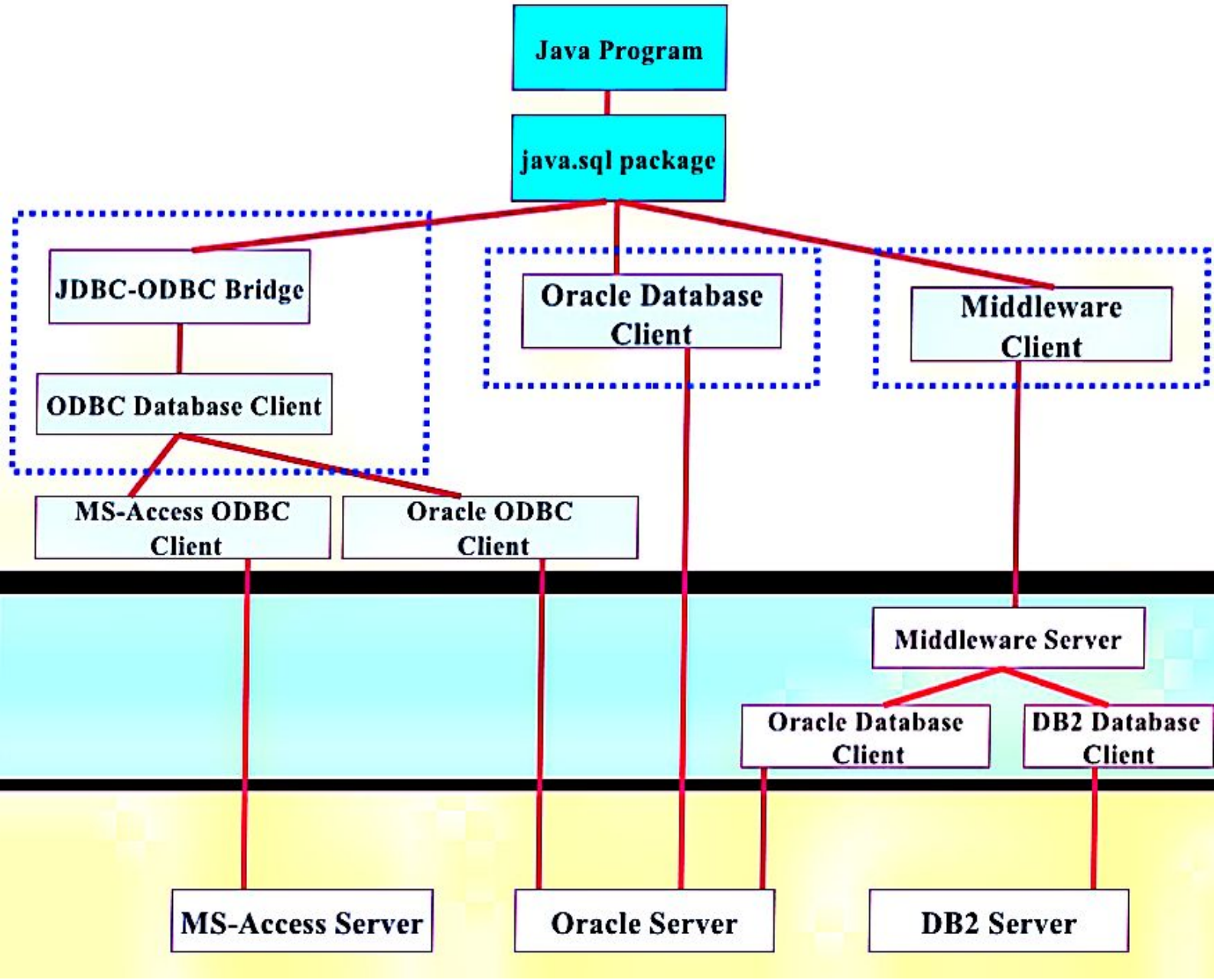
JDBC

- A Java API for executing SQL statements.
- Extends the Java language.
- Supported by the java.sql package

ODBC

- Open Database Connectivity developed by Microsoft to provide interaction with databases using SQL.
- Use the JDBC-ODBC bridge to make ODBC work with Java.



JDBC Components

Driver Manager - loads the database drivers, and manages the connection between the application and the driver.

Driver - translates API calls into operations for a specific data source.

Connection - A session between an application and a database.

Statement - An SQL statement to perform.

PreparedStatement - a precompiled and stored query.

