**What is JDBC API and when do we use it?**

Java DataBase Connectivity API allows us to work with relational databases. JDBC API interfaces and classes are part of java.sql and javax.sql package. We can use JDBC API to get the database connection, run SQL queries and stored procedures in the database server and process the results.

JDBC API is written in a way to allow loose coupling between our Java program and actual JDBC drivers that makes our life easier in switching from one database to another database servers easily.

**How does JDBC API helps us in achieving loose coupling between Java Program and JDBC Drivers API?**

JDBC API uses Java Reflection API to achieve loose coupling between java programs and JDBC Drivers. If you look at a simple JDBC example, you will notice that all the programming is done in terms of JDBC API and Driver comes in picture only when it’s loaded through reflection usingClass.forName() method.

**What is JDBC Connection? Explain steps to get Database connection in a simple java program.**

JDBC Connection is like a Session created with the database server. You can also think Connection is like a Socket connection from the database server.

Steps to get Database connection

•Register the driver class

•Creating connection

•Prepare SQL statement

•Create JDBC Statement

•Submit queries to Database

•Process the Result

•Closing connection

Connection con = null;

try{

// load the Driver Class

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

// create the connection now

con = DriverManager.getConnection("jdbc:mysql://localhost:3306/UserDB",

"pankaj","pankaj123");

}catch (SQLException e) {

System.out.println("Check database is UP and configs are correct");

e.printStackTrace();

}catch (ClassNotFoundException e) {

System.out.println("Please include JDBC MySQL jar in classpath");

e.printStackTrace();

}

}

### What is the use of JDBC DriverManager class?

### JDBC DriverManager is the factory class through which we get the Database Connection object. When we load the JDBC Driver class, it registers itself to the DriverManager, you can look up the JDBC Driver classes source code to check this.Then when we call DriverManager.getConnection() method by passing the database configuration details, DriverManager uses the registered drivers to get the Connection and return it to the caller program.

### How to get the Database server details in java program?

### We can use DatabaseMetaData object to get the database server details. When the database connection is created successfully, we can get the meta data object by calling getMetaData() method. There are so many methods in DatabaseMetaData that we can use to get the database product name, it’s version and configuration details.

### DatabaseMetaData metaData = con.getMetaData();

### String dbProduct = metaData.getDatabaseProductName();

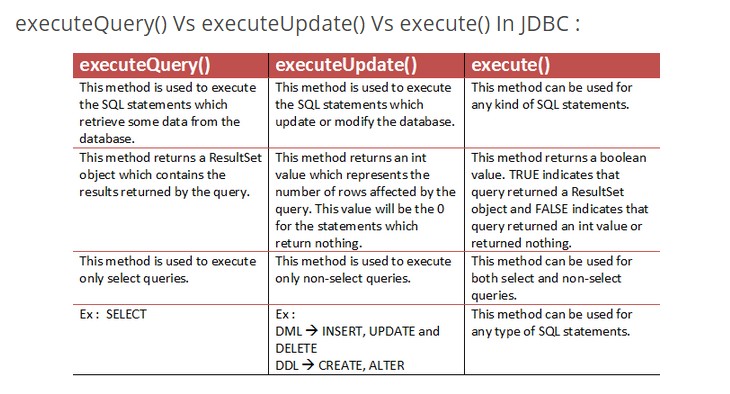
**What is JDBC Statement?**

JDBC API Statement is used to execute SQL queries in the database. We can create the Statement object by calling Connection getStatement() method. We can use Statement to execute static SQL queries by passing query through different execute methods such as execute(), executeQuery(), executeUpdate() etc.

Since the query is generated in the java program, if the user input is not properly validated it can lead to SQL injection issue.

By default, only one ResultSet object per Statement object can be open at the same time. Therefore, if we want to work with multiple ResultSet objects, then each must have been generated by different Statement objects. All execute() methods in the Statement interface implicitly close a statment’s current ResultSet object if an open one exists.

### What is the difference between execute, executeQuery, executeUpdate?

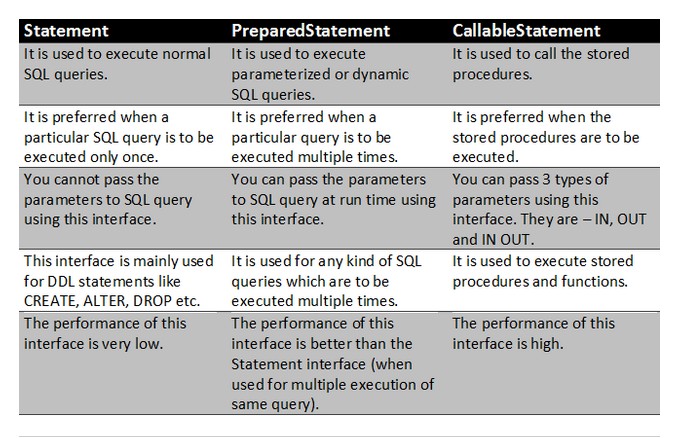


**What is JDBC PreparedStatement?**

JDBC PreparedStatement object represents a precompiled SQL statement. We can use it’s setter method to set the variables for the query.

