数据库 - 实验报告

基于Linux下的Mysql

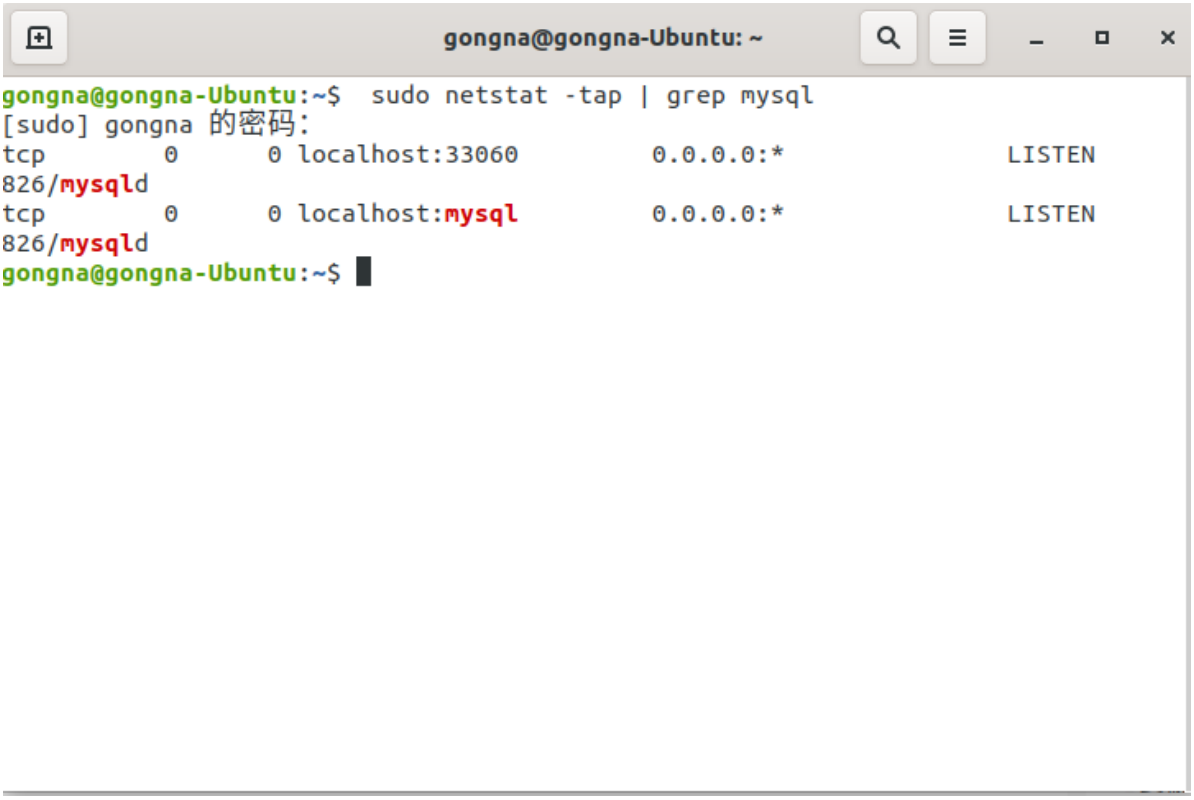
指导老师:
喻莹

学生：龚娜 学号：2020213760

实验一:数据库准备⭐

1.首先检查系统中是否已经安装了MySQL

```
sudo netstat -tap | grep mysql
```



如上图所：证明已经安装好了mysql

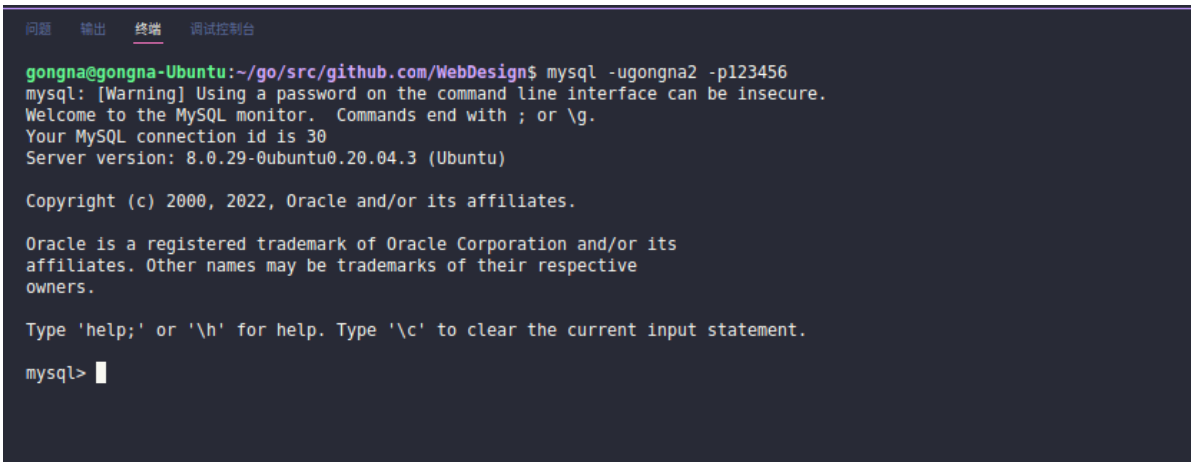
2.如果没有安装，则安装MySQL

在终端输入安装：

```
sudo apt-get install mysql-server mysql-client
```

登录MySQ：

```
mysql -ugongna2  -p123456
```



- `gongna2`是用户，默认为`root`
- -p后面是密码 `-p123456`

修改访问权限的操作:

```
进入mysql,输入如下命令，输入密码，进入mysql命令行
mysql -u root -p

grant all privileges on *.* to gongna2@"%"identified by"123456"with grant option;

flush privileges;

exit;
```

重启mysql服务

```
service mysql restart
```

实验二：数据库管理👩

1. MySQL用户设置

- 检查MySQL服务器是否启动

  ```
  ps -ef | grep mysqld
  ```

- MySQL 用户设置

- 以root用户进入

```
gongna@gongna-Ubuntu:~/go/src/github.com/WebDesign$ sudo mysql -u root -p
[sudo] gongna 的密码：
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 31
Server version: 8.0.29-0ubuntu0.20.04.3 (Ubuntu)

Copyright (c) 2000, 2022, 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>
```

- 使用mysql
-

```
mysql> use mysql;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql>
```

