More SQL, views, access control

This homework builds on the sample database created for the first homework. You need to create the necessary tables, and populate it with the some initial data.

You can find `create table` and `insert` statements in files `hw5-create-table.sql` and `hw5-insert.sql`.
They are also available on the web as
`http://www.let.rug.nl/coltekin/courses/db2013/hw5-create-table.sql.txt`
and
`http://www.let.rug.nl/coltekin/courses/db2013/hw5-insert.sql.txt` respectively.

Some table names may conflict with the tables you already have in your database. Rename (or remove if you do not need them) the tables with the same name before creating the ones required for this homework.

You can either use `source` command from MySQL command line, or copy and paste the statements into phpmyadmin to run the statements in these files.

Exercises

(10%) 1. Inspect the tables created by `hw5-create-table.sql`. Answer the following question briefly. Note that the database is very similar to but not the same as the example solution for the first homework.

   (a) Can we store more than one address for a customer?

   (b) Can a customer order one or more books to be sent to another person?

   (c) Can we add a new column to `author` table for storing additional information (such as phone numbers) about authors without causing any (potential) anomalies?

   (d) Can a customer order multiple copies of a book in different conditions?

   (e) Is the table `book_order` in BCNF? Explain your answer briefly.

(10%) 2. Write SQL statements for the following tasks.

   (a) Insert an additional book and an additional customer to the database. Make sure that you also add at least one copy of the book to the stock.

   (b) Place a new order with at least two books for the customer you have added.

   (c) When we do not know the exact weight, we estimate weight of a book as $2 \times pagenumber$ grams. Update weights of all books for which we do not have weight data using this formula. Make sure not to change the books for which the exact weight is known.

   (d) Modify the database (add necessary table(s), column(s) or constraint(s)) so that customers can rate the books using a scale between 1 to 10. Add following ratings to the database,

book	customer	rating
I, Robot	Donald Duck	8
I, Robot	Sherlock Holmes	7
I, Robot	Clark Kent	9
A Wizard of Earthsea	Clark Kent	8
A Wizard of Earthsea	Sherlock Holmes	9
Roadside Picnic	Donald Duck	10

   (e) Modify the orders table to include a `cardnum` column, where the credit card number for the order is stored.

(15%) 3. Write the SQL commands for the following tasks:

(a) Display average rating for each book.

(b) For each ISBN in `book_order` table, display the number of copies ordered.

(c) List the names of the authors for whom we have more than one book record in the database.

(d) List the title of all books in the database, together with their authors. Make sure that you have only a single row for each book, where multiple author names are listed together as a single comma-separated list. (TIP: you can use the aggregate function `group_concat()`.)

(e) List all authors together with the number of times their books were ordered. Count only the orders, multiple copies of the same book in a single order should count as one. Make sure all authors are in your list, and the list is sorted such that best-selling authors are on top.

(15%) 4. Write SQL statements for the following tasks. Remember that for the parts below, MySQL will not enforce your check constraints, but it will accept the statement if it is well formed.

(a) The database specification indicates that the column `status` in table `orders` should be one of `N` for new, `P` for processing, `S` for sent. Modify the table by adding a check constraint that disallows any other value.

(b) We know that MySQL do not enforce the check constraints, and we do not want to use the type `enum`, both because it is not the behavior we expect and it is non-standard. There is another alternative (trick): you can utilize foreign key constraints for this purpose. Create necessary table(s) and alter the `orders` table to include a foreign key constraint for this purpose.

(c) Add a check constraint to enforce correctness of email addresses (to some extent). You should only allow email addresses that contain an at sign '`@`' sign. There should be at least one character to the left, and at least two strings with one or more characters separated by a dot '`.`' on the right.

Your check should reject `abc`, `@abc`, `a@bc`, `a@bc.`, `a@.bc`. But `a@b.c` is fine.

(d) To prevent a wrong date insertion, we want to refuse the updates where `send_date` before the `order_date`. Add another check constraint on `orders` table for this purpose.