ResultSet - logical set of data returned by executing a statement.

MetaData - Information about the returned data (resultset) including the attribute names and types.

Steps In Using JDBC

- **Import the necessary classes**

```
import java.sql.*;
```

- **Load the JDBC driver**

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

- **Identify the data source**

```
String sourceURL = "jdbc:odbc:testdb";
```

- **Allocate a Connection object**

```
Connection dbConnect =
 DriverManager.getConnection(sourceURL,usr,pwd);
```

- **Allocate a Statement object**

```
Statement stmt = dbConnect.createStatement( );
```

Steps In Using JDBC

- **Execute a query using the Statement object**

```
ResultSet rst = stmt.executeQuery("Select * from
  EmpTbl where salary >20000);
```

- **Retrieve data from the ResultSet object**

```
if ( rst != null ){
    rst.next();
    String Pname = rst.getString("name");
    double Psalary= rst.getDouble("salary"); }
```

- **Close the ResultSet**

```
rst.close( );
```

- **Close the Statement object**

```
stmt.close( );
```

- **Close the Connection object**

```
dbConnect.close( );
```

```
import java.sql.*;
public class PlainJDBC {
    public static void main(String args[ ]) {
            if (args.length != 1) {
               System.out.println("Usage: java JDBC custid");              System.exit(1);
            }
        int custID =  Integer.parseInt(args[0]);
        String query = new String ("Select Title, Year, Type from Orders O, Titles T, Tapes V " +
                                   "where V.TapeId=O.TapeId and T.TitleId=V.TitleId and " +
                                   "Status = 'O' and " +     "O.CustomerID= " + custID);
        try {
            // set the driver
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            //make connection
            Connection dbConnect = DriverManager.getConnection("jdbc:odbc:jdbc_book","jdbc_user","guest");
            // create a statement
            Statement stmt = dbConnect.createStatement();
            // make a query
            ResultSet rs = stmt.executeQuery(query);
            if (!rs.next())      System.out.println("No records for customer.");
            else {
                    do {
                        System.out.println(rs.getString("Title") + " c " + rs.getInt("Year") +": " + rs.getString("Type"));
                       } while (rs.next());
               stmt.close();              dbConnect.close();
             }//end of else
            }// end of try
            catch (Exception e) {
                        System.out.println(e.getMessage());  }
   }//end of main
}//end of class definition
```

A Servlet Using JDBC

```
package myservlets;

import java.io.*;
import java.sql.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class EmpData1 extends HttpServlet {
        public void doGet(
                HttpServletRequest request,
                HttpServletResponse response)
        throws ServletException, IOException {

        String eid,last, first, ext;

                response.setContentType("text/html");
                PrintWriter out = response.getWriter();
                out.println("<HTML>\n<HEAD<TITLE> Employee Database");
                out.println("</TITLE></HEAD> \n <BODY>");
                out.println("<h1>Employee Listing</h1>");
                out.println("<table border=\"1\" width=\"400\">");
                out.println("<tr>\n<td><b>ID</b></td><td><b>Last" +
                                "Name</b></td>");
                out.println("<td><b>First Name</b></td><td>" +
                                "<b>Extension</b></td></tr>");
```

```
try {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        Connection myConn = DriverManager.getConnection("jdbc:odbc:empdir");
        Statement stmt = myConn.createStatement();
        ResultSet myResultSet = stmt.executeQuery("select * from employee");
        if (myResultSet != null) {
            while (myResultSet.next()) {
            eid = myResultSet.getString("empid");
            last = myResultSet.getString("lastname");
            first = myResultSet.getString("firstname");
            ext = myResultSet.getString("extension");
                    out.println(" <tr>\n<td>" + eid + "</td><td>" +
                        last + "</td><td>" + first + "</td><td>" +
                                "</td></tr>");
           } /* of while */
        }  /* of if */
        stmt.close();
        myConn.close();
        out.println("</table>\n </body>\n</html>");
    }
    catch (Exception e) {
        out.println("Could not connect!\n");
    }
}// end of doGet
public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
                doGet(request, response);
        }// end of doPost
}// end of class EmpData
```

HTML File Using EmpData Servlet

```
<html>
<head>
<title>TEST WEB PAGE</title>
<meta http-equiv="Content-Type" content="text/html">
</head>
<body bgcolor="#FFFFFF">
<center><h1>Testing Servlets</h1></center>
<p></p><h2>Test Modules:</h2>
<a href="servlet/myservlets.EmpData">Employee Database Listing
</a><br>
</body>
</html>
```

