

## Lab6 – Transactions

This lab gives you the opportunity to get familiar with transactions and to reflect about the consistency and isolation guarantees provided by MySQL.

Consider the following table:

**Account**(account\_number, branch\_name, balance) (account\_number is key). The file account.sql gives you SQL code for this schema as well as an instance of the account table.

### 1 – Consistency

A - Enforce that a transaction that updates or inserts into the account table cannot set the balance lower than 0.

B - Formulate the following transactions in SQL and describe their effect (ensure that every statement is run inside its own transaction by typing 'set autocommit = 1' – this option guarantees that each time statement is submitted as a transaction – see <http://dev.mysql.com/doc/refman/5.0/en/innodb-transaction-model.html> for details) - Do the effects correspond to what you would expect?:

- update balance to -1 where account number equals 'A333'
- insert ('A-555', 'Central', -10)
- update balance to existing value minus 400 for all accounts
- update balance to existing value minus 400 for all accounts and then update balance to existing value plus 400 for all accounts (group the two update commands as a single statement by separating them with a semi column ).

### 2 – Isolation

Consider the following transactions:

- T1: compute the sum of all balances
- T2: debit 1000 kr from account 'A-102' and credit 1000 Kr to account 'A-333'.
- T3: update the balance for account 'A-102' by 10%.
- T4: debit 1000 kr from account 'A-333' and credit 1000 Kr to account 'A-102'.

1. Write SQL code for transactions T1, T2 and T3.
2. Open two terminals. Run a mysql client in each terminal
3. By default, the MySQL server runs at the isolation level

By default, MySQL runs a transaction per submitted statement. This is called the autocommit option. You have to turn this option off in order to control transaction boundaries:  
*set autocommit = 1;*

**REMEMBER TO USE THIS OPTION EVERY TIME YOU OPEN A NEW TERMINAL: THIS COMMAND IS ONLY VALID FOR A SESSION!**

From now on, every statement you execute runs in the context of a transaction. Note that MySQL implements chained transactions. You do not enter begin transactions: all statements are in the context of the same transaction until you commit or abort. – see <http://dev.mysql.com/doc/refman/5.0/en/innodb-transaction-model.html>

We focused on consistency in the first section. Focusing on durability would require stopping/restarting the MySQL server. This is quite cumbersome to control. We thus now focus on isolation and atomicity.

You can explicitly set the isolation level, using the SET TRANSACTION command. See <http://dev.mysql.com/doc/refman/5.0/en/set-transaction.html>

Conduct the following steps a-e using the isolation levels serializable and then read committed.

a. Drop the account table (using *drop table account;* ). Using one of the terminals you have opened, create the account table and populate it using the statements available in account.sql (available on the web site). Do not commit. Try and run ‘select \* from account’ on the other terminal. Now commit in the first terminal (enter the command *commit;*). Try and run again ‘select \* from account’ on the other terminal. What do you observe? Explain what happens.

b. Run stepwise T2 in one terminal and T3 in the other. (1 instruction at a time one terminal after the other). What do you observe (include a trace of the sessions on the mysql clients)? Is there a difference whether you start with T2 or T3? Explain what happens.

c. Start T3 in one window (do not commit). Start transaction T1 in the other terminal. What do you observe (include a trace of the sessions on the mysql clients)? Explain what happens.

d. Run T1 then T2. Run T2 then T1. What are the values you get from the query in T1? Start T2 in one terminal (execute the debit operation but not the credit). Run T1 in the other terminal. Resume the execution of T2. What happens? What value do you get for T1?

e. Run T1 then T4. Run T4 then T1. What are the values you get from the query in T1? Start T4 in one terminal (execute the debit operation but not the credit). Run T1 in the other terminal. Resume the execution of T4. What do you observe? What happens if you abort the transaction T4 (enter the command *rollback;*) Explain what happens.