201300180 Data & Information – Test 3 ANSWERS

5 June 2015, 13:45 – 15:15

WARNING: The answers below are not complete, i.e., there may be many more correct and almost correct answers. Furthermore, the assessment is not based on a number of points per sub-question, but on the level of mastery on a certain number of aspects.

Question 1 (XQuery) (50 points)

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

```
distinct-values(//module/@id)
distinct-values(//call/module/@id)
distinct-values(//module[parent::call]/@id)
```

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

```
for $call in //call
let $os := $call//os/@id
group by $os
return <os id="{$os}">{count($call)}</os>
```

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

```
//call[./os = "Vista"]/module
//module[preceding-sibling::os = "Vista" or following-sibling::os="Vista"]
```

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

```
//call[contains(.,"Suse") and not(./os/@id="1")]
```

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

```
//call[@id=7735] /following-sibling::call[./module="ABC"]
```

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

```
<helpdesk>{
for $call in //call
let $mod := $call//module/@id
group by $mod
return <module id="{$mod}">{$call}</module>
}</helpdesk>
```

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

```
delete //call[./os="Linux"]
delete //os[. = "Linux"]/parent::call
```

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

Yes. Well-formed means the opening and closing tags match and that there is a root element. Valid means it conforms to a schema. So, if the tags match and there is a root element, but it doesn't conform to a given schema.

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

Mixed content.

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

No, there is no root element.
And, <modules>{//module}</module> is a possible adaptation.

Question 2 (Database transactions) (50 points)

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?

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

Index: Unique index on id.
Constraints: primary key on id + foreign key on bestfriend.
(NOT NULL constraint on id is not wrong; I just want to see the difference in index and constraint and that both are created for primary key and that no index is created for foreign key)

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

CREATE VIEW personview AS (SELECT id,name FROM person);

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

For security. If there is a user group that should not be able to access the bestfriend information, then they can be granted only access to the view and not the table.

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"
```

Read-lock on the row in account belonging to account "A" (for the subquery)
Write-lock on the row in account belonging to account "B"

You can also mention a read-lock on the row in account belonging to account "B", because the query reads it before updating it.

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

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

No. $r_1(x)$ and $w_2(x)$ conflict, so if it were equivalent with a serial schedule, T_1 should be before T_2. Also, $w_2(x)$ and $w_1(x)$ conflict, which means that if it were equivalent with a serial schedule, T_2 should be before T_1. These two necessities cannot be true at the same time.

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

Dirty writes ($w_1(x)$ after $w_2(x)$). The other ones not.

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) are delayed by the locking implementation of that isolation level.

At all isolation levels, dirty writes are prohibited. Write locks delay $w_1(x)$ (hence also $w_1(y)$ and c_1). At REPEATABLE READ, the long term read locks also delay $w_2(x)$, also $w_1(x)$, and subsequently also $w_3(y)$. Deadlock.