mysql 命令

> CentOS | Ubuntu | macOS
>
> Windows 需要使用 MySQL 8.0 Command Line Client
>
> https://dev.mysql.com/doc/refman/8.0/en/tutorial.html
>
> https://dev.mysql.com/doc/refman/8.0/en/data-types.html
>
> https://dev.mysql.com/doc/refman/8.0/en/sql-statements.html
>
> - 连接数据库服务器
> - 创建、选中、删除数据库
> - 创建、更新、删除表
> - 插入、查询、更新、删除数据

1. 连接到数据库服务器

使用下面的命令来连接数据库，需要指定主机名、用户名、密码。

如果连接的是本电脑上的数据库，可以省略 `-h`；如果用户名与当前登录的用户名一致，可以省略 `-u`；如果该用户未设置密码，可以省略 `-p`。

```
mysql -h host -u user -p
```

我们第一次使用时，通常只有一个 root 用户，先使用 root 用户连接到数据库，输入安装时设定的 root 密码：

```
mysql -u root -p
```

连接成功后，会打印出一段文字，并出现 `mysql>` 提示符：

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 14
Server version: 8.0.23 MySQL Community Server - GPL

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

2. 查看、创建、选中数据库与查看、创建表

当打开一个新的 MySQL 会话时，没有选中任何一个数据库。查看目前选中的数据库：

```
mysql> select database();
+------------+
| database() |
+------------+
| NULL       |
+------------+
1 row in set (0.00 sec)
```

查看所有数据库：

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)
```

在进行数据的存取操作前，需要先创建一个数据库，假设名称为 `lab01`：

```
mysql> create database lab01;
Query OK, 1 row affected (0.01 sec)

mysql> show databases;
+--------------------+
| Database           |
```

```
+--------------------+
| information_schema |
| lab01              |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)
```

创建数据库后，我们以后也可以在登录时直接指定数据库：

```
mysql -h host -u user -p dbname
```

然后选中 lab01 数据库（use 语句可以不加分号，其他绝大部分语句都要加分号）：

```
mysql> use lab01
Database changed
mysql> select database();
+------------+
| database() |
+------------+
| lab01      |
+------------+
1 row in set (0.00 sec)
```

接下来就可以进行表相关的操作了，首先查看当前数据库中的表：

```
mysql> show tables;
Empty set (0.00 sec)
```

当前数据库为空。

以第二次作业中第 2 题的 “商品 (商品编号, 名称, 类别, 单位, 单价)” 为例，创建一个名为 “商品” 的表：

```
create table 商品 (
    商品编号 char(30) primary key,
    名称 varchar(50),
    类别 varchar(30),
    单位 char(10),
    单价 dec(10, 2)
);
```

这里涉及到 MySQL 支持的数据类型，具体描述可以查看 官方文档，可以分为 5 类。

下面是一些常用语法：

- Numeric

M 表示显示长度，不是二进制位数，比如 10 表示最长为 10 位的数字，比如 1234567890，而不是 -2^9 到 2^9-1。

默认都是有符号 SIGNED，所以只需要为无符号类型指定 UNSIGNED。

ZEROFILL 表示填充 0 到指定长度 M，已经 deprecated，官方建议用 `LPAD()` 函数代替。

```
TINYINT[(M)] [UNSIGNED] [ZEROFILL]
```

1 字节整型，有符号范围 -128 到 127，无符号范围 0 到 255。

```
BOOL
BOOLEAN
```

等于 TINYINT(1)，也就是范围 -9 到 9，其中 0 是 false，其他值是 true。

```
SMALLINT[(M)] [UNSIGNED] [ZEROFILL]
```

2 字节整型，有符号范围 -32768 到 32767，无符号范围 0 到 65535。

