Homework 6

The deadline for this assignment is the same day as the exam date. However, if you turn in your answers before October 19, you will get individual comments which may help you prepare for the exam.

1. *High-level DB design with E-R diagrams* (15%)

Design a database for a venue with multiple rooms used for various events (for example, concerts or other performances) with the following requirements. Present your design using an E-R diagram, use the textbook notation for the E-R diagram. Make sure you specify all attributes and primary keys for the entity sets and attributes, participation and cardinality constraints on relationship sets. Follow the requirements given below.

- The venue has multiple rooms that are rented for different events.
- Each room has a unique name. The rooms only differ in their capacity. Currently there are three rooms with varying number of seats, but design should not be limited only to these rooms.
- For each room, the venue also keeps some more information independent of events organized in the room. You should be aware of this fact, but you are not required to model this part.
- An event can take place only in one of the rooms within a certain period. The smallest time period for an event is a day.
- Events can be periodic. In that case we want to link the current event to its most recent past occurrence.
- The database is required to support generic events that do not take place in a particular place.
- Each event has a title, description and one or more organizers.
- For each organizer, we require a name and a single email address and we store possibly multiple phone numbers, but a phone number is not required.
- Either the venue manages the ticket sales, or organizers handle it. In case the venue sells the tickets, we store the ticket price for the event, and allow a number of tickets to be reserved for the event organization.
- There is only one type of ticket (e.g., no front-back seat difference), and no seat number is specified on the tickets.

The requirements are intentionally simplified. Avoid making the design more complicated. If you make any additional assumptions, state them clearly.

2. *E-R to SQL* (15%)

Convert the E-R diagram you created in exercise 1 to SQL `create table` statements. Specify all primary and foreign key constraints. Additionally specify a `check` constraint for room names to be one of `Room 1`, `Room 2`, `Room 3` or `null`.

List all SQL statements you used in your report.

3. *Normal forms* (15%)

Assume we have the tables specified with the following partial database schema (note that this schema is not necessarily a good choice for the design above. For example, `phone_type` is not part of the requirements above.),

```
organizer(email, name)
organizer_phone(email, number, phone_type)
```

We assume that,

- emails are not shared.
- phone numbers may be shared by multiple people.

- a phone can have only one type, but one cannot decide the type of the phone based on the number.

State whether the following functional dependencies ($\rightarrow$) and multivalued dependencies ($\twoheadrightarrow$) hold.

a. `email` $\rightarrow$ `name`

b. `name` $\rightarrow$ `email`

c. `email` $\rightarrow$ `number`

d. `number` $\rightarrow$ `type`

e. `email number` $\rightarrow$ `type`

f. `email` $\twoheadrightarrow$ `name`

g. `email` $\twoheadrightarrow$ `number`

h. `type` $\twoheadrightarrow$ `number`

Assuming that the keys indicated in the schemata above are the only candidate keys for the respective tables, answer the following questions

. Explain your answers briefly, referring to above dependencies if possible.

i. If the table `organizer_phone` in Boyce-Codd normal form (BCNF)?

j. Is the table `organizer` in (BCNF)?

k. Is the table `organizer_phone` in 4^{th} normal form (4NF)?

l. Is whether the table `organizer_phone` in 3^{rd} normal form?

m. State whether knowing that the table is in BCNF would allow you to determine if it is also in 4NF?

n. State whether knowing that the table is in BCNF would allow you to determine if it is also in 3NF?

o. State whether knowing the table is **not** in BCNF would allow you to determine if it is not in 3NF?

For the rest of the questions, we will use the SQL statements from the files `/home/p252191/hw6-create.sql` and `/home/p252191/hw6-insert.sql` on siegfried for creating and populating the database. They are also available on the web as `http://www.let.rug.nl/coltekin/db2012/hw6-create.sql.txt` and `http://www.let.rug.nl/coltekin/db2012/hw6-insert.sql.txt` respectively.

The database structure and content created by the commands in these files are almost identical to the data from homework 4. However, you are recommended to use the data from the files above.

4.*Altering the database structure* (10%)

Write down the SQL statements necessary for the following tasks.

a. Alter the `groups` table such that the default value for the `ring_tone` is 'default.mp3'.

b. The current design does not allow per-contact ring tones, ring-tones are only possible for groups. Modify the database such that one can set per-contact ring tones.

c. We want to be able to store more than one photo for contacts. In case there are multiple photos, one of them is marked as 'primary' (e.g., to show it when a call is received from this contact).

