

DBA2 DAY04



# DBA进阶

**NSD DBA2**

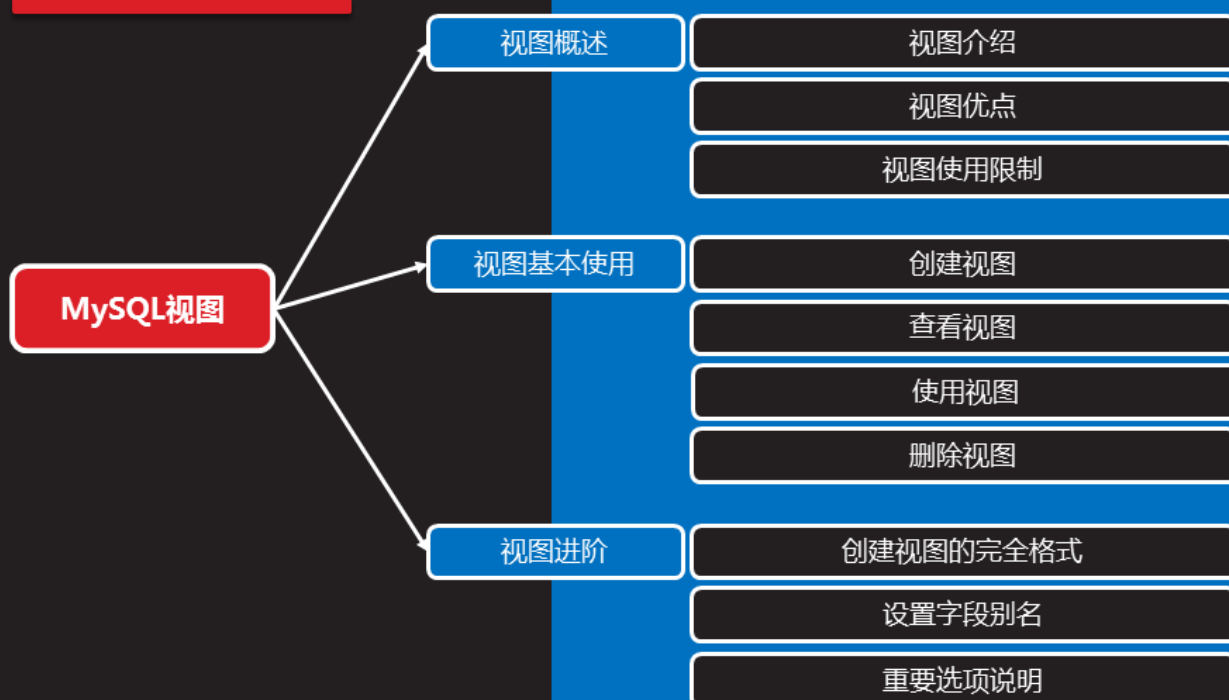
**DAY04**

# 内容

上午	09:00 ~ 09:30	作业讲解和回顾
	09:30 ~ 10:20	MySQL视图
	10:30 ~ 11:20	
	11:30 ~ 12:00	
下午	14:00 ~ 14:50	MySQL存储过程
	15:00 ~ 15:50	
	16:10 ~ 17:00	
	17:10 ~ 18:00	总结和答疑



## MySQL视图



# 视图概述

## 视图介绍

- 什么是视图(View)
  - 虚拟表
  - 内容与真实的表相似，有字段有记录
  - 视图并不在数据库中以存储的数据形式存在
  - 行和列的数据来自定义视图时查询所引用的基表，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



## 视图优点

知识讲解

- 简单
  - 用户不需关心视图中的数据如何查询获得
  - 视图中的数据已经是过滤好的符合条件的结果集
- 安全
  - 用户只能看到视图中的数据
- 数据独立
  - 一旦视图结构确定，可以屏蔽表结构对用户的影响



## 视图使用限制

知识讲解

- 不能在视图上创建索引
- 在视图的FROM子句中不能使用子查询
- 以下情形中的视图是不可更新的
  - 包含以下关键字的SQL语句：聚合函数(SUM、MIN、MAX、COUNT等)、DISTINCT、GROUP BY、HAVING、UNION或UNION ALL
  - 常量视图、JOIN、FROM一个不能更新的视图
  - WHERE子句的子查询引用了FROM子句中的表
  - 使用了临时表