```
MEDIUMINT[(M)] [UNSIGNED] [ZEROFILL]
```

3 字节整型。

```
INT[(M)] [UNSIGNED] [ZEROFILL]
INTEGER[(M)] [UNSIGNED] [ZEROFILL]
```

4 字节整型。

```
BIGINT[(M)] [UNSIGNED] [ZEROFILL]
```

8 字节整型。

```
DECIMAL[(M[,D])] [UNSIGNED] [ZEROFILL]
DEC[(M[,D])] [UNSIGNED] [ZEROFILL]
NUMERIC[(M[,D])] [UNSIGNED] [ZEROFILL]
FIXED[(M[,D])] [UNSIGNED] [ZEROFILL]
```

M 是所有数字的位数 (不包括负号和小数点)，D 是小数点后数字的位数；比如 -123.456 的 M 是 6，D 是 3。

M 最大可以取 65，默认是 10；D 最大可以取 30，默认是 0。

DECIMAL 的 UNSIGNED 已经 deprecated，官方建议用 CHECK 来代替。

```
FLOAT[(M,D)] [UNSIGNED] [ZEROFILL]
```

4 字节浮点型。语法 `FLOAT(M,D)` 已经 deprecated，建议直接使用 `FLOAT` (不加 M 和 D，表示单纯的 4 字节浮点数)。

FLOAT 大概能保存 7 位小数。

FLOAT 的 UNSIGNED 也是 deprecated，官方建议用 CHECK 来代替。

```
FLOAT(p) [UNSIGNED] [ZEROFILL]
```

占 p 个比特的浮点型。如果 p 在 0 到 24 之间，最终类型是 FLOAT；如果 p 在 25 到 53 之间，最终类型是 DOUBLE，可见这条语法并不能起到指定浮点数的存储空间的作用，只能从 `FLOAT` (4 字节) 和 `DOUBLE` (8 字节) 之间二选一，这条语法只是为了兼容 ODBC。所以建议直接使用 `FLOAT` 或下面的 `DOUBLE` 语法。

```
DOUBLE[(M,D)] [UNSIGNED] [ZEROFILL]
DOUBLE PRECISION[(M,D)] [UNSIGNED] [ZEROFILL]
REAL[(M,D)] [UNSIGNED] [ZEROFILL]
```

8 字节浮点型。`DOUBLE(M,D)` 也是 deprecated，直接使用 `DOUBLE` 就好。

DOUBLE 大概能保存 15 位小数。

DOUBLE 的 UNSIGNED 也是 deprecated，官方建议用 CHECK 来代替。

注：REAL 默认等于 DOUBLE，但当开启 `REAL_AS_FLOAT` 时，REAL 等于 FLOAT。

> For maximum portability, code requiring storage of approximate numeric data values should use `FLOAT` or `DOUBLE_PRECISION` with no specification of precision or number of digits.

```
BIT[(M)]
```

占 M 个比特的数值，主要用于存储二进制数值。M 可以取 1 到 64，默认是 1。

- Date and Time

  fsp 表示小数秒的位数，范围从 0 到 6，默认是 0。比如我想表示当前是 16点30分12.1234 秒，fsp 需要设置为 4。

  ```
  DATE
  ```

  日期，包含年、月、日。范围从 1000-01-01 到 9999-12-31。显示格式是 YYYY-MM-DD。

  ```
  DATETIME[(fsp)]
  ```

  日期和时间，包含年、月、日、时、分、秒以及可选的小数秒。范围从 1000-01-01 00:00:00.000000 到 9999-12-31 23:59:59.999999。显示格式是 YYYY-MM-DD hh:mm:ss[.fraction]。

  ```
  TIMESTAMP[(fsp)]
  ```

  时间戳，包含年、月、日、时、分、秒以及可选的小数秒。范围从 1970-01-01 00:00:01.000000 到 2038-01-19 03:14:07.999999。以 1970-01-01 00:00:00 以来经过的秒数来存储，但是不能表示 1970-01-01 00:00:00，因为 0 表示无效值。

```
TIME[(fsp)]
```

时间，包含时、分、秒以及可选的小数秒。范围从 -838:59:59.000000 到 838:59:59.000000。

