Logical database design

This homework consists of two parts. In the first part of this homework, we will go through some of the design decisions for a mobile phone contacts database, and try to argue if they are good or they need to be replaced. The second part consists of lab exercises. You can do them independently of the first part.

As before, please submit your report as a single PDF attachment on Nestor.

Part I: logical DB design and normalization

You are hired in a company that produces mobile phones. They already have a working phone. However, the programmers and some customers are upset with the structure of the contacts database, and you are appointed to the task of fixing the database design for the next release. The database allows multiple phone numbers, email addresses and addresses for each contact. It also keeps optional birth date information as well as some free-form notes about each contact. Addresses consist of street name, number on the street, post code and city. The following decisions are already made:

- We do not want to add new information to the contacts for the next release,
- We only consider addresses in the Netherlands.
- We assume that contact names are unique, name collisions are left for the user to resolve.
- We assume that an address is assigned to only one contact.

The current database schema consists of the following table schemata.

```
contact(name, email, notes, birth_date)
phone(contact, phone, type)
address(contact, street, number, pcode, city)
```

1. Identify all candidate keys for each table.
2. If we did not assume that contact names are unique, how would it affect your choice of keys?
3. Is `contact` table in first normal form? Explain your answer briefly.
4. For the `contact` table, do the following functional dependencies hold? If you make any assumptions, state them clearly.
   (a) name $\rightarrow$ email
   (b) name $\rightarrow$ notes
   (c) name email $\rightarrow$ notes
   (d) birth_date $\rightarrow$ name
   (e) name birth_date $\rightarrow$ name
5. Is the `contacts` table in BCNF? If your answer is 'yes', write down a hypothetical functional dependency that would violate BCNF. If your answer is 'no', state at least one functional dependency that violates BCNF.
6. Is the `phone` table in BCNF? If your answer is 'yes', write down a hypothetical functional dependency that would violate BCNF. If your answer is 'no', state at least one functional dependency that violates BCNF.
7. Is the `phone` table in 3NF? Do you need to go through all functional dependencies to answer this question? If not, why?
8. Is the `address` table above in BCNF?

9. State whether the following *multivalued* dependencies hold for `phone(contact, phone, type)` table or not.

   > **REMINDER:** Remember that a *multivalued dependency* (MVD) is a generalization of functional dependency. A MVD $A \twoheadrightarrow B$ holds, even if for a certain value of A there are multiple values for B (so $A \rightarrow B$ may not hold). However, for MVD to hold, the values that B takes has to be independent of the other attributes of the table.

   (a) type $\twoheadrightarrow$ contact
   (b) contact phone $\twoheadrightarrow$ type
   (c) contact $\twoheadrightarrow$ phone
   (d) phone $\twoheadrightarrow$ type
   (e) type $\twoheadrightarrow$ phone

10. Is this table in 4NF? Justify your answer using multivalued dependencies.

11. Suggest a new (decomposed) structure for the database above except the `address` table (we leave it out for simplicity, but you are welcome to work on this as well). Make sure that all your tables are in BCNF and they allow lossless join (that you do not loose any information by decomposing).

12. Sketch an E-R diagram that would correspond to the database schema you suggested in in exercise 11. Would the E-R diagram prevent the normal-form problems you observed in exercises above?

Part II: SQL exercises

SQL knowledge required for some of the exercises below are not yet covered in the class. Before starting with these exercises, you may want to have a look at the SQL tips and reminders at the end of the document. In any case, the exercises only touch on basic SQL. You can also refer to the textbook, or online resources.

For this part, we are switching to the bookshop database. We are only interested in the following two tables:
`books(ISBN, title, year, publisher)` and `authors(ISBN, author)`

13. Write down the SQL statements to create the tables. Do not forget to specify your primary and foreign key constraints. Additionally we do not want the book titles to be **`null`**. Make sure you include this constraint as well. Please use the table and attribute names as in the schema description above. If you have tables with the same name that you would like to save, you can rename them (see the information at the end of the document). The data in Exercise 16 may help you chose the data types for the attributes.

14. The MySQL command **`describe`** `table_name`. Type the commands **`describe`** `books;` and **`describe`** `authors;` and include the outputs of these commands in your report.

15. Which fields of the `books` table are allowed to take **`null`** values? Explain why `ISBN` cannot be null despite the fact that you did not specify a **`not null`** constraint for it?

16. Insert the following data into your bookshop database.

```
isbn =   0073523321
title =  Database System Concepts
author = Abraham Silberschatz, Henry F. Korth and S. Sudarshan
year =   2010
publisher = McGraw Hill

isbn =   013600637X
title =  A first course in database systems
author = Jeffrey D. Ullman and Jennifer Widom
year =   2008,
publisher = Prentice Hall

isbn =   0130319953
title =  Database systems: the complete book
author = Hector Garcia-Molina, Jeffrey D. Ullman and Jennifer Widom
year =   2002
publisher = Prentice Hall
```

include the output of the commands `select * from books;` and `select * from authors;` in your report.

