NSD DBA1 DAY03

1 MySQL存储引擎的配置

1.1 问题

本案例要求MySQL数据存储引擎的使用，完成以下任务操作：

- 查看服务支持的存储引擎
- 查看默认存储类型
- 更改表的存储引擎
- 设置数据库服务默认使用的存储引擎

1.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：查看存储引擎信息

登入MySQL服务器，查看当前支持哪些存储引擎。

使用mysql命令连接，以root用户登入：

```
[root@dbsvr1 ~]# mysql -u root –p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 5.7.17 MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved

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 statemen

```

Top

```
mysql>
```

执行SHOW ENGINES\G指令可列表查看，MySQL 5.6可用的存储引擎有9种（除最后的FEDERATED以外，其他8种都支持），其中默认采用的存储引擎为InnoDB：

```
mysql> SHOW ENGINES\G
*************************** 1. row ********************
      Engine: InnoDB
     Support: DEFAULT                    //此存储引擎为默认
     Comment: Supports transactions, row-level locking, and foreign keys
Transactions: YES
          XA: YES
  Savepoints: YES
*************************** 2. row ********************
      Engine: MRG_MYISAM
     Support: YES
     Comment: Collection of identical MyISAM tables
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 3. row ********************
      Engine: MEMORY
     Support: YES
     Comment: Hash based, stored in memory, useful for temporary table
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 4. row ********************
      Engine: BLACKHOLE
     Support: YES
     Comment: /dev/null storage engine (anything you write to it disappea
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 5. row ********************
      Engine: MyISAM
```

Top

```
     Support: YES
     Comment: MyISAM storage engine
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 6. row ********************
      Engine: CSV
     Support: YES
     Comment: CSV storage engine
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 7. row ********************
      Engine: ARCHIVE
     Support: YES
     Comment: Archive storage engine
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 8. row ********************
      Engine: PERFORMANCE_SCHEMA
     Support: YES
     Comment: Performance Schema
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 9. row ********************
      Engine: FEDERATED
     Support: NO                    //此引擎不被支持
     Comment: Federated MySQL storage engine
Transactions: NULL
          XA: NULL
  Savepoints: NULL
9 rows in set (0.01 sec)
```

Top

步骤二：查看默认存储类型

查看系统变量default_storage_engine 的值，确认默认采用的存储引擎是InnoDB：

```
mysql> SHOW VARIABLES LIKE 'default_storage_engine';
+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| default_storage_engine | InnoDB |
+------------------------+--------+
1 row in set (0.00 sec)
```

步骤三：修改默认存储引擎

在 mysql> 环境中，可以直接通过SET指令更改默认的存储引擎（只在本次连接会话过程中有效，退出重进即失效）。比如临时修改为MyISAM，可执行下列操作：

```
mysql> SET default_storage_engine=MyISAM;            //改用MyISAM引
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW VARIABLES LIKE 'default_storage_engine';        //确认
+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| default_storage_engine | MyISAM |
+------------------------+--------+
1 row in set (0.00 sec)
```

若希望直接修改MySQL服务程序所采用的默认存储引擎，应将相关设置写入配置文件/etc/my.cnf，并重启服务后生效。比如：

Top

```
[root@dbsvr1 ~]# vim /etc/my.cnf
[mysqld]
.. ..
default_storage_engine=MEMORY                    //改用MEMORY引

[root@dbsvr1 ~]# systemctl  restart mysqld.service          //重启服务
```

重新登入 mysql> 确认修改结果：

```
[root@dbsvr1 ~]# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.7.17 MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved

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 statemer


mysql> SHOW VARIABLES LIKE 'default_storage_engine';
+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| default_storage_engine | MEMORY |                //默认引擎已修改
+------------------------+--------+
1 row in set (0.00 sec)

mysql> exit
Bye
```

Top

步骤四：设置数据库服务默认使用的存储引擎

为了避免后续实验障碍，测试完后记得恢复原状——移除默认引擎设置，或者将其修改为InnoDB即可：

```
[root@dbsvr1 ~]# vim /etc/my.cnf
[mysqld]
.. ..
default_storage_engine=InnoDB
[root@dbsvr1 ~]# systemctl  restart mysqld.service
```

