

# 实验报告7

---

## 实验目的

## 实验内容

## 实验环境

## 课内实验

1. 创建一个行列子集视图，给出选课成绩合格的学生的编号，所选课程号和该课程成绩
2. 创建基于多个基表的视图，这个视图由学生姓名和其所选修的课程名及讲授该课程的教师姓名构成
3. 创建带表达式的视图，由学生姓名、所选课程名和所有课程成绩都比原来多5分这几个属性组成
4. 创建分组视图，将学生的学号及其平均成绩定义为一个视图
5. 创建一个基于视图的视图，基于(1)中建立的视图，定义一个包括学生编号，学生所选课程数目和平均成...
6. 查询所有选修课程Software Engineering的学生姓名
7. 插入元组(600000000,823069829,10010,59)到视图CS中。若是在视图的定义中存在WITH CHECK O...
8. 将视图CS (包含定义WITH CHECK OPTION)中，所有课程编号为10010的课程的成绩都减去5分。这个...
9. 在视图CS (包含定义WITH CHECK OPTION)删除编号为804529880学生的记录，会产生什么结果？
10. 取消视图SCT和视图CS

## 自我实践

1. 定义选课信息和课程名称的视图VIEWC
2. 定义学生姓名与选课信息的视图VIEWS
3. 定义年级低于1998的学生的视图S1(SID,SNAME,GRADE)
4. 查询学生为“uxjof”的学生的选课信息
5. 查询选修课程“UML”的学生的编号和成绩
6. 向视图S1插入记录(“600000001,Lily,2001”)
7. 定义包括更新和插入约束的视图S1，尝试向视图插入记录(“600000001,Lily,1997”)，删除所有年级为19...
8. 在视图VIEWS中将姓名为“uxjof”的学生的选课成绩都加上5分
9. 取消以上建立的所有视图

## 实验问题

视图更新限制

WITH CHECK OPTION

带表达式的视图更新问题

## 实验目的

熟悉SQL语言支持的有关视图的操作

能够熟练使用SQL语句来创建需要的视图

对视图进行查询和取消视图。

## 实验内容

1. 定义常见的视图形式，包括：  
行列子集视图。  
WITH CHECK OPTION的视图。基于多个基表的视图。  
基于视图的视图。带表达式的视图。分组视图。
2. 通过实验考察WITH CHECK OPTION这一语句在视图定义后产生的影响，包括对修改操作、删除操作、插入操作的影响。
3. 讨论视图的数据更新情况，对子行列视图进行数据更新。
4. 使用DROP语句删除一个视图，由该视图导出的其他视图定义仍在数据字典中，但已不能使用，必须显式删除。同样的原因，删除基表时，由该基表导出的所有视图定义都必须显式删除。

## 实验环境

- 数据库管理系统 PostgreSQL
- 包含所需表和数据的数据数据库实例。

在该数据库中存在4张表格，分别为：

STUDENTS(sid,sname,email,grade)

TEACHERS(tid,tname,email,salary)

COURSES(cid,cname,hour)

CHOICES(no,sid,tid,cid,score)

## 课内实验

1. 创建一个行列子集视图，给出选课成绩合格的学生的编号，所选课程号和该课程成绩

Python

```
1 create view good as
2 select sid,cid,score
3 from choices
4 where score>59
5
```

视图 (1)

good

列 (3)

sid

cid

score

规则

触发器

Python

```
1 select * from good
```

	sid character (9) 🔒	cid character (5) 🔒	score integer 🔒
1	823069829	10037	76
2	829348273	10010	87
3	847061074	10025	92
4	860635914	10039	82
5	829785562	10028	77
6	822137137	10011	67
7	826310502	10005	90
8	817636568	10047	60
9	801967882	10021	70
10	875434315	10048	82
11	830180555	10016	76
12	848035070	10007	88
13	834091581	10049	72
14	809548802	10002	64
15	833961570	10004	80
16	894256303	10018	75

