

# Full Untyped XQuery Canonization

Nicolas Travers<sup>1</sup> Tuyêt Trâm Dang Ngoc<sup>2</sup> Tianxiao Liu<sup>2</sup>

(1) PRISM Laboratory-University of Versailles, France

(2) ETIS Laboratory - University of Cergy-Pontoise, France

16 June 2007

Webetrends - APWeb/WAIM workshop 2007

# Plan

- 1 Context
- 2 Existing works
- 3 Canonization rules
- 4 Conclusion

## 1 Context

- XQuery

## 2 Existing works

## 3 Canonization rules

## 4 Conclusion

# XQuery : A rich syntax

```

declare function local:f($doc as xs:string) as element()
{
  for $x in (doc("rev.xml")/review|doc("$doc")/catalog)
    [ . contains("Robin Hobb")]/book/[//price > 15]
  where
    some $y in $x/comments
      satisfies contains ($y, "Excellent")
  order by $x/@isbn
  return
    <book>
      {$x/@isbn}
      <price>{$x//price/text()}</price>
    {
      if (count($x/title) > 2)
        then
        {
          for $z in doc("books.xml")/book
            where $z/@isbn = $x/@isbn
            return <title>{($z/title)[3]}</title>
        }
        else<title/>
    }
  </book>
}

```

- XPath;
- Constraints;
- Filters;
- Quantifiers;
- Document construction;
- Nesting;
- Aggregates;
- Conditional operators;
- Set operators;
- Sorts;
- Sequences;
- Functions;

# Equivalent forms

## Equivalent forms (W3C specifications) :

```
for $i in doc("cat.xml")/catalog/book
where $i//author = "Robin Hobb"
    and $i/title = "King's spy"
return
<price>
    {$i/price/text()}
</price>
```

```
for $i in doc("cat.xml")/catalog/book[./title = "King's spy"]
where $i//author = "Robin Hobb"
return
<price>
    {$i/price/text()}
</price>
```

# Equivalent forms

Equivalent forms (W3C specifications) :

```
for $i in doc("cat.xml")/catalog/book
where $i//author = "Robin Hobb"
    and $i/title = "King's spy"
return
<price>
    {$i/price/text()}
</price>
```

```
for $i in doc("cat.xml")/catalog/book[./title = "King's spy"]
where $i//author = "Robin Hobb"
return
<price>
    {$i/price/text()}
</price>
```

Need to define a unique form for all XQuery specifications.

## 1 Context

## 2 Existing works

- XPath
- NEXT
- Galax
- Canonization rules

## 3 Canonization rules

## 4 Conclusion

# XPath axes canonization [Olteanu et al. 2002]

XPath canonization rules :

- parent::n ;
- ancestor::n ;
- ancestor-or-self::n ;
- descendant-or-self::n ;

XQuery query	Canonical XQuery query

# XPath axes canonization [Olteanu et al. 2002]

XPath canonization rules :

- **parent::n** ;
- **ancestor::n** ;
- **ancestor-or-self::n** ;
- **descendant-or-self::n** ;

XQuery query	Canonical XQuery query
/catalog//title/ <b>parent::book</b>	/catalog//book[exists(./title)]

# XPath axes canonization [Olteanu et al. 2002]

XPath canonization rules :

- `parent::n` ;
- `ancestor::n` ;
- `ancestor-or-self::n` ;
- `descendant-or-self::n` ;

XQuery query	Canonical XQuery query
<code>/catalog//title/ancestor::book</code>	<code>/catalog//book[exists(.//title)]</code>

# XPath axes canonization [Olteanu et al. 2002]

XPath canonization rules :

- parent::n ;
- ancestor::n ;
- **ancestor-or-self::n** ;
- descendant-or-self::n ;

XQuery query	Canonical XQuery query
/catalog//title/ <b>ancestor-or-self::book</b>	/catalog//(/book[exists(.//title)]   /book//title)

# XPath axes canonization [Olteanu et al. 2002]

XPath canonization rules :

- parent::n ;
- ancestor::n ;
- ancestor-or-self::n ;
- descendant-or-self::n ;

