MySQL中表视图使用操作详解

【1】视图的定义

MySQL从5.0.1版本开始提供视图功能。一种虚拟存在的表，行和列的数据来自定义视图的查询中使用的表，并且是在使用视图时动态生成的，只保存了sql逻辑，不保存查询结果。

> 视图*(view)*，是一种有结构（有行有列）但是没数据*(*结构中不真实存放数据*)*的虚拟表，虚拟表的结构来源不是自己定义，而是从对应的基表中产生*(*视图的数据来源*)*

特征

- 创建视图后会自动从基表里面拉取数据到视图里面显示；
- 视图是一张虚拟的表；
- 视图一旦创建，系统会在视图对应的数据库文件夹下创建一个对应的结构文件–`*.frm`；
- 使用视图主要是为了查询数据；

应用场景

- 多个地方用到同样的查询结果
- 该查询结果使用的sql语句较复杂

【2】视图创建

① 基本语法

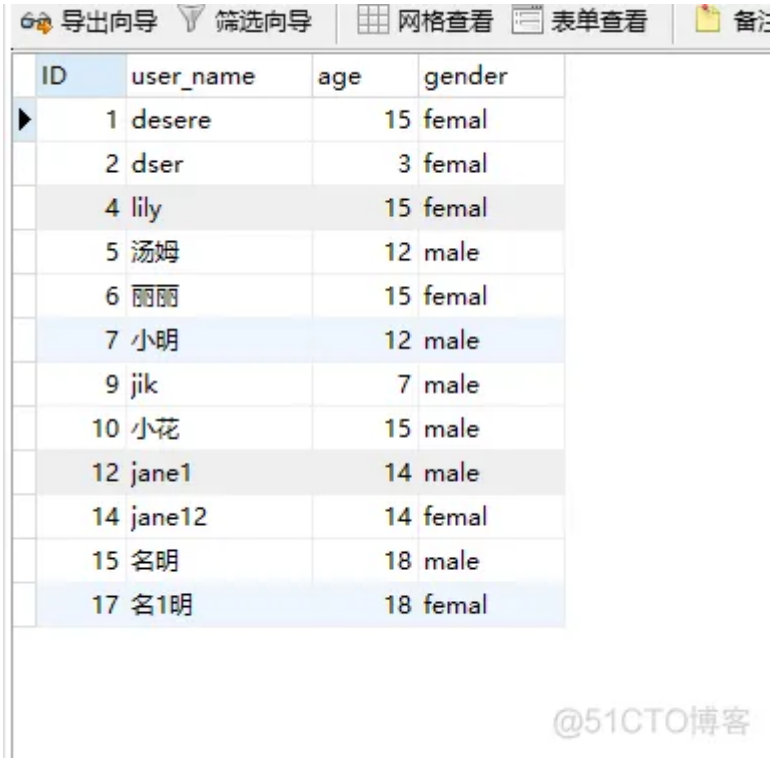
```
create view 视图名字
as
select 语句;
```

select语句可以是普通查询、连接查询、联合查询和子查询。

② 创建单表视图

```
create view v_p_user AS
SELECT p.id as ID,
 p.name AS user_name,
 p.age AS age,
 p.sex AS gender
FROM p_user p ORDER BY p.id;
```

如下图所示，并未向视图里面插入数据。数据源来自基表：



ID	user_name	age	gender
1	desere	15	femal
2	dser	3	femal
4	lily	15	femal
5	汤姆	12	male
6	丽丽	15	femal
7	小明	12	male
9	jik	7	male
10	小花	15	male
12	jane1	14	male
14	jane12	14	femal
15	名明	18	male
17	名1明	18	femal

③ 多表视图(需要注意列名不能重复)

```
create view v_p_s_user AS

SELECT p.id as id,
 p.name AS user_name,
 s.user_birthday AS birthday,
 s.user_salary AS salary
FROM p_user p,s_user s where p.id = s.user_id ORDER BY p.id;
```

等于如下形式：

```
create view v_p_s_user AS

SELECT p.id as id,
 p.name AS user_name,
 s.user_birthday AS birthday,
 s.user_salary AS salary
FROM p_user p INNER JOIN s_user s on p.id=s.user_id;
```

数据源来自两个表：

id	user_name	birthday	salary
1	desere	2017-04-27 03:11:51	5000.6
2	dser	2017-04-27 01:58:28	580
4	lily	(Null)	5000.6
5	汤姆	2017-04-27 10:12:08	5000.6
6	丽丽	2017-04-27 10:33:10	5000.6
7	小明	2017-04-27 10:46:15	5000.6
9	jik	2017-04-27 10:55:28	5000.6
10	小花	2017-04-27 10:55:45	5000.6
12	jane1	2017-04-27 11:32:26	5000.6
14	jane12	(Null)	5000.6
15	名明	2017-04-27 17:50:16	5000.6

