Application Programming for Relational Databases

Cyrus Shahabi

Computer Science Department

University of Southern California

shahabi@usc.edu

- Overview
- JDBC Package
- Connecting to databases with JDBC
- Executing select queries
- Executing update queries

Overview

- **Role of an application: Update databases, extract info, through:**
  - **User interfaces**
  - **Non-interactive programs**
- **Development tools (Access, Oracle):**
  - **For user Interfaces**
- **Programming languages (C, C++, Java,…):**
  - **User Interfaces**
  - **Non-Interactive programs**

Client server architecture

- **Database client:**
  - **Connects to DB to manipulate data:**
    - **Software package**
    - **Application (incorporates software package)**
- **Client software:**
  - **Provide general and specific capabilities**
  - **Oracle provides different capabilities as Sybase (its own methods, …)**

Client server architecture

- **Client-Server architectures:**
  - **2 tier : data server and client**
  - **3 tier**
    - **Tier 1: Client-tier**
      - **user interface : responsible for the presentation of data, receiving user events and controlling the user interface**
    - **Tier 2: Application-server-tier (new tier)**
      - **Middleware : protects the data from direct access by the clients.**
    - **Tier 3: Data-server-tier**
      - **DB server : responsible for data storage**
- **Boundaries between tiers are logical. It is quite easily possible to run all three tiers on one and the same (physical) machine**
- **Clear separation of user-interface-control and data presentation from application-logic**

Client server architecture

- **3-tier architecture**

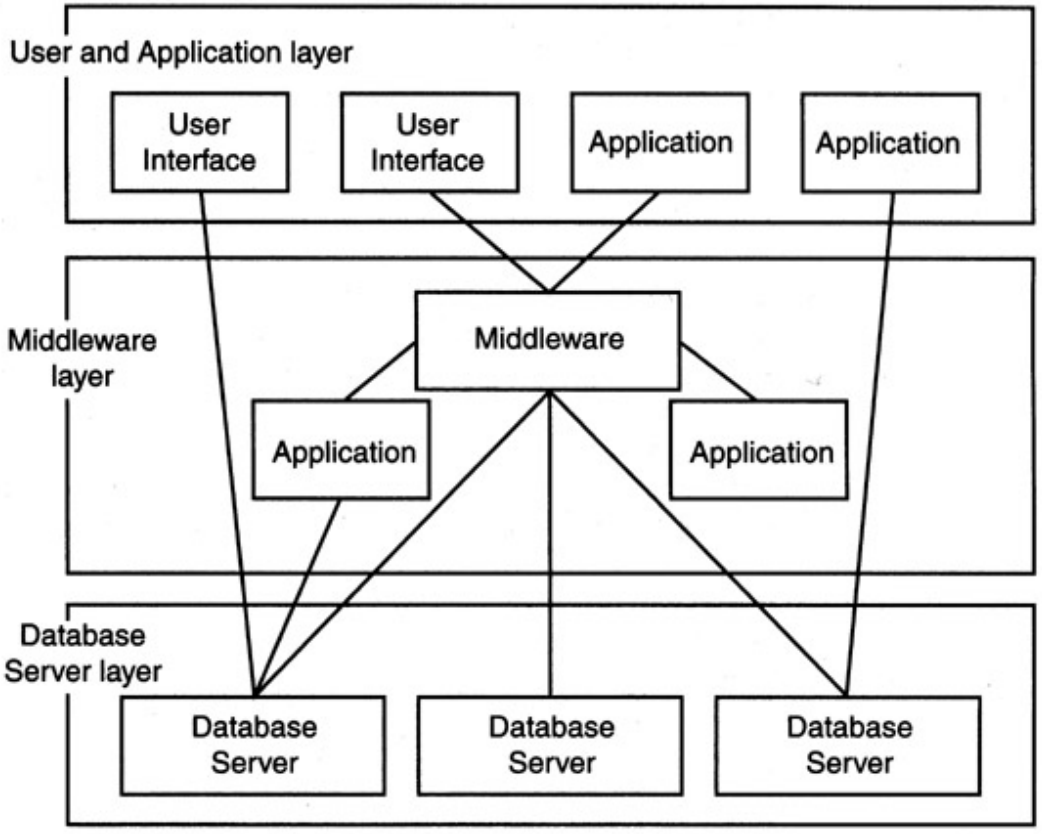


FIGURE 8.1
A variety of client-server architectures for information systems

- **Middleware: Server for client & Client for DB**

Client server architecture

- **Example: Web interaction with DB**
  - **Layer 1: web browser**
  - **Layer 2: web server + cgi program**
  - **Layer 3: DB server**

