Making a java application MS-SQL compatible

Mindfire Solutions

www.mindfiresolutions.com

September 3, 2003

Abstract:

This paper discusses the aspects of making a java application MSSQL compliant. Based on practical experience we talk about the few differences in Oracle and MSSQL using JDBC. Also how to choose the JDBC driver suited for your job is touched upon inside.

Introduction

We are working on enchancements of a product called *Revize.* One of the features we implemented was making Revize *MS-SQL compliant*. That is, Revize which currently has an interface to Oracle and Pointbase databases, will also be able to talk to MS-SQL now. Thereby making it more potent and increasing its market reach.
In this article we give an overview of what might be few common problems faced by many developers who are trying to something like this somewhere in outside world.

Little bit about Revize

Revize is a content management system, developed completely in Java as backend. For front end JSPs have been used along with Javascript for doing some routine front end jobs like validations etc. The databases supported by Revize currently are Oracle and Pointbase. Revize runs on a Resin webserver which comes integrated with Revize installation.
It is a quite potent tool to manage websites and creating new ones. Absolutely anyone (i.e. including non technical person) can make changes to the website, once it is *Revize enabled*, without doing any coding.
For future reference in this article we need to define what is a Revize Webspace. *A Webspace is a collection of all the Revize Resources for a particular web site or part of a larger web site.* When we create a new Webspace, its equivalent to creating a new distinct space in database (for instance, a Schema in Oracle).

Choosing the JDBC Driver

After surveying we decided to go ahead with *jTDS open source jdbc driver* (http://jtds.sourceforge.net/). jTDS is TYPE 4 JDBC driver, i.e. fully Java technology-enabled driver.
It supports most of the features we required and is being continuously improved upon by the community. Also we were looking for an open source to cut cost on development and deployment, which makes jTDS an obvious choice.

Instantiating JDBC Connection

The code to load a driver and creating a connection using the jdbc api is:

```
// load the driver
try {
        DriverManager.registerDriver(
                (Driver)Class.forName(driverName).newInstance() );
```

```
} catch ( Throwable t ) {
        //throw new Error( "Error registering database driver: " + driverName );
}

// get the connection
try {
        Connection con = DriverManager.getConnection( URL, userName, password );
} catch(SQLException sqe) {
        //throw new Error( "Error getting database connection");
}
```

where

driverName = “net.sourceforge.jtds.jdbc.Driver”,
URL = “jdbc:jtds:sqlserver://DatabaseServerName/DatabaseName”

Differences between Oracle and MS-SQL

We found the following differences between Oracle and MS-Sql due to which we had to handle MS-SQL seperately:

1) Oracle has a concept of *Schema* where as in MSSQL its counter part is a *Database*.

2) To create a new webspace in Oracle we execute “Create SCHEMA webspacename”. This creates a new Schema in Oracle and hence the objects related to this schema can be referenced as *schema_name.object_name*
If we need to imitate same thing in MS-SQL we can either make a new database each time, which is inefficient, or just create a new *Role* in the existing database and use it to divide a single database into multiple workspaces, which sounds better. So any item created in relation to a role can be referenced as *role_name.object_name* inside that particular database. For this, in MSSQL, we need to call a system defined stored procedure “call sp_addrole(webspacename)”.

3) Likewise to delete a webspace in Oracle we do “Drop Schema webspacename CASCADE”. This deletes every dependent object for this schema on its own.
But in MSSQL we need to take care of this cascading effect on our own. So we need to delete any items tables/stored procedures/index/etc, related to that particular role, before dropping the role by “call sp_droprole(webspacename)”.

4) Point to be noted before using sp_addrole and sp_droprole is that these two always needs *autocommit* property of connection to be set to *TRUE*.

5) Few datatypes are different in Oracle and MS-SQL. The mapping from Oracle to MSSQL that we needed was:
    a) Date -> datetime

b) Long Raw -> image
c) Clob -> text/image
d) Number(1) -> bit

Differences between MS-SQL2000 and MS-SQL7.0

When trying to run the same code on MS-SQL7.0 we found a few discrepancies there too:

1) "ON DELETE CASCADE" option used while creating a table doesn't really work when deleting a table in MS-SQL7.0. Hence we need to delete all dependant tables manually before deleting a primary table

2) *autocommit* set to *true* for a *connection* object doesn't work as expected in MS-SQL7.0, so we need to do a *connection.commit()* explicitly after every execution of a *statement*. This creates the problem of handling rollbacks in case of any failures. This can be resolved by making stored procedures, which are run within transactions, and then can be called using JDBC from java code.

Testing

Discussion

Conclusion

Writing a good article is as painful as reading a bad article.

To know more about Mindfire Solutions, please visit us on www.mindfiresolutions.com

---

Mindfire Solutions is an IT and software services company in India, providing expert capabilities to the global market. Mindfire possesses experience in multiple platforms, and has built a strong track record of delivery. Our continued focus on fundamental strengths in computer science has led to a unique technology-led position in software services.

To explore the potential of working with Mindfire, please drop us an email at info@mindfiresolutions.com. We will be glad to talk with you.

To know more about Mindfire Solutions, please visit us on www.mindfiresolutions.com

---