JDBC: Transactions

Last update: 2014-06-23

A transaction is a set of actions to be carried out as a single, atomic action. Either all of the actions are carried out, or none of them are.

The classic example of when transactions are necessary is the example of bank accounts. You need to transfer $100 from one account to the other. You do so by subtracting $100 from the first account, and adding $100 to the second account. If this process fails after you have subtracted the $100 fromt the first bank account, the $100 are never added to the second bank account. The money is lost in cyber space.

To solve this problem the subtraction and addition of the $100 are grouped into a transaction. If the subtraction succeeds, but the addition fails, you can "rollback" the fist subtraction. That way the database is left in the same state as before the subtraction was executed.

You start a transaction by this invocation:

```
connection.setAutoCommit(false);
```

Now you can continue to perform database queries and updates. All these actions are part of the transaction.

If any action attempted within the transaction fails, you should rollback the transaction. This is done like this:

```
connection.rollback();
```

If all actions succeed, you should commit the transaction. Committing the transaction makes the actions permanent in the database. Once committed, there is no going back. Committing the transaction is done like this:

```
connection.commit();
```

Of course you need a bit of try-catch-finally around these actions. Here is a an example:

```
Connection connection = ...
try{
    connection.setAutoCommit(false);

    // create and execute statements etc.

    connection.commit();
} catch(Exception e) {
    connection.rollback();
} finally {
    if(connection != null) {
        connection.close();
    }
}
```