XQuery query	Canonical XQuery query
/catalog/book/ <b>descendant-or-self::title</b>	/catalog/book/(.   /title)

# NEXT queries [Deustch et al. 2004]

- Transformations for XQuery ;
- For **strong nested** oriented queries ;
- New types of clause : "groupby" :

```
for $a in distinct-values($doc//book[title]/author)
return
  <bibentry>
    {
      $a,
      for $b in $doc//book,
        $a1 in $b/author,
        $t in $b/title
      where $a1 eq $a
      groupby [$b], [$t]
      return $t
    }
  </bibentry>
```

# The Galaxy experience [Fernández et al. 2003]

- Navigational based XQuery processing system ;
- Fully support by rewriting XQuery expressions ;
- Series of nested loops for normalization ;

```
snap {  
    element results {  
        for $b in  
            fs:distinct-docorder(  
                for $fs:dot in  
                    fs:distinct-docorder(for $fs:dot in $bib return child::bib)  
                    return child::book  
            )  
        return  
            element result {  
                fs:distinct-docorder(let $fs:dot := $b return child::title),  
                fs:distinct-docorder(let $fs:dot := $b return child::author)  
            }  
    }  
}
```

# Canonization rules [Chen 2004]

Existing rules :

- Filters ;
- Nesting ;
- Aggregates ;
- Quantifiers.

XQuery query	Canonical XQuery query

# Canonization rules [Chen 2004]

Existing rules :

- Filters ;
- Nesting ;
- Aggregates ;
- Quantifiers.

XQuery query	Canonical XQuery query
<pre>for \$i in doc("cat.xml")/catalog/book     [@isbn="12351234"] return {\$i}</pre>	<pre>for \$j in doc("cat.xml")/catalog/book where \$j/@isbn = "12351234" return {\$j}</pre>

# Canonization rules [Chen 2004]

Existing rules :

- Filters ;
- Nesting ;
- Aggregates ;
- Quantifiers.

XQuery query	Canonical XQuery query
<pre>for \$i in doc("cat.xml")/catalog/book     [@isbn="12351234"]/title return {\$i}</pre>	<pre>for \$j in doc("cat.xml")/catalog/book for \$i in \$j/title where \$j/@isbn = "12351234" return {\$i}</pre>

# Canonization rules [Chen 2004]

Existing rules :

- Filters ;
- Nesting ;
- Aggregates ;
- Quantifiers.

XQuery query	Canonical XQuery query
<pre>for \$i in doc("cat.xml")/catalog/book return     &lt;book&gt;         {for \$j in \$i/title return {\$j}}     &lt;/book&gt;</pre>	<pre>for \$i in doc("cat.xml")/catalog/book let \$I :=     (for \$j in \$i/title return {\$j}) return &lt;book&gt;{\$I}&lt;/book&gt;</pre>

# Canonization rules [Chen 2004]

Existing rules :

- Filters ;
- Nesting ;
- Aggregates ;
- Quantifiers.

XQuery query	Canonical XQuery query
<pre>for \$i in collection("catalog")/catalog/book return &lt;count&gt;{count(\$i/author)}&lt;/count&gt;</pre>	<pre>for \$i in collection("catalog")/catalog/book let \$l := count(\$i/author) return &lt;count&gt;{\$l}&lt;/count&gt;</pre>

# Canonization rules [Chen 2004]

Existing rules :

- Filters ;
- Nesting ;
- Aggregates ;
- Quantifiers.

XQuery query	Canonical XQuery query
<pre>for \$i in doc("cat.xml")/catalog/book where some \$s in \$i/price       satisfies \$s &gt; 15 return {\$i}</pre>	<pre>for \$i in doc("cat.xml")/catalog/book let \$l :=   (for \$s in \$i/price    where \$s &gt; 15    return {\$s}) where count(\$l) &gt; 0 return {\$i}</pre>

# Canonization rules [Chen 2004]

Existing rules :

- Filters ;
- Nesting ;
- Aggregates ;
- Quantifiers.