# 视图基本使用

## 创建视图

- 语法格式
  - create view 视图名称 as SQL查询;
  - create view 视图名称(字段名列表) as SQL查询;

知识讲解

```
mysql> create view t11 as select * from t1;  
Query OK, 0 rows affected (0.05 sec)
```

在视图表中不定义字段名的话，默认使用基表的字段名，  
若定义字段名的话，视图表中的字段必须和基表的字段个数相等。



## 查看视图

知识讲解

- 查看当前库下所有表的状态信息
  - show table status;
  - show table status where comment="view" \G;

```
mysql> show table status where comment="view" \G;
***** 1. row *****
      Name: t11
      Engine: NULL
      Auto_increment: NULL
      .. ..
      Create_options: NULL
      Comment: VIEW
```

//视图表



## 查看视图（续1）

知识讲解

- 查看创建视图具体命令
  - show create view 视图名 ;

```
mysql> show create view t11\G;
***** 1. row *****
      View: t11
      Create View: CREATE ALGORITHM=UNDEFINED
      DEFINER=`root`@`localhost` SQL SECURITY DEFINER VIEW `t11`
      AS select `t1`.`name` AS `name` from `t1`
      character_set_client: utf8
      collation_connection: utf8_general_ci
```



## 使用视图

知识讲解

- 查询记录
  - Select 字段名列表 from 视图名 where 条件 ;
- 插入记录
  - Insert into 视图名(字段名列表) values(字段值列表) ;
- 更新记录
  - Update 视图名 set 字段名=值 where 条件 ;
- 删除记录
  - Delete from 视图名 where 条件 ;

对视图操作即是对基本操作，反之亦然！！



## 删除视图

知识讲解

- 语法格式
    - drop view 视图名 ;
- ```
mysql> drop view t11;  
Query OK, 0 rows affected (0.00 sec)  
  
mysql>
```



## 案例1：视图的基本使用

课堂练习

- 具体要求如下：
  - 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



## 视图进阶



## 创建视图的完全格式

知识讲解

- 命令格式
  - CREATE
  - [OR REPLACE]
  - [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
  - [DEFINER = { user | CURRENT\_USER }]
  - [SQL SECURITY { DEFINER | INVOKER }]
  - VIEW view\_name [(column\_list)]
  - AS select\_statement
  - [WITH [CASCADED | LOCAL] CHECK OPTION]



## 设置字段别名

知识讲解

- 视图中的字段名不可以重复 所以要定义别名
  - create view 视图名
  - as
  - select 表别名.源字段名 as 字段别名
  - from 源表名 表别名 left join 源表名 表别名
  - on 条件;

```
mysql> create view v2
as
select a.name as aname , b.name as bname , a.uid as auid , b.uid
as buid from user a left join info b on a.uid=b.uid;
```



## 重要选项说明

知识讲解

- OR REPLACE

- Create or replace view 视图名 as select 查询;
- 创建时, 若视图已存在, 会替换已有的视图

```
mysql> create view v2 as select * from t1;  
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> create view v2 as select * from t1;  
ERROR 1050 (42S01): Table 'v2' already exists //提示已存在  
mysql>
```

```
mysql> create or replace view v2 as select * from t1; //无提示  
Query OK, 0 rows affected (0.00 sec)
```

```
mysql>
```



## 重要选项说明 ( 续1 )

知识讲解

- 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 )

```
mysql> create view v1 as  
select * from a where uid < 10 with check option;  
Query OK, 0 rows affected (0.09 sec )
```

```
mysql> create view v2 as  
select * from v1 where uid >=5 with local check option;  
Query OK, 0 rows affected (0.09 sec)
```