```
YEAR[(4)]
```

四个数字的年。`YEAR(4)` 已经 deprecated，直接使用 `YEAR`。

- String

```
[NATIONAL] CHAR[(M)] [CHARACTER SET charset_name] [COLLATE
collation_name]
```

定长字符串。M 表示字符数，从 0 到 255，默认是 1。字符集默认使用 utf8。

适合用在长度固定不变的字段，比如身份证号、银行卡号、手机号；也适用于长度很短的字段，省去了 VARCHAR 用来保存长度的额外字节。

```
[NATIONAL] VARCHAR(M) [CHARACTER SET charset_name] [COLLATE
collation_name]
```

可变长度字符串。M 表示最大字符数，从 0 到 65535，但实际字符数也受每行的最大空间 (maximum row size, 65535 B) 和 使用的字符集影响 (比如 utf8 每个字符 3 B)。VARCHAR 会在数据前面用额外的 1 或 2 个字节记录字符串的长度，最大长度 <= 255 用 1 个字节，> 255 用两个字节。

适合用在长度可能变化的字段，可以节省空间。

```
TINYTEXT [CHARACTER SET charset_name] [COLLATE collation_name]
```

最长为 255 个字符的 字符串。使用 1 个字节的前缀存储长度。

```
TEXT[(M)] [CHARACTER SET charset_name] [COLLATE collation_name]
```

默认是最长为 65,535 个字符的字符串。使用 2 个字节的前缀存储长度。

如果指定了 M，会选择能满足要求的最小类型，比如 TINYTEXT、MEDIUMTEXT 等。

```
MEDIUMTEXT [CHARACTER SET charset_name] [COLLATE collation_name]
```

最长为 16,777,215 个字符的 字符串。使用 3 个字节的前缀存储长度。

```
LONGTEXT [CHARACTER SET charset_name] [COLLATE collation_name]
```

最长为 4,294,967,295 个字符的 字符串。使用 4 个字节的前缀存储长度。

TEXT 系列可以看做 4 种固定 M 的 VARCHAR，给出了一个相对粗略的最大长度，当我们不清楚字符串的最大长度时，可以使用 TEXT 进行大概的设定；TEXT 也支持更长的字符串 (MEDIUMTEXT、LONGTEXT)。

这里还涉及到建表语句的语法，可以查看官方文档，目前我们用到的主要有 `primary key`、`foreign key`。

查看目前存在的表以及 “商品” 表的描述：

```
mysql> show tables;
+-----------------+
| Tables_in_lab01 |
+-----------------+
| 商品            |
+-----------------+
1 row in set (0.00 sec)

mysql> describe 商品;
+--------------+---------------+------+-----+---------+-------+
| Field        | Type          | Null | Key | Default | Extra |
+--------------+---------------+------+-----+---------+-------+
| 商品编号     | char(30)      | NO   | PRI | NULL    |       |
| 名称         | varchar(50)   | YES  |     | NULL    |       |
| 类别         | varchar(30)   | YES  |     | NULL    |       |
| 单位         | char(10)      | YES  |     | NULL    |       |
| 单价         | decimal(10,2) | YES  |     | NULL    |       |
+--------------+---------------+------+-----+---------+-------+
5 rows in set (0.02 sec)
```

再创建两个实体 “营业员” 和 “门店” 以及关系 “销售” 的表：

```
create table 门店 (
    门店编号 char(30) primary key,
    名称 varchar(50),
    地址 varchar(100)
);

create table 营业员 (
    营业员编号 char(30) primary key,
    姓名 varchar(50),
    业绩 dec(10, 2),
    所属门店编号 char(30),
    foreign key (所属门店编号)
        references 门店 (门店编号)
        on update cascade
);

create table 销售记录 (
    销售单号 char(30) primary key,
    营业员编号 char(30),
    门店编号 char(30),
    商品编号 char(30),
```

```
    数量 int,
    日期 date,
    foreign key (营业员编号)
        references 营业员 (营业员编号)
        on update cascade,
    foreign key (门店编号)
        references 门店 (门店编号)
        on update cascade,
    foreign key (商品编号)
        references 商品 (商品编号)
        on update cascade
);
```

这里 “营业员” 表的定义中用到了外键，外键的语法如下：