A JSP(ShowEmp.jsp) Using JDBC

```
<html> <head><title>Employee Database</title></head>
<%@ page language="java" import="java.sql.*" %>
<body>
<h1>Employee Listing</h1>
<table border="1" width="400">
<tr>    <td><b>ID</b></td><td><b>Last Name</b></td>
        <td><b>First Name</b></td><td><b>Extension</b></td>  </tr>
<% Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   Connection myConn = DriverManager.getConnection("jdbc:odbc:empdir");
   Statement stmt = myConn.createStatement();

   ResultSet myResultSet = stmt.executeQuery("select * from employee");
   if (myResultSet != null) {
      while (myResultSet.next()) {
         String eid = myResultSet.getString("empid");
         String last = myResultSet.getString("lastname");
         String first = myResultSet.getString("firstname");
         String ext = myResultSet.getString("extension");
%>
<tr>
      <td><%= eid %></td>    <td><%= last %></td>
      <td><%= first %> </td> <td><%= ext %> </td>
</tr>
<%  } /* of while */
  }  /* of if */
  stmt.close();            myConn.close();
%>
</table></body></html>
```

HTML File Using JSP

```
<html>
<head>
<title>TEST WEB PAGE</title>
<meta http-equiv="Content-Type" content="text/html">
</head>
<body bgcolor="#FFFFFF">
<center><h1>Testing Java Server Pages</h1></center>
<p></p><h2>Test Modules:</h2>
<a href=”ShowEmp.jsp">Employee Database Listing
</a><br>
</body>
</html>
```

JDBC Reference

ResultSet

- Provides access to records resulting from a query.

METHODS:

```
public void afterLast () throws SQLException
```
- moves the cursor to the end of the result set.

```
public void beforeFirst () throws SQLException
```
- moves the cursor to the beginning of the resultset.

```
public void cancelRowUpdates () throws SQLException
```
- cancels any updates made to this row.

```
public void close() throws SQLException
```
- closes the resultset.

```
public void deleteRow () throws SQLException
```
- deletes the current row from the resultset and the database.

```
public void first () throws SQLException
```
- moves the cursor to the first row of the resultset.

```
public void last () throws SQLException
```
- moves the cursor to the last row of the resultset.

JDBC Reference

ResultSet

METHODS:

`public boolean getBoolean (String cname) throws SQLException`
- returns the value of the named column(cname) in Boolean format.

`public Date getDate (String cname) throws SQLException`
- returns the value of the named column(cname) in Date format.

`public double getDouble (String cname) throws SQLException`
- returns the value of the named column(cname) in Double format.

`public float getFloat (String cname) throws SQLException`
- returns the value of the named column(cname) in Float format.

`public int getInt (String cname) throws SQLException`
- returns the value of the named column(cname) in Int format.

`public String getString (String cname) throws SQLException`
- returns the value of the named column(cname) in String format.

`public ResultSetMetaData getMetaData ( ) throws SQLException`
- returns the meta-data object for this resultset.

JDBC Reference

ResultSet

METHODS:

`public void insertRow ( ) throws SQLException`
- inserts the contents of the insert row into the result set and database.

```
ResultSet rst = stmt.executeQuery("SELECT * FROM EMPLOYEE");
rst.moveToInsertRow( );
rst.updateString("lname", "Doe");          rst.updateString("fname","John");
rst.updateDouble("salary", 235550.0);      rst.updateInt("empNo", 7078);
rst.insertRow( );
rst.moveToCurrentRow( );
```

`public boolean isFirst ( ) throws SQLException`
- returns true if the resultset is positioned on the first row of the the result set.

`public boolean isLast ( ) throws SQLException`
- returns true if the resultset is positioned on the last row of the the result set.

`public void moveToCurrentRow ( ) throws SQLException`
- moves the resultset to the current row.

`public void moveToInsertRow ( ) throws SQLException`
- moves the resultset to a new insert row. Call moveToCurrentRow() to move back.

