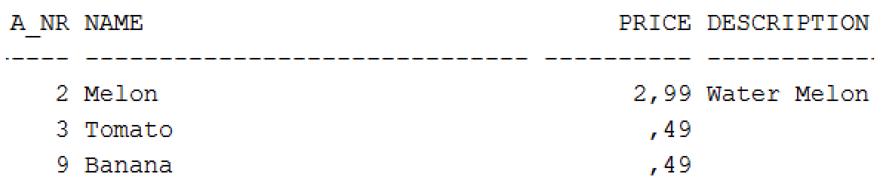
HAW Hamburg
Information Engineering
Databases
Lab /4/

Sarah Ammar **German Garcia Angus**

8.a

SELECT * FROM article WHERE price **=** (SELECT MAX (price) FROM article) OR price = (SELECT MIN (price) FROM article);
or
SELECT * FROM article WHERE price **IN** (SELECT MAX (price) FROM article) OR price IN (SELECT MIN (price) FROM article);
or
SELECT * FROM article WHERE price **=** ((SELECT MAX (price) FROM article) UNION (SELECT MIN (price) FROM article));

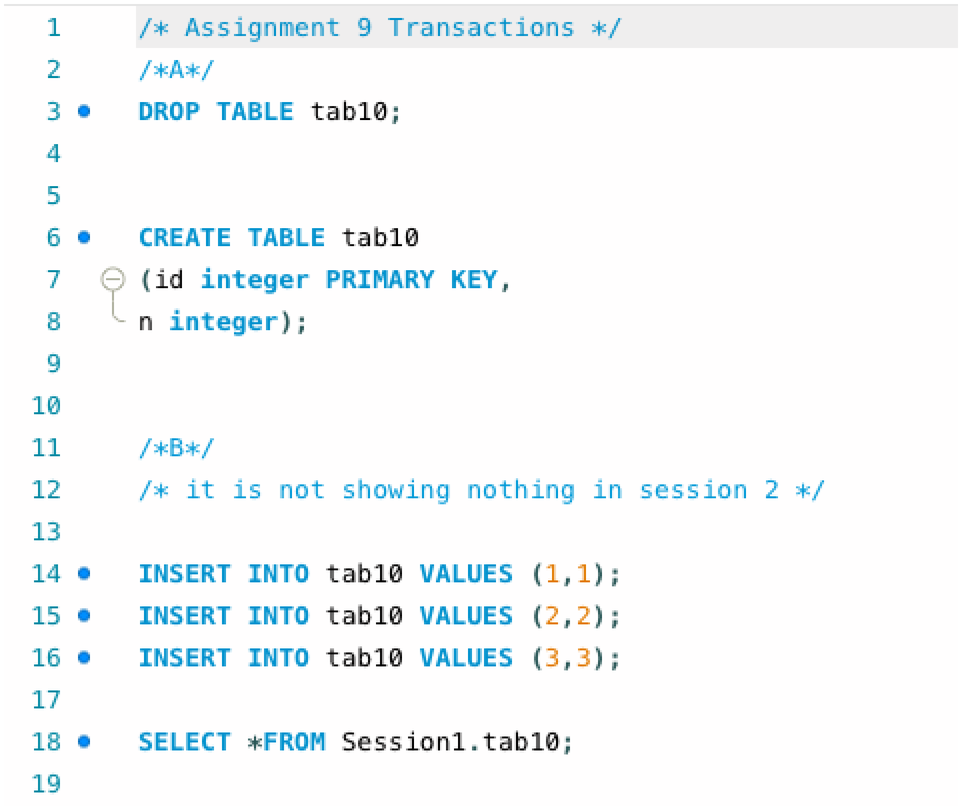


```
A_NR NAME                                PRICE DESCRIPTION
---- ------------------------------ ---------- -----------
   2 Melon                                2,99 Water Melon
   3 Tomato                                ,49
   9 Banana                                ,49
```

8.b

UPDATE person SET father = null WHERE name1 = 'Bob';

