XQuery Update Facility

Based on the slides from Ioana Manolescu, accessible from:
http://www-rocq.inria.fr/~abitebou/Master-SSD/slxquupdate.pdf
Outline

- **Fundamental Aspects**
  - Why use XQuery Update?

- **Design Principles**
  - Requirements for the XQuery Update Language
  - Main Concepts for XQuery Update
  - Query Processing with XQuery Update

- **XQuery Updates**
  - Inserting, removing, updating, renaming, and transforming
  - Programming with XQuery Update

- **Comparison Between SQL and XQuery Update**
Why use XQuery Update?

XQuery is a read-only language: it can return a new instance of the XQuery data model, but it cannot modify an existing instance.

The parallel with SQL is having:

select... from... where...

without having:

insert into table... update table...

Since applications require both reading and updating XML data,..

XQuery Update Facility: an extension for the XQuery language

http://www.w3.org/TR/xquery-update-10/
Requirements for XQuery Update

Expressive power:
- Insertion
- Removal
- Updating
- Copy with new identity

An extension of the existing XQuery language:
- Avoids learning a new language, and thus facilitates program development and comprehension

- It should have a well-defined semantics
- It should be a concise language
- It should lend itself to efficient implementation...
Fundamental Concepts in the XQuery Update Facility

All Xquery expressions can now be classified as:
- Expressions performing updates (with side-effects)
- Expressions that do not perform updates

XQuery Update introduces **five new types of expressions**:
- `insert`, `delete`, `replace`, `rename`: expressions that perform updates
- `transform`: expressions that do not perform updates

The XQuery Update Facility Specifies:
- The places where each type of expression can be used
- The syntax and semantics for each of the new expressions
Query Processing in XQuery Update

The evaluation of an expression introduces:

- An instance of the XQuery data model
- A list of pending updates: this is a set of update primitives, i.e., changes in the state of a node, which should be applied after evaluation

- In the current specification of XQuery Update, one of the two should be empty. Nonetheless, this may change in the future.
  - The evaluation of a standard XQuery expression produces an instance of the XQuery data model

- Each update primitive has a target node.
- The update primitives are checked in order to detect eventual update conflicts. If no such conflicts exist, then they are applied.
Insert is an XQuery updating expression

General form:

```
InsertExpr ::= insert (node|nodes) SourceExpr
           InsertExprTargetChoice TargetExpr

InsertExprTargetChoice ::= ((as (first|last))? into)
                       | after | before
```

- SourceExpr ::= ExprSingle
- TargetExpr ::= ExprSingle

- The first expression is called the source and the second is called the target.
- The source and target expressions should not themselves perform updates.
- Examples:
  - Insert an element year after the publisher for the 1st book
    ```
    insert nodes <year>2005</year>
    after doc(bib.xml)/books/book[1]/published
    ```
  - Through an $article variable, insert the associated list of authors as descendants of the 3rd book
    ```
    insert nodes $article/author
    as last into doc(bib.xml)/books/book[3]
    ```
ExprSingle ::= FLWORExpr
| QuantifiedExpr
| TypeswitchExpr
| IfExpr
| InsertExpr
| DeleteExpr
| RenameExpr
| ReplaceExpr
| TransformExpr
| OrExpr
Insert Expressions (2)

The list of pending updates (lpu) is obtained as follows:

- Evaluate the updating target, which are the nodes that should receive the new descendants as part of the updates
- For each of those nodes, add to the lpu the corresponding append-descendant operations

**Example:** Navigating through multiple variables, insert a new police report in the list of reports for a given accident

```xml
insert node $new-police-report as last
into doc("insurance.xml")//policies/policy[id=$pid]
/driver[licence=$licence]/accident[date=$dateacc]
/police-reports
```

- Find the appropriate police-reports element
- For each element inside the $new-police-report variable, add a new append-last-descendant operation to the lpu
Delete Expressions

`delete` is an XQuery expression that performs updates. It produces a non-empty list of pending updates.

General form:

```
deleteExpr ::= delete (node|nodes) TargetExpr
TargetExpr ::= ExprSingle
```

Examples:

- Delete the name of the last author of the first book from a given document
  ```
delete nodes doc(bib.xml)/books/book[1]/
  author[last()]
  ```

- Delete the email messages that are more than 365 days old
  ```
delete nodes /email/message[fn:currentDate()- date >
  xdt:dayTimeDuration(P365D)]
  ```
Replace Expressions

`replace` is an XQuery expression that performs updates. It produces a non-empty list of pending updates.

General form:

```
ReplaceExpr ::= replace (value of)? node TargetExpr
               with ExprSingle
TargetExpr ::= ExprSingle
```

If `value of` is specified, then the `replace` expression is used to modify the value of a node, although preserving its identity.

Examples:

- Replace the publisher of the first book by the publisher of the second
  
  ```
  replace node doc("bib.xml")/books/book[1]/publisher
          with doc("bib.xml")/books/book[2]/publisher
  ```

- Increase the price of the first book by 10%
  
  ```
  replace value of node doc("bib.xml")/books/book[1]/price
          with doc("bib.xml")/books/book[1]/price*1.1
  ```
rename is an XQuery expression that performs updates.

