Top 40 JDBC interview questions and answers in java

Time to impress interviewer, crack JDBC interview questions in java. Best set of questions, your interview will comprise of mostly these JDBC questions. I have tried to cover almost all the possible JDBC questions which could be framed in an interview by interviewer.

JDBC interview Question 1. What is JDBC?

Answer. It's the basic JDBC interview question. JDBC stands for Java database connectivity, it allows you to connect to database from java code. To start up you must read Oracle 11g and SQL Developer installation.

JDBC interview Question 2. What are different type of JDBC drivers? And which all drivers you have us

Answer. It's very popular JDBC interview question. There are 4 types of JDBC driver.
Type 1 JDBC driver - JDBC-ODBC Bridge driver (Bridge driver)
Type 2 JDBC driver - Native-API/partly Java driver (Native driver)
Type 3 JDBC driver - All Java/Net-protocol driver (Middleware driver)
Type 4 JDBC driver - All Java/Native-protocol driver (Pure java driver)

Type 1 JDBC driver - JDBC-ODBC Bridge driver (Bridge driver)

- Type 1 JDBC driver **translates all JDBC calls into the ODBC calls**.
- Then **send** those **ODBC calls to the ODBC driver.**
- **ODBC driver sends those calls to database.**

Type 2 JDBC driver - Native-API/partly Java driver (Native driver)

Type 2 JDBC drivers **converts all the JDBC calls into database specific calls.**

Example: When we use Oracle database -
Type 2 JDBC drivers converts all the JDBC calls into **Oracle** database specific calls using the **oracle native api's**.

Type 3 JDBC driver - All Java/Net-protocol driver (Middleware driver)

- In Type 3 JDBC driver, the database **requests are passed to the middle-tier server** through the network.
- Then **middle-tier translates and sends the request to the database**. (middle-tier server may use Type 1, Type 2 or Type 4 drivers)

Type 4 JDBC driver - All Java/Native-protocol driver (Pure java driver)

- Type JDBC 4 drivers uses **java api's to communicate directly with the database server.**
- Type JDBC 4 driver is most popular and widely used driver.
- Type JDBC 4 drivers are purely written in java,

And regarding driver you have used - These days probably you must be working in Type 4 JDBC driver, it's the most popular and widely used d

JDBC interview Question 3. How to connect to database from java code using JDBC in java?

Answer. Every fresher must know this answer. Interviewers tend to know whether interviewee knows basic steps to connect with database o
It's a very easy interview question.

Registering the Driver class >

Here we will be registering Oracle driver class

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

OR

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

Setting up connection with Oracle database >

```
Connection con =
DriverManager.getConnection("jdbc:oracle:thin:@Hostname:Port:SID",
                            "Username", "Password");
```

Get the java.sql.Connection object >

Hostname = localHost,
Port =1521 (default port)
SID= orcl
Username =ankit
Password=Oracle123

```
Connection con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl",
                            "ankit", "Oracle123");
```

For full program read : JDBC tutorial connection with **Oracle 11g**

You must read : JDBC connection with **MySql** database

JDBC connection with **MsSql (Microsoft)** database

JDBC connection with **PostgreSQL** database

JDBC interview Question 4. What is Statement in JDBC java?

Answer. These 11 points will describe Statement in detail.

1. The **java.sql.Statement** object used for executing a static SQL statement and returning the results it produces.

2. Statement **cannot accept parameters at runtime**.

3. Statement is **slower** as compared to PreparedStatement.

4. Statement is **suitable for executing DDL commands** - CREATE, drop, alter and truncate.

5. Statement **can't be used for storing/retrieving image and file in database** (i.e. using BLOB, CLOB datatypes)

6. Statement **enforces SQL injection**, because we end up using query formed using **concatenated SQL strings**.

   Example >

```
String s1= "select * from EMPLOYEE where id = ";
int i1 = 2 ;
stmt = con.createStatement();
rs = stmt.executeQuery(s1 + String.valueOf(i1));
```

7. same SQL query **can't** be **executed repeatedly in** Statement **.**

8. Statement makes **code less readable and understandable - We may need to write concatenated SQL strings**

9. java.sql.**Statement** is an interface.

10. By default, only one ResultSet object per Statement object is allowed to be opened at the same time.

11. java.sql.Statement Important **methods** -

    - executeUpdate
    - executeQuery
    - addBatch("sql")
    - executeBatch()
    - getResultSet
    - execute
    - getUpdateCount

Read: **What is java.sql.Statement in java**

JDBC interview Question 5. What is PreparedStatement in JDBC java?

Answer. This is important interview question for freshers. These 12 points will describe PreparedStatement in detail.

1. PreparedStatement is used for **executing a precompiled SQL statement.**

2. PreparedStatement can be **executed repeatedly**, it **can accept different parameters at runtime**.

3. PreparedStatement is **faster as compared to java.sql.Statement** because it is used for executing precompiled SQL statement

4. Prepared statements are executed through a **non sql binary protocol.**

   In binary protocol communications to the server is **faster** because **less data packets are transferred**.

5. PreparedStatement is **suitable for executing DML commands** - SELECT, INSERT, UPDATE and DELETE

6. PreparedStatement **can be used for**

   storing/*retrieving* image and

   ***Storing* /*retrieving* file in database**

   (i.e. by using BLOB, CLOB datatypes)

7. PreparedStatement **can be used for setting java.sql.Array using setArray method.**

   While sending it to database the driver converts this java.sql.Array to an SQL ARRAY

8. PreparedStatement **prevents SQL injection**, because text for all the parameter values is escaped.

   Example >

```
prepStmt = con.prepareStatement("select * from EMPLOYEE where ID=? ");
prepStmt.setInt(1, 8);
```

QUESTION. Here comes one very **important** question, are **PreparedStatement vulnerable to SQL injections?**
ANSWER. YES, when we use **concatenated SQL strings** rather than using input as a parameter for preparedStatement

9. PreparedStatement extends Statement and inherits all methods from Statement and additionally adds **addBatch()** method.

   addBatch() method - adds a set of parameters to the PreparedStatement object's batch of commands.

