

Nikolay Pavlovich Laptev

*Professor: Carlo Zaniolo**Final Project 1*

Website: <http://www.cs.ucla.edu/~nlaptev/cs240a/project1/>

Also note that full results are not included here for the sake of brevity.

Query 1

Temporal Projection: Retrieve the employment history of employee 'Anneke Preusig' (i.e., the departments where she worked and the periods during which she worked there).

Solution:

xquery:

```

declare function local:get_anneke() {
  element employment_history {
    for $s in doc("v-emps.xml")/employees/employee
    [firstname="Anneke" and lastname="Preusig"]
    return ($s/firstname, $s/lastname, $s/deptno)
  }
};

let $result := local:get_anneke()
return element result{$result}

```

Explanation:

The idea here is quite simple, we simply get all employees such that their first name is Anneke and lastname is Presusig. Then we project on firstname, lastname and deptno to get the result we see below. Also note that our result automatically includes the start and end time time of when our employee was working in a particular department.

Results:

```

<?xml version="1.0" encoding="UTF-8"?>
<result>
  <employment_history>
    <firstname tend="9999-12-31" tstart="1990-08-05">Anneke</firstname>
    <lastname tend="9999-12-31" tstart="1990-08-05">Preusig</lastname>
    <deptno tend="9999-12-31" tstart="1990-08-05">d005</deptno>
  </employment_history>

```

</result>

Query 2

Temporal Snapshot: Retrieve the name and the salary of each employees on 1995-05-06, who was making less than \$45000 at that time.

Solution:

xquery:

```
declare function local:get_emps() {
  for $r in doc("v-emps.xml")/employees/employee
  let $n:=$r/salary[xs:date(@tstart) <= xs:date("1995-05-06")
    and xs:date(@tend) >= xs:date("1995-05-06")]
  where xs:integer($n) < xs:integer(45000)
  return (element emp$n, $r/firstname)
};

let $r := local:get_emps()
return (element result{$r})
```

Explanation:

In this query we simply get the salary of every employee who was making less than \$45000 on 1995-05-06. Here we also must not forget to cast all values to their appropriate type (date and integer).

Results:

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
  <emp>
    <salary tend="1995-12-18" tstart="1994-12-18">42318</salary>
    <firstname tend="9999-12-31" tstart="1992-12-18">Patricio</firstname>
  </emp>
  <emp>
    <salary tend="1996-04-01" tstart="1995-04-02">40000</salary>
    <firstname tend="9999-12-31" tstart="1995-04-02">Divier</firstname>
  </emp>
  <emp>
    <salary tend="1995-08-31" tstart="1994-08-31">44191</salary>
    <firstname tend="9999-12-31" tstart="1991-09-01">Karsten</firstname>
  </emp>
```

```

<emp>
  <salary tend="1995-05-21" tstart="1994-05-21">40919</salary>
  <firstname tend="9999-12-31" tstart="1994-05-21">Mingsen</firstname>
</emp>
<emp>
  <salary tend="1995-06-20" tstart="1994-06-20">43485</salary>
  <firstname tend="9999-12-31" tstart="1992-06-20">Lucien</firstname>
</emp>
<emp>
  <salary tend="1996-05-03" tstart="1995-05-04">41291</salary>
  <firstname tend="9999-12-31" tstart="1992-05-04">Basil</firstname>
</emp>
.....
..... (Please look at the website for all results)
.....
</result>

```

Query 3

Temporal Slicing: For all departments, show their manager history in the period starting on 1994-05-06 and ending 1995-05-06.

Solution:

xquery

```

declare function local:overlap($manager, $tstart,$tend) {
  if (xs:date($manager/@tstart) <= xs:date($tstart) and
      xs:date($manager/@tend) >= xs:date($tend))
  then (
    element mgrno
      attribute tstart$tstart,
      attribute tend$tend,
      $manager/text()
  )

  else if (xs:date($manager/@tstart) <= xs:date($tstart) and
           (xs:date($manager/@tend) >= xs:date($tstart) and
            (xs:date($manager/@tend) <= xs:date($tend)))
  then (
    element mgrno
      attribute tstart{$tstart},
      attribute tend{$manager/@tend},

```