(e) Can you use the trick suggested in (b) for enforcing the check constraint in (c) and (d)? Explain your answer briefly.

(15%) 5. Write SQL statements necessary for the following tasks.

(a) Create a view, called `new_books`, that lists ISBN and price of only the new books in the stock. Include the statement you used in your report.

(b) Using the view created in the previous exercise, update the database such that all new books are sold with 10% discount. Did the update work as expected?

(c) Create a view that only includes order number and the total price of the order.

(d) Test whether the view you created in the previous step updateable? If not, state one reason why it is not.

(e) The employees who process the orders need a list of orders together with the name and address of the recipient. It is convenient to have a view that shows all the necessary information at once. Create a view that contains order number, title, quantity and the condition of the book(s) ordered and the name and address of the recipient.

Note: the database design indicates that if the recipient name is **`null`** the name and address should be taken from customer table, otherwise the name and address from the orders table should be used. (TIP: using **`union`** is probably the most straightforward way to do this.)

(15%) 6. It is difficult to show the utility of indexes with the small tables we used in above exercises. For this exercise and the next, we will use two somewhat large tables. The table data is available as `hw5-t1.txt` and `hw5-t2.txt` `http://www.let.rug.nl/coltekin/courses/db2013/hw5-t1.txt` and `http://www.let.rug.nl/coltekin/courses/db2013/hw5-t2.txt`.

(a) Create two tables using the following commands. Use the commands as they are presented.

```
create table t1(word varchar(100), freq int);
create table t2(word varchar(100), spelling varchar(100));
```

Load the data from respective files to the tables you have created using the command `load data local in`. Report the commands you used, and the times it took for completing the insertion for both tables. (Note: you will need to copy the data files to the server siegfried, and start mysql with command line option `--local-infile=1`.)

Alternatively you can load the files to the tables from PHPMyAdmin interface.

(b) Run the following query, report the time it took the complete the query.
```
select * from t1 natural join t2;
```

(c) The statement **`describe`** gives you some information on how the query is executed. Run the command
`describe select` `*` **`from`** `t1` **`natural join`** `t2;`. Do you see any indication of the use of indexes for this query?

(d) Create all (and only) the necessary indexes to speed up this query.

(e) Compare the run time of the query and the result of **`describe`** command on the query before and after creating the indexes.

(10%) 7. (a) Drop the indexes and the tables you have created for exercise 6.

(b) Re-create the tables `t1` and `t2`, including the indexes in your **`create table`** statement.

(c) Repeat the exercise 6a. Do the time for loading tables differ from times you noted in exercise 6a? If they differ, explain briefly why it is different.

(d) Do you expect a speed difference on a very large set of data between the following two queries,
```
select * from t1 where word like 'a%';
select * from t1 where word like '%a';
```
Explain your answer briefly.

(10%) 8. For part of this exercise, we assume that we have two different departments: 'orders' which process the orders, and 'finance' which deal with the payments. Furthermore, assume that there are 'roles' (or DB users) with these names that the employees in the relevant department use to access the DB.

Note that you will not be able to test the result of these exercises directly, unless you are able to create these database users (or roles). You can, of course, replace these usernames with DB usernames of other students and ask them to test if you achieved the desired effect.

(a) The 'finance' user needs read access to the information in the `customer` and read and update access to the `orders` table. Write necessary SQL **`grant`** statements to give finance user the appropriate rights.

(b) We also realize that the finance user also needs to know the total price of each order. Allow finance user to see the view created in exercise 5c.

(c) The 'orders' user needs to see the information in `customer`, `book_orders`, `stock` and `orders` tables. They should be able update status of an order, and add new items to the `stock` table. Write the necessary SQL **`grant`** statements.

(d) You realize that credit card number information is visible in `orders` table. The 'orders' user does not need to see this information. Write the necessary **`revoke`** statement to make sure that the 'orders' user cannot see this information.

End of exercises. Veel succes!