MySQL Reference

1 Introduction

The MySQL is a relational database system (RDBS) and a query language. The query processor of MySQL accepts commands from your keyboard, as well as through an embedded high-level programming language (such as PHP), executes them through the MySQL RDBS, and format the results as you specify.

2 Downloading & Setting up

The MySQL RDBS (version 5.6) can be freely downloaded using the URL http://dev.mysql.com/downloads/mysql/. After you have successfully downloaded MySQL, you can access MySQL using the Command Line Client on your PC. Hereafter, follow the instructions given below to work on your database.

1. Log on to MySQL. Using the Command Line Client, you must enter your password (which is the password created by you when you installed the MySQL RDBS earlier). After you have entered your password, press [Return].

2. The command starts MySQL, which should display its prompt, ‘mysql>’. The mysql prompt (i.e., mysql>) means that MySQL is ready for you to enter a command. For more details on connecting to and disconnecting from MySQL, access the Web page (http://dev.mysql.com/doc/mysql/en/Connecting-disconnecting.html).

3 Use MySQL databases

Before you can create any tables and insert any table data for querying, you must first create a database where new tables can be defined and stored. You should create a MySQL database after you have accessed the database system for the first time. The following command allows you to create a new database:

```
mysql> CREATE DATABASE db_name
```

where db_name is the new database name which is a character string, such as cs452.

You can access and define tables in your MySQL database using the following command at the MySQL command prompt:

```
mysql> USE db_name
```

where db_name is your MySQL user account name, as mentioned earlier. The USE command allows you to access a previously created database.
4 MySQL Commands

Listed below are some of the basic SQL commands which you may use to create new tables, append rows in tables, join different tables, and query tables in an MySQL database.

4.1 Creating a table

In the CREATE TABLE command, you name the table (e.g., student) and its columns (e.g., ssno, sname, etc.) You specify what kind of information each column may contain (INT for whole numbers, CHAR for character values, FIXED for real numbers, etc.), and indicate the maximum width of the values stored in each column. (For different data types allowed in MySQL, see http://dev.mysql.com/doc/maxdb/en/d0/763898d21c11d2a97400a0c9449261/content.htm.)

The following command created a sample ‘student’ table:

```
CREATE TABLE student
(ssno INT(9),
sname CHAR(20),
address CHAR(25),
gpa FIXED(3,2));
```

4.2 Describing a table

The DESC or DESCRIBE command displays a brief description of a table. The description includes each column (field)’s position, data type, key constraint, default value, etc. For example, to describe the table ‘student,’ enter:

```
mysql> DESC student;
```

4.3 Inserting rows into a table

The INSERT command contains a value for each column in the table. Enter the values, separated by commas, in the same order as the columns are defined. Enclose constant character values in apostrophes (’). For example, to add a new student to the ‘student’ table, enter:

```
INSERT INTO student
VALUES (123456789, 'Mark Anderson',
'155 Denison, Manhattan', 3.75);
```

Instead of inserting each row one by one into an existing table, you can first create a batch file which includes a number of records to be inserted into the table and then load all the records into the table. For example, suppose the following four records are in a file, called batch.txt, which resides in your current directory:

```
A 1001 pc
A 1002 pc
B 1003 pc
B 2004 laptop
```

where each field is delimited by a tab and each record is ended with a newline. The following command loads the records into an existing table Product(Maker, Model, Type) all at once:
LOAD DATA LOCAL INFILE ‘batch.txt’
INTO TABLE Product
FIELDS TERMINATED BY ‘\t’
LINES TERMINATED BY ‘\n’;

(See http://dev.mysql.com/doc/mysql/en/LOAD_DATA.html for detailed discussion on loading data files.)

4.4 Creating temporary tables

MySQL allows the user to create temporary tables. A temporary table is visible only to the current connection, and is dropped automatically when the connection is closed. During two different connections, you can create the same temporary table without conflicting with each other or with an existing non-temporary table of the same name. (In the latter case, the existing table is hidden until the temporary table is dropped.)