```
[CONSTRAINT [symbol]] FOREIGN KEY
    [index_name] (col_name, ...)
    REFERENCES tbl_name (col_name,...)
    [ON DELETE reference_option]
    [ON UPDATE reference_option]

reference_option:
    RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT
```

最后两行的 on delete 和 on update 可以指定在被参照表的行被更新或删除时，参照表（当前定义的表）的相关行做如何操作：

- cascade：做相同的更新或删除
- set null：把外键字段设为 NULL
- restrict: 拒绝更新或删除（默认）
- no action: 同 restrict，SQL 标准

3. 插入、查询、更新、删除数据

用 INSERT 语句添加一些数据，INSERT 的文档：

```
insert into 商品
values ('00001', '农夫山泉', '矿泉水', '元', '2.00');

insert into 商品
values ('00012', '怡宝', '矿泉水', '元', '2.50');

insert into 商品
values ('00123', '数值分析', '书籍', '元', '35.00');

insert into 商品
values ('01234', 'C++ Primer', '书籍', '元', '128.00');

insert into 商品
```

```
values ('12345', '树莓派4B', '数码产品', '元', '300.00');

insert into 门店
values ('00001', '北航优购', '合一楼一层');

insert into 营业员
values ('00001', '营业员1号', '350.00', '00001');

insert into 销售记录
values ('00001', '00001', '00001', '00001', 3, '2020-3-1');

insert into 销售记录
values ('00002', '00001', '00001', '00012', 5, '2020-3-3');

insert into 销售记录
values ('00003', '00001', '00001', '00123', 4, '2020-3-5');

insert into 销售记录
values ('00004', '00001', '00001', '01234', 2, '2020-3-1');
```

用 SELECT 语句查看 “商品” 表的所有数据，SELECT 的文档:

```
mysql> select * from 商品;
+--------------+--------------+--------------+--------+--------+
| 商品编号     | 名称         | 类别         | 单位   | 单价   |
+--------------+--------------+--------------+--------+--------+
| 00001        | 农夫山泉     | 矿泉水       | 元     |   2.00 |
| 00012        | 怡宝         | 矿泉水       | 元     |   2.50 |
| 00123        | 数值分析     | 书籍         | 元     |  35.00 |
| 01234        | C++ Primer   | 书籍         | 元     | 128.00 |
| 12345        | 树莓派4B     | 数码产品     | 元     | 300.00 |
+--------------+--------------+--------------+--------+--------+
5 rows in set (0.00 sec)
```

查询所有书籍的编号和名称:

```
mysql> select 商品编号, 名称 from 商品 where 类别 = '书籍';
+--------------+--------------+
| 商品编号     | 名称         |
+--------------+--------------+
| 00123        | 数值分析     |
| 01234        | C++ Primer   |
+--------------+--------------+
2 rows in set (0.00 sec)
```

查询价格 > 2元的矿泉水:

```
mysql> select * from 商品 where 类别 = '矿泉水' and 单价 > 2;
+--------------+--------+-----------+--------+--------+
| 商品编号      | 名称    | 类别       | 单位    | 单价    |
+--------------+--------+-----------+--------+--------+
| 00012        | 怡宝    | 矿泉水     | 元      |   2.50 |
+--------------+--------+-----------+--------+--------+
1 row in set (0.00 sec)
```

查询所有矿泉水和数码产品：

```
mysql> select * from 商品 where 类别 = '矿泉水' or 类别 = '数码产品';
+--------------+--------------+--------------+--------+--------+
| 商品编号      | 名称          | 类别          | 单位    | 单价    |
+--------------+--------------+--------------+--------+--------+
| 00001        | 农夫山泉      | 矿泉水        | 元      |   2.00 |
| 00012        | 怡宝          | 矿泉水        | 元      |   2.50 |
| 12345        | 树莓派4B      | 数码产品      | 元      | 300.00 |
+--------------+--------------+--------------+--------+--------+
3 rows in set (0.00 sec)
```

将所有商品按单价升序和降序排列：