【3】视图删除

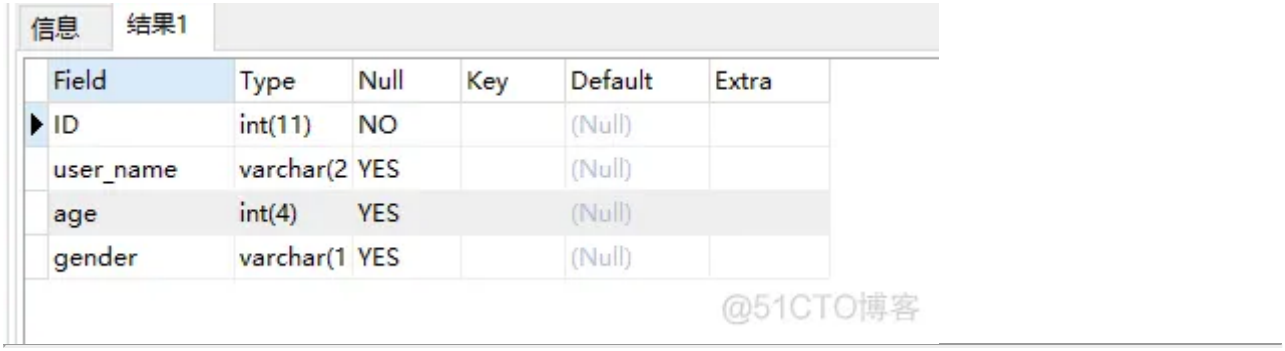
- 视图删除的是结构，视图没有数据。
- 和删除表语法格式一样：

```
drop view view_name;
```

【4】查看视图结构和创建语句

- 查看视图结构

```
desc view_name;
```



Field	Type	Null	Key	Default	Extra
ID	int(11)	NO		(Null)	
user_name	varchar(2	YES		(Null)	
age	int(4)	YES		(Null)	
gender	varchar(1	YES		(Null)	

- 查看视图创建语句

```
show create view/table view_name;
```

```
17
18  CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost`
19  SQL SECURITY DEFINER VIEW `v_p_user` AS
20  select `p`.`id` AS `ID`,`p`.`name` AS `user_name`,`p`.`age` AS `age`,`p`.`sex` AS `gender`
21  from `p_user` `p` order by `p`.`id`
```

信息 结果1

View	Create View	character_set_client	collation_connection
v_p_user	CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`l	utf8	utf8_general_ci

【5】视图修改

视图本身是不可修改的，但是视图来源可以修改。

修改视图：修改视图本身的来源语句(select 语句)。

如下所示，去掉视图v_p_user里面的age字段：

```
alter view v_p_user AS
SELECT p.id as ID,
 p.name AS user_name,
 p.sex AS gender
FROM p_user p ORDER BY p.id;
```