   Hence, same SQL query **can** be **executed repeatedly in** PreparedStatement**.**

10. PreparedStatement provides methods like **getMetadata()** and **getParameterMetadata()**

    - **getMetadata() -** Method retrieves ResultSetMetaData object that contains information about the **columns of the ResultSet object** th
      be returned when PreparedStatement object is executed.

    - **getParameterMetadata() -** method retrieves the number, types and properties of PreparedStatement object's parameters.

11. java.sql.**PreparedStatement** is an interface.

12. java.sql.Statement Important **methods** -

    - executeUpdate
    - executeQuery
    - addBatch()
    - executeBatch()

```
prepStmt.setInt(1, 8); //substitute first occurrence of ? with 8
prepStmt.setString(2, "javaMadeSoEasy"); //substitute second occurrence of ? with "javaMadeSoEasy"
```

    - Storing **File and Image** in database methods -
      - setBinaryStream
      - setCharacterStream

Read : JDBC tutorial - What is java.sql.PreparedStatement in java

JDBC interview Question 6. What is CallableStatement in JDBC java? When to use CallableStatement in java?

Answer. Another important interview question for freshers. These 12 points will describe CallableStatement in detail.

1. The java.sql.**CallableStatement** is an interface which is used to **execute SQL stored procedures and Functions in java.**

2. **CallableStatement** extends the PreparedStatement interface.

3. As CallableStatement is sub interface of PreparedStatement it adds a level of abstraction, so the execution of stored procedure or function
   not to be dbms specific.
4. **How to deal with SQL stored procedures IN, OUT and IN OUT parameter in java-**
   a. **How to deal with SQL stored procedures IN parameter in java-**
      1) set methods are used for setting IN parameter values.
         (Must know : set methods are inherited from java.sql.PreparedStatement)

b. **How to deal with SQL stored procedures OUT parameter in java-**

1) OUT parameters must be registered in java before executing the stored procedure,
2) Execute database stored procedure,
3) Then retrieve values of OUT parameters using using get methods.

c. **How to deal with SQL stored procedures IN OUT parameter in java-**

1. Like IN parameters, **set methods are used for setting IN OUT parameter values.**
2. Like OUT parameters, **IN OUT parameters must be registered in java before executing the stored procedure,**
3. Execute database stored procedure,
4. Then like OUT parameters, **retrieve values of IN OUT parameters using using get methods.**

5. **Important methods of** java.sql.**CallableStatement -**

a. Calling Oracle database STORED PROCEDURE and pass its IN parameter
   i. **setInt**
   ii. **setString**
   iii. **setDate**
   iv. **executeUpdate**

b. Calling Oracle database STORED PROCEDURE- OUT parameter - CallableStatement example in java
   c. **registerOutParameter**
   d. **getString**
   e. **getInt**
   f. **getDate**

Programs where java.sql.***CallableStatement*** is used >

Read : JDBC tutorial - What is java.sql.CallableStatement in java

JDBC interview Question 7. How to execute database Stored procedures in JDBC java?

Answer. It is another very important JDBC interview question.

Please Read : Use CallableStatement to execute *Execute database*

- *STORED PROCEDURE* - IN parameter,
- *STORED PROCEDURE* - OUT parameter *and*
- *STORED PROCEDURE* - IN OUT parameter

JDBC interview Question 8. How to call database cursors from java??

Answer. It might sound to be a tricky and difficult interview questions but the solution is very easy.

Read : JDBC program/example to Call/Execute Oracle CURSOR in java

JDBC interview Question 9. What are differences between Statement, PreparedStatement and CallableStatement in JDBC java?

Answer.

	*java.sql.**Statement***	*java.sql.**PreparedStatement***
1	Statement is used for **executing a static SQL statement.**	PreparedStatement is used for **executing a precompiled SQL statement.**
2	Statement **cannot accept parameters at runtime**.	PreparedStatement can be **executed repeatedly**, it **can accept different parameters at runtime**.

3	Statement is **slower** as compared to PreparedStatement.	PreparedStatement is **faster** because it is used for executing precompiled SQL statement
4	No such protocol.	Prepared statements are executed through a **non sql binary protocol.** In binary protocol communications to the server is **faster** because **less data packets are transferred**.
5	Statement is **suitable for executing DDL commands** - CREATE, drop, alter and truncate.	PreparedStatement is **suitable for executing DML commands** - SELECT, INSERT, UPDATE and DELETE
6	Statement **can't be used for storing/retrieving image and file in database** (i.e. using BLOB, CLOB datatypes)	PreparedStatement **can be used for** **storing/*retrieving* image and** ***Storing* /*retrieving* file in database** (i.e. by using BLOB, CLOB datatypes)
7	Statement **does not have setArray method.**	PreparedStatement **can be used for setting java.sql.Array using setArray method.** **While sending it to database the driver converts this java.sql.Array to an SQL ARRAY**

8	Statement **enforces SQL injection**, because we end up using query formed using **concatenated SQL strings**.
	
	Example >
	
	`String s1= "select * from EMPLOYEE where id = ";`
	`int i1 = 2 ;`
	`stmt.executeQuery(s1 + String.valueOf(i1));`
		PreparedStatement **prevents SQL injection**, because text for all the parameter values is escaped.
		
