Test Pearl 010 — Databases

Pearls of Computer Science (201300070)
Bachelor module 1.1, Technical Computer Science, EWI

22 september 2016, 08:45–09:45

Module coordinator: Maurice van Keulen
Instructor: Maurice van Keulen

- You may use 1 A4 document with your own notes for this exam and a *simple* calculator.
- Scientific or graphical calculators, laptops, mobile phones, books etc. are not allowed. ***Put those in your bag now!***

Total number of points: 40.
Total number of pages: 5.

1 *SQL (20 points)*

For this question, we use the tables belonging to a 'shared notebook' database. A description of the table structure can be found in Figure 2. The database stores notes of users (who each have an account). The notes are organized in notebooks. Moreover, individual notes as well as whole notebooks can be shared with other users.

The sharing mechanism perhaps needs some more explanation. If a user shares a whole notebook with another user, a row in 'shared_nb' is created with the id of the notebook (nb_id) and the id of the account (account_id) the notebook is shared with. Moreover, the attribute 'access' is a string that either contains "edit" or "read" which determines whether the other user is given read/write or read-onlly access to the notebook, respectively.

Alternatively, it is also possible to share an individual note with another user. In that case, a row in shared_note is created. This row additionally contains the notebook_id of the notebook where the other user has placed this shared note.

Figure 3 contains some example data. There are 3 accounts for 3 pearl teachers you now know. Maurice has two notebooks, "Private" and "Teaching" of which the latter is shared in full with Pieter-Tjerk. The notebook "Private" contains a note "Quotes". The notebook "Teaching" contains 2 notes: "TA Planning" and "Sign off list", which are both shared with Arend who keeps them in his notebook "From Maurice". Pieter-Tjerk has one notebook "My Notes" with two notes "Test" and "4ArendAndMaurice". The latter note is shared with read access with both Arend (who keeps it in notebook "From Other") and Maurice (who keeps it in notebook "Teaching").

Tip: Figure 4 contains an informal description of the syntax of SQL.

(a) Write an SQL-query that produces the names of all notebooks owned by user Maurice (i.e., disregard sharing).
(b) Maurice (id 1) wants to share his note "Quotes" (id 100) with read access with Arend (id 3) who likes to keep it in notebook "From Maurice" (id 13). Write an SQL-statement that inserts the row(s) needed to establish this sharing. You may directly use the mentioned id's.
(c) Write an SQL-query that shows per user, how many notebooks are shared with the user.
(d) Write an SQL-query that shows for user Arend the title of all the notes he has access to (so the ones he owns as well as the ones that are shared with him).

(e) Write an SQL-query that shows per notebook name how many shared notes are in the notebook.
(f) Pieter-Tjerk would like to change the access to 'edit' for everyone he shared his note "4ArendAndMaurice" with. Write an SQL-statement that makes this change in the database.

□

Answer to 1. In the evaluation of these questions, special attention was given to correct usage of

1. join conditions (J),
2. value conditions (C),
3. group by and count, i.e., aggregation (A),
4. inserts, deletes, and updates (U), and
5. SQL in general (S).

(a) This question required a join and value conditions (J, C).

```
SELECT notebook.name
FROM account, notebook
WHERE account.account_id = notebook.owner_account_id
  AND account.user_name = "Maurice"
```

Note that a value condition "`account.account_id=1`" is *not correct*. The question does not contain the information about an account_id being one, hence this needs to be looked up in the database by the query as well. The only thing given is the name "Maurice", so you should query with only this information! Any variations using the `JOIN` keyword or subqueries are fine as well, provided they correctly answer the question.

(b) Simple insert statement (U)

```
INSERT INTO shared_note(note_id, account_id, nb_id, access)
VALUES (100, 3, 13, 'read')
```

(c) Simple group by query with a join condition (A,J).

```
SELECT account.user_name, COUNT(*)
FROM account, shared_nb
WHERE account.account_id = shared_nb.account_id
GROUP BY account.user_name
```

Note that both the `GROUP BY` as well as the `COUNT` are needed and the same attributes should be mentioned with both.

