ACCESS TO AN ORACLE DATABASE USING JDBC

ANTAL Tiberiu Alexandru

Abstract. The paper presents how to access a Oracle instance from Sun's NetBeans IDE 3.5.1 using JDBC.

1. About ORACLE

Oracle is the world's leading supplier of information management software. Oracle is best known for its DataBase Management Systems (a suite of programs which typically manage large structured sets of persistent data, offering ad hoc query facilities to many users) vendor and relational DBMS products. A RDBMS (Relational DataBase Management System) is a database based on the relational model developed by E.F. Codd. A relational database allows the definition of data structures, storage and retrieval operations, and integrity constraints. In such a database, the data and relations between them are organized in tables. A table is a collection of records and each record in a table contains the same fields. Certain fields may be designated as keys, which means that searches for specific values of that field will use indexing to speed them up. Records in different tables may be linked if they have the same value in one particular field in each table. Today, Oracle develops and markets Oracle9i Database Server, Oracle9i Application Server and the Oracle9i family of software products for database management and developer tools. Oracle software runs on IBM PCs, workstations, minicomputers, mainframes and massively parallel computers.

2. About Java

The Java programming language, named after the Indonesian island, started as a simple, object-oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, multithreaded, dynamic, general-purpose programming language developed by Sun Microsystems in the 90'. Java supports programming for the Internet in the form of platform-independent Java "applets". Java is similar to C++ without operator overloading (though it does have method overloading), without multiple inheritance, and extensive automatic coercions. It has automatic garbage collection. Java programs can run stand-alone on small computers. Java extends C++'s object-oriented facilities with those of Objective C for dynamic method resolution. Java has an extensive library of routines for

TCP/IP protocols like HTTP and FTP. Java applications can access objects across the Internet via URLs as easily as on the local file system. The Java compiler and linker both enforce strong type checking - procedures must be explicitly typed. Java supports the creation of virus-free, tamper-free systems with authentication based on public-key encryption. The Java compiler generates an architecture-neutral object file executable on any processor supporting the Java runtime system. The object code consists of bytecode instructions designed to be both easy to interpret on any machine and easily translated into native machine code at load time.

4. ODBC

ODBC stands for Open DataBase Connectivity, and it's a standard for accessing different database systems. There are interfaces for Visual Basic, Visual C++, SQL and the ODBC driver pack contains drivers for the Access, Paradox, dBase, Text, Excel and Btrieve databases. An application can submit statements to ODBC using the ODBC dialect of SQL. ODBC then translates these to whatever dialect the database understands. ODBC 1.0 was released in September 1992. ODBC is based on Call-Level Interface and was defined by the SQL Access Group. Microsoft was one member of the group and was the first company to release a commercial product based on its work (under Microsoft Windows) but ODBC is not a Microsoft standard (as many people believe). ODBC drivers and development tools are available now for Microsoft Windows, Unix, OS/2, and Macintosh.

4. JDBC

Java DataBase Connectivity is one of the most robust and mature approaches to access a RDBMS (Relational DataBase Management System) from Java. It was modeled after ODBC and defines the a package of API for database access that supports SQL functionality and enables access to a wide range of RDBMS. With JDBC, Java can be used as host language for writing database applications. API (Application Programming Interface) define the interface by which an application program accesses services. An API is defined at source code level and provides a level of abstraction between the application and the services to ensure the portability of the code. The JDBC API consists mainly of two interfaces: an API for application writers and a low level driver. Applications can access databases using ODBC drivers, database client drivers or pure Java JDBC drivers. The following options are available:

- *The JDBC-ODBC bridge*: it was developed by Sun and Intersolv to provide JDBC access using ODBC drivers in 1996. Here, ODBC is between the JDBC driver and the vendor's client libraries. In most cases the ODBC client software must be loaded on each client machine that uses this driver. This approach has some performance overheads due to the translation between JDBC and ODBC and it's not appropriate for large-scale applications, it is however a decent solution for machines connected on the Internet;
- *The partial JDBC driver*: this converts JDBC calls into calls on the client API for the DBMS. The communication is direct with the database, but still requires some software to be loaded on each client machine. It is a simple solution for machines connected to the Internet but it also offers a possible solution for intranet applications;
- *The pure Java JDBC driver for database middleware*: this driver translates JDBC calls intro middleware vendor's protocol, this will subsequently translate to a DBMS protocol by a middleware server. Probably, this is the most flexible JDBC alternative, but it requires the middleware.

