CS 240A : Databases and Knowledge Base
Project #1

Ronak Sumbaly
UID : 604591897

December 1, 2015

Project Details

The transaction-time history of employees and departments for the **XYZ corporation** are in the stored in the following XML documents: **v-emps.xml** and **v-depts.xml**
The project was to write the following queries using XQuery. The queries and results are provided below.

The project makes uses of a CustomFunctions.xquery file to create external dependent queries. Each query imports this module for usage. The file has been attached at the end of all the query files.

Click the **Query** to go to the file located on Github and click on the **Results** to go to the output of the query.

Query - 1 : Selection and Temporal Projection

Print the salary history of employee "Anneke Preusig"

```
xquery version "1.0";

import module namespace customFunctions = "customFunctionsforXML"
at "file:/Users/RonakSumbaly/Documents/UCLA/Project-1/CustomFunctions.xquery";

declare variable $employee-xml as xs:string := "v-emps.xml";
declare variable $department-xml as xs:string := "v-depts.xml";

(:Query 1 - Selection and Temporal Projection. Print the salary history of employee "Anneke Preusig.":)
element history
{
    for $employee in doc($employee-xml)//employee[firstname="Anneke" and
        lastname="Preusig"]
      return $employee//salary
}
```

Results - Query 1

Query - 2 : Temporal Snapshot

Print the name, salary and department of each employee who, on 1995-01-01, was making more than $80,000

```
xquery version "1.0";

import module namespace customFunctions = "customFunctionsforXML" at
    "file:/Users/RonakSumbaly/Documents/UCLA/Project-1/CustomFunctions.xquery";

declare variable $employee-xml as xs:string := "v-emps.xml";
declare variable $department-xml as xs:string := "v-depts.xml";

declare variable $date := '1995-01-01';

(:Temporal Snapshot. Print the name, salary and department of each employee who, on 1995-01-01,
    was making more than $80,000.:)
element snapshot
{
   for $emp in doc($employee-xml)//employee[@tstart <= $date and $date <= @tend]
 let $salary := $emp/salary[@tstart <= $date and $date <= @tend],
     $deptno := $emp/deptno[@tstart <= $date and $date <= @tend]
 where($salary and $deptno and $salary > 80000 )
 return element

 {node-name($emp)}
 {
   customFunctions:snapshot(($emp/firstname, $emp/lastname,
       customFunctions:deptNumber($deptno), $salary))
 }
}
```

Results - Query 2

Query - 3 : Temporal Slicing

For all departments, print their histories for the period starting on 1994-05-01 and ending 1996-05-06

```
xquery version "1.0";

import module namespace customFunctions = "customFunctionsforXML" at
    "file:/Users/RonakSumbaly/Documents/UCLA/Project-1/CustomFunctions.xquery";

declare variable $employee-xml as xs:string := "v-emps.xml";
declare variable $department-xml as xs:string := "v-depts.xml";

declare variable $tstart := '1994-05-01';
declare variable $tend := '1996-05-06';

(:Temporal Slicing. For all departments, print their histories for the period starting on 1994-05-01 and
    ending 1996-05-06.:)
element slicing
{
    for $dept in doc($department-xml)//department[not( @tstart > $tend or $tstart >=
        @tend)]
        return element

        {node-name($dept)}
         {
          customFunctions:slice($dept, $tstart, $tend),
          customFunctions:sliceAll($dept/*[not( @tstart > $tend or $tstart >=
              @tend)], $tstart, $tend)
         }
}
```

Results - Query 3

Query - 4 : Duration after Coalescing

For each employee, find the longest period (or periods) during which he/she went with no change in salary: for each employee print the period(s), his/her name and the actual salary

```
xquery version "1.0";

import module namespace customFunctions = "customFunctionsforXML" at
    "file:/Users/RonakSumbaly/Documents/Project-1/CustomFunctions.xquery";

declare variable $employee-xml as xs:string := "v-emps.xml";
declare variable $department-xml as xs:string := "v-depts.xml";

(:Duration after Coalescing. For each employee, find the longest period (or periods)
during which he/she went with no change in salary: for each employee print the period(s), his/her name
    and the actual salary.:)

element durationCoalescing
{
    for $emp in doc($employee-xml)//employee
     let $durations := (for $salary in $emp/salary
                        return customFunctions:untilChangedToNow($salary/@tend) -
                            xs:date($salary/@tstart)
                       )
     return element

     {node-name($emp)}

     {
       customFunctions:slice($emp, '1900-01-01', customFunctions:currentDate()),
       customFunctions:untilChangedToAll(($emp/firstname, $emp/lastname)),

       element LongestPeriod {max($durations)},
       for $salary in
           $emp/salary[customFunctions:untilChangedToNow(@tend) -
                       xs:date(@tstart)=max($durations)]
           order by $salary/@tstart, $salary/@tend
        return element
         {node-name($salary)}
         {
          customFunctions:slice($salary, '1900-01-01', customFunctions:currentDate()),
          string($salary)
         }
     }
}
```

