Exercises

Michail Kazimianec
Werner Nutt
Damiano Somenzi

Transactions and JDBC Applications

For the first part of these exercises you are asked to experiment with the transaction processing features of PostgreSQL.

Exercise 1 Create a deadlock of two transactions. To do so, create an appropriate table. Then define two transactions, by writing them into your pgAdmin editor, that will lead to a deadlock. Execute them concurrently so that the deadlock shows up. Observe what happens.

Recall that a DBMS ensures *serializable execution* of transactions if for any transactions $T_1, \ldots, T_n$ that are executed concurrently there is some ordering $T_{i_1}, \ldots, T_{i_n}$ such that executing the transaction in this order yields identical outputs and the same final result.

Exercise 2 Test whether PostgreSQL is able to ensure serializable execution of transactions. To do so, run the boat reservation example of the lecture under the four different isolation levels. What are the differences? Does the level "serializable" guarantee serializability?

For the second part, you are asked to write several Java applications that manage a database through JDBC. An example application, showing how to connect to a database with JDBC, is found at:

`http://www.inf.unibz.it/~nutt/IDBs0910`
`/IDBExercises/ExampleJDBC.zip.`

Exercise 3 Choose a table in the database and write an application that (i) prints the number of records in the table and (ii) prints the number of records without NULL as attribute value. Is it possible to write the application in such a way that it executes only one query "`select * from` ⟨*tablename*⟩" to answer all the above questions?

Exercise 4 Write an application that inserts numerical data into a table. The application (i) retrieves the input values from the command line and (ii) uses a preparedStatement for insert operations. The application must restore the initial state of the table as soon as a negative number appears in the input.

Exercise 5 Write an application that connects to a database through JDBC and (i) prints the names of all tables in the database, (ii) for each table of the database prints its columns, (iii) for each column of the table prints its type.

Hints: You find sample code with the Java classes discussed in the lectures on the Web page with the coursework instructions. For further information on JDBC consult the online tutorial at

`http://java.sun.com/docs/books/tutorial/jdbc/TOC.html.`

To retrieve data about the database, like in the first exercise, note that JDBC has an interface DatabaseMetaData and that the method getMetaData, when applied to a connection, returns an instance of this interface.