确认恢复结果（选项 -e 可调用指定的SQL操作后返回Shell命令行）：

```
[root@dbsvr1 ~]# mysql -u root -p -e "SHOW VARIABLES LIKE 'default_st
Enter password:
+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| default_storage_engine | InnoDB |
+------------------------+--------+
```

2 数据导入/导出

2.1 问题

使用SQL语句完成下列导出、导入操作：

1. 将/etc/passwd文件导入userdb库user表并给每条记录加编号
2. 将userdb库user表中UID小于100的前10条记录导出，存为/myload/user2.txt文件

2.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：将/etc/passwd文件导入MySQL数据库

导入后的表结构取决于/etc/passwd配置文件。若一时记不住各字段的含义，也可以查看passwd配置文件的man手册页，找到格式描述相关的说明，比如：

```
[root@dbsvr1 ~]# man 5 passwd
.. ..
       Each line of the file describes a single user, and contains seven col
       arated fields:

              name:password:UID:GID:GECOS:directory:shell

The field are as follows:                    //以下详细解释各字段的作用

       name        This is the user's login name.  It should  not  contain  ca
                   letters.

       password    This  is either the encrypted user password, an asterisk
                   the letter 'x'.  (See pwconv(8) for an explanation of 'x'.)

       UID         The privileged root login account (superuser) has the user

       GID         This is the numeric primary group ID for this user.  (Additio
                   groups  for  the  user are defined in the system group file; s
                   group(5)).

GECOS  stands for "General Electric Comprehensive Operating Sys-
                   tem", which was renamed to GCOS when GE's large system
                   was  sold to Honeywell.  Dennis Ritchie has reported: "Som
                   we sent printer output or batch jobs to the GCOS  machine.
                   gcos  field in the password file was a place to stash the inf
                   mation for the $IDENTcard.  Not elegant."

       directory   This is the user's home directory: the initial  directory  wh
                   the user is placed after logging in.  The value in this field is
                   used to set the HOME environment variable.

       shell       This is the program to run at login (if empty, use /bin/sh).
                   set  to  a  nonexistent  executable,  the user will be unable
                   login through login(1).  The value in this field is used to  set
                   the SHELL environment variable.
```

Top

```
……
```

1）新建userdb库、user表

以数据库用户root登入MySQL服务：

```
[root@dbsvr1 ~]# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.7.17 MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved

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 statemen

mysql>
```

新建userdb库，切换到userdb库：

```
mysql> CREATE DATABASE userdb;
Query OK, 1 row affected (0.00 sec)

mysql> USE userdb;
Database changed
```

新建user表，字段设置及相关操作参考如下：

Top

```
mysql> CREATE TABLE user(
    -> username varchar(24) NOT NULL,
    -> password varchar(48) DEFAULT 'x',
    -> uid int(5) NOT NULL,
    -> gid int(5) NOT NULL,
    -> fullname varchar(48),
    -> homedir varchar(64) NOT NULL,
    -> shell varchar(24) NOT NULL
    -> );
Query OK, 0 rows affected (0.70 sec)
```

确认user表的结构：

```
mysql> DESC user;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| username | varchar(24) | NO   |     | NULL    |       |
| password | varchar(48) | YES  |     | x       |       |
| uid      | int(5)      | NO   |     | NULL    |       |
| gid      | int(5)      | NO   |     | NULL    |       |
| fullname | varchar(48) | YES  |     | NULL    |       |
| homedir  | varchar(64) | NO   |     | NULL    |       |
| shell    | varchar(24) | NO   |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
7 rows in set (0.01 sec)
```

2）如果直接导入会报错。在MySQL 5.7.6版本之后，导入文件只能在secure_file_priv指定的文件夹下。执行show variables like '%secure%'命令显示文件目录：

```
mysql>  LOAD DATA INFILE '/etc/passwd' INTO TABLE userlist FIELDS T
ERROR 1290 (HY000): The MySQL server is running with the --secure-fil
mysql> show variables like '%secure%';
+--------------------------+----------------------+
| Variable_name            | Value                |
+--------------------------+----------------------+
| require_secure_transport | OFF                  |
```

