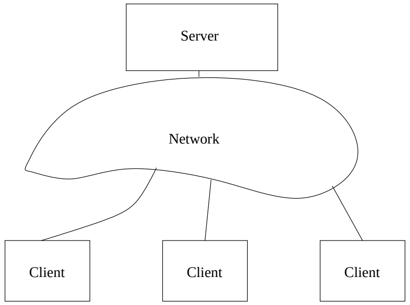
An Introduction to JDBC

S. Sudarshan

Computer Science and Engg. Dept
Indian Institute of Technology, Bombay

Introduction

- Client Server Database Architectures:
  - Backend: database server
  - Frontend: forms/gui/web interface



Communicating With the Server

- Standard way of expressing queries: SQL
- Problem: How to send queries to server and receive results in a standard *database independent* way?
- API (application program interface) standards for database communication:
  - ODBC (Open Database Connectivity, for C)
  - JDBC ("Java Database Connectivity", for Java)
- There are also database specific ways of communication, e.g. Oracle OCI

Steps in Database Access Using JDBC

1. Open connection to database (including login/password)
2. Send SQL statements
3. Receive results (set of tuples – rowset)
4. Iterate over result set, extracting fields
5. Exceptions

All above similar to ODBC, but calls are simpler

Example JDBC Code

```
try {
  DriverManager.registerDriver(
                  new oracle.jdbc.driver.OracleDriver());
  Connection myCom =
           DriverManager.getConnection(
                  "jdbc:oracle:thin:@everest:1521:GEN",
                  "sudarsha","mypasswd");
  Statement stmt = con.createStatement();
  ResultSet rs = stmt.executeQuery(
                           "SELECT a, b, c FROM Table1");
  while (rs.next()) {
    int i = rs.getInt("a");
    String s = rs.getString("b");
    float f = rs.getFloat("c");
    System.out.println("ROW = " + i + " " + s + " " + f);
  }
} catch(SQLException sqle){
  System.out.println("Exception " + sqle);
}
```


More on JDBC

- More on drivers and connections later
- Getting result fields:
  - `rs.getString("b")` and `rs.getString(2)` equivalent since "b" is second argument of select result.
- Null values

```
int a = rs.getInt("a");
if (rs.wasNull()) System.out.println("Got null value");
```

- Exceptions
- Three forms of execute calls
  - executeQuery(..) : for queries
  - executeUpdate(..) : for updates
  - execute(..) : general interface, rarely used

PreparedStatement

Subclass of Statement, allows queries to be compiled and executed with different arguments

```
PreparedStatement pstmt = con.prepareStatement(
    "UPDATE table4 SET m = ? WHERE x = ?");
pstmt.setString(1, "Hi");
for (int i = 0; i < 10; i++) {
  pstmt.setInt(2, i);
  int rowCount = pstmt.executeUpdate();
}
pstmt.setInt(2, NULL) -- sets value to null
```


Callable Statement

Subclass of Statement, allows SQL stored procedures/functions to be invoked.

```
CallableStatement cs1 = conn.prepareCall
                       ( "{call proc (?,?)}" ) ;
CallableStatement cs2 = conn.prepareCall
                       ( "{? = call func (?,?)}" ) ;
```

JDBC at Client and Server Side

- JDBC at client: Java class library for communicating with server
- JDBC server: Invoke database with SQL, send results back to client
- Inbetween: Network protocol
  - Not part of JDBC standard)
  - ⇒ need matching clientside/serverside libraries
  - ⇒ link to appropriate library depending on server (interface same, library implementation may differ)
  - Example: Oracle JDBC Thin driver communicates with Oracle using Sockets

JDBC Drivers

- Types of drivers
  1. JDBC-ODBC bridge plus ODBC driver
     JDBC client code communicates with ODBC server, which communicates with database
  2. Native-API partly-Java driver
     Part of JDBC driver in native language like C/C++. Not useful for applet/servlet.
  3. JDBC-Net pure Java driver
     Database independent protocol
  4. Native-protocol pure Java driver
     Java code communicates with database proprietary protocol.
- E.g., Oracle Thin driver is Type 4
- Drivers must be registered with system. Appropriate driver chosen by system when connection is opened.

JDBC URLs

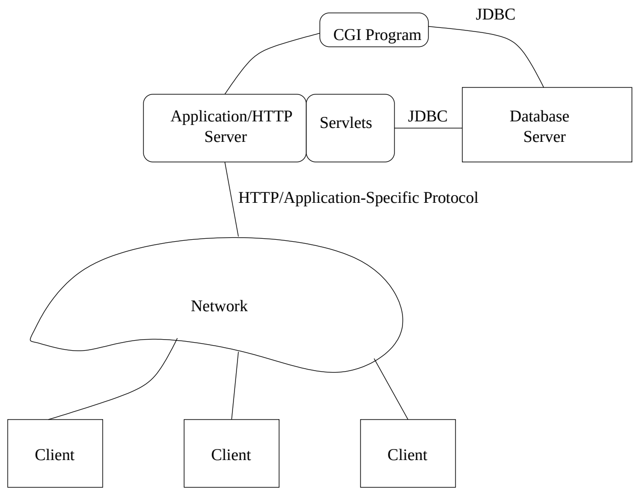
- Three parts, separated by colons: jdbc:¡subprotocol¿:¡subname¿
- Eg. `jdbc:odbc:fred` uses odbc bridge, and odbc style subname
- Eg. `jdbc:oracle:thin:@everest:1521:GEN`
  - Subname `thin` says use Oracle's ODBC Thin driver
  - Subname `@everest` specifies machine name (everest.cse.iitb.ernet.in)
  - `1521` is the port number
  - `GEN` is the name of Oracle database to be used.

Two-tier Model

Java+JDBC at client + backend server

- Benefits:
  - flexible, need not be restricted to predefined queries
- Problems:
  - Security: passwords available at client site, any database operation possible
  - More code shipped to client
  - Not appropriate across organizations, or large ones like universities

Three Tier Model



Three-tier Model (Cont.)

- E.g. Web client + Java Servlet
- Client sends request over http or application-specific protocol
- Application or Web server receives request
- Request handled by cgi program or servlet
- Security handled by application at server
  - Better security
  - Fine granularity security
- Simple client, but only packaged transactions

JDBC vs. ODBC

- ODBC for Java (ODBC is based on C)
- Improves on ODBC: cleaner, easier to learn
- JDBC-ODBC bridge:
  - translates JDBC calls to ODBC calls,
  - can use standard ODBC support at backend

Other Features

- Support for large object access via streams
  - streams are like files
  - can open a column of a result row as a stream and read bytes from it
- Dynamic Database Access
  - Can be used to query database schema
  - Look up java.sql.DatabaseMetaData and java.sql.ResultSetMetaData for more information
- Transaction support

```
connection.setAutoCommit(false);
connection.commit();  or  connection.rollback();
```

JDBC 2.0 Extensions

- Advanced data types inluding SQL3 data types in database
  - Reference types, structured types
  - Inheritance
- Direct storage of Java objects (using automatic serialization)
- Scrollable and updatable result sets
- Rowsets and disconnected operation
- Batching of inserts and updates (reduce calls on server)
  - Add multiple queries to a batch in a statement, then execute the statement
  - Oracle provides its own non-standard extensions

For More Information

- http://java.sun.com/products/jdk/1.2/docs/guide/jdbc/
  - JDBC Technology Guide: getting started
  - JDBC 1.2 and 2.0 specifications in html/ps/pdf
  - JDBC SDK including JDBC-ODBC bridge
- Oracle online manuals, JDBC section (comes with Oracle 8)
- Plenty of books