



	id	name	age	address
1	1	sachin	49	MI
2	2	dhoni	41	CSK
3	3	kohli	35	RCB
4	4	Gill	23	GT
5	5	sky	32	MI
6	6	jadeja	33	CSK
7	7	rohit	38	MI
8	8	rahul	31	LSG

- ↳ JDBCCrudApp
 - ↳ JRE System Library [JavaSE-1.8]
 - ↳ src
 - ↳ in.neuron.controller
 - ↳ TestApp.java
 - ↳ in.neuron.daofactory
 - ↳ StudentDaofactory.java
 - ↳ in.neuron.dto
 - ↳ in.neuron.persistence
 - ↳ IStudentDao.java
 - ↳ StudentDaoImpl.java
 - ↳ in.neuron.properties
 - ↳ in.neuron.service
 - ↳ IStudentService.java
 - ↳ StudentServiceImpl.java
 - ↳ in.neuron.servicefactory
 - ↳ in.neuron.util
 - ↳ mysqlJlib

TransactionManagement

=====

Process of combining all the related operations into a single unit and executing on the rule "either all or none" is referred as "Transaction Management".

Case1 : Fund transfer

1. debit funds from sender account (update query with bal=bal-amt)
2. credit funds into receiver account(update query with bal=bal+amt)

All operations should be performed as single unit.
if money from sender account is debited, and if the money doesn't reach to receiver's account then there may be a chance of "Data inconsistency problem".

Case2: Movie Ticket Reservation

1. verify the status
2. Reserve the ticket
3. Payment
4. Issue ticket

All operations should be performed as single unit.
If some operations succeed and some operations fail then there may be "Data inconsistency problem".

Transaction Properties

=====

Every Transaction should follow the following four ACID properties

- a. A -> Atomicity
Either all operations should be done or none.
- b. C -> Consistency
It should ensure bringing the database from one state to another state.
- c. I -> Isolation
It ensures the transactions are isolated from other transactions.
- d. D -> Durability
It means once the transaction is committed, the results are permanent even in the case of system restart, errors, etc..

What are the types of Transactions available?

- a. Local Transaction
- b. Global Transaction

a. Local Transaction

All operations in transaction are executed over the same database.
eg: Funds transfer from one account to another account where the both the accounts are in the same bank.

b. Global Transaction

All operations in transaction are executed over the different database.
eg: Funds transfer from one account to another account where the accounts are related to different banks.

Note:

JDBC supports only local transactions.

If we want global transactions then we need to go for frameworks like Spring/hibernate or EJB's.

Process of transaction management in JDBC

=====

1. Disable the autocommit nature of JDBC
 `connection.setAutoCommit(false);`

2. If all operations are completed means then we can commit the transaction using the following method
 `connection.commit();`

3. if any sql query fails, then we need to rollback the operations which are already completed using the following method
 `connection.rollback();`

SavePoint(I)

=====

Within a transaction, if we want to rollback a particular group of operations based on some condition then we need to use

 savepoint

 a. getting savepoint using the following method

`Savepoint sp = connection.setSavePoint();`

 b. To perform rollback operation for a particular group of operations w.r.t savepoint then we need to use rollback .

`connection.rollback(sp);`

 c. we can release the savepoint or delete the savepoint as shown below

`connection.releaseSavePoint(sp);`

Case study

=====

`connection.setAutoCommit(false);`

`operation-1`

`operation-2`

`operation-3`

`SavePoint sp = connection.setSavePoint();`

`operation-4`

`operation-5`

`if(balance<1000)`

`connection.rollback(sp);`

`else`

`connection.releaseSavepoint(sp);`

`connection.commit();`

if balance< 1000 then operation 4,5 will be rolledback,otherwise all the operations will be committed.

eg:

`connection.setAutoCommit(false);`

`st.executeUpdate("insert into politicians values('BJP', 'Modi'));`

`st.executeUpdate("insert into politicians values('TRS', 'KCR'));`

`SavePoint sp = connection.setSavePoint();`

`st.executeUpdate("insert into policititions values('BJP', 'siddu'));`

`connection.rollback(sp);`

`;;;`

`;;;`

`connection.releaseSavePoint(sp);`

`;;;`

`;;;`

`connection.commit();`

Types of ResultSet

=====

refer: ****.png

`absolute()` -> it works from the BFR or from ALR.

`relative()` -> it works w.r.t current position.

In both the methods positive means move in forward direction, negative means move in backward direction.

Note:

`rs.last()` and `rs.absolute(-1)` both are equal

`rs.first()` and `rs.absolute(1)` both are equal