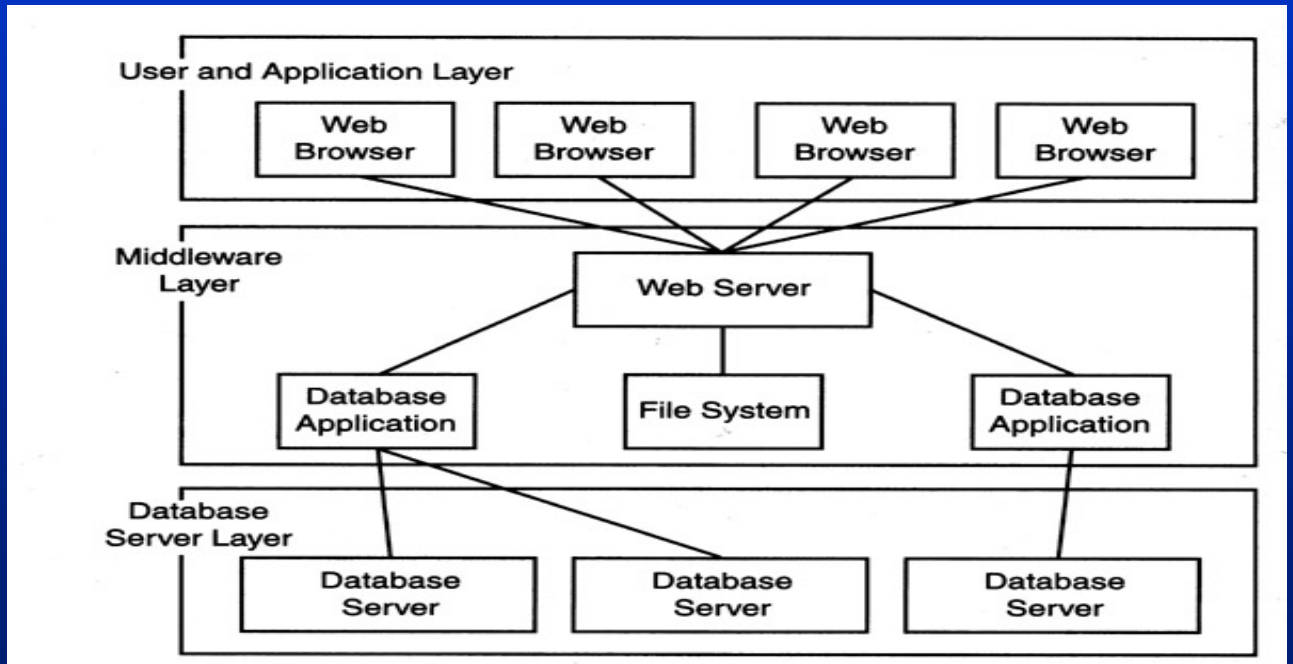


FIGURE 8.2
Architecture of a Web site supported by databases

Database Interaction

- Direct interaction with DB
- For implementing applications
- Not professional!
- Generates stand alone application
- Access application:
  - GUI + “Visual Basic for Applications” code

Database Interaction

- **Connection to DB through:**
  - **Microsoft Jet database engine**
    - **Support SQL access**
    - **Different file formats**
  - **Other Database Connectivity (ODBC)**
    - **Support SQL DBs**
    - **Requires driver for each DB server**
      - **Driver allows the program to become a client for DB**
    - **Client behaves Independent of DB server**

