InterviewBit

JDBC Interview Questions

To view the live version of the page, click here.

Contents

JDBC Interview Questions for Experienced

(.....Continued)

Let's get Started

Introduction to JDBC:

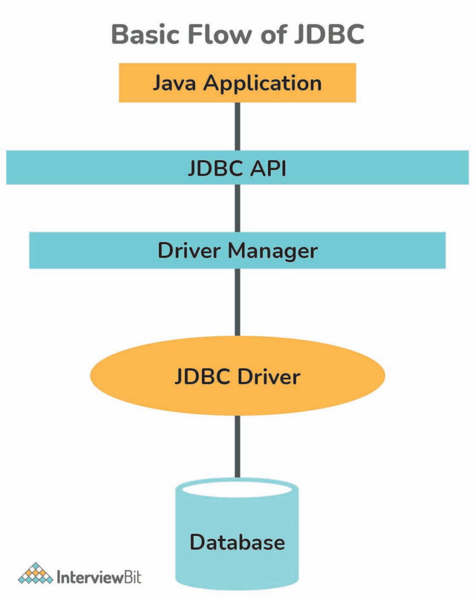
JDBC is an Application Programming Interface(API) for Java, which is helpful for interaction with the database and for executing the SQL query. JDBC is an abbreviation used for Java Database Connectivity. It uses JDBC drivers for connecting with the database. JDBC API is used to access tabular data stored in relational databases like Oracle, MySQL, MS Access, etc.

Components of JDBC:

There are four major components of JDBC using which it can interact with a database. They are:

1. **JDBC API**: It provides different methods and interfaces for easier communication with the database. By using this, applications are able to execute SQL statements, retrieve results and make updation to the database. It has two packages as follows which consist of Java SE and Java EE platforms to exhibit Write Once Run Everywhere(WORA) capabilities.
    1. `java.sql.*;`
    2. `javax.sql.*;`

    Also, it provides a standard for connecting a database to a client application.
2. **JDBC DriverManager**: It is the class in JDBC API. It loads the JDBC driver in a Java application for establishing a connection with the database. It is useful in making a database-specific call for processing the user request.
3. **JDBC Test suite**: It is used to test the operations like insertion, deletion, updation etc., being performed by JDBC Drivers.
4. **JDBC-ODBC bridge drivers**: It will connect database drivers to the database. JDBC-ODBC bridge interprets JDBC method call to the ODBC function call. It will use sun.jdbc.odbc package, which consists of the native library to access characteristics of ODBC.



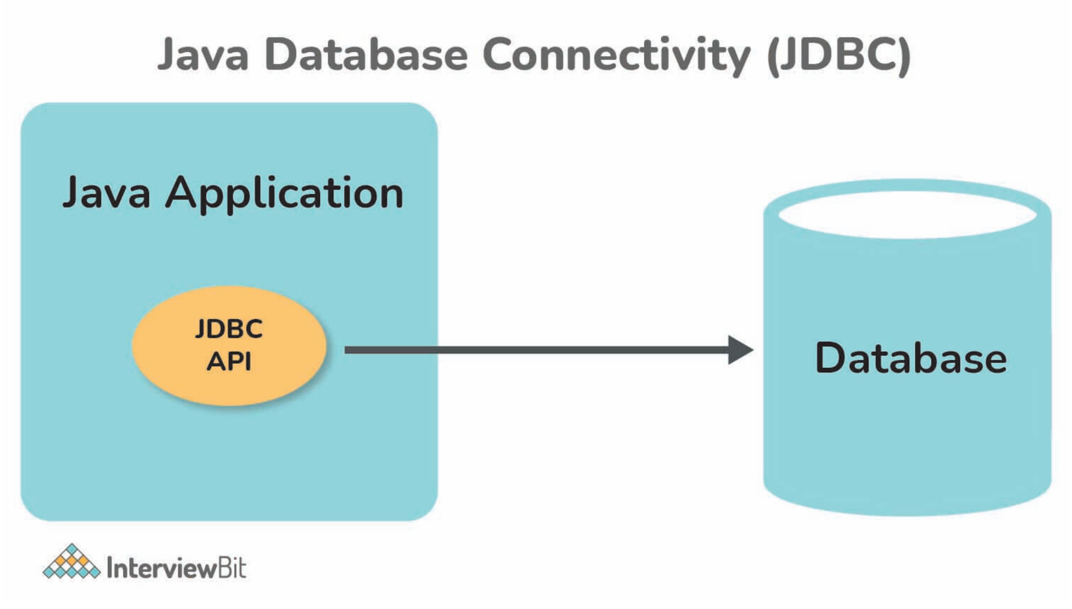
Scope of JDBC:

Earlier, ODBC API was used as the database API to connect with the database and execute the queries. But, ODBC API uses C language for ODBC drivers(i.e. platform-dependent and unsecured). Hence, Java has defined its own JDBC API that uses JDBC drivers, which offers a natural Java interface for communicating with the database through SQL. JDBC is required to provide a “pure Java” solution for the development of an application using Java programming.

JDBC Interview Questions for Freshers

1. What is JDBC in Java?

JDBC(Java Database Connectivity) is a Java API, which is helpful in interaction with the database to retrieve, manipulate and process the data using SQL. It will make use of JDBC drivers for connecting with the database. By using JDBC, we can access tabular data stored in various types of relational databases such as Oracle, MySQL, MS Access, etc.



2. What is ResultSet?

- The `java.sql.ResultSet` interface represents the database result set, which is obtained after the execution of SQL query using Statement objects.
- Object of ResultSet maintains a cursor pointing to the current row of data in the result set. Initially, the cursor is located before the first row. Then the cursor is moved to the next row by using the `next()` method. The `next()` method can be used to iterate through the result set with the help of a while loop. If there are no further rows, the `next()` method will return false.
- Example for the creation of ResultSet is given below:

  ```
  ResultSet rs = con.executeQuery(sqlQuery);
  ```

3. What is JDBC driver?

JDBC driver is a software component having various classes and interfaces, that enables the Java application to interact with a database.

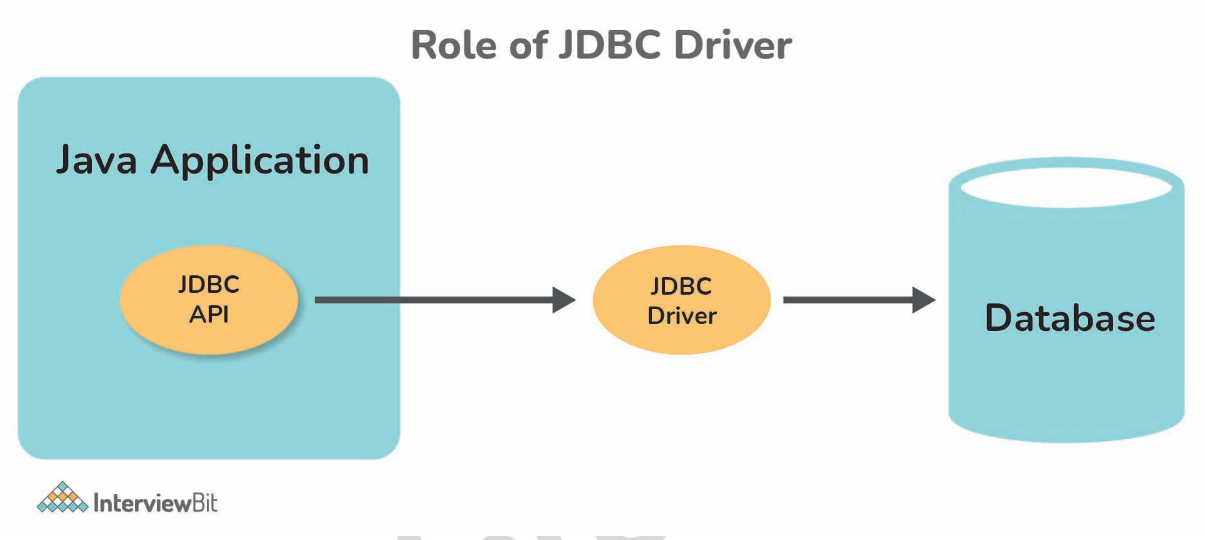
To connect with individual databases, JDBC requires particular drivers for each specific database. These drivers are provided by the database vendor in addition to the database. For example:

