Chapter 5

XML Updates
Updating XML Documents

- Naive procedural approach based on DOM operations
  - Problem: DOM is not declarative and requires a programming language

- XQuery supports arbitrary transformations
  - Why there is a need for an XML data manipulation language?

- Requirements for an XML data manipulation language
  - Arbitrary manipulation of XML documents must be supported
  - Declarative definition of updates
  - Interplay with declarative XML query languages
XQuery Update Facility 1.0

- W3C Recommendation 17 March 2011
- Extends XQuery by operations for updating XML documents
- Comprises operations on an XQuery data model instance (node sequence / atomic value)
  - Insert node in/after/before of a node
  - Remove a node
  - Modify/Replace node properties
  - Rename a node
  - Create and modify a node copy

- Hint: Tutorials on XQuery Update
  - [http://docs.basex.org/wiki/XQuery_Update#replace](http://docs.basex.org/wiki/XQuery_Update#replace)
**XQuery Update Operations**

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>`insert (node</td>
<td>nodes) Source (before</td>
</tr>
<tr>
<td>`delete (node</td>
<td>nodes) Target`</td>
</tr>
<tr>
<td><code>replace (value of)? Target with SingleExpr</code></td>
<td>ReplaceExpr</td>
</tr>
<tr>
<td><code>rename node Target as SingleExpr</code></td>
<td>RenameExpr</td>
</tr>
<tr>
<td><code>copy $variable := SingleExpr modify SingleExpr return SingleExpr</code></td>
<td>CopyExpr</td>
</tr>
</tbody>
</table>

- Source and target nodes are `SingleExpr`, i.e., XQuery expressions that yield sequences.
- `SingleExpr = FLWORExpr|QuantifiedExpr|IfExpr|TypeswitchExpr|OrExpr|InsertExpr|DeleteExpr|ReplaceExpr|RenameExpr|CopyExpr` → update expressions can be used among others in XQuery return clauses.

- Keywords node and nodes can be used exchangeable independent of the number of nodes inserted/removed.

- Insert position is determined as follows:
  - `before`/`after`: node as preceding/succeeding of the target node
  - `as first/last into`: first/last child node of the target node
  - `into`: position in the target node such that another insert is not affected
XQuery Updates : Example (1)

where $book/title = "Data on the Web"
return (  
  delete node $book/@year,
  insert node <author>Buneman</author> into $book,
  insert node (attribute isbn {"1-55860-622-X"},
    <author>Suciu</author>) into $book,
  replace value of node $book/abstract with "New Abstract",
  rename node $book/abstract as "summary"
)

(: state before update :)
<book year="1999" >
 <title>Data on the Web</title>
 <abstract>This book</abstract>
 <author>Abiteboul</author>
</book>

(: state after update :)
<book isbn="1-55860-622-X">
 <title>Data on the Web</title>
 <summary>New Abstract</summary>
 <author>Abiteboul</author>
 <author>Buneman</author>
 <author>Suciu</author>
</book>
**XQuery Updates : Example (2)**

```xml
<book isbn="1-55860-622-X">
  <title>Copy of: Data on the Web</title>
  <summary>New Abstract</summary>
  <author>Abiteboul</author>
  <author>Buneman</author>
  <author>Suciu</author>
</book>
```

copy creates **transient** copy of a node and modifies this node. If this node should be persistent, then it must be made persistent explicitly.

To modify the example:

```xml
modify (  
  replace value of node $book/title with concat('Copy of: ', $book/title),  
  delete node $book/@year,  
  insert node <author>Buneman</author> into $book,  
  insert node (attribute isbn {"1-55860-622-X"}, <author>Suciu</author>) into $book,  
  replace value of node $book/abstract with "New Abstract",  
  rename node $book/abstract as "summary"
)
return $book
```

(: state after update :)  
```
<book isbn="1-55860-622-X">
  <title>Copy of: Data on the Web</title>
  <summary>New Abstract</summary>
  <author>Abiteboul</author>
  <author>Buneman</author>
  <author>Suciu</author>
</book>
```
Conclusions

- We know how XML documents can be updated declaratively, too!
- We now want to know how XML documents can be stored
  - Mapping XML to databases