(d) This is a very difficult one. You should have at least a query with the value condition `account.user_name = "Arend"` in it (C). With the rest you can obtain *bonus* points, especially for showing that you understand that a note should be returned for three possible reasons: it is your own, it is explicitly shared with you, and it is contained in a notebook that is shared with you. The solution below specifies this with three conditions combined with an `OR`. Variations with subqueries are also possible.

```
SELECT DISTINCT note.title
FROM account, notebook, note, shared_nb, shared_note
WHERE account.user_name = "Arend"
  AND (   (account.account_id = notebook.owner_account_id
           AND notebook.nb_id = note.nb_id)
        OR (note.note_id = shared_note.note_id
           AND shared_note.account_id = account.account_id)
        OR (shared_nb.nb_id = note.nb_id
           AND shared_nb.account_id = account.account_id))
```

(e) Again a group by query with a join condition (A,J).

```
SELECT notebook.name, COUNT(*)
FROM notebook, shared_note
WHERE notebook.nb_id = shared_note.nb_id
GROUP BY notebook.name
```

(f) This is an update statement, of course. You should have a correct value and join condition as well as all ingredients for a correct update statement: table that is updated, attribute the receives new value and `FROM` and `WHERE` clauses to specify which particular row needs to be updated, which you can only specify by joining with the `note` table to obtain the needed `note_id` that is associated with the note title "4ArendAndMaurice" (U, J, C).

```
UPDATE shared_note
   SET access = edit
  FROM note
 WHERE shared_note.note_id = note.note_id
   AND note.title = "4ArendAndMaurice"
```

2 *Databases (10 points)* For this questions, please use correct database terminology as much as possible.

(a) Give an important reason why the ANSI/SPARC architecture distinguishes between the conceptual schema and the physical schema.

(b) SQL is at the same time a QL, DML, DDL, and SDL. To which of these kinds of languages does the CREATE-statement belong?

(c) In one transaction we execute the following four statements with the intent to transfer 200 euro from account A to B:

- a:=read account A
- b:=read account B
- put b+200 in account B
- put a-200 in account A

Initially, A=100 and B=1000. Suppose an integrity constraint has been defined that the value in every account should be positive, i.e., above zero. The fourth statement, however, violates this constraint, i.e., it tries to put the value -100 in account A.
Explain what happens and give the values of both accounts after execution of this transaction.

(d) What is the difference between *data distribution* and *data replication*?

□

Answer to 2. These questions can all be answered based on the information from the book chapter "Databases".

(a) "data independence" is the short and correct answer that I intended. Many variations that described things about having to change software if the schema changes using phrases like "not hanving to program on physical level" or "feature proof" are fine as well.
(b) "DDL" is the correct answer: '`CREATE TABLE`' adds a table to the schema of the database, hence is data definition. I also accepted "SDL" as correct: officially not correct, but the statement may contain details on how to store certain attributes (e.g., how many characters a string can contain) which you could consider as 'storage definition'.
(c) There are two important ingredients in this answer that I wanted to see both: the transaction *aborts* and all changes are *rolled back*. Any terms like 'error' or 'execution stops' are acceptable for describing the abort, but I do wanted to see an indication as well that the DBMS makes sure to bring everything back to the state before the transaction started as if nothing happened, i.e., A=100 and B=1000.
(d) Data distribution refers to putting pieces of the data onto different servers; these servers hence will contain different data. Data replication refers to putting copies of the data onto different servers; these servers hence will contain the same data.

3 *Database design (10 points)*

Figure 1 contains part of the datamodel belonging to the database of a social media provider. The main entity is 'Account' with an attribute 'name' for the user's name and 'fullname' contain his or her full name. Below the entity 'Account' you see a line going from 'Account' back to 'Account' with a '*' at each end and a dashed line to 'Connection'. This represents the connections between accounts with their 'role' (e.g., "family" for connections between family members, or "business" for business relations). Every account obviously has messages (with a title and body) and a message may have zero or 1 picture associated with it. Finally, an account may also have a profile picture.

(a) According to the ER model of Figure 1, is it possible for the same picture to be associated with a message as well as with an account as profile picture? Explain your answer.
(b) According to the ER model of Figure 1, is it possible for the same picture to be the profile picture for two different accounts? Explain your answer.
(c) Give a table structure for the ER-model of Figure 1. Do this in terms of a list of tables and attribute names. The attribute types may be omitted.

□

Answer to 3.

