

Introduction to MySQL

Aleksandr Zaks zaks@cs.nyu.edu

New York University, March, 2005

Installation

- Download <http://www.cs.nyu.edu/~zaks/my.cnf>
- Save my.cnf in your home directory as .my.cnf
- Use `ls -a .my.cnf` if you want to "see" the file
- Use emacs (M-%) to replace /home/zaks with /home/yourname in .my.cnf
- `setenv PATH " $ {PATH} :/usr/local/pkg/mysql/bin:/usr/local/pkg/mysql/libexec"`
- `mysql_install_db`

Starting a Database

- `mysqld &` or `mysqld --skip-grant-tables &`
- Use `kill pid` or `mysqladmin -u root shutdown` to shutdown `mysqld`.
- Type `ps -A | grep mysqld` to check that the daemon is running
- You may use `kill pid` or `mysqladmin -u root shutdown` to shutdown `mysqld`.

Now you are ready to connect to the database by typing: `mysql -u root` (execute `setenv EDITOR "emacs"` if you do not like vi as a default editor).

Exploring a Database

- `show databases;` press enter
- `use mysql;`
- `show tables;`
- `desc db;` or `explain db;`
- `use test;`
- `show tables;`
- `desc mysql.db;`

Table Creation

Note that you are welcome to use <http://www.cs.nyu.edu/zaks/script.txt> if you do not like to type yourself.

```
CREATE TABLE person (  
    id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,  
    name CHAR(60) NOT NULL,  
    PRIMARY KEY (id)  
) TYPE = InnoDB;
```

```
CREATE TABLE shirt (  
    id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,  
    style ENUM('t-shirt', 'polo', 'dress') NOT NULL,  
    color ENUM('red', 'blue', 'orange', 'white', 'black') NOT NULL,  
    owner SMALLINT UNSIGNED NOT NULL,  
    PRIMARY KEY (id)  
) TYPE = InnoDB;
```

Populate Data

```
INSERT INTO person VALUES (NULL, 'Antonio Paz');  
SELECT @last := LAST_INSERT_ID();
```

```
INSERT INTO shirt VALUES  
(NULL, 'polo', 'blue', @last),  
(NULL, 'dress', 'white', @last),  
(NULL, 't-shirt', 'blue', @last);
```

```
INSERT INTO person VALUES (NULL, 'Lilliana Angelovska');  
SELECT @last := LAST_INSERT_ID();
```

```
INSERT INTO shirt VALUES  
(NULL, 'dress', 'orange', @last),  
(NULL, 'polo', 'red', @last),  
(NULL, 'dress', 'blue', @last),  
(NULL, 't-shirt', 'white', @last);
```

Joining Tables

- % Matches any number of characters, even zero characters
- _ Matches exactly one character

```
SELECT s.* FROM person p, shirt s
WHERE p.name LIKE 'Lilliana%'
      AND s.owner = p.id
      AND s.color <> 'white';
```

Transactions(1/2)

- `set autocommit = 0;`
- `select @@autocommit;`
- `commit;`
- `rollback;`

```
BEGIN;
```

```
SELECT @A:=SUM(salary) FROM table1 WHERE type=1;
```

```
UPDATE table2 SET summary=@A WHERE type=1;
```

```
COMMIT;
```

Transactions(2/2)

- `set autocommit = 0;`
- `delete from person;`
- `select * from person;`
- `rollback;`
- `select * from person;`

Easy Table Backup

- `create table person_bkp type = InnoDB as select * from person;`
- `show create table person_bkp;`
- `show create table person;`
- `select * from person_bkp;`

More Advance Backup

Backup via SQL script

The `mysqldump` client can be used to dump a database or a collection of databases for backup or for transferring the data to another SQL server (not necessarily a MySQL server). The dump contains SQL statements to create the table and/or populate the table.

