# JDBC

## API (Application programming interface)

API stands for application programming interface, which is a set of definitions and protocols for building and integrating application software.

## JDBC

JDBC stands for Java Database Connectivity. JDBC is a Java API to connect and

execute the query with the database. JDBC API uses JDBC drivers to connect with the database.

## 5 steps to connect any java application

**1) Register the driver class**

Class.forName("oracle.jdbc.driver.OracleDriver");

class.forName -> this will get a string as parameter & creates instance of driver and it will register with driverManager.

**2) Create the connection object**

Connection con = DriverManager.getConnection(url, name, password);

**3) Create the Statement object**

**(old)**

Statement s = con.createStatement();

**(new)**

PreparedStatement ps = con.prepareStatement("query");

ps.setInt(1, empId);

**4) Execute the query**

**for retrive -** ResultSet rs = ps.executeQuery();

**for update -** int x = ps.executeUpdate();

**5) Close the connection object**

it automatically closes in try catch block.

## Connection

- Connection interface maintains a session with database. It can be used for transaction management.
- It provides methods that returns the instance of Statement, PreparedStatement, CallableStatement & DatabaseMetaData.

## Transaction Management

A transaction is a set of logically related data operations. For example, you are transferring money from your bank account to your friend's account.

If it fails to transers in half way then our data will be in in-consistant state. So me use transaction to make it **Atomic**.

**Commit:** If all the operations in a transaction are completed successfully then commit those changes.

**Rollback:** If any of the operation fails then rollback all the changes done by previous operations.

## ResultSet

The resultset object represents a row of table. It can be used to change the curser pointer and get the information from the database.

## Statement

The Statement interface accepts strings as SQL queries. Thus,

- the code becomes **less readable**.
- it is vulnerable to **SQL injection**.
- the query will not appear as the same to the database and it will prevent cache usage.
- the Statement interface is suitable for DDL queries like CREATE, ALTER and DROP.

If we are going to fire query only once then we can use statement as it is faster than prepared statement.

**DB engine steps;**

1 . compile query (check the systex)

2. query plan generation (how the engine is planning to get data)

3. query optimisation (0ptimise planning)

### PreparedStatement

the PreparedStatement extends the Statement interface.

It has methods to bind various object types. Hence, the code becomes easy to understand.

- the code is **readable**.
- it is not vulnerable to **SQL injection**.
- Good  cache usage
- this is used to fetch and update data.

### Data Access Object Pattern

DAO pattern is used to separate low level data accessing API or operations from high level business services.

## MAVEN

### What are build tools?

*Build tool is a program that automates the process of compiling, testing ,packaging and deploying of source code in the most efficient manner. It requires very little human interaction helping developers improve their productivity.*

### maven (appache foundation)

*it is a powerful project management tool for java(or) or a build tool, which automatically does following work.*

1.write a source code

2. add jar files to the classpath of our project.

3.compile our java code.

4.prepare test cases, compile the test cases and run

4.1 : put all the classes and test cases in proper folders (properties file, image file, html files)

5.generte the report of the test case

6.build the jar or war file.

*7.deploy the jar, war file to the server*

***pom.xml :- project object model***

*POM is  Project Object Model. The pom.xml file contains information of project and configuration information for the maven to build the project such as dependencies, build directory, source directory, test source directory, plugin, goals etc.*

*it treats our entire project as a object. It provides dependencies required for  the project, we need to mention the dependencies in the POM.*

**Maven terminology:-**

===================

**1.artifact:-**

--An artifact is an outcome in Maven,it can be a file like .class or jar or war etc.

**2.Archetype:-**

--An atchetype is an project template for creating similar type of project in maven.

**3.Groupid:-**

--it is an id used to identify the atrifacts of perticular organization(package name).

**4.artifactid:-**

--it is an id for outcome which is going to be generated by Maven,maven generates an artifact with same name as root folder name,so we can say 'artifactid' is a root folder.