Results - Query 4

Query - 5 : Temporal Join

For each employee find the longest consecutive period in which he/she worked in the same department and the same department manager: Print the employee number, his/her department number and name, his/her manager number, and the period.

```
xquery version "1.0";

import module namespace customFunctions = "customFunctionsforXML" at
    "file:/Users/RonakSumbaly/Documents/Project-1/CustomFunctions.xquery";

declare variable $employee-xml as xs:string := "v-emps.xml";
declare variable $department-xml as xs:string := "v-depts.xml";

(:Temporal Join. For each employee find the longest consecuitive period in which he/she worked in the
    same department and the same department manager:
Print the employee number, his/her department number and name, his/her manager number, and the
    period.:)

element temporalJoin
{
    for $emp in doc($employee-xml)//employee
    return element

        {node-name($emp)}
         {
          customFunctions:slice($emp, '1900-01-01', customFunctions:currentDate()),
          customFunctions:untilChangedToAll(($emp/empno,
              $emp/firstname,$emp/lastname)),
          customFunctions:untilChangedToAll(($emp/title, $emp/deptno)),

          element managers
          {
           for $deptno in $emp/deptno, $manager in
               doc($department-xml)//department[deptno=$deptno]
                  /mgrno[@tstart<=$deptno/@tend and $deptno/@tstart<=@tend]
                    let $deptDuration := customFunctions:slice($deptno, '1900-01-01',
                        customFunctions:currentDate())
                    return customFunctions:sliceAll(($manager),
                        string($deptDuration[1]), string($deptDuration[2]))
          }
         }
}
```

Results - Query 5

Query - 6a : Temporal Avg

Print the history of the average salary for (i) the whole company

Explanation

1. Get all distinct dates for the employees (start-dates and end-dates attributes)
2. The distinct dates are sorted in the ascending order and calculate the average salary of the employees for each date.
3. Each average salary is paired up with the next date to form the element with start and end.

```
xquery version "1.0";

import module namespace customFunctions = "customFunctionsforXML" at
    "file:/Users/RonakSumbaly/Documents/UCLA/Project-1/CustomFunctions.xquery";

declare variable $employee-xml as xs:string := "v-emps.xml";
declare variable $department-xml as xs:string := "v-depts.xml";

(:Temporal Avg. Print the history of the average salary for the whole company:)

declare variable $dept-no := doc("v-emps.xml")//deptno;
declare variable $salary := doc("v-emps.xml")//salary;

declare variable $start-dates :=
    for $i in distinct-values($salary/@tstart)
        order by $i
        return xs:date($i);

declare variable $end-dates :=
    for $i in distinct-values($salary/@tend)
        order by $i
        return xs:date($i);

declare variable $combined-dates :=
 for $i in distinct-values(($start-dates, $end-dates))
  order by $i
  return $i;

declare variable $average-salary :=
    for $start at $pos in $combined-dates
        let $x := $salary[@tstart <= $start and $start < @tend]
        let $avg := avg($x)
        order by $start
        return <avg date="{$start}">{xs:float($avg)}</avg>;
```

```
declare variable $max := count($average-salary);

<whole-company>
{
    for $tstart at $pos in $average-salary
      let $tend := $average-salary[$pos + 1]
       where( $pos < $max )
       return <average tstart="{$tstart/@date}" tend="{$tend/@date}">
       {string($average-salary[$pos])}</average>
}
</whole-company>
```

Results - Query 6a

Query - 6b : Temporal Avg

Print the history of the average salary for (ii) each job title.

Explanation

1. Similar to Query 6-a, we get the distinct dates for each job-title and from the sorted dates get the average salary for each job-title.

```
xquery version "1.0";

import module namespace customFunctions = "customFunctionsforXML" at
    "file:/Users/RonakSumbaly/Documents/UCLA/Project-1/CustomFunctions.xquery";

declare variable $employee-xml as xs:string := "v-emps.xml";
declare variable $department-xml as xs:string := "v-depts.xml";

(:Temporal Avg. Print the history of the average salary for each job title:)

declare variable $titles := doc($employee-xml)//title;
declare variable $emps := doc($employee-xml);

declare variable $job-titles :=
    for $i in distinct-values($titles)
        order by $i
        return xs:string($i);

<company>
{
    for $job in $job-titles
        let $jobs := $emps/employees/employee[title=$job]/salary
        let $dates :=
            for $date in distinct-values(($jobs/@tstart, $jobs/@tend))
                order by $date
                return ($date)
        let $max := count($dates)
        return
        <job>
                <job-title>{$job}</job-title>
                {
                    for $tstart at $pos in ($dates)
                        let $y := $jobs[@tstart <= $tstart and $tstart < @tend],
                            $tend := $dates[$pos + 1]
                            where $pos < $max and not($tstart = "9999-12-31")
                        return <average tstart="{$tstart}"
                            tend="{$tend}">{avg($y)}</average>
                }
        </job>
}
</company>
```

Results - Query 6b

Query - 7 : Temporal Max

For the employees in department 'd005', find the maximum of their salaries over time, and print the history of such a maximum.

