

NSD DBA2 DAY04

1. [案例1：视图的基本使用](#)
2. [案例2：视图进阶操作](#)
3. [案例3：创建存储过程](#)
4. [案例4：存储过程参数的使用](#)
5. [案例5：使用循环结构](#)

1 案例1：视图的基本使用

1.1 问题

- 把/etc/passwd文件的内容存储到db9库下的user表里
- 添加新字段id 存储记录的行号(在所有字段的前边)
- 创建视图v1 结构及数据user表的字段、记录一样。
- 创建视图v2 只有user表shell是/bin/bash用户信息。
- 分别对视图表和基表执行insert update delete 操作。
- 删除视图v1 和 v2

1.

1.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：视图的基本使用

什么是视图：是一种虚拟存在的表

内容与真实的表相似，包含一系列带有名称的列和行数据。

视图并不在数据库中以存储的数据的形式存在。

行和列的数据来自定义视图时查询所引用的基本表，并且在具体引用视图时动态生成。

更新视图的数据，就是更新基表的数据

更新基表数据，视图的数据也会跟着改变

1) 把/etc/passwd文件的内容存储到db9库下的user表里

```

01. [root@mysql51 ~]# mysql -u root -p123456
02. mysql> create database db9;
03. Query OK, 1 row affected ( 10.00 sec)
04. mysql> create table db9.user( username char( 20 ), password char( 1 ), uid \
05. int( 2 ), gid int( 2 ), comment char( 100 ), homedir char( 100 ), shell char( 50 ) );
06. //创建存储数据的表结构
07. Query OK, 0 rows affected ( 0.02 sec)
08. [root@mysql51 ~]# cp /etc/passwd /var/lib/mysql-files/
09. [root@mysql51 ~]# ls /var/lib/mysql-files/
10. passwd

```

[Top](#)

11. mysql> load data infile "/var/lib/mysql-files/passwd" into table db9.user fields terminate
12. Query OK, 41 rows affected (0.02 sec)
13. Records: 41 Deleted: 0 Skipped: 0 Warnings: 0

2) 添加新字段id 存储记录的行号(在所有字段的前边)

01. mysql> alter table db9.user add id int(2) primary key auto_increment first;
02. Query OK, 0 rows affected (0.04 sec)
03. Records: 0 Duplicates: 0 Warnings: 0
04. mysql> use db9;
05. mysql> desc user;
06. +-----+-----+-----+-----+-----+-----+
07. | Field | Type | Null | Key | Default | Extra |
08. +-----+-----+-----+-----+-----+-----+
09. | id | int(2) | NO | PRI | NULL | auto_increment |
10. | username | char(20) | YES | | NULL | |
11. | password | char(1) | YES | | NULL | |
12. | uid | int(2) | YES | | NULL | |
13. | gid | int(2) | YES | | NULL | |
14. | comment | char(100) | YES | | NULL | |
15. | homedir | char(100) | YES | | NULL | |
16. | shell | char(50) | YES | | NULL | |
17. +-----+-----+-----+-----+-----+-----+
18. 8 rows in set (0.00 sec)

3) 创建视图v1 结构及数据user表的字段、记录一样

01. mysql> create view v1 as select * from user;
02. Query OK, 0 rows affected (0.00 sec)

4) 创建视图v2 只有user表shell是/bin/bash用户信息

01. mysql> create view v2 as select shell from user;
02. Query OK, 0 rows affected (0.01 sec)

[Top](#)

5) 分别对视图表和基表执行insert update delete 操作

```
01.  my sql> insert into v1( username,uid) values( "jarry",9);      //插入记录
02.  Query OK, 1 row affected ( 0.00 sec)
03.
04.  my sql> update v1 set uid=9 where username="adm";      //更新记录
05.  Query OK, 1 row affected ( 0.01 sec)
06.  Rows matched: 1 Changed: 1 Warnings: 0
07.
08.  my sql> delete from v1 where uid=9;      //删除记录
09.  Query OK, 2 rows affected ( 0.01 sec)
```

6) 删除视图v1 和 v2

```
01.  my sql> drop view v1;
02.  Query OK, 0 rows affected ( 0.00 sec)
03.  my sql> drop view v2;
04.  Query OK, 0 rows affected ( 0.00 sec)
```

