Расширение языка XQuery функциональными update-выражениями*

А.А. Болдаков, М.Н. Гринев
boldakov@ispras.ru maxim@grinev.net

Аннотация. Язык XQuery – это универсальный язык запросов к XML-данным, содержащий средства как для выборки, так и для трансформации XML-данных. В статье анализируется ограниченность средств трансформации XML-данных языка XQuery. Демонстрируется важный на практике класс запросов, для которых выражения на языке XQuery громоздки и сложны для эффективного вычисления.
В данной работе для этого класса запросов предлагается расширение языка XQuery функциональными update-выражениями. Такие выражения близки по синтаксису к выражениям языков модификации XML-данных, но вычисляются без побочных эффектов, что позволяет естественным образом интегрировать их в среду XQuery.
В статье демонстрируется выразительность расширенного языка и рассматриваются подходы к эффективной реализации предложенного расширения. В последнем разделе статьи обсуждается проблема интеграции языковых средств выборки и модификации XML-данных (проблема вложенных update-выражений). Мы показываем, каким образом идея функциональных update-выражений может быть применена для решения этой проблемы.

1. Введение

Язык XQuery [1], разработанный консорциумом W3C, – это язык запросов к XML-данным. Язык XQuery позволяют компактно и выразительно описывать выборку XML-данных, операции соединения, объединения, упорядочивания XML-данных, а также построение новых фрагментов данных на основе использования конструкторов. В настоящее время язык XQuery рассматривается как одно из основных универсальных средств обработки XML-данных, наряду с языком трансформации XML-данных XSLT[4].

Важной задачей обработки XML-данных является трансформация данных. В языке XQuery для описания трансформаций XML-данных предназначены конструкторы для всех основных видов узлов XML-документа (элементов, атрибутов, документов и т.п.). Однако анализ практических приложений демонстрирует неадекватность средств описания трансформаций языка XQuery для ряда задач.

* Работа поддержана грантами РФФИ 04-07-08003 и 05-07-90204.

Существует класс таких запросов к XML-данным, которых, с одной стороны, трудно выражаются средствами языка XQuery, а с другой стороны, являются сложными для эффективного вычисления. Запросы из этого класса характеризуются тем, что они вычисляют некоторое модифицированное значение данных.

В качестве простых примеров можно перечислить следующие трансформационные запросы:

- получить XML-данные, заменив внутри некоторое содержимое;
- получить XML-данные, удалив (вставив) внутри (во внутрь) некоторое содержимое;
- получить XML-данные на основе исходного и модифицированного состояний данных.

В приведенных примерах речь идет о выборке относительно сложной со структурной точки зрения порции данных с некоторой модификацией. Иными словами, интересующий нас класс запросов можно охарактеризовать необходимостью получения данных с относительно небольшой их модификацией.

Проблема работы с такими запросами на языке XQuery сводится к двум аспектам. С одной стороны, такие запросы сложны для выражения средствами XQuery. С другой стороны, XQuery-выражения, описывающие такие запросы, сложны для эффективного вычисления и, следовательно, не подходят для работы над большими объемами данных.

Рассмотрим оба этих аспекта более детально. Имеет смысл рассмотреть возникающие проблемы для двух случаев. В первом случае требуется вычислить запрос над данными, структура которых не известна. Такая ситуация часто возникает при работе с XML-документами, ориентированными на данные (*document-oriented XML data*) [10]. Во втором случае структура данных может быть известной и сложной, что характерно для XML-документов, ориентированных на данные (*data-oriented XML documents*) [10].

Рассмотрим проблемы, связанные с выразительностью запросов на языке XQuery, на следующем примере, характерным для задач управления контентом (*content management*). Предположим, что имеется база XML-данных, содержащая обзоры кинофильмов. Каждый обзор представляет собой текст, в котором некоторые слова или выражения обрамлены XML-тэгами с некоторыми атрибутами. Например, имя режиссера заключено в тэг `<director id="3">`. Такая встроенная разметка (*in-line markup*) часто используется для того, чтобы внести структуру фрагментов текстовых данных и приписать им семантику. В дальнейшем эта разметка может быть использована при генерации различных представлений данных для публикации.

Предположим, что требуется сгенерировать XHTML-представление обзора, в котором имена режиссеров должны стать гиперссылками, то есть таким образом, чтобы встречающиеся в тексте имена режиссера были заменены в HTML-представлении гиперссылками на страницы, содержащие информацию

об этом режиссере. Пусть информация о ссылках хранится отдельно, например, в другом документе базы XML-данных.

Если структура данных не известна, мы не можем заранее сказать, в каких именно узлах встречается элемент `director`. Тогда единственным методом решения задачи является использование рекурсивной функции, обходящей узлы документа и формирующей результат в зависимости от имени и типа текущего узла. Примерный вариант запроса с рекурсивной функцией обхода XML-дерева на языке XQuery будет выглядеть следующим образом.

```
declare function local:replace($n as xs:node) as xs:node
{
  typeswitch($n)
    case $d as element(director)
     return <a href="{doc('url')/…/url}">{$d/text()}</a>
    case $e as element()
     return element
            { fn:local-name($e) }
            { for $c in $e/(* | @* | text())
              return local:replace($c) }
    case $d as document-node()
     return document
            { for $c in $d/* return local:replace($c) }
    default
     return $n
};

for                          $r                          in
doc("db.xml")/db/movie[gender="fiction"]/review
return
    local:replace($r)
```

Запрос 1. Получение XHTML-представления обзоров кинофильмов в жанре `fiction` на языке XQuery при неизвестной структуре данных

В приведенном Запросе 1 рекурсивная функция `replace` обходит все возможные узлы документа и в зависимости от типа и имени узла возвращает результат. Необходимо отметить, что функция `replace` по сути заново конструирует весь элемент средствами конструкторов языка XQuery. Подобные запросы с рекурсивными функциями трудны для понимания и не отличаются компактностью и выразительностью записи.

