Data & Information – Test 3 (1.5 hours)

9 June 2017, 13:45–15:15

Program: Technical Computer Science / Business & IT
Module: Data & Information (201300180)
Module Coordinator: Klaas Sikkel

Please note:

- Please answer every question on a different sheet of paper (the answers will be distributed to different person for grading).
- You are not allowed to bring any study materials to the test; essential excerpts from the study materials are available as appendices.
- You are allowed to use a calculator.

Grade = #points/10

Question 1: Information Retrieval / Full-text Search (35 points)

Tip: See Appendix 1 for an informal syntax of SQL including types, functions and operators important for full-text search.

a) [FT-search] Suppose we have the tables below. The table 'entry' contains titles of entries and the table 'review' contains reviews for these entries.
Write an SQL query that uses the full-text search capabilities for finding all entries that have a review that matches "awful | horrible".

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

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

b) [FT-search] Given the query below. The function 'string_agg' is a special kind of function. What is the technical term for this kind of function? And, describe exactly what the function does.

```
SELECT eid, string_agg(txt) AS reviews
FROM review
GROUP BY eid
```

(continued on page 2)

c) [IR; tf.idf] Given a table with lines of poems as illustrated below (filled with Sonnet 27 from Shakespeare), calculate the tf.idf scores for the query 'my limbs work'. Explain your answer by giving the full calculation. The formulas for ranking and idf are given below the table. If you don't have a calculator for computing log, just invent a number between 0 and 1 (explicitly mention that you did this!). The terms of the query are given in bold for your convenience.

Poem_id	Line_id	Line
1	1	Weary with toil, I haste me to my bed,
1	2	The dear repose for **limbs** with travel tired;
1	3	But then begins a journey in **my** head,
1	4	To **work my** mind, when body's **work**'s expired:
1	5	For then **my** thoughts, from far where I abide,
1	6	Intend a zealous pilgrimage to thee,
1	7	And keep **my** drooping eyelids open wide,
1	8	Looking on darkness which the blind do see:
1	9	Save that **my** soul's imaginary sight
1	10	Presents thy shadow to **my** sightless view,
1	11	Which, like a jewel hung in ghastly night,
1	12	Makes black night beauteous and her old face new.
1	13	Lo, thus, by day **my limbs**, by night **my** mind,
1	14	For thee and for myself no quiet find.

Idf(t) = log (N/df)

Rank(d,q) = $\sum_{t \in q}$ tf(d,t) idf(t)

d) [IR; language models] Calculate P(my,limbs|D) where D is line 13 of the above sonnet. You may assume that terms are independent. Explain your answer by giving the full calculation.

Question 2: Database transactions, isolation levels, indices, constraints, views (35 points)

Tip: See Appendix 1 for an informal syntax of SQL.
Tip: See Appendix 2 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) Given the schedule below, which pairs of operations are conflicting?

$r_1(x)\ r_1(y)\ w_2(x)\ w_3(y)\ w_1(x)\ r_4(y)\ c_1\ c_2\ c_3\ c_4$

b) Is the schedule above serializable or not? Explain your answer

c) If there were no locking used and the schedule would execute as above, which dirty reads and/or dirty writes would take place?

d) Suppose the database would run under isolation level SERIALIZABLE, i.e., it would use long-term read- and write-locks, what schedule would emerge for the above stream of operations? Explain your answer.

e) Suppose you execute the two SQL queries below on the tables of Question 1a.

```
UPDATE entry                    SELECT validated, COUNT(*)
SET validated = 'V'             FROM entry, review
WHERE eid=23846                 WHERE entry.eid = review.eid
                                GROUP BY validated
```

i. Which anomalies may occur if they would run concurrently in two separate transactions?

ii. What is the lowest isolation level for each transaction, such that these anomalies are prevented? Explain your answer.

f) In the table definition of Question 1a, there is a clause "FOREIGN KEY (eid) REFERENCES entry(eid)". What things are being created because of this clause? Explain your answer.

Question 3: REST (30 points)

1. How does JAXB facilitate the implementation of RESTful services in Java? How can you indicate that you are using JAXB in Jersey? (10 pt)

2. Suppose you have to design and implement a Web service to manage a collection of students of the university, which have a (unique) student id, a full name and a list of courses.

   a) Which data contents should be given in an HTTP request (HTTP method, URL, HTTP message body) in order to create a student called "Peter Perfect" with student id S234567 who follows only the "Data and Information" course? (10 pt)

   b) How do the contents of the HTTP message depend on the resource representation and why? Give the concrete representation of this student in a format of your preference to justify and illustrate your answer. (10 pt)

Try to be as concrete as possible when answering this question. Vague answers like the 'information mentioned should be given' will not be considered for marking.

Appendix 1: Informal syntax of SQL

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.

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, ...)
...
Full-text types: tsvector, tsquery
Full-text functions: to_tsvector, to_tsquery, ts_rank, ts_headline, string_agg, setweight, coalesce
Important operators: @@ (match), || (concatenate), :: (cast)

Appendix 2: 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 in P)	Long-term predicate locks