Top

```
| secure_auth            | ON                    |
| secure_file_priv       | /var/lib/mysql-files/ |
+------------------------+-----------------------+
3 rows in set (0.00 sec)
```

3）执行导入操作

将/etc/passwd文件复制到/var/lib/mysql-files/目录下，

读取/var/lib/mysql-files/passwd文件内容，以 “:” 为分隔，导入到user表中：

```
[root@dbsvr1 ~]#cp /etc/passwd /var/lib/mysql-files/
mysql> LOAD DATA INFILE '/var/lib/mysql-files/passwd'
    -> INTO TABLE userlist
    -> FIELDS TERMINATED BY ':';
Query OK, 39 rows affected (0.11 sec)
Records: 39  Deleted: 0  Skipped: 0  Warnings: 0
```

上述操作中省略了行分隔 LINES TERMINATED BY '\n'，因为这是默认的情况（每行一条原始记录），除非需要以其他字符分割行，才需要用到这个。比如，以下操作指定了行分隔为'\n'，将/var/lib/mysql-files/passwd文件的内容导入另一个表user2，最终user2表的内容与user的内容是一样的：

代码

4）确认导入结果

分别统计user、user2表内的记录个数：

```
mysql> SELECT COUNT(*) FROM user;
+----------+
| COUNT(*) |
+----------+
|       39 |                    //user表有39条记录
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM user2;
```

Top

```
+----------+
| COUNT(*) |
+----------+
|       39 |                     //user2表也有39条记录
+----------+
1 row in set (0.00 sec)
```

查看user表的前10条记录，列出用户名、UID、GID、宿主目录、登录Shell：

```
mysql> SELECT username,uid,gid,homedir,shell
    -> FROM user LIMIT 10;
+----------+-----+-----+-----------------+----------------+
| username | uid | gid | homedir         | shell          |
+----------+-----+-----+-----------------+----------------+
| root     |   0 |   0 | /root           | /bin/bash      |
| bin      |   1 |   1 | /bin            | /sbin/nologin  |
| daemon   |   2 |   2 | /sbin           | /sbin/nologin  |
| adm      |   3 |   4 | /var/adm        | /sbin/nologin  |
| lp       |   4 |   7 | /var/spool/lpd  | /sbin/nologin  |
| sync     |   5 |   0 | /sbin           | /bin/sync      |
| shutdown |   6 |   0 | /sbin           | /sbin/shutdown |
| halt     |   7 |   0 | /sbin           | /sbin/halt     |
| mail     |   8 |  12 | /var/spool/mail | /sbin/nologin  |
| operator |  11 |   0 | /root           | /sbin/nologin  |
+----------+-----+-----+-----------------+----------------+
10 rows in set (0.00 sec)
```

查看user2表的前10条记录，同样列出用户名、UID、GID、宿主目录、登录Shell：

```
mysql> SELECT username,uid,gid,homedir,shell
    -> FROM user2 LIMIT 10;
+----------+-----+-----+-----------------+----------------+
| username | uid | gid | homedir         | shell          |
+----------+-----+-----+-----------------+----------------+
| root     |   0 |   0 | /root           | /bin/bash      |
```

Top

```
| bin      |   1 |   1 | /bin           | /sbin/nologin  |
| daemon   |   2 |   2 | /sbin          | /sbin/nologin  |
| adm      |   3 |   4 | /var/adm       | /sbin/nologin  |
| lp       |   4 |   7 | /var/spool/lpd | /sbin/nologin  |
| sync     |   5 |   0 | /sbin          | /bin/sync      |
| shutdown |   6 |   0 | /sbin          | /sbin/shutdown |
| halt     |   7 |   0 | /sbin          | /sbin/halt     |
| mail     |   8 |  12 | /var/spool/mail | /sbin/nologin  |
| operator |  11 |   0 | /root          | /sbin/nologin  |
+----------+-----+-----+----------------+----------------+
10 rows in set (0.00 sec)
```

步骤二：为user表中的每条记录添加自动编号

这个只要修改user表结构，添加一个自增字段即可。

比如，添加一个名为sn的序号列，作为user表的第一个字段：

1）添加自增主键字段sn

```
mysql> ALTER TABLE user
    -> ADD sn int(4) AUTO_INCREMENT PRIMARY KEY FIRST;
Query OK, 0 rows affected (0.62 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

2）验证自动编号结果

查看user表的前10条记录，列出序号、用户名、UID、GID、宿主目录：