Эффективное выполнение приведенного XQuery-запроса также порождает ряд проблем. Во-первых, для получения результата, необходимо обойти абсолютно все узлы элемента `review`. То есть для любого объема данных необходимо просканировать все эти данные целиком. Во-вторых, для получения результата необходимо заново сконструировать каждый узел элемента `review`. Операция конструирования XML-узла во многих XML-СУБД (например, в XML-СУБД Sedna[5]) является “дорогой” операцией, что затрудняет эффективное

вычисление запросов. Таким образом, такой подход не приемлем при работе с большими объемами данных.

Описанный пример является примером общей задачи “найти и заменить”. Такая задача, например, часто возникает в задачах управления контентом, когда приходится обрабатывать документ-ориентированные XML-данные с частично известной структурой, в которых активно используется линейная разметка текста.

Рассмотрим другой пример. Допустим, что база XML-данных содержит данные о заказах в XML-элементах `order`, где каждый заказ состоит из набора наименований `orderLine`. Каждое из наименований характеризуется названием `name`, количеством `quantity` и ценой `price` в валюте `cur`. Пример такого элемента показан на рис. 1.

```
<orders>
   <order id=”1”>
      <orderLine>
          <name>tire</name>
          <quantity>4</quantity>
          <price cur=”rubles”>1200</price>
       </orderLine>
      <orderLine>
          <name>oil</name>
          <quantity>1</quantity>
          <price cur=”rubles”>750</price>
       </orderLine>
      …
   </order>
   …
</orders>
```

Рис. 1. Пример элемента `orders`

В качестве результата запроса требуется выбрать все заказы с ценой, конвертированной в другую валюту по заданному курсу. Если структура данных известна, то требуемый запрос можно выразить на языке XQuery без использования рекурсивных функций следующим образом.

```
let $euro-
rate:=doc(“rates”)/rates/rate[@name=”euro”]/text()
return
 <orders>
 {
  for $o in doc(“orders.xml”)/orders/order
  return
    <order id=”$o/@id”>
       { for $l in $o/orderLine
```

```
        return
            <orderLine>
               {$ol/name,
                $ol/quantity}
                <price cur=”euro”>
                  {$ol/price/text()*$euro-rate}
                </price>
             </orderLine>
       }
    </order>
 }
</orders>
```

Запрос 2. Получение списка заказов с конвертированной ценой на языке XQuery

Необходимо обратить внимание на то, что, как и в случае с использованием рекурсивных функций, в данном примере XQuery-выражение явно конструирует всю структуру результата. Схема XML-данных может быть достаточно сложной, и воссоздание этой схемы при построении результата ведет, с одной стороны, к ошибкам и громоздкости XQuery-выражения и, с другой стороны, к проблеме эффективного выполнения таких запросов.

Приведенный на рис. 1 фрагмент XML-документа является примером XML-данных, ориентированных на хранение данных. Такие XML-данные хорошо структурированы, но структура их может быть сложной, с большим количеством уровней вложенности XML-узлов. Даже при известной схеме XML-данных выражение запросов из рассматриваемого класса средствами языка XQuery порождает указанные проблемы.

Мы показали, что существует важный с практической точки зрения класс запросов к XML-данным, который сложен для выражения на языке XQuery и вычисления. Такие запросы характерны как для документ-ориентированных XML-данных, так и для XML-структур, ориентированных на хранение данные. Выражение запросов из этого класса средствами языка XQuery, с одной стороны, приводит к громоздкому и не выразительному представлению запросов, а с другой стороны, такие запросы сложны для эффективного вычисления, что важно при работе с большими объемами данных.

Таким образом, для обозначенного класса запросов актуальны задачи компактного и выразительного представления запросов на языке XQuery и их эффективного вычисления.

Дальнейшее изложение материала в работе построено следующим образом. В разделе 2 мы введем расширение для языка XQuery – функциональные update-выражения. Будет продемонстрирована возможность компактного и выразительного представления запросов при помощи функциональных update-выражений на примерах, описанных во введении. Затем мы представим синтаксис и семантику функциональных update-выражений как расширения XQuery. Далее будут рассмотрены различные альтернативы реализации

функциональных update-выражений в прирожденных базах XML-данных, и будет показана возможность эффективного вычисления таких выражений над большими объемами XML-данных. Последний раздел работы посвящен проблеме тесной интеграции языковых средств выборки и модификации XML-данных. Будет предложен подход реализации такой интеграции на основе идеи функциональных update-выражений.

2. Функциональные update-выражения

Для преодоления указанных проблем мы предлагаем расширить язык XQuery *функциональными update-выражениями*. Функциональные update-выражения составляют язык модификации XML-данных с функциональной семантикой. Основная идея функциональных update-выражений состоит в том, что в них используется синтаксис, близкий к синтаксису традиционных update-выраженияй, но они выполняются без побочных эффектов. В отличие от обычных update-выражений, функциональные update-выражения не изменяют состояние данных, а возвращают копию данных с измененным состоянием.

Компактность и выразительность функциональных update-выражений достигается благодаря использованию синтаксиса, близкого к традиционным update-выражениям. Результатом вычисления функциональных update-выражений являются XML-данные, то есть такие выражения не расширяют модель данных языка XQuery и, следовательно, могут быть естественным образом включены в состав выражений языка XQuery.

