Arundeep Singh

1. What is JDBC driver?

A JDBC driver is a software component enabling a Java application to interact with a database.

1. What are the steps to connect to the database in java?

Registering the driver class

Creating connection

Creating statement

Executing queries

Closing connection

1. What are the JDBC API components?

**Interfaces are:**

Connection

Statement

PreparedStatement

ResultSet

ResultSetMetaData

DatabaseMetaData

CallableStatement

**Classes:**

DriverManager

Types

SQLException

1. What is the difference between Statement and PreparedStatement interface?

|  |  |
| --- | --- |
| Statement | PreparedStatement |
| Statement object is used to run static SQL queries as it cannot accept parameters. | PreparedStatement object is used to run dynamic SQL queries as it can accept parameters at runtime. |
| When Statement object is used, the SQL query is not precompiled. | The query is supplied to the object when it is created and It is precompiled at that time. So it is used when the query needs to be executed many times. |
| It is the basic type of interface provided in java to run SQL queries. | PreparedStaemnt extends the Statement interface and thus provides all the features of Statement and also the above mentioned advantages over the same. |

1. What are the benefits of PreparedStatement over Statement?

Prepared Statement queries are **pre-compiled on database** and there access plan will be reused to execute further queries which allows them to execute much quicker than normal queries generated by Statement object

1. Which interface is responsible for transaction management in JDBC?

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

1. What does the JDBC ResultSetMetaData interface?

The ResultSetMetaData interface returns the information of table such as total number of columns, column name, column type etc.

1. How to set NULL values in JDBC PreparedStatement?

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);.

1. What are different types of ResultSet?
2. What are different types of ResultSet?

There are three types of ResultSet object.

1. **ResultSet.TYPE\_FORWARD\_ONLY**: This is the default type and cursor can only move forward in the result set.
2. **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.
3. **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.

1. **ResultSet.CONCUR\_READ\_ONLY**: The result set is read only, this is the default concurrency type.
2. **ResultSet.CONCUR\_UPDATABLE**: We can use ResultSet update method to update the rows data.
3. What is JDBC Batch Processing and what are it’s benefits?

When we have to run bulk of queries of similar kind for a database, we have option of using Statement otr PreparedStatement to execute those queries. Apart from JDBC API provides Batch Processing feature which can execute bulk of queries in one go for a database. JDBC API supports batch processing through Statement and PreparedStatement addBatch() and executeBatch() methods.

Benefits of using Batch processing is this that it is faster than executing one statement at a time because the number of database calls are less.

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

When we create a database connection, by default it runs in auto-commit mode. It means whenever we execute a query and it’s completed, the commit is done. So every SQL query fired is a transaction and all the finished queries are getting saved into database after every SQL statement finishes. Sometimes we want group of queries to commit once all the queries run fine so that we can have 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.

1. 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

1. 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.

1. 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:

1. **Connected RowSet Objects** – These objects are connected to database and are most similar to ResultSet object. JDBC API provides only one connected RowSet object javax.sql.rowset.JdbcRowSet and it’s standard implementation class iscom.sun.rowset.JdbcRowSetImpl
2. **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.
3. What are common JDBC Exceptions?

Some of the common JDBC Exceptions are:

1. java.sql.SQLException – This is the base exception class for JDBC exceptions.
2. 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.
3. java.sql.SQLWarning – For warning messages in SQL operations.
4. java.sql.DataTruncation – when a data values is unexpectedly truncated for reasons other than its having exceeded MaxFieldSize.
5. 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.

1. What is 2 phase commit?

When we work in distributed systems where multiple databases are involved, we are required to use 2 phase commit protocol. 2 phase commit protocol is an atomic commitment protocol for distributed systems. In the first phase, transaction manager sends commit-request to all the transaction resources. If all the transaction resources are OK, then transaction manager commits the transaction changes for all the resources. If any of the transaction resource responds as Abort, then the transaction manager can rollback all the transaction changes.

1. 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, and Call etc. are example of DML statements.

1. 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.

1. How to insert an image or raw data into database?

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

1. 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.

* 1. Optimistic Locking – Locking the record only when update is taking place
  2. 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.