- 列出databases
-

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| Test               |
| ccnu               |
| firstdb            |
| information_schema |
| library            |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
8 rows in set (0.00 sec)

mysql>
```

- 列出所有的表

```
mysql> show tables;
+-----------------------------------------------------------+
| Tables_in_mysql                                           |
+-----------------------------------------------------------+
| columns_priv                                              |
| component                                                 |
| db                                                        |
| default_roles                                             |
| engine_cost                                               |
| func                                                      |
| general_log                                               |
| global_grants                                             |
| gtid_executed                                             |
| help_category                                             |
| help_keyword                                              |
| help_relation                                             |
| help_topic                                                |
| innodb_index_stats                                        |
| innodb_table_stats                                        |
| password_history                                          |
| plugin                                                    |
| procs_priv                                                |
| proxies_priv                                              |
| replication_asynchronous_connection_failover              |
| replication_asynchronous_connection_failover_managed      |
| replication_group_configuration_version                   |
| replication_group_member_actions                          |
| role_edges                                                |
| server_cost                                               |
| servers                                                   |
| slave_master_info                                         |
| slave_relay_log_info                                      |
| slave_worker_info                                         |
| slow_log                                                  |
| tables_priv                                               |
| time_zone                                                 |
| time_zone_leap_second                                     |
| time_zone_name                                            |
| time_zone_transition                                      |
| time_zone_transition_type                                 |
| user                                                      |
+-----------------------------------------------------------+
37 rows in set (0.00 sec)
```

- 如果想要新建一个用户就要往user 表格里面插入用户
- 查找是否有 `gongna2` 这个用户。

```
mysql> SELECT Host, User  FROM user WHERE user = 'gongna2';
+------+---------+
| Host | User    |
+------+---------+
| %    | gongna2 |
+------+---------+
1 row in set (0.00 sec)

mysql>
```

- 查询和这个用户相关的其他的信息

```
mysql> SELECT Host, User password_expired  FROM user WHERE user = 'gongna2';
+------+-----------------+
| Host | password_expired |
+------+-----------------+
| %    | gongna2         |
+------+-----------------+
1 row in set (0.00 sec)

mysql> SELECT Host, User, password_expired FROM user WHERE user = 'gongna2';
+------+---------+------------------+
| Host | User    | password_expired |
+------+---------+------------------+
| %    | gongna2 | N                |
+------+---------+------------------+
1 row in set (0.00 sec)

mysql> SELECT Host, User ,Password_require_current  FROM user WHERE user = 'gongna2';
+------+---------+--------------------------+
| Host | User    | Password_require_current |
+------+---------+--------------------------+
| %    | gongna2 | NULL                     |
+------+---------+--------------------------+
1 row in set (0.00 sec)

mysql>
```

- MySQL提供的 PASSWORD() 函数来对密码进行加密。

```
mysql> SELECT Host, User ,authentication_string FROM user WHERE user = 'gongna2';
+------+---------+-------------------------------------------------------------------------+
| Host | User    | authentication_string                                                   |
+------+---------+-------------------------------------------------------------------------+
| %    | gongna2 | $A$005$a!X;!3m0.M|jl
                                uZa2XE7Y5IhUxOVGajgOWnNUOl6MzRUHCLpGUWy8m84o32 |
+------+---------+-------------------------------------------------------------------------+
1 row in set (0.00 sec)

mysql>
```

- 另外一种添加用户的方法为通过SQL的 GRANT 命令，以下命令会给指定数据库TUTORIALS添加用户 test 密码为 123 。

```
mysql -u root -p
Enter password:*******
mysql> use mysql;
Database changed

mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
    -> ON TUTORIALS.*
    -> TO 'test'@'localhost'
    -> IDENTIFIED BY '123';
```

2,MySQL的命令

USE

选择要操作的Mysql数据库，使用该命令后所有Mysql命令都只针对该数据库

```
mysql> use library;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-------------------+
| Tables_in_library |
+-------------------+
| books             |
+-------------------+
1 row in set (0.00 sec)
```

SHOW DATABASES

列出 MySQL 数据库管理系统的数据库列表。

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| Test               |
| ccnu               |
| firstdb            |
| information_schema |
| library            |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
8 rows in set (0.01 sec)
```

SHOW TABLES

```
mysql> show tables;
+-----------------+
| Tables_in_Test  |
+-----------------+
| book_table      |
| borrow_table    |
| librarian_table |
| reader_table    |
+-----------------+
4 rows in set (0.01 sec)
```

SHOW COLUMNS FROM 表

```
mysql> show columns  from book_table;
+------------------+--------------+------+-----+---------+----------------+
| Field            | Type         | Null | Key | Default | Extra          |
+------------------+--------------+------+-----+---------+----------------+
| id               | bigint       | NO   | PRI | NULL    | auto_increment |
| book_id          | varchar(20)  | YES  | UNI | NULL    |                |
| book_name        | varchar(20)  | NO   |     | NULL    |                |
| book_information | varchar(100) | NO   |     | NULL    |                |
| book_price       | int          | NO   |     | NULL    |                |
| book_state       | int          | NO   |     | NULL    |                |
+------------------+--------------+------+-----+---------+----------------+
6 rows in set (0.01 sec)
```

SHOW INDEX FROM 表

```
mysql> show index from book_table;
+------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table      | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| book_table |          0 | PRIMARY  |            1 | id          | A         |           4 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| book_table |          0 | book_id  |            1 | book_id     | A         |           4 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
+------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
2 rows in set (0.01 sec)
```

SHOW TABLE STATUS FROM 数据库

```
mysql> show table status from Test;
+-----------------+--------+---------+------------+------+----------------+-------------+-----------------+--------------+-----------+----------------+---------------------+---------------------+---------------------+-----------+----------------------+---------+
| Name            | Engine | Version | Row_format | Rows | Avg_row_length | Data_length | Max_data_length | Index_length | Data_free | Auto_increment | Create_time         | Update_time         | Check_time          | Collation | Checksum | Create_options | Comment |
+-----------------+--------+---------+------------+------+----------------+-------------+-----------------+--------------+-----------+----------------+---------------------+---------------------+---------------------+-----------+----------+----------------+---------+
| book_table      | InnoDB |      10 | Dynamic    |    5 |           3276 |       16384 |               0 |        16384 |         0 |              6 | 2022-04-21 20:36:05 | 2022-06-04 11:53:16 | NULL                | utf8mb4_0900_ai_ci |     NULL |                |         |
| borrow_table    | InnoDB |      10 | Dynamic    |   14 |           1170 |       16384 |               0 |            0 |         0 |             18 | 2022-04-21 20:36:05 | NULL                | NULL                | utf8mb4_0900_ai_ci |     NULL |                |         |
| librarian_table | InnoDB |      10 | Dynamic    |    0 |              0 |       16384 |               0 |        16384 |         0 |              1 | 2022-04-21 20:36:05 | NULL                | NULL                | utf8mb4_0900_ai_ci |     NULL |                |         |
| reader_table    | InnoDB |      10 | Dynamic    |    0 |              0 |       16384 |               0 |        16384 |         0 |              2 | 2022-04-21 20:36:05 | NULL                | NULL                | utf8mb4_0900_ai_ci |     NULL |                |         |
+-----------------+--------+---------+------------+------+----------------+-------------+-----------------+--------------+-----------+----------------+---------------------+---------------------+---------------------+-----------+----------+----------------+---------+
4 rows in set (0.01 sec)
```

实验三 数据库实战🌒

掌握DDL

> **DDL（Data Definition Languages）语句**
>
> 数据定义语言，这些语句定义了不同的数据段、数据库、表、列、索引等数据库对象的定义。常用的语句关键字主要包括 create、drop、alter等

1.CREATE DATABASE 数据库

create 命令创建数据库

2.DROP DATABASE 数据库

drop 命令删除数据库

3.USE 数据库

在 `mysql>` 提示窗口中可以很简单的选择特定的数据库。你可以使用Use命令来选择指定的数据库。

4.CREATE TABLE IF NOT EXISTS 数据库

创建MySQL数据表的SQL通用语法

5.DROP TABLE 表

在`mysql>`命令提示窗口中删除数据表SQL语句为 **DROP TABLE**

掌握DML

> **DML（Data Manipulation Language）语句**
> 数据操纵语句，用于添加、删除、更新和查询数据库记录，并检查数据完整性，常用的语句关键字主要包括 insert、delete、udpate 和select 等。(增添改查）

1.NSERT INTO 表

MySQL 表中使用 **INSERT INTO** SQL语句来插入数据。

2.DELETE FROM 表 WHERE 列=

你可以使用 SQL 的 DELETE FROM 命令来删除 MySQL 数据表中的记录。

3.UPDATE 表 SET 列1 =，列2 =

如果我们需要修改或更新 MySQL 中的数据，我们可以使用 SQL UPDATE 命令来操作。

4.SELECT 列1，列2 FROM 表

MySQL 数据库使用SQL SELECT语句来查询数据。

掌握DCL

> **DCL（Data Control Language）语句**
>
> 数据控制语句，用于控制不同数据段直接的许可和访问级别的语句。这些语句定义了数据库、表、字段、用户的访问权限和安全级别。主要的语句关键字包括 grant、revoke 等

异同

- DDL 是对数据库内部的对象进行创建、删除、修改的操作语言。
- DML 只是对表内部数据的操作，而不涉及到表的定义、结构的修改，更不会涉及到其他对象。
- DCL 语句更多的被数据库管理员（DBA）所使用，一般开发人员很少使用。

实验四：Transact-SQL语言🔥

mysql内置函数的使用:

MySQL 字符串函数

- ASCII(s) 返回返回某个字符串字段第一个字母的 ASCII 码:

```
mysql> select Ascii(book_information) as NumberCodeOfFirstChar from book_table;
+----------------------+
| NumberCodeOfFirstChar |
+----------------------+
|                  100 |
|                  101 |
|                   56 |
|                   48 |
|                   50 |
+----------------------+
5 rows in set (0.01 sec)
```

- CHAR_LENGTH(s) 返回某个字符串字段的字符数

```
mysql> select char_length(book_information) as lengthOfString from book_table;
+----------------+
| lengthOfString |
+----------------+
|              5 |
|              2 |
|              3 |
|              4 |
|              7 |
+----------------+
5 rows in set (0.00 sec)

mysql>
```

- CONCAT(s1,s2...sn) 返回多个字符串字段合并后的字符串

```
mysql> select concat(book_information) as conectedString  from book_table;
\+----------------+
| conectedString |
+----------------+
| dager          |
| er             |
| 888            |
| 0000           |
| 2383628        |
+----------------+
5 rows in set (0.01 sec)
```

- FORMAT(x,n) 函数可以将某个数字字段x 进行格式化。将 x 保留到小数点后 n 位，最后一位四舍五入

```
mysql> select format(book_price,2) as price  from book_table;
+--------+
| price  |
+--------+
| 56.00  |
| 77.00  |
| 67.00  |
| 90.00  |
| 428.00 |
+--------+
5 rows in set (0.01 sec)

mysql>
```

- INSERT(s1,x,len,s2) 返回某个字符串字段下字符串 s2 替换 s1 的 x 位置开始长度为 len 的字符串。

```
mysql> select insert(book_information,0,4,"info") as changedBookInfo  from book_table;
+-----------------+
| changedBookInfo |
+-----------------+
| dager           |
| er              |
| 888             |
| 0000            |
| 2383628         |
+-----------------+
5 rows in set (0.00 sec)

mysql> select insert(book_information,1,4,"info") as changedBookInfo  from book_table;
+-----------------+
| changedBookInfo |
+-----------------+
| infor           |
| info            |
| info            |
| info            |
| info628         |
+-----------------+
5 rows in set (0.00 sec)
```

- LEFT(s,n) 返回某个字符串字段字符串中的前两个字符：

```
mysql> select left(book_information) as LeftInfo  from book_table;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that correspo
nds to your MySQL server version for the right syntax to use near ') as LeftInfo  from b
ook_table' at line 1
mysql> select left(book_information,4) as LeftInfo  from book_table;
+----------+
| LeftInfo |
+----------+
| dage     |
| er       |
| 888      |
| 0000     |
| 2383     |
+----------+
5 rows in set (0.01 sec)
```

- REVERSE(s) 返回某个字符串字段字符串反过来的顺序

```
mysql> select left(book_information,4) as LeftInfo  from book_table;
+----------+
| LeftInfo |
+----------+
| dage     |
| er       |
| 888      |
| 0000     |
| 2383     |
+----------+
5 rows in set (0.01 sec)

mysql> select reverse(book_information) as ReverseString  from book_table;
+---------------+
| ReverseString |
+---------------+
| regad         |
| re            |
| 888           |
| 0000          |
| 8263832       |
+---------------+
5 rows in set (0.00 sec)

mysql>
```

mysql数字函数

- ABS(x) 返回 x 的绝对值

```
mysql> select abs(book_price) as AbsPrice  from book_table;
+----------+
| AbsPrice |
+----------+
|       56 |
|       77 |
|       67 |
|       90 |
|      428 |
+----------+
5 rows in set (0.01 sec)
```

