



数据库系统课程实验报告

实验名称:	实验六：数据库的安全性
实验日期:	2023/5/5
实验地点:	文宣楼 A402
提交日期:	2023/5/5
学号:	22920212204392
姓名:	黄勛
专业年级:	软工 2021 级
学年学期:	2022-2023 学年第二学期

1.实验目的

- 理解数据库系统用户(user)、权限(privilege)和角色(role)的概念和作用
- 熟练掌握用户的管理：创建、查看、删除和权限的授予与回收
- 熟练掌握通过数据字典查看用户权限、表和视图权限的方法
- 熟练掌握使用 Grant 命令给用户、角色授权的方法
- 熟练掌握使用 Revoke 命令回收已授权限的方法
- 熟练掌握角色定义、重命名和删除的方法
- 熟练掌握修改角色中权限的方法
- 理解视图的安全性作用库

2.实验内容和步骤

(0) 登录 ECS 服务器，以 omm 操作系统管理员身份登录数据库，使用 gsql 连接到数据库。

```
su - omm
```

```
gs_om -t start
```

```
gsql -d postgres -p 26000 -r
```

```
(gsql -d sales -p 26000 -U hx -W HX@123pass -r)
```

(1) 学习并完成 <https://bokai.blog.csdn.net/article/details/117912175> 的内容。

(附加说明) 一、用户及角色

(1)用户

创建、修改、删除用户：

创建用户 jim，登录密码为 Bigdata@123。

CREATE USER jim PASSWORD 'Bigdata@123';

```
postgres=# CREATE USER jim PASSWORD 'Bigdata@123';
CREATE ROLE
postgres=#
```

查看用户列表

SELECT * FROM pg_user;

```
postgres=# SELECT * FROM pg_user;
 username | usesysid | usecreatedb | usesuper | usecatupd | userepl | passwd | valbegin | valuntil | respool | parent | s
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
 oradmin  | usepolicyadmin
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
 omm      | 10      | t          | t        | t        | t        | ***** |          |          | default_pool | 0
 maoli    | 16384   | f          | f        | f        | f        | ***** |          |          | default_pool | 0
 hx       | 16521   | f          | f        | f        | f        | ***** |          |          | default_pool | 0
 jim      | 16797   | f          | f        | f        | f        | ***** |          |          | default_pool | 0
```

为用户 jim 追加有创建角色的 CREATE ROLE 权限

ALTER USER jim CREATEROLE;

```
postgres=# ALTER USER jim CREATEROLE;
ALTER ROLE
postgres=#
```

删除用户

DROP USER jim CASCADE;

```
postgres=# DROP USER jim CASCADE;
DROP ROLE
```

(2)角色

创建、修改、删除角色：

创建一个角色，名为 manager，密码为 Bigdata@123

CREATE ROLE manager IDENTIFIED BY 'Bigdata@123';

```
postgres=# CREATE ROLE manager IDENTIFIED BY 'Bigdata@123';
CREATE ROLE
```

查看角色

SELECT * FROM PG_ROLES;

```
postgres=# SELECT * FROM PG_ROLES;
rolname | rolsuper | rolinherit | rolcreaterole | rolcreatedb | rolcatupdate | rolcanlogin | rolreplication | rolauditadmin | rolsystemadmin | rolconnlimit | rolpassword |
entid | roltablespace | rolconfig | oid | roluseft | rolkind | nodegroup | roltemp space | rolspill space | rolmonitoradmin | roloperatoradmin | rolpolicyadmin |
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
omm | t | t | t | t | t | t | t | t | t | t | t |
0 | f | f | f | f | f | f | f | f | f | f | f |
maoli | f | t | f | f | f | f | f | f | f | f | f |
0 | f | f | f | f | f | f | f | f | f | f | f |
hx | f | t | f | f | f | f | f | f | f | f | f |
0 | f | f | f | f | f | f | f | f | f | f | f |
manager | f | t | f | f | f | f | f | f | f | f | f |
0 | f | f | f | f | f | f | f | f | f | f | f |
(4 rows)
```

修改角色 manager 的密码为 abcd@123。

ALTER ROLE manager IDENTIFIED BY 'abcd@123' REPLACE

'Bigdata@123';

```
postgres=# ALTER ROLE manager IDENTIFIED BY 'abcd@123' REPLACE 'Bigdata@123';
ALTER ROLE
postgres=#
```

修改角色 manager 为系统管理员

ALTER ROLE manager SYSADMIN;

```
postgres=# ALTER ROLE manager SYSADMIN;
ALTER ROLE
postgres=#
```

