Information Management

XML and Databases

hussein suleman

uct cs 303 2004

XQuery

- XQuery specifies advanced functional queries over XML documents and collections.
- XQuery is a superset of XPath 1.0, and parallel specification for XPath 2.0.
- Not yet a standard!
  - Only Working Draft as of 23 July 2004.
  - Expect some changes before finalisation.

XQuery Expressions 1/2

- Primary expressions
  - `12.1, “Hello world”` (literals)
  - `$firstauthor` (variable)
  - `xq:string-concat ()` (function call)
- Path expressions
  - `document(“test.xml”)//author`
  - `para[5][@type="warning"]`
  - `child::chapter[child::title='Intro']`

XQuery Expressions 2/2

- Arithmetic/Comparison/Logic expressions
  - `$unit-price - $unit-discount`
  - `//product[weight gt 100]`
  - `1 eq 1 and 2 eq 2`
- Sequence expressions
  - `(1, 2, (3))`
  - `(10, 1 to 4)`
  - `(1 to 100)[. mod 5 eq 0]`
  - `$seq1 union $seq2`

FLWOR Expressions

- For-Let-Where-OrderBy-Return
- Iterates over a sequence of nodes, with intermediate binding of variables.
- Most useful for database-like “join” operations.

FLWOR Example

```
for $d in fn:doc("depts.xml")//deptno
let $e := fn:doc("emps.xml")//emp[deptno = $d]
where fn:count($e) >= 10
order by fn:avg($e/salary) descending
return
   <big-dept>
      {
      $d,
      <headcount>{fn:count($e)}</headcount>,
      <avgsal>{fn:avg($e/salary)}</avgsal>
      }
   </big-dept>
                           (from specification)
```

FLWOR For, Let

- `for` and `let` create a sequence of tuples with bound variables.
- Can have multiple `for`s and multiple `let`s.
- Multiple `for`s result in a Cartesian product of the sequences.

  -
    ```
    for $car in ("Ford", "Chevy"),
        $pet in ("Cat", "Dog")
    ```
- Multiple `let`s result in multiple intermediate variable bindings per tuple of nodes.

FLWOR Where, OrderBy, Return

- `where` filters the list of tuples, by removing those that do not satisfy the expression.
- `return` specifies result for each tuple.
- `order by` specifies the expression to use to order the tuples – the expression can use nodes not included in the result.

  -
    ```
    for $e in $employees
    order by $e/salary descending
    return $e/name
    ```

FLWOR for DB Joins

```
<ucthons>
   {
      for $stud in fn:doc(“students.xml”)//student
      for $proj in
        fn:doc(“projects.xml”)//project[id = $stud/id]
      order by $stud/name
      return
         <honsproj>
           <studentname>{$stud/name}</studentname>
           <projectname>{$proj/name}</projectname>
         </honsproj>
   }
</ucthons>
```

XML Databases

- Databases must be Unicode-compliant! (usually UTF-8)
- Options:
  - Blob: Store XML documents or fragments in tables.
  - Tree: Store XML as sequence of nodes with child relationships explicitly indicated.
  - Relation: Store XML in specialised tables/relations as defined by XML structure.
  - Flat files: Store each XML document in a file.

Blob/Clob/etc.

Id	XMLBlob
TestXML	`<uct>` `<title>test XML document</title>` `<author email=“pat@cs.uct.ac.za” office=“410” type=“lecturer”>Pat Pukram</author>` `<version>` `<number>1.0</number>` `</version>` `</uct>`

