XQuery midflight: 
Emerging Database-Oriented Paradigms and a Classification of Research Advances

Ioana Manolescu
INRIA, France
Yannis Papakonstantinou
UCSD, USA

April 5, 2005
Outline

Tuple-based evaluation of XQuery
XQDMA: an abstraction of the XQuery data model
Unified tuple-based algebra
Related work
Tuple-based Evaluation of XQuery
Database Research Meets XQuery Processing

Database research has succeeded on querying large data volumes

Tuple-based algebras have been key ingredient
- Relational, OQL
- Algebraic cost-based optimization
- Set-at-a-time query processing primitives (eg join)

Several tuple-based algebras for XQuery
- similar, yet different

PRIMARY GOAL: Formal definition of a unifying tuple-based algebra for processing XQuery
- functional coverage

Better communication in research and education
The Need for XQuery Data Model Abstraction (XQDMA)

Formal XML query languages and algebras have used labeled tree abstraction

XQuery data model is extremely complex for research

SECONDARY GOAL: XQuery Data Model Abstraction

- Faithful to XQuery Data Model, leads to relevance to XQuery itself
- Yet modularization of issues enables redacted models
Abstraction of the XQuery Data Model
Sample XML Document

<department>
  <project name="NexT" url="next.org"/>
  <project name="KadoP" url="kadop.net"/>
  <faculty>
    <person> <inproject>NexT</inproject> <inproject>KadoP</inproject>
        <name><first>John</first><last>Smith</last></name>
        <mail>j@u.edu</mail>
    </person>
    <person>
        <name><first>Mary</first><last>Jones</last></name>
        <mail>m@u.edu</mail> <mail>m@acm.org</mail>
    </person>
  </faculty>
  <students>
    <person> <inproject>KadoP</inproject>
        <name><first>Lily</first><last>Liu</last></name> <mail>lil@u.edu</mail>
    <person>
  </students>
</department>
Sample XML Document

"group.xml"

- group
  - faculty
    - person
      - name: John, Smith, KadoP
        - inproject: Next
        - mail: j@u.edu
      - name: Mary, Jones, KadoP
        - inproject: Next
        - mail: m@u.edu
      - name: Lily, Liu, lil@u.edu
        - inproject: Next
        - mail: m@acm.org
XQDMA Definition

Labeled (ordered) trees

Nodes
- Four kinds: Document, element, attribute, text
- Single document node
- Element/attribute nodes labeled with XQDM element/attribute names
  - by convention, attribute names start with @
- Text nodes labeled with XQDM values
- Nodes have unique identities
- Type function $T$ labels every node with XQDM type

Edges
- Document node is root and has exactly one child, which is element
- Attribute nodes may appear only as children of element nodes
- Attribute nodes may only have text node children
- Text nodes may only be leaves
Equality Relationships (1/2)

Node ID-based equality $=_{id}$ (XQuery $is$)
- Two nodes are id-equal if they are the same

Value-based equality $=_{v}$ (XQuery $eq$)
- Two values are equal if the results of casting one or both into a common domain are equal. Depends on values' types.
- 24 atomic types [XQDM,XSch]; casting rules in [XQFO]
Equality Relationships (2/2)

Node ID-based equality $=_{\text{id}}$ (XQuery is)

Value-based equality $=_{\text{v}}$ (XQuery eq)

Limited value-based equality

- Text nodes are value-based equal if their labels are equal
- Attribute/Element nodes $n_1$ and $n_2$ are value-based equal if their labels are equal and
  - (unordered) for every child of $n_1$, there is an equal child of $n_2$ and vice versa
  - (obvious generalization to ordered)

Limited value-based equality implies Value-based equality

Not vice-versa (eg, text node with "005" is equal to text node with "05", considering typing and coercion)
Order Relationship

Node order relationship \( \ll (\text{XQuery} \ before) \) in ordered

- Parent before children
- Attribute nodes directly follow parent (precede non-attribute nodes)
- Undefined order between attributes
Node and Value Comparisons

- document node
- element node
- attribute node
- text node

Diagram:

- r1 "group.xml"
- g1 group
  - faculty
    - j1
    - j2
  - project
    - n1 KadoP
    - n2 NextT
  - inproject
    - i1
    - i2
    - mail
      - m1
    - name
      - n1 John
  - project
    - n2 KadoP
    - n3 kadop.net
  - inproject
    - i1
    - i2
    - mail
      - m1
    - name
      - n3
  - project
    - n1
    - n2
  - students
    - s1
  - name
    - n5
  - inproject
    - i1
    - i2
    - mail
      - m1
    - name
      - n5
  - person
    - p1
    - p2
    - p3
  - name
    - n4
    - n5
  - mail
    - m1
    - m2
    - m3
  - URL
    - u2
    - u1
  - name
    - n4
    - n5
  - mail
    - m1
    - m2
    - m3
  - URL
    - u2
    - u1
  - name
    - n4
    - n5
  - mail
    - m1
    - m2
    - m3
  - URL
    - u2
    - u1
Equality Comparisons

\( t_1 \neq \text{id } t_5 \)
\( t_1 =_v t_5 \)
\( t_5 =_v t_{12} \)
Equality Comparisons

- t1 ≠ id t5
- t1 = v t5
- t5 = v t12
- i1 ≠ id i3
- i1 = v i3
Order Comparisons

\[ j_1 \ll n_1, j_1 \ll u_1 \]
\[ p_1 \ll i_1 \ll t_5 \ll i_2 \ll t_6 \]
\[ u_1 \ll n_1, n_1 \ll u_1 \]
Equalities on XQDMA Lists

Deep-equal =

- Two lists are deep-equal if they have the same length and their items at corresponding positions are value-based equal
- The "=" of XQuery does not translates to existential equality comparison

Existential list equality (ele) comparison =\exists

- l1 =\exists l2 iff \exists o1 \in l1, o2 \in l2 such that o1 =_v o2
- ele not transitive
Generalization to Other Operators

Other relationships over nodes and lists

≠, >, <, ...

In limited value-based comparisons are induced by corresponding relationships on values
Operations on Lists

\[ l_1 = [ i_1, i_1, i_3 ] \]
\[ l_2 = [ i_1, t_5 ] \]
\[ l_3 = [ i_1, t_12 ] \]
\[ l_4 = [ i_3, t_5, "Hello" ] \]
Operations on Lists

\[ l_2 = l_3 \]

\[ l_1 = [i_1, i_1, i_3] \]
\[ l_2 = [i_1, t_5] \]
\[ l_3 = [i_1, t_{12}] \]
\[ l_4 = [i_3, t_5, "Hello"] \]
Operations on Lists

\[ l_1 = \{ i_1, i_1, i_3 \} \]
\[ l_2 = \{ i_1, t_5 \} \]
\[ l_3 = \{ i_1, t_{12} \} \]
\[ l_4 = \{ i_3, t_5, "Hello" \} \]
Operations on Lists

\[ l_1 = [i_1, i_1, i_3] \]
\[ l_2 = [i_1, t_5] \]
\[ l_3 = [i_1, t_{12}] \]
\[ l_4 = [i_3, t_5, "Hello"] \]

\[ l_1 \not\subseteq l_2 \]
Operations on Lists

\[ l_2 = \exists l_3 \]

\[ l_1 = [i_1, i_1, i_3] \]
\[ l_2 = [i_1, t_5] \]
\[ l_3 = [i_1, t_{12}] \]
\[ l_4 = [i_3, t_5, "Hello"] \]
operations on lists

\[ l_2 \neq l_3 \]

\[ l_1 = [i_1, i_1, i_3] \]
\[ l_2 = [i_1, t_5] \]
\[ l_3 = [i_1, t_{12}] \]
\[ l_4 = [i_3, t_5, "Hello"] \]
Operations on Lists

\[ l_1 = [i_1, i_1, i_3] \]
\[ l_2 = [i_1, t_5] \]
\[ l_3 = [i_1, t_{12}] \]
\[ l_4 = [i_3, t_5, "Hello"] \]

\[ l_1 \supseteq l_3 \]
Operations on Lists

- \( l_1 = [i_1, i_1, i_3] \)
- \( l_2 = [i_1, t_5] \)
- \( l_3 = [i_1, t_{12}] \)
- \( l_4 = [i_3, t_5, "Hello"] \)

\( l_1 \not\subseteq l_3 \)
Operations on Lists

\( l_1 = [i_1, i_1, i_3] \)
\( l_2 = [i_1, t_5] \)
\( l_3 = [i_1, t_{12}] \)
\( l_4 = [i_3, t_5, "Hello"] \)

