

Full Untyped XQuery Canonization

Nicolas Travers¹ Tuyêt Trâm Dang Ngoc² Tianxiao Liu²

(1) PRiSM Laboratory-University of Versailles, France

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

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

```

- 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 $i/title = "King's spy"
return
  <prix>
    {$i/price/text()}
  </prix>

```

```

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

```

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
  <prix>
    { $i/price/text() }
  </prix>
```

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

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 |
|---|--|
| <code>/catalog//title/parent::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 |
|---|---|
| <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 |
|---|---|
| <code>/catalog//title/ancestor-or-self::book</code> | <code>/catalog/(/book[exists(./title)] /book//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/book/ descendant-or-self ::title | /catalog/book/(. /title) |

NEXT queries [Deustch et al. 2004]

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

NEXT queries [Deustch et al. 2004]

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

Too much specific.

The Galax experience [Fernández et al. 2003]

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

The Galax experience [Fernández et al. 2003]

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

Need more efficiency and identify dependencies.

Canonization rules [Chen 04]

Existing rules :

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

| XQuery query | Canonical XQuery query |
|--------------|------------------------|
| | |

Canonization rules [Chen 04]

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 04]

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 04]

Existing rules :

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

| XQuery query | Canonical XQuery query |
|---|--|
| <pre>for \$i in doc("cat.xml")/catalog/book return <livre> {for \$j in \$i/title return {\$j}} </livre></pre> | <pre>for \$i in doc("cat.xml")/catalog/book let \$l := (for \$j in \$i/title return {\$j}) return <livre>{\$l}</livre></pre> |

Canonization rules [Chen 04]

Existing rules :

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

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

Canonization rules [Chen 04]

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 > 15 return {\$i}</pre> | <pre>for \$i in doc("cat.xml")/catalog/book let \$I := (for \$s in \$i/price where \$s > 15 return {\$s}) where count(\$I) > 0 return {\$i}</pre> |

Canonization rules [Chen 04]

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 > 15 return {\$i}</pre> | <pre>for \$i in doc("cat.xml")/catalog/book let \$I := (for \$s in \$i/price where \$s <= 15 return {\$s}) where count(\$I) = 0 return {\$i}</pre> |

- 1 Context
- 2 Existing works
- 3 Canonization rules
 - New canonization rules
 - Example
- 4 conclusion

Untyped XQuery queries

Canonical XQuery [Chen 04] :

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

Need rules for :

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

Untyped XQuery queries

Canonical XQuery [Chen 04] :

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

Need rules for :

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

Canonization rules

New canonization rules :

- Sorts;
- Set operators;
- Conditional operators;
- Sequences;
- Functions;

| XQuery query | Canonical XQuery query |
|--------------|------------------------|
| | |

Canonization rules

New canonization rules :

- Sorts;
- Set operators;
- 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> |

Canonization rules

New canonization rules :

- Sorts;
- **Set operators;**
- Conditional operators;
- Sequences;
- Functions;

| XQuery query | Canonical XQuery query |
|--|--|
| <pre>for \$i in (/catalog /review)/book return \$i/title</pre> | <pre>let \$i₃ := for \$i₁ in /catalog for \$i₂ in /review return (\$i₁ \$i₂) for \$i in \$i₃/book return \$i/title</pre> |

Canonization rules

New canonization rules :

- Sorts;
- Set operators;
- **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> |

Canonization rules

New canonization rules :

- Sorts;
- Set operators;
- 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;
- 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 <book> {\$j/title} {for \$s in \$i/section/title return <section>{\$s/text()}} </book> } 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 <section> {\$s/text()}) </section>) return <book> {\$j/title} {\$l} </book> } for \$f in doc(" catalog.xml")/catalog return local:section(\$f) </pre> |

Illustrating example

```

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

```

Example

Illustrating example

```

declare function local:f($doc as xs:string) as element()
{
  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 $x/@isbn
  return
    <livre>
      {$x/@isbn}
      <prix>{$x//price/text()}</prix>
      {
        if (count($x/title) > 2)
        then
          {
            for $z in doc(" books.xml")/book
            where $z/@isbn = $x/@isbn
            return <titre>{($z/title)[3]}</titre>
          }
        else <titre/>
      }
    </livre>
}

```

Example

Illustrating example

```

declare function local:f($doc as xs:string) as element()
{
  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 $x/@isbn
  return
    <livre>
      {$x/@isbn}
      <prix>{$x//price/text()}</prix>
      {
        if (count($x/title) > 2)
        then
          {
            for $z in doc(" books.xml")/book
            where $z/@isbn = $x/@isbn
            return <titre>{($z/title)[3]}</titre>
          }
        else <titre/>
      }
    }
</livre>
}

```

Illustrating example

```

declare function local:f($doc as xs:string) as element()
{
  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
    <livre>
      { $x/@isbn }
      <prix> { $x//price/text() } </prix>
      {
        if (count($x/title) > 2)
        then
          {
            for $z in doc(" books.xml")/book
            where $z/@isbn = $x/@isbn
            return <titre> { ($z/title)[3] } </titre>
          }
        else <titre/>
      }
    </livre>
}

```

Illustrating example

```

declare function local:f($doc as xs:string) as element()
{
  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
    <livre>
      { $x/@isbn }
      <prix> { $x//price/text() } </prix>
      {
        if (count($x/title) > 2)
        then
          {
            for $z in doc(" books.xml")/book
            where $z/@isbn = $x/@isbn
            return <titre> { ($z/title)[3] } </titre>
          }
        else <titre/>
      }
    </livre>
}

```

Illustrating example

```

declare function local:f($doc as xs:string) as element()
{
  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, $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
    <livre>
      { $x/@isbn }
      <prix> { $x//price/text() } </prix>
      {
        if (count($x/title) > 2)
        then
          {
            for $z in doc(" books.xml")/book
            where $z/@isbn = $x/@isbn
            return <titre> { ($z/title)[3] } </titre>
          }
        else <titre/>
      }
    </livre>
}

```

Example

Illustrating example

```

declare function local:f($doc as xs:string) as element()
{
  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, $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
    <livre>
      { $x/@isbn }
      <prix> { $x//price/text() } </prix>
      {
        if ($I4 > 2)
        then
          {
            for $z in doc(" books.xml")/book
            where $z/@isbn = $x/@isbn
            return <titre> { ($z/title)[3] } </titre>
          }
        else <titre/>
      }
    </livre>
}

```

Illustrating example

```

declare function local:f($doc as xs:string) as element()
{
  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, $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 = $x/@isbn
              return <titre>{($z/title)[3]}</titre>)
  return
    <livre>
      {$x/@isbn}
      <prix>{$x//price/text()}</prix>
      {
        if ($I4 > 2)
        then {$I5}
        else <titre/>
      }
    </livre>
}

```

Example

Illustrating example

```

declare function local:f($doc as xs:string) as element()
{
  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, $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 $f5/position () = 3
              return <titre>{$f5}</titre>)
  return
  <livre>
    {$x/@isbn}
    <prix>{$x//price/text()}</prix>
    {
      if ($I4 > 2)
      then {$I5}
      else <titre/>
    }
  </livre>

```

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

Conclusion

Thanks to this canonization rules :

- All untyped XQuery queries are bound to a unique form ;
- Simplify treatments identification :
 - Operations orders ;
 - Modeling XQuery (TGV [Travers et al. 2007]) ;
 - Distributing sub-queries (mediation context).