2. 创建基于多个基表的视图，这个视图由学生姓名和其所选修的课程名及讲授该课 程的教师姓名构成

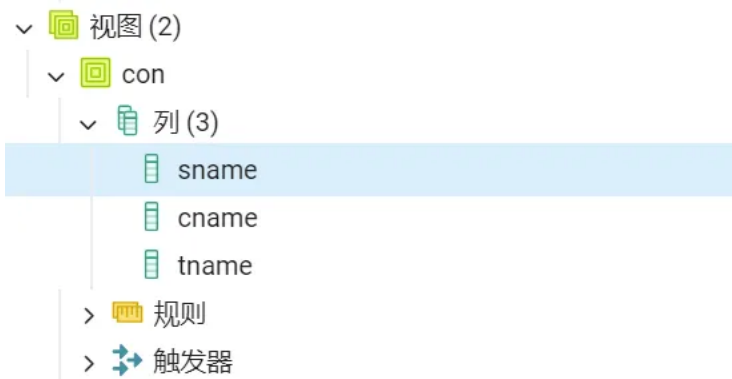


Python |

```

1  create view con as
2  select students.sname, courses.cname, teachers.tname
3  from choices
4  join courses on courses.cid=choices.cid
5  join teachers on teachers.tid=choices.tid
6  join students on choices.sid=students.sid

```



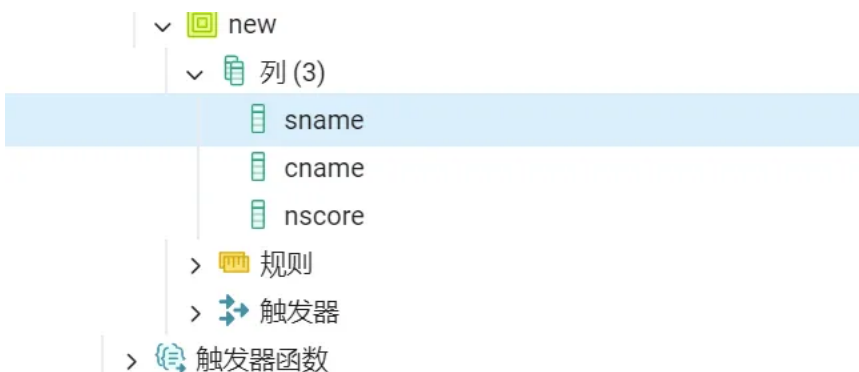
### 3. 创建带表达式的视图，由学生姓名、所选课程名和所有课程成绩都比原来多5分这几个属性组成

Python

```

1
2 create view new as
3 select students.sname,courses.cname,choices.score+5 as nscore
4 from choices
5 join courses on courses.cid=choices.cid
6 join students on choices.sid=students.sid
7

```



### 4. 创建分组视图，将学生的学号及其平均成绩定义为一个视图

```
1
2 create view score as
3 select sid,avg(COALESCE(score,0))
4 from choices
5 group by sid
```

score

- 列 (2)
  - sid
  - avg
- 规则
- 触发器

5. 创建一个基于视图的视图，基于(1)中建立的视图，定义一个包括学生编号，学生所选课程数目和平均成绩的视图

```
1 create view p5 as
2 select sid,count(*),avg(score)
3 from good
4 group by sid
```

p5

- 列 (3)
  - sid
  - count
  - avg
- 规则
- 触发器

6. 查询所有选修课程Software Engineering的学生姓名

```

1
2 select students.sname from students
3 where sid in
4 (select sid from choices where cid =
5 (select cid from courses where cname='software engineering'))

```

	sname character varying (30) 🔒
1	ucsqywg
2	axgrcq
3	affmpv
4	dertere
5	tulnfr
6	idmwbzye
7	zuehilkka
8	ffvov
9	airnnfv
10	hhhskiq
11	nphfm

总行数: 1000 / 5881 查询完成 00:00:00.151 行数 34, 列数 18

7. 插入元组(600000000,823069829,10010,59)到视图CS中。若是在视图的定义中存在WITH CHECK OPTION子句对插入操作有什么影响?

```

1 create view cs
2 as
3 select no,sid,cid,score
4 from choices
5 where score>=60 WITH CHECK OPTION

```

```

1 insert into choices(no,sid,cid,score)
2 values(600000000,823069829,10010,59)
3

```

**WITH CHECK OPTION** 确保任何通过视图插入的数据都必须满足视图定义中的 **WHERE** 子句条件。如果插入的数据不满足这些条件，数据库将拒绝插入操作，并返回一个错误。没有check时可以直接插入，有check会报错

错误： 失败，行包含(6000000000, 823069829, null, 10010, 59).null value in column "tid" of relation "choices" violates not-null constraint

错误： null value in column "tid" of relation "choices" violates not-null constraint

SQL 状态： 23502

详细： 失败，行包含(6000000000, 823069829, null, 10010, 59).

