Exam XML & Databases [211096]

Wednesday 9 April 2008; 09:00 – 12:30 h.

Allowed on exam: slides, reader, print outs, notes on paper

Not allowed on exam: electronic devices

About the exam

There are 10 questions. For each question, an indication of the associated paper from the reader is given, if appropriate. Moreover, the number of points for each question is mentioned. The points add up to a total of 90 points. You receive 10 points bonus for showing up at the exam. The final grade is determined by dividing the total score by 10.

1 XML Data

The data below concerns a helpdesk of a software development company. Each "call" concerns one problem sent to the helpdesk. A call has a reference number (ref) and contains a description of the problem. In the description, module names (module) and references to operating systems (os) are annotated with an identifier (id) to allow for easy reporting on the call list.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE problems [
  <!ELEMENT problems (call*)>
  <!ELEMENT call (#PCDATA | module | os)*>
  <!ELEMENT module (#PCDATA)>
  <!ELEMENT os (#PCDATA)>
  <!ATTLIST call ref CDATA #REQUIRED>
  <!ATTLIST module id CDATA #REQUIRED>
  <!ATTLIST os id CDATA #REQUIRED>
] >

<problems>
   <call ref="7735">
      Module <module id="235">ABC</module> does not compile on
      <os id="2">Vista</os>
   </call>
   <call ref="7736">
      Application crashes on <os id="1">Linux</os> with SuSE distribution
      older than 8.0
   </call>
</problems>
```

Question 1 *(12 points)*
Give XQuery queries that are as short as possible for the questions below. Note that the queries need to properly answer the questions for any document that is valid according to the above DTD, i.e., not only for the given document.

a) *Give all modules for which there was a call.*
b) *For each module, give the number of calls and a list of operating systems for which the module has caused a problem.*
c) *Select all calls that contain the word "SuSE", but no 'os'-element with id 1. You are not allowed to use "and" and "or".*

Question 2 *(6 points)*
Given the XQuery update