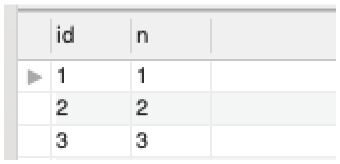
9.a Create a table.



```
/* Assignment 9 Transactions */
/*A*/
DROP TABLE tab10;


CREATE TABLE tab10
(id integer PRIMARY KEY,
n integer);


/*B*/
/* it is not showing nothing in session 2 */

INSERT INTO tab10 VALUES (1,1);
INSERT INTO tab10 VALUES (2,2);
INSERT INTO tab10 VALUES (3,3);

SELECT *FROM Session1.tab10;

```

<u>When is the table visible in session2?</u>

In session 2 The table is visible, before and after committing in session 1



id	n
1	1
2	2
3	3

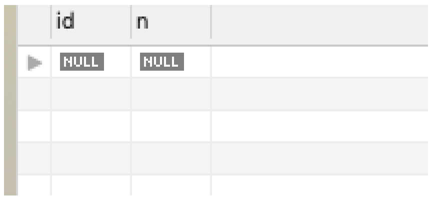
9.b Insert Values

<u>What is the table's content visible in session2 before and after you commit your changes in session1?</u>

- Before commit changes in Session 1 not showing data in Session 2 .

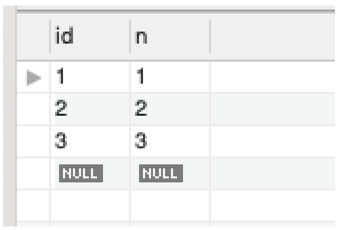
- After commit changes in Session 1, it shows all changes made, and can be accessible in Session 2 .

<u>Session 2 before commit in session 1</u>



id	n
NULL	NULL

<u>Session 2 after commit in session 1</u>

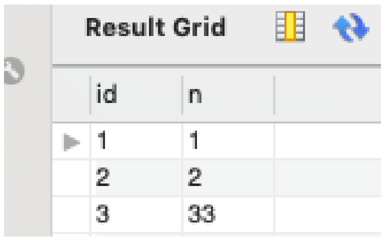


id	n
1	1
2	2
3	3
NULL	NULL

9.c Update Values

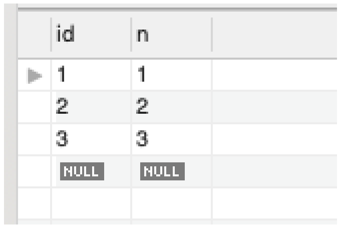
UPDATE TAB10 SET N=33 WHERE id=3;

Before rollback in Session 1



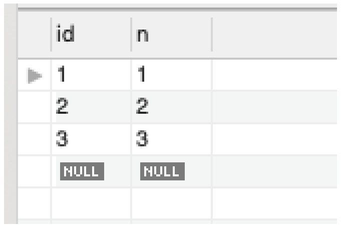
id	n
1	1
2	2
3	33

Before rollback in Session 2



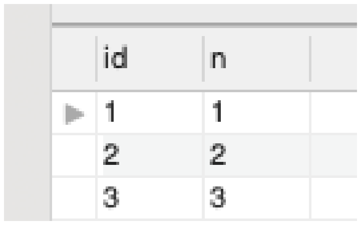
id	n
1	1
2	2
3	3
NULL	NULL

After rollback Session 1



id	n
1	1
2	2
3	3
NULL	NULL

After Rollback Session 2

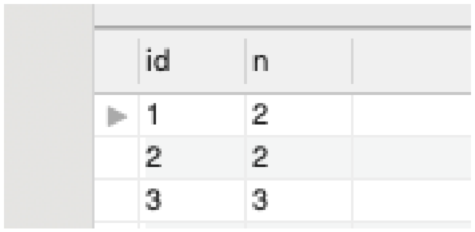


id	n
1	1
2	2
3	3

In session 1 no commit happened : the value won’t change by update neither after rollback.

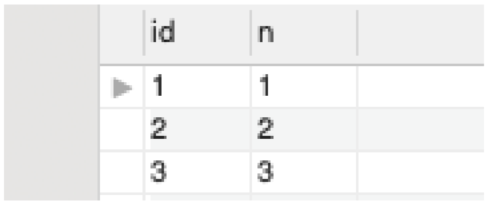
9d.