**Maven Repository**

--A repository is a store,where maven maintains maven plugins,archtypes and all jars,used for building diff kinds of projects.

--Maven repository r of 3 types:-

1.Central repository

2.Remote repository

3.Local repository

**1.Central repository:-**

----------------------

--it is maven's own repository in which it maintains all kinds of projects related plugins,archtypes and jars.

https://repo1.maven.org/maven2/

mvnrepository :- it is a website:

**2.Remote repository:-**
----------------------

--it is maintained within organization for sharing plugins,archtypes and jars for multiple projects within organization

**3.Local repository:-(inside user computer)**
--------------------

--it is created by maven to store plugins,archtypes and jars that r downloaded from either Remote or Central repository.

Note:- Maven downloads for 1st time into local repository and for next time it will use from local repository.

--A remote repository also downloads from central repository for 1st time.

## Maven build life cycle:-
-----------------------

--Maven build life-cycle contains diff phases:-

**1.Validate:-**

--Maven checks directory structure is valid or not and it has pom.xml(project object model) file is there or not.

**2.Compile:-**

--Maven compiles all source code files by downloading and adding required jars in classpath.

**3.test-compile:-**

--Maven compiles test cases.

**4.test:-**

--maven will run the test cases and it will show how many r sucess and how many r failed.

**5.package:-**

--maven bundles all the classes and other relates jar files into a jar/war/ear. inside 'target' folder

**6.install:-**

--maven stores the jar/war/ear into the local repository

**7.deploy:-**

--Maven stores the jar/war/ear into the central repository

**8.clean:-**

--maven deletes or removes all the files that r generated in previous build.

## 4) What are the JDBC API components?

The java.sql package contains following interfaces and classes for JDBC API.

**Interfaces:**

- **Connection:** The Connection object is created by using getConnection() method of DriverManager class. DriverManager is the factory for connection.
- **Statement:** The Statement object is created by using createStatement() method of Connection class. The Connection interface is the factory for Statement.
- **PreparedStatement:** The PrepareStatement object is created by using prepareStatement() method of Connection class. It is used to execute the parameterized query.
- **ResultSet:** The object of ResultSet maintains a cursor pointing to a row of a table. Initially, cursor points before the first row. The executeQuery() method of Statement interface returns the ResultSet object.
- **ResultSetMetaData:** The object of ResultSetMetaData interface cotains the information about the data (table) such as numer of columns, column name, column type, etc. The getMetaData() method of ResultSet returns the object of ResultSetMetaData.
- **DatabaseMetaData:** DatabaseMetaData interface provides methods to get metadata of a database such as the database product name, database product version, driver name, name of the total number of tables, the name of the total number of views, etc. The getMetaData() method of Connection interface returns the object of DatabaseMetaData.
- **CallableStatement:** CallableStatement interface is used to call the stored procedures and functions. We can have business logic on the database through the use of stored procedures and functions that will make the performance better because these are precompiled. The prepareCall() method of Connection interface returns the instance of CallableStatement.

**Classes:**

- **DriverManager:** The DriverManager class acts as an interface between the user and drivers. It keeps track of the drivers that are available and handles establishing a connection between a database and the appropriate driver. It contains several methods to keep the interaction between the user and drivers.

- **Blob:** Blob stands for the binary large object. It represents a collection of binary data stored as a single entity in the database management system.
- **Clob:** Clob stands for Character large object. It is a data type that is used by various database management systems to store character files. It is similar to Blob except for the difference that BLOB represent binary data such as images, audio and video files, etc. whereas Clob represents character stream data such as character files, etc.
- **SQLException** It is an Exception class which provides information on database access errors.

## Frequently Asked JDBC Interview Questions

### Q #1) What is JDBC?

**Answer:** Java Database Connectivity is unofficially known as JDBC. It is used to perform DB operations in Database from Java application. It supports interaction with any kind of DB like Oracle, MySQL, MS Access, etc.