If you are not sure whether (the name of) a (temporary) table already exists, you can specify the keywords IF NOT EXISTS in the CREATE (TEMPORARY) TABLE command so that an error does not occur if there is an existing table with the same name and you don’t care the new table overwriting an existing one. Note that there is no verification that the existing table has a structure identical to that indicated by the CREATE TABLE statement. The following SQL statement creates a new temporary table TmpStd using the IF NOT EXISTS option:

```
CREATE TEMPORARY TABLE IF NOT EXISTS TmpStd
(ssno INT(9),
 sname CHAR(20),
 address CHAR(25),
 gpa FIXED(3,2));
```

Note that all table names in MySQL are case sensitive. Thus, table TmpStd and table tmpStd are two different tables in MySQL. For more detailed discussion on the CREATE TABLE command, see http://dev.mysql.com/doc/mysql/en/CREATE_TABLE.html.

4.5 Retrieving information from a table

To retrieve information from one or more tables, use the SQL command SELECT. SELECT retrieves information from an existing table (or tables), especially the designated columns to retrieve. For example, to list the social security number, name, address, and gpa of each student in the ‘student’ relation, enter:

```
SELECT ssno, sname, address, gpa FROM student;
```

‘*’ makes SELECT display all columns of the table(s) specified (in the FROM clause), in the order they were defined when the table(s) were created.

To eliminate duplicate rows before they are displayed, use the DISTINCT clause in the SELECT command. For example, to list all distinct student names, enter:

```
SELECT DISTINCT sname FROM student;
```
To select rows from a table, include a **WHERE** clause in the SELECT command. A SELECT command with a WHERE clause retrieves only those rows that meet the search conditions.

Suppose you want to list information about the student ‘Mark Anderson’, enter:

```sql
SELECT *
FROM student
WHERE sname = 'Mark Anderson';
```

When you use a character value in a WHERE clause, you must enclose it in single quotes. You do not need to enclose number values in quotes.


### 4.6 Copying rows between tables

You can use an INSERT command with a subquery to select rows from one or many tables and insert them into another table. The subquery of INSERT replaces the VALUES clauses. Only these columns and rows selected by the subquery will be inserted. For example, suppose you want to append all student records with grade point average 3.70 or above in the table ‘student’ to the table ‘outstanding’ (assume that the table ‘outstanding’ has already been created), enter:

```sql
INSERT INTO outstanding (ssno, sname, address, gpa)
SELECT * 
FROM student
WHERE gpa ≥ 3.70;
```

You may also create a table and copy rows into it in a single operation by using the CREATE TABLE command. If you had created ‘outstanding’ and copied the rows into it in one operation, the CREATE TABLE command would have looked like this:

```sql
CREATE TABLE outstanding (ssno, sname, address, gpa)
(SELECT *
FROM student
WHERE gpa >= 3.70);
```

The columns’ types and widths of the table ‘outstanding’ are not specified: they are determined by the types and widths of the columns returned by the subquery.

### 4.7 Deleting rows from a table

To delete one or more rows from a table, you must select the rows with an appropriate condition in a WHERE clause. For example, to delete all students with grade point averages below 2.0, enter:

```sql
DELETE FROM student
WHERE gpa < 2.0;
```

To delete all the rows from a table, enter the DELETE command *without* a WHERE clause.