		Example >
		
		`prepStmt = con.prepareStatement("select * from EMPLOYEE where ID=? ");`
		`prepStmt.setInt(1, 8);`
		
		Here comes one very **important** question, are **PreparedStatement vulnerable to SQL injections? YES**, when we use **concatenated SQL strings** rather than using input as a parameter for preparedStatement

9	Statement **does not provide addBatch() method, it provides only addBatch(String sql) method.** Hence, same SQL query **can't** be **executed repeatedly in** Statement **.**	PreparedStatement extends Statement and inherits all methods from Statement and additionally adds **addBatch()** method. **addBatch() method -** adds a set of parameters to the PreparedStatement object's batch of commands. Hence, same SQL query **can** be **executed repeatedly in** PreparedStatement**.**
10	Statement makes **code less readable and understandable - We may need to write**	PreparedStatement **makes code more** readable and understandable - We **need not to**

	concatenated SQL strings	**write concatenated SQL strings**, we can use queries and pass **different parameters at runtime using** setter methods.
11	Statement **does not provide such methods.**	PreparedStatement provides methods like **getMetadata()** and **getParameterMetadata()** **getMetadata() -** Method retrieves ResultSetMetaData object that contains information about the **columns of the ResultSet object** that will be returned when PreparedStatement object is executed. **getParameterMetadata() -** method retrieves the number, types and properties of PreparedStatement object's parameters.

For more read : [12 Differences between Statement, PreparedStatement and CallableStatement in JDBC java]

JDBC interview Question 10. What is java.sql.ResultSet in detail in JDBC java?

Answer. It is very important JDBC interview question for experienced developers.

1. java.sql.**ResultSet** is an interface

2. ResultSet is a table of data which represent a database result set, result is generated by executing a statement that queries the database.

3. **ResultSet object maintains a cursor pointing to its current row of data. cursor initially is pointing before the first row.**

4. The **next** method moves the cursor to the next row.
   It returns false when there are no more rows in the ResultSet object that's the reason why
   it can be used in a while loop to iterate through the ResultSet.

5. If ResultSet object does not contain any row and is used in while loop then while loop will never be executed.

Question 11. What are ResultSet Types in JDBC java?

Answer. Types of ResultSet in java >

TYPE_FORWARD_ONLY:

- In **TYPE_FORWARD_ONLY** cursor can only **move forward** on the result set, not backward.
- It is default ResultSet type in java.

Note : TYPE_FORWARD_ONLY is the default ResultSet type in java. uuu

TYPE_SCROLL_INSENSITIVE:

- In TYPE_SCROLL_INSENSITIVE cursor can move (scroll) both forward and backward,
- Cursor can be moved to absolute/specific/relative position as well.
- TYPE_SCROLL_INSENSITIVE is **not sensitive to the changes made to the data that underlies the ResultSet**. (It means that if s thread modifies the data in the database which ResultSet currently holds **won't** impact/change the already opened ResultSet's data)

TYPE_SCROLL_SENSITIVE:

- In TYPE_SCROLL_SENSITIVE cursor can move (scroll) both forward and backward,
- Cursor can be moved to absolute/specific/relative position as well.
- TYPE_SCROLL_SENSITIVE is **sensitive to the changes made to the data that underlies the ResultSet.** (It means that if some th modifies the data in the database which ResultSet currently holds **will** impact/change the already opened ResultSet's data).

Must know : **getType()** - method returns ResultSet object type. It may be TYPE_FORWARD_ONLY (1003), TYPE_SCROLL_INSENSITIVE (1004) or TYPE_SCROLL_SENSITIVE (1005)

JDBC interview Question 12. What are ResultSet concurrency in JDBC java?

Answer. ResultSet concurrency read/update >

CONCUR_READ_ONLY - CONCUR_READ_ONLY mode means ResultSet object is read only it may not be updated.

CONCUR_UPDATABLE - CONCUR_UPDATABLE mode means ResultSet object can be read and updated as well.

Note : All databases and drivers may not support CONCUR_UPDATABLE mode.

Must know : **getConcurrency()** - method returns concurrency mode of this ResultSet object. It may be CONCUR_READ_ONLY (1007) or CONCUR_UPDATABLE

JDBC interview Question 13. What are ResultSet holdability in JDBC java?

ResultSet holdability >

HOLD_CURSORS_OVER_COMMIT - HOLD_CURSORS_OVER_COMMIT holdability indicates that open ResultSet objects will remain open when the c transaction is committed.

CLOSE_CURSORS_AT_COMMIT - CLOSE_CURSORS_AT_COMMIT holdability indicates that open ResultSet objects will be closed when the current transaction is committed.

Note : All databases and drivers may not support ResultSet holdability.

Must know : **getHoldability()** - method returns holdability of this ResultSet object. It may be HOLD_CURSORS_OVER_COMMIT (1) or CLOSE_CURSORS_AT_CO (2)

JDBC interview Question 14. What are some important and most frequently used ResultSet methods in JDBC java?

Answer. ResultSet methods - To navigate over ResultSet >

first() - first method makes cursor to point to the first row in the ResultSet object.

last() - last method makes cursor to point to the last row in the ResultSet object.

next() - next method makes cursor to point to the next row in the ResultSet object.

previous() - previous method makes cursor to point to the previous row in the ResultSet object.

beforeFirst() - beforeFirst method makes cursor to point to the front of the ResultSet object, just before the first row.

afterLast() - afterLast method makes cursor to point to the last/end of the ResultSet object, just after the last row.

absolute(int row) - absolute method moves the cursor to the specified row number in this ResultSet object.

relative(int rows) - relative method moves the cursor a **relative number of rows**, either positive or negative.

ResultSet information methods >

isFirst() - method returns true if cursor points to first row in ResultSet object.

isBeforeFirst() - method returns true if cursor is before the first row in ResultSet object.

isAfterLast() - method returns true if cursor is after the last row in ResultSet object.

getRow() - method retrieves the current row number. The first row is number 1, the second number 2.

getType() - method returns ResultSet object type. It may be TYPE_FORWARD_ONLY (1003), TYPE_SCROLL_INSENSITIVE (1004) or TYPE_SCROLL_SENSITIVE (1

getConcurrency() - method returns concurrency mode of this ResultSet object. It may be CONCUR_READ_ONLY (1007) or CONCUR_UPDATABLE (1008)

getHoldability() - method returns holdability of this ResultSet object. It may be HOLD_CURSORS_OVER_COMMIT (1) or CLOSE_CURSORS_AT_COMMIT (2)

refreshRow() - method refreshes the current row with its most recent value available in the database.

