ICON OF JAVA

ADV. JAVA

JDBC

9.Transaction Management

Mr. Nagoor Babu M.Tech

[ICON of JAVA]
(Sun certified & Realtime Expert)
Ex. HCL Employee
Trained Lakhs of Students
for last 14 years across INDIA

Transaction Management

1. Atomicity
2. Consistency
3. Isolation
4. Durability

Transaction Management:-

Transaction:- Transaction is an unit of work performed by the front end application on back end system.
1. Deposit some amount in an account.
2. Withdraw some amount from an account.
3. Transfer some amount from one account to another account.

In database applications, every transaction must satisfy the following 4 properties.
1. Atomicity
2. Consistency
3. Isolation
4. Durability

1. Atomicity:-

In general we are able to perform multiple number of operations in a particular transaction, where performing all the operations or performing none of the operations is called as Atomicity property.
If we perform all the operations successfully in a transaction then the state of the transaction should be success.

If we perform none of the operations in a transaction then state of the transaction should be failure.

Note: While performing operations in a transaction, if we encounter a problem with any operation then we should perform rollback operation over all the operations which are available in the transactions.

2. Consistency:

In database applications, before the transaction and after the transaction the state of the database must be stable is called as Consistency property.

To achieve consistency we will perform some checkings called as Consistency Checkings.
Consistency checkings will check correctness of the results after the transactions.

3. Isolation:

In database applications, if we perform more than one transaction on a single data item then that transactions are called as Concurrent Transactions. In concurrent transactions, one transaction execution should not be effect to the another transaction. This nature of the transaction is called as Isolation.

While performing concurrent transactions, there may be a chance to get concurrency problems, to resolve these problems we will use Isolation Levels in database application.

4. Durability:

In database applications, after committing the transaction if we have any catastrophic failures like power failure, operating system crashing and so on then we have to preserve the modifications which we performed on the database during respective transaction. This nature of the transaction is called as Durability.

In Jdbc applications, when we establish connection then automatically that connection will have a default mode i.e. auto commit mode. In case of auto commit mode, when we submit SQL query to the connection, where connection will carry that SQL query to Database Engine and make the Database Engine to execute that SQL query and store the results on to the database table permanently.

The above auto commit nature of connection may not satisfy the transaction's atomicity property. To preserve transaction atomicity property in Jdbc applications we have to change connection's auto commit mode.

To change connection's auto commit mode we have to use the following method from Connection.
public void setAutoCommit(boolean b)throws SQLException
If b==true then the connection will be in auto commit mode else the connection will be in non-auto commit mode.

Ex: con.setAutoCommit(false);

If we change connection's auto commit mode then we have to perform either commit or rollback operations to complete the transactions.
To perform commit and roll back operations we have to use the following methods from Connection.

public void commit()throws SQLException
public void rollback()throws SQLException

Note: In case of connection's non-auto commit mode, when we submit SQL query to the connection then connection will send that SQL query to Database Engine and make the Database Engine to execute that SQL query and store the results on to the database table temporarily. In this case, Database Engine may wait for commit or rollback signal from client application to complete the transactions.

JdbcApp32:

```
package com.durgasoft;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
public class JdbcApp32 {

    public static void main(String[] args) {
        Connection con=null;
        try {
            Class.forName("oracle.jdbc.OracleDriver");
            con=DriverManager.getConnection("jdbc:oracle:thin:
            @localhost:1521:xe","system", "durga");
            con.setAutoCommit(false);
            Statement st=con.createStatement();
            st.executeUpdate("insert into student values(111,'AAA',78)");
            st.executeUpdate("insert into student values(222,'BBB',87)");
            st.executeUpdate("insert into student values(333,'CCC',96)");
            con.commit();
            System.out.println("Transaction Success");
        } catch (Exception e) {
            try{
            con.rollback();
                System.out.println("Transaction Failure");
            }catch(Exception e1){
                e1.printStackTrace();
            }
            //e.printStackTrace();
        }
}

}
```

JdbcApp33:

```
package com.durgasoft;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

public class JdbcApp33 {

    public static void main(String[] args) {
        Connection oracle_conn=null;
        Connection mysql_conn=null;
        try {
                Class.forName("oracle.jdbc.OracleDriver");
                Class.forName("com.mysql.jdbc.Driver");
                oracle_conn=DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe","system","durga");
                        mysql_conn=DriverManager.getConnection
                ("jdbc:mysql://localhost:3306/durgadb","root","root");

                oracle_conn.setAutoCommit(false);
                mysql_conn.setAutoCommit(false);

                Statement oracle_st=oracle_conn.createStatement();
                Statement mysql_st=mysql_conn.createStatement();
```

```
BufferedReader br=new BufferedReader
(new InputStreamReader(System.in));
System.out.print("Source Account     :");
String source_Account=br.readLine();
System.out.print("Target Account     :");
String target_Account=br.readLine();
System.out.print("Amount To Transfer  :");
int trans_Amt=Integer.parseInt(br.readLine());

int oracle_rowCount=oracle_st.executeUpdate
("update account set balance=balance-"+trans_Amt+" where
accNo='"+source_Account+"'");
int mysql_rowCount=mysql_st.executeUpdate("update account set
balance=balance+"+trans_Amt+" where accNo='"+target_Account+"'");
if((oracle_rowCount==1) && (mysql_rowCount==1)){
        oracle_conn.commit();
        mysql_conn.commit();
        System.out.println(trans_Amt+" Transfered Successfully from
        "+source_Account+" to "+target_Account);
        System.out.println("Transaction Success");
}else{
        oracle_conn.rollback();
        mysql_conn.rollback();
        System.out.println("Transaction Failure");
}
} catch (Exception e) {
try {
        oracle_conn.rollback();
        mysql_conn.rollback();
        System.out.println("Transaction Failure");
} catch (Exception e2) {
        e2.printStackTrace();
}
}

    }

}
```