Answer the following questions briefly.

d. Assuming that we have created a separate table for photos, using the schema `photo(fname,lname, photo, photo_type)`, can the DBMS enforce the requirement that there can be only one primary photo?

e. This database design uses two different methods to store unstructured data. Photos are stored as blobs, and ring tones are stored in the filesystem and database only stores path name of the file. List one advantage for each approach.

5.*Manipulating data* (10%)

Write down the SQL statements for the tasks below.

a. Add a new group, called `colleagues`.

b. Add another hypothetical contact with a name, one email address, and two phone numbers (`home` and `mobile`). Your new contact is working for 'University of Groningen'. Add the new contact to group `colleagues`.

c. Update the birthday for Clark Kent as 'June 18, 1938'.

d. Try an **`insert`** statement that would violate one of your foreign key constraints.

e. Try a **`delete`** statement that would violate one of the foreign key constraints.

6.*Simple queries* (5%)

Write down SQL statements for the tasks below.

a. Display full names of all contacts.

b. Display names of contacts who are older than 75 years, or whose birth year is unknown.

c. Find the contacts whose birthdays are within the current month.

d. List the first name of the contacts whose last name does not include the letter '`e`'.

e. Display the unique list of phone numbers in the database.

7.*Queries with joins* (10%)

Write down SQL statements for the tasks below.

a. List the last names and work addresses of the contacts who work for a company and have a work address in the database.

b. List names and ages of contacts that belong to a group together with the names of the groups (contact names may appear multiple times for each group they belong to).

c. List names of contacts and the groups they are assigned to (contacts may show up multiple times for each group they belong to). Make sure all contacts in the database are listed, even if they do not belong to any groups.

d. List the number of phone numbers stored for each group of contacts.

e. List the phone numbers together with associated ring tones from the `groups` table only if there is no ring tone associated with in the `contact` table.

8.*More complex queries* (10%)

Write down SQL statements for the tasks below.

a. Display the last name, email address, phone number, company name and occupation of all contacts who live outside US. Make sure all contacts living outside US are listed even if the database do not have a phone or work/workplace record for the contact.

b. List names of contacts and the groups they are assigned to. Make sure each contact is listed only once, and the groups are displayed as a comma-separated list.

c. Using a single SQL statement list the phone numbers of contacts either in group `friends` or `colleagues`.

d. List the ring tone(s) assigned to the contact for an incoming call (given a particular phone number). Use the phone number `333-3333` for your example query.

e. Using a single query, list the 'preferred' ring tone assigned to a contact. Use phone number `333-3333` as an example, but make sure your query works for any phone number. The preferred ring tone is determined as follows:

- if there is a per-contact ring tone assigned it is the 'preferred' ring tone.
- otherwise, if there is a ring tone assigned to one of the groups that contact belongs to one of these ring tones should be used (you can choose arbitrarily)
- if multiple ring tones are applicable (e.g., the same number is used by multiple contacts) pick the first one on the list.

9. *Views, Indexes, Access control* (10%)

Write down SQL statements for the tasks below.

a. Create a view that lists the ring tone to be used for each number in the database (similar to 8e, but view includes all phone numbers). You are allowed to list multiple ring tones for a single phone number if it is ambiguous according to specification in 8e.

b. Create a view that lists the names, upcoming birthdays, and the age that they will be in only for the contacts in group 'family' or 'friends'.
TIP: this is relatively tricky to do it only in SQL. However, here are a two (My)SQL commands that may be useful:

```
select adddate('1999-01-01', interval 2 * 5 year);
select datediff('2013-01-01', current_date()) as days_left_this_year;
```

You may want to consult MySQL documentation for use of these functions.

c. Grant read access (select) on the `contact` table, and update access to ring tone columns on `groups` and `contact` table for the user 'backup'.

For the next two questions, assume that the database management system does not create any indexes by default.

d. Create the most effective index that speeds up the following query. Explain briefly why the index you suggest is the best alternative.

```
select fname, lname, ringt from groups natural join contact_group;
```

e. Create indexes that would speed up the following two queries.

```
select * from contact_group natural right outer join groups;
select * from contact_group natural left outer join groups;
```