MySQL and Spk

Revision History

Revision 1.1 September 21, 2006 Revised by: ernst
Revised instructions for Fedora Core 5.
Revision 1.0 January 29, 2004 Revised by: afw
Initial version.

The installation, use, and management of the MySQL relational database management system is discussed.

Table of Contents

Introduction **1**
Installing MySQL on Fedora Core **1**
Installing MySQL in Production (RHEL) **2**
Post-installation Configuration **2**
Backing Up the Spk Database **5**
Installing the Spk Database Source **5**
Modify the Definition of the Database **6**

Introduction

MySQL is a relational database management system that is available under an open source GPL license when included as a part of an application that also bears an open source license. Because Spk is distributed as open souce, this powerful tool can be distributed as part of Spk at no cost to users.

Spk requires the ability to make updates to the database as *atomic transactions*.. This feature was first available in Version 4 of MySQL. An Spk installation should, therefore, install MySQL Version 4.0 or later.

This document describes how to install, configure, and manage MySQL as part of Spk.

Installing MySQL on Fedora Core

The following installation instructions are specific to RFPK and Fedora Core 5, which is installed by default on all of the developers' machines.

Installing the Database Software

The preferred way to install software on a Fedora system is with packages assembled for installation using the *RPM* package format, which can be identified by the `.rpm` filename suffix. These files are distributed on Fedora's network of update servers, and can be installed using the *yum* package management tool. *Yum* is similar to the *up2date* tool on a RHEL system.

In a terminal window:

```
yum install mysql-server mysql-devel perl-DBI perl-DBD-MySQL
```

After the set of commands listed above have been executed, MySQL will have been installed on your machine, and the MySQL server should be started. You can start MySQL by executing the following command:

```
/etc/init.d/mysqld start
/sbin/chkconfig --level 345 mysqld on
```

The second command *chkconfig* is used to tell the operating system to start the MySQL server when the computer is started in runlevel 3, 4 or 5.

Installing MySQL in Production (RHEL)

The following installation instructions are specific to RPFK and RedHat Enterprise Linux version 4 (RHEL).

Installing the Database Software

The preferred way to install software in an RHEL system is with packages assembled for installation using the *rpm* command (Redhat Package Manager). Files for *rpm*, which can be identified by the `.rpm` filename suffix, are available for MySQL at the MySQL[1] web site. The needed files, however, have already been downloaded, and are available on `whitechuck.rfpk.washington.edu`.

In a terminal window:

```
cd /tmp
scp whitechuck:/opt/download/mysql-db.tar .
tar xvf mysql-db.tar
su
cd mysql-db
rpm -Uhv *
cd /tmp
rm -rf mysql-db.tar mysql-db
```

After the set of commands listed above have been executed, MySQL should be installed on your machine and the MySQL server should be running. Furthermore, the server will start automatically each time that you boot your machine.

Post-installation Configuration

Adding the Root User

As the first step toward using your MySQL database, you must provide the MySQL root user with a password. The MySQL root user is not the same as the Linux root user, and should have a different password, because these two users have different roles and responsibilities.

In the following, assume that the password is *secret-code*. Be certain, however, that you do not actually use *secret-code* as your password, but rather provide a string of alphabetic characters, digits, and special characters that is very difficult to guess.

From a terminal window:

```
su
/usr/bin/mysqladmin -u root password secret-code
```

Creating the Databases

In this step we will create two database, *spkdb* and *spktest*. These databases will have exactly the same structure. For now, we will create them empty:

```
mysql -uroot -p
use mysql;
create database spkdb;
create database spktest;
quit;
```

Adding the Test User

For testing of software, we need a user with access rights to the *spktest* database. Assume that *hostname* is the full DNS hostname of the computer. For example, *hostname* might be *rose.rfpk.washington.edu*

```
mysql -u root -p
grant all on spktest.* to tester@localhost identified by 'tester';
grant all on spktest.* to tester@localhost.localdomain identified by 'tester';
grant all on spktest.* to tester@hostname identified by 'tester';
quit;
```

