Idaho National Laboratory

Interfacing Fedora with Microsoft SQL Server

David Handy - Developer
Wayne Simpson - Software Architect/Engineer
Technology Enhancement and Software Services
Idaho National Laboratory

1 - Introduction

The Idaho National Laboratory Research Library is in the process of digitizing a large portion of their collections. As part of this effort, a little more than fifty years worth of reports written by INL's researchers are being scanned and converted to PDF format. Up until this point, these documents have never been available in anything but hard copy. In order to facilitate access to these digitized documents, we have employed the use of Fedora – the Flexible Extensible Digital Object Repository Architecture.[1]

Fedora, by default, can work with a number of database engines, including MySQL, Oracle, and PostgreSQL, among others. For various reasons, it was decided that the best course of action was to use Microsoft® SQL Server. Fedora does not support SQL Server, so this functionality needed to be added.

Fedora's documentation suggests two possible methods for successfully using a database other than those Fedora is natively set-up to use.[2] The first method is to manually create a database. Whoever implements this method must add the tables, columns, keys, and so forth on their own. The second method is to write code that will convert Fedora's Data Description Language (DDL) definition of the database to SQL that is specific to the target database engine.

After writing an implementation of each method, it was concluded that the DDL Converter is the better of the two. It is more versatile and only needs to be written once. This paper contains details about how each method was implemented as well as conclusions drawn from the process.

2 - Implementation Details

The source distribution of Fedora as well as a proper JDBC driver are required for either of the two solutions to work. This implementation uses the JTDS driver.[3] For the manual option, this driver must be placed in the application server as specified in the documentation (see footnote 2). For the DDL Converter option, the driver can be placed in the *lib* directory of the source code.

[1] To find more information about Fedora, see www.fedora-commons.org
[2] Installation and Configuration Guide. Fedora Commons. http://fedora-commons.org/confluence/display/FCR30/Installation+and+Configuration+Guide
[3] Available at jtds.sourceforge.net

2.1 – Manual Option

The manual option was deemed easier of the two alternatives and by using the SQL Server's user interface, a database was manually created which followed the specification in DefaultDOManager.dbspec.[4]

In order to make reproduction of the database easy, the database schema was exported as a SQL script, which is included in this document as Appendix 1. It was made for the Fedora 3.0 Beta 1 release.

Of note, the process of creating database schemas is beyond the scope of this document. Readers who are unfamiliar with database management are encouraged to consult documentation describing the basics of database creation.[5]

To continue, Fedora's database schema can change from version to version; therefore, this exercise may need to be repeated each time one desires to upgrade to a newer version.

2.2 –DDL Converter

Since a given implementation of the manual option only works for the version of Fedora for which it was created, implementing the Data Description Language (DDL) Converter option was more desirable. This allows Fedora to programmatically create and alter the database schema, which means less extra work when upgrading to newer versions.

The first step was examining the java source for the MySQL DDL Converter.[6] This code was used as a template for the new class called MsSQLDDLConverter.java. It needed to be edited to conform to SQL Server's version of schema creation syntax, since all SQL database engines handle the creation of tables differently. Required changes included the syntax of specifying binary data, primary and foreign keys and other constraints.

The source code for the DDL Converter is included in this document as Appendix 2.

3 – Conclusion

The DDL Converter is the better option in the long run. Although it requires more thought, it only needs to be done once.

Creating a DDL converter is not, however, the only step in the process. For both options, there is still a significant amount of configuration that has to be done manually in the server's configuration file. Sufficient time has not been invested yet into adding the new database engine to the Fedora installer, which would automatically set up the database the way it does with the native database engines. Currently, Fedora must be installed with a database it recognizes such as MySQL. The person installing must then go and point Fedora to the new database by editing fedora.fcfg. Further work is needed to add this option to the automatic installation process and remove the hassle. One would hope that doing so would involve only a slight modification of the source code. For now, though, the work done is counted as a success and further development in this venture will be continued.

---

[4] The full path in the source distribution is src/dbspec/server/fedora/server/storage/resources/DefaultDOManager.dbspec

[5] See Microsoft's Books Online at http://msdn.microsoft.com/en-us/library/bb418471%28SQL.10%29.aspx

[6] This can be found in /src/java/fedora/server/utilities/MySQLDDLConverter.java.

