Unit -5

The Concept of JDBC:

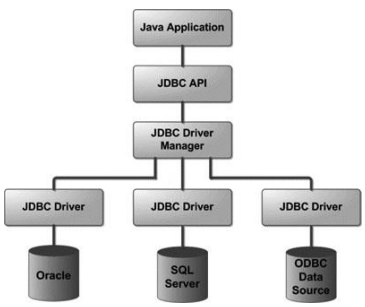
1. Java was not considered industrial strength programming language since java was unable to access the DBMS.
2. Each dbms has its own way to access the data storage. low level code required to access oracle data storage need to be rewritten to access db2.

3. JDBC stands for **J**ava **D**ata**b**ase **C**onnectivity, which is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases

4. JDBC drivers has to do the following

- Open connection between DBMS and J2EE environment.
- Translate low level equivalents of sql statements sent by J2EE component into messages that can be processed by the DBMS.
- Return the data that conforms to JDBC specifications to the JDBC driver
- Return the error messages that conforms to JDBC specifications to the JDBC driver
- Provides transaction management routines that conforms to JDBC specifications to the JDBC driver
- Close connection between the DBMS and the J2EE component.

JDBC Architecture



June 2012 (Briefly discuss the various JDBC driver types 10 M)

JDBC Driver Types

Type 1 driver JDBC to ODBC Driver

1. It is also called JDBC/ODBC Bridge , developed by MicroSoft.
2. It receives messages from a J2EE component that conforms to the JDBC specifications
3. Then it translates into the messages understood by the DBMS.
4. This is DBMS independent database program that is ODBC open database connectivity.

Type 2 JAVA / Native Code Driver

1. Generates platform specific code that is code understood by platform specific code only understood by specific databases.
2. Manufacturer of DBMS provides both java/ Native code driver.
3. Using this provides lost of portability of code.
4. It won't work for another DBMS manufacturer

Type 3 JDBC Driver

1. Most commonly used JDBC driver.
2. Coverts SQL queries into JDBC Formatted statements.
3. Then JDBC Formatted statements are translated into the format required by the DBMS.
4. Referred as Java protocol

Type 4 JDBC Driver

1. Referred as Type 4 database protocol
2. SQL statements are transferred into the format required by the DBMS.
3. This is the fastest communication protocol.

JDBC Packages

JDBC API contains two packages. First package is called java.sql, second package is called javax.sql which extends java.sql for advanced JDBC features.

Explain the various steps of the JDBC process with code snippets.

1. Loading the JDBC driver

- The jdbc driver must be loaded before the J2EE compnet can be connected to the database.
- Driver is loaded by calling the method and passing it the name of driver

Class.forName("sun:jdbc.odbc.JdbcOdbcDriver");

2. Connecting to the DBMS.

- Once the driver is loaded , J2EE component must connect to the DBMS using DriverManager.getConnection() method.
- It is highest class in hierarchy and is responsible for managing driver information.

- It takes three arguments URL, User, Password
- It returns connection interface that is used through out the process to reference a database

```
String url=”jdbc:odbc:JdbcOdbcDriver”;
String userId=”jim”
String password=”Keogh”;
Statement DatRequest;
Private Connection db;

try{
Class.forName(“sun:jdbc.odbc.JdbcOdbcDriver”);
Db=DriverManager.getConnection(url,userId,password);
}
```

3. Creating and Executing a statement.
   - The next step after the JDBC is loaded and connection is successfully made with a particular database managed by the dbms, is to end a particular query to the DBMS for processing.
   - SQL query consists series of SQL command that direct DBMS to do something example Return rows.
   - Connect.createStatement() method is used to create a statement Object.
   - The statement object is then used to execute a query and return result object that contain response from the DBMS

```
Statement DataRequest;
ResultSet Results;
try {
        String query=“select * from Customers”;
        DataRequest=Database.createStatement();
        Results= DataRequests.executeQuery(query);
  }
```

4. Processing data returned by the DBMS
   - **java.sql.ResultSet** object is assigned the result received from the DBMS after the query is processed.
   - **java.sql.ResultSet** contain method to interct with data that is returned by the DBMS to the J2EE Component.

```
Results= DataRequests.executeQuery(query);
do
{
Fname=Results.getString(Fname)
}
While(Results.next())
```

In the above code it return result from the query and executes the query. And getString is used to process the String retrived from the database.

5. Terminating the connection with the DBMS.
   To terminate the connection Database.close() method is used.

With proper syntax, explain three types of getConnection() method.