- MySQL Connector/J is the official JDBC driver for MySQL and we can locate the `mysql-connector-java-<version>-bin.jar` file among the installed files. On windows, this file can be obtained at

```
C:\Program Files (x86)\MySQL\MySQL Connector J\mysql-connector-java-5.1.30-bin.jar.
```

- JDBC driver of Oracle 10G is ojdbc14.jar and it can be obtained in the installation directory of an Oracle at `…/Oracle/app/oracle/product/10.2.0/server/jdbc/lib` .

JDBC driver provides the connection to the database. Also, it implements the protocol for sending the query and result between client and database.



4. What is DriverManager in JDBC?

- JDBC DriverManager is a static class in Java, through which we manage the set of JDBC drivers that are available for an application to use.
- Multiple JDBC drivers can be used concurrently by an application, if necessary. By using a Uniform Resource Locator(URL), each application specifies a JDBC driver.
- When we load the JDBC Driver class into an application, it registers itself to the DriverManager by using `Class.forName()` or `DriverManager.registerDriver()` . To check this, you can have a look into the source code of JDBC Driver classes. After this, when we call `DriverManager.getConnection()` method by passing the details regarding database configuration, DriverManager will make use of registered drivers to obtain the connection and return it to the caller program.

5. Which JDBC driver is fastest and used more commonly?

JDBC Net pure Java driver(Type 4 driver) is the fastest driver for localhost and remote connections because it directly interacts with the database by converting the JDBC calls into vendor-specific protocol calls.

6. Which data types are used for storing the image and file in the database table?

- **BLOB** data type is used to store the image in the database. We can also store videos and audio by using the BLOB data type. It stores the binary type of data.
- **CLOB** data type is used to store the file in the database. It stores the character type of data.

7. What is stored procedure? What are the parameter types in stored procedure?

- Stored procedure is a group of SQL queries that are executed as a single logical unit to perform a specific task. Name of the procedure should be unique since each procedure is represented by its name.
- For example, operations on an employee database like obtaining information about an employee could be coded as stored procedures that will be executed by an application. Code for creating a stored procedure named GET_EMP_DETAILS is given below:

```
DELIMITER $$
DROP PROCEDURE IF EXISTS `EMP`.`GET_EMP_DETAILS` $$
CREATE PROCEDURE `EMP`.`GET_EMP_DETAILS`
  (IN EMP_ID INT, OUT EMP_DETAILS VARCHAR(255))
BEGIN
  SELECT first INTO EMP_DETAILS
  FROM Employees
  WHERE ID = EMP_ID;
END $$
DELIMITER ;
```

Stored procedures are called using CallableStatement class available in JDBC API. Below given code demonstrates this:

```
CallableStatement cs = con.prepareCall("{call GET_EMP_DETAILS(?,?)}");
ResultSet rs = cs.executeQuery();
```

- Three types of parameters are provided in the stored procedures. They are:
    - **IN**: It is used for passing the input values to the procedure. With the help of setXXX() methods, you can bind values to IN parameters.
    - **OUT**: It is used for getting the value from the procedure. With the help of getXXX() methods, you can obtain values from OUT parameters.
    - **IN/OUT**: It is used for passing the input values and obtaining the value to/from the procedure. You bind variable values with the setXXX() methods and obtain values with the getXXX() methods.

8. What do you mean by DatabaseMetaData and why we are using it?

- DatabaseMetaData is an interface that provides methods to obtain information about the database.
- We can use this for getting database-related informations, such as database name, database version, driver name, the total number of tables or views, etc.

9. What are the differences between ODBC and JDBC?

ODBC(Open Database Connectivity)	JDBC(Java Database Connectivit
ODBC can be used for languages like C, C++, Java, etc.	JDBC is used only for the Java language
We can use ODBC only for the Windows platform, thus it is platform-dependent.	We can use JDBC on any platform, thus it is platform-independent
Most of the ODBC Drivers developed in native languages like C, C++	JDBC drivers are developed using the Java language
It is not recommended to use ODBC for Java applications, because of low performance due to internal conversion.	It is highly recommended to use JDBC for Java applications becaus there are no performance issues.
ODBC is procedural.	JDBC is Object Oriented.

10. What is Rowset?

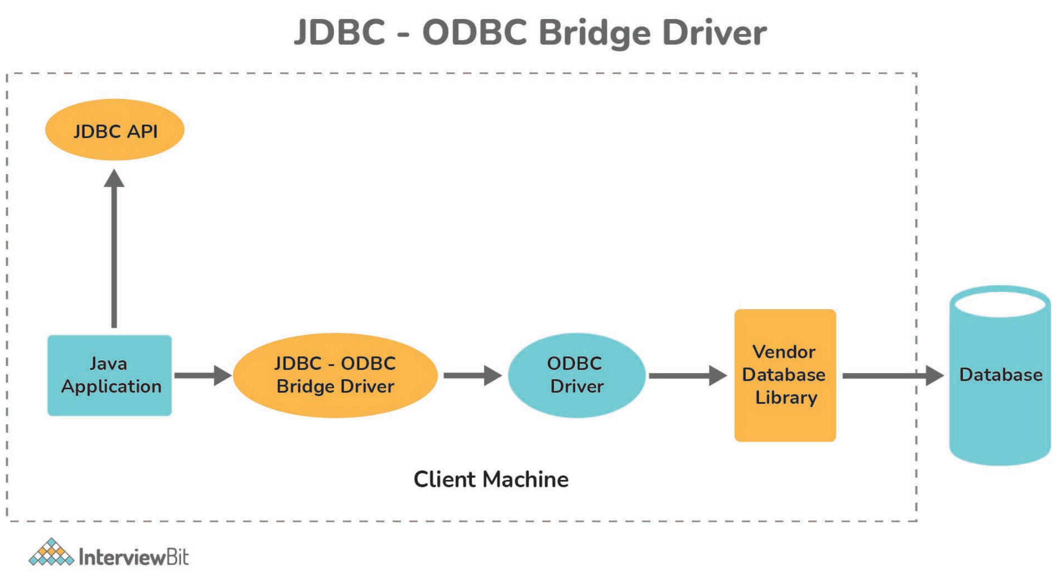
- A RowSet is an object that encapsulates a row set from either JDBC result sets or tabular data sources such as files or spreadsheets. It supports component-based development models like JavaBeans, with the help of a standard set of properties and event notifications.
- The advantages of using RowSet are:
    - It is easier and flexible to use.
    - It is Scrollable and Updatable by default.