(a) YES: The lines 'has' and 'profile' going from Picture to Message and Account, respectively, both end in an '*'. In other words, the same picture can be associated with 0 or more messages and 0 or more accounts.

(b) YES: Again, the line 'profile' going from Picture to Account ends in an '*'. This means that one particular picture can be associated with 0 or more accounts.

(c) I looked at the identified tables (T), that all plain attributes have a place (A), that some attributes (appear to) be used as primary key (I), that the one-to-many relationships have been correctly incorporated (R), and finally that the many-to-many relationship Connection has been correctly incorporated (M).

- Account: **name**, fullname, profile_picid
- Message: **msgid**, title, body, has_name, has_picid
- Picture: **picid**, filename, size, data
- Connection: **account_name1, account_name2**, role

There are of course many variations possible that are also correct.

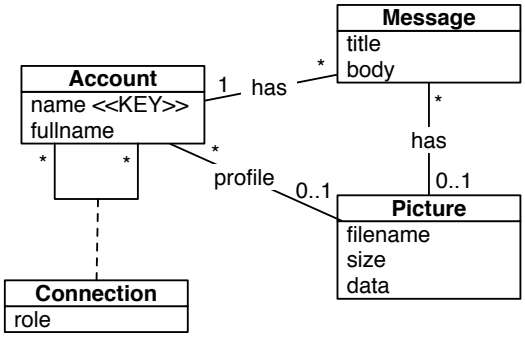

Figure 1: ER model of a social media provider's database.

Table	Attributes / Description
account	**account_id**, name, user_name, picture, ...
	For every account a unique number (account_id; also primary key) and the name of the account, the user's name, his/her picture, etc..
notebook	**nb_id**, name, owner_account_id
	For every notebook a unique number (nb_id; also primary key), the name of the notebook, and the account of the owner of the notebook (i.e., the account_id of the account the notebook belongs to.
note	**note_id**, title, body, nb_id
	For every note a unique number (note_id; also primary key), the title and body of the note, and the id of the notebook the note belongs to.
shared_nb	**nb_id, account_id**, access
	This table contains all sharing definitions for notebooks. A row specifies that notebook with id nb_id is shared with account account_id and provides either 'edit' or 'read' access.
shared_note	**note_id, account_id, nb_id**, access
	This table contains all sharing definitions for individual notes. A row specifies that note with id note_id is shared with account account_id who has placed it in his/her notebook with id nb_id, and provides either 'edit' or 'read' access.

Figure 2: Table structure of a "shared notebook" database (Primary keys in **bold**).

account

account_id	name	user_name	...
1	mvkeulen	Maurice	...
2	ptdeboer	Pieter-Tjerk	...
3	arensink	Arend	...

notebook

nb_id	name	owner_account_id
10	Private	1
11	Teaching	1
12	My Notes	2
13	From Maurice	3
14	From Other	3

shared_nb

nb_id	account_id	access
11	2	edit

note

note_id	title	body	nb_id
100	Quotes	...	10
101	TA Planning	...	11
102	Sign off list	...	11
103	Test	...	12
104	4ArendAndMaurice	...	12

shared_note

note_id	account_id	nb_id	access
101	3	13	edit
102	3	13	edit
104	3	14	read
104	1	11	read

Figure 3: Example data for the "shared notebook" database

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*∗ ')'
columndef: *colname type* [`NOT NULL`] [`UNIQUE`] [`PRIMARY KEY`] [`REFERENCES` *tablename* (*colname*+)]
constraint: `PRIMARY KEY` (*colname*, ...) | `CHECK` (*condition*)
| `FOREIGN KEY`(*colname*, ...) `REFERENCES` *tablename*(*colname*, ...)
query: `SELECT` (*column* [`AS` *colname*])+
`FROM` (*tablename* [`AS` *colname*])+
`WHERE` *condition* [`GROUP BY` *column*+] [`ORDER BY` *column*+]
column: [*tablename* '.'] *colname* | '∗'
Examples of *condition*: *column* = *value* [(`OR` | `AND`) [`NOT`] *column* <> *value*]
| *column* `IS` [`NOT`] `NULL`
| *column* [`NOT`] `IN` (*value*, ...) ...

Figure 4: Informal syntax of SQL