Database Interaction

- **Making data source available to ODBC application:**
  - **Install ODBC driver manager**
  - **Install specific driver for a DB server**
  - **Database should be registered for ODBC manager**
- **How application works with data source:**
  - **Contacts driver manager to request for specific data source**
  - **Manager finds appropriate driver for the source**

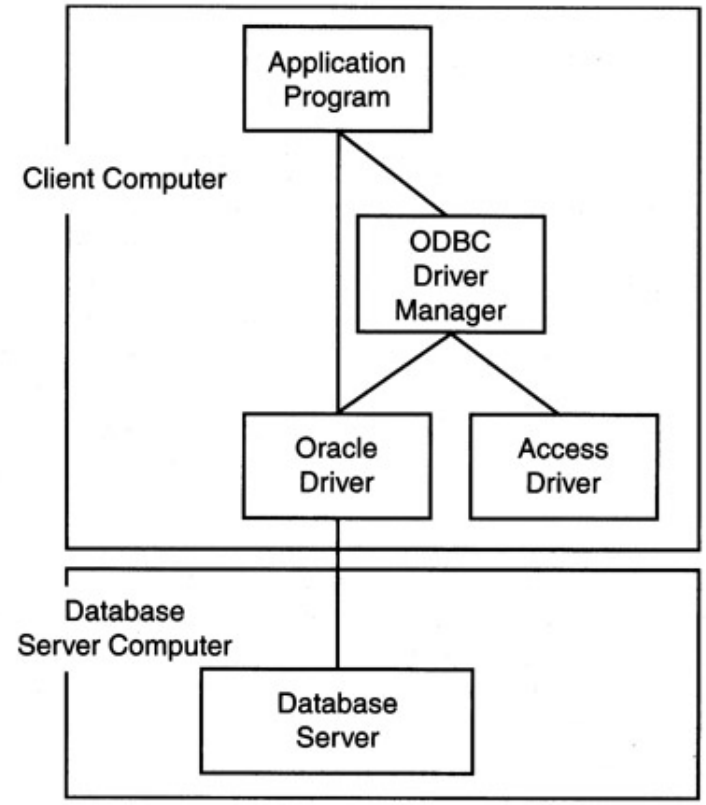


FIGURE 8.3
The ODBC architecture for database access

Database Interaction

- **Embedded SQL**
- **Extension of a language (C++,C) with new commands:**
  - **Void addEmployee(char *ssn, char *lastname,char *firstname)**
  - **Exec SQL**

    - Insert into customer(ssn, lastname, firstname)

    values(:ssn, :lastname, :firstname)

  - **Not legal language**
  - **Compilation precedes by a translation preprocessor from embedded SQL into legal C**
  - **Advantages: ???**
  - **Disadvantages:**
    - **Not portable between database systems**
    - **Difficult debugging**

Database Interaction

- **ODBC :**
  - **ODBC (Open Database Connectivity)**
    - **provides a way for client programs (eg Visual Basic, Excel, Access, Q+E etc) to access a wide range of databases or data sources**
  - **ODBC stack**
    - **ODBC Application :Visual Basic, Excel, Access**
    - **Driver Manager :ODBC.DLL**
    - **ODBC Driver :ODBC Driver varies for data source**
    - **Database Transport :database transport**
    - **Network Transport :TCP/IP or other protocol driver**
    - **Data Source :data source (Oracle, MySQL)**

Database Interaction in Java

- **JDBC (Java Database Connectivity):**
  - **Java.sql package**
  - **More user-friendly**
  - **Less Programming**
  - **Less involvement with details**
  - **Difference between JDBC and ODBC:**
    - **JDBC driver manager is part of the application**

JDBC: Architecture

- **Four Architectural Components:**
  - **Application (initiates and terminates connections, submits SQL statements)**
  - **Driver manager (load JDBC driver)**
  - **Driver (connects to data source, transmits requests and returns/translates results and error codes)**
  - **Data source (processes SQL statements)**

JDBC package

- **Collection of interfaces and classes:**
  - **DriverManager: Loads the driver**
  - **Driver: creates a connection**
  - **Connection: represents a collection**
  - **DatabaseMetaData: information about the DB server**
  - **Statement: executing queries**
  - **PreparedStatement: precompiled and stored query**
  - **CallableStatment: execute SQL stored procedures**
  - **ResultSet: results of execution of queries**
  - **ResultSetMetaData: meta data for ResultSet**

