**DBA2-day04**

# MYSQL视图

## 视图概述

### 视图介绍

— 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

**简单**

— 用户不需关心视图中的数据如何查询获得

— 视图中的数据已经时过滤号的符合条件的结果集

**安全**

— 用户只能看到视图种的数据

**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

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

操作用的数据示范：

mysql> create database db9;

Query OK, 1 row affected (0.00 sec)

mysql> create table db9.user( name char(20), password char(1), uid int(2), gid int(2), comment char(150),homedir char(150), shell char(30) );

mysql> system cp /etc/passwd /var/lib/mysql-files

mysql> load data infile "/var/lib/mysql-files/passwd" into table db9.user

-> fields terminated by ":" lines terminated by "\n";

mysql> alter table db9.user add id int(2) primary key auto\_increment first;

### 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;

在视图表中不定义字段名的时候，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

mysql> use db9;

mysql> create view v1 as select name,uid from user;

mysql> create view v2(user,userid) as select name,uid from user;

mysql> desc v1;desc v2;

mysql> select \* from v1;

mysql> grant select on db9.v1 to u1@"%" identified by "123asd...A";

### 查看视图

查看当前库下所有表的状态信息

— show table status;

— show table status where comment=”view”\G;

Comment: VIEW

查看创建视图具体命令（查看视图使用的主库）

— show create view 视图名;

mysql> show create view v1;<===>mysql> show create table v1;

### 使用视图

**查询**记录

— Select 字段名列表 from 视图名 where 条件;

mysql> select \* from v1 where uid=0;

**插入**记录

— Insert into 视图名（字段名列表） values（字段值列表）;

mysql> insert into v1 values("admin",0);

**更新**记录

— Update 视图名 set 字段名=值 where 条件;

mysql> update v1 set name="adm" where uid=0;

**删除**记录

— Delete from 视图名 where 条件;

mysql> delete from db9.user where uid=0;

对视图操作即是对基本操作，反之亦然

### 删除视图

语法格式 —— drop view 视图名;

mysql> drop view user; //user不是视图，删除会报错

mysql> drop view v1;

mysql> drop view v2;

mysql> system ls /var/lib/mysql/db9;

## 视图进阶

### 创建视图的完全格式

**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**

**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;

**//关联查询建的视图 默认不允许修改视图字段的值**

mysql> use db9;

mysql> create table t1 select name,uid,shell from user limit 3;

mysql> create table t2 select name,uid,homedir,gid from user limit 5;

给视图字段赋名时的三种格式：

mysql> create view v3(a,b,c,d,e,f,g) as select \* from t1 left join t2 on t1.name=t2.name and t1.uid=t2.uid;

mysql> create view v4 as select a.name as aname,b.name as bname,a.uid as auid,b.uid as buid from t1 a left join t2 b on a.uid=b.uid;

mysql> create view v5 as select t1.name as aname,t2.name as bname,t1.uid as auid,t2.uid as buid from t1 left join t2 on t1.uid=t2.uid;

### 重要选项说明

**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 v1 as select \* from t1;

mysql> create view v2 as select \* from t1;

mysql> create view v1 as select \* from t1; //提示已存在

mysql> create or replace view v2 as select \* from t1;

//无提示，替换已有视图

**ALGORITHM**

• 定义处理视图的方式

– ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}

• MERAGE (替换方式)

– 视图名直接使用视图的公式替换掉,把视图公式合并到了 select 中。

• TEMPTABLE (具体化方式)

– 先得到视图的执行结果,该结果形成一个中间结果暂时存在内存中,之后,外面的 select 语句就调用了这些中间结果。

• UNDEFINED (未定义)

– ALGORITHM 选项的值是 UNDEFINED 表示使用的是 MERAGE 替换方式。

**WITH CHECK OPTION**

• 当视图是根据另一个视图定义时 , 对视图更新 / 删除 / 插入

– LOCAL 和 CASCADED 关键字决定了检查的范围。

– 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 )

**with local check option**

— 对视图里的数据访问时，只检查视图创建条件，不检查基表创建条件

**with cascaded check option**

— 对视图里的数据访问时，既检查视图创建条件，又检查基表创建条件

mysql> create table user2 select name,uid,shell,homedir from user where uid>=10 and uid<=50; //从user表中筛选数据创建表user2

mysql> select \* from user2;

mysql> create view v7 as select \* from user2 where uid>=25 with local check option; //以user2表为基表，创建视图v7

mysql> select \* from v7;

mysql> update v7 set uid=7 where name="rpc";

//执行更改会报以下错误，因为不满足大于等与25的条件

ERROR 1369 (HY000): CHECK OPTION failed 'db9.v7'

mysql> update v7 set uid=30 where name="rpc";

//执行成功，满足大于等与25的条件

mysql> create view v8 as select name,uid,shell from user where uid>=20 and uid<=300;

mysql> create view v9 as select \* from v8 where uid>=40 with cascaded check option;

mysql> update v9 set uid=39 where name="gdm";

