSTRING_INDEX(a_id,'.', 2)
  From
  (Select a1.node_id AS a_id
    From Element_Table a1, Path_Table p1,
    Value_Table v1
    Where p1.path Like '#/book#/authors#/author' and
    a1.path_id = p1.path_id and
    v1.path_id = p1.path_id and
    v1.value = 'Brian W.K.' and
    v1.node_id Like CONCAT(a1.node_id,'%')
   ) AS T
  ) and
  a2.node_id Like CONCAT(a1.node_id,'%') and
  p2.path Like '#/book#/cover' and
  a2.path_id = p2.path_id and
  a2.node_id = ec1.node_id and
  ec1.world_vector LIKE '000111' and
  a3.node_id Like CONCAT(a2.node_id,'%') and
  p3.path Like '#/book#/cover#/->material' and
  a3.path_id = p3.path_id
```

Summary

- MXML
- Storing MXML in Relational DB & Context Representation (path-based approach)
- MXML querying using MXPath
- Converting MXPath queries to SQL queries

Future work

- Conversion Algorithm implementation and evaluation
- Further optimization of the relational schema

References

1. N. Fousteris, Y. Stavrakas, and M. Gergatsoulis. *Multidimensional XPath*. In *Proc. of iiWAS 2008* , pp. 162-169. ACM, 2008 .
2. M. Gergatsoulis, Y. Stavrakas, and D. Karteris. *Incorporating Dimensions in XML and DTD*. In Database and Expert Systems Applications, 12th International Conference, *DEXA 2001* Munich, Germany, September 3-5, 2001, Proceedings, volume 2113 of Lecture Notes in Computer Science, pp. 646-656. Springer, 2001.
3. M. Yoshikawa, T. Amagasa, T. Shimura, and S. Uemura. *XRel: a path-based approach to storage and retrieval of XML documents using relational databases*. ACM Transactions on Internet Technology, 1(1):110-141, 2001.

Thank you..