JDBC Interview Questions for Experienced

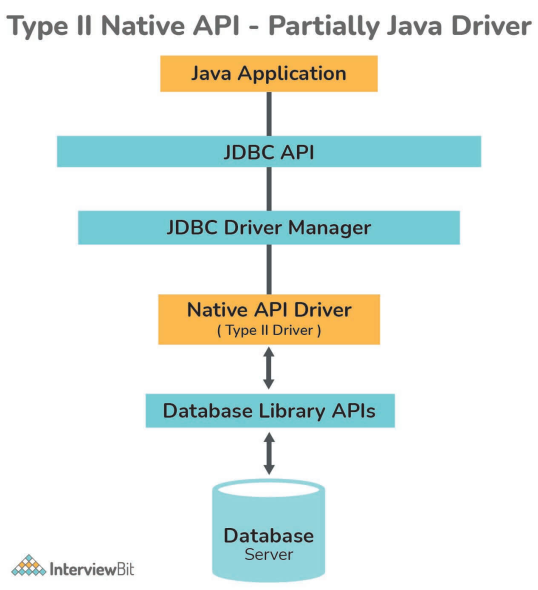
11. What are the different types of JDBC drivers in Java? Explain each with an example.

There are four types of JDBC drivers in Java. They are:

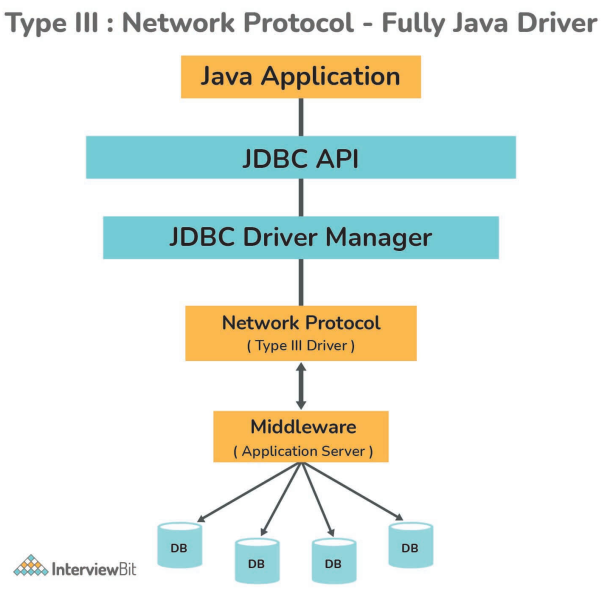
- **Type I: JDBC - ODBC bridge driver**
    - In this, the JDBC–ODBC bridge acts as an interface between the client and database server. When a user uses a Java application to send requests to the database using JDBC–ODBC bridge, it converts the JDBC API into ODBC API and then sends it to the database. When the result is received from the database, it is sent to ODBC API and then to JDBC API.
    - It is platform-dependent because it uses ODBC which depends on the native library of the operating system. In this, JDBC–ODBC driver should be installed in every client system and database must support for ODBC driver.'
    - It is easier to use but it gives low performance because it involves the conversion of JDBC method calls to the ODBC method calls.



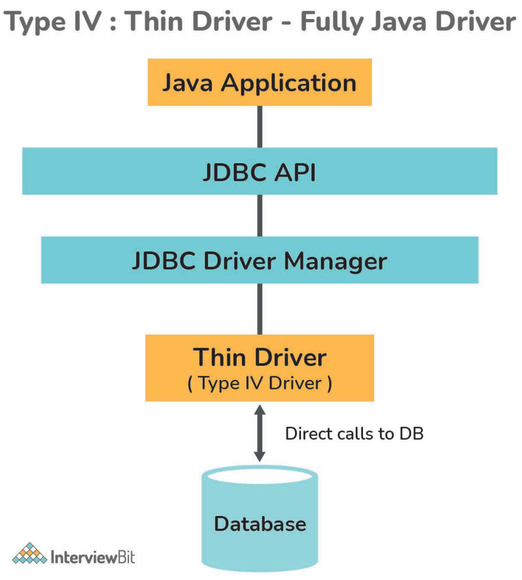
- **Type II: Native API – Partially Java Driver:**
  - It is almost similar to a Type I driver. Here, native code replaces the ODBC part. This native code part is targeted at a particular database product. It uses libraries of the client-side of the database. This Type II Driver converts the JDBC method calls to native calls of the database native API.
  - When the database gets the requests from the user, the requests are processed and sends the results back in the native format which is then converted into JDBC format and pass it to the Java application.
  - It was instantly adopted by the database vendors because it was quick and cheaper to implement. This driver gives faster response and performance compared to the Type I driver.



- **Type III: Network Protocol - Fully Java Driver:**
  - The type III driver is completely written in Java. It is similar to the 3-tier approach to access the database. It helps to send the JDBC method calls to an intermediate server. The intermediate server communicates with the database on behalf of JDBC. The application server converts the JDBC calls either directly or indirectly to the database protocol which is vendor-specific.
  - This approach does not increase the efficiency of architecture and it is costlier, due to this most of the database vendors don't choose this driver. You need to have good knowledge about the application server for using this approach since the application server is used here.



- **Type IV: Thin Driver - Fully Java Driver**
  - Type IV driver is directly implemented and it directly converts JDBC calls into vendor-specific database protocol. Most of the JDBC Drivers used today are type IV drivers.
  - It is platform-independent since it is written fully in Java. It can be installed inside the Java Virtual Machine(JVM) of the client, so there is no need of installing any software on the client or server side. This drive architecture is having all the logic to communicate directly with the database in a single driver.
  - It provides better performance compared to other driver types. It permits easy deployment. Nowadays, this driver is developed by the database vendor itself so that programmers can use it directly without any dependencies on other sources.



12. What are difference between ResultSet and RowSet?

ResultSet	RowSet
ResultSet cannot be serialized as it handles the connection to the database.	RowSet is disconnected from the database so it can be serialized.
By default, ResultSet object is non-scrollable and non-updatable.	By default, the RowSet object is scrollable and updatable.
ResultSet object is not a JavaBean object.	RowSet object is a JavaBean object.
ResultSet is returned by the `executeQuery()` method of Statement interface	Rowset extends the ResultSet interface and it is returned by calling the `RowSetProvider.newFactory().createJdbcRowSet()` method.
It is difficult to pass ResultSet from one class to another class as it has a connection with the database.	It is easier to pass RowSet from one class to another class as it has no connection with the database.

13. Explain the types of ResultSet.

ResultSet refers to the row and column data contained in a ResultSet object. The object of ResultSet maintains a cursor pointing to the current row of data in the result set.
There are three types of ResultSet which have constants to control the movement of the cursor in backward, forward, and in a particular row. If we do not declare any ResultSet, then by default TYPE_FORWARD_ONLY will be called.

- **ResultSet.TYPE_FORWARD_ONLY**: Using this, the cursor can only move forward from start to end in the result set.
- **ResultSet.TYPE_SCROLL_INSENSITIVE**: Using this, the cursor can move in both forward and backward directions. Here, the result set is insensitive to the changes done in the database by others, that occur after the result set was created.
- **ResultSet.TYPE_SCROLL_SENSITIVE**: Using this, the cursor can move in forward and backward direction, and the result set is sensitive to changes made to the database by others, that occur after the result set was created.