- ACOS(x) 求 x 的反余弦值

```
mysql> select acos(book_price) as ACOSPrice  from book_table;
+-----------+
| ACOSPrice |
+-----------+
|      NULL |
|      NULL |
|      NULL |
|      NULL |
|      NULL |
+-----------+
5 rows in set (0.00 sec)
```

- AVG(expression) 返回一个表达式的平均值，expression 是一个字段

```
mysql> select avg(book_price) as AvgPrice  from book_table;
+----------+
| AvgPrice |
+----------+
| 143.6000 |
+----------+
1 row in set (0.00 sec)

mysql>
```

- GREATEST(expr1, expr2, expr3, ...) 返回以下数字列表中的最大值：

```
mysql> select greatest(book_price,id) as BigNumber  from book_table;
+-----------+
| BigNumber |
+-----------+
|        56 |
|        77 |
|        67 |
|        90 |
|       428 |
+-----------+
5 rows in set (0.01 sec)

mysql>
```

- LEAST(expr1, expr2, expr3, ...) 返回以下数字列表中的最小值：

```
mysql> select least(book_price,id) as LeastNumber  from book_table;
+-------------+
| LeastNumber |
+-------------+
|           1 |
|           2 |
|           3 |
|           4 |
|           5 |
+-------------+
5 rows in set (0.00 sec)

mysql>
```

- MOD(x,y) 返回 x 除以 y 以后的余数

```
mysql> select mod(book_price,id) as ModNumber  from book_table;
+-----------+
| ModNumber |
+-----------+
|         0 |
|         1 |
|         1 |
|         2 |
|         3 |
+-----------+
5 rows in set (0.01 sec)

mysql>
```

mysql日期函数

- CURDATE()　返回当前日期

```
mysql> select curdate()  as Date;
+------------+
| Date       |
+------------+
| 2022-06-06 |
+------------+
1 row in set (0.02 sec)

mysql>
```

- CURRENT_TIME （） 返回当前时间

```
mysql> select current_time()  as CurrentDate;
+-------------+
| CurrentDate |
+-------------+
| 15:05:36    |
+-------------+
1 row in set (0.00 sec)

mysql>
```

MySQL 高级函数

- BIN(x) 返回 x 的二进制编码

```
mysql> select bin(book_price)  as BinPrice from book_table;
+-----------+
| BinPrice  |
+-----------+
| 111000    |
| 1001101   |
| 1000011   |
| 1011010   |
| 110101100 |
+-----------+
5 rows in set (0.00 sec)

mysql>
```

- DATABASE() 返回当前数据库名

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

mysql>
```

- ISNULL(expression) 判断表达式是否为 NULL

```
mysql> select book_price  from book_table where id=3;
+------------+
| book_price |
+------------+
|         67 |
+------------+
1 row in set (0.00 sec)

mysql> select isNull(book_price)  from book_table where id=3;
+--------------------+
| isNull(book_price) |
+--------------------+
|                  0 |
+--------------------+
1 row in set (0.00 sec)

mysql>
```

- SYSTEM_USER()

```
mysql> select system_user();
+--------------------+
| system_user()      |
+--------------------+
| gongna2@localhost |
+--------------------+
1 row in set (0.00 sec)

mysql>
```

- USER() 返回当前用户

```
mysql> select user();
+--------------------+
| user()             |
+--------------------+
| gongna2@localhost |
+--------------------+
1 row in set (0.00 sec)

mysql>
```

mysql的操作符：

UNION 操作符

MySQL UNION 操作符用于连接两个以上的 SELECT 语句的结果组合到一个结果集合中。多个 SELECT 语句会删除重复的数据。

```
+----+---------+-----------+
| id | book_id | reader_id |
+----+---------+-----------+
|  1 |         |        56 |
|  2 |         |        56 |
|  3 |         |        56 |
|  4 | 7777    |        56 |
|  5 | 7777    |        56 |
|  6 | 7777    |        56 |
|  7 | 7777    |        56 |
|  8 | 7777    |        56 |
|  9 | 7777    |        56 |
| 10 | 7777    |        56 |
| 11 | 7777    |        56 |
| 12 | 7777    |        56 |
| 13 | 7777    |        56 |
| 14 | 7777    |        56 |
| 15 | 7777    |        56 |
| 16 | 11      |        56 |
| 17 | 00      |        56 |
+----+---------+-----------+
17 rows in set (0.01 sec)

mysql> select id from borrow_table union select id from book_table;
+----+
| id |
+----+
|  1 |
|  2 |
|  3 |
|  4 |
|  5 |
|  6 |
|  7 |
|  8 |
|  9 |
| 10 |
| 11 |
| 12 |
| 13 |
| 14 |
| 15 |
| 16 |
| 17 |
+----+
17 rows in set (0.00 sec)

mysql> select *from reader_table;
+----+-----------+----------+
| id | reader_id | password |
+----+-----------+----------+
|  1 |        56 | 78       |
+----+-----------+----------+
1 row in set (0.00 sec)

mysql> select  reader_id from reader_table union select reader_id from borrow_table;
+-----------+
| reader_id |
+-----------+
|        56 |
+-----------+
1 row in set (0.00 sec)
```

```
mysql> select  reader_id from reader_table union all select reader_id from borrow_table;

+-----------+
| reader_id |
+-----------+
|        56 |
|        56 |
|        56 |
|        56 |
|        56 |
|        56 |
|        56 |
|        56 |
|        56 |
|        56 |
|        56 |
|        56 |
|        56 |
|        56 |
|        56 |
|        56 |
|        56 |
|        56 |
+-----------+
18 rows in set (0.00 sec)
```

LIKE 子句

WHERE 子句中使用 SQL LIKE 子句。

SQL LIKE 子句中使用百分号 **%**字符来表示任意字符

```
mysql> select  *from book_table;
+----+---------+-----------+------------------+------------+------------+
| id | book_id | book_name | book_information | book_price | book_state |
+----+---------+-----------+------------------+------------+------------+
|  1 | 3333    | dsa       | dager            |         56 |        200 |
|  2 | 7777    | ds        | er               |         77 |        400 |
|  3 | 11      | gh        | 888              |         67 |        200 |
|  4 | 00      | yh        | 0000             |         90 |        400 |
|  5 | 80      | 80        | 2383628          |        428 |        200 |
+----+---------+-----------+------------------+------------+------------+
5 rows in set (0.00 sec)

mysql> select  book_information from book_table where book_information like "%8%";
+------------------+
| book_information |
+------------------+
| 888              |
| 2383628          |
+------------------+
2 rows in set (0.00 sec)

mysql>
```

mysql的事务：

MySQL 事务主要用于处理操作量大，复杂度高的数据。比如说，在人员管理系统中，你删除一个人员，你既需要删除人员的基本资料，也要删除和该人员相关的信息，如信箱，文章等等，这样，这些数据库操作语句就构成一个事务

MYSQL 事务处理主要有两种方法：

- 第一种：
  - **BEGIN** 开始一个事务
  - **ROLLBACK** 事务回滚
  - **COMMIT** 事务确认

- 第二种:
  - **SET AUTOCOMMIT=0** 禁止自动提交
  - **SET AUTOCOMMIT=1** 开启自动提交

回滚:

```
mysql> begin;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into reader_id,password value("testAccount","123456");
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that correspo
nds to your MySQL server version for the right syntax to use near ',password value("test
Account","123456")' at line 1
mysql> insert into (reader_id,password) value("testAccount","123456") from reader_table;

ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that correspo
nds to your MySQL server version for the right syntax to use near '(reader_id,password)
value("testAccount","123456") from reader_table' at line 1
mysql> show columns from reader_table;
+-----------+-------------+------+-----+---------+----------------+
| Field     | Type        | Null | Key | Default | Extra          |
+-----------+-------------+------+-----+---------+----------------+
| id        | bigint      | NO   | PRI | NULL    | auto_increment |
| reader_id | bigint      | YES  | UNI | NULL    |                |
| password  | varchar(20) | NO   |     | NULL    |                |
+-----------+-------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)

mysql> insert into reader_table (reader_id,password) value(2022,"123456");
Query OK, 1 row affected (0.00 sec)

mysql> rollback;
Query OK, 0 rows affected (0.01 sec)

mysql> select  *from reader_table;
+----+-----------+----------+
| id | reader_id | password |
+----+-----------+----------+
|  1 |        56 | 78       |
+----+-----------+----------+
1 row in set (0.00 sec)

mysql>
```

成功提交:

```
mysql> select  *from reader_table;
+----+-----------+----------+
| id | reader_id | password |
+----+-----------+----------+
|  1 |        56 | 78       |
+----+-----------+----------+
1 row in set (0.00 sec)

mysql> begin;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into reader_table (reader_id,password) value(2022,"123456");
Query OK, 1 row affected (0.00 sec)

mysql> commit;
Query OK, 0 rows affected (0.01 sec)

mysql> select  *from reader_table;
+----+-----------+----------+
| id | reader_id | password |
+----+-----------+----------+
|  1 |        56 | 78       |
|  3 |      2022 | 123456   |
+----+-----------+----------+
2 rows in set (0.00 sec)

mysql>
```

实验五：安全管理😄

基础实验：

①掌握创建登录名、用户操作。

- 如果想要新建一个用户就要往user 表格里面插入用户
  - 查找是否有 `gongna2` 这个用户。
  -
```
mysql> SELECT Host, User  FROM user WHERE user = 'gongna2';
+------+---------+
| Host | User    |
+------+---------+
| %    | gongna2 |
+------+---------+
1 row in set (0.00 sec)

mysql>
```
  - 查询和这个用户相关的其他的信息

```
mysql> SELECT Host, User password_expired  FROM user WHERE user = 'gongna2';
+------+------------------+
| Host | password_expired |
+------+------------------+
| %    | gongna2          |
+------+------------------+
1 row in set (0.00 sec)

mysql> SELECT Host, User, password_expired FROM user WHERE user = 'gongna2';
+------+---------+------------------+
| Host | User    | password_expired |
+------+---------+------------------+
| %    | gongna2 | N                |
+------+---------+------------------+
1 row in set (0.00 sec)

mysql> SELECT Host, User ,Password_require_current  FROM user WHERE user = 'gongna2';
+------+---------+--------------------------+
| Host | User    | Password_require_current |
+------+---------+--------------------------+
| %    | gongna2 | NULL                     |
+------+---------+--------------------------+
1 row in set (0.00 sec)

mysql>
```

- MySQL提供的 PASSWORD() 函数来对密码进行加密。

```
mysql> SELECT Host, User ,authentication_string FROM user WHERE user = 'gongna2';
+------+---------+-------------------------------------------------------------------------+
| Host | User    | authentication_string                                                   |
+------+---------+-------------------------------------------------------------------------+
| %    | gongna2 | $A$005$a!X;!3m0.M|jl
                                uZa2XE7Y5IhUxOVGajgOWnNUOl6MzRUHCLpGUWy8m84o32 |
+------+---------+-------------------------------------------------------------------------+
1 row in set (0.00 sec)

mysql>
```

- 另外一种添加用户的方法为通过SQL的 GRANT 命令，以下命令会给指定数据库TUTORIALS添加用户 test 密码为 123 。

```
mysql -u root -p
Enter password:*******
mysql> use mysql;
Database changed

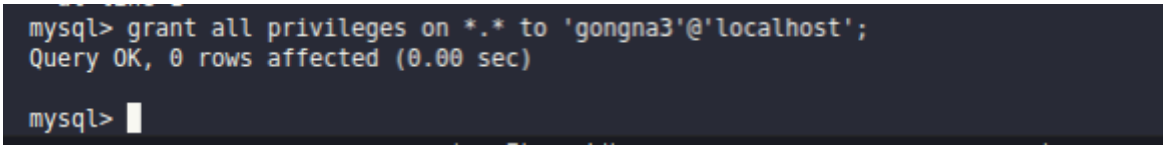
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
    -> ON TUTORIALS.*
    -> TO 'test'@'localhost'
    -> IDENTIFIED BY '123';
```

②掌握授权、回收的操作

- 添加用户的方法为通过SQL的 GRANT 命令，以下命令会给指定数据库TUTORIALS添加用户 test 密码为 123 。

```
mysql -u root -p
Enter password:*******
mysql> use mysql;
Database changed

mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
    -> ON TUTORIALS.*
    -> TO 'test'@'localhost'
    -> IDENTIFIED BY '123';
```



③掌握最基本角色的创建以及其权限的管理二

- `GRANT`的`PRIVILEGES`类型`ALL`（当然是一切）。 *注意：大多数现代 MySQL 安装不需要* `PRIVILEGES` *关键字。*
- `*.*`代表这些权限适用于所有的数据库，以及数据库中的所有表。第一个`*`代表数据库，第二个`*`代表表名
- 这些权限分配给通过本地连接要通过`@'localhost'`指定 `@'%'`指定所有主机。如果想要指定特定的主机，就要输入主机对应的IP地址。
- 一个只为某个用户指定特定数据库访问的例子

```
mysql> grant all privileges ON books.authors TO
'commonUserxiaowang'@'localhost';
```

- **保存更改**

```
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> show tables;
+-----------------------------------------------------+
| Tables_in_mysql                                     |
+-----------------------------------------------------+
| columns_priv                                        |
| component                                           |
| db                                                  |
| default_roles                                       |
| engine_cost                                         |
| func                                                |
| general_log                                         |
| global_grants                                       |
| gtid_executed                                       |
| help_category                                       |
| help_keyword                                        |
| help_relation                                       |
| help_topic                                          |
| innodb_index_stats                                  |
| innodb_table_stats                                  |
| password_history                                    |
| plugin                                              |
| procs_priv                                          |
| proxies_priv                                        |
| replication_asynchronous_connection_failover        |
| replication_asynchronous_connection_failover_managed |
| replication_group_configuration_version             |
| replication_group_member_actions                    |
| role_edges                                          |
| server_cost                                         |
| servers                                             |
| slave_master_info                                   |
| slave_relay_log_info                                |
| slave_worker_info                                   |
| slow_log                                            |
| tables_priv                                         |
| time_zone                                           |
| time_zone_leap_second                               |
| time_zone_name                                      |
| time_zone_transition                                |
| time_zone_transition_type                           |
| user                                                |
+-----------------------------------------------------+
37 rows in set (0.00 sec)

mysql> create user 'gongna3'@'localhost' identified by '123456';
Query OK, 0 rows affected (0.03 sec)

mysql> grant all privileges on *.* to 'gongna3'@'localhost' identified by'123456';
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresp
nds to your MySQL server version for the right syntax to use near 'identified by'123456'
' at line 1
mysql> grant all privileges on *.* to 'gongna3'@'localhost';
Query OK, 0 rows affected (0.00 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.02 sec)

mysql>
```

拓展实验：

①管理登录名、固定服务器角色

- 连接到 MySQL 命令行工具

  对于这个例子，我们假设`gongna3`是主 MySQL 帐户。要开始使用 MySQL 命令行工具 (`mysqlcli`)，请以用户身份连接到您的服务器，然后发出`mysql`命令：

  ```
  mysql --user=username -p密码
  ```

```
gongna@gongna-Ubuntu:~/go/src/github.com/WebDesign$ mysql -ugongna3 -p123456;
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 32
Server version: 8.0.29-0ubuntu0.20.04.3 (Ubuntu)

Copyright (c) 2000, 2022, 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管理数据库用户

- 查看所有的用户
  - 检查MySQL服务器是否启动

```
ps -ef | grep mysqld
```

```
gongna@gongna-Ubuntu:~/go/src/github.com/WebDesign$ ps -ef | grep mysqld
mysql        826       1  0 6月01 ?       00:08:28 /usr/sbin/mysqld
gongna    381334  381298  0 11:11 pts/3    00:00:00 grep --color=auto mysqld
gongna@gongna-Ubuntu:~/go/src/github.com/WebDesign$
```

  - MySQL 用户设置
    - 以root用户进入

```
gongna@gongna-Ubuntu:~/go/src/github.com/WebDesign$ sudo mysql -u root -p
[sudo] gongna 的密码：
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 31
Server version: 8.0.29-0ubuntu0.20.04.3 (Ubuntu)

Copyright (c) 2000, 2022, 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>
```

    - 使用mysql
    -

```
mysql> use mysql;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql>
```

    - 列出databases

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| Test               |
| ccnu               |
| firstdb            |
| information_schema |
| library            |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
8 rows in set (0.00 sec)

mysql>
```

- 列出所有的表

```
mysql> show tables;
+-----------------------------------------------------------+
| Tables_in_mysql                                           |
+-----------------------------------------------------------+
| columns_priv                                              |
| component                                                 |
| db                                                        |
| default_roles                                             |
| engine_cost                                               |
| func                                                      |
| general_log                                               |
| global_grants                                             |
| gtid_executed                                             |
| help_category                                             |
| help_keyword                                              |
| help_relation                                             |
| help_topic                                                |
| innodb_index_stats                                        |
| innodb_table_stats                                        |
| password_history                                          |
| plugin                                                    |
| procs_priv                                                |
| proxies_priv                                              |
| replication_asynchronous_connection_failover              |
| replication_asynchronous_connection_failover_managed      |
| replication_group_configuration_version                   |
| replication_group_member_actions                          |
| role_edges                                                |
| server_cost                                               |
| servers                                                   |
| slave_master_info                                         |
| slave_relay_log_info                                      |
| slave_worker_info                                         |
| slow_log                                                  |
| tables_priv                                               |
| time_zone                                                 |
| time_zone_leap_second                                     |
| time_zone_name                                            |
| time_zone_transition                                      |
| time_zone_transition_type                                 |
| user                                                      |
+-----------------------------------------------------------+
37 rows in set (0.00 sec)
```

- 如果想要新建一个用户就要往user 表格里面插入用户
- 查找是否有 `gongna2` 这个用户。

-
```
mysql> SELECT Host, User  FROM user WHERE user = 'gongna2';
+------+---------+
| Host | User    |
+------+---------+
| %    | gongna2 |
+------+---------+
1 row in set (0.00 sec)

mysql>
```
- 查询和这个用户相关的其他的信息
-
```
mysql> SELECT Host, User password_expired  FROM user WHERE user = 'gongna2';
+------+-----------------+
| Host | password_expired |
+------+-----------------+
| %    | gongna2         |
+------+-----------------+
1 row in set (0.00 sec)

mysql> SELECT Host, User, password_expired FROM user WHERE user = 'gongna2';
+------+---------+-----------------+
| Host | User    | password_expired |
+------+---------+-----------------+
| %    | gongna2 | N               |
+------+---------+-----------------+
1 row in set (0.00 sec)

mysql> SELECT Host, User ,Password_require_current  FROM user WHERE user = 'gongna2';
+------+---------+--------------------------+
| Host | User    | Password_require_current |
+------+---------+--------------------------+
| %    | gongna2 | NULL                     |
+------+---------+--------------------------+
1 row in set (0.00 sec)

mysql>
```

实验六:数据库完整性 🐱

> 数据库的完整性:
>
> （1）实体完整性（**行与行**） 主键约束、外键约束
> （2）**域（字段）完整性**（各个字段数值） 非空约束、默认值约束、检查
> （3）参照完整性（**表与表**） 外键约束
> （4）用户（自定义）完整性
>
> 约束的类型:
>
> （1）主键约束
> （2）外键约束
> （3）非空约束
> （4）唯一约束
> （5）默认值约束
> （6）检查约束

实体完整性:

实体完整性是在关系模型中，数据库完整性三项规则的其中之一。实体完整性这项规则要求每个数据表都必须有主键，而作为主键的所有字段，其属性必须是独一及非空值。

1.主键约束

用来唯一标识表中的一个列，一个表中主键约束最多只能有一个，不同的行上，主键值不能相等。在创建表时设置联合主键约束。联合主键，即多个列联合作为主键。这是，主键列的值不能同时都相等。

修改表时添加主键约束：

```
ALTER TABLE  表名
ADD CONSTRAINT  约束名  PRIMARY KEY(列名);
//表名：要添加约束的表
//约束名：由用户指定，用于标识约束
//列名：要添加主键约束的列
```

修改表时删除主键约束：

```
ALTER TABLE  表名
DROP PRIMARY KEY;
//注意：由于主键约束在一个表中只能有一个，因此不需要指定主键名就可以删除。
```

```
mysql> alter table book_table drop primary key;
ERROR 1075 (42000): Incorrect table definition; there can be only one auto column and it
 must be defined as a key
mysql> alter table book_table add constraint secondPrimaryKey primary key(book_id);
ERROR 1068 (42000): Multiple primary key defined
mysql>
```

- **因为已经添加了主键。且表中只有一个主键。因此删除和添加新的主键会出错。**

2.外键约束

- 外键是指引用另一个表中的一列或多列，被引用的列应该具有**主键约束**或**唯一约束**。
- 外键用于建立和加强两个表数据之间的连接。
- 外键用于建立和加强两个表数据之间的连接。
- 外键用于建立多个表之间的关系。

  ```
  ALTER TABLE  表名
  ADD CONSTRAINT  约束名
  FOREIGN KEY (外键字段名)  REFERENCES 外键表名(列名);
  ```

- 删除外键

  ```
  ALTER TABLE  表名  DROP FOREIGN KEY 外键名;
  ```

参照完整性：

若属性或属性组F是基本关系R的外键，它与基本关系S的主键Ks相对应(基本关系R和S不一定是不同的关系)，则对于R中的每个元组在F上的值必须为:

(1)空值，F的每个属性值均为空值。

(2)S中某个元组中的主键值(主码值)。

即参照的关系中的属性值必须能够在被参照关系找到或者取空值，否则不符合数据库的语义。在实际操作时如更新、删除、插入一个表中的数据，通过参照引用相互关联的另一个表中的数据，来检查对表的数据操作是否正确，不正确则拒绝操作。

用户自定义完整性：

实体完整性和参照完整性适用于任何关系型数据库系统，它主要是针对关系的主关键字和外部关键字取值必须有效而做出的约束。用户定义完整性则是根据应用环境 的要求和实际的需要，对某一具体应用所涉及的数据提出约束性条件。这一约束机制一般不应由应用程序提供，而应有由关系模型提供定义并检验，用户定义完整性 主要包括字段有效性约束和记录有效性。

非空约束（NOT NULL）

```
alter table 表名 modify 列名 类型 not null;
```

唯一约束(UNIQUE)

```
alter table 表名 add constraint UniqueConstraint  unique(列名)
```

默认值约束（DEFAULT）

```
alter table 表名  alter 列名 set dafault 默认值;
```

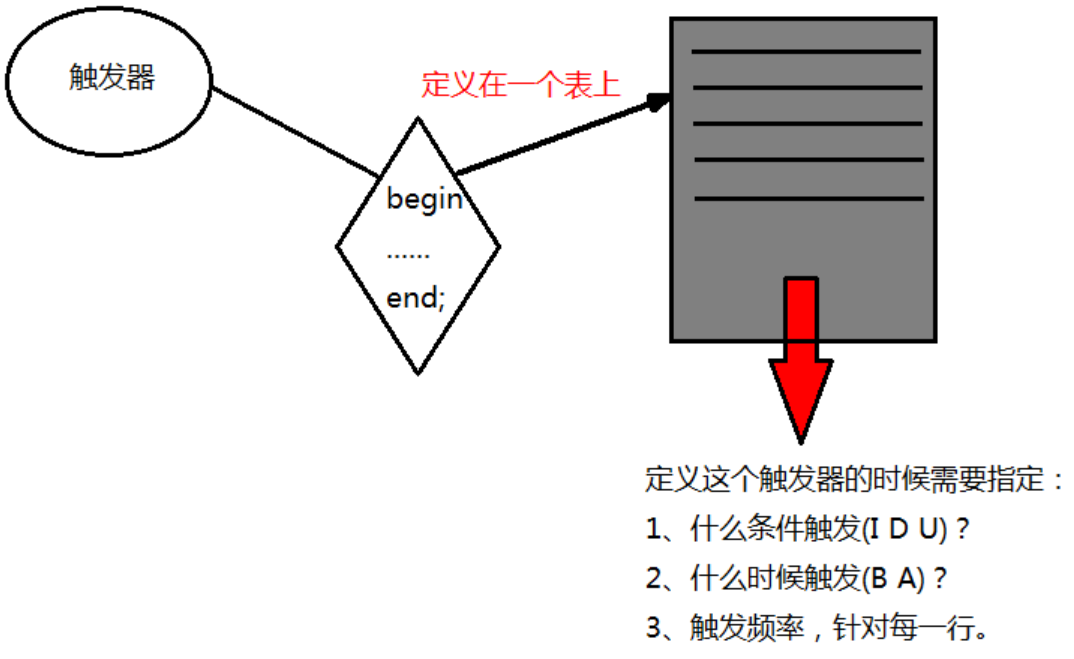
删除默认值约束

