Lab: Transactions in SQL

Preamble

For this lab, you can use your local MySQL server and MySQL Workbench as the client. In order to have complete control over your transactions, you must disable autocommit every time you open a new connection[1] to your database. To do so, click on the "Toggle autocommit mode" icon.

From that moment on, transactions will be managed by the `start transaction`, `commit` or `rollback` statements you issue or by hitting the "Commit the current transaction" / "Rollback the current transaction" icons.

Note that for transactions to work properly, your tables must use the `InnoDB` storage engine: make sure you created the Company database by executing the script provided on Campus.

In some exercises, you will have to open two distinct connections to your database. To do so, open two connections under the same MySQL Workbench instance: this will create two *main* tabs in the workbench window, one for each connection. Note that all the query tabs of any one main tab relate to the same connection and *thus share* the same transaction environment.

Tasks

1. Commit and rollback

You need to use only one connection in this exercise. After disabling autocommit, start a new transaction. Then insert a new department in the DEPT table. Check that the new department is visible by the current transaction by issuing a `select` statement. Next, issue a `rollback` statement and then a new select statement. What do you observe?

Repeat this experiment and replace the rollback statement with a `commit` statement.

2. Client Failure

As in the previous exercise, start a new transaction and insert a new department in the DEPT table. Then close the connection tab. Next, launch MySQL Workbench again and check whether the department you have just inserted is in the DEPT table.

Repeat this experiment, but this time terminate the client program abruptly by stopping the process with the task manager. What do you observe? Conclusion.

3. Transaction Isolation

Determine the default transaction isolation level of your database by executing the following command:

```
show variables like '%isolation%'
```

Next, devise the simplest possible experiment that demonstrates that by default, the modifications a transaction makes to the database are only visible to that transaction. Again, consider for example the insertion of a new department in the DEPT table. Repeat the experiment by deleting the department you have just inserted.

4. Isolation levels

Then repeat the previous experiment and set the transaction isolation level to `read uncommitted` by issuing the corresponding statement. This will set the isolation level of all the transactions to come.

Which transaction do you need to modify so that the behavior is different from the previous exercise? Check the effect on both transactions. Conclusion.

5. Isolation levels – Continued

Repeat the previous experiment this time by using the serializable isolation level. What do you observe on the second transaction? Explain.

6. JDBC code

We now consider the solution to the JDBC lab. (The solution is available on Campus. Please modify the connection parameters as needed.)

Modify the code of the `DataAccess` class to display (1) the default value of the autocommit mode, (2) the default level of transaction isolation. Use the Java API documentation for help.

Further modify your code so that (1) every Java method executes a transaction, (2) all transactions abort (or rollback) whenever an SQLException is thrown.