Output in session 1



id	n
1	2
2	2
3	3

Output in session 2



id	n
1	1
2	2
3	3

- Session 1 no commit was made after update and therefore Session 2 didn’t get new updated data (deadlock).

- In Session 2 if we tried to do (N*3) it would be the same result (1) having a deadlock as a result.

- Looks like we did 1*3, but we didn’t and (1) stayed the same, without being multiplied by (3).

- After commit in Session 1 the lock will be released and in Session 2 new data will be visible.

10.a

- **Main**

```
package de.haw.ie4lab4;

import java.io.IOException;
import java.sql.SQLException;

public class Main {

	// TODO: provide the correct class name of the driver!
	static final String driverName =
"oracle.jdbc.driver.OracleDriver";
	// TODO: provide the correct JDBC-URL for the HAW database!
	static final String url =
"jdbc:oracle:thin:@ora14.informatik.haw-hamburg.de:1521:inf14";

	static final String user     = MyDBUserPassword.user;
	static final String password = MyDBUserPassword.password;

	/**
	 * @param args
	 * @throws IOException
	 */
	public static void main(String[] args) throws IOException {
		DbHandler db = new DbHandler();

		try {
			db.connectDB(driverName, url, user, password);

			db.printOrderNumbers("Ringo");
			db.printOrderNumbers("John");
			db.printOrderNumbers("O'Hara");

			int orderNumber = 5;
			db.printInvoiceForOrder(orderNumber);

			db.insertNewCustomer(5, "Whoever");

			db.changeArticlePrice("Apple", 1.23);
```

```
            }
            catch (SQLException e)
            {
                  // TODO: print stack trace!
                  e.printStackTrace();
                  // TODO: Print nice error message using
System.err.println() and db.getSql()!
                  System.err.println(db.getSql());
            }
            finally
            {
                  // TODO: close connection
                  db.close();
            }
      }

}
```

- **DbHandler:**

```
package de.haw.ie4lab4;

import java.sql.Connection;
import java.sql.DatabaseMetaData;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class DbHandler {
  /**
   * Database connection
   */
  private Connection conn;

  /**
   * The current SQL statement
   */
  private String sql;

  /**
   * Getter for the current SQL statement
```

```
     *
     * @return the SQL statement
     */
    public String getSql() {
      return sql;
    }

    /**
     * Connect to the database.
     *
     * @param driverName
     *            - name of JDBC driver class
     * @param url
     *            - JDBC URL
     * @param user
     *            - DB user name
     * @param password
     *            - DB password
     * @throws SQLException
     */

    public void connectDB(String driverName, String url, String user,
String password) throws SQLException {
      System.out.println("Trying to connect to " + url);

      // TODO: connect to the DB!
      try {
           Class.forName(driverName);
        } catch (ClassNotFoundException e) {
           // TODO Auto-generated catch block
           e.printStackTrace();
        }
         conn = DriverManager.getConnection(url, user, password);

      // TODO: disable autoCommit!
         conn.setAutoCommit(false);
        //
      // Print success message and some meta data:
      //
      DatabaseMetaData metaData = conn.getMetaData();
      System.out.println("Connected to DB " + metaData.getURL() + " as
user " + metaData.getUserName());
      System.out.println(metaData.getDatabaseProductName() + " " +
metaData.getDatabaseMajorVersion() + "."
          + metaData.getDatabaseMinorVersion());
    }
```