Рассмотрим синтаксис и семантику функциональных update-выражений на примерах, описанных во введении. В первом примере требовалось вернуть XHTML-представления обзоров кинофильмов в жанре `fiction`, заменив все вхождения элемента `director` соответствующей гиперссылкой. Такой запрос, записанный при помощи функциональных update-выражений, выглядит следующим образом:

```
for $r in doc("db.xml")/db/movie[gender="fiction"]/review
transform replace $d in $r//director
with
  <a
href="{doc("b")//person[name=$d/@name]/homepage/text()}"
>
    {$d/text()}
  </a>
```

Запрос 3. Получение HTML-представления обзоров кинофильмов в жанре fiction с использованием функциональных update-выражений

Данное функциональное update-выражение следует интерпретировать следующим образом. Конструкция `for` связывает XQuery-переменную `$r` с последовательностью XML-узлов, возвращаемой XPath-выражением, которое следует за ключевым словом `in`. В функциональном update-выражении ветка `for`, помимо связывания переменной с последовательностью XML-узлов, определяет результат выражения, то есть *новые* узлы, которые будут являться

результатом вычисления всего выражения. В данном примере результатом вычисления выражения будет последовательность узлов `review`. Возвращаемые элементы `review` представляют собой копию узлов `review` оригинального документа с модификациями, описанными в конструкции `transform replace` функционального update-выражения. В данном примере запрос описывает трансформацию данных из класса “найти и заменить”. Переменная `$d` связывается с узлами, расположенными внутри возвращаемого элемента `review`. Каждое из таких связываний в возвращаемом элементе заменяется на новый элемент, описываемый XQuery-конструктором. Таким образом, в примере функциональное update-выражение возвращает последовательность узлов `review`, в которых все вхождения элемента `director` заменены элементом `a`. Выражение вычисляется без побочных эффектов, т.е. узлы `review` документа `db.xml` не изменяются.

Рассмотрим второй пример из предыдущей главы. Нам требуется выполнить запрос над XML-данными с известной схемой данных, ориентированными на хранение данных. Логика запроса заключается в выборке всех заказов с ценой, конвертированной в другую валюту. Ниже представлена запись такого запроса с использованием функциональных update-выражений.

```
let $euro-
rate:=doc(“rates”)/rates/rate[@name=”euro”]/text()
for $orders in doc(“orders.xml”)/orders
transform replace $p in $orders/order/orderLine/price
with
    <price>{$p/text()*$euro-rate}</price>
```

Запрос 4. Получение списка заказов с конвертированной ценой с использованием функциональных update-выражений

По синтаксису и семантике запрос аналогичен предыдущему запросу. Отличия заключаются в двух аспектах. Во-первых, вследствие известной схемы данных, все XPath-выражения прописаны без использования оси *descendant-or-self* [3]. Второе, более принципиальное отличие состоит в том, что запрос возвращает один элемент `orders`, в котором все вложенные элементы `order` изменены требуемым образом. Так происходит вследствие того, что XPath выражение *doc(“orders.xml”)/orders* возвращает только один элемент `orders`.

Как уже отмечалось выше, функциональное update-выражение может возвращать любые данные из модели данных языка XQuery. В предыдущих примерах результатом вычисления выражений были последовательности узлов элементов XML-документа. Необходимо подчеркнуть, что функциональное update-выражение может возвращать и узел документа. Предыдущий пример (Запрос 4), модифицированный соответствующим образом, представлен ниже (Запрос 5).


```
let $euro-
rate:=doc(“rates”)/rates/rate[@name=”euro”]/text()
for $orders in doc(“orders.xml”)
transform replace $p in
$orders/orders/order/orderLine/price
with
    <price>{$p/text()*$euro-rate}</price>
```

Запрос 5. Получение узла XML-документа `orders.xml` с конвертированной ценой с использованием функциональных update-выражений

Функциональные update-выражения естественным образом расширяют язык XQuery и не нарушают его замкнутости. В сложных XQuery-запросах, update-выражения с функциональной семантикой могут участвовать равноправно по отношению к выражениями языка XQuery. Благодаря этому принципиальной особенностью расширения языка XQuery функциональными update-выражениями является доступность в одном XQuery-выражении как исходного, так и модифицированного состояния XML-данных.

Последняя возможность может требоваться в целом ряде случаев . Например, такая потребность возникает при необходимости произвести некоторую аналитику над возможным изменением данных, то есть проанализировать изменения по отношению к текущему состоянию данных относительно некоторого модифицированного состояния.

В примере Запрос 6 выражение на расширенном языке конструирует сводные XML-данные над исходным и модифицированным состояниями данных. Несмотря на то, что этот запрос может быть относительно просто выражен средствами языка XQuery, в практических приложениях его запись в представленном виде может быть более приемлемой.

```
let $euro-rate:=document(“stock”)/rates/euro/text()
let $orders:=document(“orders”)/orders/order
let $new-orders:=for $o in
document(“orders”)/orders/order
                  transform replace $p in $o//price
                with <price>
                   {$p/@euro-price/text() * $euro-rate}
                     </price>

return
   <price-change>
      <initial>{sum($orders/price/text()}</initial>
      <new>{sum($new-orders/price/text()}</new>
   </price-change>
```

Запрос 6. Использование в одном запросе исходного и модифицированного состояний данных

2.1. Синтаксис функциональных update-выражений

Для удобства выражения логики обработки данных мы предлагаем расширить язык XQuery функциональными update-выражениями трех видов. Выражения первого вида описывают логику “найти и заменить” (*TRANSFORM REPLACE*). Выражения второго вида описывают логику “вставить новый элемент” (*TRANSFORM INSERT*). Выражения третьего вида описывают логику “найти и удалить элемент” (*TRANSFORM DELETE*). Синтаксис этих видов выражений выглядит следующим образом.

1) TRANSFORM REPLACE

```
for $var1 in locpath1
transform replace $var2 in locpath2($var1)
          with expr($var1,$var2)
```

2) TRANSFORM INSERT

```
for $var1 in locpath1
transform insert [into|preceding|following]
                  $var2 in locpath2($var1)
          value expr($var1,$var2)
```

3) TRANSFORM DELETE

```
for $var1 in locpath1
transform delete locpath2($var1)
```

В приведенных синтаксических конструкциях слова, выделенные курсивом, означают следующее:

- *var1, var2* – т переменные языка XQuery;
- *locpath1* – XPath выражение;
- *locpath2($var1)* – XPath выражение, записанное относительно переменной;
- *$var1*;*expr($var1,$var2)* – выражение расширенного языка XQuery, возможно зависящее;от XQuery-переменных *$var1* и *$var2*.