1 After the JDBC driver is successfully loaded and registered, the J2EE component must connect to the database. The database must be associated with the JDBC driver.
2 The datasource that JDBC component will connect to is identified using the URL format. The URL consists of three format.
- These are jdbc which indicate jdbc protocol is used to read the URL.
- <subprotocol> which is JDBC driver name.
- <subname> which is the name of database.

3 Connection to the database is achieved by using one of three getConnection() methods. It returns connection object otherwise returns SQLException
4 Three getConnection() method
- getConnection(String url)
- getConnection(String url, String pass, String user)
- getConnection(String url, Properties prop)

5 getConnection(String url)

- Sometimes the DBMS grant access to a database to anyone that time J2EE component uses getConnection(url) method is used.

```
String url=" jdbc:odbc:JdbcOdbcDriver ";
try{
Class.forName("sun:jdbc.odbc.JdbcOdbcDriver");
Db=DriverManager.getConnection(url);
}
```

6 getConnection(String url, String pass, String user)

- Database has limited access to the database to authorized user and require J2EE to supply user id and password with request access to the database. The this method is used.

```
try{
Class.forName("sun:jdbc.odbc.JdbcOdbcDriver");
Db=DriverManager.getConnection(url,userId,password);
}
```

7 getConnection(String url, Properties prop)

- There might be occasions when a DBMS require information besides userid and password before DBMS grant access to the database.

- This additional information is called properties and that must be associated with Properties object.
- The property is stored in text file. And then loaded by load method of Properties class.

```
Connection db;
Properties props=new Properties();
try {
 FileInputStream inputfile=new FileInputStream("text.txt");
Prop.load(inputfile);
}
```

Write short notes on Timeout:

1. Competition to use the same database is a common occurrence in the J2EE environment and can lead to performance degradation of J2EE application
2. Database may not connect immediately delayed response because database may not available.
3. Rather than delayed waiting timeJ2EE component can stop connection. After some time. This time can bet set with the following method: **DriverManager.setLoginTimeout(int sec).**
4. **DriverManager.getLoginTimeout(int sec)** return the current timeout in seconds.

Explain Connection Pool

1. Client needs frequent that needs to frequently interact with database must either open connection and leave open connection during processing or open or close and reconnect each time.
2. Leaving the connection may open might prevent another client from accessing the database when DBS have limited no of connections. Connecting and reconnecting is time consuming.
3. The release of JDBC 2.1 Standard extension API introduced concept on connection pooling
4. A connection pool is a collection of database connection that are opened and loaded into memory so these connection can be reused with out reconnecting to the database.
5. DataSource interface to connect to the connection pool. connection pool is implemented in application server.
6. There are two types of connection to the database 1.) Logical 2.) Physical
7. The following is code to connect to connection pool.

```
Context ctext= new IntialContext()
DataSource pool =(DataSource) ctext.lookup("java:comp/env/jdbc/pool");
Connection db=pool.getConnection();
```

Briefly explain the Statement object. Write program to call a stored procedure.(10)

1. Statement object executes query immediately with out precompiling.
2. The statement object contains the **excuteQuery()** method , which accept query as argument then query is transmitted for processing.It returns ResultSet as object.

Example Program

```
String url="jdbc:odbc:JdbcOdbcDriver";
String userId="jim"
String password="Keogh";
Statement DatRequest;
Private Connection db;
ResultSet rs;

// code to load driver
//code to connect to the database
 try{
String query="SELECT * FROM Customers;
DatRequest=Db.createStaement();
rs=DatRequest.executeQuery(query);// return result set object
}catch(SQLException  err)
{
System.err.println("Error");
System.exit(1);
}
```


3. Another method is used when DML and DDL operations are used for processing query is executeUpdate(). This returns no of rows as integer.

```
try{
String query="UPDATE Customer set PAID='Y' where BALANCE ='0';
DatRequest=Db.createStaement();
int  n=DatRequest.executeUpdate(query);// returns no of rows updated
}catch(SQLException  err)
{
System.err.println("Error");
System.exit(1);
}
```

Briefly explain the prepared statement object. Write program to call a stored procedure.(10)

1. A SQL query must be compiled before DBMS processes the query. Query is precompiled and executed using Prepared statements.
2. Question mark is placed as the value of the customer number. The value will be inserted into the precompiled query later in the code.