删除角色 manager

DROP ROLE manager;

```
postgres=# DROP ROLE manager;
DROP ROLE
postgres=#
```

二、权限设置及回收

1.将系统权限授权给用户或者角色

创建名为 joe 的用户:

CREATE USER joe PASSWORD 'Bigdata@123';

```
postgres=# CREATE USER joe PASSWORD 'Bigdata@123';
CREATE ROLE
postgres=#
```

将 sysadmin 权限授权给 joe:

GRANT ALL PRIVILEGES TO joe;

```
postgres=# GRANT ALL PRIVILEGES TO joe;
ALTER ROLE
```

撤销 joe 用户的 sysadmin 权限

REVOKE ALL PRIVILEGES FROM joe;

```
postgres=# REVOKE ALL PRIVILEGES FROM joe;
ALTER ROLE
```

2. 将数据库对象授权给角色或用户

创建 tpcds 模式

CREATE SCHEMA tpcds;

```
postgres=# CREATE SCHEMA tpcds;
CREATE SCHEMA
```

tpcds 模式下创建一张 reason 表

```
CREATE TABLE tpcds.reason
(
    r_reason_sk      INTEGER      NOT NULL,
    r_reason_id      CHAR(16)     NOT NULL,
    r_reason_desc    VARCHAR(20)
);
```

```
postgres=# CREATE TABLE tpcds.reason
postgres=# (
postgres(#      r_reason_sk      INTEGER      NOT NULL,
postgres(#      r_reason_id      CHAR(16)     NOT NULL,
postgres(#      r_reason_desc    VARCHAR(20)
postgres(# );
CREATE TABLE
```

将模式 tpcds 的使用权限和表 tpcds.reason 的所有权限授权给用户 joe

GRANT USAGE ON SCHEMA tpcds TO joe;

GRANT ALL PRIVILEGES ON tpcds.reason TO joe;

```
postgres=# GRANT USAGE ON SCHEMA tpcds TO joe;
GRANT
postgres=# GRANT ALL PRIVILEGES ON tpcds.reason TO joe;
GRANT
postgres=#
```

将 tpods.reason 表中 r_reason_sk、r_reason_id、r_reason_desc 列的查询权限，r_reason_desc 的更新权限授权给 joe

```
GRANT select (r_reason_sk,r_reason_id,r_reason_desc),update (r_reason_desc) ON  
tpods.reason TO joe;
```

```
postgres=# GRANT select (r_reason_sk,r_reason_id,r_reason_desc),update (r_reason_desc) ON tpods.reason TO joe;  
GRANT  
postgres=#
```

将数据库 postgres 的连接权限授权给用户 joe，并给予其在 postgres 中创建 schema 的权限，而且允许 joe 将此权限授权给其他用户

```
GRANT create,connect on database postgres TO joe WITH GRANT OPTION;
```

```
postgres=# GRANT create,connect on database postgres TO joe WITH GRANT OPTION;  
GRANT
```

创建角色 tpods_manager

```
CREATE ROLE tpods_manager PASSWORD 'Bigdata@123';
```

```
postgres=# CREATE ROLE tpods_manager PASSWORD 'Bigdata@123';  
CREATE ROLE  
postgres=#
```

将模式 tpods 的访问权限授权给角色 tpods_manager，并授予该角色在 tpods 下创建对象的权限，不允许该角色中的用户将权限授权给其它人

```
GRANT USAGE,CREATE ON SCHEMA tpods TO tpods_manager;
```

```
postgres=# GRANT USAGE,CREATE ON SCHEMA tpods TO tpods_manager;  
GRANT  
postgres=#
```

查看表 reason 权限：

```
SELECT * FROM information_schema.table_privileges WHERE table_name='reason';
```

```
postgres=# SELECT * FROM information_schema.table_privileges WHERE table_name='reason';
```

grantor	grantee	table_catalog	table_schema	table_name	privilege_type	is_grantable	with_hierarchy
omm	omm	postgres	tpcds	reason	INSERT	YES	NO
omm	omm	postgres	tpcds	reason	SELECT	YES	YES
omm	omm	postgres	tpcds	reason	UPDATE	YES	NO
omm	omm	postgres	tpcds	reason	DELETE	YES	NO
omm	omm	postgres	tpcds	reason	TRUNCATE	YES	NO
omm	omm	postgres	tpcds	reason	REFERENCES	YES	NO
omm	omm	postgres	tpcds	reason	TRIGGER	YES	NO
omm	joe	postgres	tpcds	reason	INSERT	NO	NO
omm	joe	postgres	tpcds	reason	SELECT	NO	YES
omm	joe	postgres	tpcds	reason	UPDATE	NO	NO
omm	joe	postgres	tpcds	reason	DELETE	NO	NO
omm	joe	postgres	tpcds	reason	TRUNCATE	NO	NO
omm	joe	postgres	tpcds	reason	REFERENCES	NO	NO
omm	joe	postgres	tpcds	reason	TRIGGER	NO	NO
omm	joe	postgres	tpcds	reason	ALTER	NO	NO
omm	joe	postgres	tpcds	reason	DROP	NO	NO
omm	joe	postgres	tpcds	reason	COMMENT	NO	NO
omm	joe	postgres	tpcds	reason	INDEX	NO	NO
omm	joe	postgres	tpcds	reason	VACUUM	NO	NO

(19 rows)

3.将用户或者角色的权限授权给其他用户或角色

创建角色 manager

```
CREATE ROLE manager PASSWORD 'Bigdata@123';
```

```
postgres=# CREATE ROLE manager PASSWORD 'Bigdata@123';
CREATE ROLE
postgres=#
```

将 joe 的权限授权给 manager，并允许该角色将权限授权给其他人

```
GRANT joe TO manager WITH ADMIN OPTION;
```

```
postgres=# GRANT joe TO manager WITH ADMIN OPTION;
GRANT ROLE
```

创建用户 senior_manager

```
CREATE ROLE senior_manager PASSWORD 'Bigdata@123';
```

```
postgres=# CREATE ROLE senior_manager PASSWORD 'Bigdata@123';
CREATE ROLE
```

将用户 manager 的权限授权给该用户

```
GRANT manager TO senior_manager;
```

```
postgres=# GRANT manager TO senior_manager;
GRANT ROLE
postgres=#
```

4.权限回收并清理用户

逐步回收 manager 权限

```
REVOKE joe FROM manager;
```

```
REVOKE manager FROM senior_manager;
```

删除 manager 用户

```
DROP USER manager;
```

逐步回收 joe 权限

```
REVOKE ALL PRIVILEGES ON tpcds.reason FROM joe;
```

```
REVOKE ALL PRIVILEGES ON SCHEMA tpcds FROM joe;
```

逐步回收 tpcds_manager 权限

```
REVOKE USAGE,CREATE ON SCHEMA tpcds FROM tpcds_manager;
```

删除 tpcds_manager 用户

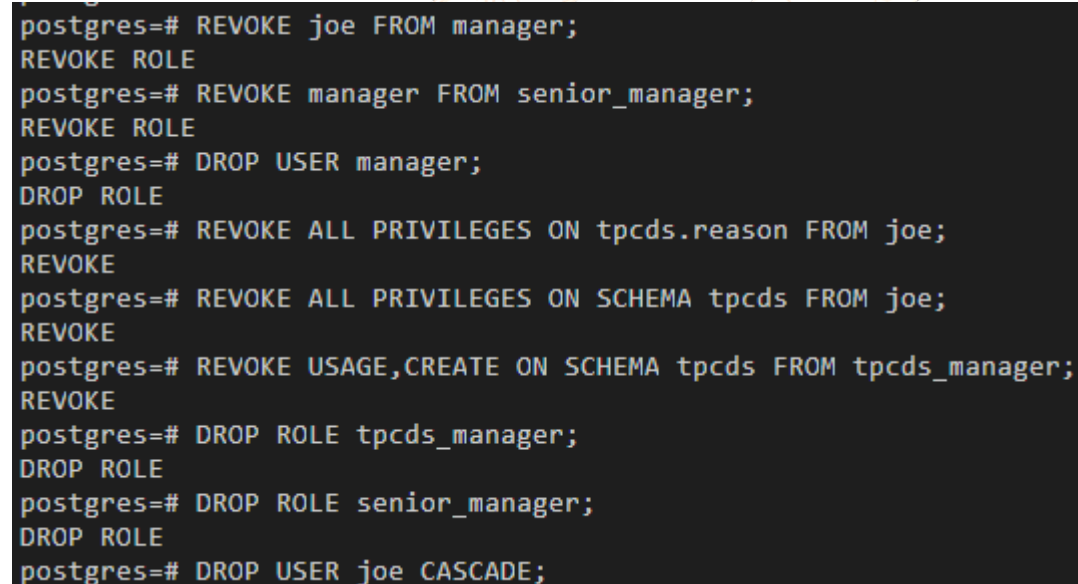
```
DROP ROLE tpcds_manager
```

删除 senior_manager 用户

```
DROP ROLE senior_manager;
```

删除 joe 用户

```
DROP USER joe CASCADE;
```



```
postgres=# REVOKE joe FROM manager;
REVOKE ROLE
postgres=# REVOKE manager FROM senior_manager;
REVOKE ROLE
postgres=# DROP USER manager;
DROP ROLE
postgres=# REVOKE ALL PRIVILEGES ON tpcds.reason FROM joe;
REVOKE
postgres=# REVOKE ALL PRIVILEGES ON SCHEMA tpcds FROM joe;
REVOKE
postgres=# REVOKE USAGE,CREATE ON SCHEMA tpcds FROM tpcds_manager;
REVOKE
postgres=# DROP ROLE tpcds_manager;
DROP ROLE
postgres=# DROP ROLE senior_manager;
DROP ROLE
postgres=# DROP USER joe CASCADE;
```

