School of Computing and Information Technology
University of Wollongong

Session: **Autumn 2016**
Lecturer: **Janusz R. Getta**

CSIT115/CSIT815 Data Management and Security
Assignment 4

Scope

This assignment consists of the tasks related to implementation of discretionary access control, granting system resources, verification of complex consistency constraints, using backup and restore features of DBMS to find the differences between two states of a relational table, and implementation of a simple auditing system.

This assignment consists of 4 tasks and specification of each task starts from a new page.

Prologue

Download and unzip a file `assignment4-all-files.zip`. You should get the files `Assignment4.pdf`, `dbcreate4.sql`, `dbdrop4.sql`, and `dbchange4.sql`. Copy the files to your USB drive such that you can access both files either through command line interface mysql or graphical user interface MySQL Workbench.

Tasks

Task1 (1 mark)

(1) Connect to MySQL as a user `root` either through command line interface mysql or graphical user interface MySQL Workbench and create a new database with a name the same as a *prefix of your University email account*.

(2) While connected as a user `root` use SQL script `dbcreate4.sql` to create the relational tables in a database created in the previous step. A script `dbcreate4.sql` creates and loads data into the relational tables that contain information about the employees, drivers, administration people, trucks, trips, and trip legs. You can use a script `dbdrop4.sql` to drop the relational tables. Do not drop the relational tables now !

There is no need to create and to submit any report from processing of the actions listed above.

Implement SQL script `solution1.sql` that performs the following actions.

(1) First, the script creates a new user with a name the same as *a prefix of your University email account*.

(2) Next, the script grants to the new user the privileges to create relational tables, to alter relational tables, to drop relational tables, to read and write relational tables, and to create views located in a database with the same name as *a prefix of your University email account*. The privileges must be granted such that the new user is able to grant the privileges to the other users.

(3) Next, the script grants to the new user the rights to reference a primary key in a relational table `EMPLOYEE` already created in a database with the same name as *a prefix of your University email account*. A privilege must be granted such the new user is not able to grant it to other users.

(4) Next, the script sets the following values of resource limits: total number of queries an account owner can issue per hour must be set to 100 and total number of simultaneous connections to the server by an account owner must be set to 10. A name of account owner is the same as *a prefix of your University email account* (the account of the new user created earlier).

(5) Next, the script locks an account of the new user.

(6) Finally, the script lists the privileges granted to the new user, the values of resource limits set in a step (4) and a status of the account of the new user.

Deliverables

A file `solution1.rpt` with a report from processing of SQL script `solution1.sql`. The report MUST have no errors and the report MUST list all SQL statements processed.

A report that contains no listing of executed SQL statements scores no marks and report that contains errors also scores no marks !

A name of the new user and a name of database created in Task 1 must be the same as *a prefix of your University account*. The names different from *a prefix of your University account* mean that your work has been done by another student with all consequences implied by such fact.

Task 2 (2 marks)
Connect to MySQL either through command line interface mysql or graphical user interface MySQL Workbench and execute a script file `dbdrop4.sql` and immediately after that a script `dbcreate4.sql` to refresh a sample database.

Implement SQL script `solution2.sql` that performs the following actions.

(1) First, the script changes a value of an attribute `LEGNUM` in one of the trip legs from `1` to a different number that does not violate a primary key constraint in a relational table `TRIPLEG`.

(2) Next, the script changes a value of an attribute `LEGNUM` in one of the trip legs in a trip different from the one changed in the previous step from a value equal to the total number of legs in a trip to a different number that does not violate a primary key constraint in a relational table `TRIPLEG`.

(3) Finally, the script finds all trips that violate the following consistency constraint.

"The first leg of each trip must have number 1 and the last leg of each trip must be equal to the total number of legs in a trip"

The script must list the outcomes of verification of the consistency constraint as a single column table with the following messages as the rows in the table.

`"Trip number` <insert trip-number here> `does not have leg number 1"`

`"Total number of legs in a trip number` <insert trip-number here> `is not equal to the largest number of a leg in the trip"`

Use a function `CONCAT` to create the messages above.

Deliverables
A file `solution2.rpt` with a report from processing of SQL script `solution2.sql`. The report MUST have no errors and the report MUST list all SQL statements processed.