XQuery query	Canonical XQuery query
<pre>for \$i in doc("cat.xml")/catalog/book where every \$s in \$i/price       satisfies \$s &gt; 15 return {\$i}</pre>	<pre>for \$i in doc("cat.xml")/catalog/book let \$l :=   (for \$s in \$i/price    where \$s &lt;= 15    return {\$s}) where count(\$l) = 0 return {\$i}</pre>

## 1 Context

## 2 Existing works

## 3 Canonization rules

- New canonization rules
- Example

## 4 Conclusion

New canonization rules

# Untyped XQuery queries

Canonical XQuery [Chen 04] :

- XPath;
- Constraints;
- Filters;
- Quantifiers;
- Document construction;
- Nesting;
- Aggregates.

Need rules for :

- Sorts;
- Set operators;
- Conditionnal operators;
- Sequences;
- Functions.

# Untyped XQuery queries

Canonical XQuery [Chen 04] :

- XPath;
- Constraints;
- Filters;
- Quantifiers;
- Document construction;
- Nesting;
- Aggregates.

Need rules for :

- Sorts;
- Set operators;
- Conditionnal operators;
- Sequences;
- Functions.

## New canonization rules

# Canonization rules

New canonization rules :

- Sorts;
- Set operators (intersect, union, except);
- Conditional operators;
- Sequences;
- Functions;

XQuery query	Canonical XQuery query

## New canonization rules

# Canonization rules

New canonization rules :

- Sorts;
- Set operators (intersect, union, except);
- Conditional operators;
- Sequences;
- Functions;

XQuery query	Canonical XQuery query
<pre>for \$i in /catalog/book order by \$i/title return \$i/title</pre>	<pre>for \$i in /catalog/book let \$j := orderby (\$i, \$i/title) for \$k in \$j return \$k/title</pre>

## New canonization rules

# Canonization rules

New canonization rules :

- Sorts;
- Set operators (intersect, union, except);
- Conditional operators;
- Sequences;
- Functions;

XQuery query	Canonical XQuery query
<pre>for \$i in (/catalog   /review)/book return \$i/title</pre>	<pre>let \$i3 :=      for \$i<sub>1</sub> in /catalog     for \$i<sub>2</sub> in /review     return (\$i<sub>1</sub>   \$i<sub>2</sub>) for \$i in \$i<sub>3</sub>/book return \$i/title</pre>

## New canonization rules

# Canonization rules

New canonization rules :

- Sorts;
- Set operators (intersect, union, except);
- **Conditional operators;**
- Sequences;
- Functions;

XQuery query	Canonical XQuery query
<pre>for \$i in /catalog/book return   {if contains (\$i/author, "Hobb")   then ( for \$j in \$i//title return \$j )   else ( \$i/author )}</pre>	<pre>for \$i in /catalog/book let \$l := for \$j in \$i//title return \$j return   {if contains (\$i/author, "Hobb")   then ( \$l )   else ( \$i/author )}</pre>

## New canonization rules

# Canonization rules

New canonization rules :

- Sorts;
- Set operators (intersect, union, except);
- Conditional operators;
- Sequences;
- Functions;

XQuery query	Canonical XQuery query
<pre>for \$i in (/catalog/book)[2] return \$i/title</pre>	<pre>let \$i1 := for \$x in /catalog/book            return \$x for \$i in \$i1 where \$i/position() == 2 return \$i/title</pre>

# Canonization rules

New canonization rules :

- Sorts;
- Set operators (intersect, union, except);
- Conditional operators;
- Sequences;
- Functions;

XQuery query	Canonical XQuery query
<pre> declare function local:section   (\$i as element() ) as element ()* {   for \$j in \$i/book   return     &lt;book&gt;       {\$j/title}       {for \$s in \$i/section/title         return &lt;section&gt;{\$s/text()}&lt;/section&gt;}     &lt;/book&gt; } for \$f in doc("catalog.xml")/catalog return local:section(\$f) </pre>	<pre> declare function local:section   (\$i as element() ) as element ()* {   for \$j in \$i/book   let \$l := (for \$s in \$i/section/title             return &lt;section&gt;               {\$s/text()})   &lt;/section&gt;)   return     &lt;book&gt; {\$j/title} {\$l} &lt;/book&gt; } for \$f in doc("catalog.xml")/catalog return local:section(\$f) </pre>