17. Delete the book with `ISBN` 0073523321 from the `books` table. Does MySQL allow you to delete it? Is there any effect of this deletion on `authors` table? Is the result what you would expect? Why?

18. Insert the book record `'0073523321','Database System Concepts',2010,'McGraw Hill'` back to `books` table.

Before running the following commands, check the database engine used for your tables. You can use the MySQL command,

```
select engine
from information_schema.tables
where table_schema = 'my_db_name'and table_name = 'books';
```

note that you should replace `my_db_name` with name of your database (likely your student number).
If the above SQL command returns `MyISAM`, replace the string `MyISAM` below with `InnoDB`.

Create and populate two tables `books2` and `authors2` with the following commands:

```
create table books2(ISBN char(10),
                    title varchar(40) not null,
                    year int,
                    publisher varchar(20),
                    primary key(ISBN)) engine=MyISAM;
create table authors2(ISBN char(10),
                     author varchar(20),
                     primary key (ISBN, author),
                     foreign key (ISBN) references books2(ISBN)
                        on delete restrict on update cascade) engine=MyISAM;

insert into books2 select * from books;
insert into authors2 select * from authors;
```

repeat the exercise 17 with `books2` and `authors2` tables. (You need to answer the same questions.)

19. We would like the 'X' in the ISBN number '013600637X' to be lowercase. Update both `books` and `books2` tables such that the ISBN number is '013600637x'. Inspect the tables `authors` and `authors2` tables. Are there any differences? Why?

20. Write SQL queries for the the following tasks:
    - Display all information from the `books` table.
    - Display titles of all books.
    - List titles printed before 2010.
    - Display authors who have published in 2010.

Do not hesitate ask if you have any questions.
Veel succes!

SQL tips/reminders

The following lists a few example SQL commands you may need for this homework. You do not have to read/study them unless you find it useful.

- To create a table, we use the command **`create table`**. The following example creates two tables `instructor(ID, name, dept_id)` and `inst_phone(inst_id, phone_nr)`.

```
create table instructor (ID int,
                         name varchar(50),
                         dept_id int,
                         primary key (ID));
create table inst_phone (inst_id int ,
                          phone_nr int ,
                          primary key(inst_id , phone_nr) ,
                          foreign key (inst_id)
                             references instructor(ID));
```

  The first table has the primary key `ID` and the second table has a primary key consisting two attributes `inst_id` and `phone_nr`. Furthermore, on `inst_phone` table, `inst_id` attribute is defined as a foreign key referencing to the primary key of the `instructor` table (`ID`).

- You can add a constraint that prevents **`null`** values for a certain attribute in **`create table`** statement . This is done by inserting **`not null`** statement after the relevant attribute. For example, the following creates the `instructor` table of the previous example. However it also states that instructor names cannot be null

```
create table instructor (ID int,
                         name varchar(50) not null,
                         dept_id int,
                         primary key (ID));
```

- To insert values into a table, we use **`insert into`** statement. For example, if we want to insert a phone number `4444` for the instructor with the ID number `1`, we would use the SQL statement `insert into inst_phone (inst_id, phone_nr) values (1, 4444);` Note that you could skip attribute specification if you know the order of attributes. As a result the following SQL statement would be equivalent: `insert into inst_phone values (1, 4444);`

- To delete records (rows) from a table we use SQL `delete from` statement. The list of records to be deleted is specified with a `where` clause. For example, if we want to remove the record we inserted in the previous example, we would use:
  `delete from inst_phone where inst_id = 1 and phone_nr = 4444;` Note that if you skip the `where` statement, you would delete all rows from the table.

- We already experimented with the SQL command `select * from table_name;` For the exercises above, you need a little bit more: queries that combine two (or more) tables. We will go through this type of queries in detail for the next two weeks, but for this exercise (which lists all phone numbers for instructor with `ID` = 1) the following examples should be enough to guide you.

  ```
  select ID, name, phone_nr
     from instructor, inst_phone
     where inst_id = ID and ID = 1;
  ```

  Note that we are taking advantage of the fact that the column names are unique across the tables. A general form that would work even if the tables had identically named fields can be expressed with the following syntax:

  ```
  select instructor.ID, instructor.name, inst_phone.phone_nr
  from instructor, inst_phone
  where instructor.ID = 1
    and instructor.ID = inst_phone.inst_id;
  ```

  This can quickly get cumbersome if you have multiple tables, another solution is to use short aliases for your tables, with the following syntax:

  ```
  select i.ID, i.name, p.phone_nr
  from instructor i , inst_phone p
  where i.ID = 1
    and i.ID = p.inst_id;
  ```

  you can still use the attribute names without prefixing with names or aliases if they are unambiguous.

  TIP: Reading (or writing) SQL query statements with an alternative order is more intuitive for most people. First, the `from` clause tels you which tables you are using in your query. Second, `where` clause tells which records you are choosing, and the last, `select` statement picks which columns to display.

- SQL `alter table` command changes the structure of an existing table. It can be used to add column, rename columns, add additional constraints ...For this homework you may need to rename tables. The following example which renames a table named `books` to new name `books_new` shows how to do it:
  `alter table books rename to books_old`.