```

    $manager/text()
  )

  else if (($manager/xs:date(@tend) >= xs:date($tend)) and
  ($manager/xs:date(@tstart) >= xs:date($tstart)) and
  ($manager/xs:date(@tstart) <= xs:date($tend)))
  then (
    element mgrno
      attribute tstart{$manager/@tstart},
      attribute tend{$tend},
      $manager/text()

  else if (($manager/xs:date(@tstart) >= xs:date($tstart)) and
  ($manager/xs:date(@tend) <= xs:date($tend)))
  then (
    element mgrno {
      attribute tstart{$manager/@tstart},
      attribute tend{$manager/@tend},
      $manager/text()
    }
  )
  else ()
};

declare function local:final_query() {
  for $r in doc("v-depts.xml")/departments/department
  let $k := for $m in $r/mgrno return
    local:overlap($m, '1994-05-06', '1995-05-06')
  where exists($k)
  return element department{$r/deptno, $k}
};
let $final := local:final_query()
return element departments{$final}

```

Explanation:

Here we first get all departments and then for each manager we look if tstart and tend of manager overlap 1994-05-06 and 1995-05-06. here notice statement beginning with let \$k := where we retrieve a list of managers which overlap at once, counter to another possible solution where we could have a for loop outside the “let” statement and retrieve overpping managers one by one which would produce a result for each overlapping manager, rather than a list of managers which is more efficient.

Results

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<results>
  <event>
    <deptno tend="9999-12-31" tstart="1985-01-01">d001</deptno>
    <mgrno tstart="1994-05-06" tend="1995-05-06">110039</mgrno>
  </event>
  <event>
    <deptno tend="9999-12-31" tstart="1985-01-01">d002</deptno>
    <mgrno tstart="1994-05-06" tend="1995-05-06">110114</mgrno>
  </event>
  <event>
    <deptno tend="9999-12-31" tstart="1985-01-01">d003</deptno>
    <mgrno tstart="1994-05-06" tend="1995-05-06">110228</mgrno>
  </event>
  <event>
    <deptno tend="9999-12-31" tstart="1985-01-01">d004</deptno>
    <mgrno tstart="1994-05-06" tend="1995-05-06">110386</mgrno>
  </event>
  <event>
    <deptno tend="9999-12-31" tstart="1985-01-01">d005</deptno>
    <mgrno tstart="1994-05-06" tend="1995-05-06">110567</mgrno>
  </event>
  <event>
    <deptno tend="9999-12-31" tstart="1985-01-01">d006</deptno>
    <mgrno tstart="1994-05-06" tend="1994-06-28">110800</mgrno>
    <mgrno tstart="1994-06-28" tend="1995-05-06">110854</mgrno>
  </event>
  <event>
    <deptno tend="9999-12-31" tstart="1985-01-01">d007</deptno>
    <mgrno tstart="1994-05-06" tend="1995-05-06">111133</mgrno>
  </event>
  <event>
    <deptno tend="9999-12-31" tstart="1985-01-01">d008</deptno>
    <mgrno tstart="1994-05-06" tend="1995-05-06">111534</mgrno>
  </event>
  <event>
    <deptno tend="9999-12-31" tstart="1985-01-01">d009</deptno>
    <mgrno tstart="1994-05-06" tend="1995-05-06">111877</mgrno>
  </event>
</results>
```

Query 4

Duration: For each employee, show the longest period during which he/she went with no change in salary and his/her salary during that time.

Solution:

xquery

```
declare function local:difference($e,$s, $salary) {
  let $n := (xs:date($e) - xs:date($s))
  return element
  difference{element days{xs:integer(fn:replace(xs:string($n),"P—D", ""))
  },element end{ xs:date($e)},element start{ xs:date($s)}, $salary}
};
declare function local:init() {
  for $r in doc("v-emps.xml")/employees/employee
  return (element emp{element diff{
    (for $d in $r/salary
    order by local:difference($d/@tend, $d/@tstart, $d) ascending
    return
    (local:difference($d/@tend, $d/@tstart, $d), $r/empno)}})
};

declare function local:final_call() {
  let $diff := local:init()
  for $d in $diff/diff
  return (element entry{$d/difference[days=
  max($d/difference/days)]/days,$d/difference[days=
  max($d/difference/days)]/salary, $d/empno[1]})
};

let $final := local:final_call()
return element all_entries{$final}
```

Explanation:

In this query we first compute the difference between the start time and end time of salary for every employee. This result is returned as the number of days. Next, we simply order this result for every employee from maximum to minimum, and then return the first entry, which will be the maximum for that specific employee. Note: We took into consideration end dates of "9999". Also if there are more than 1 longest periods when employee's salary was held at a constant rate then we return all of those periods.