Appendix 1

SQL Script for Manually Creating SQL Server Database

The following script will add all necessary tables to a database. It follows the schema outlined in the Fedora 3.0 Beta 1 release.

```
if exists (select * from dbo.sysobjects where id =
object_id(N'[fedoraAdmin].[datastreamPaths]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [fedoraAdmin].[datastreamPaths]
GO

if exists (select * from dbo.sysobjects where id =
object_id(N'[fedoraAdmin].[dcDates]') and OBJECTPROPERTY(id, N'IsUserTable')
= 1)
drop table [fedoraAdmin].[dcDates]
GO

if exists (select * from dbo.sysobjects where id =
object_id(N'[fedoraAdmin].[doFields]') and OBJECTPROPERTY(id, N'IsUserTable')
= 1)
drop table [fedoraAdmin].[doFields]
GO

if exists (select * from dbo.sysobjects where id =
object_id(N'[fedoraAdmin].[doRegistry]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [fedoraAdmin].[doRegistry]
GO

if exists (select * from dbo.sysobjects where id =
object_id(N'[fedoraAdmin].[modelDeploymentMap]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [fedoraAdmin].[modelDeploymentMap]
GO

if exists (select * from dbo.sysobjects where id =
object_id(N'[fedoraAdmin].[objectPaths]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [fedoraAdmin].[objectPaths]
GO

if exists (select * from dbo.sysobjects where id =
object_id(N'[fedoraAdmin].[pidGen]') and OBJECTPROPERTY(id, N'IsUserTable') =
1)
drop table [fedoraAdmin].[pidGen]
GO

CREATE TABLE [fedoraAdmin].[datastreamPaths] (
      [tokenDbID] [int] IDENTITY (1, 1) NOT NULL ,
      [token] [varchar] (199) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
      [path] [varchar] (255) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL
) ON [PRIMARY]
```

```
GO

CREATE TABLE [fedoraAdmin].[dcDates] (
      [pid] [varchar] (64) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
      [dcDate] [bigint] NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [fedoraAdmin].[doFields] (
      [pid] [varchar] (64) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
      [label] [varchar] (255) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
      [state] [varchar] (1) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
      [ownerId] [varchar] (64) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
      [cDate] [bigint] NOT NULL ,
      [mDate] [bigint] NOT NULL ,
      [dcmDate] [bigint] NULL ,
      [dcTitle] [text] COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
      [dcCreator] [text] COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
      [dcSubject] [text] COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
      [dcDescription] [text] COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
      [dcPublisher] [text] COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
      [dcContributor] [text] COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
      [dcDate] [text] COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
      [dcType] [text] COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
      [dcFormat] [text] COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
      [dcIdentifier] [text] COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
      [dcSource] [text] COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
      [dcLanguage] [text] COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
      [dcRelation] [text] COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
      [dcCoverage] [text] COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
      [dcRights] [text] COLLATE SQL_Latin1_General_CP1_CI_AS NULL
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

CREATE TABLE [fedoraAdmin].[doRegistry] (
      [doPID] [varchar] (64) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
      [systemVersion] [smallint] NOT NULL ,
      [ownerId] [varchar] (64) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
      [objectState] [varchar] (1) COLLATE SQL_Latin1_General_CP1_CI_AS NOT
NULL ,
      [label] [varchar] (255) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
) ON [PRIMARY]
GO

CREATE TABLE [fedoraAdmin].[modelDeploymentMap] (
      [cModel] [varchar] (64) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
      [sDef] [varchar] (64) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
      [sDep] [varchar] (64) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [fedoraAdmin].[objectPaths] (
      [tokenDbID] [int] IDENTITY (1, 1) NOT NULL ,
      [token] [varchar] (64) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
      [path] [varchar] (255) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL
) ON [PRIMARY]
GO
```

```
CREATE TABLE [fedoraAdmin].[pidGen] (
      [namespace] [varchar] (255) COLLATE SQL_Latin1_General_CP1_CI_AS NOT
NULL ,
      [highestID] [int] NOT NULL
) ON [PRIMARY]
GO

ALTER TABLE [fedoraAdmin].[datastreamPaths] WITH NOCHECK ADD
      CONSTRAINT [PK_datastreamPaths] PRIMARY KEY  CLUSTERED
      (
            [tokenDbID]
      )  ON [PRIMARY]
GO

ALTER TABLE [fedoraAdmin].[doRegistry] WITH NOCHECK ADD
      CONSTRAINT [PK_doRegistry] PRIMARY KEY  CLUSTERED
      (
            [doPID]
      )  ON [PRIMARY]
GO

ALTER TABLE [fedoraAdmin].[objectPaths] WITH NOCHECK ADD
      CONSTRAINT [PK_objectPaths] PRIMARY KEY  CLUSTERED
      (
            [tokenDbID]
      )  ON [PRIMARY]
GO

ALTER TABLE [fedoraAdmin].[datastreamPaths] ADD
      CONSTRAINT [DF__datastrea__token__7C8480AE] DEFAULT ('') FOR [token],
      CONSTRAINT [DF__datastream__path__7D78A4E7] DEFAULT ('') FOR [path],
      CONSTRAINT [DF_datastreamPaths_token] UNIQUE  NONCLUSTERED
      (
            [token]
      )  ON [PRIMARY]
GO

 CREATE  INDEX [IX_dcDates_pid] ON [fedoraAdmin].[dcDates]([pid]) ON
[PRIMARY]
GO

ALTER TABLE [fedoraAdmin].[doFields] ADD
      CONSTRAINT [DF__doFields__state__07020F21] DEFAULT ('A') FOR [state]
GO

 CREATE  INDEX [IX_doFields_pid] ON [fedoraAdmin].[doFields]([pid]) ON
[PRIMARY]
GO

ALTER TABLE [fedoraAdmin].[doRegistry] ADD
      CONSTRAINT [DF__doRegistr__syste__014935CB] DEFAULT ('0') FOR
[systemVersion],
      CONSTRAINT [DF__doRegistr__objec__023D5A04] DEFAULT ('A') FOR
[objectState],
      CONSTRAINT [DF__doRegistr__label__03317E3D] DEFAULT ('') FOR [label]
GO
```

```
ALTER TABLE [fedoraAdmin].[objectPaths] ADD
      CONSTRAINT [DF__objectPat__token__77BFCB91] DEFAULT ('') FOR [token],
      CONSTRAINT [DF__objectPath__path__78B3EFCA] DEFAULT ('') FOR [path],
      CONSTRAINT [DF_objectPaths_token] UNIQUE  NONCLUSTERED
      (
            [token]
      )  ON [PRIMARY]
GO
```