Следует обратить внимание, что XPath-выражение `locpath2` ограничено тем, что оно должно быть записано относительно переменной `var1`. Узлы, соответствующие переменной `var1`, являются корневыми, то есть XPath-выражение `locpath2` вида `var1/.` вернет пустую последовательность элементов.

Далее приводится описание семантики функциональных update-выражений вида `transform replace`. Семантика выражений вида `transform insert` и `transform delete` будет определена путем явного представления этих выражений через выражения вида `transform replace`.

2.2. Семантика функциональных update-выражений вида TRANSFORM REPLACE

Функциональное update-выражение вида `transform replace` возвращает модифицированную копию *последовательности* узлов, возвращаемую XPath-выражения `locpath1`, в которой каждый узел из последовательности, являющейся результатом вычисления XPath-выражением `locpath2`, заменяется на последовательность узлов, возвращаемую выражением `expr`.

Тонкости определения семантики связаны с вычислением выражения `expr`. Пусть результатом вычисления выражения `locpath2` является последовательность узлов `L2`. В случае, когда вычисление выражения expr зависит от узлов из последовательности `L2`, возникают следующие альтернативы:

вычислять выражение `expr` на узлах исходного документа;

1) вычислять выражение `expr` на модифицированных копиях узлов документа.

Чтобы проиллюстрировать различия, рассмотрим следующий абстрактный пример. Пусть необходимо трансформировать XML-данные следующего вида:.

```
<root>
   <a>
      <a>text1</a>
      <b>text2</b>
      <a>text3</a>
   </a>
</root>
```

Рассмотрим функциональное update-выражение, описывающее трансформацию данных, заключающуюся в переименовании всех XML-элементов `a` на `b`:

```
for $r in doc("some.xml")/root
transform replace $a in $r//a with
  <b>
    { ($a/@*,$a/*) }
  </b>
```

В этом выражении `expr($var2)` представляет собой конструктор `<b>{($a/@*, $a/*)}</b>`. Если вычислять выражение на узлах оригинального документа, то представленное функциональное update-выражение вернет следующий результат:

```
<root>
   <b>
      <a>text1</a>
      <b>text2</b>
```

```
      <a>text3</a>
   </b>
</root>
```

Если же вычислять выражение на модифицированных копиях узлов документа, то результат будет отражать рекурсивную семантику трансформации:

```
<root>
   <b>
      <b>text1</b>
      <b>text2</b>
      <b>text3</b>
   </b>
</root>
```

Будем называть семантику вычисление функционального update-выражения с вычислением выражения `expr` на узлах оригинального документа *прямой*, а семантику с вычислением выражения `expr` на модифицированных копиях узлов документа – ь *рекурсивной*.

Выбор семантики функциональных update-выражений зависит от того, какие запросы к XML-данным являются более востребованными. Функциональные update-выражения с прямой семантикой соответствуют логике трансформации данных, когда необходимо трансформировать элементы без учета возможной их вложенности, то есть только XML-элементы, самые близкие к вершине XML-дерева, которые соответствуют критерию выборки. В отличие от этого, функциональные update-выражения с рекурсивной семантикой соответствуют логике обработки данных, когда необходимо выполнить рекурсивную трансформацию данных, то есть, при изменении некоторым образом XML-элемента, удовлетворяющего критерию выборки, требуется выполнить трансформацию и его содержимого, если оно также удовлетворяет критерию выборки.

Нам представляется целесообразным расширять язык XQuery функциональными update-выражениями с рекурсивной семантикой, так как запросы, соответствующие такой семантике, более характерны для обработки иерархической структуры XML-данных.

Для более точного определения рекурсивной семантики функциональных update-выражений, мы приведем его ‘эталонную’ интерпретацию, в которой не учитываются вопросы эффективного вычисления, а лишь определяется результат вычисления функционального update-выражения.

Пусть даны дерево XML-документа `T` и функциональное update-выражение вида

```
for $var1 in locpath1
transform replace $var2 in locpath2($var1)
       with expr($var2)
```


1-ый шаг интерпретации:

Построить T' – полную копию исходного дерева документа T. Построение означает, что уникальные идентификаторы узлов дерева T' будут отличаться от уникальных идентификаторов соответствующих узлов дерева T. Все остальные элементы модели данных языка XQuery для деревьев T и T' будут совпадать. Поскольку построена полная копия, то результаты вычисления XPath-выражений `locpath1` и `locpath2` на деревьях T и T' будут совпадать.

2-ой шаг интерпретации:

Вычислить XPath-выражение `locpath1` над XML-деревом T', получив последовательность узлов $L1'$. Для каждого узла из последовательности $L1'$ получить последовательность узлов $L2'$, являющуюся результатом вычисления выражения `locpath2` над деревом T'.

3-ий шаг интерпретации:

Представить последовательность $L2'$ в обратном порядке документа, получая последовательность $L2''$.

4-ый шаг интерпретации:

Каждый узел из последовательности $L2''$, начиная с первого, заменить последовательностью узлов, являющихся результатом вычисления выражения `expr` на дереве T'.

5-ый шаг интерпретации:

Вернуть последовательность XML-узлов $L1'$. Если в результате выполнения 4-го шага интерпретации, какой-либо из узлов последовательности $L1'$ был заменен последовательностью узлов, то в результат добавляется вся эта последовательность узлов.

2.3. Семантика функциональных update-выражений вида TRANSFORM INSERT и TRANSFORM DELETE

Для определения семантики функциональных update-выражений вида `TRANSFORM INSERT` и `TRANSFORM DELETE` выразим их через функциональные update-выражения вида `TRANSFORM REPLACE`, для которых семантика была определена выше.