```
alter table 表格 ALTER 列名 DROP DEFAULT ;
```

实验七：触发器🌵

深入理解触发器的工作原理以及如何创建触发器

> 触发器是与表有关的数据库对象，在满足定义条件时触发，并执行触发器中定义的语句集合。



触发器的特性：

- **有begin end体，begin end;之间的语句可以写的简单或者复杂。**
- 什么条件会触发：I、D、U
- 什么时候触发：在增删改前或者后

- 触发频率：针对每一行执行
- 触发器定义在表上，附着在表上。

> 本质：**由事件来触发某个操作，事件包括INSERT语句，UPDATE语句和DELETE语句**
>
> cannot associate a trigger with a temporary table or a view.
>
> 不能将触发器与 暂时的表或视图相关联。

触发事件详解：

- INSERT型触发器：插入某一行时激活触发器，可能通过INSERT、LOAD DATA、REPLACE 语句触发(LOAD DAT语句用于将一个文件装入到一个数据表中，相当与一系列的INSERT操作)；
- UPDATE型触发器：更改某一行时激活触发器，可能通过UPDATE语句触发；
- DELETE型触发器：删除某一行时激活触发器，可能通过DELETE、REPLACE语句触发。
- trigger_order是MySQL5.7之后的一个功能，用于定义多个触发器，使用follows(尾随)或precedes(在...之先)来选择触发器执行的先后顺序

使用CREATE TRIGGER语句创建DML和DDL触发器

1. 只有一个执行语句的触发器:

```
mysql> CREATE TABLE `data`(
    ->       `id` bigint(20) AUTO_INCREMENT PRIMARY KEY ,
    ->        `data`  DATE
    ->      )ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=UTF8MB4;
Query OK, 0 rows affected, 1 warning (0.02 sec)

mysql> CREATE TABLE `tigger_table`(
    ->       `id` bigint(20) AUTO_INCREMENT PRIMARY KEY ,
    ->       `user_id`  varchar (20) NOT NULL
    ->      )ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=UTF8MB4;
Query OK, 0 rows affected, 1 warning (0.03 sec)

mysql> CREATE TRIGGER TRIG1 after insert on tigger_table for each row  insert into data
(data)values (current_date());
Query OK, 0 rows affected (0.01 sec)

mysql> insert into tigger_table (user_id) values("2020213760");
Query OK, 1 row affected (0.02 sec)

mysql> select * from tigger_table;
+----+------------+
| id | user_id    |
+----+------------+
|  1 | 2020213760 |
+----+------------+
1 row in set (0.00 sec)

mysql> select * from data;
+----+------------+
| id | data       |
+----+------------+
|  1 | 2022-06-06 |
+----+------------+
1 row in set (0.00 sec)

mysql>
```

   - ```
     //创建两个表
     mysql> CREATE TABLE `data`(
         ->       `id` bigint(20) AUTO_INCREMENT PRIMARY KEY ,
         ->        `data`  DATE
         ->      )ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=UTF8MB4;
     Query OK, 0 rows affected, 1 warning (0.02 sec)

     mysql> CREATE TABLE `tigger_table`(
         ->       `id` bigint(20) AUTO_INCREMENT PRIMARY KEY ,
     ```
     ```

```
    ->       `user_id`  varchar (20) NOT NULL
    ->      )ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=UTF8MB4;
Query OK, 0 rows affected, 1 warning (0.03 sec)

//然后创建触发器
mysql> CREATE TRIGGER TRIG1 after insert on tigger_table for each row
insert into data(data)values (current_date());
Query OK, 0 rows affected (0.01 sec)

//事件触发触发器
mysql> insert into tigger_table (user_id) values("2020213760");
Query OK, 1 row affected (0.02 sec)
```

2. 删除触发器

```
mysql> drop trigger TRIG1;
Query OK, 0 rows affected (0.01 sec)

mysql> insert into tigger_table (user_id) values("2020213740");
Query OK, 1 row affected (0.02 sec)

mysql> select * from data;
+----+------------+
| id | data       |
+----+------------+
|  1 | 2022-06-06 |
+----+------------+
1 row in set (0.00 sec)

mysql> select * from tigger_table;
+----+------------+
| id | user_id    |
+----+------------+
|  1 | 2020213760 |
|  2 | 2020213740 |
+----+------------+
2 rows in set (0.00 sec)

mysql>
```

3. 多个语句的触发器

```
mysql> DELIMITER $$
mysql> CREATE TRIGGER TRIG1 before update on tigger_table for each row begin insert into  data set user_id=old.user_id, data = current_time(); end$$
Query OK, 0 rows affected (0.01 sec)
```

```
DELIMITER $$
mysql> CREATE TRIGGER TRIG1 before update on tigger_table for each row begin
insert into  data set user_id=old.user_id, data = current_time(); end $$
Query OK, 0 rows affected (0.01 sec)
DELIMITER ;
```

   - delimiter命令指定了mysql解释器命令行的结束符默认为 $$

4. 查看触发器

```
mysql> show triggers;
+---------+--------+-----------+-----------------------------------------------------------------------------------+--------+---------+----------+----------------+----------------------+----------------------+--------------------+
| Trigger | Event  | Table     | Statement                                                                         | Timing | Created | sql_mode | Definer | character_set_client | collation_connection | Database Collation |
+---------+--------+-----------+-----------------------------------------------------------------------------------+--------+---------+----------+----------------+----------------------+----------------------+--------------------+
| TRIG1   | UPDATE | tigger_table | begin insert into  data set user_id=old.user_id, data = current_time(); end | BEFORE | 2022-06-06 19:24:49.32 | ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION | gongna@% | utf8mb4 | utf8mb4_0900_ai_ci | utf8mb4_0900_ai_ci |
+---------+--------+-----------+-----------------------------------------------------------------------------------+--------+---------+----------+----------------+----------------------+----------------------+--------------------+
1 row in set (0.00 sec)
```

```
mysql> show triggers;
```

5. 触发事件

```
mysql> insert into tigger_table (user_id) values ("2020213760");
Query OK, 1 row affected (0.02 sec)

mysql> update tigger_table set user_id="2020213788" where user_id ="2020213760";
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from data;
+----+------------+------------+
| id | user_id    | data       |
+----+------------+------------+
|  1 | 2020213760 | 2022-06-06 |
+----+------------+------------+
1 row in set (0.00 sec)

mysql> select * from trigger_table;
ERROR 1146 (42S02): Table 'Test.trigger_table' doesn't exist
mysql> select * from tigger_table;
+----+------------+
| id | user_id    |
+----+------------+
|  1 | 2020213788 |
+----+------------+
1 row in set (0.00 sec)

mysql>
```