```
mysql> select * from 商品 order by 单价;
+--------------+--------------+--------------+--------+--------+
| 商品编号      | 名称          | 类别          | 单位    | 单价    |
+--------------+--------------+--------------+--------+--------+
| 00001        | 农夫山泉      | 矿泉水        | 元      |   2.00 |
| 00012        | 怡宝          | 矿泉水        | 元      |   2.50 |
| 00123        | 数值分析      | 书籍          | 元      |  35.00 |
| 01234        | C++ Primer    | 书籍          | 元      | 128.00 |
| 12345        | 树莓派4B      | 数码产品      | 元      | 300.00 |
+--------------+--------------+--------------+--------+--------+
5 rows in set (0.00 sec)

mysql> select * from 商品 order by 单价 desc;
+--------------+--------------+--------------+--------+--------+
| 商品编号      | 名称          | 类别          | 单位    | 单价    |
+--------------+--------------+--------------+--------+--------+
| 12345        | 树莓派4B      | 数码产品      | 元      | 300.00 |
| 01234        | C++ Primer    | 书籍          | 元      | 128.00 |
| 00123        | 数值分析      | 书籍          | 元      |  35.00 |
| 00012        | 怡宝          | 矿泉水        | 元      |   2.50 |
| 00001        | 农夫山泉      | 矿泉水        | 元      |   2.00 |
+--------------+--------------+--------------+--------+--------+
5 rows in set (0.00 sec)
```

使用正则匹配进行查询，稍微详细一点的介绍可以看这里：

```
mysql> select * from 商品 where 名称 like '%山泉';
+--------------+--------------+-----------+--------+--------+
| 商品编号      | 名称          | 类别       | 单位    | 单价    |
+--------------+--------------+-----------+--------+--------+
| 00001        | 农夫山泉      | 矿泉水     | 元      |   2.00 |
+--------------+--------------+-----------+--------+--------+
1 row in set (0.01 sec)


mysql> select * from 商品 where 名称 like '__';
+--------------+--------+-----------+--------+--------+
| 商品编号      | 名称    | 类别       | 单位    | 单价    |
+--------------+--------+-----------+--------+--------+
| 00012        | 怡宝    | 矿泉水     | 元      |   2.50 |
+--------------+--------+-----------+--------+--------+
1 row in set (0.00 sec)
```

对行进行计数：

```
mysql> select count(*) from 商品;
+----------+
| count(*) |
+----------+
|        5 |
+----------+
1 row in set (0.01 sec)

mysql> select 类别, count(*) from 商品 group by 类别;
+--------------+----------+
| 类别          | count(*) |
+--------------+----------+
| 矿泉水        |        2 |
| 书籍          |        2 |
| 数码产品      |        1 |
+--------------+----------+
3 rows in set (0.01 sec)
```

使用多表连接查询在 2020 年 3 月 1 日出售的商品名称、单价、单位、数量：

```
mysql> select t1.名称, t1.单价, t1.单位, t2.数量
    -> from 商品 as t1 inner join 销售记录 as t2
    -> on t1.商品编号 = t2.商品编号
    -> where t2.日期 = '2020-3-1';
+--------------+--------+--------+--------+
| 名称         | 单价   | 单位   | 数量   |
+--------------+--------+--------+--------+
| 农夫山泉     |   2.00 | 元     |      3 |
| C++ Primer   | 128.00 | 元     |      2 |
+--------------+--------+--------+--------+
2 rows in set (0.00 sec)
```

用 update 语句更新一下 “C++ Primer” 的单价：

```
mysql> update 商品 set 单价 = '110.00' where 名称 = 'C++ Primer';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from 商品 where 名称 = 'C++ Primer';
+--------------+------------+--------+--------+--------+
| 商品编号     | 名称       | 类别   | 单位   | 单价   |
+--------------+------------+--------+--------+--------+
| 01234        | C++ Primer | 书籍   | 元     | 110.00 |
+--------------+------------+--------+--------+--------+
1 row in set (0.00 sec)
```

删除 2020 年 3 月 3 日 的销售记录：