【6】视图的意义

① 视图可以节省sql语句：

将一条复杂的查询语句使用视图进行保存–以后可以直接对视图进行操作。

② 数据安全：

视图操作是主要针对查询的，如果对视图结构进行处理（删除），不会影响基表数据(相对安全)。

③ 使用环境：

视图往往是在大项目中使用，而且是多系统使用。可以对外提供有用的数据，但是隐藏关键(无用)的数据，进一步保障了数据安全。

④ 对外提供友好型：

不同的视图提供不同的数据，看起来对外好像是专门设计。

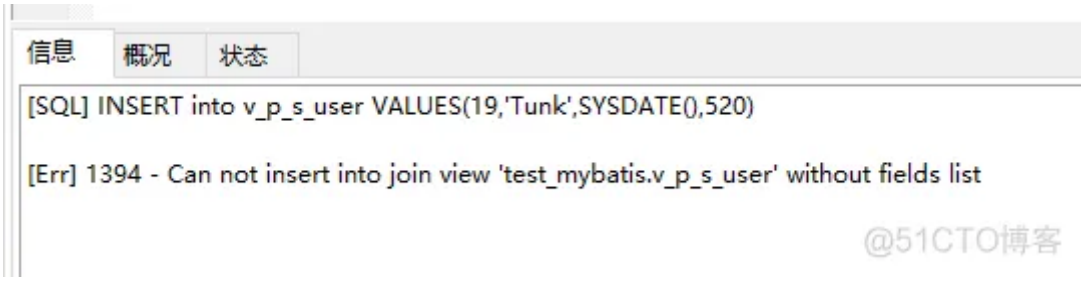
⑤ 视图可以更好(容易的)进行权限控制。

【7】视图数据操作

① 数据新增

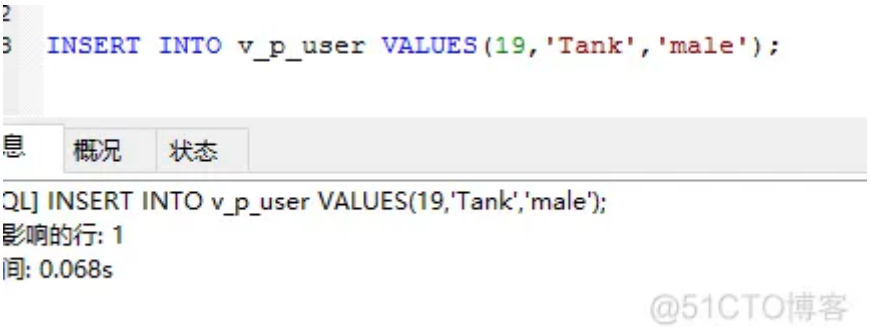
- 多表视图不能新增数据；

```
INSERT into v_p_s_user VALUES(19,'Tunk',SYSDATE(),520)
```

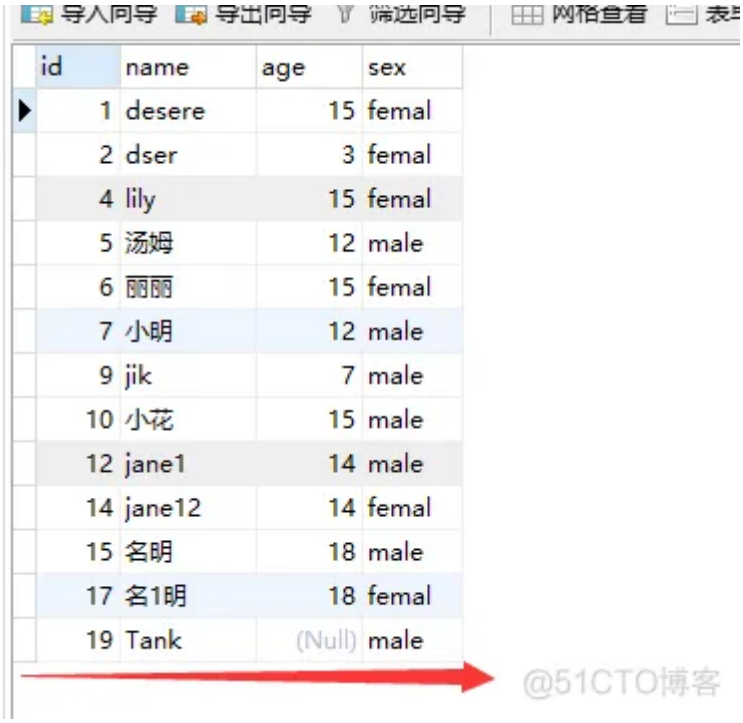


- 可以向单表视图插入数据：但是视图中包含的字段必须有基表中所有不为空(或者没有默认值)的字段。

```
INSERT INTO v_p_user VALUES(19,'Tank','male');
```



查询基表，发现基表响应添加数据，其中没有插入的列值默认为null(前提是允许为空)：



这**也说明，视图是可以向基表中插入数据的！！！插入前一定要注意，视图是否包含所有基表中不为空或者没有默认值的字段。**

② 数据更新

- 更新单表视图(成功)

- 与更新表语法一样，需要注意视图字段名，更新的数据值等；
- 更新视图主要还是更新基表，因为视图没有数据；

```
UPDATE v_p_user set user_name ='Tank2' where id = 19
```

- 更新多表视图(成功)

- 更新字段对应基表记录，需要注意视图字段名，更新的数据值等；

```
UPDATE v_p_s_user set user_name ='Tank3' where id = 2
```

视图**更新限制**

更新限制：with check option；如果在视图创建的时候对某个字段进行了限制。那么在对视图进行数据更新操作的时候，系统会进行验证----保证更新之后，数据依然可以被实体查询出来，否则不让更新。

示例如下：

- 创建视图对age字段进行限制：

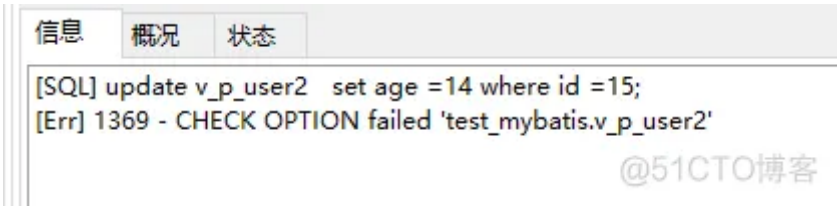
```
create view v_p_user2  AS
select * from p_user where age >15
with check option ;

-- with check option 一定要添加，否则无效。
```

