

CS 580 Client-Server Programming
Spring Semester, 2006
Doc 10 SQL & Programs
Feb 23, 2006

Copyright ©, All rights reserved. 2006 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA. OpenContent (<http://www.opencontent.org/opl.shtml>) license defines the copyright on this document.

References

<http://java.sun.com/j2se/1.5/docs/guide/jdbc/index.html> Sun's on-line JDBC Tutorial & Documentation

Client/Server Programming with Java and CORBA, Orfali and Harkey, John Wiley and Sons, Inc. 1997

Connector-J, MySql JDBC Driver Documentation, <http://www.mysql.com/products/connector/j/>

PostgreSQL JDBC Documentation, <http://jdbc.postgresql.org/documentation/docs.html>

Ruby PostgreSQL, <http://ruby.scripting.ca/postgres/rdoc/>

Few More SQL Commands

```
mysql> ALTER TABLE students ADD column foo CHAR(40);
```

```
Query OK, 1 row affected (0.03 sec)
```

```
Records: 1 Duplicates: 0 Warnings: 0
```

```
mysql> DROP TABLE students;
```

```
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> DROP DATABASE lectureexamples;
```

```
Query OK, 0 rows affected (0.00 sec)
```

Documentation

MySQL

<http://dev.mysql.com/doc/>

PostgreSQL

<http://www.postgresql.org/docs/>

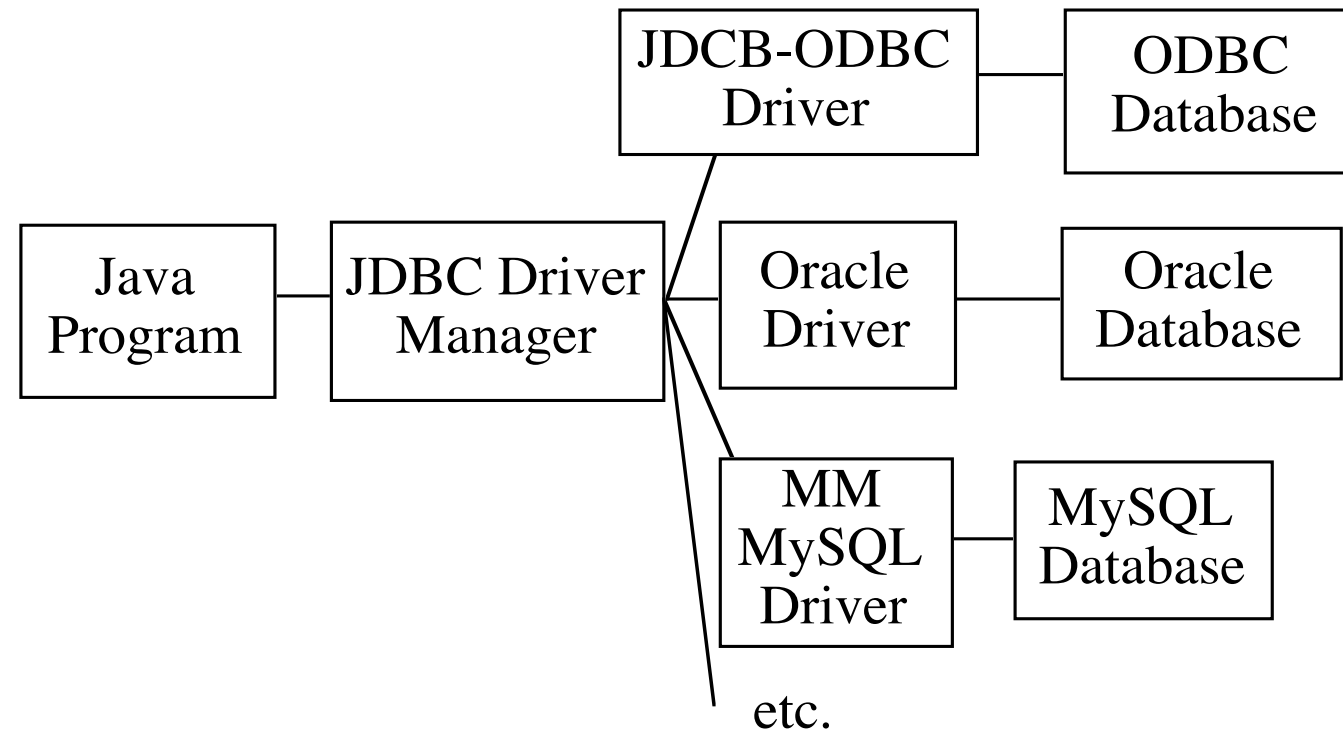
Java – Connecting To Database

```
import java.sql.*;

public class SampleConnection
{
    public static void main (String args[]) throws Exception
    {
        String dbUrl = "jdbc:mysql://rugby.sdsu.edu:8777/test";
        String user = "whitney";
        String password = "mylittleSecret";
        System.out.println("Load Driver!");

        Class.forName("com.mysql.jdbc.Driver");
        Connection rugby;
        rugby = DriverManager.getConnection( dbUrl, user, password);
        Statement getTables = rugby.createStatement();
        ResultSet tableList =
            getTables.executeQuery("SELECT * FROM name");
        while (tableList.next() )
            System.out.println("Last Name: " + tableList.getString(1) + '\t' +
                "First Name: " + tableList.getString( "first_name"));
        rugby.close();
    }
}
```

JDBC



MySQL jdbc driver

<http://dev.mysql.com/downloads/connector/j/3.0.html>

PostgreSQL jdbc driver

<http://jdbc.postgresql.org/index.html>

Course accounts use PostgreSQL 7.4

Drivers must be in your classpath

JDBC Drivers

Java supports four types of JDBC drivers

JDBC-ODBC bridge plus ODBC driver

Java code access ODBC native binary drivers

ODBC driver accesses databases

ODBC drivers must be installed on each client

Native-API partly-Java driver

Java code accesses database specific native binary drivers

JDBC-Net pure Java driver

Java code accesses database via DBMS-independent net protocol

Native-protocol pure Java driver

Java code accesses database via DBMS-specific net protocol

JDBC URL Structure

`jdbc:<subprotocol>:<subname>`

`<subprotocol>`

Name of the driver or database connectivity mechanism

`<subname>`

Depends on the `<subprotocol>`, can vary with vender

PostgreSQL

`jdbc:postgresql:database`

`jdbc:postgresql://host/database`

`jdbc:postgresql://host:port/database`

MySQL

`jdbc:mysql://[host][,failoverhost...][:port]/[database]`

`[?propertyName1][=propertyValue1][&propertyName2][=propertyValue2]...`

Loading Driver

In your code

```
Class.forName("com.mysql.jdbc.Driver");
```

Command line

```
java -Djdbc.drivers=org.postgresql.Driver yourProgramName
```

DriverManager.getConnection - Using JDBC URL

Three forms:

```
getConnection(URL, Properties)
getConnection(URL, userName, Password)
getConnection(URLWithUsernamePassword)
```

Form 1

```
static String ARS_URL = "jdbc:oracle:@PutDatabaseNameHere";

DriverManager.getConnection(ARS_URL, "whitney","secret");
```

Form 2

```
DriverManager.getConnection(
    "jdbc:oracle:whitney/secret@PutDatabaseNameHere");
```

Form 3

```
java.util.Properties info = new java.util.Properties();
info.addProperty ("user", "whitney");
info.addProperty ("password","secret");

DriverManager.getConnection (ARS_URL ,info );
```

java.sql verses javax.sql

java.sql
DriverManager

javax.sql
DataSource
 Connection Pools
 Distributed Transactions
 Requires JNDI

Queries

`executeUpdate`

Use for INSERT, UPDATE, DELETE or SQL that return nothing

`executeQuery`

Use for SQL (SELECT) that return a result set

`execute`

Use for SQL that return multiple result sets

Uncommon

ResultSet

ResultSet - Result of a Query

JDBC returns a ResultSet as a result of a query

A ResultSet contains all the rows and columns that satisfy the SQL statement

A cursor is maintained to the current row of the data

The cursor is valid until the ResultSet object or its Statement object is closed

next() method advances the cursor to the next row

You can access columns of the current row by index or name

ResultSet has getXXX methods that:

- have either a column name or column index as argument

- return the data in that column converted to type XXX

getObject

A replacement for the getXXX methods

Rather than

```
ResultSet tableList =  
    getTables.executeQuery("SELECT * FROM name");  
String firstName = tableList.getString( 1);
```

Can use

```
ResultSet tableList =  
    getTables.executeQuery("SELECT * FROM name");  
String firstName = (String) tableList.getObject( 1);
```

getObject(int k) returns the object in the k'th column of the current row

getObject(String columnName) returns the object in the named column

Data Conversion

SQL type	Java type
CHAR	String
VARCHAR	String
LONGVARCHAR	String
NUMERIC	java.math.BigDecimal
DECIMAL	java.math.BigDecimal
BIT	boolean
TINYINT	byte
SMALLINT	short
INTEGER	int
BIGINT	long
REAL	float
FLOAT	double
DOUBLE	double
BINARY	byte[]
VARBINARY	byte[]
LONGVARBINARY	byte[]
DATE	java.sql.Date
TIME	java.sql.Time
TIMESTAMP	java.sql.Timestamp

Some Result Set Issues

What happens when we call next() too many times?

What happens if we try to access data before we call next?

In both cases an `java.sql.SQLException` is thrown

Mixing ResultSets

Can't have two active result sets on same statement

```
Connection rugby;
rugby = DriverManager.getConnection( dbName, user, password);
Statement getTables = rugby.createStatement();
ResultSet count =
    getTables.executeQuery("SELECT COUNT(*) FROM name");
ResultSet tableList =
    getTables.executeQuery("SELECT * FROM name");

while (tableList.next() )
    System.out.println("Last Name: " + tableList.getObject(1) + "\t" +
        "First Name: " + tableList.getObject( "first_name"));

// Raises java.sql.SQLException
count.getObject(1);

rugby.close();
```

this can happen when two threads have access to the same statement

Two Statements on one Connection work

```
Connection rugby;
rugby = DriverManager.getConnection( dbUrl, user, password);
Statement getTables = rugby.createStatement();
Statement tableSize = rugby.createStatement();

ResultSet count =
    getTables.executeQuery("SELECT COUNT(*) FROM name");
ResultSet tableList =
    tableSize.executeQuery("SELECT * FROM name");

while (tableList.next() )
    System.out.println("Last Name: " + tableList.getObject(1) + '\t' +
        "First Name: " + tableList.getObject( "first_name"));

count.next();
System.out.println("Count: " + count.getObject(1) );
count.close();
tableList.close();
rugby.close();
```

Threads & Connections

Some JDBC drivers are not thread safe

If two threads access the same connection results may get mixed up

PostgreSQL & MySQL drivers are thread safe

When two threads make a request on the same connection

The second thread blocks until the first thread get it its results

Can use more than one connection but

Each connection requires a process on the database

Ruby MySQL

Documentation & Directions

<http://www.kitebird.com/articles/ruby-mysql.html>

Ruby PostgreSQL

Install

```
gem install ruby-postgres --rdoc
```

Docs

<http://ruby.scripting.ca/postgres/rdoc/>

Examples (Unix)

```
/usr/lib/ruby/gems/1.8/gems/ruby-postgres-0.7.1.2005.12.21/sample
```

Ruby PostgreSQL Example

```
require "postgres"

cs580 = PGconn.connect('bismarck.sdsu.edu',5432, nil, nil, 'cs580whitney', 'cs580whitney', 'password')

cs580.exec("DROP TABLE test") rescue nil
cs580.exec("CREATE TABLE test (first_name VARCHAR(20), last_name VARCHAR(20))")
cs580.exec("INSERT INTO test VALUES ('Roger', 'Whitney')")
cs580.exec("INSERT INTO test VALUES ('Roger', 'Rabbit')")

result = cs580.exec("SELECT * FROM test")
for field in result.fields
  printf("%-15s",field)
end
printf("\n")
result.result.each do |tuple|
  tuple.each do |fld|
    printf("%-15s",fld)
  end
  printf("\n")
end
```