GNU SQL

A Command Line Tool for Accessing Different Databases Using DBURLs

OLE TANGE

Ole Tange works in bioinformatics in Copenhagen. He is active in the free software community and is best known for his "patented Web shop" that shows the dangers of software patents (http://ole.tange.dk/swpat). He will be happy to go to your conference to give a talk about GNU Parallel and GNU SQL.

ole@tange.dk

Today there are more ways to access a database from the command line than there are databases. Very few of them unify the access into a single URL and make it possible to give the SQL command to run on the command line.

GNU SQL introduces DBURL and a common way to run queries on databases from the command line.

Background

Around 2003 I was assigned to admin a MySQL database. I found it annoying that I could not use my Web browser for simple browsing of the content of the database. I was thinking that URLs more or less had the same information as was needed for accessing a database: protocol, username, password, host, port, path, and query ought to be enough to do at least simple database manipulation.

In 2007 I became admin of a forest of MySQL, PostGreSQL, and Oracle databases. It was a nuisance to remember the different ways to log into the databases from the command line. In discussions with my colleague Hans Schou, we came up with a common way of addressing the databases as a URL, which we call DBURL. I wrote a wrapper for the different command line tools that uses the same DBURL syntax to access the different databases. This wrapper is today known as GNU SQL.

After developing the first versions of GNU SQL, I realized that others have been thinking along the same lines: Drupal, SQLObject, SQLAlchemy, and Transifex all use some sort of DBURL. I would encourage others to adopt DBURL as a condensed way of writing the information to access a database.

```
 sql:mysql://user:password@example.com:3306/my_database?select%20*%20from%20foo;
 \_/ \___/   \___________________________/ \__________/ \______________________/
  |    |                  |                      |                  |
scheme vendor         authority                 path              query
```

Figure 1

DBURL Syntax

A DBURL has the following syntax:

```
[sql:]vendor://[user[:password]@][host][:port]/[database][?sqlquery]
```

Only vendor is required. The rest of the elements are optional, with the following defaults:

Element	**Default value**
sql:	sql:
user	your UNIX user name
password	no password
host	localhost
port	the default port for the vendor
database	your UNIX user name
sqlquery	"" = No query

The DBURL is modeled after the syntax from RFC3986 to resemble a normal URL and is partitioned similarly (see Figure 1). Quoting of special characters is done using %-quoting, thus space=%20 and /=%2F.

Aliases

To avoid having to write the full DBURL all the time aliases has been defined as DBURLs starting with ':'. Aliases are defined in ~/.sql/aliases and are simply the alias followed by the DBURL:

```
:myalias sql:mysql://user:password@example.com:3306/my_database
```

Logging in with GNU SQL

GNU SQL is part of GNU Parallel, which is available for most UNIX distributions. See http://www.gnu.org/s/parallel if it is not obvious how to install it on your system.

You get an interactive prompt by:

```
sql sql:oracle://scott:tiger@oracleserver:1521/xe
```

If the database runs on localhost, the database name is your login name and there is no password. You can simply do:

```
sql postgresql:///
```

When the DBURL is working and you get an interactive prompt, set up an alias in ~/.sql/aliases. Substitute the DBURL with one that works for you.

```
:myalias mysql://user:pass@example.com/my_database
```

Then you can get your interactive prompt by:

```
sql :myalias
```

Executing a Query

GNU SQL lets you execute a query by putting it on the command line:

```
sql :myalias "DELETE FROM users WHERE name LIKE '%tange%';"
```

If you want to run multiple queries you can put them as separate arguments:

```
sql :myoracle "SELECT 1 FROM dual;" "SELECT 2 FROM dual;"
```

or you can pipe the SQL commands into GNU SQL:

```
cat my_query.sql | sql :myalias
```

You can also execute the query by putting it in the DBURL:

```
sql :myalias?select%20*%20from%20foo\;
sql postgresql://user:pass@host/my_database?select%20*%20from%20foo\;
```

Using the Power of GNU Parallel

GNU Parallel is a good companion for GNU SQL. Using GNU Parallel it is easy to empty all tables without dropping them. Here are two ways to do it:

```
sql -n mysql:/// 'show tables' | parallel sql mysql:/// DELETE FROM {}\;
sql -n --list-tables mysql:/// | parallel sql mysql:/// DELETE FROM {}\;
```

But if you want to drop the tables, that is easy, too. Here are two ways to drop all tables in a PostgreSQL database:

```
sql -n pg:/// '\dt' | parallel --colsep '\|' -r sql pg:/// DROP TABLE {2}\;
sql -n --list-tables pg:/// | parallel --col-sep '\|' -r sql pg:/// DROP TABLE {2}\;
```

Run as a Script

When running an SQL script on a database you often end up with two scripts: one that does the connection to the database and one that contains the actual SQL to run. With GNU SQL these can easily be combined using UNIX's shebang (#!). Instead of doing:

```
sql mysql:/// < file.sql
```

you can combine file.sql with the DBURL to make a UNIX script. Create a script called demosql:

```
#!/usr/bin/sql -Y mysql:///
SELECT * FROM users;
```

Then do:

```
chmod +x demosql ; ./demosql
```

When Connections Fail

If the connection to the database is bad or the query is very long, the risk of the connection breaking increases. If the access to the database fails occasionally, retries can help make sure the query succeeds:

```
sql --retries 5 :myalias 'SELECT * FROM really_big_table;'
```

Getting General Info about the Database

Database vendors have not only each chosen their own syntax for logging in, they have also each chosen their own way of getting general information about the database.

GNU SQL contains a wrapper for getting some of the info.

Show how big the database is	sql --db-size :myalias
List the tables	sql --list-tables :myalias
List the size of the tables	sql --table-size :myalias
List the running processes	sql --show-processlist :myalias

See You on the Mailing List

I hope you have thought of situations where GNU SQL can be of benefit to you. If you like GNU SQL, please let others know about it through email lists, forums, blogs, and social networks. If GNU SQL saves you money, please donate (or have your company donate) to the FSF: https://my.fsf.org/donate. If you have questions about GNU SQL, join the mailing list at http://lists.gnu.org/mailman/listinfo/parallel.