Results

```

.....
.... (Please visit webpage for full results)
.....
<entry>
  <days>365</days>
  <days>365</days>
  <salary tend="1999-03-11" tstart="1998-03-11">46671</salary>
  <salary tend="2000-03-10" tstart="1999-03-11">48584</salary>
  <empno tend="2000-07-31" tstart="1998-03-11">10008</empno>
</entry>
<entry>
  <days>2921164</days>
  <salary tend="9999-12-31" tstart="2002-02-14">94409</salary>
  <empno tend="9999-12-31" tstart="1985-02-18">10009</empno>
</entry>
<entry>
  <days>2921247</days>
  <salary tend="9999-12-31" tstart="2001-11-23">80324</salary>
  <empno tend="9999-12-31" tstart="1996-11-24">10010</empno>
</entry>
.....
.... (Please visit webpage for full results)
.....

```

Query 5

Temporal Join. For each employee, show his/her title history and his/her manager history.

xquery:

```

declare function local:overlap($input, $tstart,$tend) {

  if (xs:date($input/@tstart) <= xs:date($tstart) and
      xs:date($input/@tend) >= xs:date($tend))
  then (
    element mgrno {
      attribute tstart{$tstart},
      attribute tend{$tend},
      $input/text()
    })

  else if (xs:date($input/@tstart) <= xs:date($tstart) and
          (xs:date($input/@tend) >= xs:date($tstart) and

```

```

    (xs:date($input/@tend)) <= xs:date($tend))
  then (
    element mgrno {
      attribute tstart{$tstart},
      attribute tend{$input/@tend},
      $input/text()
    }

  else if ((($input/xs:date(@tend) >= xs:date($tend)) and
    ($input/xs:date(@tstart) >= xs:date($tstart)) and
    ($input/xs:date(@tstart) <= xs:date($tend)))
  then (
    element mgrno {
      attribute tstart{$input/@tstart},
      attribute tend{$tend},
      $input/text()
    }
  )
  else if ((($input/xs:date(@tstart) >= xs:date($tstart)) and
    ($input/xs:date(@tend) <= xs:date($tend)))
  then (
    element mgrno {
      attribute tstart{$input/@tstart},
      attribute tend{$input/@tend},
      $input/text()
    }
  )
  else ()
};

declare function local:final() {
  for $r in doc("v-emps.xml")/employees/employee
  for $m in doc("v-depts.xml")/departments
  /department[deptno=$r/deptno]
  let $ov := local:overlap($r/deptno, $m/mgrno/@tstart, $m/mgrno/@tend)

  return (element emp{$r/empno, $r/title, $r/deptno,
    for $e_temp in $r/deptno
    for $m_temp in $m/mgrno
    let $ov :=
      local:overlap($e_temp, $m_temp/@tstart, $m_temp/@tend)
    where not (empty($ov)) return $m_temp})
};

let $f := local:final()

```

return element all_emps{\${f}}

Explanation:

In this query we again employ the overlapping function from the previous question. We first retrieve all employees and then all departments, from v-depts.xml, where those employees had worked. After, we invoke the overlapping function to see during what time did a particular employee have a particular manager. Note that we first return a title of the employee as well as the department in which he worked, and only then do we check for a manager who served in the same time as the employee worked in the departemnt. xquery does not allow for a sequence of tuples to be type-casted at the same time, so it is important to have another for loop in the return statement.