\( l_1 \subseteq l_4 \)
Operations on Lists

\[ l_1 = [i_1, i_1, i_3] \]
\[ l_2 = [i_1, t_5] \]
\[ l_3 = [i_1, t_{12}] \]
\[ l_4 = [i_3, t_5, "Hello"] \]

\[ l_1 \not\subseteq l_4 \]
Operations on Lists

\[ l_1 = [i_1, i_1, i_3] \]
\[ l_2 = [i_1, t_5] \]
\[ l_3 = [i_1, t_{12}] \]
\[ l_4 = [i_3, t_5, "Hello"] \]

\[ l_1 \cup l_1 = [i_1, i_3] \]
Operations on Lists

\[ l_1 = [i_1, i_1, i_3] \]
\[ l_2 = [i_1, t_5] \]
\[ l_3 = [i_1, t_{12}] \]
\[ l_4 = [i_3, t_5, "Hello"] \]
\[ l_1 \# l_1 = [i_1, i_1, i_3, i_1, i_1, i_3] \]
Operations on Lists

\[ l_1 = [i_1, i_1, i_3] \]
\[ l_2 = [i_1, t_5] \]
\[ l_3 = [i_1, t_{12}] \]
\[ l_4 = [i_3, t_5, "Hello"] \]
\[ l_1 \cap l_2 = [i_1] \]
A Quick Refresh of Your XPath and XQuery
Path Expressions

In the second chapter of the document zoo.xml find the figures with caption “Tree Frogs”

./book//chapter[2]//figure[caption=“tree frogs”]
More Path Expressions

Find the first immediate chapter subelements of immediate part subelements of the document zoo.xml and retrieve figures that have ...

doc("zoo.xml")/part/chapter[1]//figure[caption="Tree Frogs"]
In the second chapter of the document zoo.xml find the figures with caption “Tree Frogs” and place them into an element called result.

```xml
<result>
{
    doc("zoo.xml")//chapter[2]//figure[caption="Tree Frogs"]
}
</result>
```

```
result
| figure
| caption
| “Tree Frogs”
```
Bibliography Example Data Set

<bib>
  <book>
    <author> Aho </author>
    <author> Hopcroft </author>
    <author> Ullman </author>
    <title> Automata Theory </title>
    <publisher> Morgan Kaufmann </publisher>
    <year> 1998 </year>
  </book>
  <book>
    <author> Ullman </author>
    <title> Database Systems </title>
    <publisher> Morgan Kaufmann </publisher>
    <year> 1998 </year>
  </book>
  <book>
    <author> Abiteboul </author>
    <author> Buneman </author>
    <author>Suciu </author>
    <title> Automata Theory </title>
    <publisher> Prentice Hall </publisher>
    <year> 1998 </year>
  </book>
</bib>
Reviews Example Data Set

<reviews>
  <review>
    <title> Automata Theory </title>
    <comment> It’s the best in automata theory </comment>
    <comment> A definitive textbook </comment>
  </review>
  ...
</reviews>
For-Let-Where-Return (FLWR)

List the titles of books published by "Morgan Kaufmann"

FOR $b$ in doc("bib.xml")/book
WHERE $b$/publisher = "Morgan Kaufman"
RETURN $b$/title
Think (tuples of) variable bindings

FOR/LET

WHERE

RETURN

Ordered lists of tuples of variable bindings

Tuples of that satisfy the conditions

List of trees

$b$

book

book

book

book

$\text{title}$

$\text{title}$

year
FOR $b$ in doc("bib.xml")/book
WHERE $b$/year > 1990
RETURN $b$/author

Return the list of authors who published after 1990
Tuples

List publishers who have published more than 1 book

FOR $p$ in distinct(doc("bib.xml")//publisher)
LET $b := document("bib.xml")//book[publisher = $p]$
WHERE count($b) > 1
RETURN $p

Tuples ($p$, $b$) are formulated
Boolean Expressions in WHERE

FOR $b$ in doc("bib.xml")/book
WHERE $b$/publisher = "Morgan Kaufmann"
   AND $b$/year = "1998"
RETURN $b$/title

List the titles of books published by "Morgan Kaufmann" in 1998.
An Unified Tuple-Based Algebra for XQuery
Unified Data model (UDM) Extending XQDMA

(XQDMA) lists \( l = [o_1, o_2, ..., o_n] \)

\( o_i \) are XQDMA nodes

Tuples \( t = (v_1 = a_1, v_2 = a_2, ..., v_n = a_n) \)