В синтаксисе функциональных update-выражений вида `TRANSFORM INSERT` выражение `expr($var1,$var2)` определяет упорядоченную последовательность узлов, которые необходимо вставить. Для каждого целевого узла последовательности, возвращаемой выражением locpath2, результат вычисления выражения `expr($var1,$var2)` вставляется на позицию, определяемую ключевыми словами `into,preceding` и `following`. Если указано ключевое слово `into`, то узлы вставляются в начало последовательности дочерних узлов целевого узла. Если указано ключевое слово `preceding`, то узлы вставляются непосредственно перед

каждым целевым узлом. В случае, если позиция определяется ключевым словом `following`, узлы вставляются после каждого целевого узла.

В синтаксисе функциональных update-выражений вида `TRANSFORM DELETE` выражение `locpath2($var1)` определяет последовательность узлов, которые подлежат удалению из результата. Узлы удаляются вместе со всем их содержимым.

Ниже приводится формальное определение функциональных update-выражений вида `TRANSFORM INSERT` и `TRANSFORM DELETE` через выражения вида `TRANSFORM REPLACE`.

1) INSERT INTO

```
for $var1 in locpath1
transform insert into $var2 in locpath2($var1)
          value expr($var1,$var2)
```

определяется, как

```
for $var1 in locpath1
transform replace $var2 in locpath2($var1)
          with
             element {node-name($var2)}

{($var2/@*,expr($var1,$var2),$var2/*)}
```

2) INSERT PRECEDING

```
for $var1 in locpath1
transform insert preceding $var2 in locpath2($var1)
          value expr($var1,$var2)
```

определяется, как

```
for $var1 in locpath1
transform replace $var2 in locpath2($var1)
          with (expr($var1,$var2),$var2)
```

3) INSERT FOLLOWING

```
for $var1 in locpath1
transform insert following $var2 in locpath2($var1)
          value expr($var1,$var2)
```

определяется, как

```
for $var1 in locpath1
transform replace $var2 in locpath2($var1)
          with ($var2, expr($var1,$var2))
```


4) DELETE

```
for $var1 in locpath1
transform delete locpath2($var1)
```

определяется, как

```
for $var1 in locpath1
transform replace $var2 in locpath2($var1)
          with ()
```

Далее мы рассмотрим некоторые подходы к реализации функциональных update-выражений. Будут выделены возможные ограничения семантики таких выражений, которые позволяют упростить реализацию и сделать ее более эффективной.

3. Подходы к реализации

В первом разделе статьи были выделены две основные проблемы, присущие языку XQuery при выражении логики работы с модифицированным состоянием XML-данных. С точки зрения программиста на языке XQuery, – это громоздкость и невыразительность выражений XQuery, описывающих требуемую логику обработки данных. С точки зрения разработчика XQuery-процессора, – это вопрос эффективного выполнения таких запросов.

Во втором разделе были продемонстрированы компактность и выразительность запросов, записанных на варианте языке XQuery, который расширен предложенными функциональными update-выражениями. Для практического применения необходим способ эффективного вычисления функциональных update-выражений, пригодный для работы с большими объемами XML-данных. При вычислении выражений неприемлемо сканирование данных целиком для выполнения, например, замены одного элемента другим. Необходимо гарантировать произвольный доступ к данным (*random-access*). Далее будут рассматриваться возможные подходы к реализации, удовлетворяющие условию произвольного доступа к данным.

При введении функциональных update-выражений в язык XQuery обсуждалась доступность в одном XQuery-выражении исходного и модифицированного состояний данных. Предлагаемые ниже подходы к реализации вносят некоторые ограничения на это свойство функциональных update-выражений, о чем будет говориться более детально в каждом конкретном случае.

3.1. Прямое выполнение с откатом

Один их возможных механизмов реализации функциональных update-выражений – это замена таких выражений традиционными (не функциональными) update-выражениями с последующим откатом к исходному состоянию данных. Несколько похожие идеи возникали и в контексте объектно-реляционных баз данных [15].

Реализованный во многих XML-СУБД [5,11,12,13] язык модификации XML-данных можно использовать для изменения состояния данных, а язык XQuery – для вычисления запроса над измененным состоянием. Заимствование в качестве синтаксиса функциональных update-выражений синтаксиса обычных update-выражений делает такую замену вполне естественной. Еще раз рассмотрим Запрос 3 из раздела 2:

```
for                          $r                          in
doc("db.xml")/db/movie[gender="fiction"]/review
transform replace $d in $r//director with
  <a
href="{doc("b")//person[name=$d/@name]/homepage/text()}"
>
    {$d/text()}
  </a>
```

Такой запрос можно вычислить, перезаписав его в виде трех выражений языка XQuery и языка модификации XML-данных (на примере синтаксиса языка модификации XML-данных, реализованного в XML СУБД Sedna).

1-ый шаг:

```
UPDATE
replace $d in
   doc("db.xml")/db/movie[gender="fiction"]/review
$r//director with
  <a
href="{doc("b")//person[name=$d/@name]/homepage/text()}">
    {$d/text()}
  </a>
```

2-ой шаг:

```
doc("db.xml")/db/movie[gender="fiction"]/review
```

3-ий шаг:

```
ROLLBACK
```

Первое update-выражение изменяет состояние данных, второе выражение вычисляется над измененным состоянием данных и формирует результат запроса. Третье выражение ‘придает’ функциональную семантику вычислению первых двух выражений – откатывает базу данных к начальному состоянию. Очевидно, что из-за побочных эффектов от вычисления первого выражения теряется возможность доступа к начальному состоянию данных во втором выражении.