Explanation

1. Get distinct start and end dates for change in salaries for employees in department = d005.
2. After sorting the distinct dates, get the maximum salary for the given period.
3. Since salaries can remain maximum over multiple time period, perform coalescing to make multiple time periods with same salary into a single entry.
4. To perform coalescing get the distinct-values of salaries and for each value get the minimum start-date and maximum end-date.
5. Print the coalesced output.

```
xquery version "1.0";

import module namespace customFunctions = "customFunctionsforXML" at
    "file:/Users/RonakSumbaly/Documents/UCLA/Project-1/CustomFunctions.xquery";

declare variable $employee-xml as xs:string := "v-emps.xml";
declare variable $department-xml as xs:string := "v-depts.xml";

 (:Temporal Max. For the employees in department d005, find the maximum of their salaries over time,
 and print the history of such a maximum.:)

declare variable $emps := doc($employee-xml)/employees/employee[deptno='d005'];

declare variable $start-dates :=
       for $i in distinct-values($emps/salary/@tstart)
       order by $i
       return xs:date($i);

declare variable $end-dates :=
       for $i in distinct-values($emps/salary/@tend)
       order by $i
       return xs:date($i);

declare variable $combined-dates :=
 for $i in distinct-values(($start-dates, $end-dates))
  order by $i
  return $i;

declare variable $temporal-max :=
    for $start at $pos in $combined-dates
        let $x := $emps/salary[@tstart <= $start and $start < @tend]
        let $max-salary := max($x)
        order by $start
        return <max date="{$start}">{xs:float($max-salary)}</max>;
```

```
declare variable $maxCount := count($temporal-max);

declare variable $max-date :=
    for $tstart at $pos in $temporal-max
       let $tend := $temporal-max[$pos + 1]
        where( $pos < $maxCount )
        return <max tstart="{$tstart/@date}"
            tend="{$tend/@date}">{string($temporal-max[$pos])}</max>;

declare variable $unique-salaries := distinct-values($max-date) ;

declare variable $coalesce :=
    for $v in $unique-salaries
        let $sal := $max-date[text()=$v]
        let $start := '9999-12-31'
        let $end := '1900-12-31'
        let $s :=
            for $x in $sal
               let $start := customFunctions:minDate($x/@tstart,$start)
            return min($start)
        let $e :=
            for $x in $sal
               let $end := customFunctions:maxDate($x/@tend,$end)
            return max($end)
        return <max tstart="{min($s)}" tend="{max($e)}">{string($v)}</max>;


<company>
{
     for $value in $coalesce
        return $value
}
</company>
```

Results - Query 7

CustomFunctions.xquery

```
module namespace customFunctions = "customFunctionsforXML";

(:Return the current date – timestamp:)
declare function customFunctions:currentDate() as xs:string
{
 xs:string(fn:adjust-date-to-timezone(current-date(), ()))
};

(:Convert 'Until Changed' to current timestamp:)
declare function customFunctions:untilChangedToNow($x as xs:string) as xs:date
{
 if( $x="9999-12-31" )
 then xs:date(customFunctions:currentDate())
 else xs:date($x)
};

(:Return the minimum of two dates:)
declare function customFunctions:minDate($x1 as xs:string, $x2 as xs:string) as
    xs:date
{
 if(xs:date($x1)>xs:date($x2))
 then xs:date($x2)
 else xs:date($x1)
};

(:Return Maximum of two dates:)
declare function customFunctions:maxDate($x1 as xs:string, $x2 as xs:string) as
    xs:date
{
    if(xs:date($x1)>xs:date($x2))
    then
        xs:date($x1)
    else
        xs:date($x2)
};

(:Convert all elements from Until Changed to Current-Timestamp:)
declare function customFunctions:untilChangedToAll($elements as element()*) as
    element()*
{
 for $element in $elements
  order by $element/@tstart, $element/@tend
  return element
   {node-name($element)}
   {
    customFunctions:slice($element, '1900-01-01', customFunctions:currentDate()),
    string($element)
   }
};
```

```
(:Return the snapshot of the data:)
declare function customFunctions:snapshot($elements as element()*) as element()*
{
 for $element in $elements
    return element
         {node-name($element)}
      {
         $element/@*[name(.)!="tend" and name(.)!="tstart"],
         data($element)
      }
};

(:Get the department number of each element:)
declare function customFunctions:deptNumber( $deptnos as element()* ) as element()*
{
 for $deptno in $deptnos
  return element
  {node-name($deptno)}
 {
  $deptno/@*,
  attribute deptname {string(doc("v-depts.xml")//department[deptno=$deptno]/deptname)},
  string($deptno)
 }
};

(:Return element which lie between start & end date:)
declare function customFunctions:slice( $element as element(), $start as xs:string,
    $stop as xs:string ) as attribute()*
{
 attribute tstart {customFunctions:maxDate($start,$element/@tstart)},
    attribute tend {customFunctions:minDate($stop,$element/@tend)},
 $element/@*[name(.)!="tend" and name(.)!="tstart"]
};


declare function customFunctions:sliceAll( $elements as element()*,
        $start as xs:string, $stop as xs:string ) as element()*
{
 for $element in $elements
  return
   element {node-name($element)}
   {
    customFunctions:slice($element, $start, $stop),
    string($element)
   }
};
```