"(in \( t \)) the variable \( v_i \) binds to variable binding \( a_i \)"

\( t.v_i \) may be: an XQDMA list, or

a set/bag/list (collection) of homogenous tuples

\( (v_1, v_2, ..., v_n) \): tuple schema
UDM Tuple Equality

b1 = (v1=[p1], v2=[t4], v3=[i1, t5])
b2 = (v1=[p1], v2=[t11], v3=[i3, t14])

b1 = b2
UDM Operations on Tuples

t_1 = (v_1 = a_1, ..., v_n = a_n), t_2 = (u_1 = b_1, ..., u_n = b_n)

\( t_1 + (v_{n+1} = a_{n+1}) = (v_1 = a_1, ..., v_n = a_n, v_{n+1} = a_{n+1}) \)

t_1 = t_2 \text{ iff:}

- n = m
- for every \( i = 1, ..., m \), one of the following holds:
  - \( t_1.v_i \) and \( t_2.u_i \) are \textit{values} and \( t_1.v_i = v t_2.u_i \)
  - \( t_1.v_i \) and \( t_2.u_i \) are \textit{nodes} and \( t_1.v_i = id t_2.u_i \)
  - \( t_1.v_i \) and \( t_2.u_i \) are \textit{lists / sets / bags} and \( t_1.v_i = t_2.u_i \)
Unified Tuple-Based Algebra Operators
Unified tuple-based algebra operators

Navigation
XML construction
Nested plans
Relational-style operators
Other operators
XPath navigation

Based on tree patterns

$R$/faculty/person \quad $R$/\ast/person \quad $R$/person[email]/name

$T_F$ *
faculty

$T_F$ *
s$P$:person

$T_P$ *

$P$:person

$T_P$ *

$P$:person

$T_N$ *

person

e-mail

$N$:name

Navigation operator $\text{nav}_{R,T_F} (\text{Tuple Collection}): \text{Tuple Collection}$
XPath navigation

$R//person[mail]/name

"group.xml"

person

inproject

NexT

j@u.edu

mail

group

mail

name

John

N e x T

m@u.edu

m@acm.org

Mary

T_N

name

$N:name

mail

java

$N

$R

r1

n3

n4

$r

[r1]
Tree patterns capture navigation of "for"

for \( \$P \) in \( \$R//person \), \( \$M \) in \( \$P/mail \), \( \$N \) in \( \$P/name \)
return \{ \( \$M \), \( \$N \) \}
Generalized "for" navigation

for $P$ in $R$ //person, $M$ in $P$/mail, $N$ in $P$/name
return { $M$, $N$ }

\[
\begin{array}{ccc}
  & P & M & N \\
 r1 & p1 & m1 & n3 \\
p2 & m2 & n4 \\
p2 & m3 & n4 \\
\end{array}
\]
Generalized "for" navigation

for $P$ in $R$//person, $M$ in $P$/mail, $N$ in $P$/name
return { $M$, $N$ }
Generalized "for" and "where" navigation

for $P$ in $R//person, $N$ in $P/name$
where $P/email$
return { $M, $N }

\[
\begin{array}{c|c}
  $P$ & $N$\\
  p1 & n3 \\
p2 & n4 \\
p2 & n4 \\
\end{array}
\]
XML result construction

for $P$ in $R$/person,
$N$ in $P$/name,
$M$ in $P$/mail
return <person>
  { $M$, $N$ }
</person>

$P$ : person
  $M$ : mail
  $N$ : name

$R$, $T_N$

$T_C$ : person
  $M$ : mail
  $N$ : name

$T_C$ : crList
$T_C$ : nav

$P$ : $M$ : $N$

$p1$ : $m1$ : $n3$
p2 : $m2$ : $n4$
p2 : $m3$ : $n4$
j@u.edu : John
m@u.edu : Mary
m@acm.org : Mary
j@u.edu : t9'
m@u.edu : t12'
m@acm.org : t13'
t6'
t10'
t12'
t10'
t13'

John
Mary
Mary

$P$ : person
$M$ : mail
$N$ : name

$T_N$ : *

$P$ : person
$M$ : mail
$N$ : name
XML result construction

for $P$ in $R$/person,
$N$ in $P$/name,
$M$ in $P$/mail
return <person>
   { $M$, $N$ }
</person>

Back to XQDMA
for $P$ in $R$//person
return
<person>
  { for $N$ in $P$//name
    return
      { $N
      }
  }
</person>

```plaintext
$P : person$
$N : name$
```