## 案例2：视图进阶操作

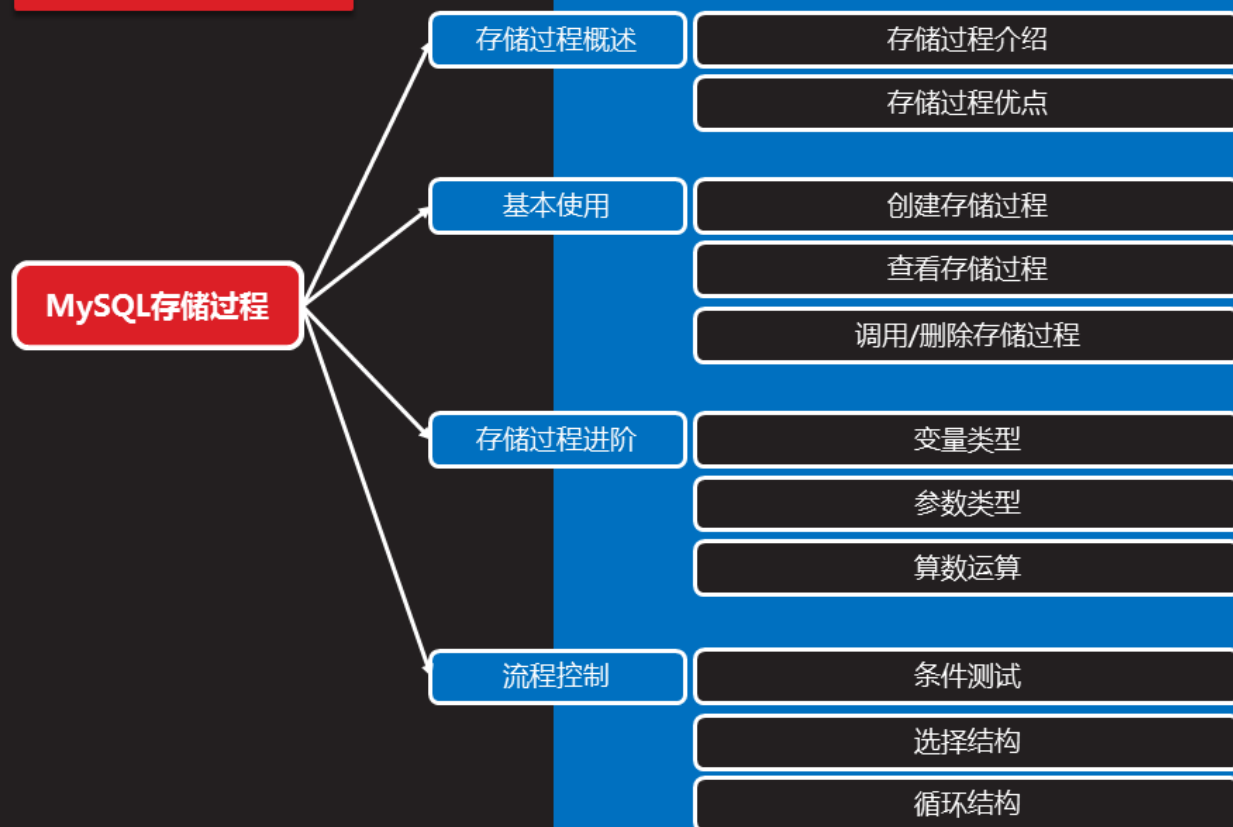
具体要求如下：

- 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

课堂练习



### MySQL存储过程



# 存储过程概述

## 存储过程介绍

- 存储过程，相当于是MySQL语句组成的脚本
  - 指的是数据库中保存的一系列SQL命令的集合
  - 可以在存储过程中使用变量、条件判断、流程控制等



# 存储过程优点

知识讲解

- 提高性能
- 可减轻网络负担
- 可以防止对表的直接访问
- 避免重复编写SQL操作



# 基本使用

## 创建存储过程

知识讲解

- 语法格式

```

- > delimiter //
create procedure 名称()
begin
... 功能代码
end
//          //结束存储过程
- Delimiter;
  
```

```

mysql> delimiter //
mysql> create procedure say()
-> begin
-> select * from studydb.user where
name="root";
-> end
-> //
Query OK, 0 rows affected (0.05 sec)
mysql> delimiter ;
  