Read : JDBC tutorial - What is ResultSet in java - Types, concurrency, holdability of ResultSet in java

JDBC interview Question 15. What is difference between TYPE_SCROLL_INSENSITIVE and TYPE_SCROLL_SENSITIVE ResultSet type in java? **?**

Answer.

TYPE_SCROLL_INSENSITIVE	TYPE_SCROLL_SENSITIVE
TYPE_SCROLL_INSENSITIVE is **not sensitive to the changes made to the data that underlies the ResultSet**. (It means that if some thread modifies the data in the database which ResultSet currently holds **won't** impact/change the already opened ResultSet's data).	TYPE_SCROLL_SENSITIVE is **sensitive to the changes made to the data that underlies the ResultSet.** (It means that if some thread modifies the data in the database which ResultSet currently holds **will** impact/change the already opened ResultSet's data).
getType() - method returns ResultSet object type. It returns 1004 if ResultSet object type is TYPE_SCROLL_INSENSITIVE. In ResultSet interface TYPE_SCROLL_INSENSITIVE = 1004	**getType()** - method returns ResultSet object type. It returns 1005 if ResultSet object type is TYPE_SCROLL_SENSITIVE. In ResultSet interface TYPE_SCROLL_SENSITIVE = 1005;
Example - Let' say you have multiple users in the database. You issued select query at 1:00 Than some other user updated the underlying database at 1:01	Example - Let' say you have multiple users in the database. You issued select query at 1:00 Than some other user updated the underlying database at 1:01

With TYPE_SCROLL_INSENSITIVE ResultSet, at 1:02 you will still be scrolling (using next(), previous(), first(), last(), absolute() and relative() methods) at old data fetched the at 1:00 and not the updated one.	With TYPE_SCROLL_SENSITIVE ResultSet, at 1:02 you will be scrolling (using next(), previous(), first(), last(), absolute() and relative() methods) at fresh data updated at 1:01 and not the old one.

JDBC interview Question 16. How to store and retrieve image from database?

Answer. *BLOB data type is used to store and retrieve IMAGE (Storing in and retrieving out from database)*

IMAGE - Storing in and retrieving out from database

JDBC program- Insert/Store/save IMAGE in database by using PreparedStatement's setBinaryStream and executeUpdate methods, using **BLOB data type** - in

JDBC program- Retrieve IMAGE from database by using PreparedStatement's executeQuery, ResultSet's getBlob method - using **BLOB data type** - in java

JDBC interview Question 17. How to store and retrieve file from database?

Answer. *CLOB data type is used to store and retrieve FILE (Storing in and retrieving out from database)*

FILE - Storing in and retrieving out from database

JDBC program- Insert/Store/save FILE in database by using PreparedStatement's executeUpdate and setCharacterStream methods, using **CLOB data type** - i

JDBC program- Retrieve FILE from database by using PreparedStatement's executeQuery and ResultSet's getClob method, using **CLOB data type** - in java

JDBC interview Question 18. What are Difference between CLOB and CLOB data type in Oracle?

Answer.

CLOB data type	BLOB data type
CLOB stands for **Character** Large Object.	BLOB stands for **Binary** Large Object.
CLOB stores values in **character streams**.	BLOB stores values in **bitstreams**.
CLOB is used for **storing single-byte character data** (Character string made up of single-byte character data)	BLOB is used for **storing binary data**.
CLOB data type is appropriate for storing text information. Example > • **text files**, • pdf, • doc, • docx and • odf formats.	BLOB data type is appropriate for storing following > • **image**, • graphical, • voice and • some application specific data.

Difference between CLOB and CLOB data type in Oracle

JDBC interview Question 19. What are JDBC Transactions in java?

Answer. It is very important JDBC interview question, both fresher and experienced developers must be able to answer this question. If any transaction fails in between then rollback the transaction by calling **`con.rollback()`**, commit the transaction by using **`con.commit()`** only if it successful.

We can set autocommit mode of connection to false using connection.setAutoCommit(false) and then accordingly use **`connection.commit()`** **`connection.rollback()`**

Example -
Let's say we have to update salary of two employees, and salary of both employees must be updated simultaneously in database.

And let's say salary of first employee is updated successfully. But, if anything goes wrong in updating salary of second employee than any upda done to first employee's salary will be rolled back.

Read : JDBC Transactions program- commit and rollback(TCL command) - using PreparedStatement's executeUpdate method, setting connection.setAutoCommit(false) in java

JDBC Transactions program - using Statement in java

JDBC interview Question 20. What is DriverManager class in java?

Answer. java.sql.**DriverManager** is the basic service for managing set of JDBC drivers

- DriverManager helps in establishing connection between driver and database.
- java.sql.DriverManager class allows user to customize the JDBC Drivers used in the application by because the java.sql.DriverManager cl attempts to load the driver classes referenced in the 'jdbc.drivers' system property.
- When getConnection method is called, the DriverManager attempts to locate suitable driver from amongst those which were loaded at initialization and those which were loaded explicitly using the same classloader as the current applet or application.

JDBC interview Question 21. What is Connection class in java JDBC ?

Answer. Another basic but important JDBC interview question. java.sql.**Connection** **helps in establishing connection/session with datab**

- java.sql.Connection extends java.lang.AutoCloseable interface from which java 7
- ***Important methods >***
  - Database information related methods -
    - **getMetaData** - Method returns DatabaseMetaData. DatabaseMetaData can be used to obtain information about the database as whole.
  - Transaction related methods -
    - **setAutoCommit**(boolean autoCommit) = for setting autocommit mode of transactions
    - **commit**() - method can be used to commit the transactions.
    - **rollback**() - method can be used to rollback the transactions.
    - **close**() - It closes the connection
  - Statement related methods -
    - **createStatement**() -
    - **prepareStatement**(String sql) -

JDBC interview Question 22. What is ResultSetMetaData in JDBC java?