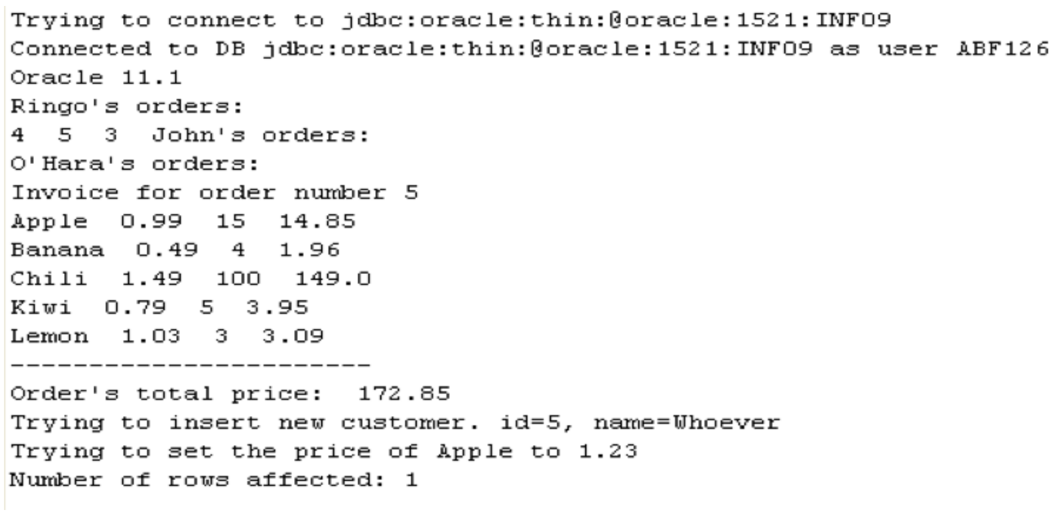
```
  /**
   * Close the connection
   */
  public void close() {
    /*
     * TODO: rollback the transaction (in real life, you'd want to
commit -> but
     * then you cannot call insertNewCustomer() twice.)
     */

    // TODO: close the connection (if it has been initialized)
       try {
                conn.rollback();
                conn.close();
          } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
          }
  }

  /**
   * Print the list of order numbers for the given customer
   *
   * @param customer
   *            - Name of customer
   * @throws SQLException
   */
  public void printOrderNumbers(String customer) throws SQLException {
     System.out.println("\n" + customer + "'s orders:");

    // TODO: SQL see assignment 7b-10
    Statement st = conn.createStatement();
    sql= "SELECT o_nr FROM customer c, orders o WHERE c.name = ? AND
c.c_id = o.c_id";
     PreparedStatement pst =conn.prepareStatement(sql);
     pst.setString(1, customer);
     ResultSet cursor = pst.executeQuery();
     while (cursor.next()) {
          int i = cursor.getInt(1);
          System.out.print(i+"  ");
          }
     cursor.close();
     st.close();
  }
  /**
   * Print an invoice for the given order. The invoice shall contain
every
   * single order item and the total price.
   *
   * @param orderNumber
   *            - value for o_nr
   * @throws SQLException
   */
  public void printInvoiceForOrder(int orderNumber) throws
SQLException {
    System.out.println("\nInvoice for order number " + orderNumber);

    // Optional: You could print customer information here!

    /*
     * TODO: For every order item, print the article name, the
article's price
     * per unit, the quantity, and the price of the order item. SQL
see
     * assignment 7b-14
     */
    Statement st = conn.createStatement();
     sql= "SELECT name,price , quantity , price * quantity FROM orders
o, order_item ort INNER JOIN article art ON ort.a_nr=art.a_nr WHERE
o.o_nr=? and ort.o_nr=? ORDER BY name ASC";
     PreparedStatement pst =conn.prepareStatement(sql);
     pst.setInt(1, orderNumber);
     pst.setInt(2, orderNumber);
     int n = pst.executeUpdate();
     ResultSet cursor = pst.executeQuery();
     double d3=0;
     while (cursor.next()) {
           // position in cursor starts at 1!
           String s1 = cursor.getString(1);
           double d1 = cursor.getDouble(2);
           int i1 = cursor.getInt(3);
           double d2 = cursor.getDouble(4);

           d3+=d2;
           //int i2 = cursor.getInt(2);
           System.out.println(s1+"  "+d1+"  "+i1+"  "+d2+"  ");
           //System.out.println(i2);
           }
     cursor.close();
     st.close();
```

```
      System.out.println("-----------------------");

     /*
      * TODO: Print the total price of the order. You can calculate the
sum via
      * SQL (see assignment 7b-15), or in Java.
      */
      System.out.println("Order's total price:  "+d3);
     //sql = "";

   }

   /**
    * Insert a new customer
    *
    * @param id
    *            - customer ID
    * @param name
    *            - customer name
    * @throws SQLException
    */
   public void insertNewCustomer(int id, String name) throws
SQLException {
     System.out.println("Trying to insert new customer. id=" + id + ",
name=" + name);

     // TODO: insert a new customer with the given values
     Statement st = conn.createStatement();
      sql = "INSERT INTO customer VALUES (?, ?)";
      PreparedStatement pst =conn.prepareStatement(sql);
      pst.setInt(1, id);
      pst.setString(2, name);
      pst.execute();
      st.close();
   }

   /**
    * Change the article's price
    *
    * @param articleName
    *            - identifies the article
    * @param price
    *            - the new price
    * @throws SQLException
    */
```

```
  public void changeArticlePrice(String articleName, double price)
throws SQLException {
    System.out.println("Trying to set the price of " + articleName + "
to " + price);
    //sql = "";
    int n = 0;

    // TODO: change the article's price
    sql = "UPDATE article SET price = ? WHERE name= ?";
      PreparedStatement pst =
      conn.prepareStatement(sql);
      pst.setDouble(1, price);
      pst.setString(2, articleName);
      n = pst.executeUpdate();

    System.out.println("Number of rows affected: " + n);
  }
}
```