- *The pure Java JDBC driver with a direct database connection*: this driver converts JDBC calls into network protocol used by the DBMS. This way a client call goes directly to the DBMS server. These drivers cam be downloaded dynamically from the Internet and are implemented in Java to achieve the platform independence. However, for this approach the developer requires a different driver for each database. Since many of the protocols are proprietary, the database vendors themselves (on their Web sites) are the primary source.

5. JDBC and ORACLE

Oracle provides three types of JDBC drivers to access and manipulate databases from Java: thin, fat and server-side. The chosen driver depends on the placement of the application that is built.

- The *thin JDBC driver* is written completely in Java, is small, and because of that, ideal for Web based applications. This driver is using a TCP/IP port on the client side to connect to SQL *Net on the server-side. SQL *net doesn't have to be loaded on the machine that is using this driver;
- The *fat JDBC driver* also called JDBC/OCI (Oracle Call Interface) uses Oracle's client library to access the remote or local database thought SQL *Net. SQL *Net must be installed on the machine that uses the fat driver. This driver is faster than the first one as it calls the database using native database call that use OCI.
- The *server-side JDBC driver* can be used only from inside a database and it connects to a local database. As Oracle can store Java procedures this driver is used for server-side application stored in database.

5.1 Portability issues

All three drivers implement the same JDBC API . This gives a great development, deployment and real portability of Java applications. The following code, when is run from the NetBeans IDE it display the results in the Output Window. Outside form an IDE the application will run from the command prompt and will display results on the screen. To display the output to a Web page the cod will be simply dropped into a servlet or a JSP program.