注意：对视图操作即是对基本操作，反之亦然！！

2 案例2：视图进阶操作

2.1 问题

- 练习OR REPLACE的选项使用
- 练习WITH LOCAL CHECK OPTION 选项的使用
- 练习WITH CASCADED CHECK OPTION 选项的使用

2.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：视图进阶操作

1) 创建视图完全格式

```
01.  my sql> create table user2 select  username,uid,gid from user limit 3;
02.  //快速建表 (user2表)
03.  Query OK, 3 rows affected ( 0.01 sec)
04.  Records: 3 Duplicates: 0 Warnings: 0
05.
06.  my sql> create table info select  username,uid,homedir,shell from user limit 5;
07.  //快速建表 (info表)
```

[Top](#)

08. Query OK, 5 rows affected (0.02 sec)
09. Records: 5 Duplicates: 0 Warnings: 0

查询user2.username=info.username的字段

```
01. mysql> select * from user2 left join info on user2.username=info.username;
02. +-----+-----+-----+-----+-----+-----+
03. | username | uid | gid | username | uid | homedir | shell |
04. +-----+-----+-----+-----+-----+-----+
05. | root    | 0   | 0   | root     | 0   | /root   | /bin/bash |
06. | bin     | 1   | 1   | bin      | 1   | /bin    | /sbin/nologin |
07. | daemon  | 2   | 2   | daemon   | 2   | /sbin   | /sbin/nologin |
08. +-----+-----+-----+-----+-----+-----+
09. 3 rows in set (0.00 sec)
```

2) 关联查询建的视图 默认不允许修改视图字段的值

```
01. mysql> create view v4 as select * from user2 left join info on user2.username=info.username;
02. ERROR 1060 (42S21): Duplicate column name 'username'
03.
04. mysql> create view v4 as select a.username as ausername,b.username as busername, a.
05. //创建成功
06. Query OK, 0 rows affected (0.00 sec)
07.
08. mysql> select * from v4;
09. +-----+-----+-----+-----+
10. | ausername | busername | auid | buid |
11. +-----+-----+-----+-----+
12. | root     | root     | 0    | 0    |
13. | bin      | bin      | 1    | 1    |
14. | daemon   | daemon   | 2    | 2    |
15. +-----+-----+-----+-----+
16. 3 rows in set (0.00 sec)
17.
18. mysql> desc v4;
19. +-----+-----+-----+-----+-----+-----+
20. | Field | Type | Null | Key | Default | Extra |
21. +-----+-----+-----+-----+-----+-----+
22. | ausername | char(20) | YES | | NULL | |
```

[Top](#)

```

23. | username | char(20) | YES | | NULL | |
24. | auid | int(2) | YES | | NULL | |
25. | buid | int(2) | YES | | NULL | |
26. +-----+-----+-----+-----+-----+
27. 4 rows in set (0.00 sec)

```

3) OR REPLACE的选项使用

创建时，若视图已存在，会替换已有的视图

语法格式：create or replace view 视图名 as select 查询； //达到修改已有视图的目的

```

01. mysql> create or replace view v4 as select a.username as ausername,b.username as bus
02. Query OK, 0 rows affected (0.00 sec)

```

4) WITH LOCAL CHECK OPTION

LOCAL和CASCADED关键字决定检查的范围

LOCAL 仅检查当前视图的限制

CASCADED 同时要满足基表的限制（默认值）

```

01. mysql> create table user1 select username,uid,shell from user where uid>=5 and uid <=40;
02. Query OK, 11 rows affected (0.01 sec)
03. Records: 11 Duplicates: 0 Warnings: 0
04.
05. mysql> create view v1 as select username,uid from user1 where uid<=20;
06. Query OK, 0 rows affected (0.01 sec)
07.
08. mysql> update v1 set uid=21 where username="sync";
09. //操作超过视图的条件限制 (uid<=20) 之后，在视图表里面查看不到，在基表里可以
10. Query OK, 1 row affected (0.01 sec)
11. Rows matched: 1 Changed: 1 Warnings: 0
12.
13. mysql> update user1 set uid=41 where username="ftp";
14. //基表在超过条件限制 (uid>=5 and uid <=40),在基表里依然可以查看到
15. Query OK, 1 row affected (0.00 sec)
16. Rows matched: 1 Changed: 1 Warnings: 0
17.
18. mysql> create table a select * from user where uid < 10;
19. //快速创建一个新表a

