**MySQL queries with Perl**

We have already established a connection to the database. Now we are going modify and fetch the data from the database.

Data is retrieved from the database with the SELECT statement. In Perl DBI, first we prepare the SQL statement with the prepare() method. The SQL string is sent to the database engine, which checks the statement validity, syntax and in some databases also the user permissions to perform certain queries. If all is OK, a reference to the statement handle is returned to the Perl script. The next step is the call to the execute() method. The method executes the query within the database. At this moment the result stays in the database. The Perl script does not contain the data yet. For non-select statements, the execute() method returns the number of rows affected if known. In the last step the data is fetched from the database. The data is pulled row by row and populated into the Perl data structures.

The Perl DBI has several methods to fetch data from database tables.

|  |  |
| --- | --- |
| **Method** | **Description** |
| fetchrow\_arrayref() | Fetches the next row of data and returns a reference to an array. |
| fetchrow\_array() | Fetches the next row of data and returns it as a list. |
| fetchrow\_hashref() | Fetches the next row of data and returns it as a reference to a hash. |
| fetchall\_arrayref() | Fetches all data & returns a reference to an array that has one reference per row. |
| fetch() | The method is an alias for fetchrow\_arrayref(). |
| fetchrow() | The method is an alias for fetchrow\_array(). |

After the SQL statement was prepared and executed, we call one of the available fetch methods.

|  |  |
| --- | --- |
| **Method** | **Description** |
| selectrow\_arrayref() | Combines prepare(), execute() and fetchrow\_arrayref() into a single call. |
| selectrow\_hashref() | Combines prepare(), execute() and fetchrow\_hashref() into a single call. |
| select\_row\_array() | Combines prepare(), execute() and fetchrow\_array() into a single call. |
| selectall\_arrayref() | Combines prepare(), execute() and fetchall\_arrayref() into a single call. |
| selectall\_hashref() | Combines prepare(), execute() and fetchall\_hashref() into a single call. |
| selectcol\_arrayref() | Combines prepare(), execute() and fetching one col from all rows into a single call. |

In the second table we have a list of utility methods which combine three methods into one call. They are convenience methods.

**The fetch methods**

In the first example we will demonstrate the usage of the fetchrow\_arrayref() method.

#!/usr/bin/perl

use strict;

use DBI;

my $dbh = DBI->connect(

"dbi:mysql:dbname=mydb",

"user12",

"34klq\*",

{ RaiseError => 1 },

) or die $DBI::errstr;

my $sth = $dbh->prepare("SELECT \* FROM Cars LIMIT 5");

$sth->execute();

my $row;

while ($row = $sth->fetchrow\_arrayref()) {

print "@$row[0] @$row[1] @$row[2]\n";

}

$sth->finish();

$dbh->disconnect();

In the example we select 5 rows from the Cars table. The data is retrieved with the fetchrow\_arrayref() method.

my $sth = $dbh->prepare("SELECT \* FROM Cars LIMIT 5");

$sth->execute();

These are the first two phases of the data retrieval process. We prepare and execute the SELECT statement.

my $row;

while ($row = $sth->fetchrow\_arrayref()) {

print "@$row[0] @$row[1] @$row[2]\n";

}

Now we are fetching the data. The fetchrow\_arrayref() method fetches the next row of data and returns a reference to an array holding the field values. We put the method in the while loop which terminates, when there are no more rows left.

$ ./fetchrow\_arrayref.pl

1 Audi 52642

2 Mercedes 57127

3 Skoda 9000

4 Volvo 29000

5 Bentley 350000

Example output.

In the second example, we will use the fetchrow\_array() method.

#!/usr/bin/perl

use strict;

use DBI;

my $dbh = DBI->connect(

"dbi:mysql:dbname=mydb",

"user12",

"34klq\*",

{ RaiseError => 1 },

) or die $DBI::errstr;

my $sth = $dbh->prepare( "SELECT \* FROM Cars LIMIT 5" );

$sth->execute();

my @row;

while (@row = $sth->fetchrow\_array()) {

print "@row\n";

}

$sth->finish();

$dbh->disconnect();

In this script we connect to the database and fetch 5 rows of the Cars table one by one using the fetchrow\_array() method.

my @row;

while (@row = $sth->fetchrow\_array()) {

print "@row\n";

}

The fetchrow\_array() method fetches the next row of data and returns it as a list containing the field values. We use the while loop to go through all 5 rows.

