Transaction Management in JDBC

Transaction represents **a single unit of work**.

The ACID properties describes the transaction management well. ACID stands for Atomicity, Consistency, isolation and durability.

Atomicity means either all successful or none.

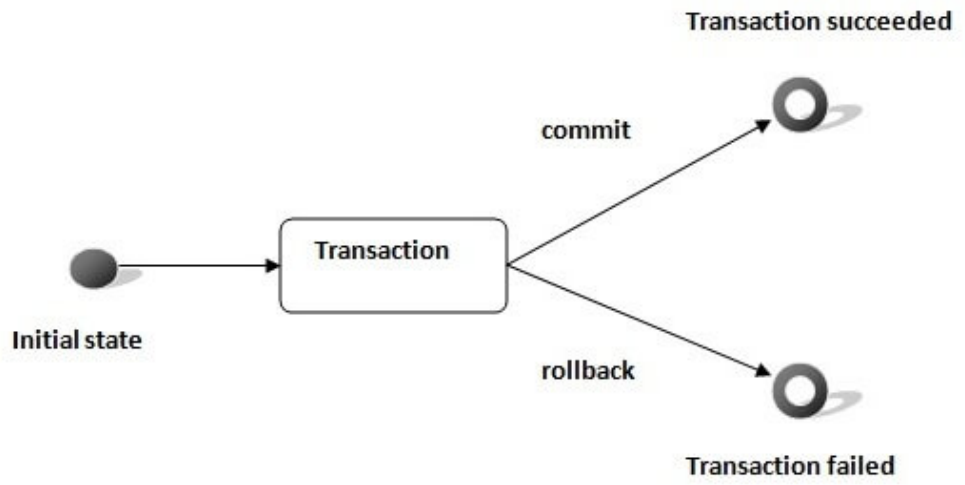
Consistency ensures bringing the database from one consistent state to another consistent state.

Isolation ensures that transaction is isolated from other transaction.

Durability means once a transaction has been committed, it will remain so, even in the event of errors, power loss etc.

Advantage of Transaction Mangaement

fast performance It makes the performance fast because database is hit at the time of commit.



In JDBC, **Connection interface** provides methods to manage transaction.

Method	Description
void setAutoCommit(boolean status)	It is true bydefault means each transaction is committed bydefault.
void commit()	commits the transaction.
void rollback()	cancels the transaction.

Simple example of transaction management in jdbc using Statement

Let's see the simple example of transaction management using Statement.

```
import java.sql.*;
class FetchRecords{
public static void main(String args[])throws Exception{
Class.forName("oracle.jdbc.driver.OracleDriver");
```

```
Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
con.setAutoCommit(false);

Statement stmt=con.createStatement();
stmt.executeUpdate("insert into user420 values(190,'abhi',40000)");
stmt.executeUpdate("insert into user420 values(191,'umesh',50000)");

con.commit();
con.close();
}}
```

If you see the table emp400, you will see that 2 records has been added.

Example of transaction management in jdbc using PreparedStatement

Let's see the simple example of transaction management using PreparedStatement.

```
import java.sql.*;
import java.io.*;
class TM{
public static void main(String args[]){
try{

Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
con.setAutoCommit(false);

PreparedStatement ps=con.prepareStatement("insert into user420 values(?,?,?)");

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
while(true){

System.out.println("enter id");
String s1=br.readLine();
int id=Integer.parseInt(s1);

System.out.println("enter name");
String name=br.readLine();

System.out.println("enter salary");
String s3=br.readLine();
int salary=Integer.parseInt(s3);
```

```
ps.setInt(1,id);
ps.setString(2,name);
ps.setInt(3,salary);
ps.executeUpdate();

System.out.println("commit/rollback");
String answer=br.readLine();
if(answer.equals("commit")){
con.commit();
}
if(answer.equals("rollback")){
con.rollback();
}



System.out.println("Want to add more records y/n");
String ans=br.readLine();
if(ans.equals("n")){
break;
}

}
con.commit();
System.out.println("record successfully saved");

con.close();//before closing connection commit() is called
}catch(Exception e){System.out.println(e);}


}}
```

It will ask to add more records until you press n. If you press n, transaction is committed.