### Q #2) What is the use of the JDBC driver?

**Answer:** It is a software component and is used to make the Java application to interact with the Database.

### Q #3) What are the different types of drivers in JDBC?

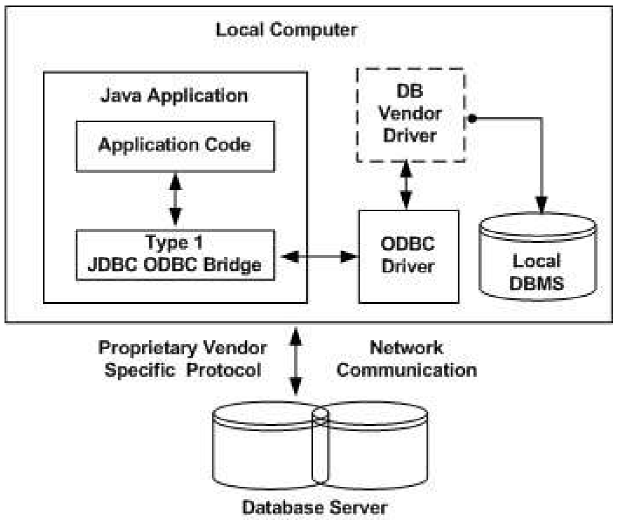
**Answer:** There are 4 different JDBC drivers out there in the market.

**They are:**

- **Type I:** JDBC – ODBC Bridge
- **Type II:** Native API – Half Java Driver
- **Type III:** Network Protocol– Totally Java Driver
- **Type IV:** Thin Driver- Totally Java Driver

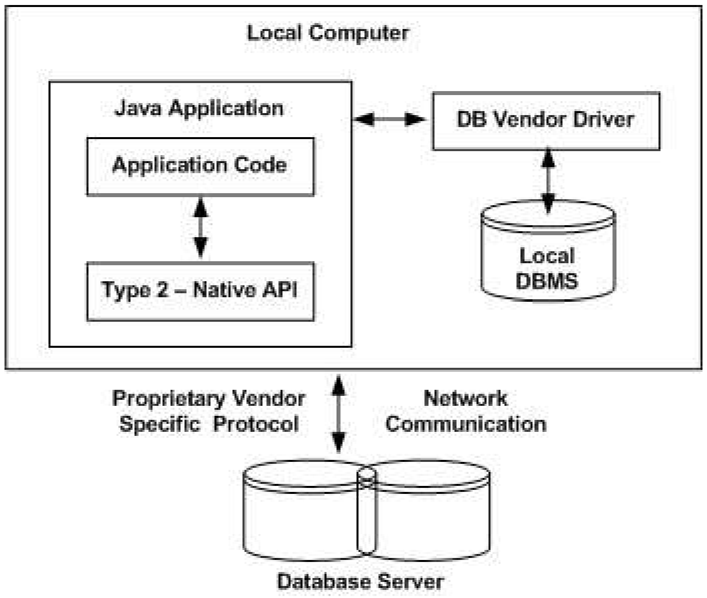
**Type I: JDBC-ODBC Bridge**

JDBC-ODBC bridge is going to behave as an interface between the client and the DB server. The client should put the JDBC-ODBC driver in it. The database ought to support the ODBC driver. If we are not concerned about the driver installation within the client system, we will use this driver.



