201300180 Data & Information – Test 3 (1.5 hours)

3 June 2015, 13:45 – 15:15

Please note:

- ***Please answer questions 1, 2 and 3 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 questions 1 and 3.

Grade = #points/10

Question 1 (XQuery) (35 points)

The data below concerns a book review application on the web. The reviews submitted on the website are stored as XML fragments "entry". The community around the website has a procedure for checking the reviews. If an entry is checked, the attribute "validated" is set to "1"; otherwise it is "0". Reviews have a title and the review text. In the review text, certain technical terms are annotated with a "term" element to ease search for them.

```
<!DOCTYPE reviews [
<!ELEMENT reviews (entry)*>
<!ELEMENT entry (title , review*)>
<!ATTLIST entry validated CDATA #REQUIRED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT term (#PCDATA)>
<!ELEMENT review ANY>
]>
<reviews>
  <entry validated="1">
    <title>Data on the Web</title>
    <review>A very good discussion of <term>semi-structured
database</term> systems and <term>XML</term>.</review>
  </entry>
  <entry validated="0">
    <title>Advanced Programming in the Unix environment</title>
    <review>A clear discussion of <term>UNIX</term>
programming.</review>
    <review>Part on <term>TCP/IP</term> is especially good</review>
  </entry>
  <entry validated="1">
    <title>TCP/IP Illustrated</title>
    <review>One of the best books on <term>TCP/IP</term>.</review>
  </entry>
</reviews>
```

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 example document.

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

a) Give all titles of entries that were validated.

b) Give all entries that are about the term "TCP/IP".

c) Give the count of how many entries have more than one review.

d) For each unique term, give the number of reviews with that term.

e) Give the entry that comes after the one with title "Data on the Web"

f) Produce an XML document with root element "index", its children are all terms, i.e., "term" elements with each an attribute "name" containing the term. The term elements have as children the titles of all entries with that term.

g) Insert a review with some made-up text to the entry about "TCP/IP Illustrated"

h) Set attribute "validated" of entry "Advanced Programming in the Unix environment" to "1".

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

i) Can an XML document be valid but not well-formed? Explain your answer.

j) Is any XQuery function that produces (X)HTML, a RESTXQ function? Explain your answer.

Question 2 (Database transactions) (35 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 definitions below contain primary and foreign key declarations. Suppose the database contains several entries (including one with eid 6234) and for each entry there exist one or more reviews. What happens if you issue the command "DELETE FROM entry WHERE eid=6234"? Explain your answer.

```
CREATE TABLE entry (
 eid INT,
 title TEXT,
 validated CHAR,
 PRIMARY KEY (id)
)
```

```
CREATE TABLE review (
        rid INT,
        txt TEXT,
        eid INT,
        PRIMARY KEY (rid),
        FOREIGN KEY (eid) REFERENCES entry(eid)
)
```

b) Give the CREATE VIEW statement(s) for providing a certain set of users only access to validated entries including only the reviews of validated entries.

c) Given the SQL statement below. How many write locks are obtained for this statement?

```
UPDATE entry
SET validated = '1'
WHERE eid IN (SELECT eid FROM review)
```

d) Given the schedule below. Which pairs of operations in the schedule are conflicting?

$w_1(x)\ r_2(y)\ w_1(y)\ r_1(z)\ r_3(x)\ r_3(y)\ r_3(z)$

e) Is the schedule above serializable or not? Explain why.

f) Suppose your Java-program connects to the database and performs the following transaction: it reads one particular entry and counts the number of reviews about it, prepares some HTML-template for the entry, and then reads the reviews from the database and adds them to the template. What is the lowest isolation level for this transaction to run guaranteed correctly? Explain your answer?

Question 3 (REST) (30 points)

Tip: See Appendix 4 for supporting information about REST.

a) How can the programmer of a web service implemented with Jersey indicate which method of which Java class has to be called in order to handle an HTTP request message, depending on the URI, HTTP method and supported encodings indicated in this HTTP request message? Illustrate your answer with a simple example.

b) Suppose a web service is available that manages a collection of profiles in a social media application. Assume that the REST URL conventions have been properly followed so that

- `http://anyhost/profiles` represents the whole collection of profiles
- `http://anyhost/profiles/`*`id`* represents a message with profile *id*

Explain how a client can *create a profile* (Create REST action) by describing the HTTP messages (request/response) that are exchanged to do this and the contents of these messages (HTTP method, URL and message body).

c) Explain why we have to use Tomcat to implement RESTful services in Java with JAX-RS. Draw a schema that illustrates your answer.

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 | let | groupby | where | order by)+ RETURN expr
for: FOR $var IN expr
let: LET $var := expr
groupby: GROUP BY expr
where: WHERE expr
orderby: ORDER BY expr
expr: path | flower | XML-fragment | copy-expr | update-expr
update-expr: DO INSERT expr [AS FIRST | AS LAST] INTO expr | DO DELETE expr
| DO INSERT expr (AFTER | BEFORE) expr 2| DO REPLACE expr WITH expr | DO RENAME expr AS expr
copy-expr: COPY ($var := expr)* MODIFY expr RETURN expr
In an XML-fragment '{' expression '}' is replaced by the result of evaluating expression.
Commonly used functions: count(...), sum(...), distinct-values(...), not(...), contains(...,...) 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

Appendix 4: Supporting information REST

JAX-RS annotations

```
@Path("somePath")
@GET, @POST, @PUT, @DELETE
@Produces("...")
@Consumes("...")
@*Param("...")
```

REST action vs. HTTP method mapping

REST action	HTTP method
Create	POST
Read	GET
Update	PUT
Delete	DELETE