Since PreparedStatement is precompiled, it can then be used to efficiently execute this statement multiple times. PreparedStatement is better choice that Statement because it automatically escapes the special characters and avoid SQL injection attacks.

**What are the difference between JDBC’s Statement, PreparedStatement, and CallableStatement**

****

**How to set NULL values in JDBC PreparedStatement?**

We can use PreparedStatement setNull() method to bind the null variable to a parameter. The setNull method takes index and SQL Types as argument, for example

ps.setNull(10, java.sql.Types.INTEGER);.

**What is the use of getGeneratedKeys() method in Statement?**

Sometimes a table can have auto generated keys used to insert the unique column value for primary key. We can use Statement getGeneratedKeys() method to get the value of this auto generated key.

**What are the benefits of PreparedStatement over Statement?**

Some of the benefits of PreparedStatement over Statement are:

PreparedStatement helps us in preventing SQL injection attacks because it automatically escapes the special characters.

PreparedStatement allows us to execute dynamic queries with parameter inputs.

PreparedStatement is faster than Statement. It becomes more visible when we reuse the PreparedStatement or use it’s batch processing methods for executing multiple queries.

PreparedStatement helps us in writing object Oriented code with setter methods whereas with Statement we have to use String Concatenation to create the query. If there are multiple parameters to set, writing Query using String concatenation looks very ugly and error prone.

**What is the limitation of PreparedStatement and how to overcome it?**

One of the limitation of PreparedStatement is that we can’t use it directly with IN clause statements. Some of the alternative approaches to use PreparedStatement with IN clause are;

Execute Single Queries – very slow performance and not recommended

Using Stored Procedure – Database specific and hence not suitable for multiple database applications.

Creating PreparedStatement Query dynamically – Good approach but looses the benefit of cached PreparedStatement.

Using NULL in PreparedStatement Query – A good approach when you know the maximum number of variables inputs, can be extended to allow unlimited parameters by executing in parts.

**What is JDBC ResultSet?**

JDBC ResultSet is like a table of data representing a database result set, which is usually generated by executing a statement that queries the database.

ResultSet object maintains a cursor pointing to its current row of data. Initially the cursor is positioned before the first row. The next() method moves the cursor to the next row. If there are no more rows, next() method returns false and it can be used in a while loop to iterate through the result set.

A default ResultSet object is not updatable and has a cursor that moves forward only. Thus, you can iterate through it only once and only from the first row to the last row. It is possible to produce ResultSet objects that are scrollable and/or updatable using below syntax.

Statement stmt = con.createStatement(ResultSet.TYPE\_SCROLL\_INSENSITIVE, ResultSet.CONCUR\_UPDATABLE);

A ResultSet object is automatically closed when the Statement object that generated it is closed, re-executed, or used to retrieve the next result from a sequence of multiple results.

We can use ResultSet getter method with column name or index number starting from 1 to retrieve the column data.

**What are different types of ResultSet?**

There are different types of ResultSet objects that we can get based on the user input while creating the Statement. If you will look into the Connection methods, you will see that createStatement() and prepareStatement() method are overloaded to provide ResultSet type and concurrency as input argument.

There are three types of ResultSet object.

ResultSet.TYPE\_FORWARD\_ONLY: This is the default type and cursor can only move forward in the result set.

ResultSet.TYPE\_SCROLL\_INSENSITIVE: The cursor can move forward and backward, and the result set is not sensitive to changes made by others to the database after the result set was created.

ResultSet.TYPE\_SCROLL\_SENSITIVE: The cursor can move forward and backward, and the result set is sensitive to changes made by others to the database after the result set was created.

Based on the concurrency there are two types of ResultSet object.

ResultSet.CONCUR\_READ\_ONLY: The result set is read only, this is the default concurrency type.

ResultSet.CONCUR\_UPDATABLE: We can use ResultSet update method to update the rows data.