Results:

```
<?xml version="1.0" encoding="UTF-8"?>
<all_emps>
  <emp>
    <empno tend="9999-12-31" tstart="1986-06-26">10001</empno>
    <title tend="9999-12-31" tstart="1986-06-26">Senior Engineer</title>
    <deptno tend="9999-12-31" tstart="1986-06-26">d005</deptno>
    <mgrno tend="1992-04-25" tstart="1985-01-01">110511</mgrno>
    <mgrno tend="9999-12-31" tstart="1992-04-25">110567</mgrno>
  </emp>
  <emp>
    <empno tend="9999-12-31" tstart="1996-08-03">10002</empno>
    <title tend="9999-12-31" tstart="1996-08-03">Staff</title>
    <deptno tend="9999-12-31" tstart="1996-08-03">d007</deptno>
    <mgrno tend="9999-12-31" tstart="1991-03-07">111133</mgrno>
  </emp>
  .....
  ..... (Please visit webpage for full results)
  .....
  <emp>
    <empno tend="9999-12-31" tstart="1996-12-16">10500</empno>
    <title tend="9999-12-31" tstart="1996-12-16">Engineer</title>
    <deptno tend="9999-12-31" tstart="1996-12-16">d005</deptno>
    <mgrno tend="9999-12-31" tstart="1992-04-25">110567</mgrno>
  </emp>
</all_emps>
```

Query 6

Snapshot Count: For each department, show the count of employees who worked in the department in year 1999.

Query:

Solution:

```
declare function local:overlap($manager, $tstart,$tend) {

if (xs:date($manager/@tstart) i= xs:date($tstart) and
  xs:date($manager/@tend) i= xs:date($tend))
then (
  element mgrno {
    attribute tstart{$tstart},
    attribute tend{$tend},
    $manager/text()
  })

  else if (xs:date($manager/@tstart) i= xs:date($tstart) and
    (xs:date($manager/@tend) i= xs:date($tstart) and
    (xs:date($manager/@tend) i= xs:date($tend)))
then (
  element mgrno {
    attribute tstart{$tstart},
    attribute tend{$manager/@tend},
    $manager/text()
  })

  else if (($manager/xs:date(@tend) i= xs:date($tend)) and
    ($manager/xs:date(@tstart) i= xs:date($tstart)) and
    ($manager/xs:date(@tstart) i= xs:date($tend)))
then (
  element mgrno {
    attribute tstart{$manager/@tstart},
    attribute tend{$tend},
    $manager/text()
  }
)
  else if (($manager/xs:date(@tstart) i= xs:date($tstart)) and
    ($manager/xs:date(@tend) i= xs:date($tend)))
then (
  element mgrno {
    attribute tstart{$manager/@tstart},
    attribute tend{$manager/@tend},
```

```

    $manager/text()
  }
)
else ()
};

declare function local:get_all_emps_at($d) {
for $e in doc("v-emps.xml")/employees/employee/deptno[text()=$d/text()]
  where local:overlap($e,"1999-12-31","1999-01-01")
return ($e)
};

declare function local:final() {
for $r in doc("v-depts.xml")/departments/department
  let $emps := local:get_all_emps_at($r/deptno)
  return (element department{$r/deptno, element noemps{count($emps)}})
};

let $f := local:final()
return element departments{$f}

```

Explanation:

In this query we first get all departments which were valid during the year 1999. Then, for each of those departments we get all employees who were also working in that department in the year 1999. We then simply return the count of these employees.

Results:

```

<?xml version="1.0" encoding="UTF-8"?>
<departments>
  <department>
    <deptno tend="9999-12-31" tstart="1985-01-01">d001</deptno>
    <noemps>14</noemps>
  </department>
  <department>
    <deptno tend="9999-12-31" tstart="1985-01-01">d002</deptno>
    <noemps>22</noemps>
  </department>
  <department>
    <deptno tend="9999-12-31" tstart="1985-01-01">d003</deptno>
    <noemps>24</noemps>
  </department>
  <department>

```

```

    <deptno tend="9999-12-31" tstart="1985-01-01">d004</deptno>
    <noemps>109</noemps>
  </department>
  <department>
    <deptno tend="9999-12-31" tstart="1985-01-01">d005</deptno>
    <noemps>112</noemps>
  </department>
  <department>
    <deptno tend="9999-12-31" tstart="1985-01-01">d006</deptno>
    <noemps>30</noemps>
  </department>
  <department>
    <deptno tend="9999-12-31" tstart="1985-01-01">d007</deptno>
    <noemps>83</noemps>
  </department>
  <department>
    <deptno tend="9999-12-31" tstart="1985-01-01">d008</deptno>
    <noemps>32</noemps>
  </department>
  <department>
    <deptno tend="9999-12-31" tstart="1985-01-01">d009</deptno>
    <noemps>31</noemps>
  </department>
</departments>

```

Query 7

Temporal Aggregate. For each department show the history of employee count in the department over time.