8. 将视图CS (包含定义WITH CHECK OPTION)中，所有课程编号为10010的课程的成绩都减去5分。这个操作数据库是否会正确执行，为什么?如果加上5分(原来95分以上的不变)呢?

▼ Python

```
1 update cs
2 set score=score-5
3 where cid=10010
```

不会执行，因为违反了check原则

数据输出 消息 通知

```
错误： 失败，行包含(500638678, 836678227, 276384625, 10010, 59). 新行违反了视图"cs"的检查选项

错误： 新行违反了视图"cs"的检查选项
SQL 状态： 44000
详细： 失败，行包含(500638678, 836678227, 276384625, 10010, 59).
```

可以成功执行



```
10 update cs
11 set score=score+5
12 where cid='10010'
13
```

数据输出 消息 通知

UPDATE 2590

耗时43 毫秒 成功返回查询。

## 9. 在视图CS (包含定义WITH CHECK OPTION)删除编号为804529880学生的记录，会产生什么结果？

```
14 delete from cs where sid='804529880'
15
```

数据输出 消息 通知

DELETE 0

耗时43 毫秒 成功返回查询。

视图中没有该学生，删除0条记录

## 10. 取消视图SCT和视图CS

成功删除

▼

Python |

```
1 DROP VIEW IF EXISTS SCT;
2 DROP VIEW IF EXISTS CS;
```

耗时38 毫秒 成功返回查询。

# 自我实践

## 1. 定义选课信息和课程名称的视图VIEWC

Python |

```
1 create view VIEWC as
2 select choices.cid,sid,tid,score,cname
3 from choices
4 join courses c on c.cid=choices.cid
```

Python |

```
1 select * from VIEWC
2
```

	cid character (5)	sid character (9)	tid character (9)	score integer	cname character varying (30)
1	10037	823069829	249596497	76	software testing
2	10010	829348273	202560416	87	software engineering
3	10021	850955252	234145610	54	j2me
4	10025	847061074	292043491	92	embeded system
5	10039	860635914	238811498	82	fortran
6	10028	829785562	273189968	77	architectonics
7	10011	822137137	218922066	67	distributed computing
8	10005	826310502	267846042	90	c++
9	10047	817636568	253205179	60	computer interface
10	10023	813520169	226385492	[null]	corba
11	10021	801967882	234419511	70	j2me
12	10048	875434315	223646385	82	data warehouse
13	10016	830180555	218440500	76	data mining
14	10007	848035070	208952048	88	uml

## 2. 定义学生姓名与选课信息的视图VIEWS

```

1 create view VIEWS as
2 select choices.cid,choices.sid,tid,score,sname
3 from choices
4 join students s on s.sid=choices.sid

```

```

1 select * from VIEWS

```

	cid character (5)	sid character (9)	tid character (9)	score integer	sname character varying (30)
1	10037	823069829	249596497	76	pxfys
2	10010	829348273	202560416	87	rfsleav
3	10021	850955252	234145610	54	baqzmo
4	10025	847061074	292043491	92	qxkbh
5	10039	860635914	238811498	82	xnhdjo
6	10028	829785562	273189968	77	bemgynei
7	10011	822137137	218922066	67	qaxwe
8	10005	826310502	267846042	90	cqkrjkuf
9	10047	817636568	253205179	60	wzinemrs
10	10023	813520169	226385492	[null]	rcypjhsnc
11	10021	801967882	234419511	70	kvamveu
12	10048	875434315	223646385	82	rekmgdbo
13	10016	830180555	218440500	76	mbfrhwz
14	10007	848035070	208952048	88	kxqlbun
15	10049	834091581	201353263	72	xxamhe

### 3. 定义年级低于1998的学生的视图S1(SID,SNAME,GRADE)

```

1 create view S1 as
2 select sid,sname,grade
3 from students
4 where grade>1998

```

```

1
2 select * from S1

```

	sid character (9)	sname character varying (30)	grade integer
1	800002933	vnbzq%svv	2002
2	800008565	ehlycg	1999
3	800009026	rcxaihj	2002
4	800013889	nahhluoe	2000
5	800015244	egwwnf	1999
6	800023487	tkbzqduq	1999
7	800028044	ztozk	2001
8	800029781	kkivmiw	2000
9	800031326	frroly	1999
10	800031798	oenbdg	2000
11	800032383	msqybykxi	2004
12	800033159	ocofw	2003
13	800034166	jnbluzg	2004
14	800036362	yahvv	2002
15	800039253	xholbutl	2002
16	800041569	pgmrkdhh	2001
17	800042483	affmpv	2004
总行数: 1000 / 40092    查询完成 00:00:00.101    行数 44, 列数 1			

#### 4. 查询学生为“uxjof”的学生的选课信息

Python

```

1  select * from choices
2  where sid=
3  (select sid from students where sname='uxjof')

```

+

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

SQL

	no [PK] integer	sid character (9)	tid character (9)	cid character (5)	score integer
1	506978093	800023963	220667042	10046	94
2	541221076	800023963	238341990	10018	84
3	567316431	800023963	258375444	10037	98

5. 查询选修课程“UML”的学生的编号和成绩

▼

Python

```
1  select sid,score
2  from choices
3  where cid=
4  (select cid from courses where cname='uml')
```

数据输出 消息 通知

≡

📄

▼

📋

▼

🗑

🗑

📥

⬇

📈

SQL

	sid character (9)	score integer
1	848035070	88
2	897664264	62
3	898453203	91
4	846108663	86
5	827984677	86
6	823352185	[null]
7	882778410	69
8	884993242	74
9	806427512	96
10	848803637	85
11	839666024	92
12	844745062	96
13	818493215	88
14	815954009	62
15	829224326	80
16	851336322	[null]
17	800666161	55

总行数: 1000 / 5965    查询完成 00:00:00.095    行数 64, 列数 11

6. 向视图S1插入记录(“60000001,Lily,2001”)

S1(SID,SNAME,GRADE)

▼

Python

```
1  insert into s1(sid,sname,grade)
2  values (60000001,'Lily',2001)
3
```

INSERT 0 1

耗时72 毫秒 成功返回查询。

```
4 select * from students where sname='Lily'
```

	sid [PK] character (9)	sname character varying (30)	email character varying (30)	grade integer
1	60000001	Lily	[null]	2001

7. 定义包括更新和插入约束的视图S1，尝试向视图插入记录 (“60000001,Lily,1997”), 删除所有年级为1999 的学生记录，讨论更新和插入约束带来的影响

- 1. 数据完整性， `WITH CHECK OPTION` 确保了通过视图进行的所有插入和更新操作都不会违反视图定义的条件，从而维护了数据的完整性。
- 2. 安全性，这个选项提供了额外的安全层，防止用户绕过视图定义的约束来插入或更新数据。
- 3. 它限制了可以通过视图进行的操作，只有满足视图定义条件的数据才能被插入或更新。

```
4  create view S1 as
5  select sid,sname,grade
6  from students
7  where grade>1998 with check option
```

数据输出 消息 通知

CREATE VIEW

耗时34 毫秒 成功返回查询。

由于之前的操作已经改变基础表中有该学号，由于唯一性所以需要先删除才可以添加

```
1  insert into S1(sid,sname,grade)
2  values (60000001,'Lily',1997)
```

```
9  insert into S1(sid,sname,grade)
10 values (60000001,'Lily',1997)
11
```

数据输出 消息 通知

错误： 键值"(sid)=(60000001 )" 已经存在重复键违反唯一约束"students\_pkey"

错误： 重复键违反唯一约束"students\_pkey"

SQL 状态： 23505

详细： 键值"(sid)=(60000001 )" 已经存在

先删除之前的记录

```
delete from students where sid='60000001'
```

删除后再次添加报了新的错误不符合check原则

错误： 失败，行包含(600000001 , Lily, null, 1997). 新行违反了视图"s1"的检查选项

错误： 新行违反了视图"s1"的检查选项

SQL 状态： 44000

详细： 失败，行包含(600000001 , Lily, null, 1997).

由于对视图的更改是在基础表上，所以由于外键约束不能删除

```
13 delete from s1 where grade=1999
```

数据输出 消息 通知

错误： 键值对(sid)=(800008565)仍然是从表"choices"引用的. 在 "students" 上的更新或删除操作违反了在 "choices" 上的外键约束 "fk\_choices\_students"

错误： 在 "students" 上的更新或删除操作违反了在 "choices" 上的外键约束 "fk\_choices\_students"

SQL 状态： 23503

详细： 键值对(sid)=(800008565)仍然是从表"choices"引用的.