**What is the use of setFetchSize() and setMaxRows() methods in Statement?**

We can use setMaxRows(int i) method to limit the number of rows that the database returns from the query. You can achieve the same thing using SQL query itself. For example in MySQL we can use LIMIT clause to set the max rows that will be returned by the query.

Understanding fetchSize can be tricky, for that you should know how Statement and ResultSet works. When we execute a query in the database, the result is obtained and maintained in the database cache and ResultSet is returned. ResultSet is the cursor that has the reference to the result in the database.

Let’s say we have a query that returns 100 rows and we have set fetchSize to 10, so in every database trip JDBC driver will fetch only 10 rows and hence there will be 10 trips to fetch all the rows. Setting optimal fetchSize is helpful when you need a lot of processing time for each row and number of rows in the result is huge.

We can set fetchSize through Statement object but it can be overridden through ResultSet object setFetchSize() method.

**How to use JDBC API to call Stored Procedures?**

Stored Procedures are group of SQL queries that are compiled in the database and can be executed from JDBC API. JDBC CallableStatement can be used to execute stored procedures in the database. The syntax to initialize CallableStatement is;

CallableStatement stmt = con.prepareCall("{call insertEmployee(?,?,?,?,?,?)}");

stmt.setInt(1, id);

stmt.setString(2, name);

stmt.setString(3, role);

stmt.setString(4, city);

stmt.setString(5, country);

//register the OUT parameter before calling the stored procedure

stmt.registerOutParameter(6, java.sql.Types.VARCHAR);

stmt.executeUpdate();

We need to register the OUT parameters before executing the CallableStatement.

**What is JDBC Batch Processing and what are it’s benefits?**

Sometimes we need to run bulk queries of similar kind for a database, for example loading data from CSV files to relational database tables. As we know that we have option to use Statement or PreparedStatement to execute queries. Apart from that JDBC API provides Batch Processing feature through which we can execute bulk of queries in one go for a database.

JDBC API supports batch processing through Statement and PreparedStatement addBatch() andexecuteBatch() methods.

Batch Processing is faster than executing one statement at a time because the number of database calls are less,

**What is JDBC Transaction Management and why do we need it?**

By default when we create a database connection, it runs in auto-commit mode. It means that whenever we execute a query and it’s completed, the commit is fired automatically. So every SQL query we fire is a transaction and if we are running some DML or DDL queries, the changes are getting saved into database after every SQL statement finishes.

Sometimes we want a group of SQL queries to be part of a transaction so that we can commit them when all the queries runs fine and if we get any exception, we have a choice of rollback all the queries executed as part of the transaction.

JDBC API provide method setAutoCommit(boolean flag) through which we can disable the auto commit feature of the connection. We should disable auto commit only when it’s required because the transaction will not be committed unless we call the commit() method on connection. Database servers uses table locks to achieve transaction management and it’s resource intensive process. So we should commit the transaction as soon as we are done with it.

**How to rollback a JDBC transaction?**

We can use Connection object rollback() method to rollback the transaction. It will rollback all the changes made by the transaction and release any database locks currently held by this Connection object.

**What is JDBC Savepoint? How to use it?**

Sometimes a transaction can be group of multiple statements and we would like to rollback to a particular point in the transaction. JDBC Savepoint helps us in creating checkpoints in a transaction and we can rollback to that particular checkpoint.

Any savepoint created for a transaction is automatically released and become invalid when the transaction is committed, or when the entire transaction is rolled back. Rolling a transaction back to a savepoint automatically releases and makes invalid any other savepoints that were created after the savepoint in question.

**What is JDBC DataSource and what are it’s benefits?**

JDBC DataSource is the interface defined in javax.sql package and it is more powerful that DriverManager for database connections. We can use DataSource to create the database connection and Driver implementation classes does the actual work for getting connection. Apart from getting Database connection, DataSource provides some additional features such as:

•Caching of PreparedStatement for faster processing

•Connection timeout settings

•Logging features

•ResultSet maximum size threshold

•Connection Pooling in servlet container using JNDI support

**How to achieve JDBC Connection Pooling using JDBC DataSource and JNDI in Apache Tomcat Server?**

For web applications deployed in a servlet container, creating JDBC connection pool is very easy and involve only few steps.

Creating JDBC JNDI resource in the container configuration files, usually server.xml or context.xml. For example

server.xml

<Resource name="jdbc/MyDB"

global="jdbc/MyDB"

auth="Container"

type="javax.sql.DataSource"

driverClassName="com.mysql.jdbc.Driver"

url="jdbc:mysql://localhost:3306/UserDB"

username="ROOT"

password="123"

maxActive="100"

maxIdle="20"

minIdle="5"

maxWait="10000"/>

context.xml

<ResourceLink name="jdbc/MyLocalDB" global="jdbc/MyDB"

auth="Container"

type="javax.sql.DataSource" />

In web application, using InitialContext to look up the JNDI resource configured in the first step and then get the connection.

Context ctx = new InitialContext();

DataSource ds = (DataSource) ctx.lookup("java:/comp/env/jdbc/MyLocalDB");

**What is Apache DBCP API?**

If you use DataSource to get the Database connection, usually the code to get the connection is tightly coupled with the Driver specific DataSource implementation. Also most of the code is boiler-plate code except the choice of the DataSource implementation class.

Apache DBCP helps us in getting rid of these issues by providing DataSource implementation that works as an abstraction layer between our program and different JDBC drivers. Apache DBCP library depends on Commons Pool library, so make sure they both are in the build path.

What is JDBC RowSet? What are different types of RowSet?

JDBC RowSet holds tabular data in more flexible ways that ResultSet. All RowSet objects are derived from ResultSet, so they have all the capabilities of ResultSet with some additional features. RowSet interface is defined in javax.sql package.

Some additional features provided by RowSet are:

Functions as Java Beans with properties and their getter-setter methods. RowSet uses JavaBeans event model and they can send notifications to any registered component for events such as cursor movement, update/insert/delete of a row and change to RowSet contents.

RowSet objects are scrollable and updatable by default, so if DBMS doesn’t support scrollable or updatable ResultSet, we can use RowSet to get these features.

**RowSet are broadly divided into two types:**

Connected RowSet Objects – These objects are connected to database and are most similar to ResultSet object. JDBC API provides only one connected RowSet objectjavax.sql.rowset.JdbcRowSet and it’s standard implementation class iscom.sun.rowset.JdbcRowSetImpl

Disconnected RowSet Objects – These RowSet objects are not required to connected to a database, so they are more lightweight and serializable. They are suitable for sending data over a network. There are four types of disconnected RowSet implementations.

CachedRowSet – They can get the connection and execute a query and read the ResultSet data to populate the RowSet data. We can manipulate and update data while it is disconnected and reconnect to database and write the changes.

WebRowSet derived from CachedRowSet – They can read and write XML document.

JoinRowSet derived from WebRowSet – They can form SQL JOIN without having to connect to a data source.

FilteredRowSet derived from WebRowSet – We can apply filtering criteria so that only selected data is visible.

**What is the different between ResultSet and RowSet?**

RowSet objects are derived from ResultSet, so they have all the features of ResultSet with some additional features. One of the huge benefit of RowSet is that they can be disconnected and that makes it lightweight and easy to transfer over a network.

Whether to use ResultSet or RowSet depends on your requirements but if you are planning to use ResultSet for longer duration, then a disconnected RowSet is better choice to free database resources.

**What are common JDBC Exceptions?**

Some of the common JDBC Exceptions are:

java.sql.SQLException – This is the base exception class for JDBC exceptions.

java.sql.BatchUpdateException – This exception is thrown when Batch operation fails, but it depends on the JDBC driver whether they throw this exception or the base SQLException.

java.sql.SQLWarning – For warning messages in SQL operations.

java.sql.DataTruncation – when a data values is unexpectedly truncated for reasons other than its having exceeded MaxFieldSize.

**What is CLOB and BLOB datatypes in JDBC?**

Character Large OBjects (CLOBs) are character string made up of single-byte characters with an associated code page. This data type is appropriate for storing text-oriented information where the amount of information can grow beyond the limits of a regular VARCHAR data type (upper limit of 32K bytes).

Binary Large OBjects (BLOBs) are binary string made up of bytes with no associated code page. This data type can store binary data larger than VARBINARY (32K limit). This data type is good for storing image, voice, graphical, and other types of business or application-specific data.

**What is “dirty read” in JDBC? Which isolation level prevents dirty read?**

When we work with transactions, there is a chance that a row is updated and at the same time other query can read the updated value. This results in a dirty read because the updated value is not permanent yet, the transaction that has updated the row can rollback to previous value resulting in invalid data.

Dirty Read is prevented by isolation levels TRANSACTION\_READ\_COMMITTED, TRANSACTION\_REPEATABLE\_READ and TRANSACTION\_SERIALIZABLE.

**What is 2 phase commit?**

Two phase commit is used in distributed environment where multiple process take part in distributed transaction process. In simple word we can understand like if any transaction is executing and it will effect multiple database then two phase commit will be used to make all database synchronized with each other.

In two phase commit, commit or rollback is done by two phases:

1. Commit request phase: in this phase main process or coordinator process take vote of all other process that they are complete their process successfully and ready to commit if all the votes are “yes” then they go ahead for next phase. And if “No “then rollback is performed.

2. Commit phase: according to vote if all the votes are yes then commit is done.

Similarly when any transaction changes multiple database after execution of transaction it will issue pre commit command on each database and all database send acknowledgement and according to acknowledgement if all are positive transaction will issue the commit command otherwise rollback is done .

**What are the different types of locking in JDBC?**

On a broad level, there are two types of locking mechanism to prevent data corruption because of more than one user working with the same data.

Optimistic Locking – Locking the record only when update is taking place

Pessimistic Locking – Locking the record from the select to read, update and commit phase.

Apart from that some DBMS systems provide locking mechanism to lock single row, table or database.

**What do you understand by DDL and DML statements?**

Data Definition Language (DDL) statements are used to define the database schema. Create, Alter, Drop, Truncate, Rename statements comes under DDL statements and usually they don’t return any result.

Data Manipulation Language (DML) statements are used to manipulate data in the database schema. Select, Insert, Update, Delete, Call etc are example of DML statements.

**What is difference between java.util.Date and java.sql.Date?**

java.util.Date contains information about the date and time whereas java.sql.Date contains information only about the date, it doesn’t have time information. So if you have to keep time information in the database, it is advisable to use Timestamp or DateTime fields.

**How to insert an image or raw data into database?**

We can use BLOB to insert image or raw binary data into database.

**What is phantom read and which isolation level prevents it?**

A phantom read is the situation where a transaction executes a query multiple times and get different data. Suppose a transaction is executing a query to get data based on a condition and then another transaction inserts a row that matches the condition. Now when same transaction will execute the query again, a new row will be part of the result set. This new row is referred as Phantom Row and this situation is termed as Phantom Read.

Phantom read can be prevented only with TRANSACTION\_SERIALIZABLE isolation level.

**What is SQL Warning? How to retrieve SQL warnings in the JDBC program?**

SQLWarning is the subclass of SQLException and we can retrieve it by calling getWarnings() method on Connection, Statement, and ResultSet objects. SQL Warnings doesn’t stops the execution of the script but alerts the user about the warning.

**How to invoke Oracle Stored Procedure with Database Objects as IN/OUT?**

If Oracle Stored Procedure has IN/OUT parameters as DB Objects then we need to create an Object array of the same size in the program and then use it to create Oracle STRUCT object. Then we can set this STRUCT object for the database object by calling setSTRUCT() method and work with it.

**What are different types of Statement?**

This is another classical JDBC interview question. Variants are Difference between Statement, PreparedStatemetn and CallableStatement in Java. Statement object is used to send SQL query to database and get result from database, and we get statement object from connection object.

**There are three types of statement:**

**1. Statement**: it’s a commonly used for getting data from database useful when we are using static SQL statement at runtime. it will not accept any parameter.

Statement stmt = conn.createStatement( );

ResultSet rs = stmt.executeQuery();

**2. PreparedStatement:** when we are using same SQL statement multiple time its is useful and it will accept parameter at runtime.

String SQL = "Update stock SET limit = ? WHERE stockType = ?";

PreparedStatement pstmt = conn.prepareStatement(SQL);

ResultSet rs = pstmt.executeQuery();

**3. Callable Statement:** when we want to access stored procedures then callable statement are useful and they also accept runtime parameter. It is called like this

CallableStatement cs = con.prepareCall("{call SHOW\_SUPPLIERS}");

ResultSet rs = cs.executeQuery();

**How cursor works in scrollable result set?**

In JDBC 2.0 API new feature is added to move cursor in resultset backward forward and also in a particular row .

There are three constant define in result set by which we can move cursor.

**TYPE\_FORWARD\_ONLY:** creates a nonscrollable result set, that is, one in which the cursor moves only forward

**TYPE\_SCROLL\_INSENSITIVE** : a scrollable result set does not reflects changes that are made to it while it is open

·**TYPE\_SCROLL\_SENSITIVE:** a scrollable result set reflects changes that are made to it while it is open

**What do you mean by cold backup, hot backup?**

This question is not directly related to JDBC but some time asked during JDBC interviews. Cold back is the backup techniques in which backup of files are taken before the database restarted. In hot backup backup of files and table is taken at the same time when database is running. A warm is a recovery technique where all the tables are locked and users cannot access at the time of backing up data.

**Does the JDBC-ODBC Bridge support multiple concurrent open statements per connection?**

No, we can open only one statement object when using JDBC-ODBC Bridge.