Installing the Perl Access Modules

To access any relational database from perl, the perl-DBI module must be installed. For access to a MySQL database, the perl-DBD-MySQL module must be added. The following procedure will install both of these modules on your machine (use only for Red Hat Enterprise Linux RHEL).

```
cd /tmp
scp whitechuck:/opt/download/mysql-perl.tar .
tar xvf mysql-perl.tar
su
cd mysql-perl
rpm -Uhv *
exit
cd /tmp
rm -rf mysql-perl.tar mysql-perl
```

If you are using Fedora Core, the Perl::DBI and Perl::DBD:MySQL modules were installed earlier in this document. If your system does not have the Perl::DBI or Perl::DBD:MySQL modules installed, they can be installed by using the following command:

```
yum install perl-DBI perl-DBD-MySQL
```

Installing Java

If you want to use java to access the database and you do not already have the Java Software Development Kit (SDK) installed on your machine, follow these instructions.

```
cd /tmp
scp whitechuck:/opt/download/java.tar .
tar xvf java.tar
cd java
chmod +x *bin
./*bin
```

At this point you will need to scroll down through a license agreement and respond that **yes**, you accept it.

```
su
rpm -Uhv *.rpm
mv *.zip /usr/java
cd /usr/java
unzip *doc.zip
rm *doc.zip
exit
cd /tmp
rm -rf java java.tar
```

Before you can use java, you will need to set up the *JAVA_HOME* environment variable. Add the following lines to your `~/.bash_profile` file:

```
JAVA_HOME=/usr/java/j2sdk1.4.2_03
export JAVA_HOME
```

The above assumes that the SDK that you installed was `j2sdk1.4.2_03`. If it is different, this document needs to be updated, but you can simple adjust the name to what is installed.

Finally, log out of the desk top and log back in again. This will insure that all processes descendent from your desktop will have *JAVA_HOME* set properly.

Installing the Java Access Classes

To use java to access a MySQL database, you must have the mysql-connector-java jar installed on your machine. The following instructions will accomplish this.

```
cd /tmp
scp whitechuck:/opt/download/mysql-java.tar .
tar xvf mysql-java.tar
cd mysql-java
tar xvzf *
rm *.gz
cd mysql*
su
```

```
cp *.jar $JAVA_HOME/jre/lib/ext
exit
cd /tmp
rm -rf mysql-java.tar mysql-java
```

Backing Up the Spk Database

Currently, a symbolic backup of the database is stored each night in the directory `dbserver:/usr/local/spk/db_backup`. The following script,

```
#!/bin/sh

# Remove any files in the current directory that have a filename suffix
# of .sql and are more than eight days old.
#
# Make a backup copy of the spkdb database in the current directory.

DB_BACKUP=/usr/local/spk/db_backup

cd $DB_BACKUP
find . -name '*.sql' -ctime +8 -exec rm -f {} \;

/usr/local/bin/dump_spkdb.pl

exit 0
```

which also resides in `dbserver:/usr/local/spk/db_backup`, is run nightly by the *cron.daily* process.

The configuration of *cron.daily* needed to make this happen consists of a symbolic link in the `/etc/cron.daily` directory. If this link should need to be replaced, the following commands executed on *dbserver* would accomplish it:

```
su -
cd /etc/cron.daily
ln -s /usr/local/spk/db_backup/spkdb_backup.sh spkdb_backup.sh
```

A week's worth of backups is retained online in the directory. Once a week, the entire *dbserver* is backed up to tape.

This backup solution is adequate at the current time. When the database gets larger and there is much more usage, another solution will need to be implemented, because the symbolic backup is not guaranteed to provide a consistent snapshot of transactions in progress while the backup is running.

Installing the Spk Database Source

The source for the Spk database API and administrative scripts is located in `r2/src/apps/spk/db` in the *cvs* tree. You can copy the `db` subtree to another location, if you want to.

Installing and Testing the Perl API

You will need to have the perl modules for access MySQL installed on your machine. If you have not done so, follow the instructions in the section "Installing the Perl Access Modules."

Start in the `r2/src/apps/spk/db` directory in a cvs workspace, or a copy of the subtree starting at `db`:

```
cd api/perl
perl Makefile.PL
make
make test
```

If all the tests run correctly, complete the installation this way:

```
su
make install
```

Installing and Testing the Java API

You will need to have Java and the MySQL access classes installed on your machine. If you have not done so, follow the instructions in the sections "Installing Java", and "Installing the Java Access Classes."