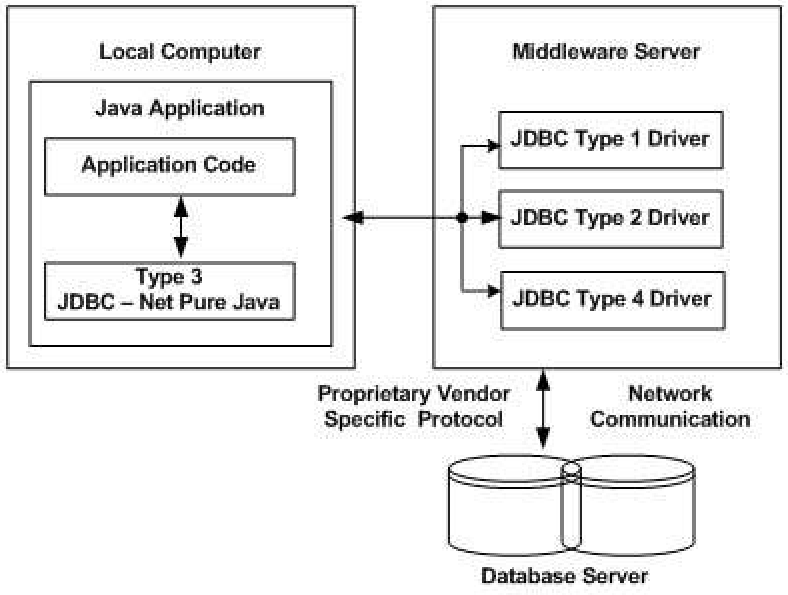
**Type II: Native API: Half Java Driver**

It is almost like a JDBC-ODBC driver. Rather than an ODBC driver, we are using native API here. Libraries of the client-side database are used.



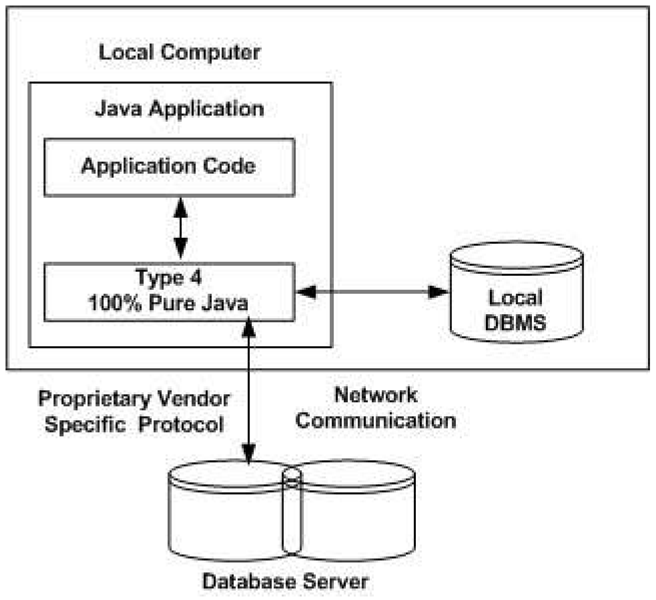
**Type III: Network Protocol**

It works like a 3-tier approach to access the database. An intermediate server will be used to connect to DB. JDBC method calls send data to an intermediate server then the server will communicate with DB.



**Type IV: Thin Driver**

It is absolutely written in Java. It explicitly converts JDBC method calls into the vendor-specific database protocol. Nowadays, Database merchant itself is providing this type of driver for their customers. So programmers don't rely on other sources. It gives higher performance than the other drivers.



## Q #4) Which type of JDBC driver is used by most people?

**Answer:** Type IV Thin driver is used in most of the applications. It is developed by the database vendor itself so the developers can use it directly without depending on any other sources. It allows for simple and easy development. It gives higher performance than the other drivers.

## Q #5) What are the types of JDBC Architecture?

**Answer:** JDBC supports 2 kinds of processing models to access the DB.

**They are:**

- **Two-tier Architecture:** Here Java programs explicitly connect with DB. We don't need any mediator like applications server to connect with DB except the JDBC driver. It is also known as a client-server architecture.
- **Three-tier Architecture:** It is totally inverse of two-tier architecture. There will be no explicit communication between the JDBC driver or Java program and Database. An application server is used as a mediator between them. Java program will send the request to an application server, and the server will send it and receive the response to/ from DB.