**Nested plans and the `apply` operator (1)**

```
for $P$ in $R$ // person
return
<person>
  { for $N$ in $P$ // name
    return
      { $N
      }
  }
</person>
```

```
$P : person$
$N : name$
```
for $P$ in $R$/person

```
for $N$ in $P$/name
    return
        { $N$, $P$/mail }

</person>
```
for $P$ in $\$R$//person

\[\text{return} \]

\[\{ \text{for } N \text{ in } P/\text{name} \]

\[\text{return} \]

\[\{ N, P/\text{mail} \} \]

\[\} \]

\[\}\]

\[\text{app} P1 \rightarrow \$1 \]

\[\text{nav} \$R, TP \]

\[\]
Nested plans and *let* clauses

Previous query:

```plaintext
for $P$ in $R$ //person
return
<person>
{ for $N$ in $P/name$
  return
   { $N, $P/mail } }</p person>
```

Same query with *let* clauses:

```plaintext
for $P$ in $R$ //person
let $L1 =
{ for $N$ in $P/name$
  let $L2 = $P/mail
  return { $N, $L2 } } return
<person>
  { $L1 } </p person>
```
Nested plans and \textit{let} clauses

Same query with \textit{let}:

for $P$ in $R$\texttt{//person}
let $L1 =$
\begin{verbatim}
  $P1$ for $N$ in $P$\texttt{/name}
    let $L2 =$
      \{ $N$, $P$\texttt{/mail} \}
    return \{ $L2$ \}
  \}
return
<person>
  \{ $L1$ \}
</person>
Nested plans and *let* clauses

Same query with *let*:

```sql
for $P in $R//person
let $L_1 = 
  { for $N in $P/name
    let $L_2 = $P/mail 
    return { $N, $L_2 } 
  }
return
<person>
  { $L_1 } 
</person>
```
Nested plans and *let* clauses

\[ \mathcal{P}_1 \rightarrow \mathcal{L}_1 \]

\[ \mathcal{P}_2 \rightarrow \mathcal{L}_2 \]

\[ \mathcal{P}_1 \]

\[ \mathcal{P}_2 \]

\[ T_P \]

\[ T_N \]

\[ T_{N2} \]

\[ T_M \]

\[ \mathcal{P}_1 \]

\[ \mathcal{P}_2 \]

\[ \text{crList} \]
Nested plans and *let* clauses

Nested queries can be equivalently rewritten using *let* clauses until return clauses reach the form:

\[
\{ \$V_1, \$V_2, ..., \$V_K \} \quad \text{or} \quad \text{<tag>} \{ \$V_1, \$V_2, ..., \$V_k \} \text{ </tag>}
\]

Nested queries with *let* can be "automatically" translated

- *for* → *nav*
- *let* → *apply*
- *return* → *crList*
Nested plans and optional navigation

Capture all navigation with a single pattern
- optional edges
- null (⊥) variable values

for $P$ in $R$ //person
return
<person>
  { $P/mail } 
</person>

groupBy [ $P ] [] → $G

$P : person$
$M : mail$

$R : *$

$P$ | $G$ | $P$ | $M$
---|---|---|---
p1 | [ | p1 | m1 |
| | | | ]
p2 | [ | p2 | m2 |
| | | | ]
p3 | [ | p3 | m3 |
| | | | ]
p3 | [ | p3 | ⊥ |
| | | | ]
Nested plans and optional navigation