Output :

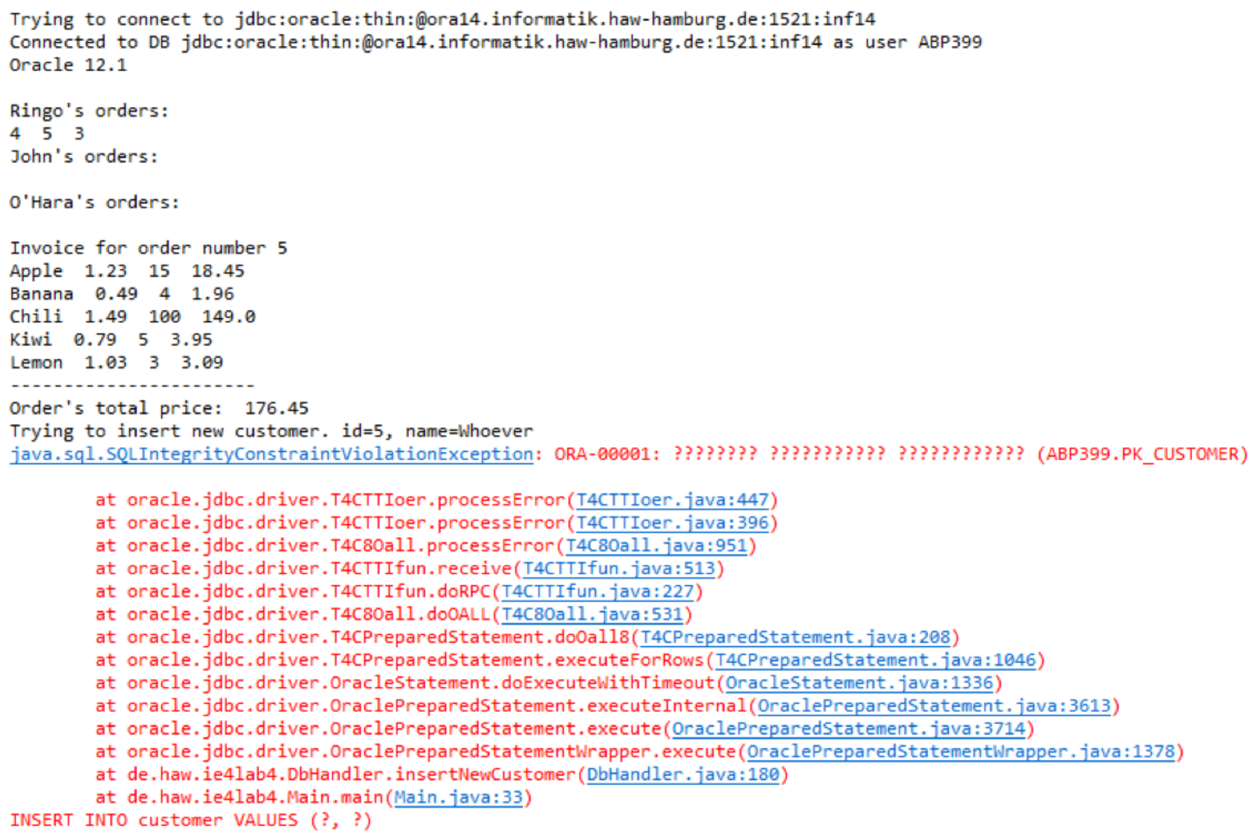
It is not showing Ringo's order, there is John's order 4 5 3, and near other orders there's nothing (we are trying to insert the prices)



```
Trying to connect to jdbc:oracle:thin:@oracle:1521:INFO9
Connected to DB jdbc:oracle:thin:@oracle:1521:INFO9 as user ABF126
Oracle 11.1
Ringo's orders:
4  5  3  John's orders:
O'Hara's orders:
Invoice for order number 5
Apple  0.99  15  14.85
Banana  0.49  4  1.96
Chili  1.49  100  149.0
Kiwi  0.79  5  3.95
Lemon  1.03  3  3.09
-----------------------
Order's total price:  172.85
Trying to insert new customer. id=5, name=Whoever
Trying to set the price of Apple to 1.23
Number of rows affected: 1
```