```
let $doc := doc("foo.xml")
for $i in (1,2,3)
return (do insert <b>{$i}</b> as last into $doc
       ,do replace $doc/b with <c>{$i}</c>)
```

what does foo.xml contain after the update when it initially contained `<a></a>`*?*

Question 3 *(4 points)*

a) *Can a document be well-formed without being valid? Explain your answer.*
b) *Can a document be valid without being well-formed? Explain your answer.*

Question 4 *(10 points)*
[Paper F/G] Tree representation and XPath Accelerator storage structure

a) *Draw the tree representation of the above document. Include in your drawing the pre-order and post-order ranks of all nodes.*
b) *Draw the pre/post plane of this document.*

Question 5 *(10 points)*
Given the XPath query `'/descendant::os[./text() = "Linux"]/parent::call'`

a) *Give the SQL-query corresponding to the above XPath query if you follow the XPath evaluation scheme of paper F.*
b) *In XPath you are also allowed to write* `'[. = "Linux"]'` *for the predicate in the above query. For the given document, this gives the same answer. Do you always get the same answer, also if we would have a different DTD? If no, give an example XML-fragment which would give a different answer for both variants; if yes, explain why you always would get the same answer*

Question 6 *(12 points)*
[Paper U] Given the prefix encoding scheme of Table 1

a) *What is ORDPATH identifier for compressed bitstring '0100011111001'?*
b) *Given ORDPATH identifier 3.2.1.-4.2.5. Can you determine the level of the node based on its ORDPATH identifier alone? If yes, what is it? If no, explain why not.*
c) *Does every compressed bitstring specify an ORDPATH identifier?*

Bitstring	L_i	O_i value range
0001	2	[-5,-2]
001	1	[-1,-0]
01	0	[1,1]
10	1	[2,3]
110	2	[4,7]

Table 1: Prefix encoding scheme with lengths L_i and value ranges O_i

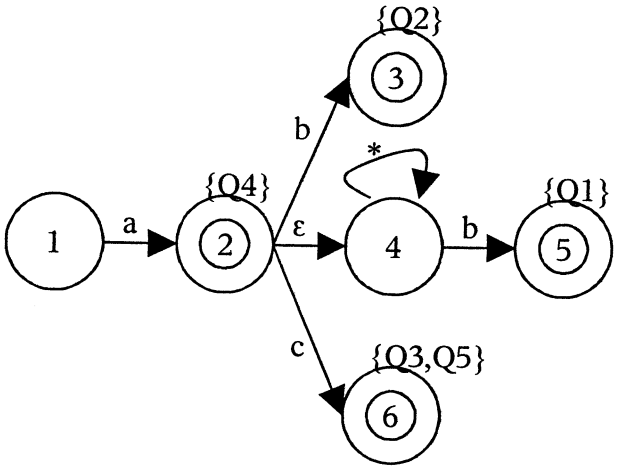


Figure 1. Example YFilter FSM

Question 7 *(14 points)*
[Paper O] Given the Example YFilter Finite State Machine (FSM) of Figure 1.

a) What are the original queries that gave rise to this FSM?
b) Suppose the input stream of elements is <a><c></c>. Give the state of the stack after each opening or closing element.
c) Which notifications are being sent out to users for this stream of elements?

2 A product database

This example is based on a simple business activity from the JavaWorld.com java j2ee documentation: a simple product database. The basic entities in this example are: Products, Categories, and Users. A user can input new products. New products are entered daily. Data is almost never updated or deleted. Consider the relational database design on the right for this case.

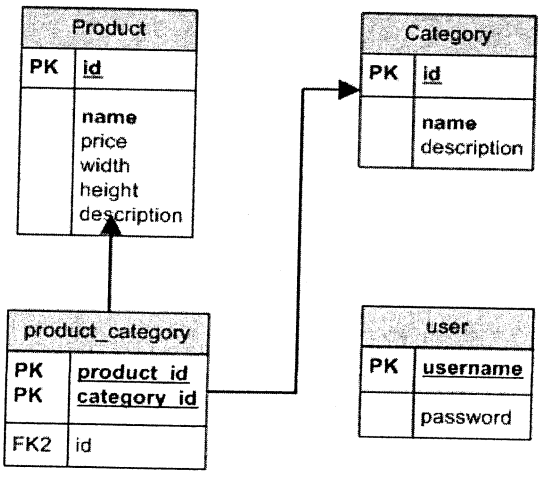


Figure 2. The product database

Question 8 *(6 points)*
[Paper A] Give an SQL/XML query that lists per category the average price and the names of products in that category.

Question 9 *(6 points)*
[Paper A] Give the most strict DTD of the SQL/XML standard mapping assuming "absent" behaviour of NULL values. Elements of type (#PCDATA) do not have to be listed.

Question 10 *(10 points)*
The company that records products using the relational database above will redesign its application using an XML database solution. Answer the following questions and motivate your answer (this question will be judged based on the arguments presented).

a) *Which of the XML database storage and query methods that were addressed in the course is most appropriate?*
b) *What XML benchmark would be most appropriate for evaluating the system?*

– SOLUTIONS –

Exam XML & Databases [211096]

Wednesday 13 April 2005; 09:00 – 12:30
Allowed on exam: slides, reader, print outs, notes on paper
Not allowed on exam: electronic devices

1 The time administration system

Many professionals are required to keep a time administration, i.e., how many hours they worked on a number of projects. The XML document given below is used by a time administration application. It is structured as follows. First there are a descriptions of projects, followed by a time log with one entry for each uninterrupted piece of work. The entry is linked to a project using the project identifier 'projid'. The 'from' and 'to'-times are stored as the number of seconds elapsed from some fixed point in the past. The document should be self explanatory, but if something is unclear to you, please ask for additional explanation.

