MySQL C API

TABD

MySQL 8.0 Reference Manual

- https://dev.mysql.com/doc/refman/8.0/en/
- https://dev.mysql.com/doc/refman/8.0/en/installing.html
- https://dev.mysql.com/doc/refman/8.0/en/c-api.html

28.6 MySQL C API

28.6.3 Building and Running C API Client Programs

```
gcc -c `mysql_config --cflags` progname.c
```

```
gcc -o progname progname.o `mysql_config --libs`
```

28.6.4 C API Data Structures

- `MYSQL`

This structure represents handler for one database connection. It is used for almost all MySQL functions.

- `MYSQL_RES`

This structure represents the result of a query that returns rows (`SELECT`, `SHOW`, `DESCRIBE`, `EXPLAIN`).

- `MYSQL_ROW`

This is a type-safe representation of one row of data. It is currently implemented as an array of counted byte strings. (You cannot treat these as null-terminated strings if field values may contain binary data, because such values may contain null bytes internally.)

28.6.4 C API Data Structures

- `MYSQL_FIELD`

  This structure contains metadata: information about a field, such as the field's name, type, and size. You may obtain the MYSQL_FIELD structures for each field by calling mysql_fetch_field() repeatedly. Field values are not part of this structure; they are contained in a MYSQL_ROW structure.

- `MYSQL_FIELD_OFFSET`

This is a type-safe representation of an offset into a MySQL field list. (Used by mysql_field_seek().) Offsets are field numbers within a row, beginning at zero.

MYSQL_FIELD

The MYSQL_FIELD structure contains the members described in the following list:

`char * name`

The name of the field, as a null-terminated string. If the field was given an alias with an AS clause, the value of name is the alias. For a procedure parameter, the parameter name.

`char * org_name`

The name of the field, as a null-terminated string. Aliases are ignored. For expressions, the value is an empty string. For a procedure parameter, the parameter name.

`char * table`

The name of the table containing this field, if it is not a calculated field. For calculated fields, the table value is an empty string. If the column is selected from a view, table names the view. If the table or view was given an alias with an AS clause, the value of table is the alias. For a UNION, the value is the empty string. For a procedure parameter, the procedure name.

....

28.6.5 C API Function Overview

- https://dev.mysql.com/doc/refman/8.0/en/c-api-function-overview.html

- mysql_init(): Gets or initializes a MYSQL structure.
- mysql_real_connect(): Connects to a MySQL server.
- mysql_real_query(): Executes an SQL query specified as a counted string.
- mysql_store_result(): Retrieves a complete result set to the client.
- mysql_fetch_row(): Fetches the next row from the result set.
- ….

EXAMPLES

```
#include <stdio.h>
#include <mysql.h>

int main(int argc, char **argv)
{
  printf("MySQL client version: %s\n", mysql_get_client_info());
  exit(0);
}
```

Here is how we compile the code example.

```
$ gcc version.c -o version  `mysql_config --cflags --libs`

$ ./version

MySQL client version: 5.1.67
```

EXAMPLES

```
#include <stdio.h>
#include <mysql.h>

int main(int argc, char **argv)
{
  MYSQL *con = mysql_init(NULL);

  if (con == NULL)
  {
      fprintf(stderr, "%s\n", mysql_error(con));
      exit(1);
  }

  if (mysql_real_connect(con, "localhost", "root", "root_pswd",
          NULL, 0, NULL, 0) == NULL)
  {
      fprintf(stderr, "%s\n", mysql_error(con));
      mysql_close(con);
      exit(1);
  }

  if (mysql_query(con, "CREATE DATABASE testdb"))
  {
      fprintf(stderr, "%s\n", mysql_error(con));
      mysql_close(con);
      exit(1);
  }

  mysql_close(con);
  exit(0);
}
```

EXAMPLES

```
#include <my_global.h>
#include <mysql.h>

void finish_with_error(MYSQL *con)
{
  fprintf(stderr, "%s\n", mysql_error(con));
  mysql_close(con);
  exit(1);
}

int main(int argc, char **argv)
{
  MYSQL *con = mysql_init(NULL);

  if (con == NULL)
  {
      fprintf(stderr, "%s\n", mysql_error(con));
      exit(1);
  }

  if (mysql_real_connect(con, "localhost", "user12", "34klq*",
          "testdb", 0, NULL, 0) == NULL)
  {
      finish_with_error(con);
  }

  if (mysql_query(con, "DROP TABLE IF EXISTS Cars")) {
      finish_with_error(con);
  }

  if (mysql_query(con, "CREATE TABLE Cars(Id INT, Name TEXT, Price INT)")) {
      finish_with_error(con);
  }

  if (mysql_query(con, "INSERT INTO Cars VALUES(1,'Audi',52642)")) {
      finish_with_error(con);
  }

  if (mysql_query(con, "INSERT INTO Cars VALUES(2,'Mercedes',57127)")) {
      finish_with_error(con);
  }

  mysql_close(con);
  exit(0);
}
```

EXAMPLES

```
#include <stdio.h>
#include <mysql.h>

void finish_with_error(MYSQL *con)
{
  fprintf(stderr, "%s\n", mysql_error(con));
  mysql_close(con);
  exit(1);
}

int main(int argc, char **argv)
{
  MYSQL *con = mysql_init(NULL);

  if (con == NULL)
  {
      fprintf(stderr, "mysql_init() failed\n");
      exit(1);
  }

  if (mysql_real_connect(con, "localhost", "user12", "34klq*",
          "testdb", 0, NULL, 0) == NULL)
  {
      finish_with_error(con);
  }

  if (mysql_query(con, "SELECT * FROM Cars"))
  {
      finish_with_error(con);
  }

  MYSQL_RES *result = mysql_store_result(con);

  if (result == NULL)
  {
      finish_with_error(con);
  }

  int num_fields = mysql_num_fields(result);

  MYSQL_ROW row;

  while ((row = mysql_fetch_row(result)))
  {
      for(int i = 0; i < num_fields; i++)
      {
          printf("%s ", row[i] ? row[i] : "NULL");
      }
          printf("\n");
  }

  mysql_free_result(result);
  mysql_close(con);

  exit(0);
}
```

TUTORIAL

- http://zetcode.com/db/mysqlc/

EXERCISE

Create a C function that print a result set in a formatted manner, just like the mysql client shell:

```
+------+------------+--------+
| Id   | Name       | Price  |
+------+------------+--------+
|    1 | Audi       |  52642 |
|    2 | Mercedes   |  57127 |
|    3 | Skoda      |   9000 |
|    4 | Volvo      |  29000 |
|    5 | Bentley    | 350000 |
|    6 | Citroen    |  21000 |
|    7 | Hummer     |  41400 |
|    8 | Volkswagen |  21600 |
+------+------------+--------+
```