General form:

\[
\text{RenameExpr} ::= \text{rename node TargetExpr as NewNameExpr}
\]

\[
\text{TargetExpr} ::= \text{ExprSingle}
\]

\[
\text{NewNameExpr} ::= \text{ExprSingle}
\]

Examples:

- Rename the first author element from the first book to main-author.

  \[
  \text{rename node doc("bib.xml")/books/book[1]/author[1] as "main-author"}
  \]

- Rename the first author element from the first book, naming it after the value for the variable $newname.

  \[
  \text{rename node doc("bib.xml")/books/book[1]/author[1] as $newname}
  \]
**Transform Expressions (1)**

**Transform** is an XQuery expression that **does not perform updates**

General form:

```
TransformExpr ::= copy $VarName := ExprSingle
               (, $VarName := ExprSingle)*
               modify ExprSingle
               return ExprSingle
```

It can be used to create modified copies of existing nodes in a given instance of the XQuery data model.

**Example**: return all managers, omitting their salaries and adding to the salary elements a new attribute `xsi:nil`.

**Nota**: Transform expressions can also be re-written as expressions from the standard XQuery language (e.g., using FLWORs and constructors).
Original Document
<employees>
  <employee mgr="true" dept="Toys">
    <name>Smith</name>
    <salary>100000</salary>
  </employee>
  <employee dept="Toys">
    <name>Jones</name>
    <salary>60000</salary>
  </employee>
  <employee mgr="true" dept="Shoes">
    <name>Roberts</name>
    <salary>150000</salary>
  </employee>
</employees>

Required result
<employee mgr="true" dept="Toys">
  <name>Smith</name>
  <salary xsi:nil="true"/>
</employee>
<employee mgr="true" dept="Shoes">
  <name>Roberts</name>
  <salary xsi:nil="true"/>
</employee>

It can be done in XQuery. Try this as an exercise....
Return all managers, omitting their salaries and adding to the salary elements a new attribute `xsi:nil`.

```xml
for $e in doc("employees.xml")//employee
  where $e/@manager=true()
  return
    copy $emp := $e
    modify (   
      replace value of node $emp/salary with "" ,  
      insert nodes (attribute xsi:nil {"true"})  
      into $emp/salary
    )
  return $em
```
Programming with XQuery Update (1)

Scenario: Address book synchronization:

- There is an archive version of the address book and two copies
- \( c_1 = a \) and \( c_2 \neq a \) => propagate \( a \) to \( c_2 \) and \( c_1 \)
- \( c_1 \neq a, c_2 \neq a \) =>
  - If possible, list all the differences and propagate them to \( a \), and afterwards to \( c_1 \) and \( c_2 \)
  - If not, return an error

- Entries in the address book have the following general form:

  ```xml
  <entry>
    <name> Benjamin </name>
    <contact> benjamin@inria.fr </contact>
  </entry>
  
  <entry>
    <name> Anthony </name>
    <contact> tony@uni-toulon.fr </contact>
  </entry>
  ```
for $a$ in doc("archive.xml")//entry,
   $v1$ in doc("copy1.xml")/version/entry,
   $v2$ in doc("copy2.xml")/version/entry
where $a$/name = $v1$/name
and $v1$/name = $v2$/name
return
   if ($a/contact = $v1/contact and $v1/contact =$v2/contact
   then ()
   else
      if ($v1/contact = $v2/contact)
         then replace value of node $a/contact with $v1/contact
      else
         if ($a/contact = $v1/contact)
            then ( replace value of node $a/contact
                     with $v2/contact ,
                     replace value of node $v1/contact
                     with $v2/contact ...
if ($a/contact = $v1/contact )
then ...
else
    if ($a/contact = $v2/contact )
then ( replace value of node $a/contact 
    with $v1/contact ,
    replace value of node $v2/contact 
    with $v1/contact )
else ( insert node
      <fail> <arch>{ $a }</arch>
      <v1>{ $v1 }</v1>
      <v2>{ $v2 }</v2>
    </fail>
    into doc("log.xml")/ log ),
replace value of node doc("archive.xml")/*/last-synch-time with current-dateTime()
XQuery Update is not a query language as powerful and as expressive as SQL, because it does not have things like:

- Control over the scope of snapshots, i.e., control on how the updates become visible to other queries (i.e., SQL isolation)

- Control over atomicity, i.e., control over which expressions should be executed atomically (i.e., SQL transactions)

- Error handling (although XQuery 3.0 introduces try/catch expressions)
XQuery Update Implementations

- eXist
- MonetDB
- XQilla
- Saxon (just on the commercial version)
- QizX (Updating expressions over XML databases)
References

- XQuery: A Guided Tour (a specific chapter from the book *XQuery from the Experts*)
  - www.perfectxml.com/XQuery.html

- Ioana Manolescu, *slides about XQuery*, course about “Web Data Management and Distribution”, Master Recherche Informatique Paris Sud,
  - www-rocq.inria.fr/~abitebou/Master-SSD/slquery.pdf
Questions?