14. Explain JDBC API components.

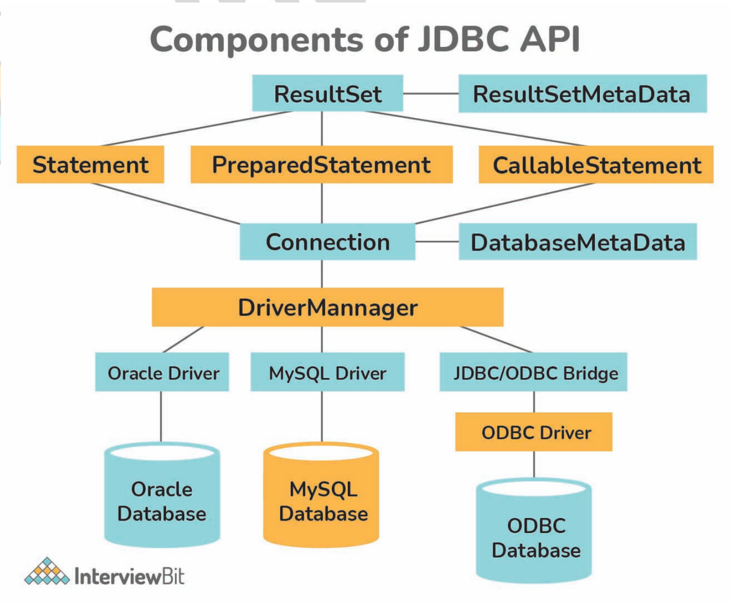
The `java.sql` package contains different interfaces and classes for JDBC API. They are:

Interfaces:

- **Connection**: The object of Connection is created by using getConnection() method of DriverManager class. DriverManager is the factory for connection.
- **Statement**: The object of the Statement is created by using createStatement() method of the Connection class. The Connection interface is the factory for Statement.
- **PreparedStatement**: The PreparedStatement object is created by using prepareStatement() method of Connection class. It is used for executing the parameterized query.
- **ResultSet:** The ResultSet object maintains a cursor pointing to a table row. At first, the cursor points before the first row. The executeQuery() method of the Statement interface returns the object of ResultSet.
- **ResultSetMetaData**: The ResultSetMetaData interface object contains the details about the data(table) such as number of columns, name of the column, column type etc. The getMetaData() method of ResultSet returns the ResultSetMetaData object.
- **DatabaseMetaData**: It is an interface that has methods to get metadata of a database, like name of the database product, version of database product, driver name, name of the total number of views, name of the total number of tables, etc. The getMetaData() method that belongs to Connection interface returns the DatabaseMetaData object.
- **CallableStatement**: CallableStatement interface is useful for calling the stored procedures and functions. We can have business logic on the database through the usage of stored procedures and functions, which will be helpful for the improvement in the performance as these are pre-compiled. The prepareCall() method that belongs to the Connection interface returns the object of CallableStatement.

Classes:

- **DriverManager**: It pretends to be an interface between the user and drivers. DriverManager keeps track of the available drivers and handles establishing a connection between a database and the relevant driver. It contains various methods to keep the interaction between the user and drivers.
- **BLOB:** BLOB stands for Binary Large Object. It represents a collection of binary data such as images, audio, and video files, etc., which is stored as a single entity in the DBMS(Database Management System).
- **CLOB:** CLOB stands for Character Large Object. This data type is used by multiple database management systems to store character files. It is the same as BLOB except for the difference, instead of binary data, CLOB represents character stream data such as character files, etc.



15. What are the types of JDBC statements?

Statements are useful for sending SQL commands to the database and receiving data from the database. There are three types of statements in JDBC. They are:

- **Statement:** It is the factory for ResultSet. It is used for general-purpose access to the database by executing the static SQL query at runtime. Example:

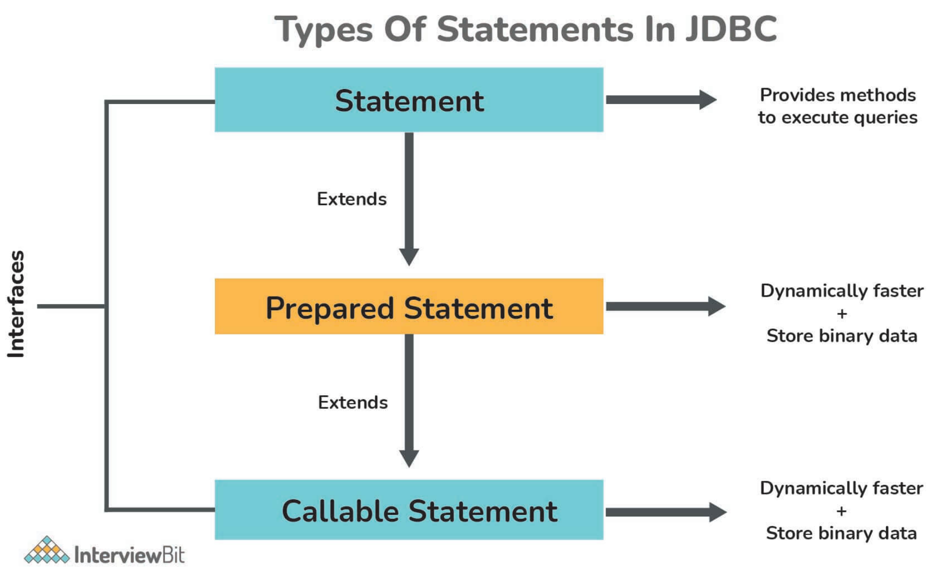
```
Statement st = conn.createStatement( );
ResultSet rs = st.executeQuery();
```

- **PreparedStatement:** It is used when we need to give input data to the query at runtime and also if we want to execute SQL statements repeatedly. It is more efficient than a statement because it involves the pre-compilation of SQL. Example:

```
String SQL = "Update item SET limit = ? WHERE itemType = ?";
 PreparedStatement  ps = conn.prepareStatement(SQL);
 ResultSet rs = ps.executeQuery();
```

- **CallableStatement:** It is used to call stored procedures on the database. It is capable of accepting runtime parameters. Example:

```
CallableStatement cs = con.prepareCall("{call SHOW_CUSTOMERS}");
ResultSet rs = cs.executeQuery();
```



Types Of Statements In JDBC
Interfaces
Statement
Provides methods to execute queries
Extends
Prepared Statement
Dynamically faster + Store binary data
Extends
Callable Statement
Dynamically faster + Store binary data
InterviewBit

16. Explain JDBC Batch processing.

- Batch processing is the process of executing multiple SQL statements in one transaction. For example, consider the case of loading data from CSV(Comma-Separated Values) files to relational database tables. Instead of using Statement or PreparedStatement, we can use Batch processing which executes the bulk of queries in a single go for a database.
- *Advantages of Batch processing:*
  - It will reduce the communication time and improves performance.
  - Batch processing makes it easier to process a huge amount of data and consistency of data is also maintained.
  - It is much faster than executing a single statement at a time because of the fewer number of database calls.
- *How to perform Batch processing?*
  To perform Batch processing, addBatch() and executeBatch() methods are used. These 2 methods are available in the Statement and PreparedStatement classes of JDBC API.