Answer. I have seen many experienced developers unable to answer this JDBC interview question. You must know this answer.
java.sql.ResultSetMetaData is an interface in java.

- java.sql.ResultSetMetaData object can be used to get information about the types and properties of the columns in a java.sql.ResultSet ob
- java.sql.ResultSetMetaData extends java.sql.Wrapper.
- ResultSetMetaData important methods -

```
getColumnCount = To find out total number of columns in table
getColumnName =  Display table's column type
getColumnTypeName=  Display table's column type
```

Read : Jdbc tutorial - ResultSetMetaData in java - Retrieve table column name and datatype

JDBC interview Question 23. What is DatabaseMetaData in JDBC java?

Answer. Another important JDBC interview question for experienced java developers. java.sql.DatabaseMetaData is an interface.

- java.sql.DatabaseMetaData can be used to **obtain information about the database as a whole.**
- java.sql.DatabaseMetaData extends java.sql.Wrapper.
- DatabaseMetaData important methods -
  - ***getDriverName()*** **-** Returns driver name.
  - ***getDriverVersion()*** **-** returns driver version.
  - ***getDatabaseProductName()*** **-** returns database name
  - ***getDatabaseProductVersion()*** **-** returns database version
  - ***getUserName()*** **-** returns username used to connect to database.
  - ***getURL()*** **-** returns URL used to connect to database.
  - ***getDatabaseMinorVersion()*** **-** returns database's minor/initial version.

***getDatabaseMajorVersion()* -** returns current database version.

Read : JDBC tutorial - DatabaseMetaData in java - retrieve database information

JDBC interview Question 24. How to do pagination in JDBC java OR How can you read all the records fr database table with huge records?

Answer.

Sql query used to perform pagination>

```
select emp.id, emp.name from
  ( select rownum rn, e.* from EMPLOYEE e) emp
        where rn >=? and rn< =? ;
```

Replace both occurrence of ? with range from which to which you want to fetch data.

Example >

```
select emp.id, emp.name from
  ( select rownum rn, e.* from EMPLOYEE e) emp
        where rn >=6 and rn< =10 ;
```

Above query will fetch rows between 6-10 (including) from Oracle database.

Pagination in ORACLE -

Alternatively you can use following query to fetch rows between 6-10 (including) i Oracle database

```
(select * from EMPLOYEE
where ROWNUM < 11 )
MINUS
(select * from EMPLOYEE
where ROWNUM < 6);

OR,

select emp.id, emp.name from ( select rownum rn, e.* from EMPLOYEE e) emp
 where rn >=6 and rn< =10 ;
```

Pagination in MySql -

Fetch rows between 6-10 (including) in MySql database

```
select * from my_schema.employee limit 5  OFFSET 5;
```

Read : JDBC pagination example ; How can we Read/Fetch all records from database tables with huge records in java

JDBC interview Question 25. How to fetch/retrieve top n records from table in java?

Answer. This JDBC interview question touches few JDBC basics and developers ability to query database **`preparedStatement.setMaxRo` method** sets the limit for the **maximum number of rows** that any **ResultSet object generated by this Statement object can contain** to the **specified** number. If in case limit is exceeded then excess number of rows are dropped without intimating user.

This can also be done by using rownum in **oracle** database query >

```
select * from EMPLOYEE where ROWNUM<3;
```

This query will fetch top 2 rows from oracle database.

Fetch top 2 rows in **MySql** database >

```
select * from my_schema.employee limit 2;
select * from my_schema.employee limit 2  OFFSET 0;
```

Read : Jdbc tutorial ; Fetch/retrieve top n records from table in java

JDBC interview Question 26. How to Insert date in JDBC java?

Answer.

```
In database >
create table EMPLOYEE(CREATION_DATE DATE);
```

For inserting date in Jdbc use -

```
new java.sql.Date(new java.util.Date().getTime())
```

Example -

```
prepStmt.setDate(1, new java.sql.Date(new java.util.Date().getTime()));
```

Read : Insert date in Jdbc program

JDBC interview Question 27. How to Insert timestamp in JDBC java?

Answer. Read : How to Insert timestamp in java Jdbc example

JDBC interview Question 28. What is RowSet in JDBC java JDBC? What are connected and disconnecte implementations of RowSet in java?

Answer. javax.sql.RowSet is an interface which can be used as a >

- **JavaBeans component** in a visual Bean development environment,
- It can be created and configured at design time and can be executed at run time in JDBC java.

RowSet interface **extends** ResultSet interface in JDBC java.

Following **classes implements RowSet** interface in java>

Connected implementations of RowSet (**connected rowset** means it continues to maintain its connection with database after retrieval of data as well)

- javax.sql.rowset.**JdbcRowSet**

Disconnected implementations of RowSet in JDBC java

- javax.sql.rowset.**CachedRowSet**
- javax.sql.rowset.**WebRowSet**
- javax.sql.rowset.**FilteredRowSet**
- javax.sql.rowset.**JoinRowSet**

More about **Connected** implementations of Rowset in JDBC java >
JdbcRowSet extends RowSet

More about **Disconnected** implementations of Rowset in JDBC java >
CachedRowSet extends RowSet
WebRowSet extends CachedRowSet
FilteredRowSet and JoinRowSet extends WebRowSet

JDBC interview Question 29. What is Connected implementations of RowSet in JDBC java ??

Answer. connected rowset means it continues to maintain its connection with database after retrieval of data as well.

JDBC interview Question 30. What is Disconnected implementations of RowSet in JDBC java ?

Answer. Disconnected rowset means it doesn't continues to maintain its connection with database after retrieval of data as well. It disconnects after retrieval of d

JDBC interview Question 31. What is Difference between ResultSet and RowSet java in JDBC java?