Start in the `r2/src/apps/spk/db` directory in a cvs workspace, or a copy of the subtree starting at `db`:

```
cd api/java
make
make docs
make test
```

The commands listed above will compile the rfpk.spk.spkdb package, generate html documentation (using javadoc) and run the test suite.

Modify the Definition of the Database

From time to time, the definition of the database (the schema of the database) must be modified in order to support new features of the Spk application. The process of making these changes must be an integral part of the processes for system test [2] and deployment [3].

Example Modification Script

You could change the database schema by using *mysql* interactively. The problem with that approach is that you might have to repeat the interaction several times, which could be error prone. It is better to place the SQL/DDL statements that you would have used interactively in a file, and then use that file as input to *mysql* in batch mode.

For example, let us suppose that you need to make the following changes:

- Add an *mda_code* field to the *job* table.
- Create a new table, *mda*, with fields *mda_code* and *mda_name*.
- Set the *mda* field of all existing jobs to 'nonmem';

The following DDL would accomplish what you need:

```
ALTER TABLE job ADD mda_code varchar(6);
CREATE TABLE mda (
  mda_code char(6) NOT NULL default '',
  mda_name char(20) default NULL,
  PRIMARY KEY  (mda_code)
) TYPE=MyISAM;
INSERT INTO mda (mda_code, mda_name) VALUES ('nonmem','NONMEM Style MDA');
INSERT INTO mda (mda_code, mda_name) VALUES ('saam2' ,'SAAM II Style MDA');
UPDATE job SET mda_code='nonmem';
```

Using a text editor as an ordinary user on *dbserver*, place the above lines in a file. You can use any name for the file, but for this example, let us call it `modscript.sql`, which is the default name used by *load_spktest.pl*.

Using the Modification Script

This section continues the above example, showing how the modification script would be used.

1. Get exclusive use of the test environment through negotiation with the other software engineers.
2. Take a snapshot of the production database to use to initialize a test database, using *take_snapshot.pl* for this purpose. The snapshot will consist of three files, called `basedata.sql`, `schema.sql` and `userdata.sql`, created by *take_snapshot.pl* in the current directory. Have *take_snapshot.pl* place several jobs in the snapshot: 10, 20, 30 (and parent and child jobs) for example.

   ```
   take_snapshot.pl 10, 20, 30
   ```

3. Create a test database, *spktest*, using *load_spktest.pl* for this purpose. As input, *load_spktest.pl* will use your `modscript.sql` and well as the three files that were created by *take_snapshot.pl*.

   ```
   load_spktest.pl
   ```

4. Use *mysql* interactively to examine the *spktest* database that you have just created. You should find that the *mda_code* field has been added to the *job* table, that all existing jobs have *nonmem* as the value of that field, and that the *mda* table has been created and populated. Also, verify that the definitions of the new fields in the *job* and *mda* tables are what you expect.
5. When you are convinced that your database modification script is working correctly, go on to test the software modifications which the database modifications were designed to support. You can use your test database to do unit testing, and then go on to perform full system testing [4] as usual.
6. After successful system testing and staging of a deployment candidate, *do not* release the test environment. You *must* perform a deployment before system testing of any additional deployment candidates can occur.

The deployment [5] procedure, as described in the *SPK System Deployment Guide*, covers both the case where the database is changed structurally and that where it is not. The differences between the two cases are as follows:

a. If the database is being modified, a modification script similar to that in the above example must be present on *aspkserver* in the directory in which *regression_test* is run.

b. The production web server must be stopped before *deploy_candidate.pl* is run, so that changes to the software and the database can be synchronized. It is best to schedule this interruption of service and to announce the time of the shutdown to the user community.

c. After *deploy_candidate.pl* is run, the modification script must be run against the production database.

d. If the modification requires the deployment of a modified web server application, that should occur at this time.

e. After the production database has been modified and the changes verified, the production web server can be restarted, in order to restart production.

f. After production has started, but before any additional testing can occur, *take_snapshot.pl* may have to be modified, especially if the database modifications added to the *job* table either a blob field or any field normally NULL at the time a job is submitted.

Notes

1. http://www.mysql.com/
2. ../../../guide/system_test/system_test.html
3. ../../../guide/deployment/deployment.html
4. ../../../guide/system_test/system_test.html
5. ../../../guide/deployment/deployment.html