## 8. 在视图VIEWS中将姓名为“uxjof”的学生的选课成绩都加上5分

```
1 update choices
2 set score=score+5
3 where sid=
4 (select sid from students where sname='uxjof')
```

## 9. 取消以上建立的所有视图



```
15 (SELECT viewname
16 FROM pg_views
17 WHERE schemaname = 'public')
```

数据输出 消息 通知



	viewname name
1	good
2	con
3	new
4	score
5	p5
6	viewc
7	views
8	s1

**CASCADE** 关键字用于级联更新和级联删除

数据输出 消息 通知

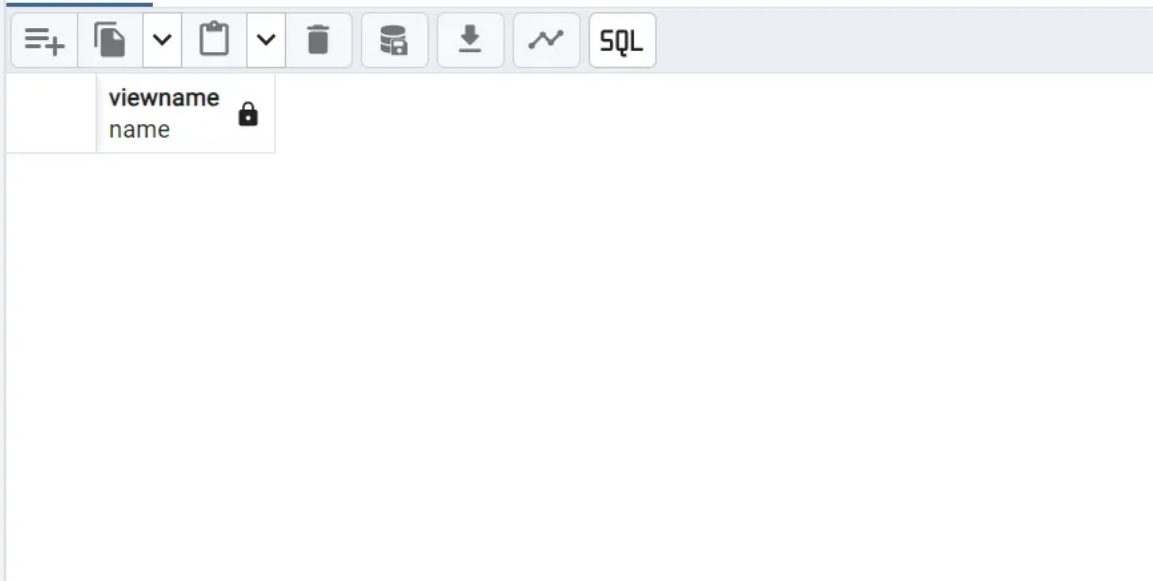
注意： 递归删除 视图 p5

DROP VIEW

耗时53 毫秒 成功返回查询。

```
15 (SELECT viewname
16 FROM pg_views
17 WHERE schemaname = 'public')
```

数据输出 消息 通知



The screenshot shows a database management tool interface. At the top, there is a toolbar with various icons: a menu icon, a file icon, a dropdown arrow, a clipboard icon, another dropdown arrow, a trash icon, a database icon, a download icon, a refresh icon, and an 'SQL' button. Below the toolbar, a table structure is displayed with two columns: 'viewname' and 'name'. A lock icon is visible next to the 'name' column header. The table is currently empty.

## 实验问题

在进行上述实验时，遇到很多问题并从中总结经验

### 视图更新限制

**问题:** 在尝试更新视图时，发现某些视图不允许更新操作。

理解视图的可更新性规则

只有当视图满足特定条件（如只涉及一个表，不包含聚合函数、GROUP BY子句等）时，视图才是可更新的。

### WITH CHECK OPTION

**问题:** 使用 `WITH CHECK OPTION` 时，可能会遇到插入或更新数据被拒绝的情况。这个选项确保了视图的完整性，防止了任何不符合视图定义条件的数据修改。

### 带表达式的视图更新问题

**问题:** 尝试更新包含表达式的视图中的数据时，可能会发现无法直接更新计算字段。

包含计算字段的视图通常不允许直接更新计算字段。

需要更新基表中的基础数据，以间接更新视图中的表达式结果。

### 删除视图的依赖性问题

**问题:** 删除视图后，发现依赖于该视图的其他视图或应用程序无法正常工作。

在删除视图之前，需要评估其对其他数据库对象的影响。

可能需要更新或删除依赖于该视图的其他视图，以保持数据库的一致性和完整性。