3. Setxxx() is used to replace the question mark with the value passed to the setxxx() method . xxx represents data type of the field.
   Example if it is string then setString() is used.
4. It takes two arguments on is position of question mark and other is value to the filed.
5. This is referred as late binding.

```
String url=”jdbc:odbc:JdbcOdbcDriver”;
String userId=”jim”
String password=”Keogh”;
ResultSet rs;

// code to load driver
//code to connect to the database
 try{

String query=”SELECT * FROM Customers where cno=?”;
PreparedStatement   pstatement=db.preparedStatement(query);
pstatement.setString( 1,”123”); // 1 represents first place holder, 123 is value
rs= pstatement.executeQuery();


}catch(SQLException  err)
{
System.err.println(“Error”);
System.exit(1);
}
```


Briefly explain the callable statement object. Write program to call a stored procedure.(10)

1. The callableStatement object is used to call a stored procedure from with in J2EE object. A stored procedure is block of code and is identified by unique name. the code can bewritten in Transact-C ,PL/SQL.

2. Stored procedure is executed by invoking by the name of procedure.

3. The callableStatement uses three types of parameter when calling stored procedure. The parameters are IN ,OUT,INOUT.

4. IN parameter contains data that needs to be passed to the stored procedure whose value is assigned using setxxx() method.

5. OUT parameter contains value returned by stored procedure.the OUT parameter should be registers by using registerOutParameter() method and then later retrieved by the J2EE component using getxxx() method.

6. INOUT parameter is used to both pass information to the stored procedure and retrieve the information from the procedure.

7. Suppose, you need to execute the following Oracle stored procedure:

```
CREATE OR REPLACE PROCEDURE getEmpName
        (EMP_ID IN NUMBER, EMP_FIRST OUT VARCHAR) AS
BEGIN
 SELECT first INTO EMP_FIRST    FROM Employees   WHERE ID = EMP_ID;
END;
```

8. The following code snippets is used

```
CallableStatement cstmt = null;
try {     String SQL = "{call getEmpName (?, ?)}";
          cstmt = conn.prepareCall (SQL);catch (SQLException e) { }
```


9. Using CallableStatement objects is much like using PreparedStatement objects. You must bind values to all parameters before executing the statement, or you will receive an SQLException.
10. If you have IN parameters, just follow the same rules and techniques that apply to a PreparedStatement object; use the setXXX() method that corresponds to the Java data type you are binding.
11. When you use OUT and INOUT parameters you must employ an additional CallableStatement method, registerOutParameter(). The registerOutParameter() method binds the JDBC data type to the data type the stored procedure is expected to return.
12. Once you call your stored procedure, you retrieve the value from the OUT parameter with the appropriate getXXX() method. This method casts the retrieved value of SQL type to a Java data type.

ResultSet

1. ResultSet object contain the methods that are used to copy data from ResultSet into java collection object or variable for further processing.

2. Data in the ResultSet is logically organized into the virtual table for further processing. Result set along with row and column it also contains meta data.

3. ResultSet uses virtual cursor to point to a row of the table.

4. J2EE component should use the virtual cursor to each row and the use other methods of the ResultSet to object to interact with the data stored in column of the row.

5. The virtual cursor is positioned above the first row of data when the ResultSet is returned by executeQuery () method.

6. The virtual cursor is moved to the first row with help of next() method of ResultSet

7. Once virtual cursor is positioned getxxx() is used to return the data. Data type of data is represents by xxx. It should match with column data type.

8. getString(fname) ….fname is column name.

9. setString(1)…….. in this 1 indicates first column selected by query.


```
stmt = conn.createStatement();
String sql;
sql = "SELECT id, first, last, age FROM Employees";
ResultSet rs = stmt.executeQuery(sql);
while(rs.next()){
   int id  = rs.getInt("id");/ / rs.getInt(1);
   int age = rs.getInt("age");
   String first = rs.getString("first");
   String last = rs.getString("last");
   System.out.print("ID: " + id);
   System.out.print(", Age: " + age);
```

```
            System.out.print(", First: " + first);
            System.out.println(", Last: " + last);
        }
```

Explain the with an example Scrollable Result Set (6 Marks)

1. Until the release of JDBC 2.1 API , the virtual cursor can move only in forward directions. But today the virtual cursor can be positioned at a specific row.
2. There are six methods to position the cursor at specific location in addition to next() in scrollable result set. firs() ,last(), absolute(), relative(), previous(), and getRow().
3. first() …… position at first row.
4. last()……position at last row.
5. previous()…….position at previous row.
6. absolute()……. To the row specified in the absolute function
7. relative()…… move relative to current row. Positive and negative no can be given.
   Ex. relative(-4) … 4 position backward direction.
8. getRow() …… returns the no of current row.
9. There are three constants can be passed to the createStatement()
10. Default is TYPE_FORWARD_ONLY. Otherwise three constant can be passed to the create statement 1.) TYPE_SCROLL_INSENSITIVE

    2.) TYPE_SCROLL_SENSITIVE

11. TYPE_SCROLL makes cursor to move both direction. INSENSITIVE makes changes made by J2EE component will not reflect. SENSITIVE means changes by J2EE will reflect in the result set.
    Example code.

```
String sql=" select * from emp";
DR=Db.createStatement(TYPE_SCROLL_INSENSITIVE);
RS= DR.executeQuery(sql);
```

12. Now we can use all the methods of ResultSet.

Explain the with an example updatable Result Set.

