Data & Information – Test 3: Solutions

8 June 2018, 13:45–15:15

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

a) [FT-search] Suppose we have the tables below. The table 'person' contains both students and teachers and the table 'project' contains student project descriptions as well as which student the project belongs to and which teacher is supervising the project.

Write an SQL query that uses the full-text search capabilities for finding all projects that have a description that matches "databases | (data & science)".

```
CREATE TABLE person (
    nr INTEGER,
    name TEXT,
    PRIMARY KEY (nr)
)
```

```
CREATE TABLE project (
        pid INTEGER,
        student INTEGER,
        supervisor INTEGER,
        description TEXT,
        PRIMARY KEY (pid),
        FOREIGN KEY (student) REFERENCES person(nr)
        FOREIGN KEY (supervisor) REFERENCES person(nr)
)
```

➢
```
SELECT project.pid
FROM project
WHERE to_tsquery('english','databases | (data & science)')
        @@ to_tsvector('english',project.description)
```

b) [FT-search] Given the tables above. Write an SQL query that uses the full-text search capabilities for finding projects that match a search keyword *K* in either the description or the name of the student or the supervisor.

➢
```
SELECT project.pid
FROM project, person sup, person stud
WHERE sup.nr = project.supervisor
AND stud.nr = project.student
AND to_tsquery('english',K)
     @@ to_tsvector('english',sup.name || ' ' || stud.name || ' ' ||
person.description)
```

c) Which of the following statements is true; explain your answers

i. Converting a string to a tsvector leaves the words intact.

ii. When converting a string to a tsvector, every word in that string becomes part of the resulting tsvector.

iii. There exists an index structure capable of speeding up full-text queries.

- i: FALSE; techniques like ‘stemming’ may cut off parts of words, e.g., ‘rats’ -> ‘rat’.
  ii: FALSE; due to stop-word removal, words like “a” and “the” will not end up in the resulting tsvector.
  iii: TRUE; there is such an index structure. In PostgreSQL it is called GIN. The general term is an ‘inverted index’, an index that per word stores in which documents it occurs possibly combined with some other statistics such as the term frequency.

d) [IR; tf.idf] Given a table with quotes as illustrated below (filled with quotes from https://www.coolfunnyquotes.com), calculate the tf.idf scores for the query ‘why friend’. 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 (we assume ‘stemming’ so the words “friends” and “friend” are both considered the same term “friend”).

Idf(t) = log (N/df)

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

- For documents 1, 2, 7: tf(d,t) = 0 for both ‘why’ and ‘friend’
  tf(4,why)=tf(6,why); otherwise tf(d,why)=0
  tf(3,friend)=2; tf(5,friend)=tf(8,friend)=1; otherwise tf(d,friend)=0
  note that in document 8, “friends” is mapped to “friend” due to stemming.
  df(why) = 2, so idf(my) = log(8/2) = log(4) = ±0.6
  df(friend) = 3, so idf(friend) = log(8/3) = ±0.426
  Hence, Rank(d,q) = 0 for d=1, 2, and 7;
  Rank(3,q) = 0*0.6 + 2*0.426 = 0.852
  Rank(4,q) = 1*0.6 + 0*0.426 = 0.6
  Rank(5,q) = 0*0.6 + 1*0.426 = 0.426
  Rank(6,q) = 1*0.6 + 0*0.426 = 0.6
  Rank(8,q) = 0*0.6 + 1*0.426 = 0.426
  Hence the most relevant document is 3, then 4 and 6, then 5 and 8.

e) [IR; language models] In language models, it is often assumed that terms are independent (this is called the unigram model). In the bigram model, probabilities P(w1 w2|D) are determined and stored for all combinations of w1/w2, so that we need not assume independence P(w1 w2|D) = P(w1 | D) * P(w2 | D). Can you give a typical example from the quotes above, where P(w1 w2|D) >> P(w1 | D) * P (w2 | D)?

- “best friend” or “good friend” or “better than” … any combination of two words that frequently occur together.

Question 2: Database Transactions, Isolation Levels, Indices, Constraints, Views

a) Given the schedule below, which pairs of operations are conflicting?

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

- 1: $r_1(x)$ and $w_2(x)$
  2: $r_2(x)$ and $w_1(x)$
  3: $w_1(x)$ and $w_2(x)$
  4: $r_1(y)$ and $w_3(y)$

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

- T1 should be before T2 because of conflict 1
  T2 should be before T1 because of conflict 2
  T1 should be before T2 because of conflict 3
  T1 should be before T3 because of conflict 4
  The first and second cannot be true at the same time, hence not serializable

c) 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.