Appendix 2

Source Code for MsSQLDDLConverter

```
package fedora.server.utilities;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

/**
 * A DDLConverter for MS SQL Server.
 *
 * @author David Handy
 */
public class MsSQLDDLConverter implements DDLConverter
{

    public MsSQLDDLConverter()
    {
    }

    public boolean supportsTableType()
    {
        return true;
    }

    public String getDropDDL(String command)
    {
        String[] parts = command.split(" ");
        String tableName = parts[2];
        return "DROP TABLE " + tableName;
    }

    public List<String> getDDL(TableSpec spec)
    {
        StringBuffer out = new StringBuffer();
        StringBuffer end = new StringBuffer();
        StringBuffer indexes = new StringBuffer();
        out.append("CREATE TABLE " + spec.getName() + " (\n");
        Iterator<ColumnSpec> csi = spec.columnSpecIterator();
        int csNum = 0;
        while (csi.hasNext()) {
            if (csNum > 0) {
                out.append(",\n");
            }
            csNum++;
            ColumnSpec cs = csi.next();
            out.append("  ");
            out.append(cs.getName());
            out.append(' ');
            if (cs.getType().equalsIgnoreCase("varchar")) {
```

```
    if (cs.getBinary()) {
        out.append("BINARY");
    } else {
        out.append(cs.getType());
    }
}
else
{
    out.append(cs.getType());
}
if (cs.isNotNull()) {
    out.append(" NOT NULL");
}
if (cs.isAutoIncremented()) {
    out.append(" IDENTITY (1, 1)");
}
if (cs.getDefaultValue() != null) {
    out.append(" DEFAULT '");
    out.append(cs.getDefaultValue());
    out.append("'");
}
if (cs.isUnique())
{
    if (!end.toString().equals(""))
    {
        end.append(",\n");
    }
    end.append(" CONSTRAINT");
    end.append(cs.getName());
    end.append("_unique UNIQUE KEY NONCLUSTERED (");
    end.append(cs.getName());
    end.append(" ON PRIMARY)");
}
if (cs.getIndexName() != null)
{
    indexes.append(" CREATE INDEX ");
    indexes.append(cs.getIndexName());
    indexes.append("ON ");
    indexes.append(spec.getName());
    indexes.append(" (");
    indexes.append(cs.getName());
    indexes.append(") ON PRIMARY GO");
}
if (cs.getForeignTableName() != null)
{
    if (!end.toString().equals(""))
    {
        end.append(",\n");
    }
    end.append(" CONSTRAINT ");
    end.append(cs.getName());
    end.append("_fk FOREIGN KEY (");
    end.append(cs.getName());
    end.append(") REFERENCES ");
    end.append(cs.getForeignTableName());
    end.append(" (");
    end.append(cs.getForeignColumnName());
```

```
                end.append(")");
                if (cs.getOnDeleteAction() != null) {
                    end.append(" ON DELETE ");
                    end.append(cs.getOnDeleteAction());
                }
            }
        }
        if (spec.getPrimaryColumnName() != null)
        {
            end.append(",\n CONSTRAINT ");
            end.append(spec.getName() + "_fk");
            end.append(" PRIMARY KEY CLUSTERED (");
            end.append(spec.getPrimaryColumnName());
            end.append(")");
        }
        out.append("\n");
        out.append(") ON PRIMARY GO");
        if (!end.toString().equals(""))
        {
            out.append("\n ALTER TABLE ");
            out.append(spec.getName());
            out.append(" ADD \n");
            out.append(end);
            out.append(" GO ");
        }
        if (!indexes.toString().equals(""))
        {
           out.append(indexes);
        }
        ArrayList<String> l = new ArrayList<String>();
        l.add(out.toString());
        return l;
    }

}
```