Using a Program to List Databases

Professor Don Colton, BYU Hawaii

March 19, 2004

In this handout, we show how to work with a database from another program. We will illustrate using the Perl DBI (data base interface).

1 Connect to MySQL

In your Perl program, you can use the following lines to connect to and disconnect from a MySQL database. You are telling the computer the same things you had to specify when you connected to MySQL by hand in a previous assignment. The difference is that this protocol is easier for programming, and the by-hand protocol is easier for humans.

```
  use DBI; # to include needed definitions
  $st = DBI->connect( "DBI:mysql:DDD:HHH",
    "UUU", "PPP" );
```

Replace `DDD` with the name of your database, `HHH` with the name of your host, `UUU` with your user name, and `PPP` with your password. If your host is the "localhost" you can leave ":HHH" out of the connect statement. This is the programming equivalent of doing the following steps by hand.

```
  mysql -p -u UUU -h HHH
  (enter PPP)
  use DDD;
```

The `$st` is your session ticket to use your connection. There is nothing magic about the variable name `$st`. You can call it `$rainbow` if you like. If the connect fails, the ticket (also known as a session key) will be false. You can say `if($st)` to test whether the connection was established.

After connecting, you may perform as many or as few queries as you like. When you are done, you should disconnect as follows.

```
  $st->disconnect(); # when you are done
```

This is the equivalent of typing `exit` or `quit` by hand.

2 Issue a Query

When you are connected to the database, you can issue queries (inquiries, requests, commands). We will use a four-step process for performing a query. The first two steps are prepare and execute.

```
$query = "show databases"; # or ...
$query = "show tables"; # or ...
$query = "insert TTT values (...)"; # or ...
$query = "select * from TTT"; # or ...
$query = "select * from TTT order by cost";

$qt = $st->prepare($query); # analyze task
if ( !$qt ) { die "query analysis failed\n" }
$qt->execute(); # carry out the task
```

The `$qt` created by `prepare` is your query ticket to execute your query and retrieve its results (if any). Again, there is nothing magic about the variable name `$qt`.

3 Task 1: List One Database

Write a program to list one database that mySQL is managing. Connect. Query. Display results. Disconnect. Quit. Hint: This will do the job.

```
#!/usr/bin/perl -w
use DBI;
$st = DBI->connect ( "DBI:mysql:DDD:HHH",
  "UUU", "PPP" );
  $qt = $st->prepare ( "show databases" );
    $qt->execute();
    @z = $qt->fetchrow_array();
    print "$z[0]\n";
  $qt->finish();
$st->disconnect();
```

4 Viewing Many Results

After you have executed your query, the results are made available for you to process. If your query creates many rows of results (like `show` does), you can fetch the results one row at a time. A loop might prove helpful. There are several ways to get each row. Here we simply retrieve all the columns for one row into an array.

```
while ( @z = $qt->fetchrow_array() ) { ... }
```

The `while` will stop when `@z` is zero or undefined. This happens after the last row of data has been fetched.

After processing the rows we want, even if we have not processed them all, we can end at any time with the `finish` command.

5 Task 2: List Databases

Write a program to list all of the databases that mySQL is managing. Connect. Query. Display results. Disconnect. Quit. Hint: Replace the fetch/print in Task 1 with this while/fetch/print.

```
while ( @z = $qt->fetchrow_array() ) {
  print $z[0] }
```

6 Task 3: List Tables

Write a program to list all of the tables in your database. Connect. Query. Display results. Disconnect. Quit. You must have at least two tables to get credit.

7 Task 4: List Rows

Write a program to list all of the rows in one of your database tables. Connect. Query. Display results. Disconnect. Quit. You must have at least three rows and three columns to get credit. Hint:

```
while ( @z = $qt->fetchrow_array() ) {
  foreach $v (@z) { print "  $v  " }
  print "\n" }
```