```

[Top](#)

```

20. Query OK, 7 rows affected ( 0.01 sec)
21. Records: 7 Duplicates: 0 Warnings: 0
22.
23. mysql> create view v3 as select * from a where uid < 10 with check option;
24. //不写默认为CASCADDED检查自己和a要满足的要求即可
25. Query OK, 0 rows affected ( 0.00 sec)
26.
27. mysql> update v3 set uid=9 where username="adm"; //更改成功
28. Query OK, 0 rows affected ( 0.01 sec)
29. Rows matched: 0 Changed: 0 Warnings: 0
30.
31.
32. mysql> create view v2 as select * from v1 where uid >= 5 with local check option;
33. //满足自身v2的要求
34. Query OK, 0 rows affected ( 0.00 sec)
35. mysql> update v2 set uid=9 where username="sync";
36. Query OK, 0 rows affected ( 0.00 sec)
37. Rows matched: 0 Changed: 0 Warnings: 0

```

5) WITH CASCADED CHECK OPTION

```

01. mysql> create view v5 as select * from v1 where uid >= 5 with cascaded check option;
02. Query OK, 0 rows affected ( 0.00 sec)

```

3 案例3：创建存储过程

3.1 问题

- 存储过程名称为p1
- 功能显示user表中 shell是/bin/bash的用户个数
- 调用存储过程p1

3.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：存储过程基本使用

1) 创建存储过程

```

01. mysql> delimiter // //定义定界符
02. mysql> create procedure say() //say() 随便写括号一定要有

```

[Top](#)

```

03.      - > begin
04.      - > select * from user where id<=10;
05.      - > end
06.      - > //
07.      Query OK, 0 rows affected ( 0.01 sec)
08.      my sql> delimiter ;      //把命令的定界符改回来，分号前有空格
09.      my sql> call say ( ) ;    //调用存储过程名,在括号里面不写参数时，可以不加括号

```

2) 查看存储过程

方法一：

```
01.      my sql> show procedure status\G;
```

方法二：

```
01.      my sql> select db,name,type from my sql.proc where name= "say ";
```

3) 删除存储过程

```

01.      my sql> drop procedure say ;
02.      Query OK, 0 rows affected ( 0.00 sec)

```

4) 创建存储过程名称为p1

▣ 功能显示user表中 shell是/bin/bash的用户

▣ 调用存储过程p1

```

01.      my sql> delimiter //
02.      my sql> create procedure p1( )
03.      - > begin
04.      - > select count( username)  from user where shell="/bin/bash";
05.      - > end
06.      - > //
07.      my sql> delimiter ;
08.      my sql> call p1( ) ;
09.      +-----+
10.      | shell |

```

[Top](#)

```

11.  +-----+
12.  | /bin/bash |
13.  | /bin/bash |
14.  +-----+
15.  2 rows in set ( 0.01 sec)
16.
17.  Query OK, 0 rows affected ( 0.01 sec)

```

4 案例4：存储过程参数的使用

4.1 问题

- 创建名为p2的存储过程
- 可以接收用户输入shell的名字
- 统计user表中用户输入shell名字的个数

4.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：存储过程参数的使用

1) 参数类型

MySQL存储过程，共有三种参数类型IN,OUT,INOUT

Create procedure 名称(

类型 参数名 数据类型,

类型 参数名 数据类型

)

in 输入参数 传递值给存储过程，必须在调用存储过程时指定，在存储过程中修改该参数的值不能；默认类型是in

out 输出参数 该值可在存储过程内部被改变，并可返回

inout 输入/输出参数 调用时指定，并且可被改变和返回

```

01.  my sql> delimiter //
02.  my sql> create procedure say2(in username char(10))
03.      - > begin
04.      - > select username;
05.      - > select * from user where username=username;
06.      - > end
07.      - > //
08.  Query OK, 0 rows affected ( 0.00 sec)
09.
10.  my sql> delimiter ;
11.  my sql> call say2("tom");

```

[Top](#)

2) 创建名为p2的存储过程，可以接收用户输入shell的名字，统计user表中用户输入shell名字的个数

```

01.  my sql> delimiter //
02.  my sql> create procedure p2( out number int)
03.      - > begin
04.      - > select count( username) into @number from user where shell!="/bin/bash";
05.      - > select @number;
06.      - > end
07.      - > //
08.  Query OK, 0 rows affected ( 0.01 sec)
09.  my sql> delimiter ;
10.  my sql> call p2( @number);
11.  +-----+
12.  | @number |
13.  +-----+
14.  |    38   |
15.  +-----+
16.  1 row in set ( 0.00 sec)
17.
18.  Query OK, 0 rows affected ( 0.00 sec)

```

5 案例5：使用循环结构

5.1 问题

- 定义名称为p3的存储过程
- 用户可以自定义显示user表记录的行数
- 若调用时用户没有输入行数，默认显示第1条记录

5.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：算数运算

1) 算数运算符号，如图-1所示：

[Top](#)

符号	描述	示例
+	加法运算	SET @var1=2+2; 4
-	减法运算	SET @var2=3-2; 1
*	乘法运算	SET @var3=3*2; 6
/	除法运算	SET @var4=10/3; 3.333333333
DIV	整除运算	SET @var5=10 DIV 3; 3
%	取模	SET @var6=10%3; 1

图-1

```

01.  my sql> set @z=1+2; select @z;
02.  Query OK, 0 rows affected ( 0.00 sec)
03.
04.  +-----+
05.  | @z |
06.  +-----+
07.  | 3 |
08.  +-----+
09.  1 row in set ( 0.00 sec)
10.  my sql> set @x=1; set @y=2; set @z=@x*@y; select @z;
11.  Query OK, 0 rows affected ( 0.00 sec)
12.
13.  Query OK, 0 rows affected ( 0.00 sec)
14.
15.  Query OK, 0 rows affected ( 0.00 sec)
16.
17.  +-----+
18.  | @z |
19.  +-----+
20.  | 2 |
21.  +-----+
22.  1 row in set ( 0.00 sec)
23.
24.  my sql> set @x=1; set @y=2; set @z=@x-@y; select @z;
25.  Query OK, 0 rows affected ( 0.00 sec)
26.
27.  Query OK, 0 rows affected ( 0.00 sec)
28.
29.  Query OK, 0 rows affected ( 0.00 sec)

```

[Top](#)

```

30.
31.  +-----+
32.  | @z |
33.  +-----+
34.  | - 1 |
35.  +-----+
36.  1 row in set ( 0.00 sec)
37.  my sql> set @x=1; set @y=2; set @z=@x/@y; select @z;
38.  Query OK, 0 rows affected ( 0.00 sec)
39.
40.  Query OK, 0 rows affected ( 0.00 sec)
41.
42.  Query OK, 0 rows affected ( 0.00 sec)
43.
44.  +-----+
45.  | @z |
46.  +-----+
47.  | 0.50000000 |
48.  +-----+
49.  1 row in set ( 0.00 sec)

```

declare调用变量不需要@其他都需要

调用变量时，有@符号的变量 如@x：调用的是用户自定义变量

没有@符号的变量 如x：调用的是存储过程的参数变量

```

01.  my sql> delimiter //
02.  my sql> create procedure say5( in bash char( 20) , in nologin char( 25) , out x int , out y int)
03.      - > begin
04.      - > declare z int ;
05.      - > set z=0;
06.      - > select count( username) into @x from user where shell=bash;
07.      - > select count( username) into @y from user where shell=nologin;
08.      - > set z=@x+@y;
09.      - > select z;
10.      - > end
11.      - > //
12.  Query OK, 0 rows affected ( 0.00 sec)
13.
14.  my sql> delimiter ;
15.  my sql> call say5( "/bin/bash", "/sbin/nologin", @x, @y );

```

[Top](#)

```

16.  +-----+
17.  | z |
18.  +-----+
19.  | 36 |
20.  +-----+
21.  1 row in set ( 0.00 sec)
22.
23.  Query OK, 0 rows affected ( 0.00 sec)

```

2) 条件判断，数值的比较如图-2所示：

类 型	用 途
=	等于
>、>=	大于、大于或等于
<、<=	小于、小于或等于
!=	不等于
BETWEEN .. AND ..	在 .. 与 .. 之间

图-2

逻辑比较、范围、空、非空、模糊、正则，如图-3所示：

类 型	用 途
OR、AND、!	逻辑或、逻辑与、逻辑非
IN ..、NOT IN ..	在 .. 范围内、不在 .. 范围内
IS NULL	字段的值为空
IS NOT NULL	字段的值不为空
LIKE	模糊匹配
REGEXP	正则匹配

图-3

顺序结构（if判断）当“条件成立”时执行命令序列,否则，不执行任何操作

```

01.  my sql> delimiter //
02.  my sql> create procedure say6( in x int( 1 ) )
03.      - > begin
04.      - > if x <= 10 then
05.      - > select * from user where id <=x;

```

[Top](#)

```

06.      - > end if ;
07.      - > end
08.      - > //
09.      Query OK, 0 rows affected ( 0.01 sec)
10.
11.      my sql> delimiter ;
12.      my sql> call say6( 1);      //条件判断成立，等于1是否成立
13.      +---+-----+-----+-----+-----+-----+-----+-----+
14.      | id | username | password | uid | gid | comment | homedir | shell |
15.      +---+-----+-----+-----+-----+-----+-----+-----+
16.      | 1 | root   | x       | 0   | 0   | root   | /root   | /bin/bash |
17.      +---+-----+-----+-----+-----+-----+-----+-----+
18.      1 row in set ( 0.00 sec)
19.
20.      Query OK, 0 rows affected ( 0.00 sec)
21.
22.      my sql> call say6( 2);
23.      +---+-----+-----+-----+-----+-----+-----+-----+
24.      | id | username | password | uid | gid | comment | homedir | shell |
25.      +---+-----+-----+-----+-----+-----+-----+-----+
26.      | 1 | root   | x       | 0   | 0   | root   | /root   | /bin/bash |
27.      | 2 | bin    | x       | 1   | 1   | bin    | /bin    | /sbin/nologin |
28.      +---+-----+-----+-----+-----+-----+-----+-----+
29.      2 rows in set ( 0.00 sec)
30.
31.      Query OK, 0 rows affected ( 0.00 sec)

```

3) 定义名称为p3的存储过程，用户可以自定义显示user表记录的行数，若调用时用户没有输入行数，默认显示第1条记录

```

01.      my sql> delimiter //
02.      my sql> create procedure p3( in linenum char( 10) )
03.      - > begin
04.      - > if linenum is null then
05.      - > set @linenum=1;
06.      - > select * from user where id=@linenum;
07.      - > else
08.      - > select linenum;
09.      - > select * from user where id=linenum;
10.      - > end if ;

```

[Top](#)

```

11.      - > end
12.      - > //
13.      Query OK, 0 rows affected ( 0.00 sec)
14.
15.      my sql> delimiter ;
16.      my sql> call p3( null);      //不输入查看的行数
17.      +---+-----+-----+-----+-----+-----+-----+-----+
18.      | id | username | password | uid | gid | comment | homedir | shell |
19.      +---+-----+-----+-----+-----+-----+-----+-----+
20.      | 1 | root   | x       | 0 | 0 | root   | /root   | /bin/bash |
21.      +---+-----+-----+-----+-----+-----+-----+-----+
22.      1 row in set ( 0.00 sec)
23.
24.      Query OK, 0 rows affected ( 0.00 sec)
25.
26.      my sql> call p3( 3);      //输入查看的行数
27.      +-----+
28.      | linenum |
29.      +-----+
30.      | 3      |
31.      +-----+
32.      1 row in set ( 0.00 sec)
33.
34.      +---+-----+-----+-----+-----+-----+-----+-----+
35.      | id | username | password | uid | gid | comment | homedir | shell |
36.      +---+-----+-----+-----+-----+-----+-----+-----+
37.      | 3 | daemon  | x       | 2 | 2 | daemon  | /sbin   | /sbin/nologin |
38.      +---+-----+-----+-----+-----+-----+-----+-----+
39.      1 row in set ( 0.00 sec)
40.
41.      Query OK, 0 rows affected ( 0.00 sec)

```

[Top](#)