In the next example, we will fetch data by their column names. For this we will utilise the fetchrow\_hashref() method.

#!/usr/bin/perl

use strict;

use DBI;

my $dbh = DBI->connect(

"dbi:mysql:dbname=mydb",

"user12",

"34klq\*",

{ RaiseError => 1 },

) or die $DBI::errstr;

my $sth = $dbh->prepare( "SELECT \* FROM Cars LIMIT 5" );

$sth->execute();

my $row;

while($row = $sth->fetchrow\_hashref()) {

print "$row->{Id} $row->{Name} $row->{Price}\n";

}

$sth->finish();

$dbh->disconnect();

In the example, the data is returned in the form of a reference to a Perl hash.

my $row;

while($row = $sth->fetchrow\_hashref()) {

print "$row->{Id} $row->{Name} $row->{Price}\n";

}

The fetchrow\_hashref() method fetches the next row of data and returns it as a reference to a hash containing field name and field value pairs. With this method, we can retrieve the values by their column names.

In the last example of this section, we fetch all data from the SELECT statement at one step. We use the fetchall\_arrayref() method.

#!/usr/bin/perl

use strict;

use DBI;

my $dbh = DBI->connect(

"dbi:mysql:dbname=mydb",

"user12",

"34klq\*",

{ RaiseError => 1 },

) or die $DBI::errstr;

my $sth = $dbh->prepare("SELECT \* FROM Cars LIMIT 5");

$sth->execute();

my $all = $sth->fetchall\_arrayref();

foreach my $row (@$all) {

my ($id, $name, $price) = @$row;

print "$id $name $price\n";

}

$sth->finish();

$dbh->disconnect();

The example selects and prints five rows from the Cars table.

my $all = $sth->fetchall\_arrayref();

We fetch all data in one method call. The fetchall\_arrayref() method returns a reference to an array that contains one reference per row.

foreach my $row (@$all) {

my ($id, $name, $price) = @$row;

print "$id $name $price\n";

}

We use the foreach loop to go through the retrieved data.

**Dumping data**

Perl DBI has a special method called dump\_results(). This method is designed as a handy utility for prototyping and testing queries. It uses a neat\_list() method to format and edit the string for reading by humans. It is not recommended for data transfer applications.

#!/usr/bin/perl

use strict;

use DBI;

my $dbh = DBI->connect(

"dbi:mysql:dbname=mydb",

"user12",

"34klq\*",

{ RaiseError => 1 },

) or die $DBI::errstr;

my $sth = $dbh->prepare( "SELECT \* FROM Cars LIMIT 5" );

$sth->execute();

$sth->dump\_results();

$sth->finish();

$dbh->disconnect();

In the example we will dump all the data from the result set.

my $sth = $dbh->prepare( "SELECT \* FROM Cars LIMIT 5" );

$sth->execute();

The SQL statement selects five rows from the Cars table. And all three columns.

$sth->dump\_results();

The dump\_results() selects all rows from the statement handle and prints them. It is a method used for prototyping and testing.

$ ./dump.pl

1, 'Audi', 52642

2, 'Mercedes', 57127

3, 'Skoda', 9000

4, 'Volvo', 29000

5, 'Bentley', 350000

5 rows

This is the output of the example.

**The convenience methods**

We will show two examples that will use the aforementioned convenience methods.