```
mysql> delete from 销售记录 where 日期 = '2020-3-3';
Query OK, 1 row affected (0.00 sec)

mysql> select * from 销售记录;
+--------------+----------------+--------------+--------------+--------+------
------+
| 销售单号     | 营业员编号     | 门店编号     | 商品编号     | 数量   | 日期       |
+--------------+----------------+--------------+--------------+--------+------
------+
| 00001        | 00001          | 00001        | 00001        |      3 | 2020-
03-01 |
| 00003        | 00001          | 00001        | 00123        |      4 | 2020-
03-05 |
| 00004        | 00001          | 00001        | 01234        |      2 | 2020-
03-01 |
+--------------+----------------+--------------+--------------+--------+------
------+
3 rows in set (0.00 sec)
```

4. 更新、删除表，删除数据库

为 “门店” 表添加一个字段 “联系方式”，alter table 的文档：

```
mysql> alter table 门店 add 联系方式 char(20);
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> describe 门店;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| 门店编号     | char(30)     | NO   | PRI | NULL    |       |
| 名称         | varchar(50)  | YES  |     | NULL    |       |
| 地址         | varchar(100) | YES  |     | NULL    |       |
| 联系方式     | char(20)     | YES  |     | NULL    |       |
+--------------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

删除门店表：

```
mysql> drop table 门店;
ERROR 3730 (HY000): Cannot drop table '门店' referenced by a foreign key
constraint '营业员_ibfk_1' on table '营业员'.
```

由于外键默认是 restrict，拒绝了删除。可以先删除关于门店的外键。

```
mysql> show create table 销售记录\G
*************************** 1. row ***************************
       Table: 销售记录
Create Table: CREATE TABLE `销售记录` (
  `销售单号` char(30) NOT NULL,
  `营业员编号` char(30) DEFAULT NULL,
  `门店编号` char(30) DEFAULT NULL,
  `商品编号` char(30) DEFAULT NULL,
  `数量` int DEFAULT NULL,
  `日期` date DEFAULT NULL,
  PRIMARY KEY (`销售单号`),
  KEY `营业员编号` (`营业员编号`),
  KEY `门店编号` (`门店编号`),
  KEY `商品编号` (`商品编号`),
  CONSTRAINT `销售记录_ibfk_1` FOREIGN KEY (`营业员编号`) REFERENCES `营业员` (`营业
员编号`) ON UPDATE CASCADE,
  CONSTRAINT `销售记录_ibfk_2` FOREIGN KEY (`门店编号`) REFERENCES `门店` (`门店编号
`) ON UPDATE CASCADE,
  CONSTRAINT `销售记录_ibfk_3` FOREIGN KEY (`商品编号`) REFERENCES `商品` (`商品编号
`) ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

```
1 row in set (0.00 sec)

mysql> alter table 销售记录 drop foreign key 销售记录_ibfk_2;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> show create table 营业员\G
*************************** 1. row ***************************
       Table: 营业员
Create Table: CREATE TABLE `营业员` (
  `营业员编号` char(30) NOT NULL,
  `姓名` varchar(50) DEFAULT NULL,
  `业绩` decimal(10,2) DEFAULT NULL,
  `所属门店编号` char(30) DEFAULT NULL,
  PRIMARY KEY (`营业员编号`),
  KEY `所属门店编号` (`所属门店编号`),
  CONSTRAINT `营业员_ibfk_1` FOREIGN KEY (`所属门店编号`) REFERENCES `门店` (`门店编
号`) ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
1 row in set (0.00 sec)

mysql> alter table 营业员 drop foreign key 营业员_ibfk_1;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> drop table 门店;
Query OK, 0 rows affected (0.01 sec)
```

删除数据库 lab01：

```
mysql> drop database lab01;
Query OK, 3 rows affected (0.03 sec)
```

> 也可以把语句写到一个文件中，使用 mysql 命令批量执行，具体可以看这个页面。

最后使用 \q 或 quit 命令退出 mysql 命令：

```
mysql> \q
Bye
```