Output :

“Trying to insert the price of John’s and O’Hara’s Orders … Problem it is not showing up



```
Trying to connect to jdbc:oracle:thin:@ora14.informatik.haw-hamburg.de:1521:inf14
Connected to DB jdbc:oracle:thin:@ora14.informatik.haw-hamburg.de:1521:inf14 as user ABP399
Oracle 12.1

Ringo's orders:
4  5  3
John's orders:

O'Hara's orders:

Invoice for order number 5
Apple  1.23  15  18.45
Banana  0.49  4  1.96
Chili  1.49  100  149.0
Kiwi  0.79  5  3.95
Lemon  1.03  3  3.09
-----------------------
Order's total price:  176.45
Trying to insert new customer. id=5, name=Whoever
java.sql.SQLIntegrityConstraintViolationException: ORA-00001: ???????? ??????????? ???????????? (ABP399.PK_CUSTOMER)

	at oracle.jdbc.driver.T4CTTIoer.processError(T4CTTIoer.java:447)
	at oracle.jdbc.driver.T4CTTIoer.processError(T4CTTIoer.java:396)
	at oracle.jdbc.driver.T4C8Oall.processError(T4C8Oall.java:951)
	at oracle.jdbc.driver.T4CTTIfun.receive(T4CTTIfun.java:513)
	at oracle.jdbc.driver.T4CTTIfun.doRPC(T4CTTIfun.java:227)
	at oracle.jdbc.driver.T4C8Oall.doOALL(T4C8Oall.java:531)
	at oracle.jdbc.driver.T4CPreparedStatement.doOall8(T4CPreparedStatement.java:208)
	at oracle.jdbc.driver.T4CPreparedStatement.executeForRows(T4CPreparedStatement.java:1046)
	at oracle.jdbc.driver.OracleStatement.doExecuteWithTimeout(OracleStatement.java:1336)
	at oracle.jdbc.driver.OraclePreparedStatement.executeInternal(OraclePreparedStatement.java:3613)
	at oracle.jdbc.driver.OraclePreparedStatement.execute(OraclePreparedStatement.java:3714)
	at oracle.jdbc.driver.OraclePreparedStatementWrapper.execute(OraclePreparedStatementWrapper.java:1378)
	at de.haw.ie4lab4.DbHandler.insertNewCustomer(DbHandler.java:180)
	at de.haw.ie4lab4.Main.main(Main.java:33)
INSERT INTO customer VALUES (?, ?)
```