Capture all navigation with a single pattern

- optional edges
- null (⊥) variable values

for $P$ in $R$//person
return
<person>
  { $P$/mail }
</person>

$P$ in $R$ // person

$P$ person

$M$ mail

$R$ : $*$

$P$ : $person$

$M$ : $mail$

$R$ : $*$

$P$ : $person$

$M$ : $mail$

$P$ : $person$

$M$ : $mail$

$R$ : $*$

$P$ : $person$

$M$ : $mail$

$P$ : $person$

$M$ : $mail$

$P$ : $person$

$M$ : $mail$

$P$ : $person$

$M$ : $mail$

$P$ : $person$

$M$ : $mail$

$P$ : $person$

$M$ : $mail$

$P$ : $person$

$M$ : $mail$

$P$ : $person$

$M$ : $mail$

$P$ : $person$

$M$ : $mail$

$P$ : $person$

$M$ : $mail$

$P$ : $person$

$M$ : $mail$

$P$ : $person$

$M$ : $mail$

$P$ : $person$

$M$ : $mail$

$P$ : $person$

$M$ : $mail$

$P$ : $person$

$M$ : $mail$

$P$ : $person$

$M$ : $mail$

$P$ : $person$

$M$ : $mail$

$P$ : $person$

$M$ : $mail$

$P$ : $person$

$M$ : $mail$

$P$ : $person$

$M$ : $mail$

$P$ : $person$

$M$ : $mail$

$P$ : $person$

$M$ : $mail$

$P$ : $person$

$M$ : $mail$

$P$ : $person$

$M$ : $mail$
Nested plans and optional navigation

for $P$ in $R//\text{person}$
return
$\langle \text{person} \rangle$
$\{ P/\text{mail} \}$
$\langle /\text{person} \rangle$

$\langle \text{mail} \rangle$
$\langle /\text{mail} \rangle$

$\langle \text{person} \rangle$
$\langle /\text{person} \rangle$

$\langle \text{mail} \rangle$
$\langle /\text{mail} \rangle$

$\langle \text{person} \rangle$
$\langle /\text{person} \rangle$

$\langle \text{mail} \rangle$
$\langle /\text{mail} \rangle$

$\langle \text{person} \rangle$
$\langle /\text{person} \rangle$

$\langle \text{mail} \rangle$
$\langle /\text{mail} \rangle$

$\langle \text{person} \rangle$
$\langle /\text{person} \rangle$

$\langle \text{mail} \rangle$
$\langle /\text{mail} \rangle$

$\langle \text{person} \rangle$
$\langle /\text{person} \rangle$

$\langle \text{mail} \rangle$
$\langle /\text{mail} \rangle$

$\langle \text{person} \rangle$
$\langle /\text{person} \rangle$

$\langle \text{mail} \rangle$
$\langle /\text{mail} \rangle$

$\langle \text{person} \rangle$
$\langle /\text{person} \rangle$

$\langle \text{mail} \rangle$
$\langle /\text{mail} \rangle$

$\langle \text{person} \rangle$
$\langle /\text{person} \rangle$

$\langle \text{mail} \rangle$
$\langle /\text{mail} \rangle$

$\langle \text{person} \rangle$
$\langle /\text{person} \rangle$

$\langle \text{mail} \rangle$
$\langle /\text{mail} \rangle$

$\langle \text{person} \rangle$
$\langle /\text{person} \rangle$

$\langle \text{mail} \rangle$
$\langle /\text{mail} \rangle$

$\langle \text{person} \rangle$
$\langle /\text{person} \rangle$

$\langle \text{mail} \rangle$
$\langle /\text{mail} \rangle$

$\langle \text{person} \rangle$
$\langle /\text{person} \rangle$

$\langle \text{mail} \rangle$
$\langle /\text{mail} \rangle$

$\langle \text{person} \rangle$
$\langle /\text{person} \rangle$

$\langle \text{mail} \rangle$
$\langle /\text{mail} \rangle$

$\langle \text{person} \rangle$
$\langle /\text{person} \rangle$

$\langle \text{mail} \rangle$
$\langle /\text{mail} \rangle$

$\langle \text{person} \rangle$
$\langle /\text{person} \rangle$

$\langle \text{mail} \rangle$
$\langle /\text{mail} \rangle$

$\langle \text{person} \rangle$
$\langle /\text{person} \rangle$

$\langle \text{mail} \rangle$
$\langle /\text{mail} \rangle$
Nested plans and optional navigation

for $P$ in $R$//person
return
<person>
  { $P$/mail }
</person>

$P$ \rightarrow $V$

$p1$ \[ p1 \] \[ m1 \]
$p2$ \[ p2 m2 \] \[ m2, m3 \]
$p3$ \[ p3 \] \[ ]
## Selection Predicates

