DAT151
Database and Unix System Management

Spring 2020

Assignment 5

Obligatory assignment. Deadline: Monday 17.02.2019.

Your code should be easy to read:

- *Use indentation in your code.*
- *DDL for CREATE TABLE should have each column definition on its own line.*
- *Do not use screen dumps to show code. Code should be normal text, formatted as code.*

The report should include all necessary commands to complete the tasks, printout from the system, explanation of what is done, the result and explanation of the result.

Remember to include the names of the group members on the front page of the report. The report can be in English or Norwegian. The report should be handed in via it's learning.

The assignment should be accomplished in groups of two to three students. The assignment should be accomplished in groups of one to three students. Signing up for a group will be closed on Friday 14.02.

Lecturers:
Bjarte Kileng, Bjarte.Kileng@hib.no, room E418
Faustin Ahishakiye, Faustin.Ahishakiye@hvl.no, room E506
Violet Ka I Pun, Violet.Ka.I.Pun@hvl.no, room E417

Task 1: Normal Forms

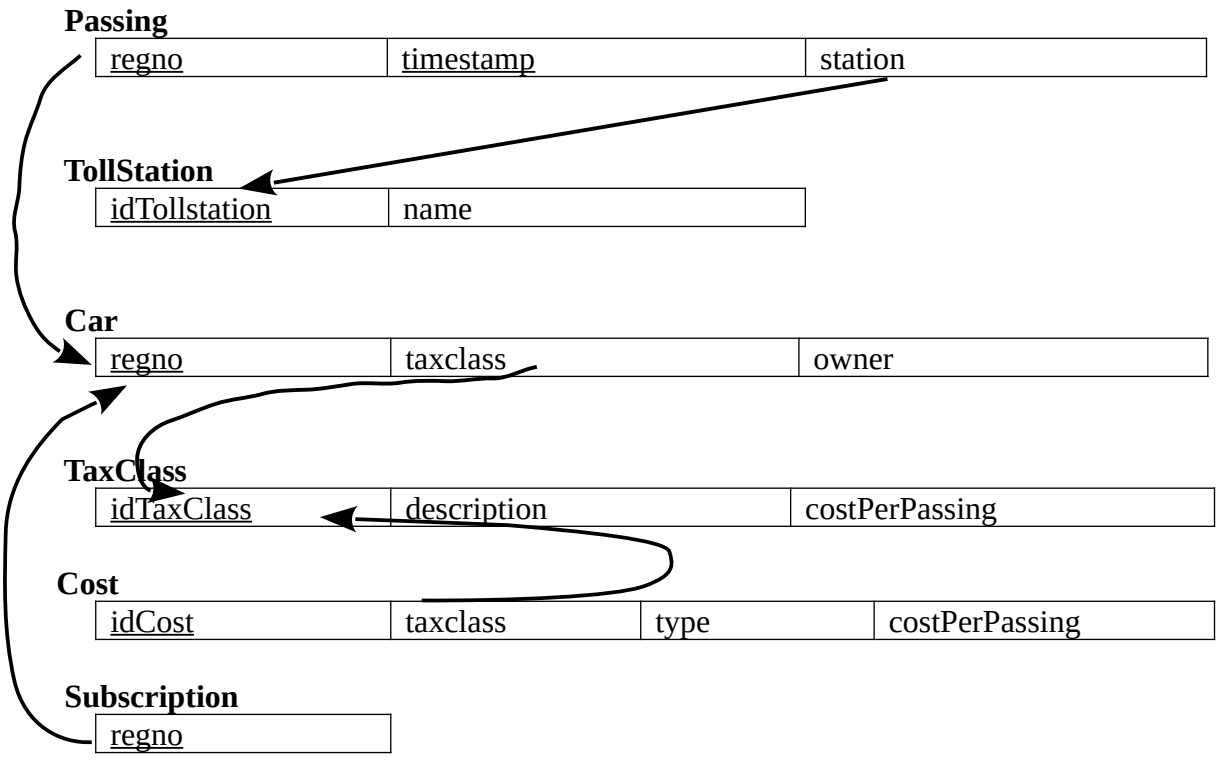
A logical database is given on the next page, and an EER Diagram for MySQL Workbench can be found in the attached file *car_passing.mwb*. Note that the file is only for reducing possible confusions, and you still need to edit it to satisfy the requirements.