#!/usr/bin/perl

use strict;

use DBI;

my $dbh = DBI->connect(

"dbi:mysql:dbname=mydb",

"user12",

"34klq\*",

{ RaiseError => 1 },

) or die $DBI::errstr;

my $ary = $dbh->selectrow\_arrayref("SELECT \* FROM Cars WHERE Id = 5");

print join(" ", @$ary), "\n";

$dbh->disconnect();

In the first code example, we will call the selectrow\_arrayref() method. We select the fifth row from the Cars table.

my $ary = $dbh->selectrow\_arrayref("SELECT \* FROM Cars WHERE Id = 5");

The selectrow\_arrayref() method combines the prepare(), execute() and fetchrow\_arrayref() into a single call. It returns a reference to the first row of data from the statement. Note that we do not use the statement handle. We use the $dbh database handle object.

print join(" ", @$ary), "\n";

We print the row to the console.

The following example shows the selectall\_arrayref() method.

#!/usr/bin/perl

use strict;

use DBI;

my $dbh = DBI->connect(

"dbi:mysql:dbname=mydb",

"user12",

"34klq\*",

{ RaiseError => 1 },

) or die $DBI::errstr;

my $all = $dbh->selectall\_arrayref("SELECT \* FROM Cars LIMIT 5");

foreach my $row (@$all) {

my ($id, $name, $price) = @$row;

print "$id $name $price\n";

}

$dbh->disconnect();

We again retrieve 5 rows from the Cars table.

my $all = $dbh->selectall\_arrayref("SELECT \* FROM Cars LIMIT 5");

The selectall\_arrayref() method returns a reference to an array containing a reference to an array for each row of data fetched. The supplied SQL statement selects 5 rows from the Cars table. Note that we did not call neither the prepare(), nor the execute() method. It is because the selectall\_arrayref() method combines prepare(), execute() and fetchall\_arrayref() into one single call.

foreach my $row (@$all) {

my ($id, $name, $price) = @$row;

print "$id $name $price\n";

}

We go through the fetched array of arrays and print the data to the terminal.

$ ./selectall\_arrayref.pl

1 Audi 52642

2 Mercedes 57127

3 Skoda 9000

4 Volvo 29000

5 Bentley 350000

This is the output of the example.

**Parameterized queries**

Now we will concern ourselves with parameterized queries. When we use parameterized queries, we use placeholders instead of directly writing the values into the statements. Parameterized queries increase security and performance.

A programmer must be always cautious when the program receives an input from the user. Instead of building a string from the user input, we bind the value later to the prepared statement.

#!/usr/bin/perl

use strict;

use DBI;

my $dbh = DBI->connect(

"dbi:mysql:dbname=mydb",

"user12",

"34klq\*",

{ RaiseError => 1 },

) or die $DBI::errstr;

my $id = 3;

my $sth = $dbh->prepare( "SELECT \* FROM Cars WHERE Id = ?" );

$sth->execute($id);

my $ret = $sth->fetch();

foreach my $row (@$ret) {

print "$row ";

}

print "\n";

$sth->finish();

$dbh->disconnect();

In the code example, we select a specific row from the table. The SQL statement has one placeholder, which is filled later in the code.

my $id = 3;

This could be an input from the user.

my $sth = $dbh->prepare( "SELECT \* FROM Cars WHERE Id = ?" );

The question mark ? is a placeholder for a value. The value is added later.

$sth->execute($id);

The execute() statement takes one parameter, which is bound to the placeholder.

$ ./parameterized.pl

3 Skoda 9000

We have retrieved one row from the Cars table using a parameterized query.

In the second example, we will use a parameterized query using one of the convenience select methods.

#!/usr/bin/perl

use strict;

use DBI;

my $dbh = DBI->connect(

"dbi:mysql:dbname=mydb",

"user12",

"34klq\*",

{ RaiseError => 1 },

) or die $DBI::errstr;

my $id = 2;

my @ary = $dbh->selectrow\_array("SELECT \* FROM Cars WHERE Id = ?", undef, $id);

print join(" ", @ary), "\n";

$dbh->disconnect();

We have one placeholder to be filled in the SELECT query.

my @ary = $dbh->selectrow\_array("SELECT \* FROM Cars WHERE Id = ?", undef, $id);

The third parameter of the selectrow\_array() method takes a value for the placeholder.

In this part of the MySQL Perl tutorial, we have demonstrated how to fetch data from the database using various Perl DBI methods.