5.2 Performance issues

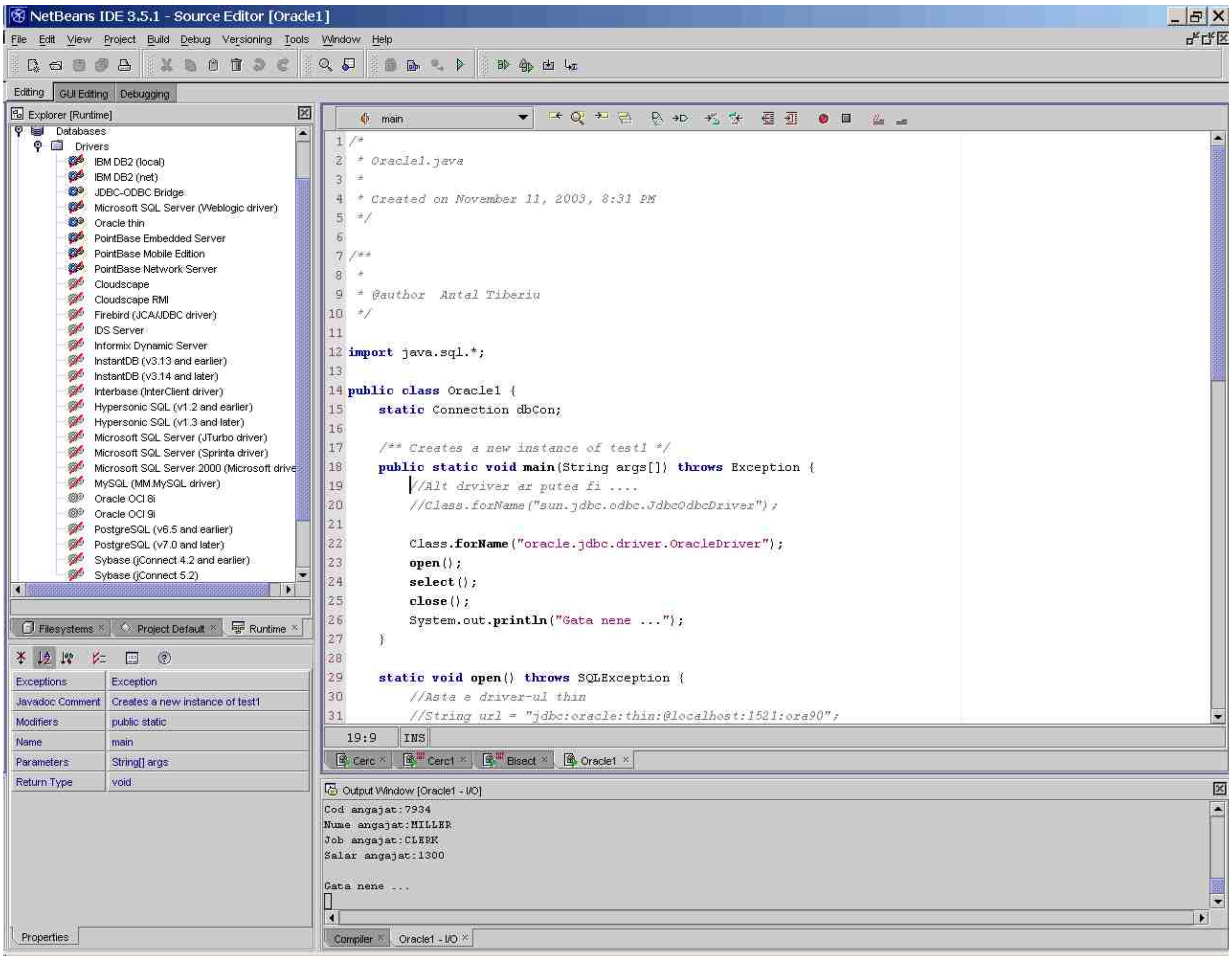
Unfortunately, the driver used depends on the application placement. If the program is located on a server that doesn't have loaded SQL*Net the thin driver must be used, otherwise the fat driver can be used. If the Java program is inside a database, that is a Java stored procedure, the server-side driver will give the best performance. The JDBC drivers perform faster in the following order: server-side (or internal) JDBC driver, fat driver (oci8 or oci9) and thin driver.

5.3 Connection issues

The fat JDBC driver is the only one that supports proxy connections. This way the application server (the middleware) can connect the Java application with a username/password to the database server on behalf of a user. This way the Java application will have granted DBA rights on the application server.

6. WHAT is the NetBeans IDE

NetBeans is an Integrated Development Environment (IDE), that can be downloaded from http://www.netbeans.org, for Java, from Sun Microsystems.


NetBeans IDE 3.5.1 - Source Editor [Oracle1]
File Edit View Project Build Debug Versioning Tools Window Help
Editing GUI Editing Debugging
Explorer [Runtime]
Filesystems Project Default Runtime
main
19:9 INS
Cerc Cerc1 Bisect Oracle1
Output Window [Oracle1 - I/O]
Properties
Compiler Oracle1 - I/O

Databases
Drivers
- IBM DB2 (local)
- IBM DB2 (net)
- JDBC-ODBC Bridge
- Microsoft SQL Server (Weblogic driver)
- Oracle thin
- PointBase Embedded Server
- PointBase Mobile Edition
- PointBase Network Server
- Cloudscape
- Cloudscape RMI
- Firebird (JCA/JDBC driver)
- IDS Server
- Informix Dynamic Server
- InstantDB (v3.13 and earlier)
- InstantDB (v3.14 and later)
- Interbase (InterClient driver)
- Hypersonic SQL (v1.2 and earlier)
- Hypersonic SQL (v1.3 and later)
- Microsoft SQL Server (JTurbo driver)
- Microsoft SQL Server (Sprinta driver)
- Microsoft SQL Server 2000 (Microsoft drive
- MySQL (MM.MySQL driver)
- Oracle OCI 8i
- Oracle OCI 9i
- PostgreSQL (v6.5 and earlier)
- PostgreSQL (v7.0 and later)
- Sybase (jConnect 4.2 and earlier)
- Sybase (jConnect 5.2)