Answer. javax.sql.RowSet is an interface which can be used as a **JavaBeans component** in a visual Bean development environment, It can created and configured at design time and can be executed at runtime in java.

*Read this post for details on ResultSet: **ResultSet in java - Types, concurrency, holdability of ResultSet in java***

javax.sql.rowset.RowSet is a **wrapper** around a ResultSet which makes it possible to use the result set as a JavaBeans component in jdbc java

ResultSet alone cannot be used as a JavaBeans component.

RowSet interface **extends** ResultSet interface in java.

javax.sql.rowset.JdbcRowSet the subclass of RowSet is a **wrapper** around a ResultSet which makes it possible to use the result set as a JavaB component.

Read : **Difference between ResultSet and RowSet java**

JDBC interview Question 32. What is Connection Pooling in JDBC java?

Answer. This is very important JDBC interview question and experienced java developers must be well versed with this. **Connection poolin** the process where we maintain cache of database **connections.**

Why we need Connection Pooling? or why to use Connection Pooling in java?
Database Connections maintained in cache can be reused whenever request comes to connect with database. So, Connection Pooling reduces databse hits and improves application performance significantly. You must remember that database hit is a very costly operation and as much a possible you must try to avoid it.

Let's learn how to do Connection Pooling in java jdbc with example java program.

We will learn how to create ConnectionPool class in java, Class will consist of following Constructor and methods -

- **ConnectionPool** Constructor - Register and initialize database driver.
- **getConnection** method - Method to get connection from ConnectionPool class in java jdbc.
- **free** method - Method to free\release the connection back to ConnectionPool class in java jdbc.
- **totalConnections** method - Method to get total number of connections in ConnectionPool class in java jdbc.
- **closeAllConnections** method - Method to close all the connections in ConnectionPool class in java jdbc.

Read : connection-pooling-in-java-with-example

JDBC interview Question 33. What are Java JDBC best practices?

Answer. Here comes interviewers favourite JDBC interview questions. It's very very important interview question. I'll be answering this quest detail.

1. It is very common JDBC best practice to use Connection pooling in java. C**onnection pooling** is the process where we maintain cache of database **connections.**

Why we need Connection Pooling? or why to use Connection Pooling?

Database Connections maintained in cache can be reused whenever request comes to connect with database. So, Connection Pooling reduces databse hits and improves application performance significantly. You must remember that database hit is a very costly operation and as much a possible you must try to avoid it.

2. It is very important JDBC best practice. **PreparedStatement** is used for **executing a precompiled SQL statement..** java.sql.**PreparedStat** **suitable for executing DML commands** - SELECT, INSERT, UPDATE and DELETE. PreparedStatement is **faster** as compared to Statement be it is used for executing precompiled SQL statement. Hence, same SQL query **can** be **executed repeatedly in** PreparedStatement**.**

3. But, Statement is used for **executing a static SQL statement**. java.sql.Statement is **suitable for executing DDL commands** - CREATE, d alter and truncate.

4. You must follow this JDBC best practice. Avoid **SQL injection in JDBC java**

PreparedStatement **prevents SQL injection**, because text for all the parameter values is escaped. Example >

```
prepStmt = con.prepareStatement("select * from EMPLOYEE where ID=? ");
prepStmt.setInt(1, 8);
```

While, Statement **enforces SQL injection**, because we end up using query formed using **concatenated SQL strings**. Example >

```
String s1= "select * from EMPLOYEE where id = ";
int i1 = 2 ;
stmt.executeQuery(s1 + String.valueOf(i1));
```

Here comes one very **important** question, are **PreparedStatement vulnerable to SQL injections?**
YES, when we use **concatenated SQL strings** rather than using input as a parameter for preparedStatement.

PreparedStatement **makes code more** readable and understandable - We **need not to write concatenated SQL strings**, we can use queries pass **different parameters at runtime using** setter methods.

5. It is another very important JDBC best practice. **Using batch statements in jdbc java -**

Batch statement sends **multiple requests from java to database in one just one call** while **without batch statements multiple requests wi in sent in multiple (one by one) calls to the database.**

About **addBatch()** method >

PreparedStatement extends Statement and inherits all methods from Statement and additionally adds **addBatch()** method.

addBatch() method - adds a set of parameters to the PreparedStatement object's batch of commands.

For details on using batch statements in jdbc java : *PreparedStatement BATCH - using executeUpdate methods* - INSERT, UPDATE *and* DELETE

6. Jdbc best practice of **specifying column name in select query -** Rather than using queries like `"select * from EMPLOYEE",` you must specify column name which you want to fetch from database like this `"select ID, NAME from EMPLOYEE".`
But, what's the advantage of specifying column name rather than using select * from table.

Assume your table has 100 columns and you have to use only 2 column, then you will unnecessarily fetch data of other columns which in turn w degrade your application's performance and also you will end up wasting precious memory.

7. This JDBC best practice is in **continuation to above discussed point.** Rather than **specifying column index we must use the column** to avoid **java.sql.SQLException:** **Invalid column index and Invalid column name Exceptions**.

Example - Let's say out SQL query is `"select ID, NAME from EMPLOYEE"`

Than rather than specifying column index

```
resultSet.getInt(1) and resultSet.getString(2));
```

We must specify column name

```
resultSet.getInt("ID") and resultSet.getString("NAME"));
```

8. It is good JDBC practice to write as much business logic as much as possible in **Stored Procedure or Functions** as **compared to writing down in java class.**
Because that reduces databse hits and improves application performance significantly. You must remember that database hit is a very costly op and you must try to avoid it as much as possible.

Read : Execute database STORED PROCEDURE - IN parameter, OUT parameter *and* IN OUT parameter *|| call* FUNCTION *from java - >*

9. **JDBC Transactions -** Another very important JDBC best practice of using connection.setAutoCommit(false), **`connection.commit` and `connection.rollback()`.**

We can set autocommit mode of connection to false using connection.setAutoCommit(false) and then accordingly use **`connection.commit()`** **`connection.rollback()`.**

If any transaction fails in between then rollback the transaction by calling **`con.rollback()`**, commit the transaction by using **`con.commit()`** only went successful.