```
mysql> SELECT sn,username,uid,gid,homedir
    -> FROM user LIMIT 10;
+----+----------+-----+-----+----------------+
| sn | username | uid | gid | homedir        |
+----+----------+-----+-----+----------------+
|  1 | root     |   0 |   0 | /root          |
|  2 | bin      |   1 |   1 | /bin           |
|  3 | daemon   |   2 |   2 | /sbin          |
|  4 | adm      |   3 |   4 | /var/adm       |
```

Top

```
| 5 | lp       |  4 |  7 | /var/spool/lpd  |
| 6 | sync     |  5 |  0 | /sbin           |
| 7 | shutdown |  6 |  0 | /sbin           |
| 8 | halt     |  7 |  0 | /sbin           |
| 9 | mail     |  8 | 12 | /var/spool/mail |
| 10 | operator | 11 |  0 | /root           |
+----+----------+-----+-----+-----------------+
10 rows in set (0.00 sec)
```

步骤三：从MySQL数据库中导出查询结果

以将userdb库user表中UID小于100的前10条记录导出为/myload/user2.txt文件为例。

1）确认存放导出数据的文件夹

```
[root@dbsvr1 ~]# ls -ld /var/lib/mysql-files/
drwxr-x---. 2 mysql mysql 19 4月   7 11:15 /var/lib/mysql-files/
```

2）修改目录及查看修改结果

```
[root@dbsvr1 ~]# mkdir  /myload  ;  chown  mysql  /myload
[root@dbsvr1 ~]# vim  /etc/my.cnf
[mysqld]
secure_file_priv="/myload"
[root@dbsvr1 ~]# systemctl  restart mysqld
mysql> show variables like "secure_file_priv";
+------------------+----------+
| Variable_name    | Value    |
+------------------+----------+
| secure_file_priv | /myload/ |
```

2）导出user表中UID小于100的前10条记录

如果以默认的'\n' 为行分隔，导出操作同样可不指定LINES TERMINATED BY Top

```
mysql> SELECT * FROM userdb.user WHERE uid<100
    -> INTO OUTFILE '/myload/user2.txt'
    -> FIELDS TERMINATED BY ':';
Query OK, 24 rows affected (0.00 sec)
```

3）确认导出结果

返回到Shell命令行，查看/myload/user2.txt文件的行数：

```
[root@dbsvr1 ~]# wc -l /myload/user2.txt
24 /myload/user2.txt
```

查看/myload/user2.txt文件的最后10行内容：

```
[root@dbsvr1 ~]# tail /myload/user2.txt
19:avahi:x:70:70:Avahi mDNS/DNS-SD Stack:/var/run/avahi-daemon:/
24:rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin
25:rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
28:radvd:x:75:75:radvd user:/:/sbin/nologin
29:ntp:x:38:38::/etc/ntp:/sbin/nologin
33:gdm:x:42:42::/var/lib/gdm:/sbin/nologin
35:postfix:x:89:89::/var/spool/postfix:/sbin/nologin
36:sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nolog
37:tcpdump:x:72:72::/:/sbin/nologin
39:mysql:x:27:27:MySQL Server:/var/lib/mysql:/bin/false
```

3 操作表记录

3.1 问题

练习表记录的操作

1. 表记录的插入
2. 表记录的更新
3. 表记录的查询
4. 表记录的删除

Top

3.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：创建stu_info表，并确保stu_info表记录为空。

在userdb库中创建stu_info表：

```
[root@dbsvr1 ~]# mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 19
Server version: 5.7.17 MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved

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 statemer

mysql> use userdb;
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> CREATE TABLE stu_info(
    -> name varchar(12) NOT NULL,
    -> gender enum('boy','girl') DEFAULT 'boy',
    -> age int(3) NOT NULL
    -> );
Query OK, 0 rows affected (0.23 sec)
```

删除stu_info表的所有记录：

Top

```
mysql> DELETE FROM stu_info;
```

```
Query OK, 0 rows affected (0.00 sec)            //stu_info表刚建立 删除
```

确认删除结果：