<table>
<thead>
<tr>
<th>Predicate</th>
<th>Meaning</th>
<th>XQuery notation / fn or op [XQFO]</th>
</tr>
</thead>
<tbody>
<tr>
<td>=id</td>
<td>same node</td>
<td>is / op:is-same-node</td>
</tr>
<tr>
<td>=v</td>
<td>same value</td>
<td>eq / fn:compare, op:numeric-equal...</td>
</tr>
<tr>
<td>&lt;v</td>
<td>smaller value</td>
<td>lt / fn:compare, op:num-less-than...</td>
</tr>
<tr>
<td>&lt;&lt;</td>
<td>node before</td>
<td>&lt;&lt; / op:node-before</td>
</tr>
<tr>
<td>=</td>
<td>list equality</td>
<td>eq / fn:deep-equal</td>
</tr>
<tr>
<td>=∃</td>
<td>existential list equality</td>
<td>= / fns. backing eq, tuple equality</td>
</tr>
</tbody>
</table>
Selection Plan (1)

\[\sigma_{\theta}(p)\]

for $P$ in $R$ //person,
$N$ in $P$ /name
where
$P$/email = "m@acm.org"
return \{ $N \}
Selection Plan (2)

for $P$ in $R$ //person,
  $N$ in $P$//name
where
$P$/mail = "m@acm.org"
return { $N$ }

```
<table>
<thead>
<tr>
<th>$P$</th>
<th>$N$</th>
<th>$G$</th>
<th>$M$</th>
<th>$M2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>n3</td>
<td>[</td>
<td>p1</td>
<td>m1</td>
</tr>
<tr>
<td>p2</td>
<td>n4</td>
<td>[</td>
<td>p2</td>
<td>m2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>p2</td>
<td>m3</td>
</tr>
</tbody>
</table>
```

$P2$
crList $T_1$

```
<table>
<thead>
<tr>
<th>$P$</th>
<th>$N$</th>
<th>$G$</th>
<th>$M$</th>
<th>$M2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>n3</td>
<td>[</td>
<td>p1</td>
<td>m1</td>
</tr>
<tr>
<td>p2</td>
<td>n4</td>
<td>[</td>
<td>p2</td>
<td>m2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>p2</td>
<td>m3</td>
</tr>
</tbody>
</table>
```

crList

```
<table>
<thead>
<tr>
<th>$P$</th>
<th>$N$</th>
<th>$G$</th>
<th>$M$</th>
<th>$M2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>n3</td>
<td>[</td>
<td>p1</td>
<td>m1</td>
</tr>
<tr>
<td>p2</td>
<td>n4</td>
<td>[</td>
<td>p2</td>
<td>m2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>p2</td>
<td>m3</td>
</tr>
</tbody>
</table>
```

$P2$ → $M2$

```
<table>
<thead>
<tr>
<th>$P$</th>
<th>$N$</th>
<th>$G$</th>
<th>$M$</th>
<th>$M2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>n3</td>
<td>[</td>
<td>p1</td>
<td>m1</td>
</tr>
<tr>
<td>p2</td>
<td>n4</td>
<td>[</td>
<td>p2</td>
<td>m2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>p2</td>
<td>m3</td>
</tr>
</tbody>
</table>
```

$M2$
**Selection Plan (2)**

for $P$ in $R$ // person, $N$ in $P/name$

where $P/mail = "m@acm.org"

return { $N }
Selection plan (3)

$\sigma_{\theta}(p)$

for $P$ in $R$::person,
$N$ in $P$::name
where
$P$::mail = "m@acm.org"
return \{ $N$ \}

**Diagram:**

- $T_N$ as the root node
  - $P$::person
  - $M$::mail
  - $N$::name
- $T_C$ as a child of $T_N$
  - $P$::person
  - $M$::mail
  - $N$::name
- $crList_{T_C}$
  - $P$,$N$,$G$
  - $\sigma_{M = v\ "m@acm.org"}$

**Table:**

<table>
<thead>
<tr>
<th>$P$</th>
<th>$M$</th>
<th>$N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>m1</td>
<td>n3</td>
</tr>
<tr>
<td>p2</td>
<td>m2</td>
<td>n4</td>
</tr>
<tr>
<td>p2</td>
<td>m3</td>
<td>n4</td>
</tr>
</tbody>
</table>
Joins

Value-based join:

for $P$ in $R//person$, $J$ in $R//projects$

where $P/inproject = J/name$

return \{ $P$, $J$ \}
Joins and nesting

for $J$ in $R$//project
return

\[
\langle\text{page}\rangle \{ \{J\},
\{ \text{for } P \text{ in } R/\text{person}
\text{where } P/\text{inproject}=J/\text{name}
\text{return } \{ P \} \} \}
\]

\[
\langle/\text{page}\rangle
\]
Joins and nesting

Group people by the project
1) (N) from the notebook cust-->person, order-->project
doing many apply plans

2) Same with outerjoins and group the project (which was the outer)

