06 MySQL

安装完MySQL后，除了MySQL Server，即真正的MySQL服务器外，还附赠一个MySQL Client程序。MySQL Client是一个命令行客户端，可以通过MySQL Client登录MySQL，然后，输入SQL语句并执行。

打开命令提示符，输入命令`mysql -u root -p`，提示输入口令。填入MySQL的root口令，如果正确，就连上了MySQL Server，同时提示符变为`mysql>`：

```
┌──────────────────────────────────────────────────────────┐
|Command Prompt                                  - □ x |
├──────────────────────────────────────────────────────────┤
|Microsoft Windows [Version 10.0.0]                        |
|(c) 2015 Microsoft Corporation. All rights reserved.      |
|                                                          |
|C:\> mysql -u root -p                                     |
|Enter password: ******                                    |
|                                                          |
|Server version: 5.7                                       |
|Copyright (c) 2000, 2018, ...                             |
|Type 'help;' or '\h' for help.                            |
|                                                          |
|mysql>                                                    |
|                                                          |
└──────────────────────────────────────────────────────────┘
```

输入`exit`断开与MySQL Server的连接并返回到命令提示符。

MySQL Client的可执行程序是mysql，MySQL Server的可执行程序是mysqld。

MySQL Client和MySQL Server的关系如下：

```
┌──────────────┐  SQL   ┌──────────────┐
| MySQL Client |──────>| MySQL Server |
└──────────────┘  TCP   └──────────────┘
```

在MySQL Client中输入的SQL语句通过TCP连接发送到MySQL Server。默认端口号是3306，即如果发送到本机MySQL Server，地址就是`127.0.0.1:3306`。

也可以只安装MySQL Client，然后连接到远程MySQL Server。假设远程MySQL Server的IP地址是`10.0.1.99`，那么就使用`-h`指定IP或域名：

```
mysql -h 10.0.1.99 -u root -p
```

小结

命令行程序`mysql`实际上是MySQL客户端，真正的MySQL服务器程序是`mysqld`，在后台运行。

管理MySQL

要管理MySQL，可以使用可视化图形界面MySQL Workbench。

MySQL Workbench可以用可视化的方式查询、创建和修改数据库表，但是，归根到底，MySQL Workbench是一个图形客户端，它对MySQL的操作仍然是发送SQL语句并执行。因此，本质上，MySQL Workbench和MySQL Client命令行都是客户端，和MySQL交互，唯一的接口就是SQL。

因此，MySQL提供了大量的SQL语句用于管理。虽然可以使用MySQL Workbench图形界面来直接管理MySQL，但是，很多时候，通过SSH远程连接时，只能使用SQL命令，所以，了解并掌握常用的SQL管理操作是必须的。

数据库

在一个运行MySQL的服务器上，实际上可以创建多个数据库（Database）。要列出所有数据库，使用命令：

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| shici              |
| sys                |
| test               |
| school             |
+--------------------+
```

其中，`information_schema`、`mysql`、`performance_schema`和`sys`是系统库，不要去改动它们。其他的是用户创建的数据库。

要创建一个新数据库，使用命令：

```
mysql> CREATE DATABASE test;
Query OK, 1 row affected (0.01 sec)
```

要删除一个数据库，使用命令：

```
mysql> DROP DATABASE test;
Query OK, 0 rows affected (0.01 sec)
```

注意：删除一个数据库将导致该数据库的所有表全部被删除。

对一个数据库进行操作时，要首先将其切换为当前数据库：

```
mysql> USE test;
Database changed
```

表

列出当前数据库的所有表，使用命令：

```
mysql> SHOW TABLES;
+---------------------+
| Tables_in_test      |
+---------------------+
| classes             |
| statistics          |
| students            |
| students_of_class1  |
+---------------------+
```

要查看一个表的结构，使用命令：

```
mysql> DESC students;
+----------+-------------+------+-----+---------+--------
--------+
| Field    | Type         | Null | Key | Default | Extra
       |
+----------+-------------+------+-----+---------+--------
--------+
| id       | bigint(20)   | NO   | PRI | NULL    |
auto_increment |
| class_id | bigint(20)   | NO   |     | NULL    |
       |
| name     | varchar(100) | NO   |     | NULL    |
       |
| gender   | varchar(1)   | NO   |     | NULL    |
       |
| score    | int(11)      | NO   |     | NULL    |
       |
+----------+-------------+------+-----+---------+--------
--------+
5 rows in set (0.00 sec)
```

还可以使用以下命令查看创建表的SQL语句：