Эффективность вычисления такой последовательности запросов определяется эффективностью реализации update-выражений в XML-СУБД и эффективностью вычисления операции *ROLLBACK*. Если реализация update-выражений удовлетворяет требованию произвольного доступа к данным, то и описанный подход отвечает этому требованию.

Такой подход к реализации расширения языка XQuery функциональными update-выражениями ограничивает их семантику тем, что выражения расширенного языка не обеспечивают доступ к исходному состоянию данных.

3.2. Функциональные update-выражения, возвращающие текстовое значение

Для эффективной реализации функциональных update-выражений можно ограничить их семантику таким образом, чтобы такие выражение возвращали сразу сериализованное представление XML-данных, то есть значения текстового типа данных. Такая семантика допускает эффективную реализацию, так как при вычислении не требуется реконструировать узлы документа.

Основное ограничение такого подхода – невозможность композиции выражений. Сериализованное представление не имеет смысла обрабатывать на языке XQuery, так как к нему не применимы XPath-выражения [3]. Из этого следует, что реализация сложной логики обработки данных на расширенном таким образом языке XQuery затруднена, так как не допускает декомпозиции. Типичный пример – невозможность отделить логику обработки данных от логики их представления, т.е невозможность следовать одному из лозунгов современных технологий разработки Web-приложений. Логично представить, что бизнес-логика обработки XML-данных и построение HTML-представления XML-данных могут реализовываться различными XQuery-функциями, которые разрабатывают разные специалисты. В таком сценарии текстовые update-выражения нельзя использовать для описания логики работы с данными, так как текстовый результат не имеет смысл передавать функции, строящей HTML-код.

Таким образом, возвращение XML-данных в сериализованном виде не вносит ограничений на доступность исходного состояния данных, но ограничивает возможности работы с модифицированной копией данных, не допуская композиции выражений.

Тем не менее, выделенное ограничение не делает такой подход неприменимым. Например, популярный язык трансформации XML-данных XSLT [4] обладает схожей семантикой. Результатом вычисления выражений XSLT также является сериализованное представление XML-данных. Мы считаем, что для большого класса трансформационных задач такой подход является достаточным и планируем более детально исследовать присущие ему ограничения.

3.3. Реализация с использованием версий

Функциональные update-выражения допускают эффективную реализацию на основе использования версий узлов XML-документа. Неформально алгоритм можно сформулировать следующим образом. Функциональные update-выражения возвращают последовательность новых узлов `{N}`, возможно, изменяя некоторые узлы `{I}`, содержащиеся внутри этих узлов.

Тогда:

1) для каждого из изменяемых узлов `i` последовательности `{I}` стоится узел `i'`, соответствующий измененному состоянию, и помечается некоторым номером версии `x`; изменяемые узлы из последовательности `{I}` в исходном документе не изменяются;
2) каждый узел из последовательности `{N}` помечается номером версии `X` и возвращается как часть результата;
3) для каждого функционального update-выражения номер версии новых узлов больше номеров версий, использовавшихся в предшествующих update-выражениях; начальное состояние данных соответствует некоторой нулевой версии;
4) При навигации по данным, начиная с некоторого начального узла, выбираются узлы с версией, номер которой равен или меньше номера версии начального узла.

Такой подход наименьшим образом ограничивает семантику функциональных update-выражений, но требует сложной реализации с поддержкой версий. При таком подходе в выражениях доступно как начальное, так и измененное состояние данных, и возможна произвольная композиция выражений, что обеспечивает наибольшую выразительность расширенного языка.

4. Интеграции языка запросов к XML-данным XQuery и языка модификации XML-данных средствами функциональных Update-выражений.

4.1. Текущая ситуация и проблемы

Язык XQuery предназначен для описания запросов к XML-данным. На момент написания данной работы XQuery все еще не получил статус рекомендации консорциума W3C, но близок к этому, являясь кандидатом к рекомендации (*candidate recommendation*). К сожалению, языковые средства модификации XML-данных отстают от средств выборки данных. Консорциум W3C в настоящее время работает над языком для описания модификаций XML-данных, но пока что существует только документ с предварительными требованиями (*facility requirements*) к этому языку.

Параллельно с работой консорциума W3C по стандартизации языка XQuery опубликован ряд исследовательских работ, посвященных языковым средствам модификации XML-данных [7,8,9]. В существующих на сегодняшний день реализациях XML-СУБД, как правило, предлагаются средства как для запросов к XML-данным, так и для модификации XML-данных. Для описания запросов в большинстве случаев предоставляется некоторое подмножество языка XQuery. Для описания модификации XML-данных разработчики предоставляют различные вариации предложенных в исследовательской среде языков модификации XML-данных. При этом различия языков бывают достаточно

серьезными и касающимися не только синтаксиса выражений, но и семантики их вычисления.

Общей чертой существующих на сегодняшний момент предложений языков модификации XML данных является их отчужденность от языка запросов к XML-данным XQuery. Если быть более точными, то необходимо отметить, что в некоторых работах [7] затрагивается вопрос о связи между языком XQuery и языком модификации XML-данных, но полное решение этой проблемы в настоящее время не представлено.

Актуальность задачи тесной интеграции языка XQuery и языка модификации XML-данных проявляется при анализе требований практических предложений. Широко распространено мнение, что язык XQuery по отношению к XML-данным, занимает такое же положение, как язык SQL по отношению к реляционным данным. Мы полагаем, что это не совсем так, потому что язык XQuery во многих случаях претендует на большую роль. Если выражения языка SQL в практических приложениях используются как средства доступа к базе данных из языка программирования общего назначения, то с языком XQuery ситуация нам представляется несколько иной. По нашему мнению, язык XQuery является самодостаточным для некоторых классов приложений, интенсивно обрабатывающих данные. Среди таких классов приложений мы видим Web-приложения, Web-сервисы, приложения класса управления контентом и ряд других. Язык XQuery содержит мощные средства выборки данных, средства трансформации данных, необходимые управляющие конструкции, что позволяет его использовать для описания бизнес-логики приложений. Главное преимущество такого подхода заключается в том, что разработка приложений ведется на одном языке в рамках одной модели данных, что позволяет избежать известной проблемы потери соответствия (*impedance mismatch*) [14], заключающейся, применительно к XML-данным, в несоответствии модели данных языка программирования и модели данных XML.