## Example

## Illustrating example

```
for $x in (doc("rev.xml")/review| doc("$doc")/catalog)
  [. contains("Robin Hobb")]
  /book[./price > 15]
```

where

```
some $y in $x/comments
  satisfies contains ($y, "Excellent")
```

order by \$x/@isbn

return

```
<book>
  {$x/@isbn}
  <price>{$x//price/text()}</price>
  {
    if (count($x/title) > 2)
      then
    {
      for $z in doc("books.xml")/book
        where $z/@isbn = $x/@isbn
        return <title>{($z/title)[3]}</title>
    }
    else <title/>
  }
</book>
```

- 1 Set operator (\$I1) ;
- 2 Bind filtered xpatheres (\$f3) ;
- 3 Transform filters (constraints) ;
- 4 Transform quantifiers (\$I2) ;
- 5 Transform sorts (\$I3 & \$f4) ;
- 6 Prepare aggregate (\$I4) ;
- 7 Prepare nested queries (\$I5) ;
- 8 Transform sequences (\$I6).

## Example

## Illustrating example

```
let $I1 := (for $f1 in doc("rev.xml")/review
            for $f2 in doc("$doc")/catalog
            return ($f1 | $f2))
for $x in $I1[. contains("Robin Hobb")]/book[.//price > 15]
where
    some $y in $x/comments
        satisfies contains ($y, "Excellent")
order by $x/@isbn
return
<book>
  {$x/@isbn}
  <price>{$x//price/text()}</price>
{
  if (count($x/title) > 2)
  then
  {
    for $z in doc("books.xml")/book
    where $z/@isbn = $x/@isbn
    return <title>{($z/title)[3]}</title>
  }
  else <title/>
}
</book>
```

- 1 Set operator (\$I1) ;
- 2 Bind filtered xpatheres (\$f3) ;
- 3 Transform filters (constraints) ;
- 4 Transform quantifiers (\$I2) ;
- 5 Transform sorts (\$I3 & \$f4) ;
- 6 Prepare aggregate (\$I4) ;
- 7 Prepare nested queries (\$I5) ;
- 8 Transform sequences (\$I6).

## Example

## Illustrating example

```
let $I1 := (for $f1 in doc("rev.xml")/review
            for $f2 in doc("$doc")/catalog
            return ($f1 | $f2))
for $f3 in $I1[. contains("Robin Hobb")], $x in $f3/book[./price > 15]
where
    some $y in $x/comments
        satisfies contains ($y, "Excellent")
order by $x/@isbn
return
<book>
  {$x/@isbn}
  <price>{$x//price/text()}</price>
{
  if (count($x/title) > 2)
  then
  {
    for $z in doc("books.xml")/book
    where $z/@isbn = $x/@isbn
    return <title>{($z/title)[3]}</title>
  }
  else <title/>
}
</book>
```

- 1 Set operator (\$I1) ;
- 2 Bind filtered xpahes (\$f3) ;
- 3 Transform filters (constraints) ;
- 4 Transform quantifiers (\$I2) ;
- 5 Transform sorts (\$I3 & \$f4) ;
- 6 Prepare aggregate (\$I4) ;
- 7 Prepare nested queries (\$I5) ;
- 8 Transform sequences (\$I6).