17. What is the difference between Statement and PreparedStatement?

Statement	PreparedStatement
The query is compiled every time we run the program.	The query is compiled only once.
It is used in the situation where we need to run the SQL query without providing parameters at runtime.	It is used when we want to give input parameters to the query at runtime.
Performance is less compared to PreparedStatement.	Provides better performance than Statement, as it executes the pre-compiled SQL statements.
It is suitable for executing DDL statements such as CREATE, ALTER, DROP and TRUNCATE.	It is suitable for executing DML statements such as INSERT, UPDATE, and DELETE.
It cannot be used for storing/retrieving images and files in the database.	It can be used for storing/retrieving images and files in the database.
It executes static SQL statements.	It executes pre-compiled SQL statements.
Less secured as it enforces SQL injection.	More secured as they use bind variables, which can prevent SQL injection.

18. What is DataSource in JDBC? What are its benefits?

- DataSource is an interface defined in `javax.sql` package and is used for obtaining the database connection. It can be used as a good alternative for a DriverManager class as it allows the details about the database to your application program.
- A driver that is accessed through a DataSource object, does not register itself with the DriverManager. Instead, a DataSource object is retrieved through a lookup operation and then it can be used to create a Connection object.
- **Benefits of DataSource:**
  - Caching of PreparedStatement for faster processing
  - ResultSet maximum size threshold
  - Logging features
  - Connection timeout settings
  - Connection Pooling in servlet container using the support of JNDI registry.

19. Explain the difference between execute(), executeQuery() and executeUpdate() methods in JDBC.

execute()	executeQuery()	executeUpdate()
It can be used for any SQL statements.	It is used to execute SQL Select queries.	It is used to execute the SQL statements such as Insert/Update/Delete which will update or modify the database data.
It returns the boolean value TRUE if the result is a ResultSet object and FALSE when there is no ResultSet object.	It returns the ResultSet object which contains the data retrieved by the SELECT statement.	It returns an integer value which represents the number of affected rows where 0 indicates that the query returns null.
Used for executing both SELECT and non-SELECT queries.	Used for executing only the SELECT Query.	Used for executing only a non-SELECT query.

The execute() method is used in the situations when you are not sure about the type of statement else you can use executeQuery() or executeUpdate() method.

20. Explain the types of RowSet available in JDBC.

A RowSet is an object that encapsulates a set of rows from JDBC result sets or tabular data sources.
There are five types of RowSet interfaces available in JDBC. They are:

- **JDBCRowSet**: It is a connected RowSet, which is having a live connection to the database, and all calls on this are percolated to the mapping call in the JDBC connection, result set, or statement. The Oracle implementation of JDBCRowSet is done by using `oracle.jdbc.rowset.OracleJDBCRowSet` .
- **CachedRowSet**: It is a RowSet in which the rows are cached and RowSet is disconnected, which means it does not maintain an active database connection. The `oracle.jdbc.rowset.OracleCachedRowSet` class is used as the Oracle implementation of CachedRowSet.
- **WebRowSet**: It is an extension to CachedRowSet and it represents a set of fetched rows of tabular data that can be passed between tiers and components so that no active data source connections need to be maintained.
  It provides support for the production and consumption of result sets and their synchronization with the data source, both in XML(Extensible Markup Language) format and in a disconnected fashion. This permits result sets to be transmitted across tiers and over Internet protocols. The Oracle implementation of WebRowSet is done by using `oracle.jdbc.rowset.OracleWebRowSet` .
- **FilteredRowSet**: It's an extension to WebRowSet and gives programmatic support to filter its content. This enables you to avoid the difficulty of query supply and processing involved. The Oracle implementation of FilteredRowSet is done by using `oracle.jdbc.rowset.OracleFilteredRowSet` .
- **JoinRowSet**: It's an extension to WebRowSet and consists of related data from various RowSets. There is no standard way to establish a SQL JOIN operation between disconnected RowSets without a data source connection. A JoinRowSet addresses this problem. The Oracle implementation of JoinRowSet is done by using `oracle.jdbc.rowset.OracleJoinRowSet` class.

21. Explain the usage of the getter and setter methods in ResultSet.

- **Getter methods:** These are used for retrieving the particular column values of the table from ResultSet. As a parameter, either the column index value or column name should be passed. Usually, the getter method is represented as getXXX() methods.
  Example:

```
int getInt(string Column_Name)
```

The above statement is used to retrieve the value of the specified column Index and the return type is an int data type.

- **Setter Methods:** These methods are used to set the value in the database. It is almost similar to getter methods, but here it requires to pass the data/values for the particular column to insert into the database and the column name or index value of that column. Usually, setter method is represented as setXXX() methods.
  Example:

```
void setInt(int Column_Index, int Data_Value)
```

The above statement is used to insert the value of the specified column Index with an int value.

22. What is meant by a locking system in JDBC?

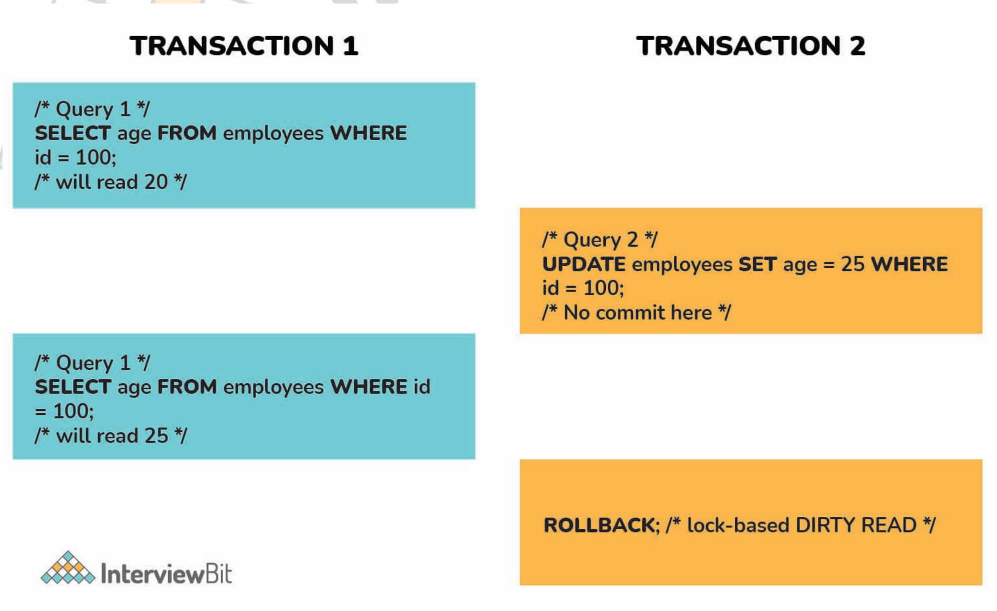
- If two users are viewing the same record, then there is no issue, and locking will not be done. If one user is updating a record and the second user also wants to update the same record, in this situation, we are going to use locking so that there will be no lost update.
- Two types of locking are available in JDBC by which we can handle multiple user issues using the record. They are:
  - **Optimistic Locking:** It will lock the record only when an update takes place. This type of locking will not make use of exclusive locks when reading or selecting the record.
  - **Pessimistic Locking:** It will lock the record as soon as it selects the row to update. The strategy of this locking system guarantees that the changes are made safely and consistently.