A report that contains no listing of executed SQL statements scores no marks and report that contains errors also scores no marks !

Task 3 (1.5 mark)

(1) Connect as a user `csit115` to MySQL either through command line interface mysql or graphical user interface MySQL Workbench and execute a script file `dbdrop4.sql` and immediately after that a script `dbcreate4.sql` to refresh a sample database.

(2) Create a backup of a relational table `ADMIN` and save it in a file with the same name as *a prefix of your University email account*`.bak`.

(3) Execute a script `dbchange4.sql`.

(4) Create an empty relational table with the same name as *a prefix of your University email account* and with the same structure as already created relational table `ADMIN`. Remember, to create appropriate consistency constraints for the new table.

(5) Use a text editor and modify a backup file obtained in a step (2) such that a backup of a relational table `ADMIN` can be restored into a relational table with the same name as *a prefix of your University email account*.

(6) Restore a backup of the original contents of a relational table `ADMIN` into a relational table with the same name as *a prefix of your University email account*. Do NOT delete a backup file !

No report is expected from the implementation of the steps listed above.

Implement SQL script `solution3.sql` that that finds the differences between the contents of a relational table `ADMIN` and a relational table with the same name as *a prefix of your University email account*. The script must first list the rows added to a relational table `ADMIN` after the backup file was created, then the rows deleted from a relational table `ADMIN` after the backup file was created, and finally the rows changed in a relational table `ADMIN` after the backup file was created. In brief, the script must first list all added rows, then all deleted rows, and finally all changed rows in a relational table `ADMIN`. It is allowed to use more than one `SELECT` statement to implement such task.

Deliverables

A file `solution3.rpt` with a report from processing of SQL script `solution3.sql` and an updated file with the same name as *a prefix of your University email account*`.bak` which contains a backup of a relational table `ADMIN`. The report MUST have no errors and the report MUST list all SQL statements processed.

A report that contains no listing of executed SQL statements scores no marks and report that contains errors also scores no marks !

A submission without a file that contains an updated backup of a relational table `ADMIN` scores no marks.

A prefix of a name of backup file and a name of relational table created in Task 3 must be the same as *a prefix of your University account*. The names different from *a prefix of your University account* mean that your work has been done by another student with all consequences implied by such fact.

Task 4 (1.5 mark)
Some of simpler Database Management Systems, like for example MySQL 5.7 Community Edition, do not have the features that allow for automated auditing the database activities. In this task you will implement your own simple method of auditing the database database activities.

Connect to MySQL either through command line interface mysql or graphical user interface MySQL Workbench and execute a script file `dbdrop4.sql` and immediately after that a script `dbcreate4.sql` to refresh a sample database.

Implement SQL script `solution4.sql` that performs the following actions.

(1) First, the script makes a relational table that contains a general log empty.

(2) Next, the script sets the appropriate values of the variables that allow to create a general log, to save a general log in a relational table, and to start recording a general log from now.

(3) Next, the script executes a script file `dbchange4.sql`. There is no need to include a listing of SQL statement included in `dbchange4.sql` into a report from processing of script file `solution4.sql`.

(4) Next, the script sets the appropriate values of all variables that stop recording a general log from now.

(5) Next, the script lists all DDL statements (`CREATE`, `ALTER`, `DROP`) processed in a period of time when a general log was recorded.

(6) Next, the script lists total number of times each one of DML statements like `INSERT`, `UPDATE`, `DELETE` has been processed in a period of time when a general log was recorded. The script must use ONE `SELECT` statement to list total number of times `INSERT` has been processed, total number of times `UPDATE` has been processed, and total number of time `DELETE` has been processed. It is acceptable to ignore DML statement if it has not been processed at all. Sort the results in an ascending order of the total number times a DML statement has been processed.

(7) Finally, the script lists the names of users together with DML statements processed by the users each time a statement was processed on the same day of week when you tested this task.

Deliverables
A file `solution4.rpt` with a report from processing of SQL script `solution4.sql`. The report MUST have no errors and the report MUST list all SQL statements processed.

A report that contains no listing of executed SQL statements scores no marks and report that contains errors also scores no marks !

End of specification