G52APR
Applications Programming

Java DataBase Connectivity – JDBC

What is JDBC?

- "An API that lets you access virtually any tabular data source from the Java programming language"
- "… access virtually any data source, from relational databases to spreadsheets and flat files."
- We'll focus on accessing Oracle type databases

2

Tabular data source

Table 1: Employees

Employee_Number	First_Name	Last_Name	Date_of_Birth	Car_Number
10001	Axel	Washington	28-Aug-43	5
10083	Arvid	Sharma	24-Nov-54	null
10120	Jonas	Ginsberg	01-Jan-69	null
10005	Florence	Wojokowski	04-Jul-71	12

3

Relational Database

Employee_Number	First_Name	Last_Name	Date_of_Birth	Car_Number
10001	Axel	Washington	28-Aug-43	5
10083	Arvid	Sharma	24-Nov-54	null
10120	Jonas	Ginsberg	01-Jan-69	null
10005	Florence	Wojokowski	04-Jul-71	12

Car_Number	Make	Model	Year
5	Honda	Civic DX	2006
12	Toyota	Corolla	2009

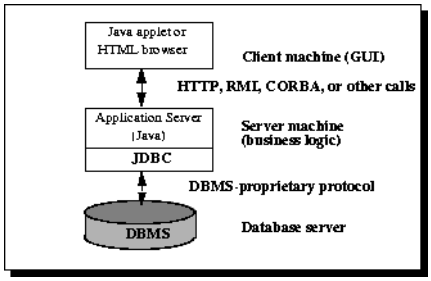
4

Relational Database

- A relational database presents information in tables with rows and columns.
- A distinguishing feature of relational databases is that it is possible to get data from more than one table in what is called a join.
- A Relational Database Management System (RDBMS) handles the way data is stored, maintained, and retrieved.
- Structural Query Language (SQL) is a language designed to be used with relational databases.

5

General Architecture



6

Basic steps to use a database

1. Establish a connection
2. Create JDBC Statements
3. Execute SQL Statements
4. GET ResultSet
5. Close the connection

7

1. Establish a connection

- **import java.sql.*;**
- **Load the vendor specific driver**

```
Class.forName("oracle.jdbc.driver.OracleDriver");
    // Dynamically loads a driver class, for Oracle database
```

- **Make the connection**

```
Connection con =
DriverManager.getConnection("jdbc:derby://localhost:1527/Employees", username, passwd);
```

// Establishes connection to database by obtaining a *Connection* object

8

Database address

- The address of the database is:
  jdbc:derby://localhost:1527/Employees
  The first part, **jdbc:derby://localhost**, is the database type and server that you're using. The 1527 is the port number. The database name is **Employees**. This can all go in a String variable:

```
String host = "jdbc:derby://localhost:1527/Employees";
```

Two more strings can be added for the username and password:

```
String uName = "Your_Username_Here";
String uPass= " Your_Password_Here ";
Connection con=DriverManager.getConnection(host, uName, uPass);
```

9

2. Create JDBC statement(s)

- Statement stmt = con.createStatement() ;

// Creates a Statement object for sending SQL statements to the database

10

3. Executing SQL Statements

- String queryThing = " SELECT * FROM Employees";
  stmt.executeQuery(queryThing);

- String insertThing = "Insert into Thing values“ +
  "(123456789,abc,100)";
  stmt.executeUpdate(insertThing);

11

4. Get ResultSet

String queryThing = “SELECT * FROM Employees";

ResultSet rs = Stmt.executeQuery(queryThing);

```
while (rs.next()) {
    int ssn = rs.getInt("Employee_Number");
    String first_name = rs.getString(“First_name");
    String last_name = rs.getString(“Last_name");
    int marks = rs.getInt(“Car_Number");
}
```