```
mysql> SELECT * FROM stu_info;
Empty set (0.00 sec)
```

步骤二：练习表记录的操作

1）插入记录时，指定记录的每一个字段的值

这种情况下，不需要明确指出字段，但每条记录的值的顺序、类型都必须与表格结构向一致，否则可能无法正确插入记录。

比如，以下操作将向stu_info表插入3条表记录：

```
mysql> INSERT stu_info VALUES
   -> ('Jim','girl',24),
   -> ('Tom','boy',21),
   -> ('Lily','girl',20);
Query OK, 3 rows affected (0.15 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

完成插入后确认表记录：

```
mysql> SELECT * FROM stu_info;
+------+--------+-----+
| name | gender | age |
+------+--------+-----+
| Jim  | girl   |  24 |
| Tom  | boy    |  21 |
| Lily | girl   |  20 |
+------+--------+-----+
3 rows in set (0.00 sec)
```

Top

2）插入记录时，只指定记录的部分字段的值

这种情况下，必须指出各项值所对应的字段；而且，未赋值的字段应设置有默认值或者有自增填充属性或者允许为空，否则插入操作将会失败。

比如，向stu_info表插入Jerry的年龄信息，性别为默认的 “boy” ，自动编号，相关操作如下：

```
mysql> INSERT INTO stu_info(name,age)
    -> VALUES('Jerry',27);
Query OK, 1 row affected (0.04 sec)
```

类似的，再插入用户Mike的年龄信息：

```
mysql> INSERT INTO stu_info(name,age)
    -> VALUES('Mike',21);
Query OK, 1 row affected (0.05 sec)
```

确认目前stu_info表的所有记录：

```
mysql> SELECT * FROM stu_info;
+-------+--------+-----+
| name  | gender | age |
+-------+--------+-----+
| Jim   | girl   |  24 |
| Tom   | boy    |  21 |
| Lily  | girl   |  20 |
| Jerry | boy    |  27 |
| Mike  | boy    |  21 |
+-------+--------+-----+
5 rows in set (0.00 sec)
```

3）更新表记录时，若未限制条件，则适用于所有记录

Top

将stu_info表中所有记录的age设置为10：

```
mysql> UPDATE stu_info SET age=10;
Query OK, 5 rows affected (0.04 sec)
Rows matched: 5  Changed: 5  Warnings: 0
```

确认更新结果：

```
mysql> SELECT * FROM stu_info;
+-------+--------+-----+
| name  | gender | age |
+-------+--------+-----+
| Jim   | girl   |  10 |
| Tom   | boy    |  10 |
| Lily  | girl   |  10 |
| Jerry | boy    |  10 |
| Mike  | boy    |  10 |
+-------+--------+-----+
5 rows in set (0.00 sec)
```

4）更新表记录时，可以限制条件，只对符合条件的记录有效

将stu_info表中所有性别为“boy”的记录的age设置为20：

```
mysql> UPDATE stu_info SET age=20
    -> WHERE gender='boy';
Query OK, 3 rows affected (0.04 sec)
Rows matched: 3  Changed: 3  Warnings: 0
```

确认更新结果：

```
mysql> SELECT * FROM stu_info;
+-------+--------+-----+
| name  | gender | age |
+-------+--------+-----+
| Jim   | girl   |  10 |
| Tom   | boy    |  20 |
| Lily  | girl   |  10 |
| Jerry | boy    |  20 |
| Mike  | boy    |  20 |
+-------+--------+-----+
5 rows in set (0.00 sec)
```

5）删除表记录时，可以限制条件，只删除符合条件的记录

删除stu_info表中年龄小于18的记录：

```
mysql> DELETE FROM stu_info WHERE age < 18;
Query OK, 2 rows affected (0.03 sec)
```

确认删除结果：

```
mysql> SELECT * FROM stu_info;
+-------+--------+-----+
| name  | gender | age |
+-------+--------+-----+
| Tom   | boy    |  20 |
| Jerry | boy    |  20 |
| Mike  | boy    |  20 |
+-------+--------+-----+
3 rows in set (0.00 sec)
```