JDBC Reference

ResultSet

METHODS:

```
public void next ( ) throws SQLException
public void previous ( ) throws SQLException
```

- use to navigate one row forward or backward in the resultset.

```
public boolean relative (int rows ) throws SQLException
```

- moves a specified number of rows from the current record. Value for rows can be + or -.

```
public void updateBoolean(String cname, boolean b) throws SQLException
public void updateByte(String cname, byte b) throws SQLException
public void updateDate(String cname, Date b) throws SQLException
public void updateDouble(String cname, double b) throws SQLException
public void updateFloat(String cname, float b) throws SQLException
public void updateInt(String cname, int b) throws SQLException
public void updateString(String cname, String b) throws SQLException
```

- methods used to update a column(cname) in the current record of the resultset.

```
public void updateRow( ) throws SQLException
```

- this updates changes made to the current record of the resultset.

```
rst.updateDouble("salary", 235550.0);
rst.updateRow( );
```

JDBC Reference

ResultSetMetaData

- Provides meta-information about the types and properties of the columns in a resultset.

```
ResultSet rst = stmt.executeQuery("SELECT * FROM EMPLOYEE");
ResultSetMetaData meta = rst.getMetaData( );
int cols = meta.getColumnCount();
String strBuffer="";
for (int k=1; k<=cols; k++)
        strBuffer += meta.getColumnName(k);
```

METHODS:

```
public int getColumnCount () throws SQLException
```
- returns the number of columns in the result set.

```
public String getColumnName (int index) throws SQLException
```
- returns the name of the column in that column index.

```
public String getColumnTypeName (int index) throws SQLException
```
- returns the name of the SQL type of the specified column.

```
public String getTableName (int index) throws SQLException
```
- returns the name of the table for the specified column.

```
public boolean isReadOnly (int index) throws SQLException
public boolean isWritable (int index) throws SQLException
```
- returns true if column is read only/writable.

JDBC Reference

Statement

- Represents an embedded SQL statement and is used by an application to perform database access.

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection myConn = DriverManager.getConnection("jdbc:odbc:empdir");
Statement stmt = myConn.createStatement();
ResultSet myResultSet = stmt.executeQuery("select * from employee");
```

METHODS:

```
public boolean execute(String sql) throws SQLException
public ResultSet executeQuery(String sql) throws SQLException
public int executeUpdate(String sql) throws SQLException
```

- The *execute( )* method returns true if the statement has resultsets to process. The *executeQuery( )* method is used to execute a query specified in the string sql. The *executeUpdate( )* method is used to perform updates; it returns the number of rows affected by the update.

```
Statement stmt = myConn.createStatement();
int r = stmt.executeUpdate("INSERT into employee " +
        "(fname, lname,salary)" +
        " VALUES('John', 'Doe', 45000.00)");
int s = stmt.executeUpdate("UPDATE employee " +
        "set salary=salary * 1.05," +
        "set fname='Mike' " +
        "WHERE lname ='Doe' ");
```

JDBC Reference

Example: Creating a scrollable ResultSet

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection myConn = DriverManager.getConnection("jdbc:odbc:empdir");
Statement stmt =
        myConn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                               ResultSet.CONCUR_READ_ONLY);
ResultSet rst = stmt.executeQuery("select * from employee");
String strBuffer="";
rst.last(); //moves to the last record
strBuffer += rst.getString("lname");
rst.previous(); //moves to the previous record
strBuffer += rst.getString("lname");
rst.first(); //moves to the first record.

stmt.close();
```

JDBC Reference

Prepared Statements

- Extends the Statement interface and enables a SQL statement to contain parameters like a function definition. You then can execute a single statement repeatedly with different values for those parameters.

Example:

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection myConn = DriverManager.getConnection("jdbc:odbc:empdir");
PreparedStatement stmt =
        myConn.prepareStatement("UPDATE employee " +
                "SET salary = salary * ? WHERE empID = ? ");
for(int eid=1010; eid<=1050 eid +=2){
        stmt.setFloat(1, 1.15);
        stmt.setInt(2, eid);
        stmt.execute();
        stmt.clearParameters();
}
myConn.commit();
stmt.close();
```

JDBC Reference

Batch Processing with Prepared Statements

- Similar processing actions are executed as a single group in batch processing.

Example:

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection myConn = DriverManager.getConnection("jdbc:odbc:empdir");
PreparedStatement stmt =
        myConn.prepareStatement("UPDATE employee " +
                "SET salary = salary * ? WHERE empID = ? ");
for(int eid=1010; eid<=1050 eid +=2){
        stmt.setFloat(1, 1.15);
        stmt.setInt(2, eid);
        stmt.addBatch();
}
int r = stmt.executeBatch();
myConn.commit();
stmt.close();
```