```
<?xml version="1.0" encoding="iso-8859-1"?>
  <!DOCTYPE timeadmin [
    <!ELEMENT timeadmin (project+ , timelog) >
    <!ELEMENT amount (#PCDATA) >
    <!ELEMENT anotherunusedelement (#PCDATA) >
    <!ELEMENT project (name , description , budget)+ >
    <!ATTLIST project projid ID #REQUIRED >
    <!ELEMENT budget (#PCDATA) >
    <!ATTLIST budget per (week | month | year) #IMPLIED >
    <!ELEMENT timelog (entry)* >
    <!ELEMENT entry (#PCDATA) >
    <!ATTLIST entry project IDREF #REQUIRED
                    from CDATA #REQUIRED
                    to CDATA #REQUIRED >
    <!ELEMENT name (#PCDATA) >
    <!ELEMENT description (#PCDATA) >
  ]>
<timeadmin>
  <project projid="MN">
    <name>Multimedian</name>
    <description>Multimedia for ambient intelligence</description>
    <budget per="week">20</budget>
  </project>
  <project projid="PF">
    <name>Pathfinder</name>
    <description>XQuery engine on top of relational DBMS</description>
    <budget per="month">40</budget>
  </project>
  <timelog>
    <entry project="PF" from="3465" to="3468">e-mail</entry>
    <entry project="MN" from="3468" to="3500">presentation</entry>
    <entry project="PF" from="3500" to="3811">paper</entry>
  </timelog>
</timeadmin>
```

Question 1 *Is the document "well-formed"? Is the document "valid"? Explain your answer.*

Answer "Well-formed" means correct XML, so opening and closing tags match, etc. "Valid" means that it conforms to the DTD. There are two element declarations in the DTD which are not used in the document. This is perfectly normal.

Question 2 *Give XQuery queries that are as short as possible for the following:*

a) The name of project with ID "PF"

Answer
```
//project[@projid="PF"]/name
```

b) All entries for which the 'from'-attribute is not the same as the 'to' attribute of the previous entry.

Answer
```
//entry[not(@from = ./preceding-sibling::entry/@to)]
```

c) An overview of per project: the project id, its name, its budget, and the total number of hours worked on that project.

Answer
```
let $doc := document("...")
for $p in $doc//project
return <project id={$p/projid}>
         {$p/name}
         {$p/budget}
         <total>{sum(for $e in //entry[@project=$p/projid]
                     return ($e/@to - $e/@from)/3600)}
         </total>
       </project>
```

Many people had difficulties with computing the sum. The sum has to be evaluated on a sequence of numbers, so putting the sum in the return-clause of the inner for does not make sense: you are computing the sum of only one number!

The following questions are related to papers E, F and G.

Question 3 *Draw the tree representation of the above document. Include in your drawing the pre-order and post-order ranks of all nodes.*

Answer Almost everybody did this correctly, so I'll save time to draw the entire tree. Note that attributes and text nodes are separate nodes, so they will receive pre-order and post-order ranks too. Furthermore, the attributes of an element come before the children of that element, so it is essential to number them in the right order.

Question 4 *Give the SQL-query corresponding to the XPath query '//entry[@project="PF"]' if you follow the XPath evaluation scheme of paper F.*

a) In (ordered) <u>select</u> mode

Answer
```
SELECT DISTINCT e.pre
FROM accel as root, accel as e, accel as a
WHERE root.pre = 0        // or some other way of locating the root
  AND e.pre>r.pre AND e.post<r.post // e is descendant of root
  AND e.kind=elem                   // do not select attributes
  AND e.name="entry"
  AND a.pre>e.pre AND a.post<e.post
  AND a.level=e.level+1                  // a is a 'child' of e
```

```
        AND a.name="project"
        AND a.kind=att                          // a is an attribute
      ORDER BY e.pre
```

b) In reconstruct mode

Answer The above query only selects the entry *nodes*. To produce an XML result, the entire element has to be produced, i.e., the subtree below the element with all its descendants and attributes. So, "reconstruct mode" means the same query with additionally: "`accel as s`" in the from clause and "`s.pre>=e.pre and s.post=<e.post`" in the where-clause. Furthermore, we select distinct "`s.pre`" instead of "`e.pre`".

The following question is related to paper H.

Question 5 *Suppose you have the following query ($doc refers to the above document)*