其中查询数据如下：

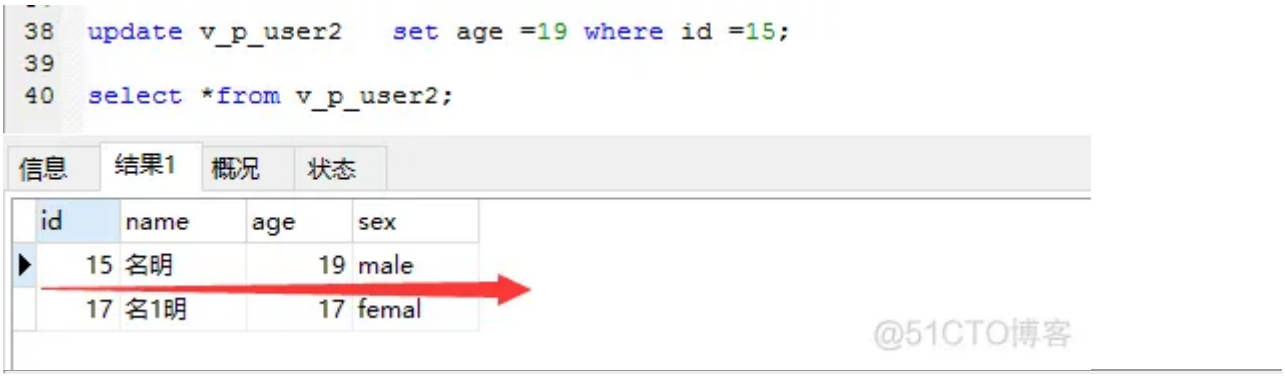
- 更改id为15的记录age为14

```
update v_p_user2   set age =14 where id =15;
```



- 更改id为15的记录age为19(很显然没问题)

```
update v_p_user2   set age =19 where id =15;
```



③ 数据删除

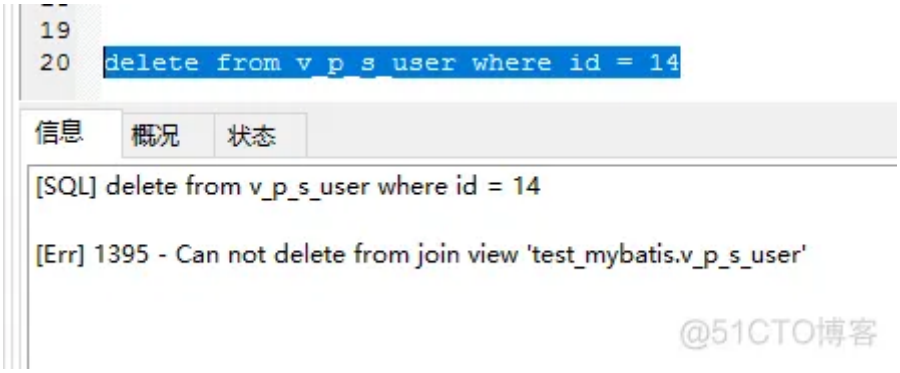
不可以删除多表视图数据，这点和数据新增一致。只能删除单表数据。

- 删除单表视图(成功)

```
delete from v_p_user where id = 19;
```

- 删除多表视图(失败)

```
delete from v_p_s_user where id = 14
```



注意：虽然可以对视图进行增删改操作，但是需要明确，视图大多大多用来查询数据，不会让其进行更新或删除操作(基于基表)。

【8】视图算法

视图算法：系统对视图以及外部查询视图的select语句的一种解析方式。

视图**算法分为三种：**

undefined：未定义(默认的)，这不是一种实际使用的算法，是一种推卸责任的算法----告诉系统，视图没有定义算法，你看着办。

temptable：临时表算法；系统应该先执行视图的select语句，后执行外部查询的语句。

merge：合并算法；系统应该先将视图对应的select语句与外部查询视图的select语句进行合并，然后执行(效率高)，系统默认值。

在创建视图的时候指定算法：

```
create algorithm = 指定算法 view view_name as select ...
```

视图算法选择：如果视图的select语句 中会包含一个查询子句，而且很有可能顺序比外部的查询语句要靠后；则选择使用temptable，其他情况可以不用指定(默认即可)。

如下所示：

- 视图创建语句：

```
create algorithm = temptable view v_p_user_3 as
select *  from p_user order by age desc;
```

- 外部查询使用语句：

```
select *  from v_p_user_3 group by age ;
```