23. What is database connection pooling? What are the advantages of connection pool?

- Connection pooling means database connections will be stored in the cache and can be reused when future requests to the database are required. So in this mechanism, the client need not make new connections every time for interacting with the database. Instead of that, connection objects are stored in the connection pool and the client will get the connection object from there.
- Advantages of using a connection pool are:
  - It is faster.
  - Easier to diagnose and analyze database connections.
  - Increases the performance of executing commands on a database.

24. What is “Dirty read” in terms of database?

- Dirty read implies the meaning "read the value which may or may not be correct". In the database, when a transaction is executing and changing some field value, at the same time another transaction comes and reads the changed field value before the first transaction could commit or rollback the value, which may cause an invalid value for that particular field. This situation is known as a dirty read.
- Consider an example given below, where Transaction 2 changes a row but does not commit the changes made. Then Transaction 1 reads the uncommitted data. Now, if Transaction 2 goes for roll backing its changes (which is already read by Transaction 1) or updates any changes to the database, then the view of the data may be wrong in the records related to Transaction 1. But in this case, no row exists that has an id of 100 and an age of 25.



```
/* Query 1 */
SELECT age FROM employees WHERE
id = 100;
/* will read 20 */
```

```
/* Query 2 */
UPDATE employees SET age = 25 WHERE
id = 100;
/* No commit here */
```

```
/* Query 1 */
SELECT age FROM employees WHERE id
= 100;
/* will read 25 */
```

```
ROLLBACK; /* lock-based DIRTY READ */
```

25. What causes "No suitable driver" error?

"No suitable driver" error occurs during a call to the `DriverManager.getConnection()` method, because of the following reasons:

- Unable to load the appropriate JDBC drivers before calling the getConnection() method.
- It can specify an invalid or wrong JDBC URL, which cannot be recognized by the JDBC driver.
- This error may occur when one or more shared libraries required by the bridge cannot be loaded.

26. What is JDBC Connection? Explain steps to get JDBC database connection in a simple Java program.

Loading the driver: At first, you need to load or register the driver before using it in the program. Registration must be done once in your program. You can register a driver by using any one of the two methods mentioned below:

- `Class.forName()` : Using this, we load the driver's class file into memory during runtime. It's not required to use a new or creation of an object.

The below given example uses Class.forName() to load the Oracle driver:

```
Class.forName(“oracle.jdbc.driver.OracleDriver”);
```

The MySQL Connector/J version 8.0 library comes with a JDBC driver class: com.mysql.jdbc.Driver. Before Java 6, we had to load the driver explicitly using the statement given below:

```
Class.forName("com.mysql.jdbc.Driver");
```

However, this statement is no longer needed, because of a new update in JDBC 4.0 that comes from Java 6. As long as you place the MySQL JDBC driver JAR file into the classpath of your program, the driver manager can find and load the driver.

- `DriverManager.registerDriver()` : DriverManager is a built-in Java class with a static member register. Here we will be calling the constructor of the driver class during compile time.

The below given example uses `DriverManager.registerDriver()` to register the Oracle driver:

```
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
```

For registering the MySQL driver, use the below-given code:

```
DriverManager.registerDriver(new com.mysql.jdbc.Driver(); );
```

Create the connections:

- After loading the driver into the program, establish connections using the code given below:

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

Here,

con: Reference to a Connection interface.

url: Uniform Resource Locator.

user: Username from which SQL command prompt is accessed.

password: Password from which SQL command prompt is accessed.

- Url in Oracle can be created as follows:

```
String url = "jdbc:oracle:thin:@localhost:1521:xe";
```

Where oracle represents the database used, thin is the driver used, @localhost is the IP(Internet Protocol) address where the database is stored, 1521 is the port number and xe represents the service provider.

All 3 parameters given above are of string type and are expected to be declared by the programmer before the function call. Use of this can be referred from the final code of an application.

- Url in MySQL can be created as follows:

```
String url = "jdbc:mysql://localhost:3306/test1";
```

Where localhost represents hostname or IP address of the MySQL server, 3306 port number of the server and by default, it is 3306, test1 is the name of the database on the server.

Create a statement:

- Once a connection establishment is done, you can interact with the database. The Statement, PreparedStatement, and CallableStatement JDBC interfaces will define the methods that permit you to send SQL commands and receive data from the database.
- We can use JDBC Statement as follows:

```
Statement st = con.createStatement();
```

Here, con is a reference to the Connection interface used in the earlier step.

Execute the query:

- Here, query means an SQL query. We can have various types of queries. A few of them are as follows:
  - Query for updating or inserting a table in a database.
  - Query for data retrieval.
- The executeQuery() method that belongs to the Statement interface is used for executing queries related to values retrieval from the database. This method returns the ResultSet object which can be used to get all the table records.
- The executeUpdate(sql_query) method of the Statement interface is used for executing queries related to the update/insert operation.

Example:

```
    int m = st.executeUpdate(sql);
    if (m==1)
        System.out.println("Data inserted successfully : "+sql);
    else
        System.out.println("Data insertion failed");
```

Here SQL is the SQL query of string type.

Close the connection:

- So finally we have sent the data to the location specified and now we are at the end of our task completion.
- On connection closing, objects of Statement and ResultSet will be automatically closed. The close() method of the Connection interface is used for closing the connection.

Example:

```
con.close();
```

Implementation of JDBC Oracle database connection using a Java program:

```java
import java.sql.*;
import java.util.*;
class OracleCon
{
   public static void main(String a[])
   {
       //Creating the connection
       String url = "jdbc:oracle:thin:@localhost:1521:xe";
       String user = "system";
       String password = "123";

       //Entering the data
       Scanner k = new Scanner(System.in);
       System.out.println("Enter employee Id");
       int empid = k.nextInt();
       System.out.println("Enter employee name");
       String empname = k.next();
       System.out.println("Enter employee address");
       String address =  k.next();

       //Inserting data using SQL query
       String sql = "insert into employee values("+empid+",'"+empname+"','"+address+"')
       Connection con=null;
       try
       {
           DriverManager.registerDriver(new oracle.jdbc.OracleDriver());

           //Reference to connection interface
           con = DriverManager.getConnection(url,user,password);

           Statement st = con.createStatement();
           int m = st.executeUpdate(sql);
           if (m == 1)
               System.out.println("Data inserted successfully : "+sql);
           else
               System.out.println("Data insertion failed");
           con.close();
       }
       catch(Exception ex)
       {
           System.err.println(ex);
       }
   }
}
```

Implementation of JDBC MySQL database connection using Java program:

```
import java.sql.*;
class MysqlCon
{
   public static void main(String args[])
   {
        //Creating the connection
       String url = "jdbc:mysql://localhost:3306/test1";
       String user = "system";
       String password = "123";
       try
       {
           Class.forName("com.mysql.jdbc.Driver");

            //Reference to connection interface
           Connection con=DriverManager.getConnection(url,user,password);

           Statement st = con.createStatement();

           //Displaying all the records of employee table
           ResultSet rs = st.executeQuery("select * from employee");
           while(rs.next())
               System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3))
           con.close();
       }
       catch(Exception e)
       {
           System.out.println(e);
       }
   }
}
```

27. How to use JDBC API to call Stored procedures?

Stored procedures are a set of SQL queries that are compiled in the database and will be executed from JDBC API. For executing Stored procedures in the database, JDBC CallableStatement can be used. The syntax for initializing a CallableStatement is:

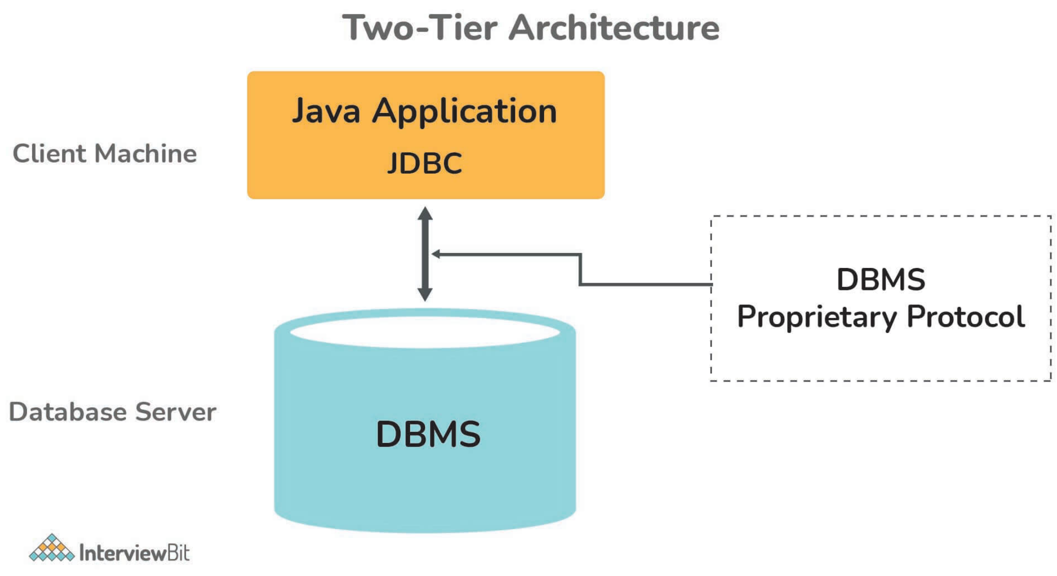
```
CallableStatement cs = con.prepareCall("{call insertEmployee(?,?,?,?,?)}");
stmt.setInt(1, id);
stmt.setString(2, name);
stmt.setString(3, role);
stmt.setString(4, address);
stmt.setString(5, salary);
//registering the OUT parameter before calling the stored procedure
cs.registerOutParameter(5, java.sql.Types.VARCHAR);

cs.executeUpdate();
```

We must register the OUT parameters before executing the CallableStatement.

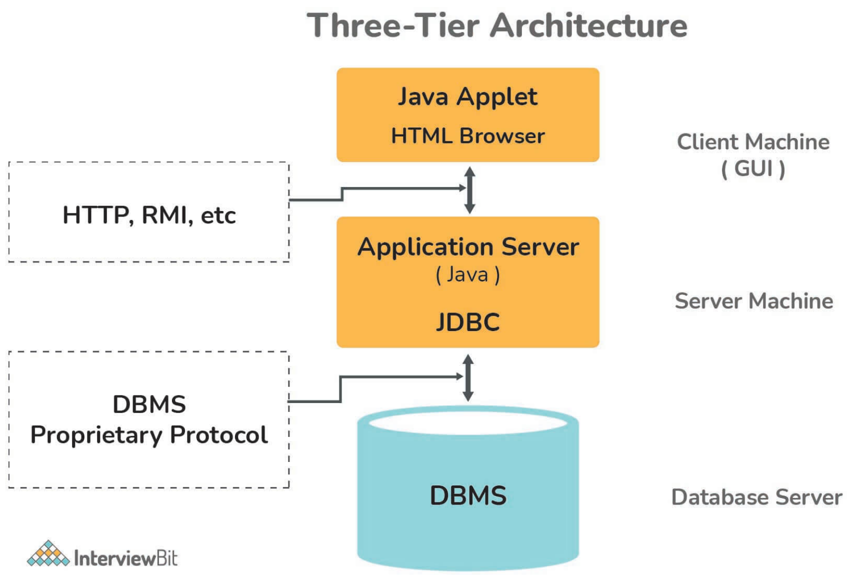
28. What are the types of JDBC architecture?

JDBC supports 2 types of processing models to access the database. They are:

- **Two-tier Architecture:** Here Java programs are explicitly connected with the database. It doesn't require any mediator such as an application server for connecting with the database except the JDBC driver. It is also called client-server architecture.



- **Three-tier Architecture**: It is the complete opposite of two-tier architecture. There will be no explicit communication between the JDBC driver or Java application and the database. It will make use of an application server as a mediator between them. Java code will send the request to an application server, then the server will send it to the database and receive the response from the database.



29. What is JDBC Transaction Management and why is it needed?

- The sequence of actions (SQL statements) served as a single unit that is called a transaction. Transaction Management places an important role in RDBMS-oriented applications to maintain data consistency and integrity.
- Transaction Management can be described well – by using ACID properties. ACID stands for Atomicity, Consistency, Isolation, and Durability.
    - **Atomicity** - If all queries are successfully executed, then only data will be committed to the database.
    - **Consistency** - It ensures bringing the database into a consistent state after any transaction.
    - **Isolation** - It ensures that the transaction is isolated from other transactions.
    - **Durability** - If a transaction has been committed once, it will remain always committed, even in the situation of errors, power loss, etc.
- Need for Transaction Management:
  When creating a connection to the database, the auto-commit mode will be selected by default. This implies that every time when the request is executed, it will be committed automatically upon completion.
  We might want to commit the transaction after the execution of few more SQL statements. In such a situation, we must set the auto-commit value to False. So that data will not be able to commit before executing all the queries. In case if we get an exception in the transaction, we can rollback() changes made and make it like before.

30. Explain the benefits of PreparedStatement over Statement.

Benefits of PreparedStatement over Statement interface are:

- It performs faster compared to the Statement because the Statement needs to be compiled each time when we run the code whereas the PreparedStatement is compiled once and then executed only on runtime.
- It can execute parametrized queries. But Statement can only run static queries.
- The query used in PreparedStatement looks similar each time, so the database can reuse the previous access plan. Statement inline the parameters into the string, so the query doesn't look to be the same every time which prevents reusage of cache.

31. Explain the methods available for Transaction Management in JDBC.

The connection interface is having 5 methods for transaction management. They are given below:

- **setAutoCommit() method:**
  The value of AutoCommit is set to TR
  UE by default. After the SQL statement execution, it will be committed automatically. By using this method we can set the value for AutoCommit.
  Syntax: `conn.setAutoCommit(boolean_value)`
  Here, boolean_value is set to TRUE for enabling autocommit mode for the connection, FALSE for disabling it.
- **Commit() method:**
  The commit() method is used for committing the data. After the SQL statement execution, we can call the commit() method. It will commit the changes made by the SQL statement.
  Syntax: `conn.commit();`
- **Rollback() method:**
  The rollback() method is used to undo the changes made till the last commit has occurred. If we face any problem or exception in the SQL statements execution flow, we may roll back the transaction.
  Syntax: `conn.rollback();`
- **setSavepoint() method:**
  If you have set a savepoint in the transaction (a group of SQL statements), you can use the rollback() method to undo all the changes till the savepoint or after the savepoint(), if something goes wrong within the current transaction. The setSavepoint() method is used to create a new savepoint which refers to the current state of the database within the transaction.
  Syntax: `Savepoint sp= conn.setSavepoint("MysavePoint")`
- **releaseSavepoint() method:**
  It is used for deleting or releasing the created savepoint.
  Syntax: `conn.releaseSavepoint("MysavePoint");`