```
for $p in distinct-values($doc//entry/@project)
for $b in $doc//project[@projid=$p]/budget
return <budget project="{$p}">
         {if ($b/@per = "week") $b*52
          else if ($b/@per = "month") $b*12
               else $b
         }
       </budget>
```

a) Is the 'distinct-values' necessary? Explain your answer.

Answer No, it is not necessary, because an XPath expression never returns any duplicates.

b) How many scopes are in this example? Explain your answer.

Answer Strictly speaking, there are three scopes: the outer scope and a scope for each 'for'. I did not evaluate it as a mistake if you omitted the outer scope, so the answer "two" is also correct provided that the correct explanation was given.

c) Give the table representing variable $p in the innermost scope.

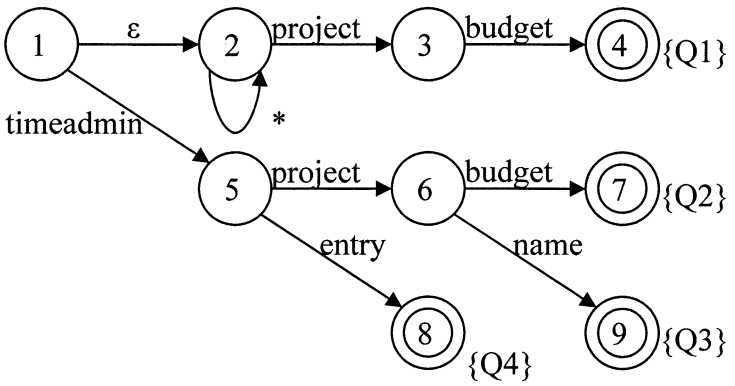
Answer Variable representations are tables with three attributes: iter, pos, and item. Since there is only one budget "$b" per project "$p", there are only two iterations for the return-clause (one for $p="PF" and one for $p="MN"), i.e., we have a table with only two rows:

iter	pos	Item
1	1	"PF"
2	1	"MN"

The following questions are related to paper O on Streaming XML.

Question 6 *Given the XPath queries Q1=//project/budget, Q2=/timeadmin/project/budget, Q3=/timeadmin/project/name, and Q4=/timeadmin/entry.*

a) Draw the YFilter NFA for these four queries.



Answer

b) Give the state of the running stack after input " <timeadmin> <a> <project> ".

Answer

2 3 6
2 5
1 2

The following question is related to paper N on data integration.

Question 7 *I have a data integration system already trained for several sources from previous data integration projects. Suppose I get hold of a new base learner, for example one that can recognize dates. Do I need to retrain the other base learners? Do I need to retrain the meta learner? Explain your answer.*

Answer It is not necessary to retrain the other base learners. They only depend on the data and nothing changed in the data. The meta learner does need to be retrained, because it needs to adapt its weights to include the new base learner.

2 The student registration system

Consider the following relational schema:

```
CREATE TABLE Courses (
  CourseCode INTEGER,
  Study CHAR(3) DEFAULT 'CS',
  CourseName VARCHAR NOT NULL,
  Description VARCHAR,
  PRIMARY KEY (CourseCode),
  CHECK (Study IN ('CS','TEL','BIT'))
);

CREATE TABLE Enrollment (
  StudentNumber INTEGER,
  CourseCode INTEGER,
  Grade INTEGER NOT NULL,
  FOREIGN KEY (CourseCode) REFERENCES Courses (CourseCode)
);
```

And consider the following DTD

```
<!DOCTYPE StudentRegistration [
```

```
  <!ELEMENT StudentRegistration (Course+) >
  <!ELEMENT Course (Name, Description?, Enrollment*) >
  <!ATTLIST Course CourseCode ID #REQUIRED
                   Study (CS|TEL|BIT) "CS" >
  <!ELEMENT Enrollment (StudentNumber, Grade) >
  <!ELEMENT Name (#PCDATA)>
  <!ELEMENT Description (#PCDATA) >
  <!ELEMENT StudentNumber (#PCDATA) >
  <!ELEMENT Grade (#PCDATA) >
]>
```

Question 8 *Give an SQL/XML query that produces the course code, name and description of courses of which none of the enrolled students has a grade that is less than 6. The XML data should conform as much as possible to the DTD above.*

Answer A 'normal' SQL query for this question would look like this:

```
SELECT C.CourseCode, C.Study, C.CourseName, C.Description
FROM Courses C
WHERE NOT EXISTS
  (SELECT  *
   FROM Enrollment E
   WHERE E.CourseCode = C.CourseCode
     AND E.Grade < 6)
```

To produce XML data conforming to the DTD, we only have to change the 'select' to (note that we do not produce the root node):

```
SELECT XMLELEMENT(NAME "Course",
          XMLATTRIBUTES(C.CourseCode, C.Study),
          XMLELEMENT(NAME "Name", C.CourseName),
          XMLELEMENT(NAME "Description", C.Description)
        ) AS Result
```

Question 9 *Give an SQL/XML query that produces the mapping of all data in the database (assume that each course has at least one student enrolled to it). Again the XML data should conform as much as possible to the DTD.*

Answer

```
SELECT XMLELEMENT(NAME "Course",
         XMLATTRIBUTES(C.CourseCode, C.Study),
         XMLELEMENT(NAME "Name", C.CourseName),
         XMLELEMENT(NAME "Description", C.Description),
         XMLAGG(NAME "Enrollment",
           XMLELEMENT(NAME "StudentNumber", E.StudentNumber),
           XMLELEMENT(NAME "Grade", E.Grade)
         )
       ) AS StudentRegistration
FROM Courses C, Enrollment E
WHERE C.CourseCode = E.CourseCode
GROUP BY C.CourseCode, C.Study, C.CourseName, C.Description
```

The following question is related to paper T on result construction in Natix. Consider the following YATL query

```
MAKE
  Result
     [*($s) Student
       [StudentNumber[$s],
         Courses[
           *Name[$n]]]]
MATCH "studentregistration.xml" WITH
  StudentRegistration
    [*Course
      [Name[$n],
        *Enrollment[StudentNumber[$s]]]]
```

Question 10 *Give the XQuery statement that produces the exact same results as the YATL query*

Answer The query reconstructs the original document (that lists for each course the participating students) such that it lists for each student the courses he/she participates in, for example:

```
<Result>
  { for $s in distinct-values (document("studentreg.xml")//StudentNumber)
    return
    <Student>
      { $s }
      <Courses>
        { for $c in doc("studentregistration.xml")//Course
          where $c/Enrollment/StudentNumber = $s
          return $c/Name
        }
      </Courses>
    </Student>
  }
</Result>
```

Question 11 *Give a construction plan for the query using the Natix result construction operators.*

Answer Let [$s, $n] be a relation of all studentnumber—coursname combinations, then a possible contruction plan is (using a slightly different notation than the paper):

```
BA-Map before:, each:, after:</Result> (
  GroupApply last:</Student> (
    BA-Map before:<Courses>, each:<Name>$n</Name>, after:</Courses> (
      Groupify groupby:$s, first:<Student><StudentNumber>$s</StudentNumber> (
        BA-Map before:<Result>, each:, after: ( [$s, $n] )
      )
    )
  )
)
```

The following question is related to paper I on information retrieval queries. Consider the following XPath query:

```
//Course[contains(Name, "XML") and
 contains(Name, "DB")]//Description[contains(.,"easy") or contains(.,"relax")]
```

Question 12 *Give an expression of the query using Burkowski's algebra for contiguous text extents.*

> **Answer** A possible expression is the following:
>
> ```
> { <Description> SW {"easy", "relax"} } SN
>    { <Course> SW { <Name> SW { "DB" } } SW { <Name> SW { "XML" } } }
> ```

The following question is related to papers K and M on benchmarking.

Question 13 *Question 11 and Question 12 give a specific view on two different challenges of XML querying. Does the XMark benchmark contain queries that specifically test these two challenges? If yes, which queries, if not, what kind of queries do you miss?*

> **Answer** Paper K lists 10 challenges of XML databases that are addressed by XMark, and paper M groups queries by similar (but slightly different) query "concepts to be tested". Considering that the query in Question 11 needs to reconstruct the entire database, it can be argued that the main challenge is *reconstruction* or *construction of complex results*. This is tested in XMark Query 10 and Query 13. Question 12 contains an example of a *full text search* query, which is tested in XMark Query 14.
>
> NB Considering the XQuery statement in Question 10, it might be argued that the possible challenge is *joins on values* (XMark Query 11 and 12).

The following question is related to paper Q on XPath rewriting. Consider the following XPath query:

```
//Courses[@CourseCode="c211096" and preceding-sibling::Courses/@CourseCode="c211086"]
```

Question 14 *Convert the reverse axis step in the following query to a forward axis step using the approach by Olteanu et al. Explain your answer by stating which equivalence rules you used.*

> **Answer** The query selects all courses that have the course code 211096 and that also have a preceeding (sibling) course with course code 211086. The query has one reverse axis: preceding-sibling in the predicate. This can be solved by Equation 4 from paper Q, except that the query has an additional condition `@CourseCode="c211096"`. In order to be able to apply Equation 4, we first rewrite the query to:
>
> ```
> //Courses[preceding-sibling::Courses/@CourseCode="c211086"][@CourseCode="c211096"]
> ```
>
> then, applying Equation 4 gives:
>
> ```
> //Courses[//Courses[@CourseCode="c211086"]/following-sibling::node() ==
>     self::node()][@CourseCode="c211096"]
> ```
>
> It is not possible to directly apply Equation 31, because of the additional step: `@CourseCode="c211086"`. However, from the semantics of the original query, it is easy to infer that a simpler version of the query is:
>
> ```
> //Courses[@CourseCode="c211086"]/following-sibling::Courses[@CourseCode="c211096"]
> ```

The following question is related to paper R on XSLT rewriting. Consider the XSLT program below.

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

```
<xsl:template match="*">
  <xsl:apply-templates select="*"/>
</xsl:template>

<xsl:template match="Course">
  <xsl:if test="@CourseCode='c211096'">
    <xsl:value-of select="Name"/>
  </xsl:if>
</xsl:template>

</xsl:stylesheet>
```

Question 15 *Generate an equivalent SQL query. Explain your answer by giving the internal representation (IR), and the QTree.*

Answer Assume the releational database defined at the top of Section 2 and the XML mapping defined by the DTD. The XSLT program will produce the name of the course with course code 211096, so (as is done in the introduction of paper Q) it is easy to see that the equivalent SQL query has to be:

```
SELECT CourseName FROM Courses WHERE CourseCode = 211096
```

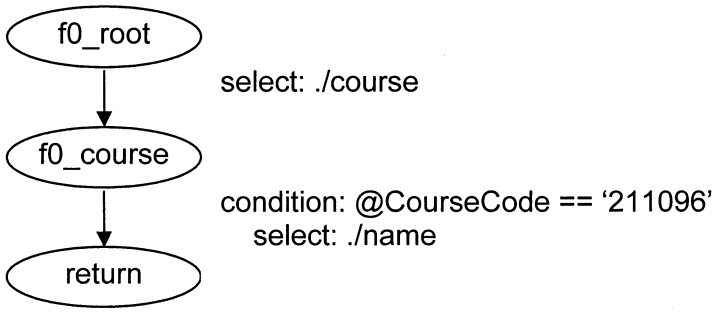
The approach to arrive at this query works as follows. The internal representation results at first in two functions:

```
f0($default) = f0($default/*)
f0_course($default) =
       if($default/@CourseCode == '211096') return ($default/Name)
```

for which all wildcard (*) are instantiated using the DTD. Since the Course elements are directly under the root, this results in:

```
f0_root($default) = f0_course($default/course)
f0_course($default) =
       if($default/@CourseCode == '211096') return ($default/Name)
```

The internal representation (the two steps above can be omitted), using the notation of the paper is:



The XSLT program does not contain parameters, so there is just one simple query tree, which can be transformed to the SQL query above using the mapping from the XML DTD to the relational tables: