

NSD DBA1 DAY03

1. [MySQL存储引擎的配置](#)
2. [数据导入/导出](#)
3. [操作表记录](#)
4. [查询及匹配条件](#)

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：

```
01. [ root@dbvr1 ~] # mysql -u root -p
02. Enter password:
03. Welcome to the MySQL monitor. Commands end with ; or \g.
04. Your MySQL connection id is 9
05. Server version: 5.7.17 MySQL Community Server (GPL)
06.
07. Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.
08.
09. Oracle is a registered trademark of Oracle Corporation and/or its
10. affiliates. Other names may be trademarks of their respective
11. owners.
12.
13. Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
14.
15. mysql>
```

[Top](#)

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：

```

01.  my sql> SHOW ENGINES\G
02.  ***** 1. row *****
03.      Engine: InnoDB
04.      Support: DEFAULT //此存储引擎为默认
05.      Comment: Supports transactions, row- level locking, and foreign keys
06.      Transactions: YES
07.      XA: YES
08.      Savepoints: YES
09.  ***** 2. row *****
10.      Engine: MRG_MYISAM
11.      Support: YES
12.      Comment: Collection of identical MyISAM tables
13.      Transactions: NO
14.      XA: NO
15.      Savepoints: NO
16.  ***** 3. row *****
17.      Engine: MEMORY
18.      Support: YES
19.      Comment: Hash based, stored in memory, useful for temporary tables
20.      Transactions: NO
21.      XA: NO
22.      Savepoints: NO
23.  ***** 4. row *****
24.      Engine: BLACKHOLE
25.      Support: YES
26.      Comment: /dev/null storage engine ( anything you write to it disappears)
27.      Transactions: NO
28.      XA: NO
29.      Savepoints: NO
30.  ***** 5. row *****
31.      Engine: MyISAM
32.      Support: YES
33.      Comment: MyISAM storage engine
34.      Transactions: NO
35.      XA: NO
36.      Savepoints: NO
37.  ***** 6. row *****
38.      Engine: CSV

```

[Top](#)

```

39.      Support: YES
40.      Comment: CSV storage engine
41. Transactions: NO
42.      XA: NO
43.      Savepoints: NO
44. ***** 7. row *****
45.      Engine: ARCHIVE
46.      Support: YES
47.      Comment: Archive storage engine
48. Transactions: NO
49.      XA: NO
50.      Savepoints: NO
51. ***** 8. row *****
52.      Engine: PERFORMANCE_SCHEMA
53.      Support: YES
54.      Comment: Performance Schema
55. Transactions: NO
56.      XA: NO
57.      Savepoints: NO
58. ***** 9. row *****
59.      Engine: FEDERATED
60.      Support: NO           //此引擎不被支持
61.      Comment: Federated MySQL storage engine
62. Transactions: NULL
63.      XA: NULL
64.      Savepoints: NULL
65. 9 rows in set (0.01 sec)

```

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：

```

01. my sql> SHOW VARIABLES LIKE 'default_storage_engine';
02. +-----+-----+
03. | Variable_name | Value |
04. +-----+-----+
05. | default_storage_engine | InnoDB |
06. +-----+-----+
07. 1 row in set (0.00 sec)

```

[Top](#)

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：

```

01.  mysql> SET default_storage_engine=MyISAM;           //改用MyISAM引擎
02.  Query OK, 0 rows affected (0.00 sec)
03.
04.  mysql> SHOW VARIABLES LIKE 'default_storage_engine'; //确认结果
05.  +-----+
06.  | Variable_name | Value |
07.  +-----+
08.  | default_storage_engine | MyISAM |
09.  +-----+
10.  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：

```

01.  [ root@dbsvr1 ~] # vim /etc/my.cnf
02.  [ mysql ]
03.  ...
04.  default_storage_engine=MEMORY           //改用MEMORY引擎
05.
06.  [ root@dbsvr1 ~] # systemctl restart mysqld.service //重启服务

```

重新登入 `mysql>` 确认修改结果：

```

01.  [ root@dbsvr1 ~] # mysql -u root -p
02.  Enter password:
03.  Welcome to the MySQL monitor. Commands end with ; or \g.
04.  Your MySQL connection id is 3
05.  Server version: 5.7.17 MySQL Community Server (GPL)
06.
07.  Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.
08.
09.  Oracle is a registered trademark of Oracle Corporation and/or its
10.  affiliates. Other names may be trademarks of their respective
11.  owners.

```

[Top](#)

```

12.
13.   Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
14.
15.
16.   my sql> SHOW VARIABLES LIKE 'default_storage_engine';
17.   +-----+-----+
18.   | Variable_name      | Value |
19.   +-----+-----+
20.   | default_storage_engine | MEMORY |           //默认引擎已修改
21.   +-----+-----+
22.   1 row in set ( 0.00 sec)
23.
24.   my sql> exit
25.   Bye

```

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：

```

01.   [ root@dbsvr1 ~] # vim /etc/my.cnf
02.   [ my sql]
03.   ...
04.   default_storage_engine=InnoDB
05.   [ 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）：

```

01.   [ root@dbsvr1 ~] # mysql -u root -p -e "SHOW VARIABLES LIKE 'default_storage_engine';
02.   Enter password:
03.   +-----+-----+
04.   | Variable_name      | Value |
05.   +-----+-----+
06.   | default_storage_engine | InnoDB |
07.   +-----+-----+

```



2 数据导入/导出

[Top](#)

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：

```

01. [root@dbsvr1 ~] # man 5 passwd
02. ...
03. Each line of the file describes a single user, and contains seven colon-separated fields:
04.
05.
06. name: password: UID: GID: GEOS: directory: shell
07.
08. The field are as follows: //以下详细解释各字段的作用
09.
10. name This is the user's login name. It should not contain capital
11. letters.
12.
13. password This is either the encrypted user password, an asterisk ( * ), or
14. the letter 'x'. ( See pwconv( 8 ) for an explanation of 'x' .)
15.
16. UID The privileged root login account ( superuser ) has the user ID 0.
17.
18. GID This is the numeric primary group ID for this user. ( Additional
19. groups for the user are defined in the system group file; see
20. group( 5 ) ).
21.
22. GEOS stands for "General Electric Comprehensive Operating Sys-
23. tem", which was renamed to GCOS when GE's large systems division
24. was sold to Honeywell. Dennis Ritchie has reported: "Sometimes
25. we sent printer output or batch jobs to the GCOS machine. The
26. gcos field in the password file was a place to stash the infor-
27. mation for the $IDENTcard. Not elegant."
28.
29. directory This is the user's home directory: the initial directory where
30. the user is placed after logging in. The value in this field is

```

[Top](#)

31. used to set the HOME environment variable.
- 32.
33. shell This is the program to run at login (if empty , use /bin/sh) . If
34. set to a nonexistent executable, the user will be unable to
35. login through login(1) . The value in this field is used to set
36. the SHELL environment variable.
37.

1) 新建userdb库、user表

以数据库用户root登入MySQL服务：

01. [root@dbsvr1 ~] # my sql - u root - p
02. Enter password:
03. Welcome to the My SQL monitor. Commands end with ; or \g.
04. Your My SQL connection id is 5
05. Server version: 5.7.17 My SQL Community Server (GPL)
- 06.
07. Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.
- 08.
09. Oracle is a registered trademark of Oracle Corporation and/or its
10. affiliates. Other names may be trademarks of their respective
11. owners.
- 12.
13. Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
- 14.
15. my sql>

新建userdb库，切换到userdb库：

01. my sql> CREATE DATABASE userdb;
02. Query OK, 1 row affected (0.00 sec)
- 03.
04. my sql> USE userdb;
05. Database changed

新建user表，字段设置及相关操作参考如下：

[Top](#)

```

01.  my sql> CREATE TABLE user(
02.      - > username varchar( 24) NOT NULL,
03.      - > password varchar( 48) DEFAULT 'x',
04.      - > uid int( 5) NOT NULL,
05.      - > gid int( 5) NOT NULL,
06.      - > fullname varchar( 48) ,
07.      - > homedir varchar( 64) NOT NULL,
08.      - > shell varchar( 24) NOT NULL
09.      - > );
10.  Query OK, 0 rows affected ( 0.70 sec)

```

确认user表的结构：

```

01.  my sql> DESC user;
02.  +-----+-----+-----+-----+-----+
03.  | Field | Type | Null | Key | Default | Extra |
04.  +-----+-----+-----+-----+-----+
05.  | username | varchar( 24) | NO | | NULL | |
06.  | password | varchar( 48) | YES | | x | |
07.  | uid | int( 5) | NO | | NULL | |
08.  | gid | int( 5) | NO | | NULL | |
09.  | fullname | varchar( 48) | YES | | NULL | |
10.  | homedir | varchar( 64) | NO | | NULL | |
11.  | shell | varchar( 24) | NO | | NULL | |
12.  +-----+-----+-----+-----+-----+
13.  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：

```

01.  my sql> LOAD DATA INFILE '/etc/passwd' INTO TABLE userlist FIELDS TERMINATED BY ':'
02.  ERROR 1290 ( HY000) : The MySQL server is running with the -- secure-file-priv option so it
03.  my sql> show variables like '%secure%';
04.  +-----+-----+
05.  | Variable_name | Value |
06.  +-----+-----+
07.  | require_secure_transport | OFF |
08.  | secure_auth | ON |

```

[Top](#)


```

09.  | secure_file_priv      | /var/lib/mysql-files/ |
10.  +-----+
11.  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：

```

01.  [root@dbsvr1 ~] #cp /etc/passwd /var/lib/mysql-files/
02.  mysql> LOAD DATA INFILE '/var/lib/mysql-files/passwd'
03.      -> INTO TABLE userlist
04.      -> FIELDS TERMINATED BY ':';
05.  Query OK, 39 rows affected (0.11 sec)
06.  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：

```

01.  mysql> SELECT COUNT(*) FROM user;
02.  +-----+
03.  | COUNT(*) |
04.  +-----+
05.  |      39 |           //user表有39条记录
06.  +-----+
07.  1 row in set (0.00 sec)
08.
09.  mysql> SELECT COUNT(*) FROM user2;
10.  +-----+
11.  | COUNT(*) |
12.  +-----+
13.  |      39 |           //user2表也有39条记录
14.  +-----+

```

[Top](#)

15. 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：

```

01.  my sql> SELECT username,uid,gid,homedir,shell
02.      - > FROM user LIMIT 10;
03.  +-----+-----+-----+-----+-----+
04.  | username | uid | gid | homedir      | shell      |
05.  +-----+-----+-----+-----+-----+
06.  | root    | 0   | 0   | /root        | /bin/bash  |
07.  | bin     | 1   | 1   | /bin         | /sbin/nologin |
08.  | daemon  | 2   | 2   | /sbin        | /sbin/nologin |
09.  | adm     | 3   | 4   | /var/adm     | /sbin/nologin |
10.  | lp      | 4   | 7   | /var/spool/lpd | /sbin/nologin |
11.  | sync    | 5   | 0   | /sbin        | /bin/sync   |
12.  | shutdown| 6   | 0   | /sbin        | /sbin/shutdown |
13.  | halt    | 7   | 0   | /sbin        | /sbin/halt   |
14.  | mail    | 8   | 12  | /var/spool/mail | /sbin/nologin |
15.  | operator| 11  | 0   | /root        | /sbin/nologin |
16.  +-----+-----+-----+-----+-----+
17.  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：

```

01.  my sql> SELECT username,uid,gid,homedir,shell
02.      - > FROM user2 LIMIT 10;
03.  +-----+-----+-----+-----+-----+
04.  | username | uid | gid | homedir      | shell      |
05.  +-----+-----+-----+-----+-----+
06.  | root    | 0   | 0   | /root        | /bin/bash  |
07.  | bin     | 1   | 1   | /bin         | /sbin/nologin |
08.  | daemon  | 2   | 2   | /sbin        | /sbin/nologin |
09.  | adm     | 3   | 4   | /var/adm     | /sbin/nologin |
10.  | lp      | 4   | 7   | /var/spool/lpd | /sbin/nologin |
11.  | sync    | 5   | 0   | /sbin        | /bin/sync   |
12.  | shutdown| 6   | 0   | /sbin        | /sbin/shutdown |
13.  | halt    | 7   | 0   | /sbin        | /sbin/halt   |
14.  | mail    | 8   | 12  | /var/spool/mail | /sbin/nologin |
15.  | operator| 11  | 0   | /root        | /sbin/nologin |

```

[Top](#)

16. +-----+-----+-----+-----+-----+-----+
 17. 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

01. my sql> ALTER TABLE user
 02. -> ADD sn int(4) AUTO_INCREMENT PRIMARY KEY FIRST;
 03. Query OK, 0 rows affected (0.62 sec)
 04. 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：

01. my sql> SELECT sn,username,uid,gid,homedir
 02. -> FROM user LIMIT 10;
 03. +---+-----+-----+-----+-----+
 04. | sn | username | uid | gid | homedir |
 05. +---+-----+-----+-----+-----+
 06. | 1 | root | 0 | 0 | /root |
 07. | 2 | bin | 1 | 1 | /bin |
 08. | 3 | daemon | 2 | 2 | /sbin |
 09. | 4 | adm | 3 | 4 | /var/adm |
 10. | 5 | lp | 4 | 7 | /var/spool/lpd |
 11. | 6 | sync | 5 | 0 | /sbin |
 12. | 7 | shutdown | 6 | 0 | /sbin |
 13. | 8 | halt | 7 | 0 | /sbin |
 14. | 9 | mail | 8 | 12 | /var/spool/mail |
 15. | 10 | operator | 11 | 0 | /root |
 16. +---+-----+-----+-----+-----+
 17. 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。

[Top](#)

1) 确认存放导出数据的文件夹

```

01. [root@dbsvr1 ~] # ls -ld /var/lib/mysql-files/
02. 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

```

01. [root@dbsvr1 ~] # mkdir /myload ; chown mysql /myload
02. [root@dbsvr1 ~] # vim /etc/my.cnf
03. [mysql]
04. secure_file_priv="/myload"
05. [root@dbsvr1 ~] # systemctl restart mysqld
06. mysql> show variables like "secure_file_priv";
07. +-----+-----+
08. | Variable_name | Value |
09. +-----+-----+
10. | 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：

```

01. mysql> SELECT * FROM userdb.user WHERE uid<100
02. -> INTO OUTFILE '/myload/user2.txt'
03. -> FIELDS TERMINATED BY ':';
04. 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：

```

01. [root@dbsvr1 ~] # wc -l /myload/user2.txt
02. 24 /myload/user2.txt

```

查看/myload/user2.txt文件的最后10行内容：

```

01. [root@dbsvr1 ~] # tail /myload/user2.txt
02. 19: avahi:x:70:70:Avahi mDNS/DNS-SD Stack:/var/run/avahi-daemon:/sbin/nologin
03. 24: rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin

```

[Top](#)

04. 25: rpcuser: x: 29: 29: RPC Service User: /var/lib/nfs:/sbin/nologin
05. 28: radvd: x: 75: 75: radvd user: /:/sbin/nologin
06. 29: ntp: x: 38: 38: /etc/ntp:/sbin/nologin
07. 33: gdm: x: 42: 42: /var/lib/gdm:/sbin/nologin
08. 35: postfix: x: 89: 89: /var/spool/postfix:/sbin/nologin
09. 36: sshd: x: 74: 74: Privilege-separated SSH: /var/empty/ssh:/sbin/nologin
10. 37: tcpdump: x: 72: 72: /:/sbin/nologin
11. 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

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：

01. [root@dbsvr1 ~] # mysql -uroot -p
02. Enter password:
03. Welcome to the MySQL monitor. Commands end with ; or \g.
04. Your MySQL connection id is 19
05. Server version: 5.7.17 MySQL Community Server (GPL)
- 06.
07. Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.
- 08.
09. Oracle is a registered trademark of Oracle Corporation and/or its
10. affiliates. Other names may be trademarks of their respective
11. owners.
- 12.
13. Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
- 14.
15. mysql> use userdb;
16. Reading table information for completion of table and column names
17. You can turn off this feature to get a quicker startup with -A

[Top](#)

```

18.
19. Database changed
20. my sql> CREATE TABLE stu_info(
21.     - > name varchar( 12) NOT NULL,
22.     - > gender enum( 'boy', 'girl') DEFAULT 'boy',
23.     - > age int( 3) NOT NULL
24.     - > );
25. Query OK, 0 rows affected ( 0.23 sec)

```

删除stu_info表的所有记录：

```

01. my sql> DELETE FROM stu_info;
02. Query OK, 0 rows affected ( 0.00 sec)           //stu_info表刚建立 删除零条记录

```

确认删除结果：

```

01. my sql> SELECT * FROM stu_info;
02. 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：

```

01. my sql> INSERT stu_info VALUES
02.     - > ( 'Jim', 'girl', 24),
03.     - > ( 'Tom', 'boy', 21),
04.     - > ( 'Lily', 'girl', 20);
05. Query OK, 3 rows affected ( 0.15 sec)
06. Records: 3 Duplicates: 0 Warnings: 0

```

完成插入后确认表记录：

```

01. my sql> SELECT * FROM stu_info;
02. +-----+-----+-----+

```

[Top](#)

```

03.  | name | gender | age |
04.  +-----+-----+-----+
05.  | Jim | girl | 24 |
06.  | Tom | boy | 21 |
07.  | Lily | girl | 20 |
08.  +-----+-----+-----+
09.  3 rows in set (0.00 sec)

```

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：

```

01.  my sql> INSERT INTO stu_info( name,age)
02.      -> VALUES( 'Jerry',27);
03.  Query OK, 1 row affected (0.04 sec)

```

类似的，再插入用户Mike的年龄信息：

```

01.  my sql> INSERT INTO stu_info( name,age)
02.      -> VALUES( 'Mike',21);
03.  Query OK, 1 row affected (0.05 sec)

```

确认目前stu_info表的所有记录：

```

01.  my sql> SELECT * FROM stu_info;
02.  +-----+-----+-----+
03.  | name | gender | age |
04.  +-----+-----+-----+
05.  | Jim | girl | 24 |
06.  | Tom | boy | 21 |
07.  | Lily | girl | 20 |
08.  | Jerry | boy | 27 |
09.  | Mike | boy | 21 |
10.  +-----+-----+-----+
11.  5 rows in set (0.00 sec)

```

[Top](#)

3) 更新表记录时，若未限制条件，则适用于所有记录
将stu_info表中所有记录的age设置为10：

```
01.  my sql> UPDATE stu_info SET age=10;
02.  Query OK, 5 rows affected ( 0.04 sec)
03.  Rows matched: 5 Changed: 5 Warnings: 0
```

确认更新结果：

```
01.  my sql> SELECT * FROM stu_info;
02.  +-----+-----+-----+
03.  | name | gender | age |
04.  +-----+-----+-----+
05.  | Jim  | girl  | 10 |
06.  | Tom  | boy   | 10 |
07.  | Lily | girl  | 10 |
08.  | Jerry| boy   | 10 |
09.  | Mike | boy   | 10 |
10.  +-----+-----+-----+
11.  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：

```
01.  my sql> UPDATE stu_info SET age=20
02.  -> WHERE gender='boy';
03.  Query OK, 3 rows affected ( 0.04 sec)
04.  Rows matched: 3 Changed: 3 Warnings: 0
```

确认更新结果：

```
01.  my sql> SELECT * FROM stu_info;
02.  +-----+-----+-----+
03.  | name | gender | age |
04.  +-----+-----+-----+
05.  | Jim  | girl  | 10 |
```

[Top](#)


```

06.  | Tom | boy | 20 |
07.  | Lily | girl | 10 |
08.  | Jerry | boy | 20 |
09.  | Mike | boy | 20 |
10.  +-----+-----+-----+
11.  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：

```

01.  my sql> DELETE FROM stu_info WHERE age < 18;
02.  Query OK, 2 rows affected (0.03 sec)

```

确认删除结果：

```

01.  my sql> SELECT * FROM stu_info;
02.  +-----+-----+-----+
03.  | name | gender | age |
04.  +-----+-----+-----+
05.  | Tom | boy | 20 |
06.  | Jerry | boy | 20 |
07.  | Mike | boy | 20 |
08.  +-----+-----+-----+
09.  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：

```

01.  my sql> DELETE FROM stu_info;
02.  Query OK, 3 rows affected (0.00 sec)

```

确认删除结果：

```

01.  my sql> SELECT * FROM stu_info;
02.  Empty set (0.00 sec)

```

[Top](#)

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：

```
01. [root@dbsvr1 ~] # mysql -u root -p
02. Enter password:
03. Welcome to the MySQL monitor. Commands end with ; or \g.
04. Your MySQL connection id is 5
05. Server version: 5.6.15 MySQL Community Server (GPL)
06.
07. Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.
08.
09. Oracle is a registered trademark of Oracle Corporation and/or its
10. affiliates. Other names may be trademarks of their respective
11. owners.
12.
13. Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
14.
15. mysql>
```

打开test库：

```
01. mysql> USE userdb;
02. Reading table information for completion of table and column names
03. You can turn off this feature to get a quicker startup with -A
04.
05. Database changed
```

[Top](#)

创建stu_info表，包括name、gender、age三个字段：

```

01.  my sql> CREATE TABLE stu_info(
02.      -> name varchar( 12) NOT NULL,
03.      -> gender enum( 'boy','girl') DEFAULT 'boy',
04.      -> age int( 3) NOT NULL
05.      -> );
06.  Query OK, 0 rows affected ( 0.03 sec)

```

确认表结构：

```

01.  my sql> DESC stu_info;
02.  +-----+-----+-----+-----+-----+
03.  | Field | Type          | Null | Key | Default | Extra |
04.  +-----+-----+-----+-----+-----+
05.  | name  | varchar( 12)  | NO   |     | NULL    |      |
06.  | gender| enum( 'boy','girl') | YES  |     | boy     |      |
07.  | age   | int( 3)       | NO   |     | NULL    |      |
08.  +-----+-----+-----+-----+-----+
09.  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

```

01.  my sql> INSERT INTO stu_info VALUES
02.      -> ( 'Jim','girl',24) ,
03.      -> ( 'Tom','boy',21) ,
04.      -> ( 'Lily','girl',20) ,
05.      -> ( 'Jerry','boy',27) ,
06.      -> ( 'Mike','boy',21)
07.      -> ;
08.  Query OK, 5 rows affected ( 0.06 sec)
09.  Records: 5 Duplicates: 0 Warnings: 0

```

确认stu_info表的所有记录内容：

[Top](#)

```

01.  my sql> SELECT * FROM stu_info;
02.  +-----+-----+-----+

```

```

03.  | name | gender | age |
04.  +-----+-----+-----+
05.  | Jim  | girl  | 24 |
06.  | Tom  | boy   | 21 |
07.  | Lily | girl  | 20 |
08.  | Jerry| boy   | 27 |
09.  | Mike | boy   | 21 |
10.  +-----+-----+-----+
11.  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）：

```

01.  my sql> SELECT count(*) FROM stu_info;
02.  +-----+
03.  | count(*) |
04.  +-----+
05.  |      5 |
06.  +-----+
07.  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：

```

01.  my sql> SELECT avg(age),max(age),min(age) FROM stu_info;
02.  +-----+-----+-----+
03.  | avg(age) | max(age) | min(age) |
04.  +-----+-----+-----+
05.  | 22.6000 | 27 | 20 |
06.  +-----+-----+-----+
07.  1 row in set (0.00 sec)

```

计算stu_info表中男学员的个数：

```

01.  my sql> SELECT count(gender) FROM stu_info WHERE gender='boy';
02.  +-----+
03.  | count(gender) |

```

[Top](#)

```

04.  +-----+
05.  |      3 |
06.  +-----+
07.  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：

```

01.  my sql> SELECT * FROM stu_info WHERE age=21;
02.  +-----+-----+-----+
03.  | name | gender | age |
04.  +-----+-----+-----+
05.  | Tom  | boy   | 21 |
06.  | Mike | boy   | 21 |
07.  +-----+-----+-----+
08.  2 rows in set ( 0.00 sec)

```

列出stu_info表中年龄超过21岁的学员记录：

```

01.  my sql> SELECT * FROM stu_info WHERE age>21;
02.  +-----+-----+-----+
03.  | name | gender | age |
04.  +-----+-----+-----+
05.  | Jim  | girl   | 24 |
06.  | Jerry | boy    | 27 |
07.  +-----+-----+-----+
08.  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：

```

01.  my sql> SELECT * FROM stu_info WHERE age>=21;
02.  +-----+-----+-----+
03.  | name | gender | age |
04.  +-----+-----+-----+
05.  | Jim  | girl   | 24 |
06.  | Tom  | boy    | 21 |
07.  | Jerry | boy    | 27 |

```

[Top](#)

```

08.  | Mike | boy | 21 |
09.  +-----+-----+-----+
10.  4 rows in set (0.00 sec)

```

列出stu_info表中年龄在20岁和24岁之间的学员记录：

```

01.  my sql> SELECT * FROM stu_info WHERE age BETWEEN 20 and 24;
02.  +-----+-----+-----+
03.  | name | gender | age |
04.  +-----+-----+-----+
05.  | Jim | girl | 24 |
06.  | Tom | boy | 21 |
07.  | Lily | girl | 20 |
08.  | Mike | boy | 21 |
09.  +-----+-----+-----+
10.  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：

```

01.  my sql> SELECT * FROM stu_info WHERE age < 23 AND gender='girl';
02.  +-----+-----+-----+
03.  | name | gender | age |
04.  +-----+-----+-----+
05.  | Lily | girl | 20 |
06.  +-----+-----+-----+
07.  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：

```

01.  my sql> SELECT * FROM stu_info WHERE age < 23 OR gender='girl';
02.  +-----+-----+-----+
03.  | name | gender | age |
04.  +-----+-----+-----+
05.  | Jim | girl | 24 |
06.  | Tom | boy | 21 |
07.  | Lily | girl | 20 |

```

[Top](#)

```

08.  | Mike | boy | 21 |
09.  +-----+-----+-----+
10.  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：

```

01.  my sql> SELECT * FROM stu_info WHERE name IN
02.      -> ('Jim','Tom','Mickey','Minnie');
03.  +-----+-----+-----+
04.  | name | gender | age |
05.  +-----+-----+-----+
06.  | Jim | girl | 24 |
07.  | Tom | boy | 21 |
08.  +-----+-----+-----+
09.  2 rows in set (0.00 sec)

```

4) 使用SELECT做数学计算

计算1234与5678的和：

```

01.  my sql> SELECT 1234+5678;
02.  +-----+
03.  | 1234+5678 |
04.  +-----+
05.  |      6912 |
06.  +-----+
07.  1 row in set (0.00 sec)

```

计算1234与5678的乘积：

```

01.  my sql> SELECT 1234*5678;
02.  +-----+
03.  | 1234*5678 |
04.  +-----+
05.  |  7006652 |
06.  +-----+
07.  1 row in set (0.00 sec)

```

[Top](#)

计算1.23456789除以3的结果：

```

01.  my sql> SELECT 1.23456789/3;
02.  +-----+
03.  | 1.23456789/3 |
04.  +-----+
05.  | 0.411522630000 |
06.  +-----+
07.  1 row in set ( 0.00 sec)

```

输出stu_info表各学员的姓名、15年后的年龄：

```

01.  my sql> SELECT name,age+15 FROM stu_info;
02.  +-----+-----+
03.  | name | age+15 |
04.  +-----+-----+
05.  | Jim  | 39    |
06.  | Tom  | 36    |
07.  | Lily | 35    |
08.  | Jerry| 42    |
09.  | Mike | 36    |
10.  +-----+-----+
11.  5 rows in set ( 0.00 sec)

```

5) 使用模糊查询，LIKE引导

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：

```

01.  my sql> SELECT * FROM stu_info WHERE name LIKE 'J%';
02.  +-----+-----+-----+
03.  | name | gender | age |
04.  +-----+-----+-----+
05.  | Jim  | girl  | 24  |
06.  | Jerry| boy   | 27  |
07.  +-----+-----+-----+
08.  2 rows in set ( 0.00 sec)

```

[Top](#)

列出stu_info表中姓名以 “J” 开头且只有3个字母的学员记录：

```

01.  my sql> SELECT * FROM stu_info WHERE name LIKE 'J__';
02.  +-----+-----+-----+
03.  | name | gender | age |
04.  +-----+-----+-----+
05.  | Jim  | girl  | 24 |
06.  +-----+-----+-----+
07.  1 row in set ( 0.00 sec)

```

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：

```

01.  my sql> SELECT * FROM stu_info WHERE name REGEXP '^J.*y$';
02.  +-----+-----+-----+
03.  | name | gender | age |
04.  +-----+-----+-----+
05.  | Jerry | boy   | 27 |
06.  +-----+-----+-----+
07.  1 row in set ( 0.00 sec)

```

效果等同于：

```

01.  my sql> SELECT * FROM stu_info WHERE name Like 'J%y';
02.  +-----+-----+-----+
03.  | name | gender | age |
04.  +-----+-----+-----+
05.  | Jerry | boy   | 27 |
06.  +-----+-----+-----+
07.  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：

```

01.  my sql> SELECT * FROM stu_info WHERE name REGEXP '^J|y$';
02.  +-----+-----+-----+
03.  | name | gender | age |
04.  +-----+-----+-----+

```

[Top](#)

```

05.  | Jim | girl | 24 |
06.  | Lily | girl | 20 |
07.  | Jerry | boy | 27 |
08.  +-----+-----+-----+
09.  3 rows in set (0.00 sec)

```

效果等同于：

```

01.  mysql> SELECT * FROM stu_info WHERE name Like 'J%' OR name Like '%y';
02.  +-----+-----+-----+
03.  | name | gender | age |
04.  +-----+-----+-----+
05.  | Jim | girl | 24 |
06.  | Lily | girl | 20 |
07.  | Jerry | boy | 27 |
08.  +-----+-----+-----+
09.  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：

```

01.  mysql> SELECT * FROM stu_info GROUP BY age;
02.  +-----+-----+-----+
03.  | name | gender | age |
04.  +-----+-----+-----+
05.  | Lily | girl | 20 |
06.  | Tom | boy | 21 |
07.  | Jim | girl | 24 |
08.  | Jerry | boy | 27 |
09.  +-----+-----+-----+
10.  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：

```

01.  mysql> SELECT * FROM stu_info GROUP BY age ASC;
02.  +-----+-----+-----+
03.  | name | gender | age |

```

[Top](#)

```

04.  +-----+-----+-----+
05.  | Lily | girl | 20 |
06.  | Tom  | boy  | 21 |
07.  | Jim  | girl | 24 |
08.  | Jerry| boy  | 27 |
09.  +-----+-----+-----+
10.  4 rows in set (0.00 sec)

```

若要按降序 (Descend) 排列，则将ASC改为DESC即可：

```

01.  my sql> SELECT * FROM stu_info GROUP BY age DESC;
02.  +-----+-----+-----+
03.  | name | gender | age |
04.  +-----+-----+-----+
05.  | Jerry| boy  | 27 |
06.  | Jim  | girl | 24 |
07.  | Tom  | boy  | 21 |
08.  | Lily | girl | 20 |
09.  +-----+-----+-----+
10.  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：

```

01.  my sql> SELECT * FROM stu_info LIMIT 3;
02.  +-----+-----+-----+
03.  | name | gender | age |
04.  +-----+-----+-----+
05.  | Jim  | girl | 24 |
06.  | Tom  | boy  | 21 |
07.  | Lily | girl | 20 |
08.  +-----+-----+-----+
09.  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](#)

```

01.  my sql> SELECT * FROM stu_info GROUP BY age DESC LIMIT 3;

```

```

02.  +-----+-----+-----+
03.  | name | gender | age |
04.  +-----+-----+-----+
05.  | Jerry | boy   | 27 |
06.  | Jim   | girl  | 24 |
07.  | Tom   | boy   | 21 |
08.  +-----+-----+-----+
09.  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：

```

01.  my sql> SELECT gender,count( gender) FROM stu_info GROUP BY gender;
02.  +-----+-----+
03.  | gender | count( gender) |
04.  +-----+-----+
05.  | boy   | 3 |
06.  | girl  | 2 |
07.  +-----+-----+
08.  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：

```

01.  my sql> SELECT gender AS '性别',count( gender) AS '人数'
02.  -> FROM stu_info GROUP BY gender;
03.  +-----+-----+
04.  | 性别 | 人数 |
05.  +-----+-----+
06.  | boy  | 3 |
07.  | girl | 2 |
08.  +-----+-----+
09.  2 rows in set (0.00 sec)

```

[Top](#)