32. Give few examples of most common exceptions in JDBC.

Some of the most common JDBC exceptions are given below:

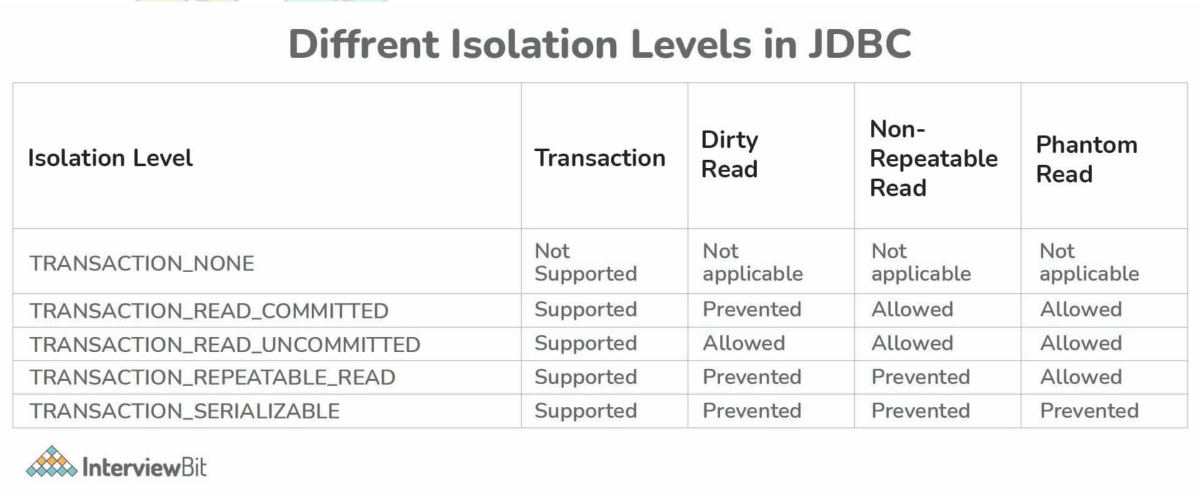
- `java.sql.SQLException` - It is the base class for JDBC exceptions.
- `java.sql.BatchUpdateException` – It occurs during the batch update operation. May depend on the JDBC driver type that the base SQLException may throw instead.
- `java.sql.SQLWarning` – It is displayed as a warning message of various SQL operations.
- `java.sql.DataTruncation` – This exception occurs when data values are unexpectedly truncated due to reasons independent of exceeding MaxFieldSize.

33. What is Two phase commit in JDBC?

- Two-phase commit is useful for a distributed environment where numerous processes take part in the distributed transaction process. In simpler words, we can say that, if a transaction is executing and it is affecting multiple databases then a two-phase commit will be used to make sure that all databases are synchronized with each other.
- In two-phase commit, commit or rollback is performed by two phases given below:
  - **Commit request phase**: In this phase, the main process or co-ordinator process take a vote of all other process that they have completed their process successfully and ready to commit, if all the votes are “yes” then they continue for the next phase. And if “No” then rollback will be performed.
  - **Commit phase**: As per vote, if all the votes are “yes” then commit is done.
- In the same way, when any transaction changes multiple databases after transaction execution, it will issue a pre-commit command on each database and all databases will send an acknowledgment. Based on acknowledgment, if all are positive transactions then it will issue the commit command otherwise rollback will be done.

34. What are the isolation levels of connections in JDBC?

- The transaction isolation level is a value that decides the level at which inconsistent data is permitted in a transaction, which means it represents the degree of isolation of one transaction from another. A higher level of isolation will result in improvement of data accuracy, but it might decrease the number of concurrent transactions. Similarly, a lower level of isolation permits for more concurrent transactions, but it reduces the data accuracy.
- To ensure data integrity during transactions in JDBC, the DBMS make use of locks to prevent access to other accesses to the data which is involved in the transaction. Such locks are necessary for preventing Dirty Read, Non-Repeatable Read, and Phantom-Read in the database.
- It is used for the locking mechanism by DBMS and can be set using setTransactionIsolation() method. You can obtain details about the level of isolation used by the connection using getTransactionIsolation() method.


Diffrent Isolation Levels in JDBC

Isolation Level	Transaction	Dirty Read	Non-Repeatable Read	Phantom Read
TRANSACTION_NONE	Not Supported	Not applicable	Not applicable	Not applicable
TRANSACTION_READ_COMMITTED	Supported	Prevented	Allowed	Allowed
TRANSACTION_READ_UNCOMMITTED	Supported	Allowed	Allowed	Allowed
TRANSACTION_REPEATABLE_READ	Supported	Prevented	Prevented	Allowed
TRANSACTION_SERIALIZABLE	Supported	Prevented	Prevented	Prevented

35. How to create a table dynamically from a JDBC application?

We can dynamically create a table by using the following code:

```
//import section
import java.sql.*;
import java.io.*;
public class CreateTableEx
{
 public static void main(String[] args)throws Exception
 {
   //create an objet of buffered reader
   BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
   //load and register the driver
   Class.forName(“sun.jdbc.odbc.JdbcOdbcDriver”);
   //establish a connection
   Connection con = DriverManager.getConnection(“jdbc:odbc:nag”,”system”,”naveen”);
   //create a statement object
   Statement st = con.createStatement();
   //receive dynamic input as a table name
   System.out.println(“Enter table name”);
   String tablename = br.readLine();
   //execute SQL query
   St.executeUpdate("create table"+tablename+"(empno number,empname varchar2(10),empsal
   System.out.println(“Successfully created the table”);
   //close the connection
   con.close();
 }
}
```

36. Conclusion

This article covers freshers to experienced level interview questions on JDBC. We hope that this will give you an overview of JDBC interview questions. JDBC questions are an important part of any Java interview, whether it is a core Java or Java EE Interview. The explanations given in this article will enrich your knowledge and increase your understanding of JDBC. All the Best…!!!

References:

“JDBC API Tutorial and Reference” by Fisher, Maydene-Ellis, Jon-Bruce, Jonathan.
“FROM ZERO TO JDBC HERO” by Vivian Siahaan, Rismon Hasiholan Sianipar.
“JDBC Pocket Reference” by Donald Bales.
https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html
https://www.youtube.com/playlist?list=PLsyeobzWxl7rU7Jz3zDRpqB-EODzBbHOI

Links to More Interview Questions

C Interview Questions	Php Interview Questions	C Sharp Interview Questions
Web Api Interview Questions	Hibernate Interview Questions	Node Js Interview Questions
Cpp Interview Questions	Oops Interview Questions	Devops Interview Questions
Machine Learning Interview Questions	Docker Interview Questions	Mysql Interview Questions
Css Interview Questions	Laravel Interview Questions	Asp Net Interview Questions
Django Interview Questions	Dot Net Interview Questions	Kubernetes Interview Questions
Operating System Interview Questions	React Native Interview Questions	Aws Interview Questions
Git Interview Questions	Java 8 Interview Questions	Mongodb Interview Questions
Dbms Interview Questions	Spring Boot Interview Questions	Power Bi Interview Questions
Pl Sql Interview Questions	Tableau Interview Questions	Linux Interview Questions
Ansible Interview Questions	Java Interview Questions	Jenkins Interview Questions