```

- delimiter关键字用来指定存储过程的分隔符（默认为;）
- 若没有指定分割符，编译器会把存储过程当成SQL语句进行处理，从而执行出错



## 查看存储过程

知识讲解

- 方法1

```
- mysql> show procedure status;
```

- 方法2

```
- mysql> select db,name,type from mysql.proc
where name="存储过程名";
```

```

mysql> select db,name,type from mysql.proc where name="say";
+-----+-----+-----+
| db    | name | type    |
+-----+-----+-----+
| studydb | say | PROCEDURE |
+-----+-----+-----+
  
```



## 调用/删除存储过程

### 知识讲解

- 调用存储过程

- call 存储过程名();

- 存储过程没有参数时，()可以省略
- 存储过程有参数时，调用时必须传给参数

- 删除存储过程

- drop procedure 存储过程名 ;

```
mysql> call say();
```

```
+-----+-----+-----+-----+-----+-----+-----+
| name | password | uid | gid | comment | homedir | shell |
+-----+-----+-----+-----+-----+-----+-----+
| root | x        | 0   | 0   | root    | /root   | /bin/bash |
+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

```
mysql> drop procedure say;
Query OK, 0 rows affected (0.00 sec)
```



## 案例3：创建存储过程

满足以下要求：

- 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

### 课堂练习



# 存储过程进阶

## 变量类型

- 变量类型：

| 名称                           | 描述                                                                                               |
|------------------------------|--------------------------------------------------------------------------------------------------|
| 会话变量                         | 会话变量和全局变量叫系统变量 使用set命令定义<br>全局变量的修改会影响到整个服务器，但是对会话变量的修改，只会影响到当前的会话。<br><b>select @@hostname;</b> |
| 全局变量                         |                                                                                                  |
| 用户变量                         | 在客户端连接到数据库服务的整个过程中都是有效的。当当前连接断开后所有用户变量失效。<br><b>定义 set @变量名=值；</b><br><b>输出 select @变量名；</b>     |
| 局部变量                         | 存储过程中的begin/end。其有效范围仅限于该语句块中，语句块执行完毕后，变量失效。<br><b>declare</b> 专门用来定义局部变量。                       |
| 注意：局部变量 和 参数变量 调用时 变量名前不需要加@ |                                                                                                  |





## 变量类型 ( 续1 )

知识讲解

```
mysql> show global variables;           //查看全局变量
mysql> show session variables;         //查看会话变量
mysql> set session sort_buffer_size = 40000; //设置会话变量
```

```
mysql> show session variables like "sort_buffer_size";
//查看会话变量
```

```
+-----+-----+
| Variable_name | Value |
+-----+-----+
| sort_buffer_size | 40000 |
+-----+-----+
```

```
mysql> show global variables like "%关键字%"; //查看全局变量
mysql> set @y = 3; //用户自定义变量，直接赋值
mysql> select max(uid) into @y from user;
//使用sql命令查询结果赋值
```



## 变量类型 ( 续2 )

知识讲解

```
mysql> delimiter //
mysql> create procedure say48()
-> begin
-> declare x int default 9; //局部变量x
-> declare y char(10); //局部变量y
-> set y = "jim";
-> select x; select y;
-> end
-> //
Query OK, 0 rows affected (0.03 sec)
mysql> delimiter;
```

```
mysql> select @x , @y;
+-----+-----+
| @x | @y |
+-----+-----+
| NULL | NULL |
+-----+-----+
1 row in set (0.00 sec)
```

```
mysql> call say48( );
+-----+
| x |
+-----+
| 9 |
+-----+
1 row in set (0.00 sec)

+-----+
| y |
+-----+
| jim |
+-----+
1 row in set (0.00 sec)
```



## 参数类型

- 调用参数时，名称前也不需要加@
  - create procedure 名称(  
类型 参数名 数据类型，  
类型 参数名 数据类型  
)

| 关键字   | 名称      | 描述                                               |
|-------|---------|--------------------------------------------------|
| in    | 输入参数    | 作用是给存储过程传值，必须在调用存储过程时赋值，在存储过程中该参数的值不允许修改；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                                  |

知识讲解



## 参数类型（续1）

```
mysql> delimiter //
mysql> create procedure say(in username char(10))
//定义in类型的参数变量username
-> begin
-> select username;
-> select * from user where name=username;
-> end
-> //
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
```

知识讲解



## 参数类型（续2）

知识讲解