1. Rows contained in the result set is updatable similar to how rows in the table can be updated. This is possible by sending CONCUR_UPDATABLE.
2. There are three ways in which result set can be changed. These are updating row , deleting a row, inserting a new row.
3. **Update ResultSet**

- Once the executeQuery() method of the statement object returns a result set. updatexxx() method is used to change the value of column in the current row of result set.
- It requires two parameters, position of the column in query. Second parameter is value
- updateRow() method is called after all the updatexxx() methods are called.

Example:

```
try{
        String query= "select Fname, Lname from Customers
        where Fname= 'Mary' and Lanme='Smith';
        DataRequest= Db.
        createStatement(ResultSet.CONCUR_UPDATABLE);
        Rs= DataRequest.executeQuery(query);
        Rs.updateString("LastName","Smith");
        Rs.updateRow();
 }
```

4. **Delete row in result set**
   - By using absolute method positioning the virtual cursor and calling deleteRow(int n) n is the number of rows to be deleted.
   - Rs.deleteRow(0) current row is deleted.
5. **Insert Row in result set**
   - Once the executeQuery() method of the statement object returns a result set. updatexxx() method is used to insert the new row of result set.
   - It requires two parameters, position of the column in query. Second parameter is value
   - insertRow() method is called after all the updatexxx() methods are called.

```
try{
        String query= "select Fname, Lname from Customers
        where Fname= 'Mary' and Lanme='Smith';
        DataRequest= Db.
        createStatement(ResultSet.CONCUR_UPDATABLE);
        Rs= DataRequest.executeQuery(query);
        Rs.updateString(1 ,"Jon");
        Rs.updateString(2 ,"Smith");
        Rs.insertRow();
 }
```

6. Whatever the changes making will affect only in the result set not in the table. To update in the table have to execute the DML(update, insert, delete) statements.

Explain the Transaction processing with example

1. A transaction may consists of a set of SQL statements, each of which must be successfully completed for the transaction to be completed. If one fails SQL statements successfully completed must be rolled back.
2. Transaction is not completed until the J2EE component calls the commit() method of the connection object. All SQL statements executed prior to the call to commit() method can be rolled back.
3. Commit() method was automatically called in the program. DBMS has set AutoCommit feature.
4. If the J2EE component is processing a transaction then it has to deactivate the auto commit() option false.

```
try {
        DataBase.setAutoCommit(false)
        String query="UPDATE Customer set Street ='5  main Street' "+
              "WHERE FirstName ='Bob' ";
        DR= DataBase.createStatement();
        DataRequest=DataBase.createStatement();
DataRequest.executeUpdate(query1);
DataBase.commit();
}
```

5. Transaction can also be rolled back. When not happened. Db.rollback().
6. A transaction may consists of many tasks , some of which no need to roll back . in such situation we can create a savepoints, in between transactions. It was introduced in JDBC 3.0. save points are created and then passed as parameters to rollback() methods.
7. releseSavepint() is used to remove the savepoint from the transaction.
8. Savepoint s1=DataBase.setSavePoint("sp1");to create the savepoint.
9. Database.rollback(sp1); to rollback the transaction.

Batch Execution of transaction

10. Another way to combine sql statements into a single into a single transaction and then execute the entire transaction .
11. To do this the addBatch() method of statement object. The addBatch() method receives a SQL statement as a parameter and places the SQL statement in the batch.
12. executeBatch() method is called to execute the entire batch at the same time. It returns an array that contains no of SQL statement that execute successfully.

```
String query1="UPDATE Customers SET street =' 5 th Main'" +
        "Where Fname='BoB' ";
String query2="UPDATE Customers SET street =' 10 th Main'" +
        "Where Fname='Tom' ";
Statement DR=DB.createStatement();
```

```
DR.addBatch(query1);
DR.addBatch(query2);
int [] updated= DR.executeBatch();
```

Write notes on metadata interface Metadata

1. Metadata is data about data. MetaData is accessed by using the DatabaseMetaData interface.
2. This interface is used to return the meta data information about database.
3. Meta data is retrieved by using getMetaData() method of connection object.
   Database metadata
   The method used to retrieve meta data informations are
4. getDatabaseProductNAme()…returns the product name of database.
5. getUserNAme() returns the usernamr()
6. getURL() returns the URL of the databse.
7. getSchemas() returns all the schema name
8. getPrimaryKey() returns primary key
9. getTables() returns names of tables in the database
   ResultSet Metadata
   ResultSetMetaData rm=Result.getMeatData()

   The method used to retrieve meta data information about result set are
10. getColumnCount() returns the number of columns contained in result set

Data types of Sql used in setXXX() and getXXX() methods.