Example -

Let's say we have to update salary of two employees, and salary of both employees must be updated simultaneously in database.

And let's say salary of first employee is updated successfully. But, if anything goes wrong in updating salary of second employee than any upda done to first employee's salary will be rolled back.

For more read : JDBC Transactions - setting connection.setAutoCommit(false), commit and rollback(TCL command) in java

10. Here is another JDBC best practice - You must ensure that you close all the JDBC ***Statement***, ***PreparedStatement***, ***CallableStatemen*** ResultSet and Connections in java to **avoid ora-01000 maximum open cursors exceeded java.sql.SQLException** in java. You must always all the above mentioned objects in finally block in java because finally block is **always executed** irrespective of exception is thrown or not by ja code.
Example of closing ***PreparedStatement***, ResultSet and Connections in finally block in java-

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
```

```
/** Copyright (c), AnkitMittal JavaMadeSoEasy.com */
public class JdbcBestPracticeExampleInJava {
    public static void main(String... arg) {
          Connection con = null;
          PreparedStatement prepStmt = null;
          ResultSet rs = null;
          try {
                // Logic ..

          } catch (ClassNotFoundException e) {
                e.printStackTrace();
          } catch (SQLException e) {
                e.printStackTrace();
          }
          finally{
                 try {
                       if(rs!=null) rs.close(); //close resultSet
                       if(prepStmt!=null) prepStmt.close(); //close PreparedStatement
                       if(con!=null) con.close(); // close connection
                 } catch (SQLException e) {
                       e.printStackTrace();
                 }
           }
    }
}
```

For more details please read : Java JDBC best practices

JDBC interview Question 34. What are important and frequently occurring SQLException in JDBC java?

Answer. This is another very important JDBC interview question.

java.sql.SQLException: Exhausted Resultset

It is thrown when cursor does not point to any row in ResultSet's object in JDBC java.

java.sql.SQLException: Exhausted Resultset

It is thrown when cursor does not point point to any row in ResultSet's object in JDBC java.
is thrown when rs.getString("COLUMN_NAME") is called before calling rs.next() on ResultSet object.

```
rs = prepStmt.executeQuery();
//At this point cursor is before the first position of ResultSet object
System.out.println(rs.getString("COLUMN_NAME"));//java.sql.SQLException: Exhausted Resultset
```

ResultSet type TYPE_FORWARD_ONLY does not allow first(), last(), previous(), absolute or relative()

ResultSet type = ResultSet.**TYPE_FORWARD_ONLY**
ResultSet concurrency = ResultSet.**CONCUR_READ_ONLY** or
ResultSet.**CONCUR_UPDATABLE**

java.sql.SQLException: Invalid operation for forward only resultset : first

```
prepStmt = con.prepareStatement("select * from EMPLOYEE",
            ResultSet.TYPE_FORWARD_ONLY, ResultSet.CONCUR_UPDATABLE);
rs = prepStmt.executeQuery();
rs.last(); //java.sql.SQLException: Invalid operation for forward only resultset : last
```

java.sql.SQLException: Invalid operation for forward only resultset : last

```
prepStmt = con.prepareStatement("select * from EMPLOYEE",
            ResultSet.TYPE_FORWARD_ONLY, ResultSet.CONCUR_UPDATABLE);
rs = prepStmt.executeQuery();
rs.last(); //java.sql.SQLException: Invalid operation for forward only resultset : last
```

java.sql.SQLException: Invalid operation for forward only resultset : previous

```
prepStmt = con.prepareStatement("select * from EMPLOYEE",
              ResultSet.TYPE_FORWARD_ONLY, ResultSet.CONCUR_UPDATABLE);
rs = prepStmt.executeQuery();
rs.next();
   rs.previous(); //java.sql.SQLException: Invalid operation for forward only resultset : last
```

java.sql.SQLException: Invalid operation for forward only resultset : absolute

```
prepStmt = con.prepareStatement("select * from EMPLOYEE",
              ResultSet.TYPE_FORWARD_ONLY, ResultSet.CONCUR_UPDATABLE);
rs = prepStmt.executeQuery();
rs.absolute(1); //java.sql.SQLException: Invalid operation for forward only resultset : absolute
```

java.sql.SQLException: Invalid operation for forward only resultset : relative

```
prepStmt = con.prepareStatement("select * from EMPLOYEE",
              ResultSet.TYPE_FORWARD_ONLY, ResultSet.CONCUR_UPDATABLE);
rs = prepStmt.executeQuery();
rs.absolute(1); //java.sql.SQLException: Invalid operation for forward only resultset : relative
```

ResultSet concurrency mode CONCUR_READ_ONLY does not allow first(), last(), previous(), absolute or relative() in JDBC java

ResultSet type = ResultSet.**TYPE_FORWARD_ONLY** or
ResultSet.**TYPE_SCROLL_INSENSITIVE** or
ResultSet.**TYPE_SCROLL_SENSITIVE**
ResultSet concurrency = ResultSet.**CONCUR_READ_ONLY**

java.sql.SQLException: Invalid operation for read only resultset: updateString

when ResultSet mode is ***CONCUR_READ_ONLY*** it cannot be update.

```
    prepStmt = con.prepareStatement("select NAME from EMPLOYEE",
              ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);

rs = prepStmt.executeQuery();

rs.next(); //move cursor to first row in ResultSet object

//Print data in first row
System.out.println(rs.getString("NAME")); //will print data in first row

rs.updateString("NAME", "ankitUpdated");//java.sql.SQLException: Invalid
                  operation for read only resultset: updateString
```

java.sql.SQLException: Invalid operation for read only resultset: deleteRow

when ResultSet mode is ***CONCUR_READ_ONLY*** row cannot be deleted from it.