```
mysql> call say("root");           //调用存储过程时给值
+-----+
| username |
+-----+
| root     |
+-----+
1 row in set (0.00 sec)

+----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | name | sex | password | pay | gid | comment | homedir | shell |
+----+-----+-----+-----+-----+-----+-----+-----+-----+
| 01 | root | boy | x         | 0   | 0   | root     | /root   | /bin/bash |
+----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```



## 案例4：存储过程参数的使用

满足以下要求：

- 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

课堂练习



# 算数运算

## • 运算符号及用法示例

| 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           |

```
mysql> set @z=1+2;select @z;
mysql> set @x=1; set @y=2;set @z=@x*@y; select @z;
mysql> set @x=1; set @y=2;set @z=@x-@y; select @z;
mysql> set @x=1; set @y=2;set @z=@x/@y; select @z;
```

知识讲解



# 算数运算（续1）

```
mysql> drop procedure if exists say;
mysql> delimiter //
mysql> create procedure say(
in bash char(20), in nologin char(25), out x int , out y int )
begin
declare z int ;
set z=0;
select count(name) into x from db9.user where shell=bash;
select count(name) into y from db9.user where shell=nologin;
set z= x + y;
select z;
end
//
mysql> delimiter;
```

```
mysql> call say("/bin/bash","/sbin/nologin",@x,@y);
+-----+
| z      |
+-----+
| 38     |
+-----+
```

知识讲解



# 流程控制

## 条件测试

- 数值的比较

| 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 |



## 条件测试（续1）

- 逻辑比较、范围、空、非空、模糊、正则

知识讲解

| 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               |



## 选择结构

- 单分支选择结构
  - 当“条件成立”时执行命令序列
- 双分支选择架构
  - 当“条件成立”时执行代码1；否则执行代码2

知识讲解

```
if 条件测试 then
    代码 ...
    ...
end if;
```

```
if 条件测试 then
    代码1 ...
    ...
else
    代码2 ...
    ...
end if;
```



# 循环结构

- while条件式循环
  - 反复测试条件，只要成立就执行命令序列

知识讲解

```
while 条件判断 do  
    循环体  
    .. ..  
end while ;
```



# 循环结构（续1）

- loop死循环
  - 无条件、反复执行某一段代码

知识讲解

```
loop  
    循环体  
    .. ..  
end loop;
```



## 循环结构（续2）

- repeat条件式循环
  - 当条件成立时结束循环

知识讲解

```
repeat  
    循环体  
    .. ..  
until 条件判断  
end repeat;
```



## 循环结构（续3）

- 控制循环的执行
  - LEAVE 标签名 //跳出循环
  - ITERATE 标签名 //放弃本次循环，执行下一次循环

知识讲解

```
mysql>  
-> declare i int;  
-> set i=1;  
-> loab1:loop //定义标签名为loab1  
->   select i;  
->   set i=i+1;  
->   if i=3 then  #i值是3时结束本次循环  
->       iterate loab1;  
->   end if;  
->   if i=7 then  #i值是7时 结束循环  
->       leave loab1;  
->   end if;  
-> end loop;
```





## 案例5：使用循环结构

满足以下要求：

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

课堂练习



### 总结和答疑

总结和答疑

创建存储过程

问题现象

故障分析及排除

# 创建存储过程

## 问题现象

- 创建存储报错
  - 报错：

知识讲解

```
MariaDB [(none)]> delimiter //
```

```
MariaDB [(none)]> create procedure p1()  
-> begin  
-> select * from mysql.user;  
-> end  
-> //
```

```
ERROR 1046 (3D000): No database selected  
MariaDB [(none)]> _
```



# 故障分析及排除

## 知识讲解

- 原因分析
  - 没有选择库
  - 存储过程必须在库里创建
- 解决办法
  - 切换到库里，在执行创建

```
MariaDB [(none)]> use test
Database changed
MariaDB [test]> delimiter //
MariaDB [test]> create procedure p1() begin select * from mysql.user; end
-> //
Query OK, 0 rows affected (0.00 sec)

MariaDB [test]> delimiter ;
MariaDB [test]>
```

