201300180 Data & Information – Test 3 (1.5 hours)

5 June 2015, 13:45 – 15:15

Please note:

- ***Please answer questions 1 and 2 each on a separate sheet of paper***
  (Not on the back side of the previous question, the questions will be distributed to different person for grading).
- You can give your answers in Dutch or English.
- Reference materials are given in the appendices, therefore you are not allowed to bring any study materials to the test
- **WAP-students** need only answer question 1.

Grade = #points/10

Question 1 (XQuery) (50 points)

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.

```
<!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>
```

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.

Tip: See Appendix 1 for an informal syntax of XPath and XQuery.

a) Give unique id's of all modules for which there was a call.

b) For each os, give the number of calls that refer to that os.

c) Give all modules for which there was a call related to operating system "Vista".

d) Give all calls that contain the word "SuSE", but no 'os'-element with id 1.

e) Call 7735 mentions module "ABC". Give the call with the next mention of this module.

f) Produce an XML document that contains all calls per module.

g) Delete all calls that mention operating system "Vista" and for which the next call mentions operating system "Linux".

Answer these questions about XML, XPath, XQuery, and RESTXQ

h) Can an XML document be well-formed but not valid?

i) Both calls contain text with embedded elements. What is the technical term for this?

j) Does the query "//module" produce a correct XML-document? If so, explain why. If not, adapt the query such that it does.

Question 2 (Database transactions) (50 points)

Tip: See Appendix 2 for an informal syntax of SQL.
Tip: See Appendix 3 for a table (the same table as in the slides of lecture DB-5) which summarizes the SQL isolation levels, the anomalies they prevent, and the locking implementation of those isolation levels.

a) The table definition below contains a primary key and a foreign key declaration. What does this mean in terms of constraints and indices? In other words, which constraints and which indices are created?
(you don't need to give the SQL for these indices or constraints, just in words which ones)

```
CREATE TABLE person (
  id INT,
  name TEXT,
  bestfriend INT,
  PRIMARY KEY (id),
  FOREIGN KEY (bestfriend) REFERENCES person(id)
)
```

b) Give the SQL-statement for creating a view that contains only the id and name of persons.

c) Give a reason why you would define such a view? In other words, what is a realistic purpose for such a view?

d) Given the SQL statement below. Which locks are obtained for this statement?

```
UPDATE account
SET amount=amount + (SELECT amount FROM account WHERE name="A")
WHERE name="B"
```

e) Given the schedule below. Is it serializable or not? Explain why.

$r_1(x)\ r_1(y)\ r_2(x)\ w_2(x)\ r_3(y)\ w_1(x)\ w_1(y)\ w_3(y)\ c_1\ c_2\ c_3$

f) If the schedule above were to be allowed to execute on a database in the order shown here, which anomalies would occur?

g) At which isolation level(s) would these anomalies be prevented? Explain for each of these isolation level(s) how the anomalies are prevented in terms of which operation(s) in the given schedule are delayed by the locking implementation of that isolation level.

Appendix 1: Informal syntax for XQuery

In the informal syntax, we use the following notations

- A | B to indicate a choice between A and B
- [A] to indicate that A is optional
- A* to indicate that A appears 0 or more times
- A+ to indicate that A appears 1 or more times
- 'A' to indicate that the symbol A is literally that symbol

We are not precise in punctuation in the syntax, but this is irrelevant in this exam anyway.

XPath

path: step+
step: /axis::nodetest predicate*
predicate: '[' expression ']'
axis: `child, descendant, descendant-or-self, parent, ancestor, ancestor-or-self, preceding, preceding-sibling, following, following-sibling, attribute`
nodetest: '*' | name | `node()` | `element()` | `text()`
expression: constant | path | expression binop expression
binop: '=' | '!=' | '<' | '>' | '<=' | '>=' | `and` | `or` | '+' | '-' | '*' | div | …

XQuery

flower: `FOR` $var `IN` expression (`LET` $var := expression)* [`WHERE` expression] [`ORDER BY` …]
`RETURN` expression
expression: path | flower | XML-fragment
In an XML-fragment '{' expression '}' is replaced by the result of evaluating expression.

Commonly used functions: `count`(…), `sum`(…), `distinct-values`(…), `not`(…), etc.

Appendix 2: Informal syntax of SQL

SQL

createtable: `CREATE TABLE` tablename '(' columndef+ constraint* ')'
createview: `CREATE VIEW` viewname `AS` query
query: `SELECT` (column [`AS` colname])+ `FROM` (tablename [`AS` colname])+ `WHERE` condition
[`GROUP BY` column+] [`ORDER BY` column+]

columndef: colname type [`NOT NULL`] [`UNIQUE`] [`PRIMARY KEY`] [`REFERENCES` tablename (colname+)]
constraint: `PRIMARY KEY` (colname, ...)
| `FOREIGN KEY` (colname, ...) `REFERENCES` tablename(colname, ...) | `CHECK` (condition)
column: [tablename '.'] colname | '*'

Examples of condition:
column = value [(`OR` | `AND`) [`NOT`] column <> value]
| column `IS` [`NOT`] `NULL`
| column [`NOT`] `IN` (value, ...)
...

Appendix 3: Isolation levels

isolation level	**prohibited anomalies**	**definition anomaly**	**implementation** prohibiting the anomalies
READ UNCOMMITTED	dirty write	… $w_2(x)$ … $w_1(x)$ …	only write locks
READ COMMITTED	dirty read	… $w_2(x)$ … $r_1(x)$ …	short-term read locks
REPEATABLE READ	non-repeatable read	$r_1(x)$ … $w_2(x)$ … c_2 … $r_1(x)$ ... c_1	Long-term read locks
SERIALIZABLE	phantom	$r_1(P)$ or $w_1(P)$ … $w_2(y \text{ in } P)$	Long-term predicate locks