```
    prepStmt = con.prepareStatement("select NAME from EMPLOYEE",
              ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);

  rs = prepStmt.executeQuery();

  rs.next(); //move cursor to first row in ResultSet object

  //Print data in first row
  System.out.println(rs.getString("NAME")); //will print data in first row

  rs.deleteRow();//java.sql.SQLException: Invalid operation for read only resultset: deleteRow
```

java.sql.SQLException: Invalid operation for read only resultset: moveToInsertRow

when ResultSet mode is ***CONCUR_READ_ONLY*** row cannot be inserted into it.

```
prepStmt = con.prepareStatement("select NAME from EMPLOYEE",
              ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);

  rs = prepStmt.executeQuery();

  rs.moveToInsertRow(); //java.sql.SQLException: Invalid operation for read only
                        resultset: moveToInsertRow
  rs.updateString("NAME", "ankitNew");
 rs.insertRow();
```

JDBC interview Question 35. What is Invalid column name/ index in JDBC java?

Answer. java.sql.SQLException: Invalid column name/ index in JDBC

JDBC interview Question 36. What is Solve ora-01000 maximum open cursors exceeded in JDBC java?

Answer. Experienced java developers must be able to provide solution to this JDBC interview question. Please read
Solve ora-01000 maximum open cursors exceeded java.sql.SQLException

What causes the Solve **ora-01000 maximum open cursors exceeded java.sql.SQLException problem** in JD
java-

- Not closing the JDBC ***Statement*** ***object can cause*** maximum open cursors exceeded java.sql.SQLException,
- Not closing the JDBC ***PreparedStatement*** ***object can cause*** maximum open cursors exceeded java.sql.SQLException,
- Not closing the JDBC ***CallableStatement*** ***object can cause*** maximum open cursors exceeded java.sql.SQLException,
- Not closing the JDBC ResultSet ***object*** and
- Not closing the JDBC **Connections** object ***can cause*** maximum open cursors exceeded java.sql.SQLException

Solution to **ora-01000** maximum open cursors exceeded java.sql.SQLException problem in JDBC java-

You must ensure that you close all the JDBC ***Statement***, ***PreparedStatement***, ***CallableStatement*** , ResultSet and Connections in java to **avo
ora-01000 maximum open cursors exceeded java.sql.SQLException** in java. You must always close all the above mentioned objects in fina
block in java because finally block is **always executed** irrespective of exception is thrown or not by java code.
Example of closing ***PreparedStatement***, ResultSet and Connections in finally block in java-

JDBC interview Question 37. How to solve java.sql.SQLException ORA-12560: TNS:protocol adapter erro occurs in JDBC java?

Answer.
This problem occurs when your database is not up. By default Oracle services start at windows startup, but they might have been stopped man
by user.

Let's resolve this problem by starting the database server in 2 ways -

Solution 1-

In windows, Go to start, **run** (Windows + r), and type **Services.msc**

And start all the below services manually -
OracleMTSRecoveryService
OracleOraDb11g_home1ClrAgent
OracleOraDb11g_home1TNSListener
OracleServiceORCL

Now, your database is up, and will resolve your problem.

```
Enter user-name: system
Enter password:

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 -
Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
```

Solution 2-

We will start Oracle services through CMD.

```
C:\Users\ankitmittal01>Set oracle_sid=ORCL

C:\Users\ankitmittal01>net start oracleserviceORCL
The OracleServiceORCL service is starting...
The OracleServiceORCL service was started successfully.



Enter user-name: system
Enter password:

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 -
Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
```

Set Oracle_Sid

- **C:>set oracle_sid=ORCL**

Then, type **net start** command.

- **C:>net start oracleserviceORCL**

JDBC interview Question 38. How to execute database functions in JDBC java?

Answer. It is another very important JDBC interview question.

Please Read : Use CallableStatement to execute *Execute database call* FUNCTION in jdbc java.

JDBC interview Question 39. What is executing BATCH statements in JDBC java?

Answer.

Using batch statements in jdbc java -

Batch statement sends **multiple requests from java to database in one just one call** while **without batch statements multiple requests wi in sent in multiple (one by one) calls to the database.**

We can use **PreparedStatement's** addBatch() and executeBatch() methods for **executing BATCH statements** in java

Read : JDBC program- Batch PreparedStatement example- Execute INSERT query

JDBC - Batch PreparedStatement example- Execute UPDATE query

JDBC program- Batch PreparedStatement example- Execute DELETE query

JDBC interview Question 40. Tell me what you know about JDBC transactions - savepoint, rollback meth in JDBC java?

Answer. Very important JDBC interview questions and it is very highly used practice in critical and complex applications

Savepoint methods in java jdbc>

setSavepoint method creates a savepoint with the no name in the current transaction in java.

setSavepoint(savepointName) method creates a savepoint with the specified **savepointName** in the current transaction in java JDBC.

Rollback methods in java jdbc>

rollback() method undoes all the transactions performed in the current transaction in java jdbc.

rollback(**savepointName**) method undoes all the transactions performed after specified **savepointName** in the current transaction in java.

Delete/remove Savepoint methods in java jdbc>

releaseSavepoint(savepointName) method deletes specified **savepointName** and all the subsequent savepoints from the current transaction java.

JDBC Transactions : Programs to create **Savepoint, rollback** and commit in JDBC java

```
Connection connection = null;
    PreparedStatement prepStmt = null;

     // . . .
     // . . .
     connection.setAutoCommit(false);



     //SAVEPOINT 1
     Savepoint savepoint1 = connection.setSavepoint();
     System.out.println("\nSavepoint1 created");

     prepStmt = connection.prepareStatement("DELETE from EMPLOYEE where ID=? ");
     prepStmt.setInt(1, 7); //substitute first occurrence of ? with 7

     prepStmt.executeUpdate(); // execute delete query
     System.out.println("deleted");

     connection.rollback(savepoint1); //Rollback to savepoint1
     System.out.println("Rolled back to savepoint1");



     //SAVEPOINT 2
```