12

5. Close connection

- stmt.close();
  // close statement
- con.close();
  // close connection

13

An Example

```
public static void viewTable(Connection con, String dbName) throws SQLException {
    Statement stmt = null;
    String query = "SELECT Employee_Number, First_Name, Last_Name, Date_of_Birth, Car_Number
    FROM " + dbName";
    try {
            stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery(query);
            while (rs.next()) {
                int EmployeeNumber = rs.getInt(" Employee_Number");
                String FirstName = rs.getString(" First_Name");
                String LastName = rs.getString(" First_Name");
                String DateOfBirth = rs.getString(" Date_of_Birth");
                int CarNumber = rs.getInt("Car_Number");
                System.out.println(EmployeeNumber + "\t" + FirstName + "\t" + LastName  + "\t" +
                        CarNumber);
            }
    } catch (SQLException e ) {
            JDBCTutorialUtilities.printSQLException(e);
    } finally {
            stmt.close();
    }
}
```

14

SQL commands

- *String query = "SELECT Employee_Number, Date_of_Birth, Car_Number FROM Employees WHERE Car_Number IS NOT NULL";*

- Data Manipulation Language (DML)
  - SELECT
  - INSERT
  - DELETE
  - …
- Data Definition Language (DDL)
  - CREATE TABLE
  - ALTER TABLE
  - …

SQL – Structured Query Language

15

SQL commands

- ResultSet rs = stmt.executeQuery("**SELECT * FROM table_name**")

// select all the records from a table

- ResultSet rs = stmt.executeQuery(" **SELECT * FROM table_name WHERE column_name=value**");

// select all the records from a table

- …("**SELECT col_blob FROM mysql_all_table**");

// select column reference from all tables

http://download.oracle.com/javase/1.4.2/docs/api/java/sql/package-summary.html

NOTE: SQL is not case sensitive
A BLOB is a reference to data in a database.

16

Transactions and JDBC

- JDBC allows SQL statements to be grouped together into a single transaction.
- "Sequence of operations performed as a single logical unit of work".
  - Atomic: all the work in the transaction is treated as a single unit. Either it is all performed or none of it is.
  - Consistent: a completed transaction leaves the database in a consistent internal state.
  - Isolations: the transaction sees the database in a consistent state. If two transactions try to update the same table, one will go first and then the other will follow.
  - Durability means that the results of the transaction are permanently stored in the system.

17

Transactions and JDBC

- Transaction control is performed by the Connection object, default mode is auto-commit, i.e., each SQL statement is treated as a transaction.
- We can turn off the auto-commit mode with con.setAutoCommit(false);
- And turn it back on with con.setAutoCommit(true);
- Once auto-commit is off, no SQL statement will be committed until an explicit commit is invoked con.commit();
- At this point all changes done by the SQL statements will be made permanent in the database.

18

Using Transactions

- Step 1: turn off autocommit:
  - conn.setAutoCommit(false);
- Step 2: create and execute statements like normal
- Step 3: fish or cut bait: commit or rollback
  - if all succeeded:
    - conn.commit();
  - else, if one or more failed:
    - conn.rollback();
- Step 4: turn autocommit back on
  - conn.setAutoCommit(true);

19

Handling Errors with Exceptions

- Programs should recover and leave the database in a consistent state.
- If a statement in the try { …} block throws an exception or warning, it can be caught in one of the corresponding catch statements.
- E.g., you could rollback your transaction in a catch { …} block or close database connection and free database related resources in finally {…} block.

20

Interface ResultSet

- A table of data representing a database result set, which is usually generated by executing a statement that queries the database.
- A ResultSet object maintains a cursor pointing to its current row of data.
  - Initially the cursor is positioned before the first row.
  - The next() method moves the cursor to the next row
  - returns false when there are no more so it can be used in a while loop to iterate through the result set.
  - Not updatable and has a cursor that moves forward only.

21

Retrieve data from ResultSet

```
while (rs.next()) {
   int EmployeeNumber = rs.getInt(" Employee_Number");
   String FirstName = rs.getString(" First_Name");
   String LastName = rs.getString(" First_Name");
   String DateOfBirth = rs.getString(" Date_of_Birth");
   int CarNumber = rs.getInt("Car_Number");
   System.out.println(EmployeeNumber + "\t" + FirstName
   + "\t" + LastName  + "\t" + CarNumber);
   }
```