- **Reminder: Each JDBC package implements the interfaces for specific DB server**

JDBC, different strategies

- **Strategies to USE JDBC**
  - **JDBC-ODBC bridge**
    - **Con: ODBC must be installed**
  - **JDBC database client**
    - **Con: JDBC driver for each server must be available**
  - **JDBC middleware client**
    - **Pro: Only one JDBC driver is required**
    - **Application does not need direct connection to DB (e.g., applet)**

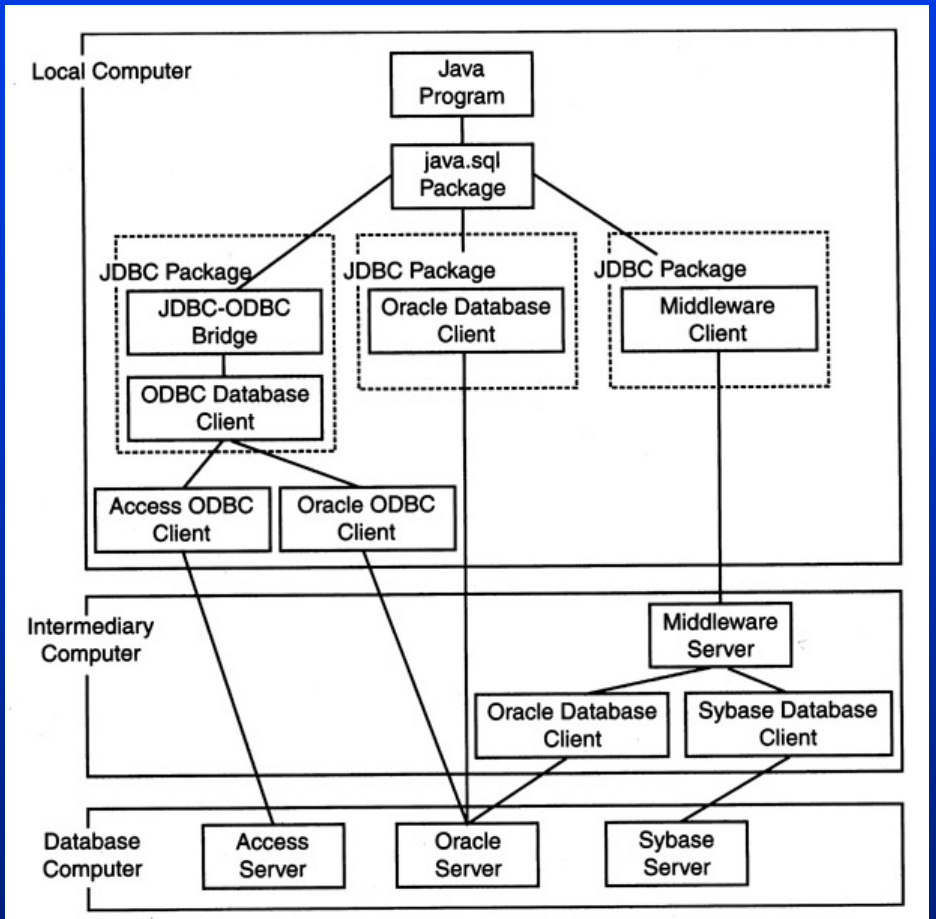
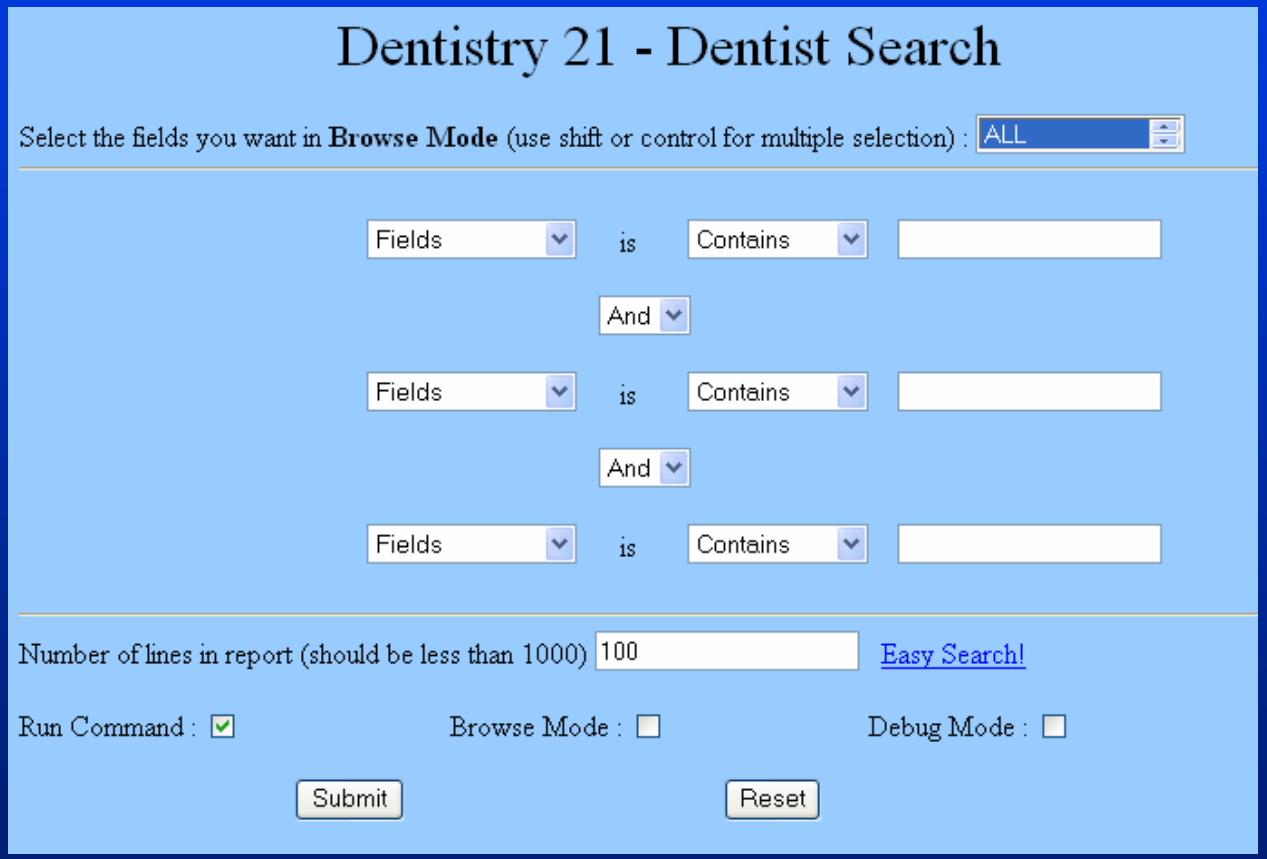


FIGURE 8.4
Strategies for implementing JDBC packages

Connecting with JDBC

- **Lets look at a real application: Dentist Search**



Connecting to DB with JDBC

- **Database connection needs two pieces**
  - **JDBC package driver class name**
    - **Package driver provide connection to DB**
  - **URL of the database**
    - **JDBC package designator**
    - **Location of the server**
    - **Database designator, in form of:**
      - **Server name, Database name, Username, password, …**
      - **Properties**

Connecting to DB with JDBC

- **Step 1: Find, open and load appropriate driver**
  - **1. Class.forName(“sun.jdbc.odbc.JdbcOdbcDriver”);**
  - **3. Class.forName(“symantec.dbAnywhere.driver”);**
  - **4. Class.forName("com.informix.jdbc.IfxDriver");**
  - **Or:**
  - **4. DriverManager.registerDriver(your jdbc driver);**
  - **Informs availability of the driver to “DriverManager” (registers the driver with DriverManager)**
  - **ORACLE JDBC**

Connecting to DB with JDBC

```
String driver = "com.mysql.jdbc.Driver";

// the "url" to our DB, the last part is the name of the DB
String url = "jdbc:mysql://localhost/dentists";

// the default DB username and password may be the same as your control panel login
String user = "system_user";
String pass = "confidential_pass";

Class.forName(driver);
Connection con = DriverManager.getConnection(url, user, pass);
```

Connecting to DB with JDBC

- **Step 2: Make connection to the DB**
  - **Connection conn = DriverManager(URL, Properties);**
    - **Properties: specific to the driver**
  - **URL = Protocol + user**
    - **Protocol= jdbc:<subprotocol>:<subname>**
      - **E.g.: jdbc:odbc:mydatabase**
      - **E.g.: jdbc:oracle:thin://oracle.cs.fsu.edu/bighit**

```
// initialize the Connection, with our DB info ...
Connection con = DriverManager.getConnection(url, user, pass);
```

Connecting to DB with JDBC

- **Step 3: Make Statement object**
  - **Used to send SQL to DB**

Statement stat = con.createStatement();

- **Step 4: issue select queries**
  - **executeQuery(): SQL that returns table**
    - **Every call to executeQuery() deletes previous results**
  - **executeUpdate(): SQL that doesn't return table**
  - **Execute(): SQL that may return both, or different thing**

- **Step 5: obtain metadata (optional)**
  - **Return the results as ResultSet object**
    - **Meta data in ResultSetMetaData object**

ResultSet res = stat.executeQuery(sql_command);

ResultSet res = stat.executeQuery(select * from dentists where specialty like "%ortho%" and city like "%los angeles%" limit 0,100 ;

Executing select queries

SQL Command: select * from dentists where specialty like "%ortho%" and (city like "%los angeles%") limit 0,100 ;

	Name	Tel	City	ST	Practice	Specialty	Modified	Reminder	Email
1	James Young	323-663-4610	Los Angeles	CA		Orthodontics/Dentofacial Orthopedics	2005-05-31	2005-05-01	jvyoung8@aol.com
2	Kang Ting	310-825-4705	Los Angeles	CA		Orthodontics/Dentofacial Orthopedics	2005-05-31	2005-05-01	
3	Peter M Sinclair	213-740-4236	Los Angeles	CA		Orthodontics/Dentofacial Orthopedics	2005-05-31	2005-05-01	sinclair@usc.edu
4	Eung-Kwon Pae	310-825-7191	Los Angeles	CA		Orthodontics/Dentofacial Orthopedics	2005-05-31	2005-05-01	epae@dent.ucla.edu
5	Yen P Miao		Los Angeles	CA		Orthodontics/Dentofacial Orthopedics	2005-05-31	2005-05-01	ymiao@ucla.edu
6	James Mah	213-740-3762	Los Angeles	CA		Orthodontics/Dentofacial Orthopedics	2005-05-31	2005-05-01	jamesmah@usc.edu
7	Gordon S Kilmer		Los Angeles	CA		Orthodontics/Dentofacial Orthopedics	2005-05-31	2005-05-01	kilmer@hsc.usc.edu
8	Shawn Kim	213-380-7900	Los Angeles	CA		Orthodontics/Dentofacial Orthopedics	2005-05-31	2005-05-01	skim777@hotmail.com
9	Roxan Humes	323-294-1170	Los Angeles	CA		Orthodontics/Dentofacial Orthopedics	2005-05-31	2005-05-01	
10	John R Garol	310-208-8651	Los Angeles	CA		Orthodontics/Dentofacial Orthopedics	2005-05-31	2005-05-01	jackgarol@aol.com
11	Nader Dayani	310-826-7494	Los Angeles	CA		Orthodontics/Dentofacial Orthopedics	2005-05-31	2005-05-01	ddsbraces@yahoo.com
12	David Cheney		Los Angeles	CA		Orthodontics/Dentofacial Orthopedics	2005-05-31	2005-05-01	cheney_98@yahoo.com
13	Maria Amorin Singson	661-259-5540	Los Angeles	CA		Orthodontics/Dentofacial Orthopedics	2005-05-31	2005-05-01	
14	Stanley M Miyawaki	310-826-6694	Los Angeles	CA		Orthodontics/Dentofacial Orthopedics	2005-05-31	2005-05-01	drmiyawaki@verizon.net

