	Course	Databases and Information Systems 2014		
	Exercise Sheet	4		
	Points	–		
	Release Date	May 6 2014	Due Date	May 14 2014

Exercise 4: Synchronisation with Locking Protocols

Note


- In the following exercises, you can use the DB2 command line tool Backspace zu or other tools, e.g. Squirrel. Using Squirrel makes the administration of two parallel sessions and changing the auto-commit setting very easy.
- `get snapshot` is a so-called CLP command which can only be executed via db2. To this end, a connection to the database server has to be established first: `db2 attach to vsisls4 user vsisp<XX>`
- The syntax of `get snapshot` is represented incorrectly on the last slide of the presentation. The syntax for the command that displays only the locks held by a certain application is: `get snapshot for locks for application agentid <appl.handle>`. The Application Handle `<appl.handle>` can be looked up in the output of the command `list applications`.
- Submitting your results is not required; they are presented to one of the tutors during the exercise course in order to gain approval until May 14. Please take notes and have them ready when presenting your results.

Exercise 4.1: Isolation Levels

- Open a connection to database VSISP. What is the current isolation level?
- Create a simple table `OPK` with columns `ID` and `NAME` without declaring a primary key. Fill the table with some example data.
- Disable the auto-commit of the current connection, e.g. with the command `export DB2OPTIONS='+c'`. Query one row from table `OPK` and find out the currently held locks with the snapshot command. What locks are held by the application? Execute a commit.
- Commit the transaction and set the isolation level for the next transaction to `RS`. Repeat the steps from c).

Exercise 4.2: Lock Conflicts

- Open a second connection to the database (without disabling the auto-commit). Query some rows via the first connection (isolation level `RS`), for example all lines with `ID>5` (do not complete the transaction). Using the second connection, add a new row to the table that satisfies the selection predicate. What happens? Execute the query via the first connection again. What can be observed? Finally, execute a commit.
- Set the isolation level of the first connecting to `RR` (with manual commit). Now repeat the steps from a). What can be observed now? Which locks are held, before the commit of the first transaction? What does the table content look like, after the commit of the first transaction has been executed?
- Query one row from table `OPK` using the first connection (selection via the `ID` attribute) and change another row's `NAME` value using the other connection. What happens? Which locks are held? Finally, commit.

	Course	Databases and Information Systems 2014		
	Exercise Sheet	4		
	Points	–		
	Release Date	May 6 2014	Due Date	May 14 2014

- d) Create a second table **MPK**, again with columns **ID** and **Name**, and declare **ID** a primary key. Fill the table with example data. (Remember to execute a manual commit, if necessary.)
- e) Repeat the steps from c) with table **MPK** (use the two existing connections). What can be observed?

Exercise 4.3: Deadlocks

- a) Disable the auto-commit of the second connection. Query one row (e.g. **ID=1**) of table **OPK** using the first connection (isolation level **RR**) and another row (e.g. **ID=2**) using the second connection. Use the second connection to change the name value of the tuple queried by the first connection (**ID=1**) and use the first connection to change the name value of the tuple queried by the second connection (**ID=2**). What happens? Which locks are held? Can the transactions be completed successfully? Finally, commit both transactions.
- b) Set the isolation level of the second connection to **RS**. Repeat the steps from a). What can be observed now and how can it be explained? What is the status of the table, after you have executed a commit on both connections?