DDB – Spring 2006

Lab#5 - Transactions

The experiment described below is quite simple. You might want to train for the lab exam and prepare a document with a consise and precise answer to the question.

Consider the following table:

Account(account_number, branch_name, balance). Check lab#3 for the account table (schema + data).

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.

The file account.sql allows you to populate the account table.

2. Open two terminals. Run a mysql client in each terminal. Use the database that has been assigned to you (you should be familiar with this procedure by now – but if you are not, you can check lab#1 for details on how to do this).

   By default, the MySQL server runs at the read committed isolation level. We have set the server so that it enforces **serializable** transactions.

   By default, the MySQL server 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: Run the following command:

   mysql> set autocommit = 0;

   REMEMBER TO RUN THIS COMMAND 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.

   MySQL does not support check and assertions, we thus cannot focus on consistency. Focusing on durability would require stopping/restarting the MySQL server. This is quite hard to control

as you are all accessing the same server. We thus focus on isolation and atomicity.

a. Check that the account table does not exist (it might be there from lab#3 – drop the table using drop table account; - You might also have to drop depositor that has a foreign key constraint on 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 (include a trace of the sessions on the mysql clients)? 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.