Solution: Query:

```

declare function local:init($r) {
  let $table := (element department{($r),
    element time{xs:date($r/@tstart)}, element change{"+"}},
    element department{($r), element time{xs:date($r/@tend)},
    element change{"-"}} )
  return $table
};

declare function local:sort($xml) {
  for $e in $xml
  order by $e/time ascending
  return ($e)
}

```

```

};

declare function local:embrace() {
  for $r in doc("v-emps.xml")/employees/employee/deptno
    let $xml := local:init($r)
  return $xml
};

declare function local:overlap($time, $d) {
  for $r in doc("v-emps.xml")/employees/employee/deptno[text()=$d/text()]
    where (xs:date($r/@tstart) <= xs:date($time) and xs:date($r/@tend) >= xs:date($time))
  return $r
};

declare function local:emp_count($e) {
  if ($e/change/text() = "+")
    then (
      let $num_overlap := count(local:overlap($e/time, $e/deptno))
      return $num_overlap
    )
  else if ($e/change/text() = "-") then (
    let $num_overlap := (count(local:overlap($e/time, $e/deptno))-1)
    return $num_overlap
  )
  else ()
};

declare function local:final() {
  let $big_xml := local:embrace()
  let $sorted_deps := local:sort($big_xml)
  order by $sorted_deps/deptno          for $e in $sorted_deps
    return element dept{($e/deptno), element emp_count{local:emp_count($e)}}
};

let $f := local:final()
return element data{$f}

```

Explanation:

The approach we took with this query is as follows:

- 1) We first “modify” the xml data to make it easier for us to work with.

- a) For each employee we looked at the start date for the time when he/she started working at a particular department. We mark this time with a “+” for that department.
- b) We did the same for tend, but we only mark it with a “-” instead.
This modification will let us know when employee was added and when he was deleted.
- c) We next sort our resulting xml by time in ascending order
- d) After sorting, for each department number (i.e. for each start/end time of each department number) we count how many employees were working at that time.
- e) We do this by counting the number of employees who overlap a particular department (at tstart and tend). Note: We must subtract 1 employee when computing count of employees at tend because of overcounting by one.
- f) The result is a list of departments’ histories, where each entry in the resulting XML represents an addition or deletion of an employee.

Results:

```

<?xml version="1.0" encoding="UTF-8"?>
<data>
  <dept>
    <deptno tend="9999-12-31" tstart="1986-01-16">d001</deptno>
    <emp_count>1</emp_count>
  </dept>
  <dept>
    <deptno tend="1994-08-15" tstart="1987-07-25">d001</deptno>
    <emp_count>2</emp_count>
  </dept>
  <dept>
    <deptno tend="9999-12-31" tstart="1988-03-30">d001</deptno>
    <emp_count>3</emp_count>
  </dept>
  <dept>
    <deptno tend="9999-12-31" tstart="1988-04-25">d001</deptno>
    <emp_count>4</emp_count>
  </dept>
  <dept>
    <deptno tend="1995-05-24" tstart="1988-09-24">d001</deptno>
    <emp_count>5</emp_count>
  </dept>
  <dept>
    <deptno tend="9999-12-31" tstart="1989-08-24">d001</deptno>
    <emp_count>6</emp_count>
  </dept>
  <dept>
    <deptno tend="9999-12-31" tstart="1990-01-17">d001</deptno>
    <emp_count>7</emp_count>
  </dept>

```

</dept>

....

.... (Please visit webpage for full results)

....

Extra Credit

Design and implement attractive stylesheets to visualize the results of your temporal queries.

Solution:

For extra credit I decided to implement a style sheet introduced first by the similie project at MIT. This is a very nice API which takes an XML file, parses it and outputs a nice looking result. My job, in order to get this to work, was to understand how to use the API, modify the xquery in order to output the XML in an appropriate format suitable for the API and modify the javascript which the API was using in order to get the results I wanted. Please refer to the website mentioned at the beginning of this document for the live data.