## Q #6) What are the components of JDBC?

**Answer:** There are 4 major components that are available in JDBC.

**They are:**

1. JDBC API
2. JDBC Driver Manager
3. JDBC Test Suite
4. JDBC – ODBC Bridge

## Q #7) What are the steps to connect with JDBC?

**Answer:** There are 6 basic steps to connect with DB in Java. **These are enlisted below:**

- Import package
- Load driver
- Establish connection
- Creation and execution of the statement
- Retrieve results
- Close connection

## Q #8) Which data types are used to store the image and the file in the database table?

**Answer:**

- **BLOB data type** is used to store the image in DB. We can store videos and audios as well in the BLOB data type. It is used to store the binary type of data.
- **CLOB data type** is used to store the file in DB. It is used to store the character type of data.

## Q #9) What is DriverManager in JDBC?

**Answer:** DriverManager is an in-built class that is present in the java.sql package. It will be used as a mediator between the Java Application and DB, which we are connecting/using in our code. As a first step, we need to register or load the driver with DriverManager. Then the driver will be available to use in the application.

The main function of DriverManager is to load the driver class of the Database and create a connection with DB.

**There are 2 ways to register or load the driver:**

- Class.forName()
- DriverManager.registerDriver()

## Q #10) What is the difference between Statement and PreparedStatement interfaces.

**Answer: The below table explains the differences:**

| STATEMENT | PREPARED STATEMENT |
|---|---|
| vill be mainly used for executing static SQL atements | It will be mainly used for executing pre-compiled SQL statements |
| It will not accept parameters at runtime | It will accept different parameters at runtime |
| Its performance is less compared to preparedStatement | Its performance is higher than Statement since it is executing the precompiled SQL statements |
| It is appropriate for executing DDL statements such as CREATE, DROP, ALTER and TRUNCATE | It is appropriate for executing DML statements such as INSERT, UPDATE, and DELETE |
| It can’t be used for storing or retrieving image and file in DB | It can be used for storing or retrieving image and file in DB |
| It enforces SQL Injection | It prevents SQL Injection |

**Suggested reading =>> JDBC PreparedStatement and Statement**

## Q #11) Explain the difference between execute(), executeQuery() and executeUpdate().

**Answer:**

| executeQuery() | executeUpdate() | Execute() |
|---|---|---|
| s used to execute the SQL atements which retrieve some ta from DB | It is used to execute the SQL statements which will update or modify the data in DB | It is used to execute any kind of SQL statements |
| It returns the resultSet object | It returns an integer value which represents the no. of affected rows | It returns a Boolean value<br>TRUE – returns a resultSet object<br>FALSE – returns an int value or nothing |

| | | |
|---|---|---|
| It is used to execute only SELECT Query | It is used to execute only a non-SELECT query | It is used to execute both SELECT and non-SELECT queries |

## Q #12) How to call Stored Procedures in JDBC?

**Answer:** We can execute the SQL Stored procedures through the CallableStatement interface. The CallableStatement object can be created using the prepareCall() method of the Connection interface.

Create procedure myProcedure (IN name VARCHAR(30), IN sal INT, IN loc VARCHAR(45))

-> BEGIN

-> INSERT INTO Employee(Name, Salary, Location) VALUES (name, sal, loc);

-> END //

## Q #13) What is the ResultSet interface?

**Answer:** ResultSet interface is used to store the output data after the SQL query execution. The object of ResultSet maintains the cursor point at the result data. As a default, the cursor points before the first row of the result data. We can traverse the data in the resultset objects as well.

**Syntax:**

**Statement Interface:**

```
atement stmnt1 = conn.createStatement();

 ResultSet resultset = stmnt1.executeQuery(“Select * from
 EMPLOYEE”);
```

**PreparedStatement Interface:**

```
eparedStatement pstmnt1 = conn.prepareStatement(insert_query);

 ResultSet resultset = pstmnt1.executeQuery(“Select * from
 EMPLOYEE”);
```

## Q #14) What are the types of ResultSet?

**Answer:** There are 3 types in ResultSet. These are:

1. **TYPE_FORWARD_ONLY:** It is the default option. The cursor will move from start to end.
2. **TYPE_SCROLL_INSENSITIVE:** In this type, the cursor will move in both forward and backward directions. Dataset has the data when the SQL query returns the data.
3. **TYPE_SCROLL_SENSITIVE:** It is the same as TYPE_SCROLL_INSENSITIVE, the difference is that it will have the updated data while iterating the resultset object.