22

Retrieve data from ResultSet

```
while (rs.next()) {
   int EmployeeNumber = rs.getInt(" Employee_Number");
   String FirstName = rs.getString(" First_Name");
   String LastName = rs.getString(" First_Name");
   String DateOfBirth = rs.getString(" Date_of_Birth");
   int CarNumber = rs.getInt("Car_Number");
   int EmployeeNumber = rs.getInt(" Employee_Number");
   System.out.println(EmployeeNumber + "\t" + FirstName +
   "\t" + LastName  + "\t" + CarNumber);
   }
```

23

More ResultSet details

- The ResultSet interface provides getter methods (getBoolean, getInt, and so on) for retrieving column values from the current row.
- Values can be retrieved using either the index number of the column or the name of the column (In general, using the column index will be more efficient).
- Columns are numbered from 1. (For maximum portability, result set columns within each row should be read in left-to-right order, and each column should be read only once).
- For the getter methods, a JDBC driver attempts to convert the underlying data to the Java type specified in the getter method and returns a suitable Java value (see table below).

24

Create a new table

```
public void createTable() throws SQLException {
 String createString = "create table " + dbName + ".COFFEES " +
    "(COF_NAME varchar(32) NOT NULL, " +
    "SUP_ID int NOT NULL, " +
    "PRICE float NOT NULL, " +
    "SALES integer NOT NULL, " +
    "TOTAL integer NOT NULL, " +
    "PRIMARY KEY (COF_NAME), " +
    "FOREIGN KEY (SUP_ID) REFERENCES " + dbName + ".SUPPLIERS (SUP_ID))";
 Statement stmt = null;
 try {
    stmt = con.createStatement();
    stmt.executeUpdate(createString);
 }catch (SQLException e) {
    JDBCTutorialUtilities.printSQLException(e);
 } finally {
    if (stmt != null) { stmt.close(); }
 }
}
```


JDBC 2 – Scrollable Result Set

…
Statement stmt =
con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY);
String query = "select students from class where type='not sleeping' ";
ResultSet rs = stmt.**executeQuery**(query);

rs.**previous**(); / / go back in the RS (not possible in JDBC 1…)
rs.**relative**(-5); / / go 5 records back
rs.**relative**(7); / / go 7 forward
rs.**absolute**(100); / / go to 100th record
…

Cursor Methods

- next: Moves the cursor forward one row. Returns true if the cursor is now positioned on a row and false if the cursor is positioned after the last row.
- previous: Moves the cursor backward one row. Returns true if the cursor is now positioned on a row and false if the cursor is positioned before the first row.
- first: Moves the cursor to the first row in the ResultSet object. Returns true if the cursor is now positioned on the first row and false if the ResultSet object does not contain any rows.
- last: Moves the cursor to the last row in the ResultSet object. Returns true if the cursor is now positioned on the last row and false if the ResultSet object does not contain any rows.
- beforeFirst: Positions the cursor at the start of the ResultSet object, before the first row. If the ResultSet object does not contain any rows, this method has no effect.
- afterLast: Positions the cursor at the end of the ResultSet object, after the last row. If the ResultSet object does not contain any rows, this method has no effect.
- relative(int rows): Moves the cursor relative to its current position.
- absolute(int row): Positions the cursor on the row specified by the parameter row.

Scrollable and Updatable ResultSet

- Assuming con is a valid Connection object

Statement stmt =
con.createStatement(
ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_UPDATABLE);

ResultSet rs = stmt.executeQuery("SELECT a, b FROM TABLE2");

// rs will be scrollable, will not show changes made by others, and will be updatable

JDBC 2 – Updateable ResultSet

…
Statement stmt =
con.createStatement(ResultSet.TYPE_FORWARD_ONLY,
ResultSet.CONCUR_UPDATABLE);
String query = " select students, grade from class
where type='really listening this presentation' ";
ResultSet rs = stmt.executeQuery(query);
…
while (rs.next())
{
int grade = rs.getInt("grade");
rs.updateInt("grade", grade+10);
rs.updateRow();
}

ResultSet setters etc 1

- A set of updater methods were added to this interface in the JDBC 2.0 API. The comments regarding parameters to the getter methods also apply to parameters to the updater methods.
- The updater methods may be used in two ways:

1. to update a column value in the current row.
   - In a scrollable ResultSet object, the cursor can be moved backwards and forwards, to an absolute position, or to a position relative to the current row.
   - The following code fragment updates the NAME column in the fifth row of the ResultSet object rs and then uses the method updateRow to update the data source table from which rs was derived.

rs.absolute(5); // moves the cursor to the fifth row of rs
rs.updateString("NAME", "Fred"); // updates the NAME column of row 5 to be Fred
rs.updateRow(); // updates the row in the data source

ResultSet setters etc 2

2. To insert column values into the insert row.
- An updatable ResultSet object has a special row associated with it that serves as a staging area for building a row to be inserted. The following code fragment moves the cursor to the insert row, builds a three-column row, and inserts it into rs and into the data source table using the method insertRow.

```
rs.moveToInsertRow(); // moves cursor to the insert row
rs.updateString(1, "Fred"); // updates the first column of the insert row to be Fred
rs.updateInt(2,35); // updates the second column to be 35
rs.updateBoolean(3, true); // updates the third column to true
rs.insertRow();
rs.moveToCurrentRow();
```

31

ResultSet setters etc 3

- A ResultSet object is automatically closed when the Statement object that generated it is closed, re-executed, or used to retrieve the next result from a sequence of multiple results.
- The number, types and properties of a ResultSet object's columns are provided by the ResulSetMetaData object returned by the ResultSet.getMetaData method.

32

Metadata from DB

- A Connection's database is able to provide schema information describing its tables, its supported SQL grammar, its stored procedures the capabilities of this connection, and so on
  - Stored procedures are a group of SQL statements that form a logical unit and perform a particular task
- This information is made available through a **DatabaseMetaData** object.

33

Interface ResultSetMetaData

- An object that can be used to get information about the types and properties of the columns in a ResultSet object.
- The following code fragment creates the ResultSet object rs, creates the ResultSetMetaData object rsmd, and uses rsmd to find out how many columns rs has and whether the first column in rs can be used in a WHERE clause.

```
ResultSet rs = stmt.executeQuery("SELECT a, b, c FROM TABLE2");
ResultSetMetaData rsmd = rs.getMetaData();
int numberOfColumns = rsmd.getColumnCount();
boolean b = rsmd.isSearchable(1);
```

34

JDBC – Metadata

```
public static void printRS(ResultSet rs) throws SQLException
{
    ResultSetMetaData md = rs.getMetaData();
    // get number of columns
    int nCols = md.getColumnCount();
    // print column names
    for(int i=1; i < nCols; ++i)
    System.out.print( md.getColumnName( i)+",");
    / / output resultset
    while ( rs.next() ){
    for(int i=1; i < nCols; ++i)
    System.out.print( rs.getString( i)+",");
    System.out.println( rs.getString(nCols) );
    }
}
```

35

JDBC and beyond

- (JNDI) Java Naming and Directory Interface
  - API for network-wide sharing of information about users, machines, networks, services, and applications
  - Preserves Java's object model
- (JDO) Java Data Object
  - Models persistence of objects, using RDBMS as repository
  - Save, load objects from RDBMS
- (SQLJ) Embedded SQL in Java
  - Standardized and optimized by Sybase, Oracle and IBM
  - Java extended with directives: # sql
  - SQL routines can invoke Java methods
  - Maps SQL types to Java classes

36

JDBC references

- JDBC Data Access API – JDBC Technology Homepage
  - http://java.sun.com/products/jdbc/index.html
- JDBC Database Access – The Java Tutorial
  - http://java.sun.com/docs/books/tutorial/jdbc/index.html
- JDBC Documentation
  - http://java.sun.com/j2se/1.4.2/docs/guide/jdbc/index.html
- java.sql package
  - http://java.sun.com/j2se/1.4.2/docs/api/java/sql/package-summary.html
- JDBC Technology Guide: Getting Started
  - http://java.sun.com/j2se/1.4.2/docs/guide/jdbc/getstart/GettingStartedTOC.fm.html
- JDBC API Tutorial and Reference (book)
  - http://java.sun.com/docs/books/jdbc/

37