## Example

## Illustrating example

```
let $I1 := (for $f1 in doc("rev.xml")/review
            for $f2 in doc("$doc")/catalog
            return ($f1 | $f2))
for $f3 in $I1, $x in $f3/book
where contains($f3, "Robin Hobb") and $x//price > 15 and
      some $y in $x/comments
          satisfies contains ($y, "Excellent")
order by $x/@isbn
return
<book>
  {$x/@isbn}
  <price>{$x//price/text()}</price>
{
  if (count($x/title) > 2)
  then
  {
    for $z in doc("books.xml")/book
    where $z/@isbn = $x/@isbn
    return <title>{($z/title)[3]}</title>
  }
  else <title/>
}
</book>
```

- 1 Set operator (\$I1) ;
- 2 Bind filtered xpatheres (\$f3) ;
- 3 Transform filters (constraints) ;
- 4 Transform quantifiers (\$I2) ;
- 5 Transform sorts (\$I3 & \$f4) ;
- 6 Prepare aggregate (\$I4) ;
- 7 Prepare nested queries (\$I5) ;
- 8 Transform sequences (\$I6).

## Example

## Illustrating example

```
let $I1 := (for $f1 in doc("rev.xml")/review
            for $f2 in doc("$doc")/catalog
            return ($f1 | $f2))
let $I2 := (for $y in $x/comments
            where contains ($y, "Excellent")
            return $y)
for $f3 in $I1, $x in $f3/book
where contains($f3, "Robin Hobb") and $x//price > 15 and count ($I2) > 0
order by $x/@isbn
return
<book>
  {$x/@isbn}
  <price>{$x//price/text()}</price>
  {
    if (count($x/title) > 2)
    then
    {
      for $z in doc("books.xml")/book
      where $z/@isbn = $x/@isbn
      return <title>{($z/title)[3]}</title>
    }
    else <title/>
  }
</book>
```

- 1 Set operator (\$I1) ;
- 2 Bind filtered xpatheres (\$f3) ;
- 3 Transform filters (constraints) ;
- 4 Transform quantifiers (\$I2) ;
- 5 Transform sorts (\$I3 & \$f4) ;
- 6 Prepare aggregate (\$I4) ;
- 7 Prepare nested queries (\$I5) ;
- 8 Transform sequences (\$I6).

## Example

## Illustrating example

```
let $I1 := (for $f1 in doc("rev.xml")/review
            for $f2 in doc("$doc")/catalog
            return ($f1 | $f2))
let $I2 := (for $y in $f4/comments
            where contains ($y, "Excellent")
            return $y)
for $f3 in $I1, $f4 in $f3/book
where contains($f3, "Robin Hobb") and $f4//price > 15 and count ($I2) > 0
let $I3 := orderby ($f4, $f4/@isbn)
for $x in $I3
return
<book>
  {$x/@isbn}
  <price>{$x//price/text()}</price>
  {
    if (count($x/title) > 2)
    then
    {
      for $z in doc("books.xml")/book
      where $z/@isbn = $x/@isbn
      return <title>{($z/title)[3]}</title>
    }
    else <title/>
  }
</book>
```

- 1 Set operator (\$I1) ;
- 2 Bind filtered xpatheres (\$f3) ;
- 3 Transform filters (constraints) ;
- 4 Transform quantifiers (\$I2) ;
- 5 **Transform sorts (\$I3 & \$f4) ;**
- 6 Prepare aggregate (\$I4) ;
- 7 Prepare nested queries (\$I5) ;
- 8 Transform sequences (\$I6).

## Example

## Illustrating example

```
let $I1 := (for $f1 in doc("rev.xml")/review
            for $f2 in doc("$doc")/catalog
            return ($f1 | $f2))
let $I2 := (for $y in $f4/comments
            where contains ($y, "Excellent")
            return $y)
for $f3 in $I1, $f4 in $f3/book
where contains($f3, "Robin Hobb") and $f4//price > 15 and count ($I2) > 0
let $I3 := orderby ($f4, $f4/@isbn)
for $x in $I3
let $I4 := count ($x/title)
return
<book>
  {$x/@isbn}
  <price>{$x//price/text()}</price>
  {
    if ($I4 > 2)
      then
      {
        for $z in doc("books.xml")/book
        where $z/@isbn = $x/@isbn
        return <title>{($z/title)[3]}</title>
      }
    else <title/>
  }
</book>
```

- 1 Set operator (\$I1) ;
- 2 Bind filtered xpatheres (\$f3) ;
- 3 Transform filters (constraints) ;
- 4 Transform quantifiers (\$I2) ;
- 5 Transform sorts (\$I3 & \$f4) ;
- 6 **Prepare aggregate (\$I4) ;**
- 7 Prepare nested queries (\$I5) ;
- 8 Transform sequences (\$I6).