## Q #15) What are the concurrency modes in ResultSet?

**Answer:** There are 2 different modes of Concurrency in ResultSet. They are:

1. **ResultSet.CONCUR_READ_ONLY**: It is the default concurrency mode. A read-only option is available. Updation is not possible.
2. **ResultSet.CONCUR_UPDATABLE:** Updation is possible.

## Q #16) How to check whether the database supports the concurrency mode?

**Answer:** We have the supportsResultSetConcurrency() method which will be used to check whether the given type and concurrency modes are supported by the database or not.

## Q #17) Can we get the data of the particular row from the resultset?

**Note:** ResultSet has the data of a set of rows

**Answer:** Yes, we can get the data of the particular row from the resultSet using the relative() method. It will move the cursor to the given row either in a forward or in a backward direction from the current row. If the positive value has been given, it will move in the forward direction. If the negative value has been given, it will move in the backward direction.

## Q #18) What is the use of the getter and setter methods in ResultSet?

**Answer:**

**Getter methods:** These are used to retrieve the values of the particular column of the table from ResultSet. Either the Column Index value or Column Name should be passed as a parameter. Normally, we will represent the getter method as getXXX() methods.

**Example:**

- **int getInt(string Column_Name):** It is used to retrieve the value of the specified column Index and int data type as a return type.

**Setter Methods:** We can set the value in the database using ResultSet setter methods. It is similar to getter methods, but here we need to pass the values/data for the particular column to insert into the database and the index value or column name of that column. Normally we will represent the setter method as setXXX() methods.

**Example:**

- **void setInt(int Column_Index, int Data_Value):** It is used to insert the value of the specified column Index with an int value.

## Q #19) What is the main purpose of the ResultSetMetaData interface?

**Answer:** This interface gives more information about ResultSet. Each ResultSet object has been associated with one ResultSetMetaData object.

This object will have the details of the properties of the columns like datatype of the column, column name, the number of columns in that table, table name, schema name, etc., getMetaData() method of ResultSet object is used to create the ResultSetMetaData object.

**Syntax:**

```
eparedStatement pstmntobj = conn.prepareStatement(insert_query);

  ResultSet resultsetobj = pstmntobj.executeQuery(“Select * from
  EMPLOYEE”);

  ResultSetMetaData rsmd obj= resultsetobj.getMetaData();
```

## Q #20) What is DatabaseMetaData?

**Answer:** The DatabaseMetaData interface gives information about the Database we are using. We will get the following information – DatabaseName, Database version, and so on.

## Q #21) What is ACID property?

**Answer:**

- **A–Atomicity ->** If all the queries have executed successfully, then the data will be committed else won’t commit.
- **C–Consistency ->** Data should be consistent after any transaction.
- **I–Isolation ->** Each transaction should be isolated.
- **D–Durability ->** If the transaction is committed once, it should be available always (if no changes have happened)

## Q #22) How to change the auto-commit mode value?

**Answer:** By default, the value of AutoCommit is TRUE. After the execution of the SQL statement, it will be committed automatically. Using the setAutoCommit() method, we can change the value to AutoCommit.

## Q #23) What is the use of Commit and Rollback methods?

**Answer:**

**Commit() method:** We have the commit() method in Java to commit the data. Once the SQL execution is done, we can call the commit method.