3) now nest outer join, show in the example and then definition
Joins

ID-based join may be used to decompose tree pattern navigation into linear navigations:

![Diagram showing tree pattern navigation with joins]
for $p$ in //person
return <person>
  { for $n$ in $p$//name
    return { $N,$P/mail } }
</person>

Both deliver some unnecessary bindings

Both delivers some unnecessary bindings
Structural Joins and Optional Navigation:

Capturing navigation of:

```plaintext
def for $p$ in //person
    return <person>
    { for $n$ in $p$//name
        return { $N$, $P$//mail } }
</person>
```

Structural joins capture data-driven nesting

Algebra needs query-driven nesting
Structural Joins and Optional Navigation

Capturing navigation of:

for $p$ in //person
 return <person>
   { for $n$ in $p$//name
     return { $N,$P/mail } } }</person>

Eliminating unnecessary bindings:

$T

$P:person

mail $N:name

$M:mail

$N:person

$P:person

$N:person

$M:person

$T

$P

$N

$M
Structural Joins and Optional Navigation

Capturing navigation of:

```xml
for $p in //person
  return <person>
    { for $n in $p//name
        return { $N, $P/mail } } }
</person>
```

Eliminating unnecessary bindings:

```
groupBy [ $N ] [ ] → $G2

groupBy [ $P ] [ ] → $G1
```

```
groupBy $P,$M
```

```
nav $R,T_P

nav $R,T_N

nav $R,T_M
```

```
T *

$P:person

mail $N:name
```

```
T_P *

$P:person
```

```
T_N *

$N:person
```

```
T_M *

$M:mail
```
Structural Joins and Optional Navigation: Equivalent Expression

for $p$ in //person
return <person>
  { for $n$ in $p$//name
    return { $N,
      $P/mail } } }
</person>
Structural Joins vs. Simple Joins With Navigation

\[ \text{groupBy}_{[P]}[\cdot] \]

Remark:

- groupBy re-builds original nesting...
- Nest structural joins
- structural join +
- group by the left hand side variables
- Such grouping often comes cheap

\[ p1(..., P, ...) \quad p2(..., N, ...) \]

equivalent to:

\[ \text{groupBy} \quad \text{app} \quad \text{nav} \quad \text{crList} \]

\[ N \in D \]

\[ \text{eq} \quad \text{equiv}

\[ P1 \rightarrow D \]

\[ \text{app} \quad \text{nav} \quad \text{crList} \]

\[ T_C \quad N \]

\[ T_N \quad * \quad N: \text{name} \]
Nest Structural Joins

$P$ | $N$
---|---
p1 | [n3]
p2 | [n4]
p3 | [n5]

Remark:
groupBy re-builds original nesting...

Nest structural joins structural join + group by the left hand side variables

Such grouping often comes cheap
Related works
Existing tuple-based algebras

Tsimmis and YAT algebras for semistructured data [Cluet et al. 1998]
  • Introduced navigation and construction patterns

SAL from U. Tel Aviv [Beeri et al. 1999]
  • Close relative of OQL

TAX [Jagadish et al. 2001], Generalized Tree Patterns [Chen et al. 2003],
Tree Logical Classes [Paparizos et al. 2004] from Michigan
  • Navigation extracts bindings to ‘hidden’ tuples (packaged as trees)
  • Grouping tracking navigation
  • Also operators for updates (not covered here)

Enosys algebra [Papakonstantinou et al. 2003]
  • Collect bindings (nav and join), do nested plans, create XML
  • Also present in NEXT system [Deutsch et al. 2004]

Xstasy from U. Pisa [Sartiani et al. 2002]

Context-based algebra [Viglas et al. 2002]

Rainbow algebra from Worcester Polytechnic Inst. [Rundensteiner et al.
  2002]
References

S. Abiteboul and N. Bidoit. "Non First Normal Form Relations: An Algebra Allowing Data Restructuring", JCSS 1986


References (2)


References (3)


References (4)


C. Sartiani and A. Albano. "Yet Another Query Algebra for XML Data", IDEAS 2002

References (5)

XQuery 1.0 and XPath 2.0 Data Model
www.w3.org/TR/xpath-datamodel

XQuery 1.0 and XPath 2.0 Functions and Operators
www.w3.org/TR/xpath-functions

XQuery 1.0 Formal Semantics
www.w3.org/TR/2005/WD-query-semantics
Thank you