Для практического воплощения такого подхода необходима тесная интеграция языка XQuery и языка модификации XML-данных. Типичные проблемы, выявленные при разработке практических приложений на языке XQuery, сводятся к необходимости выражения логики обработки данных, использующей различные состояния данных, например:

- изменить состояние данных при одном условии, изменить состояние данных другим образом при невыполнении этого условия;
- изменить состояние данных при одном условии, выполнить запрос над исходным состоянием при другом условии;
- выполнить запрос над исходным состоянием данных; изменить состояние данных; выполнить новый запрос над исходным и модифицированным состоянием данных.

Для решения подобных задач необходима возможность композиции выражений языка XQuery и языка модификации XML-данных. Именно это мы подразумеваем под *тесной* интеграцией языка XQuery и языка модификации

XML-данных. Однако решение такой задачи ‘напрямую’ порождает серьезные проблемы в силу функциональной природы языка XQuery.

По степени интеграции языка модификации XML-данных и языка запросов XQuery, существующие реализации можно разделить на два класса.

I. Неинтегрированные решения.

   Примером такого подхода является реализация языка модификации XML данных в XML-СУБД Sedna. Для выражения логики приложения разработчик может использовать любую последовательность из выражений языка XQuery и языка модификации XML-данных, но выражения их этой последовательности никак не связаны друг с другом. То есть, в сущности, это последовательное выполнение выражений из двух различных языков над одними XML-данными.

II. Интегрированные решения с ограниченной семантикой

   Такой подход реализован, например, с прирожденной XML-СУБД MarkLogic CMS [11]. В этой системе предлагается ‘прямо’ расширить язык XQuery конструкциями, модифицирующими состояние XML-данных. С точки зрения модели данных XQuery, такие конструкции возвращают пустую последовательность элементов, но их вычисление происходит с побочными эффектами, заключающимися в изменении состояния данных в базе XML-данных. Такое прямое внедрение в функциональную среду языка XQuery выражений с побочными эффектами приводит к ряду проблем. Суть этих проблем связана с тем, что выражения, модифицирующие XML-данные, могут непредсказуемо менять XML-данные, которые уже логически связаны с конструкциями языка XQuery, внешними по отношению к этим выражениям. Для преодоления этих проблем семантика интегрированного языка в XML СУБД MarkLogic CS ограничена следующим образом. Предполагается, что все изменения, связанные с update-выражениями, вступают в силу только по окончании вычисления всего выражения расширенного языка. То есть в пределах одного выражения доступно только исходное состояние данных. После вычисления выражения состояние данных меняется. Используя последовательность таких расширенных выражений языка XQuery, разработчик выражает логику обработки данных.

При таком подходе к интеграции языковых средств выборки и модификации XML-данных выражения расширенного языка обладают сложной семантикой с отложенным побочным эффектом и недоступным модифицированным состоянием данных. Программы на расширенном таким образом языке обладают ограниченной выразительностью и не являются простыми для восприятия.

Таким образом, для написания практических XML-приложений недостаточно наличие несвязанной комбинации средств языка XQuery и языка изменений

XML-данных. Анализ различных прикладных задач показывает, что требуется тесная интеграция выражений языка XQuery и выражений языка модификации XML-данных. Тесная интеграция означает возможность описывать логику обработки данных, используя композицию выражений языка XQuery и выражений языка модификации XML-данных. Требуемая интеграция в настоящее время не реализуется вовсе или предоставляется с ограничениями.

4.2. Интеграция языковых средств выборки и модификации XML-данных на основе функциональных update-выражений

Расширение языка XQuery выражениями с побочными эффектами порождает ряд проблем, связанных с нарушением целостности данных и необходимостью вносить изменения в модель данных языка XQuery. Эти проблемы являются следствием введения в функциональный язык выражений, выполняющихся с побочными эффектами.

Мы предлагаем подход к интеграции языковых средств выборки XML-данных и изменения XML-данных на основе идеи функциональных update-выражений. Такой выбор обусловлен двумя соображениями:

1) функциональные update-выражения вычисляются без побочных эффектов, и, следовательно, могут быть естественным образом интегрированы в язык XQuery;
2) функциональные update-выражения по выразительности эквиваленты языкам модификации XML-данных.

Это позволяет описывать логику обработки XML-данных на расширенном языке XQuery, используя функциональные update-выражения для получения модифицированных версий данных. Логика приложения описывается, таким образом, с доступностью начального состояния данных и всех модификаций. Единственное необходимое расширение при таком подходе заключается в возможности отражать модификации данных в базе данных. То есть требуется возможность указывать, какая именно версия данных должна стать актуальной после завершения вычислений.

Ниже приведен пример выражения на языке XQuery, расширенном функциональными update-выражениями, с возможностью сохранения состояния XML-данных в базе данных.

4.3. Синтаксис и семантика расширенного языка

Программы на языке, тесно связывающем языковые средств выборки и модификации XML-данных, могут быть записаны следующим образом:

```
Expr1 UPDATE Expr2
```

В такой записи `Expr1` и `Expr2` – это выражения на языке XQuery, расширенном функциональными update-выражениями. Первое выражение содержит всю логику обработки данных и его результат представляет результат вычисления всей программы. Второе выражение определяет

последовательность узлов, версию которых нужно сделать актуальной в базе данных, включая все вложенные элементы. Рассмотрим простой пример такой программы.