Executing select queries

- **Step 6: retrieve the results of select queries**
  - **Using ResultSet object**
    - **Returns results as a set of rows**
    - **Accesses values by column name or column number**
    - **Uses a cursor to move between the results**
    - **Supported methods:**
      - **JDBC 1: scroll forward**
      - **JDBC 2: scroll forward/backward, absolute/relative positioning, updating results.**
      - **JDBC 2: supports SQL99 data types(blob, clob,…)**

Executing select queries

```
Statement stmt = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE);
ResultSet srs = stmt.executeQuery( "SELECT NAME, SPECIALTY from DENTISTS");
while (srs.next())
{
      String name = srs.getString("NAME");
      String specialty = srs.getFloat("PRICE");
      System.out.println(name + "     " + specialty);
}

Statement stmt = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet srs = stmt.executeQuery("SELECT …");

int numCols = res.getMetaData().getColumnCount();
while (res.next()) { //note MySql start with the index 1 as the first column
        // display by column
        dispList(counter, res.getString(1), res.getString(4), res.getString(18), res.getString(21),
      res.getString(12), res.getString(13), res.getString(2), res.getString(3), res.getString(37),
      res.getString(35));
}
```

Executing select queries

```
ResultSetMetaData rsmd = res.getMetaData();
// display by column name
if (rsmd.getColumnName(col).compareTo("ID") == 0) {
        _id_ = res.getString(col);
}
dispList(counter, _id_, _name_, _practice_name_, _address1_, _address2_, _city_, _st_,
    _zip_, _tel_,  _email_,_modified_);
```

Matching Java and SQL Data Types

SQL Type	Java class	ResultSet get method
BIT	Boolean	getBoolean()
CHAR	String	getString()
VARCHAR	String	getString()
DOUBLE	Double	getDouble()
FLOAT	Double	getDouble()
INTEGER	Integer	getInt()
REAL	Double	getFloat()
DATE	java.sql.Date	getDate()
TIME	java.sql.Time	getTime()
TIMESTAMP	java.sql.TimeStamp	getTimestamp()

Executing update queries

- **Step 7: issue update queries**
  - **Queries that return a row count (integer) as result**
    - **Number of rows affected by the query**
    - **-1 if error**
  - **Using statement object**
  - **Uses executeUpdate() method**

```
Statement stat = con.createStatement();
i = stat.executeUpdate(UPDATE `dentists` SET ``phone` = '907-225-9439',
      WHERE `full_name` =  'George Allen ' );
```

  - **Meta data in ResultSetMetaData object**

Dentist Update Page

Executing update queries

- **Step 8: More Advanced**
  - **Cursors**
    - **forward, backward, absolute/relative positions**

```
// move the cursor explicitly to the position after the last row
srs.afterLast();

// first , last , beforeFirst , and afterLast move the cursor to the row indicated in their names
srs. first ();

// if number is positive, the cursor moves the given number from the beginning
// negative number moves the cursor backward the given number of rows
srs.absolute(4); // cursor is on the fourth row
srs.relative(-3); // cursor is on the first row
srs.relative(2); // cursor is on the third row
```

Executing update queries

- **Step 8: More Advanced**
  - **Use PreparedStatement**
    - **faster than regular Statement : if you need to use the same, or similar query with different parameters multiple times, the statement can be compiled and optimized by the DBMS just once**

```
PreparedStatement prepareUpdatePrice = con.prepareStatement(
                                        "UPDATE Dentists SET SalesPerson = ? WHERE Zip = ?");

prepareUpdatePrice.setString(1, “John Lee");

prepareUpdatePrice.setInt(2, 92560);
```

Mapping Objects

- **To read attributes that are retrieved as objects:**
  - **Example: Spatial data types**
    - **Read "Oracle Spatial – User's Guide and Reference"**
      - **Chapter 2 for geometry types**
      - **Chapter 9-14 for geometry functions**
    - **Read "Oracle Spatial API Document" for reading geometry types in Java**