**Syntax:** connectionobj.commit();

**Rollback() method:** We have the rollback() method in Java to rollback the data. Rollback means to undo the changes. If any of the SQL statements are failed, we can call the rollback method to undo the changes. **Syntax:** connectionobj.rollback();

## Q #24) What is savepoint and what are the methods we have in JDBC for savepoint?

**Answer:** Savepoint is used to create checkpoints in a transaction, and it allows us to perform a rollback to the specific savepoint. Once the transaction is committed or rolled backed, the savepoint that has been created for a transaction will be automatically destroyed and becomes invalid.

**Methods for Savepoint:**

- **setSavepoint() method:** It is used to create Savepoint, we can use the rollback() method to undo all the changes till the savepoint.
- **releaseSavepoint() method:** It is used to remove the given savepoint.

## Q #25) List some exceptions that come under SQLException?

**Answer:**

- SQLNonTransientException
- SQLTransientException
- SQLRecoverableException

**SQLNonTransientException:** This type of exception will be thrown when an instance where a retry of the same operation would fail unless the cause of the SQLException has been corrected.

**SQLTransientException:** This type of exception will be thrown when a previously failed operation is able to succeed when we re-tried the operation again without any change/intervention.

**SQLRecoverableException:** This type of exception will be thrown when a previously failed operation can succeed when we re-tried the operation again with any change/intervention by the application. While doing that the current connection should be closed and the new connection should be opened.

## Q #26) What is batch processing and how to do it in JDBC?

**Answer:** Batch processing is the process of executing several SQL statements in one transaction. Doing so will reduce communication time and increase performance. It makes processing a large amount of data much easier.

**Advantages of Batch Processing:**

- Improve performance
- Data consistency

**How to perform Batch Processing:**

We have addBatch() and executeBatch() methods in Java to perform Batch processing. These 2 methods are present in Statement and PreparedStatement classes.

## Q #27) What is the stored procedure?

**Answer:** A group of SQL queries that are executed as a single unit to perform a particular task is known as a Stored Procedure. We can pass 3 different types of parameters. Each procedure is represented by its name. So the name of the procedure should be unique.

## Q #28) What are the parameter types in Stored Procedures?

**Answer:** There are three types of parameters available in Stored Procedures. They are:

1. **IN:** Used to pass the input values to the procedure.
2. **OUT:** Used to get the value from the procedure.
3. **IN/OUT:** Used to pass the input values and get the value to/from the procedure.

### 29) Which interface is responsible for transaction management in JDBC?

The **Connection interface** provides methods for transaction management such as commit(), rollback() etc.

## 30) What are the different types of lockings in JDBC?

A lock is a certain type of software mechanism by using which, we can restrict other users from using the data resource. There are four type of locks given in JDBC that are described below.

- **Row and Key Locks:** These type of locks are used when we update the rows.
- **Page Locks:** These type of locks are applied to a page. They are used in the case, where a transaction remains in the process and is being updated, deleting, or inserting some data in a row of the table. The database server locks the entire page that contains the row. The page lock can be applied once by the database server.
- **Table locks:** Table locks are applied to the table. It can be applied in two ways, i.e., shared and exclusive. Shared lock lets the other transactions to read the table but not update it. However, The exclusive lock prevents others from reading and writing the table.
- **Database locks:** The Database lock is used to prevent the read and update access from other transactions when the database is open.

## 31) What is the JDBC Rowset?

JDBC Rowset is the wrapper of ResultSet. It holds tabular data like ResultSet, but it is easy and flexible to use. The implementation classes of RowSet interface are as follows:

- JdbcRowSet
- CachedRowSet
- WebRowSet
- JoinRowSet
- FilteredRowSet

The advantage of RowSet is as follows:

1. It is easy and flexible to use.
2. It is by default scrollable and can be updated by default whereas ResultSet by default is only forwardable and read-only operation is valid there only.

The advantage of RowSet is as follows:

1. It is easy and flexible to use.
2. It is by default scrollable and can be updated by default whereas ResultSet by default is only forwardable and read-only operation is valid there only.