- `mysqldump test -u root;`

Backup via ASCII files

- `select * from person into outfile "person.txt"`
- Within mysql, type `system cat mysqlDB/test/person.txt;` to see the content of the file.
- Use `mysqlimport` utility to import data from ASCII files

A Simple C Client Program

```
#include <mysql.h>
... //define def_host_name, def_user_name ...
MYSQL *conn; /* pointer to connection handler */

int main (int argc, char *argv[]) {

    conn = mysql_init (NULL);
    mysql_real_connect (
        conn,          /* pointer to connection handler */
        def_host_name, /* host to connect to */
        def_user_name, /* user name */
        def_password,  /* password */
        def_db_name,   /* database to use */
        0,             /* port (use default) */
        NULL,          /* socket (use default) */
        0);           /* flags (none) */
    mysql_close (conn);
    exit (0);
}
```

Error Handling

```
void print_error (MYSQL *conn, char *message) {
    fprintf (stderr, "%s\n", message);
    if (conn != NULL)
    {
        fprintf (stderr, "Error %u (%s)\n",
                mysql_errno (conn), mysql_error (conn));
    }
}
```

Handling Queries That Return No Data Result Set

```
if(mysql_query(
    conn,
    "INSERT INTO my_tbl SET name = 'My Name' ") != 0) {
    print_error ("INSERT statement failed");
}
else {
    printf ("INSERT statement succeeded: %lu rows affected\n",
        (unsigned long) mysql_affected_rows (conn));
}
```

Handling Queries That Result Set(1/2)

Generate result set:

```
MYSQL_RES *res_set;

if (mysql_query (conn, "SHOW TABLES FROM mysql") != 0)
    print_error (conn, "mysql_query() failed");
else {
    res_set = mysql_store_result (conn); /* generate result set */
    if (res_set == NULL)
        print_error (conn, "mysql_store_result() failed");
    else{
        /* process result set, then deallocate it */
        process_result_set (conn, res_set);
        mysql_free_result (res_set);
    }
}
```

Handling Queries That Result Set(2/2)

Processing result set:

```
void process_result_set (MYSQL *conn, MYSQL_RES *res_set) {
MYSQL_ROW    row; unsigned int  i;

    while ((row = mysql_fetch_row (res_set)) != NULL){
        for (i = 0; i < mysql_num_fields (res_set); i++){
            if (i > 0)
                fputc ('\t', stdout);
            printf ("%s", row[i] != NULL ? row[i] : "NULL");
        }
        fputc ('\n', stdout);
    }
    if (mysql_errno (conn) != 0)
        print_error (conn, "mysql_fetch_row() failed");
    else
        printf ("%lu rows returned\n", mysql_num_rows (res_set));
}
```

Compiling and Linking

For C client:

- `mysql_config --cflags // -I /usr/local/pkg/mysql/include/mysql`
- `mysql_config --libs // -L /usr/local/pkg/mysql/lib/mysql`

For C++ client:

- See <http://tangentsoft.net/mysql++/>

A Simple C++ Client Program(1/2)

Generate result set:

```
#include <iostream>
#include <iomanip>
#include "mysql++.h"
using namespace mysqlpp;

int main() {
    Connection con("test");
    // The full format for the Connection constructor is
    // Connection(cchar *db, cchar *host="",
    //            cchar *user="", cchar *passwd="")

    Query query = con.query();
    query << "select * from person";
    Result res = query.store();
    //Query::store() executes the query and returns the results

    cout << "Records Found: " << res.size() << endl << endl;
```

A Simple C++ Client Program(1/2)

Processing result set:

```
cout.setf(ios::left);  
cout << setw(5) << "id" << "name" << endl;
```

```
Row row;
```

```
Result::iterator i;
```

```
// The Result class has a read-only Random Access Iterator
```

```
for (i = res.begin(); i != res.end(); i++) {
```

```
    row = *i;
```

```
    cout << setw(5) << row[0] << row[name] << endl;
```

```
}
```

```
} //end main
```

Recommended Topics to Review

- User Management (create/drop user, grant/revoke...)
- Isolation Levels
 - read uncommitted
 - read committed
 - repeatable read
 - serializable
- Table/Record locking. Deadlocks.
- Query Optimization
- If you like GUI, checkout MySQL Control Center, MySQL Query Browser, ...