Introduction to Using Databases with Perl

Brad Marshall <brad.marshall@member.sage-au.org.au>

November 16, 2003

1 Introduction

Perl provides an excellent set of modules for talking to databases called DBI (for Database Interface). These modules allow you to connect to a wide variety of databases, and switch between them almost transparently - assuming the SQL used isn't database specific.

DBI works by providing a front end API to database access, and uses backend modules called DBD to provide the actual communication to the databases. There are DBD modules for most common (and not so common) databases, ranging from MySQL, PostgreSQL, Oracle and ODBC to Excel, Google, LDAP and flat files - see the DBI website for a full list.

This article will cover the basics of connecting, inserting and reading data from a database, using PostgreSQL as the example backend database.

2 Perl Scripting

2.1 Modules

The first thing that's required for using DBI is the following line in your perl script:

```
use DBI;
```

This will load and initialise the DBI module for your script - note you don't need to import any of the DBD modules, they will be done as required.

2.2 Connecting

The next step is to connect to the database, via the following:

```
my $dbh = DBI->connect("dbi:Pg:dbname=$db;host=$hostname",
    "$user", "$password")
    or die "Can't connect to postgres db: $!\n";
```

This creates a database handle for the connection to the database, using the given hostname, database, username and password. To change the type of database you are talking to, you simply have to change the connection string to the appropriate new database backend - for example, for MySQL the connection string would be:

```
my $dbh = DBI->connect("dbi:mysql:database=$db;host=$hostname",
    "$user", "$password")
```

2.3 Preparing Queries

Next you need to create a statement that you wish to run against the database, such as:

```
my $fullquery = "SELECT * FROM table";
my $sth = $dbh->prepare($fullquery)
   or die "Can't prepare SQL statement: ", $dbh->errstr(), "\n";
$sth->execute()
  or die "Can't execute SQL statement: ", $sth->errstr(), "\n";
```

This prepares and executes the statement handler for a simple SQL query that returns all the data in the table. Preparing the statement allows the database to parse the statement and check that it is valid SQL, and the tables and columns that you are referring to exist and you have permission to read them, among other things.

Another option with preparing a statement is to use placeholders, or bind values. To prepare the statement handler with placeholders, you can do:

```
my $sth = $dbh->prepare(SELECT ?,? FROM table")
  or die "Can't prepare SQL statement: ", $dbh->errstr(), "\n";
```

You can then fill in the placeholders either using bind values, or in the execute statement. To use bind values, you specify the values via the following:

```
$sth->bind_param(1, "user");
$sth->bind_param(2, "name");
```

Alternatively, to fill out the values in the execute statement, you do the following:

```
$sth->execute("user", "name");
```

2.4 Returning Data

Obviously, the next step will be getting the data that the query provided. The simplest way is:

```
while( my @ary = $sth->fetchrow_array) {
     # Do something with @ary
}
```

Another option to simply view the results is:

```
$rows = $sth->dump_results();
```

2.5 Closing Handlers

To close a statement handler - even if you haven't read all the data from it is simple - just do the following:

```
$sth->finish;
```

2.6 Disconnecting

Once you've finished with a database connection it is good form to close it off - even though it will automatically close once you reach the end of the script.

```
$dbh->disconnect or warn "Disconnection failed: $!\n";
```

3 Conclusion

As you can see, this was a very quick run through of the basics of using DBI which shows the easy and power of the modules. You can do far, far more with it than covered here - for more information see "Programming the Perl DBI" by Alligator Descartes and Tim Bunce, and the Perl DBI website at http://dbi.perl.org/.