## Example

## Illustrating example

```
let $I1 := (for $f1 in doc("rev.xml")/review
            for $f2 in doc("$doc")/catalog
            return ($f1 | $f2))
let $I2 := (for $y in $f4/comments
            where contains ($y, "Excellent")
            return $y)
for $f3 in $I1, $f4 in $f3/book
where contains($f3, "Robin Hobb") and $f4//price > 15 and count ($I2) > 0
let $I3 := orderby ($f4, $f4/@isbn)
for $x in $I3
let $I4 := count ($x/title)
let $I5 := (for $z in doc("books.xml")/book
            where $z/@isbn = $x/@isbn
            return <title>{($z/title)[3]}</title>)
return
<book>
  {$x/@isbn}
  <price>{$x//price/text()}</price>
  {
    if ($I4 > 2)
      then {$I5}
    else <title/>
  }
</book>
```

- 1 Set operator (\$I1) ;
- 2 Bind filtered xpatheres (\$f3) ;
- 3 Transform filters (constraints) ;
- 4 Transform quantifiers (\$I2) ;
- 5 Transform sorts (\$I3 & \$f4) ;
- 6 Prepare aggregate (\$I4) ;
- 7 Prepare nested queries (\$I5) ;
- 8 Transform sequences (\$I6).

# Illustrating example

```

let $I1 := (for $f1 in doc("rev.xml")/review
            for $f2 in doc("$doc")/catalog
            return ($f1 | $f2))
let $I2 := (for $y in $f4/comments
            where contains ($y, "Excellent")
            return $y)
for $f3 in $I1, $f4 in $f3/book
where contains($f3, "Robin Hobb") and $f4//price > 15 and count ($I2) > 0
let $I3 := orderby ($f4, $f4/@isbn)
for $x in $I3
let $I4 := count ($x/title)
let $I5 := (let $I6 := (for $z in doc("books.xml")/book
                        where $z/@isbn = $x/@isbn
                        return $z)
            for $f5 in $I6/title
            where $f5/position () = 3
            return <title>{$f5}</title>)
return
<book>
  {$x/@isbn}
  <price>{$x//price/text()}</price>
  {
    if ($I4 > 2)
      then {$I5}
      else <title/>
  }
</book>

```

- 1 Set operator (\$I1) ;
- 2 Bind filtered xpatheres (\$f3) ;
- 3 Transform filters (constraints) ;
- 4 Transform quantifiers (\$I2) ;
- 5 Transform sorts (\$I3 & \$f4) ;
- 6 Prepare aggregate (\$I4) ;
- 7 Prepare nested queries (\$I5) ;
- 8 Transform sequences (\$I6).

## 1 Context

## 2 Existing works

## 3 Canonization rules

## 4 Conclusion

# Conclusion

Thanks to this canonization rules :

- A full untyped XQuery queries are bound to a unique form ;
- Simplify treatments identification :
  - Operations orders ;
  - Modeling XQuery (TGV [Travers et al. 2007]) ;
  - Distributing sub-queries (the XLive mediator).
- Validation with W3C use-cases [8/9] (except STRONG) ;
- Future works : typing ;

Chen 2004 "From Tree Patterns to Generalized Tree Patterns : On Efficient Evaluation of XQuery", University of British Columbia, MSc Thesis, 2004

Deutsch et al. 2004 "The NEXT Framework for Logical XQuery Optimization", VLDB, 2004

Fernández et al. 2003 "Implementing XQuery 1.0: The Galax Experience", VLDB, 2003

Olteanu et al. 2002 "XPath : Looking Forward", EDBT Workshop on XML Data Management (XMLDM), 2002

Travers et al. 2006 "TGV: an Efficient Model for XQuery Evaluation within an Interoperable System", IBIS issue 3, 2006

XLive "<http://www.prism.uvsq.fr/index.php?id=xlive>",