Пример. Приостановить все заказы, содержащие товар `tires`. Изменить статус заказа и вставить соответствующие комментарии.

```
let $order-status:=
  for $order in doc(“orders”)//order[status=”ready and

OrderLine/ItemName=”tire”]
  transform replace $s in $order/status
            with <status>suspended</status>
let $order-comment:=
  for $order in $order-status
  transform insert <comment>suspended</comment>
            into $order/OrderLine[ItemName=”tire”]
return $order-comment
UPDATE
  $order-comment
```

Выражение до ключевого слова `UPDATE` представляет собой выражение на языке XQuery, расширенном функциональными update-выражениями. Переменная XQuery `$order-status` связывается с версией данных, модифицированных таким образом, что в элементах `order`, удовлетворяющих заданному предикату, изменен элемент `status`. Переменная XQuery `$order-comment` связывается с версией данных, в которой в каждый элемент `order` вставляется необходимый комментарий. Все выражение до ключевого слова `UPDATE` возвращает XML-данные – последовательность узлов, связанную с переменной `$order-comment`.

Выражение после ключевого слова `UPDATE` возвращает последовательность узлов, состояние которых надо сделать актуальным в базе данных, то есть зафиксировать как текущее. Для реализации такого подход требуются поддержка версий в реализации функциональных update-выражений и коррекция их семантики. В частности, необходима информация о соответствии между узлами в оригинальном документе и соответствующими новыми версиями узлов.

Можно заметить, что предлагаемый подход реализует отложенную семантику отражения изменений в базе данных. Однако это свойство не влияет на выразительность запросов и удобство программирования, поскольку разработчику доступно модифицированное состояние данных до его отражения в базе данных. Более того, доступны и начальное, и все модифицированные состояния данных.

Таким образом, такой подход обеспечивает естественную интеграцию языковых средств выборки и модификации XML-данных, предоставляя

единую платформу для разработки XML-приложений на расширенном языке XQuery. Основные преимущества предложенного подхода состоят в следующем:

1) возможность композиции выражений языка XQuery и выражений языка модификации XML-данных;
2) доступность при выражении логики обработки данных исходного и модифицированного состояний данных;
3) единая среда разработки XML-приложений.

5. Заключение

В настоящее время среди разработчиков XML-приложений сложился стереотип того, что язык XQuery обладает преимуществами перед языком XSLT при обработки XML-данных, ориентированных на данные, в то время как язык XSLT предпочтителен при работе с XML-данными, ориентированными на документы.

Предложенные в статье функциональные update-выражениями значительно улучшают трансформационные возможности языка XQuery, расширяя его применимость для широкого класса задач обработки XML-данных независимо от их ориентированности. Мы полагаем, что введение функциональных update-выражений позволит избежать существующей на сегодняшний день практики комбинтрования многих языков в XML-приложениях (например, языков XSLT и XQuery).

Функциональные update-выражения – это естественный подход к реализации тесной интеграции языка XQuery и языка модификации XML-данных. Такая интеграция необходима для эффективной разработки XML-приложений на одном языке в пределах одной модели данных языка XQuery.

Идеи, изложенные в работе, требует более детальной проработки и представляют собой предмет дальнейших исследований авторов.

Литература

1. S. Boag, D. Chamberlin, M. Fernández, D. Florescu, J. Robie, J. Siméon. XQuery 1.0: An XML Query Language. W3C Candidate Recommendation 3 November 2005. http://www.w3.org/TR/xquery/
2. T. Bray, J. Paoli, E. Maler, F. Yergeau. Extensible Markup Language (XML) 1.0 (Third Edition). W3C Recommendation 4th February 2004. http://www.w3.org/TR/2004/REC-xml-20040204/
3. J. Clark, S. DeRose. XML Path Language (XPath) 2.0. W3C Working Draft 15 September 2005. http://www.w3.org/TR/2005/WD-xpath20-20050915/
4. M. Kay. XSL Transformations (XSLT) Version 2.0. W3C Candidate Recommendation 3th November 2005. November 1999. http://www.w3.org/TR/xslt20/
5. М. Гринев, С. Кузнецов, А. Фомичев. Особенности СУБД Sedna. XML-СУБД Sedna: технические особенности и варианты использования. Открытые системы, #08/2004. http://www.osp.ru/os/2004/08/036.htm
6. Д. Чемберлин. XQuery: язык запросов XML. Открытые системы, #01/2003. http://www.osp.ru/os/2003/01/061.htm
7. I. Tatarinov. Updating XML. SIGMOD Conference 2001. http://www.cis.upenn.edu/~zives/research/updatingXML.pdf

8. P. Lehti, Design and Implementation of a Data Manipulation Processor for an XML Query Language, Technische Universitt Darmstadt Technical Report No. KOM-D-149, August 2001. http://www.ipsi.fraunhofer.de/lehti/
9. A. Laux and L. Martin. XUpdate — XML Update Language, 2000. http://xmldb-org.sourceforge.net/xupdate/xupdate-wd.html
10. P. Biron, K. Permanente, A. Malhotra. XML Schema Part 2: Datatypes Second Edition. W3C Recommendation 28 October 2004. http://www.w3.org/TR/xmlschema-2/
11. Technical Overview: An Informal Introduction to Mark Logic's Content Interaction Server, XQuery, and xq:zone. http://xqzone.marklogic.com/howto/tutorials/technical-overview.xqy
12. X-Hive/DB Web site. http://www.x-hive.com/products/db/index.html
13. Introduction to Berkley DB XML. http://www.sleepycat.com/xmldocs/intro_xml/BerkeleyDBXML-Intro.pdf
14. К.Ю. Лисовский. Разработка XML-приложений на языке Scheme. Программирование. М.: Наука, 2002 – Вып.28, №1 – С. 20-32.
15. J. Gana, W. Kim. Transaction Transaction management in an object-oriented database system. ACM SIGMOD Record, Volume 17 , Issue 3 (June 1988), pp. 37 – 45.