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.

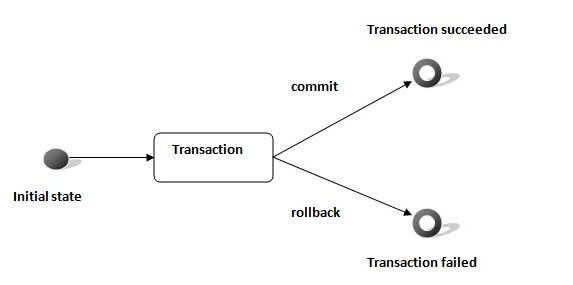
**Consistency** ensures bringing the database from one consistent state to another consistent state.

**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.

1. **import** java.sql.\*;
2. **class** FetchRecords{
3. **public** **static** **void** main(String args[])**throws** Exception{
4. Class.forName("oracle.jdbc.driver.OracleDriver");
5. Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
6. con.setAutoCommit(**false**);
8. Statement stmt=con.createStatement();
9. stmt.executeUpdate("insert into user420 values(190,'abhi',40000)");
10. stmt.executeUpdate("insert into user420 values(191,'umesh',50000)");
12. con.commit();
13. con.close();
14. }}

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.

1. **import** java.sql.\*;
2. **import** java.io.\*;
3. **class** TM{
4. **public** **static** **void** main(String args[]){
5. **try**{
7. Class.forName("oracle.jdbc.driver.OracleDriver");
8. Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
9. con.setAutoCommit(**false**);
11. PreparedStatement ps=con.prepareStatement("insert into user420 values(?,?,?)");
13. BufferedReader br=**new** BufferedReader(**new** InputStreamReader(System.in));
14. **while**(**true**){
16. System.out.println("enter id");
17. String s1=br.readLine();
18. **int** id=Integer.parseInt(s1);
20. System.out.println("enter name");
21. String name=br.readLine();
23. System.out.println("enter salary");
24. String s3=br.readLine();
25. **int** salary=Integer.parseInt(s3);
27. ps.setInt(1,id);
28. ps.setString(2,name);
29. ps.setInt(3,salary);
30. ps.executeUpdate();
32. System.out.println("commit/rollback");
33. String answer=br.readLine();
34. **if**(answer.equals("commit")){
35. con.commit();
36. }
37. **if**(answer.equals("rollback")){
38. con.rollback();
39. }

42. System.out.println("Want to add more records y/n");
43. String ans=br.readLine();
44. **if**(ans.equals("n")){
45. **break**;
46. }
48. }
49. con.commit();
50. System.out.println("record successfully saved");
52. con.close();//before closing connection commit() is called
53. }**catch**(Exception e){System.out.println(e);}
55. }}

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