6）删除表记录时，如果未限制条件，则会删除所有的表记录

删除stu_info表的所有记录：

```
mysql> DELETE FROM stu_info;
Query OK, 3 rows affected (0.00 sec)
```

Top

确认删除结果：

```
mysql> SELECT * FROM stu_info;
Empty set (0.00 sec)
```

4 查询及匹配条件

4.1 问题

练习常见的SQL查询及条件设置

1. 创建stu_info表，并插入数据
2. 练习常见SQL查询及条件设置

4.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：根据任务要求建立员工档案表stu_info（如上个实验已创建，可将上个实验stu_info表中记录清除后继续使用）

1）在userdb库中创建stu_info表

以root用户登入MySQL服务器：

```
[root@dbsvr1 ~]# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.6.15 MySQL Community Server (GPL)

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved

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 statemer
                                                             Top
mysql>
```

打开test库：

```
mysql> USE userdb;
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
```

创建stu_info表，包括name、gender、age三个字段：

```
mysql> CREATE TABLE stu_info(
    -> name varchar(12) NOT NULL,
    -> gender enum('boy','girl') DEFAULT 'boy',
    -> age int(3) NOT NULL
    -> );
Query OK, 0 rows affected (0.03 sec)
```

确认表结构：

```
mysql> DESC stu_info;
+--------+-------------------+------+-----+---------+-------+
| Field  | Type              | Null | Key | Default | Extra |
+--------+-------------------+------+-----+---------+-------+
| name   | varchar(12)       | NO   |     | NULL    |       |
| gender | enum('boy','girl') | YES  |     | boy     |       |
| age    | int(3)            | NO   |     | NULL    |       |
+--------+-------------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

2）准备测试表格

向建立的stu_info表插入几条测试记录

Top

```
mysql> INSERT INTO stu_info VALUES
```

```
    -> ('Jim','girl',24),
    -> ('Tom','boy',21),
    -> ('Lily','girl',20),
    -> ('Jerry','boy',27),
    -> ('Mike','boy',21)
    -> ;
Query OK, 5 rows affected (0.06 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

确认stu_info表的所有记录内容：

```
mysql> SELECT * FROM stu_info;
+-------+--------+-----+
| name  | gender | age |
+-------+--------+-----+
| Jim   | girl   |  24 |
| Tom   | boy    |  21 |
| Lily  | girl   |  20 |
| Jerry | boy    |  27 |
| Mike  | boy    |  21 |
+-------+--------+-----+
5 rows in set (0.00 sec)
```

步骤二：练习常见SQL查询及条件设置

1）常用的表记录统计函数

查询stu_info表一共有多少条记录（本例中为5条）：

Top

```
mysql> SELECT count(*) FROM stu_info;
+----------+
| count(*) |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)
```

计算stu_info表中各学员的平均年龄、最大年龄、最小年龄：

```
mysql> SELECT avg(age),max(age),min(age) FROM stu_info;
+----------+----------+----------+
| avg(age) | max(age) | min(age) |
+----------+----------+----------+
|  22.6000 |       27 |       20 |
+----------+----------+----------+
1 row in set (0.00 sec)
```

计算stu_info表中男学员的个数：

```
mysql> SELECT count(gender) FROM stu_info WHERE gender='boy';
+---------------+
| count(gender) |
+---------------+
|             3 |
+---------------+
1 row in set (0.00 sec)
```

2）字段值的数值比较

列出stu_info表中年龄为21岁的学员记录：

Top

```
mysql> SELECT * FROM stu_info WHERE age=21;
```

```
+------+--------+-----+
| name | gender | age |
+------+--------+-----+
| Tom  | boy    |  21 |
| Mike | boy    |  21 |
+------+--------+-----+
2 rows in set (0.00 sec)
```

列出stu_info表中年龄超过21岁的学员记录：

```
mysql> SELECT * FROM stu_info WHERE age>21;
+-------+--------+-----+
| name  | gender | age |
+-------+--------+-----+
| Jim   | girl   |  24 |
| Jerry | boy    |  27 |
+-------+--------+-----+
2 rows in set (0.00 sec)
```