```
mysql> SHOW CREATE TABLE students;
+----------+-----------------------------------------------
---------+
| students | CREATE TABLE `students` (
        |
|          |   `id` bigint(20) NOT NULL AUTO_INCREMENT,
         |
|          |   `class_id` bigint(20) NOT NULL,
        |
|          |   `name` varchar(100) NOT NULL,
        |
|          |   `gender` varchar(1) NOT NULL,
        |
|          |   `score` int(11) NOT NULL,
        |
|          |   PRIMARY KEY (`id`)
         |
|          | ) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT
CHARSET=utf8 |
+----------+-----------------------------------------------
---------+
1 row in set (0.00 sec)
```

创建表使用`CREATE TABLE`语句，而删除表使用`DROP TABLE`语句：

```
mysql> DROP TABLE students;
Query OK, 0 rows affected (0.01 sec)
```

修改表就比较复杂。如果要给`students`表新增一列`birth`，使用：

```
ALTER TABLE students ADD COLUMN birth VARCHAR(10) NOT
NULL;
```

要修改`birth`列，例如把列名改为`birthday`，类型改为`VARCHAR(20)`：

```
ALTER TABLE students CHANGE COLUMN birth birthday
VARCHAR(20) NOT NULL;
```

要删除列，使用：

```
ALTER TABLE students DROP COLUMN birthday;
```

退出MySQL

使用`EXIT`命令退出MySQL：

```
mysql> EXIT
Bye
```

注意`EXIT`仅仅断开了客户端和服务器的连接，MySQL服务器仍然继续运行。

实用SQL语句

在编写SQL时，灵活运用一些技巧，可以大大简化程序逻辑。

插入或替换

如果我们希望插入一条新记录（INSERT），但如果记录已经存在，就先删除原记录，再插入新记录。此时，可以使用`REPLACE`语句，这样就不必先查询，再决定是否先删除再插入：

```
REPLACE INTO students (id, class_id, name, gender, score)
VALUES (1, 1, '小明', 'F', 99);
```

若`id=1`的记录不存在，`REPLACE`语句将插入新记录，否则，当前`id=1`的记录将被删除，然后再插入新记录。

插入或更新

如果我们希望插入一条新记录（INSERT），但如果记录已经存在，就更新该记录，此时，可以使用`INSERT INTO ... ON DUPLICATE KEY UPDATE ...`语句：

```
INSERT INTO students (id, class_id, name, gender, score)
VALUES (1, 1, '小明', 'F', 99) ON DUPLICATE KEY UPDATE
name='小明', gender='F', score=99;
```

若`id=1`的记录不存在，`INSERT`语句将插入新记录，否则，当前`id=1`的记录将被更新，更新的字段由`UPDATE`指定。

插入或忽略

如果我们希望插入一条新记录（INSERT），但如果记录已经存在，就啥事也不干直接忽略，此时，可以使用`INSERT IGNORE INTO ...`语句：

```
INSERT IGNORE INTO students (id, class_id, name, gender,
score) VALUES (1, 1, '小明', 'F', 99);
```

若`id=1`的记录不存在，`INSERT`语句将插入新记录，否则，不执行任何操作。

快照

如果想要对一个表进行快照，即复制一份当前表的数据到一个新表，可以结合`CREATE TABLE`和`SELECT`：

```
-- 对class_id=1的记录进行快照，并存储为新表students_of_class1:
CREATE TABLE students_of_class1 SELECT * FROM students
WHERE class_id=1;
```

新创建的表结构和`SELECT`使用的表结构完全一致。

写入查询结果集

如果查询结果集需要写入到表中，可以结合`INSERT`和`SELECT`，将`SELECT`语句的结果集直接插入到指定表中。

例如，创建一个统计成绩的表`statistics`，记录各班的平均成绩：

```
CREATE TABLE statistics (
    id BIGINT NOT NULL AUTO_INCREMENT,
    class_id BIGINT NOT NULL,
    average DOUBLE NOT NULL,
    PRIMARY KEY (id)
);
```

然后，我们就可以用一条语句写入各班的平均成绩：

```
INSERT INTO statistics (class_id, average) SELECT
class_id, AVG(score) FROM students GROUP BY class_id;
```

确保`INSERT`语句的列和`SELECT`语句的列能一一对应，就可以在`statistics`表中直接保存查询的结果：

```
> SELECT * FROM statistics;
+----+----------+--------------+
| id | class_id | average      |
+----+----------+--------------+
|  1 |        1 |         86.5 |
|  2 |        2 | 73.666666666 |
|  3 |        3 | 88.333333333 |
+----+----------+--------------+
3 rows in set (0.00 sec)
```

强制使用指定索引

在查询的时候，数据库系统会自动分析查询语句，并选择一个最合适的索引。但是很多时候，数据库系统的查询优化器并不一定总是能使用最优索引。如果我们知道如何选择索引，可以使用`FORCE INDEX`强制查询使用指定的索引。例如：

```
> SELECT * FROM students FORCE INDEX (idx_class_id) WHERE
class_id = 1 ORDER BY id DESC;
```

指定索引的前提是索引`idx_class_id`必须存在。