Exceptions	Exception
Javadoc Comment	Creates a new instance of test1
Modifiers	public static
Name	main
Parameters	String[] args
Return Type	void

```
/*
 * Oracle1.java
 *
 * Created on November 11, 2003, 8:31 PM
 */

/**
 *
 * @author  Antal Tiberiu
 */

import java.sql.*;

public class Oracle1 {
    static Connection dbCon;

    /** Creates a new instance of test1 */
    public static void main(String args[]) throws Exception {
        //Alt drviver ar putea fi ....
        //Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

        Class.forName("oracle.jdbc.driver.OracleDriver");
        open();
        select();
        close();
        System.out.println("Gata nene ...");
    }

    static void open() throws SQLException {
        //Asta e driver-ul thin
        //String url = "jdbc:oracle:thin:@localhost:1521:ora90";
```

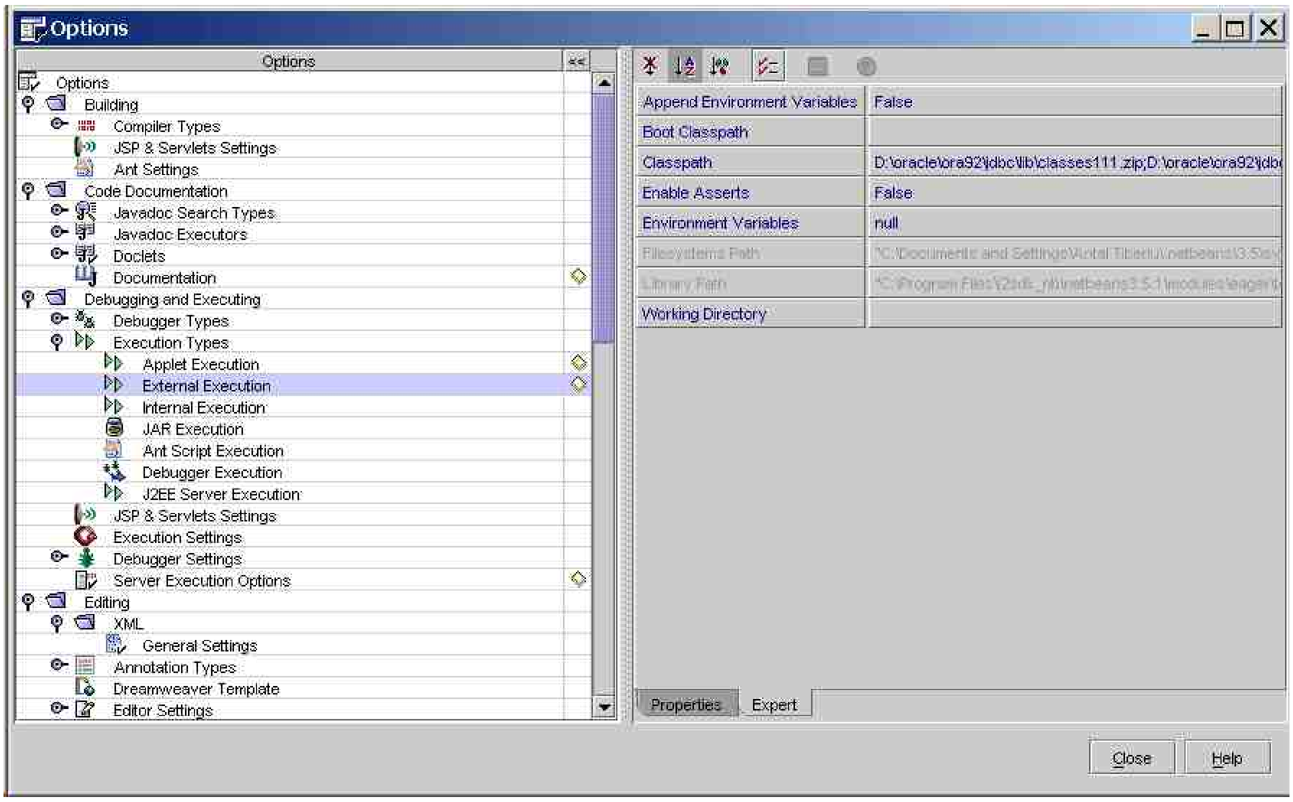
```
Cod angajat:7934
Nume angajat:MILLER
Job angajat:CLERK
Salar angajat:1300

Gata nene ...
```

Figure 1 - Some drivers used to connect Java to Oracle.

On the left side Window from *Figure 1* the available Oracle drivers can be seen. In order to have the Oracle JDBC drivers the *CLASSPATH* in the NetBeans IDE must be set by the following procedure. Start from ***Tools***, ***Options*** then make the changes to the *Classpath* property as shown in *Figure 2* (the required classes and the path for each of these are: D:\oracle\ora92\jdbc\lib\classes111.zip; D:\oracle\ora92\jdbc\lib\nls_charset11.zip. If you don't have these classes then the drivers can be downloaded from http://technet.oracle.com).

The following application connects to an Oracle 9i Database Server from Java, using the NetBeans IDE and JDBC. The machine name is **`localhost`** and the database's SID is **`ora92`**, the user name is **`scott`** and the password is **`tiger`**. The connection with the *thin JDBC driver* goes through the 1521 port. The code also shows how to use the *fat JDBC driver* (JDBC/OCI) for the same purpose and for the same database instance.



Append Environment Variables	False
Boot Classpath	
Classpath	D:\oracle\ora92\jdbc\lib\classes111.zip;D:\oracle\ora92\jdbc
Enable Asserts	False
Environment Variables	null
Filesystems Path	"C:\Documents and Settings\Antal Tiberiu\.netbeans\3.5sy
Library Path	"C:\Program Files\j2sdk_nb\netbeans3.5.1\modules\eager\
Working Directory	

Figure 2 - Setting the CLASSPATH to the Oracle JDBC drivers.

```
import java.sql.*;

public class conToOracle {
    static Connection dbCon;

    public static void main(String args[]) throws Exception {
        //Another driver could be:
        //Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

        Class.forName("oracle.jdbc.driver.OracleDriver");
        open();
        select();
        close();
        System.out.println("*** End of application. ***");
    }

    static void open() throws SQLException {
        //This would be the thin driver:
        //String url = "jdbc:oracle:thin:@localhost:1521:ora92";

        //This is the fat driver:
        String url ="jdbc:oracle:oci:@ora92";
        String user ="Scott";
        String password = "Tiger";
        dbCon = DriverManager.getConnection(url,user, password);
    }

    static void close() throws SQLException {
        dbCon.close();
    }
    static void select() throws SQLException {
        Statement stm;
        String query;
        ResultSet rs;
        boolean more;

        query = "SELECT * FROM emp";
        stm=dbCon.createStatement();
        rs=stm.executeQuery(query);

        more=rs.next();

        if (!more) {
            System.out.println("*** No records were found! ***");
            return;
        }

        while (more) {
            System.out.println("Cod angajat:" + rs.getString("EMPNO"));
            System.out.println("Nume angajat:" + rs.getString("ENAME"));
            System.out.println("Job angajat:" + rs.getString("JOB"));
            System.out.println("Salar angajat:" + rs.getString("SAL"));
            System.out.println("");
            more=rs.next();
        }

        rs.close();
        stm.close();
    }
}
```

REFERENCES

[1] Abbey, M., Corey, M. J., Abramson, I. Oracle 8i, A Beginner's Guide, Osborne, 1999, ISBN: 0-07-212204-8.

[2] Loney, K., Koch, G., Oracle 9i, The Complete Reference Osborne, 2002, ISBN:0-07-222521-1.

Accesul la ORACLE prin JDBC

Lucrarea prezintă implementarea unui aplicaţii Java pentru accesul la o instanţă Oracle prin JDBC folosind mediul de programare NetBeand 3.5.1 al firmei Sun Microsystems.

Senior lecturer dr. eng. ANTAL Tiberiu Alexandru, Technical University of Cluj-Napoca, Department of Mechanics and Computer Programming, Faculty of Machine Building, B-dul Muncii, Nr. 103-15, Cluj-Napoca, RO-3400, ROMANIA.