Universität Freiburg
Institut für Informatik
Prof. Dr. Peter Fischer
Lecuture XML and Databases
Winter Semester 2011/12

Due date/discussion: 13.1.2012

Solution Sheet 7

XQuery Update Facility, XQuery Scripting Extension

Exercise 1: XQuery with Updates

1.1.

```
<r1>{
for $e in doc("flights.xml")//Flight
return
  copy $je := $e
  modify delete node $je/seats
  return $je
}</r1>
```

1.2.

```
delete node doc("flights.xml")//Passenger[name = "Santa Claus"]
```

1.3.

```
let $passRef := "000111"
let $flightRef := "LX123"
let $newDate := "2009-12-26"
let $reservation := doc("flights.xml")//Reservation[passRef = $passRef and
flightRef = $flightRef]
let $oldFlight := doc("flights.xml")//Flight[@flightId = $flightRef]
let $newFlight := doc("flights.xml")//Flight[source = $oldFlight/source and
destination = $oldFlight/destination and date = $newDate]
return
if($newFlight)
then
  replace value of node $reservation/flightRef
  with $newFlight/@flightId
else ()
```

1.4.

```
let $passNo := "123457"
let $flightNo := "LX123"
let $date := "2009-12-24"
let $ccnumber := "1234567890"
return
if (doc("flights.xml")//Passenger[passportnumber = $passNo]
    and doc("flights.xml")//Flight[date = $date and @flightId =
$flightNo]/seats
          >
          count(doc("flights.xml")//Reservation[flightRef=$flightNo])
   )
then
  insert node <Reservation>
                <flightRef>{$flightNo}</flightRef>
                <passRef>{$passNo}</passRef>
                <creditCard>{$ccnumber}</creditCard>
```

```
            </Reservation>
  into (doc("flights.xml")//Reservation)[1]
else ()
```

1.5.

```
let $airId := "NPL"
return
(
  delete node doc("flights.xml")//Airport[@airId = $airId],
  for $fl in doc("flights.xml")//Flight
  where $fl/source = $airId or $fl/destination = $airId
  return
  (
    delete nodes $fl,
    for $res in doc("flights.xml")//Reservation
    where $res/flightRef = $fl/@flightId
    return
      delete nodes $res
  )
)
```

Exercise 2: Halloween Problem and Snapshot Semantics

All leaf nodes (without any children, i.e. only with text or empty) will be deleted. Although their parents have no more children after this, they are not deleted. This is because of snapshot semantics: during query execution, PULs are generated but updates are applied only at the end.

Exercise 3: Simple expressions and updating expressions

3.1. OK (updating expression)
3.2. Not OK (a comma expression cannot contain a simple expression and an updating expression), but OK in the Scripting Extension.
3.3. OK (errors are vacuous expressions, which are compatible with updating expressions)
3.4. Not OK (targets must be simple expressions), OK in the Scripting Extension.
3.5. Not OK (this function expects an item), OK in the Scripting Extension.
3.6. Not OK (a transform expression is a simple expression, because it returns a sequence of items, not a PUL), OK in the Scripting Extension.

Exercise 4: XQuery Scripting Extension

4.1.

```
let $flights := doc("flights.xml")
let $santa-claus:= $flights//Passenger[name="Santa Claus"]
let $reservations := $flights//Reservation[passRef=$santa-
claus/passportnumber]
let $nodes-to-delete := ($santa-claus, $reservations)
return (delete nodes $nodes-to-delete, count($nodes-to-delete))
```

4.2.

```
declare sequential function local:compute()
{
    declare $i := 2;
    declare $computed :=
     <computed>
       <result x="0">1</result>
       <result x="1">1</result>
     </computed>;
    while(max($computed/result)< 100 ) {
      insert node
          <result x="{$i}">{
            $computed/result[@x=$i - 2]+ $computed/result[@x=$i - 1]
          }</result>
        as last into $computed;
      set $i := $i+1;
    };
    exit returning $computed;
};

local:compute()
```

W3C solution:

```
declare sequential function local:compute()
{
  declare $a as xs:integer := 0;
  declare $b as xs:integer := 1;
  declare $c as xs:integer := $a + $b;
  declare $fibseq as xs:integer* := ($a, $b);

  while ($c < 100) {
     set $fibseq := ($fibseq, $c);
     set $a := $b;
     set $b := $c;
     set $c := $a + $b;
  };

  exit returning $fibseq;
};

local:compute()
```

In a block:

```
block
{
  declare $a as xs:integer := 0;
  declare $b as xs:integer := 1;
  declare $c as xs:integer := $a + $b;
  declare $fibseq as xs:integer* := ($a, $b);

  while ($c < 100) {
```

```
      set $fibseq := ($fibseq, $c);
      set $a := $b;
      set $b := $c;
      set $c := $a + $b;
  };

  $fibseq;
}
```