Important: If enabling the MySQL Community Downloads, you must disable all install groups of *mysql-community.repo*, except for the group *mysql-tools-community*.

For the model on the next page, answer the following (explain why, not just yes/no):

1. Is this schema in 1NF?
2. Is it in 2NF?
3. Is it in 3NF?

The following logical database schema is given:



Task 2: Create a physical schema and a test environment

Unless explicitly required, all work must be done as a normal database user (i.e. not root).

You will now implement the schema from the previous task in your MariaDB database and fill it with test data before we in the next task will try to optimize the performance.

Fill the database with data from the attached file *carpassingdb.txt*. The fields are separated with ";" with each row on a separate line. The fields are:

1. Car registration number
2. Data and time for passing a toll station
3. Id of toll station
4. Name of toll station
5. Name of car owner
6. Id of car tax class
7. Description of tax class
8. Toll for passing a toll station

Observe that datafile is missing the subscription data. Pick a random 40% of the cars, and add them to the Subscription table. Put the tax somewhat lower for cars that has a subscription.

Task 3: Performance of physical model

Unless explicitly required, all work must be done as a normal database user (i.e. not root).

Consider the following SQL queries:

a) Retrieve a list over car owners and time of passing:

```
SELECT SQL_NO_CACHE c.owner, p.timestamp
    FROM Car c JOIN Passing p ON p.regno = c.regno;
```

b) Show how much each car owner that has a subscription has used on toll booths in total:

```
SELECT SQL_NO_CACHE c.owner, Sum(b.costPerPassing)
    FROM Car c, Passing p, TaxClass t,
         Subscription s, Cost b
    WHERE p.regno = c.regno
        AND c.taxclass = t.idTaxClass
        AND s.regno=c.regno AND b.taxclass = t.taxclass
        AND b.type = 'withsubscription'
        GROUP BY c.owner;
```

c) Same as above, using a subselect:

```
SELECT SQL_NO_CACHE c.owner, Sum(b.costPerPassing)
    FROM Car c, Passing p, TaxClass t, Cost b
    WHERE p.regno = c.regno
        AND c.taxclass = t.idTaxClass
        AND b.taxclass = t.taxclass
        AND b.type = 'withsubscription'
        AND c.regno IN (SELECT regno FROM Subscription)
        GROUP BY c.owner;
```

d) Same as b) and c), but for cars without a subscription.

e) Show all car owners that passed the toll station in Gravdal:

```
SELECT SQL_NO_CACHE c.owner FROM Car c WHERE c.regno
    IN (SELECT p.regno FROM Passing p JOIN Tollstation s
    ON p.tollstation = s.idTollstation WHERE
      s.name LIKE 'Gravdal' );
```

First, carry out the following and document the input and output:

- Turn on the profiling, run the queries, and then use SHOW PROFILE(S) to see the statistics.
- Use EXPLAIN to see how indexes, JOINs, and subqueries are handled by the optimizer.

Now, for each of the queries, try to improve performance by the following (restore to the original schema after each test):

1. Denormalization:

   Describe the denormalization you do, and explain what normal forms it violates. You will need to implement mechanisms to make sure that any redundancy is maintained correctly without any risk of loss of data integrity.

2. Using indexes:

   Create indexes and specify index hints (use index, ignore index, force index, ...), if necessary, to influence the execution of the queries.

Use profiling and EXPLAIN to show that your optimization improves performance. If not, or if

there is no need to do any optimization in a case, then simply write down the reasons.

Last changed, 10.02.2020 (BK)