//执行失败，不满足自身条件

ERROR 1369 (HY000): CHECK OPTION failed 'db9.v9'

mysql> update v9 set uid=309 where name="gdm";

//执行失败，不满足基表条件

ERROR 1369 (HY000): CHECK OPTION failed 'db9.v9'

mysql> update v9 set uid=200 where name="gdm";

//自身与基表两者条件都满足，执行成功

mysql> select \* from v9 where name="gdm";

mysql> select \* from v8 where name="gdm";

mysql> select \* from user where name="gdm";

//视图v9,v9的基表v8，v8的基表user中数据都会修改

# 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

— 值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

**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

注：在创建存储过程的时候必须在库里创建

语法格式：

— mysql>delimiter //

create procedure 名称()

begin

........功能代码

end

// #结束存储过程

Delimiter关键子用来指定存储过程的分隔符（默认为;），若没有指定分隔符，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

方法1 ——查看已有的存储过程

— mysql>show procedure status;

方法2——指定查看存储过程的详细信息

— mysql>select db,name,type from mysql.proc where

name=”存储过程名”

### 调用/删除存储过程

**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> delimiter // #指定存储过程中的分隔符

mysql> create procedure p1()

-> begin

-> select count(name) from db9.user; 创建存储过程

-> end

-> //

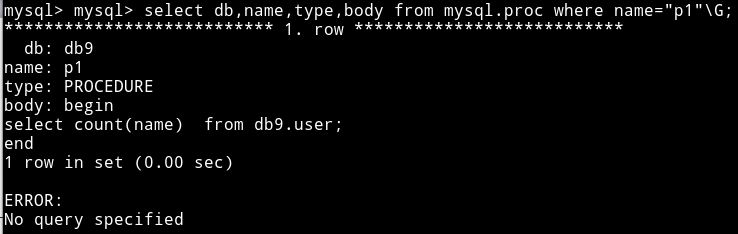
mysql> delimiter ; //恢复默认分隔符

查看存储过程代码的表结构

mysql> desc mysql.proc\G;

查看存储过程的功能代码：

mysql> select db,name,type,body from mysql.proc where name="p1";



mysql> call p1(); //调用存储过程

mysql> drop procedure p1; //删除存储过程

## 存储过程进阶

### 变量类型：

|  |  |
| --- | --- |
| 名称 | 描述 |
| 会话变量 | 绘画变量和全局变量叫系统变量 使用set命令定义 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。  **selete @@hostname;** |
| 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。当当前连接断开后多有用户变量失效。  **定义set @变量名=值;**  **输出select @变量名;** |
| 局部变量 | **存储过程中的begin/end**。其有小范围仅限于该语句块中，语句块执行完毕后，变量失效。  declare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@ | |

**查看全局变量**

mysql> show global variables;

mysql> show global variables like "%time%";

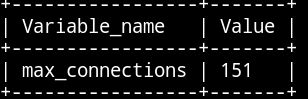
mysql> show global variables like "connect\_timeout";

**查看会话变量**

mysql> show session variables;

mysql> show session variables like "%connections%";

mysql> show session variables like "max\_connections";



mysql> set @name="jim"; **//设置会话变量**

mysql> select @name; **查看会话变量**



**把查询结果赋值给变量**

mysql> select count(name) into @x from db9.user where uid<=10;

mysql> select @x; **//查看局部变量**

mysql> delimiter //

mysql> create procedure p2()

-> begin

-> declare x int(2) default 9; //局部变量x

-> declare name char(5); //局部变量name

-> set name="jin";

-> select name;

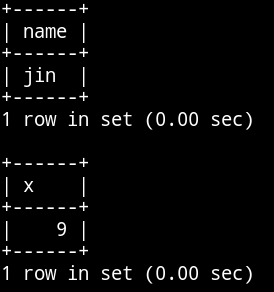
-> select x;

-> end

-> //

mysql> delimiter ;

mysql> call p2();



mysql> select @x,@name; **//输出变量值**



**//显示结果不是存储过程中定义的变量值**

用户自定义变量，直接赋值;使用sql命令查询结果赋值

### 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）

//有多个参数时以逗号分隔

|  |  |  |
| --- | --- | --- |
| 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，并且可悲改变和返回 |

++++++++++++++++++++++++++++++++++++++++++++++

**定义in类型**

mysql> delimiter //

mysql> create procedure p3(in username char(15))

//定义in类型的参数变量**username**

-> begin

-> select name from db9.user where name=username;

-> end

-> //

mysql> delimiter ;

mysql> call p3(); //错误调用

mysql> set @x="jim"; //定义用户变量

mysql> call p3(@x); //在调用存储过程的时候赋值为用户变量

mysql> call p3("adm");

++++++++++++++++++++++++++++++++++++++++++++++

**定义out类型**

mysql> delimiter //

mysql> create procedure p5( out x int(2) )

-> begin

-> select x ;

-> set x=9;

-> select x;

-> select count(name) into x from db9.user;

-> select x;