Tree Representation

Nodes

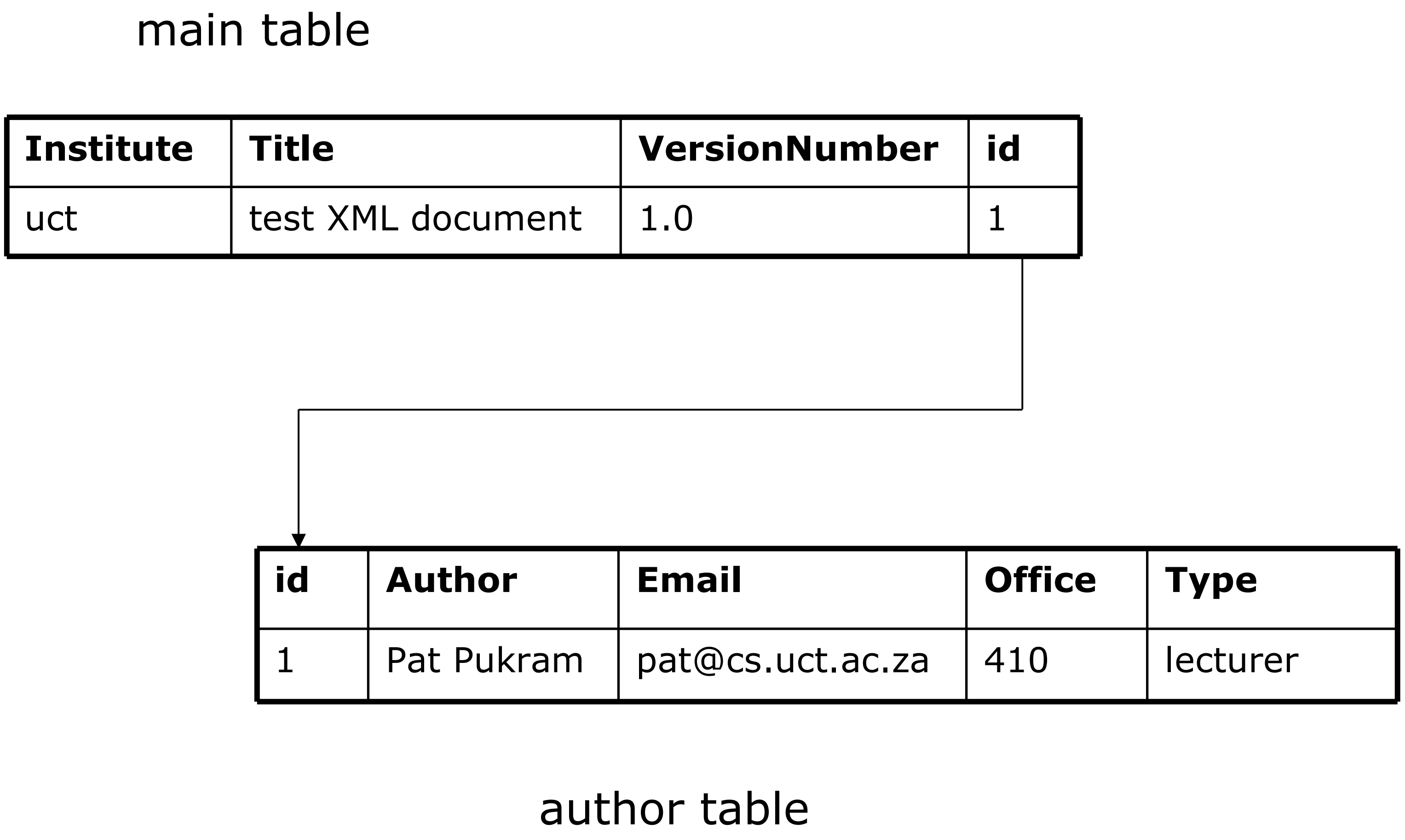
Id	Type	Label	Value
1	Element		uct
2	Element		title
3	Text		test XML document
4	Element		author
5	Attribute	email	pat@cs.uct.ac.za
6	Attribute	office	410
7	Attribute	type	lecturer
8	Text		Pat Pukram
9	Element		version
10	Element		number
11	Text		1.0

Links

Parent id	Child id
1	2
2	3
1	4
4	5
4	6
4	7
4	8
1	9
9	10
10	11

Note: Whitespace nodes have been ignored!

Relation Representation



main table

Institute	Title	VersionNumber	id
uct	test XML document	1.0	1

id	Author	Email	Office	Type
1	Pat Pukram	pat@cs.uct.ac.za	410	lecturer

author table

Evaluation

- Blob: fast insert/select for XML documents, but slow querying.
- Tree: fast location of single nodes and sequences of nodes, but slow to enforce structure of XML.
- Relation: fast data query and extraction, but could be many tables and thus slow to insert/select XML documents.
- Flat file: fast load/store, but slow queries.

Are we only interested in relational queries? Google-like queries?

References

- Boag, Scott, Don Chamberlin, Mary F. Fernández, Daniela Florescu, Jonathan Robie and Jérôme Siméon (2004). XQuery 1.0: An XML Query Language, W3C Working Draft 23 July 2004, W3C. Available http://www.w3.org/TR/xquery/