列出stu_info表中年龄大于或等于21岁的学员记录：

```
mysql> SELECT * FROM stu_info WHERE age>=21;
+-------+--------+-----+
| name  | gender | age |
+-------+--------+-----+
| Jim   | girl   |  24 |
| Tom   | boy    |  21 |
| Jerry | boy    |  27 |
| Mike  | boy    |  21 |
+-------+--------+-----+
4 rows in set (0.00 sec)
```

Top

列出stu_info表中年龄在20岁和24岁之间的学员记录：

```
mysql> SELECT * FROM stu_info WHERE age BETWEEN 20 and 24;
+------+--------+-----+
| name | gender | age |
+------+--------+-----+
| Jim  | girl   |  24 |
| Tom  | boy    |  21 |
| Lily | girl   |  20 |
| Mike | boy    |  21 |
+------+--------+-----+
4 rows in set (0.00 sec)
```

3）多个条件的组合

列出stu_info表中年龄小于23岁的女学员记录：

```
mysql> SELECT * FROM stu_info WHERE age < 23 AND gender='girl';
+------+--------+-----+
| name | gender | age |
+------+--------+-----+
| Lily | girl   |  20 |
+------+--------+-----+
1 row in set (0.00 sec)
```

列出stu_info表中年龄小于23岁的学员，或者女学员的记录：

Top

```
mysql> SELECT * FROM stu_info WHERE age < 23 OR gender='girl';
+------+--------+-----+
| name | gender | age |
+------+--------+-----+
| Jim  | girl   |  24 |
| Tom  | boy    |  21 |
| Lily | girl   |  20 |
| Mike | boy    |  21 |
+------+--------+-----+
4 rows in set (0.00 sec)
```

如果某个记录的姓名属于指定范围内的一个，则将其列出：

```
mysql> SELECT * FROM stu_info WHERE name IN
    -> ('Jim','Tom','Mickey','Minnie');
+------+--------+-----+
| name | gender | age |
+------+--------+-----+
| Jim  | girl   |  24 |
| Tom  | boy    |  21 |
+------+--------+-----+
2 rows in set (0.00 sec)
```

4）使用SELECT做数学计算

计算1234与5678的和：

Top

```
mysql> SELECT 1234+5678;
+-----------+
| 1234+5678 |
+-----------+
|      6912 |
+-----------+
1 row in set (0.00 sec)
```

计算1234与5678的乘积：

```
mysql> SELECT 1234*5678;
+-----------+
| 1234*5678 |
+-----------+
|   7006652 |
+-----------+
1 row in set (0.00 sec)
```

计算1.23456789除以3的结果：

```
mysql> SELECT 1.23456789/3;
+----------------+
| 1.23456789/3   |
+----------------+
| 0.411522630000 |
+----------------+
1 row in set (0.00 sec)
```

输出stu_info表各学员的姓名、15年后的年龄：

Top

```
mysql> SELECT name,age+15 FROM stu_info;
+-------+--------+
| name  | age+15 |
+-------+--------+
| Jim   |     39 |
| Tom   |     36 |
| Lily  |     35 |
| Jerry |     42 |
| Mike  |     36 |
+-------+--------+
5 rows in set (0.00 sec)
```

5）使用模糊查询，LIKE引领

以下划线 _ 匹配单个字符，% 可匹配任意多个字符。

列出stu_info表中姓名以 “J” 开头的学员记录：

```
mysql> SELECT * FROM stu_info WHERE name LIKE 'J%';
+-------+--------+-----+
| name  | gender | age |
+-------+--------+-----+
| Jim   | girl   |  24 |
| Jerry | boy    |  27 |
+-------+--------+-----+
2 rows in set (0.00 sec)
```

列出stu_info表中姓名以 “J” 开头且只有3个字母的学员记录：

```
mysql> SELECT * FROM stu_info WHERE name LIKE 'J__';
+------+--------+-----+
| name | gender | age |
+------+--------+-----+
| Jim  | girl   |  24 |
+------+--------+-----+
1 row in set (0.00 sec)
```

Top

6）使用正则表达式，REGEXP引领

列出stu_info表中姓名以“J”开头且以“y”结尾的学员记录：

```
mysql> SELECT * FROM stu_info WHERE name REGEXP '^J.*y$';
+-------+--------+-----+
| name  | gender | age |
+-------+--------+-----+
| Jerry | boy    |  27 |
+-------+--------+-----+
1 row in set (0.00 sec)
```

效果等同于：

```
mysql> SELECT * FROM stu_info WHERE name Like 'J%y';
+-------+--------+-----+
| name  | gender | age |
+-------+--------+-----+
| Jerry | boy    |  27 |
+-------+--------+-----+
1 row in set (0.00 sec)
```

列出stu_info表中姓名以“J”开头或者以“y”结尾的学员记录：

```
mysql> SELECT * FROM stu_info WHERE name REGEXP '^J|y$';
+-------+--------+-----+
| name  | gender | age |
+-------+--------+-----+
| Jim   | girl   |  24 |
| Lily  | girl   |  20 |
| Jerry | boy    |  27 |
+-------+--------+-----+
3 rows in set (0.00 sec)
```

Top

效果等同于：

```
mysql> SELECT * FROM stu_info WHERE name Like 'J%' OR name Like '
+-------+--------+-----+
| name  | gender | age |
+-------+--------+-----+
| Jim   | girl   |  24 |
| Lily  | girl   |  20 |
| Jerry | boy    |  27 |
+-------+--------+-----+
3 rows in set (0.00 sec)
```

7）按指定的字段排序，ORDER BY

列出stu_info表的所有记录，按年龄排序：

```
mysql> SELECT * FROM stu_info GROUP BY age;
+-------+--------+-----+
| name  | gender | age |
+-------+--------+-----+
| Lily  | girl   |  20 |
| Tom   | boy    |  21 |
| Jim   | girl   |  24 |
| Jerry | boy    |  27 |
+-------+--------+-----+
4 rows in set (0.00 sec)
```

因默认为升序（Ascend）排列，所以上述操作等效于：

```
mysql> SELECT * FROM stu_info GROUP BY age ASC;
+-------+--------+-----+
| name  | gender | age |
+-------+--------+-----+
| Lily  | girl   |  20 |
| Tom   | boy    |  21 |
| Jim   | girl   |  24 |
```

Top

```
| Jerry | boy    |  27 |
+--------+--------+-----+
4 rows in set (0.00 sec)
```

若要按降序（Descend）排列，则将ASC改为DESC即可：

```
mysql> SELECT * FROM stu_info GROUP BY age DESC;
+--------+--------+-----+
| name   | gender | age |
+--------+--------+-----+
| Jerry | boy    |  27 |
| Jim    | girl   |  24 |
| Tom    | boy    |  21 |
| Lily   | girl   |  20 |
+--------+--------+-----+
4 rows in set (0.00 sec)
```

8）限制查询结果的输出条数，LIMIT

查询stu_info表的所有记录，只列出前3条：

```
mysql> SELECT * FROM stu_info LIMIT 3;
+------+--------+-----+
| name | gender | age |
+------+--------+-----+
| Jim  | girl   |  24 |
| Tom  | boy    |  21 |
| Lily | girl   |  20 |
+------+--------+-----+
3 rows in set (0.00 sec)
```

列出stu_info表中年龄最大的3条学员记录：

Top

```
mysql> SELECT * FROM stu_info GROUP BY age DESC LIMIT 3;
```

```
+--------+--------+-----+
| name   | gender | age |
+--------+--------+-----+
| Jerry | boy    |  27 |
| Jim    | girl   |  24 |
| Tom    | boy    |  21 |
+--------+--------+-----+
3 rows in set (0.00 sec)
```

9）分组查询结果，GROUP BY

针对stu_info表，按性别分组，分别统计出男、女学员的人数：

```
mysql> SELECT gender,count(gender) FROM stu_info GROUP BY gender;
+--------+---------------+
| gender | count(gender) |
+--------+---------------+
| boy    |             3 |
| girl   |             2 |
+--------+---------------+
2 rows in set (0.00 sec)
```

列出查询字段时，可以通过AS关键字来指定显示别名，比如上述操作可改为：

```
mysql> SELECT gender AS '性别',count(gender) AS '人数'
    -> FROM stu_info GROUP BY gender;
+--------+--------+
| 性别   | 人数   |
+--------+--------+
| boy    |      3 |
| girl   |      2 |
+--------+--------+
2 rows in set (0.00 sec)
```

Top