- 1: $r_1(x)$: request by T1 for read lock on 'x' granted; T1 reads 'x'
  2: $r_1(y)$: request by T1 for read lock on 'y' granted; T1 reads 'y'
  3: $r_2(x)$: request by T2 for read lock on 'x' granted; T2 reads 'x'
  4: $r_3(y)$: request by T3 for read lock on 'y' granted; T3 reads 'y'
  5: $w_1(x)$: request by T1 for write lock on 'x' not granted (T2 holds read lock on 'x'); T1 is delayed.
  6: $w_2(x)$: request by T2 for write lock on 'x' not granted (T1 holds read lock on 'x'); T2 is delayed.
  7: $w_3(y)$: request by T3 for write lock on 'y' not granted (T1 holds read lock on 'y'); T3 is delayed.
  All transactions delayed: DEADLOCK

  So, the order of execution (the schedule) is
  $r_1(x)\ r_1(y)\ r_2(x)\ r_3(y)$ DEADLOCK

d) Suppose there are only two applications running on the same database accessing the same single table contained in this database. The first application is a read-only application where each SELECT-statement is a single transaction. The other application only appends rows to this table in batches, i.e., several rows in one transaction. For each anomaly, explain why it can or cannot occur in this situation if the database runs in the lowest isolation level READ UNCOMMITED.

- Dirty write: cannot occur. There are no two transactions running at the same time writing to the database. Other explaination: there is write locking, so dirty writes cannot occur anyway.
  Dirty read: can occur. If application 2 is writing a batch, application 1 can read already written rows before the transaction of application 2 is finished.
  Non-repeatable read: cannot occur. This anomaly refers to two reads in the same transaction, but application 1 has only transactions with one read, and application 2 doesn't read at all.
  Phantom: can occur. If application 2 is writing a batch, application 1 can execute a SELECT-query for which the WHERE-clause selects (and reads) written rows before the transaction of application 2 is finished.

e) Suppose a table "project" is created containing a declaration "FOREIGN KEY (student) REFERENCES person(nr)" and filled with data of some projects.

  i. Describe what happens when we issue a "DELETE person" statement deleting all rows in the person-table.

  ii. Describe what happens when we issue two statements "DELETE person" and "DELETE project" in the same transaction in an attempt to delete all data in the database.

- i: the foreign key constraint is violated, since there are rows in project remaining that would refer to non-existing persons. Therefore, the transaction is aborted and the database remains in its original state with table person unaffected.
  ii: constraints may be violated during a transaction. In this situation, the end result does not violate the constraint, hence commits normally resulting in both tables empty.

Question 3: REST (30 points)

a) How does a RESTful client indicate to the server the encoding to be used in the data contained in the response to a certain HTTP request? How does a RESTful server implemented using Jersey (JAX-RS) select the Java method that gives a response with the proper encoding?

Answer
A client indicates the expected encoding in the HTTP request message, in the Accept header parameter value.
The server selects a method with a `@Provides` annotation that refers to the same encoding as the Accept header parameter value of the HTTP request message.

b) Suppose a web service is available that manages a collection of students in an administrative application of a university. Assume that the REST URL conventions have been properly followed so that

- `http://anyhost/students` represents the whole collection of students
- `http://anyhost/students/`*`id`* represents a student with identifier *id*

Explain how a client can *update a student* (Update REST action) by describing the HTTP messages (request/response) that are exchanged to do this and the possible contents of these messages (HTTP method, URL and message body). Discuss both the successful and unsuccessful executions!

Answer
To update a student the client issues an HTTP PUT message to URL `http://anyhost/students/`*`id`*, where the student representation (student information to be updated) is given in the body of the message as payload, in accordance with the encoding indicated in the message (Content-type header parameter value).
In case the update is successful, the server responds with an HTTP response indicating success (code 20X). In the case of an error the server responds with an HTTP response containing an error code (40X or 50X).

c) The following statements have been added to the `/WEB-INF/web.xml` file in a project that implements a RESTful service using JAX-RS:

```
<servlet>
     <servlet-name>javax.ws.rs.core.Application</servlet-name>
</servlet>
<servlet-mapping>
     <servlet-name>javax.ws.rs.core.Application</servlet-name>
     <url-pattern>/rest/*</url-pattern>
</servlet-mapping>
```

What has been defined in these statements? How does the Application Server (Tomcat) interpret this definition in order to eventually reach the intended RESTful service?

Answer
This statement defines that a message with path `<web-app-URL>/rest` should be forwarded by the Application Server to the Jersey Servlet (RESTful container, class `javax.ws.rs.core.Application`). The Jersey Servlet works as a container for the RESTful services and determines the intended services (classes and methods) based on the JAX-RS annotations (e.g. `@Path`, `@Consumes` and `@Produces`).