SQL	JDBC/Java
VARCHAR	java.lang.String
CHAR	java.lang.String
LONGVARCHAR	java.lang.String
BIT	boolean
NUMERIC	java.math.BigDecimal
TINYINT	byte

SMALLINT	short
INTEGER	int
BIGINT	long
REAL	float
FLOAT	float
DOUBLE	double
VARBINARY	byte[ ]
BINARY	byte[ ]
DATE	java.sql.Date
TIME	java.sql.Time
TIMESTAMP	java.sql.Timestamp
CLOB	java.sql.Clob
BLOB	java.sql.Blob
ARRAY	java.sql.Array
REF	java.sql.Ref

Exceptions handling with jdbc

Exception handling allows you to handle exceptional conditions such as program-defined errors in a controlled fashion.

1. When an exception condition occurs, an exception is thrown. The term thrown means that current program execution stops, and control is redirected to the nearest applicable catch clause. If no applicable catch clause exists, then the program's execution ends.

2. JDBC Exception handling is very similar to Java Excpetion handling but for JDBC.

3. There are three kind of exception thrown by jdbc methods.

4. SQLException ,SQLWarnings, DataTruncation

   SQLException

5. The most common exception you'll deal with is **java.sql.SQLException which result in SQL sysntax errors.**

6. getNextException() method returns details about the error.

7. getErrorCode() method retrieves vendor specific error codes.

   SQLWarnings

8. it throws warnings related to connection from DBMS. getWarnings() method of connecction object retrieves t **warnings.** getNextWarnings() **returns** subsequent warnings.

   DataTruncation

9. Whenever data is lost due to truncation of the data value , a truncation exception is thrown.

Differentiate between a Statement and a PreparedStatement.

• A standard Statement is used for creating a Java representation for a literal SQL statement and for executing it on the database.

• A PreparedStatement is a precompiled Statement.

• A Statement has to verify its metadata in the database every time.

• But ,the prepared statement has to verify its metadata in the database only once.

• If we execute the SQL statement, it will go to the STATEMENT.

• But, if we want to execute a single SQL statement for the multiple number of times, it'll go to the PreparedStatement.

Explain the classes, interface , methods available in java.sql.* package.

Java.sql.package include classes and interface to perform almost all JDBC operation such as creating and executing SQL queries

1. java.sql.BLOB --------provide support to BLOB SQL data type.
2. java.sql.Connection----- creates connection with specific data type
   Methods in Connection
   setSavePoint()
   rollback()
   commit()
   setAutoCommit()
3. java.sql.CallableStatement-------- Executes stored procedures
   Methods in CallableStatement
   execute()
   registerOutParameter()
4. java.sql.CLOB ------------ support for CLOB data type.
5. java.sql.Date------------ support for Date SQL type.
6. Java.sql.Driver -----create instance of driver with the DriverManager
7. java.sql.DriverManager---- manages the data base driver
   getConnection()
   setLoginTimeout()
   getLoginTimeout()
8. java.sql.PreparedStatement—create parameterized query
   executeQuery()
   executeUpdate()
9. java.sql.ResultSet------------ it is interface to access result row by row
   rs.next()
   rs.last()
   rs.first()
10. java.sql.Savepoint------- Specify savepoint in transaction.
11. java.sql.SQLException--------- Encapsulates JDBC related exception.
12. java.sql.Statement…….. interface used to execute SQL statement.
13. java.sql.DataBaseMetaData….. returns mata data

University Questions

1. Write a program to display current content of table in database.
2. Exceptions handling with jdbc program.
3. Write notes on metadata interface Metadata
4. Explain the with an example updatable Result Set.
5. Explain the Transaction processing with example
6. Explain the with an example updatable Result Set.
7. Explain the with an example Scrollable Result Set (6 Marks)
8. Explain the various steps of the JDBC process with code snippets.
9. Briefly explain the callable statement object. Write program to call a stored procedure
10. (Briefly discuss the various JDBC driver types 10 M)
11. With proper syntax, explain three types of getConnection() method.
12. Explain J2ee multitier architecture.
13. Explain the classes, interface available in java.sql.* package.