You can delete rows from multiple tables in a single SQL statement by specifying tables involved in the FROM clause and, if necessary, selection criteria specified in a WHERE clause. The following command deletes all the rows in tables `player` and `team`:

```sql
DELETE FROM player, play_for
USING player, play_for

where the table names specified after ‘USING’ are called table_references, or

DELETE player, play_for
FROM player, play_for

where the table names specified after ‘FROM’ are table_references.

The following command deletes selected rows in tables player and team:

DELETE FROM player, play_for
USING player, play_for
WHERE player.SSNO = play_for.SSNO;


4.8 Updating rows in a table

You can update one or more rows by selecting rows with the UPDATE’s WHERE clause. All rows that satisfy the clause’s condition will be updated.

For example, to update Mark’s address, enter:

UPDATE student
SET address = '10 Anderson, J.C.'
WHERE sname = 'Mark Anderson';

MySQL allows the user to perform update operations over multiple tables. For example, the following SQL statement modifies the value of attribute year in player and team in play_for:

UPDATE player, play_for
SET Year = Year + 1, Team = 'Lion'
WHERE player.SSNO = play_for.SSNO;

Notice the use of comma (i.e., ‘,’) instead of AND in the SQL statement above. For more information on update, see http://dev.mysql.com/doc/mysql/en/UPDATE.html.

4.9 Joining different tables

To make a query in which rows of two or more tables are joined, you must specify the join columns that contain corresponding information in the joined tables. Specify the tables to be joined in the SELECT commands’ FROM clause, and specify the join columns in the WHERE clause:

SELECT columns
FROM table1,table2,....,tablen
WHERE condition;

For example, to find the employee Joe’s location, enter:
SELECT ename, loc
FROM employee, department
WHERE ename = 'Joe' AND
    employee.deptno = department.deptno;

The ‘deptno’ column name is prefixed by the table name ‘employee’ or ‘department’. This is because both ‘employee’ and ‘department’ have a column named ‘deptno’. When two columns have the same name, use table name prefixed to show exactly which columns you mean. If a column name is unique in the tables specified in a query, you need not prefix it.

See [http://dev.mysql.com/doc/mysql/en/JOIN.html](http://dev.mysql.com/doc/mysql/en/JOIN.html) for more information on the join operation, including left outer join, right outer join, etc. Note that the current version of MySQL does not support full outer join.

### 4.10 Abbreviating a table name

Although table name prefixes prevent ambiguity in a query, they can be tedious to enter. You can define temporary labels in the FROM clause and use them in the remainder of the query. Such temporary labels are sometimes called table aliases.

To use table labels to abbreviate the table names in the previous example, enter:

```
SELECT ename, loc
FROM employee E, department D
WHERE ename = 'Joe' AND
    E.deptno = D.deptno;
```

### 4.11 Matching a value in/not in a list

The **IN** operator lets you select rows that match one of the values in a list which is retrieved from more than one table. The **NOT IN** operator is the complement of the **IN** operator. For example, to list the employees located in ‘Chicago’ with the same position as ‘Joe’, enter:

```
SELECT ename, position
FROM employee E, department D
WHERE loc = 'Chicago' AND
    E.deptno = D.deptno AND
    position IN
        (SELECT position
         FROM employee
         WHERE ename = 'Joe');
```

See [http://dev.mysql.com/doc/mysql/en/ANY_IN_SUBQUERY.html](http://dev.mysql.com/doc/mysql/en/ANY_IN_SUBQUERY.html) for further discussion on **IN/NOT IN**.

### 4.12 Set operators

A query (including a subquery) may be composed of two or more queries with the set operator **UNION**, which returns all distinct rows returned by *either* of the queries it applies to. The **UNION** operator is most useful for combining different subqueries that yield different union-compatible tables, i.e., the subqueries must select the same number of columns, and corresponding columns must be of the same type. The *set intersection* and *set difference* (as well as the *except*) operators, are not available in the current version of MySQL.
5 Other useful commands

Listed as follows are some of the basic MySQL commands that might be useful.

5.1 Show databases

Show databases lists the databases that you have created on the MySQL server host. Note that you can see only those databases for which you have some kind of privilege, i.e., databases that have been created by you. The following command shows all the databases that you created earlier:

   mysql> show databases;

5.2 Show tables

Show tables lists all the (non-temporary) tables in the database that you are currently using. However, if you have no privileges for a table, e.g., a table owned by another MySQL user, the table will not show up in the output using show tables. The following command shows all the (non-temporary) existing tables:

   mysql> show tables;

5.3 Retrieving/Executing a command

If you want to work with a stored command, you must retrieve/execute the command from the file in which it is stored. You can retrieve a query from a file to the system buffer with the source or \ command:

   mysql> source filename

For example, entering “source query.txt” or “\ query.txt” at the MySQL prompt

   mysql> source query.txt;
   mysql> \ query.txt;

tells MySQL to read and execute its input from the file query.txt.

5.4 Edit a command

Edit allows you to edit a command in the buffer. Using edit, you can run the host computer’s text editor without leaving MySQL. When you type edit in MySQL, it will automatically show the current command in the buffer with EDITOR, if the buffer is not empty, i.e., if you use edit for the first time after running MySQL. While editing, you can bring in other files into the buffer using the computer’s text editor command.

To invoke the edit command in MySQL, type edit at the MySQL prompt, i.e.,

   mysql> edit

After you have inserted, modified, or saved a command using edit, you can execute the command in the buffer (after quitting the editing mode) by typing ‘;’ at the MySQL command prompt, i.e.,

   mysql> ;
References: