TTS 11.0 COOKBOOK

(NSD RDBMS1 DAY03)

版本编号 11.0

2019-06

达内 IT 培训集团

NSD RDBMS1 DAY03

1. 案例 1：数据导入导出

- **问题**
  - 修改检索目录为/myload
  - 将/etc/passwd 文件导入 db3 库的 user 表里，并添加行号字段。
  - 将 db3 库 user 表所有记录导出，存到/myload/user.txt 文件里。

步骤一：修改检索目录为/myload

1）修改配置文件，重启服务

```
]# mkdir  /myload
]# chown  mysql  /myload
]# vim  /etc/my.cnf
       [mysqld]
       secure_file_priv="/myload”
:wq
]# systemctl  restart mysqld


mysql> show  variables  like   “secure_file_priv” ;  //查看
 +------------------+-----------------------+
| Variable_name    | Value                       |
+------------------+-----------------------+
| secure_file_priv   | /myload/   |
+------------------+-----------------------+

Mysql>
```

2）新建 db3 库、user 表

```
[root@dbsvr1 ~]# mysql -u root -p123456
mysql> CREATE DATABASE db3;
create table db3.user(
       name char(50),
       password  char(1),
       uid int,
       gid int,
       comment  char(150),
       homedir char(50),
       shell   char(50)
);
Query OK, 0 rows affected (0.70 sec)
Mysql>
```

步骤二：将/etc/passwd 文件导入 db3 库的 user 表里，并添加行号字段。

1）拷贝文件到检索目录下

```
[root@dbsvr1 ~]#
```

```
[root@dbsvr1 ~]# cp  /etc/passwd   /myload/
```

2）导入数据

```
[root@dbsvr1 ~]# mysql -uroot -ptarena
mysql> load data infile "/myload/passwd" into table db3.user
      fields terminated by ":" lines terminated by "\n" ; //导入数据

mysql> select  * from  db3.user;  //查看表记录

mysql> alter table  db3.user
    -> add
    -> id  int primary key auto_increment first;  //添加行号 id 字段

mysql> select  * from  db3.user; //查看表记录
```

步骤三：将 db3 库 user 表所有记录导出，存到/myload/user.txt 文件里。

1）查询要导出的数据

```
mysql> select  * from  db3.user ;
```

2）导出数据

```
mysql> select  * from  db3.user  into outfile "/myload/user1.txt";
```

3）查看文件内容

```
]# cat  /myload/user1.txt
```

2. 案例 2：管理表记录

- **问题**

练习表记录的操作

1) 练习插入表记录
2) 练习更新表记录
3) 练习查询表记录
4) 练习删除表记录

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：练习插入表记录

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)
```

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

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)
```

确认删除结果：

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

3. 案例 3： 匹配条件

- **问题**

具体要求如下：

- 练习数值比较的使用
- 练习字符比较的使用
- 练习逻辑比较的使用
- 练习模糊匹配的使用
- 练习正则匹配的使用
- 练习查询结果分组、排序、过滤、限制显示记录行数
- 练习聚集函数的使用
- 练习四则运算的使用**步骤**

实现此案例需要按照如下步骤进行。

步骤一：匹配条件练习

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）：

```
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：

```
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)
```

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：

```
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：

```
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：

```
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

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)
```

6）使用正则表达式，REGEXP

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)
```

效果等同于：

```
mysql> SELECT * FROM stu_info WHERE name Like 'J%' OR name Like '%y';
+-------+--------+-----+
| 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 ORDER 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 ORDER BY age ASC;
+-------+--------+-----+
| name  | gender | age |
+-------+--------+-----+
| Lily  | girl   |  20 |
| Tom   | boy    |  21 |
| Jim   | girl   |  24 |
| 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 ORDER 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：

```
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)
```

4. 案例 4： MySQL 管理工具

- **问题**

- 部署 LAMP+phpMyAdmin 平台

- **方案**

1．安装 httpd、mysql、php-mysql 及相关包
2．启动 httpd 服务程序
3．解压 phpMyAdmin 包，部署到网站目录
4．配置 config.inc.php，指定 MySQL 主机地址
5．浏览器访问、登录使用

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：准备软件的运行环境 lamp

```
[root@mysql6~]# rpm -q httpd  php  php-mysql    //检测是否安装软件包
未安装软件包 httpd
未安装软件包 php
未安装软件包 php-mysql
[root@mysql6~]# yum  -y   install   httpd    php    php-mysql  //装包
[root@mysql6~]# systemctl  start  httpd      //启动服务
[root@mysql6~]# systemctl  enable httpd     //设置开机自启
Created symlink from /etc/systemd/system/multi-user.target.wants/httpd.service to
/usr/lib/systemd/system/httpd.service.
```

步骤二：测试运行环境

```
[root@mysql6~]# vim  /var/www/html/test.php     //编辑页面测试文件
[root@mysql6~]# cat /var/www/html/test.php      //查看页面测试文件
<?php
$x=mysql_connect("localhost","root","123456");
if($x){   echo "ok";    }else{    echo "no";    };
?>
[root@mysql6~]# yum  -y   install  elinks     //安装测试网页工具
[root@mysql6~]#  elinks   --dump   http://localhost/test.php
   Ok                                          //验证测试页面成功
```

步骤三：安装软件包

1）物理机传输解压包给虚拟机 192.168.4.6

```
[root@room9pc 桌面]# scp phpMyAdmin-2.11.11-all-languages.tar.gz 192.168.4.6:/root/
root@192.168.4.6's password:
phpMyAdmin-2.11.11-a 100% 4218KB 122.5MB/s   00:00
```

2）虚拟机 192.168.4.6 解压 phpMyAdmin-2.11.11-all-languages.tar.gz 压缩包

```
[root@mysql6~]# tar -zxf phpMyAdmin-2.11.11-all-languages.tar.gz -C /var/www/html/
//-C 表示改变至目录
[root@mysql6~]# cd /var/www/html/
```

```
[root@mysql6~]# mv phpMyAdmin-2.11.11-all-languages  phpmyadmin //改变目录名
[root@mysql6~]#  chown  -R  apache:apache  phpmyadmin/ //改变 phpmyadmin 目录权限
```

步骤四：修改软件的配置文件定义管理的数据库服务器

切换到部署后的 phpmyadmin 程序目录，拷贝配置文件，并修改配置以正确指定 MySQL 服务器的地址

```
[root@mysql6html]# cd  phpmyadmin
[root@mysql6 phpmyadmin]# cp   config.sample.inc.php   config.inc.php
//备份主配置文件
[root@mysql6 phpmyadmin]# vim   config.inc.php  //编辑主配置文件
17 $cfg['blowfish_secret'] = 'plj123';     //给 cookie 做认证的值，可以随便填写
31 $cfg['Servers'][$i]['host'] = 'localhost';  //指定主机名，定义连接哪台服务器
:wq
```

步骤五：在客户端访问软件 管理数据库服务器

1) 在客户端访问软件,打开浏览器输入 http://192.168.4.6/phpmyadmin(数据库服务器地址）访问软件，如图-2 所示，用户名是 root，密码是 123456

图-2

2）登入成功后，如图-3 示，即可在授权范围内对 MySQL 数据库进行管理。

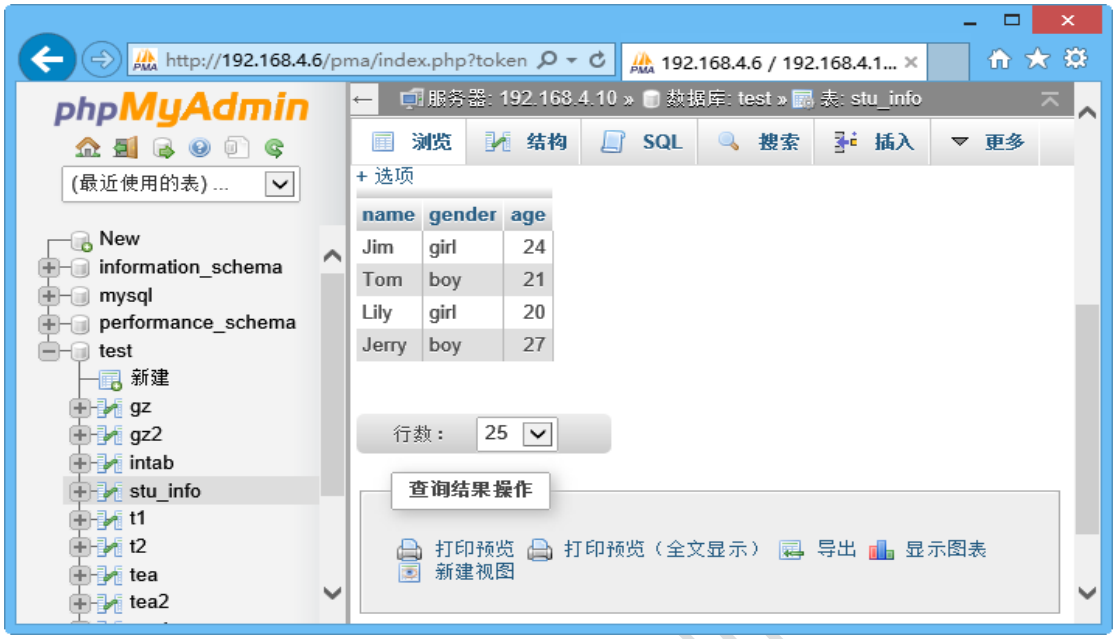

图-3