-> end

-> //

mysql> delimiter ;

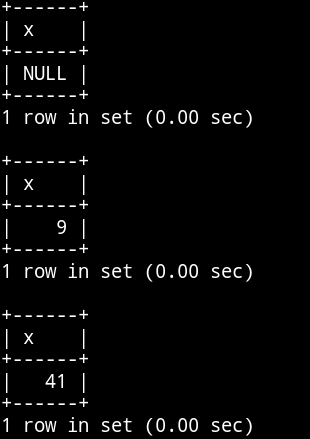
mysql> call p5(); //错误调用

mysql> call p5(1); //错误调用

mysql> set @z=100;

mysql> call p5(@z); //调用成功

#在调用的时候赋一个空值即可成功



+++++++++++++++++++++++++++++++++++++++++++++

**定义inout类型**

mysql> delimiter //

mysql> create procedure p6( inout username char(10))

-> begin

-> select username;

-> set username="jerry";

-> select username;

-> select name from db9.user where name=username;

-> select name into username from db9.user where name="bin";

-> select username;

-> end

-> //

mysql> delimiter ;

mysql> call p6("bob"); //错误调用

mysql> set @name="bob";

|  |  |
| --- | --- |
| mysql> call p6(@name);//正确调用 | mysql> call p6(@names); //正确调用 |
|  |  |

### 算术运算：

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; 3.33333333 |
| 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; //加法运算

mysql> set @x=1;set @y=2;set @z=@x \* @y;select @z; //乘法运算

mysql> set @x=1;set @y=2;set @z=@x - @y;select @z; //减法运算

mysql> set @x=1;set @y=2;set @z=@x / @y;select @z; //除法运算

## 流程控制

### 条件测试

**数值的比较**

|  |  |
| --- | --- |
| 类型 | 用途 |
| = | 等与 |
| >、>= | 大于、大于或等于 |
| <、<= | 小于、小于或等于 |
| != | 不等于 |
| BETWEEN..AND.. | 在..与..之间 |

**逻辑比较、范围、空、模糊、正则**

|  |  |
| --- | --- |
| 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 | 双分支选择结构  — 当“条件成立”时执行代码1；否则执行代码2 |
| 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; |
| mysql> delimiter //  mysql> create procedure p8(in linenum int(2))  -> begin  -> if linenum is not null then  -> select \* from db9.user where id=linenum;  -> end if;  -> end  -> //  mysql> delimiter ;  mysql> call p8(3);  mysql> call p8(@z); //赋空值时无输出 | mysql> delimiter //  mysql> create procedure p8(in linenum int(2))  -> begin  -> if linenum is not null then  -> select \* from db9.user where id=linenum;  -> else  -> select \* from db9.user where id=1;  -> end if;  -> end  -> //  mysql> delimiter ;  mysql> call p8(3);  mysql> call p8(@z);  //赋空值时有默认输出 |

### 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 | loop死循环  — 无条件、反复执行某一段代码 | repeat 条件式循环  —当条件成立时结束循环 |
| while 条件判断 do  循环体  .. ..  end while; | loop  循环体  .. ..  end loop; | repeat  循环体  .. ..  until 条件判断  end repeat; |
| mysql> delimiter // mysql>create procedure p9()  -> begin declare x int(2);  -> set x = 1;  -> while x <= 10 do  -> select x;  -> set x = x+ 1;  -> end while;  -> end  -> //  mysql> delimiter ;  mysql> call p9();  //正序显示1-10 | mysql> delimiter //  mysql>create procedure p10()  -> begin  -> loop  -> select name from db9.user where id=2;  -> end loop;  -> end  -> //  mysql> delimiter ;  mysql> call p10();  //循环显示查询结果 | mysql> delimiter //  mysql>create procedure p12()  -> begin  -> declare i int(2);  -> set i = 10;  -> repeat  -> select i;  -> set i = i -1 ;  -> until i < 1  -> end repeat;  -> end  -> //  mysql> delimiter ;  mysql> call p12;  //倒序显示10 - 1 |

### 空值循环的执行

— 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

mysql> delimiter //

mysql> create procedure p15() begin declare i int; set i=0;

-> loabl:loop set i=i+1; if i = 5 then ITERATE lobal; end if;

-> if i = 11 then LEAVE lobal; end if; select i;

-> end loop;

-> end//

mysql> delimiter ;

mysql> call p15; //遇5跳过本次循环，遇11结束整个存储过程

示例：

mysql> create procedure sumnum( in line int(2) )

-> begin

-> declare x int(2);

-> declare y int(2);

-> declare z int(2);

-> set z = 0;

-> set x = 1;

-> while x <= line do

-> select uid into y from db9.user where id=x;

-> if y % 2 = 0 then

-> set z = z + 1;

-> select name from db9.user where id=x;

-> end if;

-> set x = x + 1;

-> end while;

-> select z;

-> end

-> //

mysql> delimiter ;

mysql> call sumnum(5); //显示id<=5的uid为偶数的姓名，并统计