(2) 创建视图 salesman, 该视图只保存 employees 表中所有

job_title 为'Sales Representative'的雇员。

```
create view salesman as select * from employees where
```

```
job_title='SalesRepresentative';
```

(注：由于之前处理数据将空格去掉了，故这里 title 没有空格)


```

sales=> set search_path to sales;
SET
sales=> create view salesman as select * from employees where job_title='Sales Representative';
CREATE VIEW
sales=> █

```

(3) 创建基于 salesman 的视图

salesman_contacts(first_name,last_name,email,phone), 该视图存储的
salesman 的联系方式。

create view salesman_contacts as select first_name, last_name,email,phone
from salesman;

```

sales=> create view salesman_contacts as select first_name, last_name,email,phone from salesman;
CREATE VIEW

```

(4) 查询视图 salesman 和 salesman_contacts。

select * from salesman;

employee_id	first_name	last_name	email	phone	hire_date	manager_id	job_title
56	Evie	Harrison	evieharrison@examplecom	011441344486508	2016-11-23 00:00:00	46	SalesRepresentative
57	Scarlett	Gibson	scarlettgibson@examplecom	011441345429268	2016-09-30 00:00:00	47	SalesRepresentative
58	Ruby	Mcdonald	rubymcdonald@examplecom	011441345929268	2016-03-04 00:00:00	47	SalesRepresentative
59	Chloe	Cruz	chloecruz@examplecom	011441345829268	2016-09-01 00:00:00	47	SalesRepresentative
60	Isabelle	Marshall	isabellemarshall@examplecom	011441345729268	2016-03-10 00:00:00	47	SalesRepresentative
61	Daisy	Ortiz	daisyortiz@examplecom	011441345629268	2016-12-15 00:00:00	47	SalesRepresentative
62	Freya	Gomez	freitagomez@examplecom	011441345529268	2016-11-03 00:00:00	47	SalesRepresentative
88	Elizabeth	Dixon	elizabethdixon@examplecom	011441644429262	2016-09-04 00:00:00	50	SalesRepresentative
64	Florence	Freeman	florencefreeman@examplecom	011441346229268	2016-03-19 00:00:00	48	SalesRepresentative
65	Alice	Wells	alicewells@examplecom	011441346329268	2016-09-24 00:00:00	48	SalesRepresentative
66	Charlotte	Webb	charlottewebb@examplecom	011441346529268	2016-02-23 00:00:00	48	SalesRepresentative
67	Sienna	Simpson	siennasimpson@examplecom	011441346629268	2016-03-24 00:00:00	48	SalesRepresentative
68	Matilda	Stevens	matildastevens@examplecom	011441346729268	2016-02-21 00:00:00	48	SalesRepresentative
69	Evelyn	Tucker	evelyntucker@examplecom	011441343929268	2016-03-11 00:00:00	49	SalesRepresentative
70	Eva	Porter	evaporter@examplecom	011441343829268	2016-03-23 00:00:00	49	SalesRepresentative
71	Millie	Hunter	milliehunter@examplecom	011441343729268	2016-09-24 00:00:00	49	SalesRepresentative
72	Sofia	Hicks	sofiahicks@examplecom	011441343629268	2016-02-23 00:00:00	49	SalesRepresentative
73	Lucy	Crawford	lucycrawford@examplecom	011441343529268	2016-03-24 00:00:00	49	SalesRepresentative
74	Elsie	Henry	elsiehenry@examplecom	011441343329268	2016-02-21 00:00:00	49	SalesRepresentative
75	Imogen	Boyd	imogenboyd@examplecom	011441644429267	2016-05-11 00:00:00	50	SalesRepresentative
76	Layla	Mason	laylamason@examplecom	011441644429266	2016-03-19 00:00:00	50	SalesRepresentative
77	Rosie	Morales	rosiemorales@examplecom	011441644429265	2016-03-24 00:00:00	50	SalesRepresentative
78	Maya	Kennedy	mayakennedy@examplecom	011441644429264	2016-02-23 00:00:00	50	SalesRepresentative
79	Esme	Warren	esmewarren@examplecom	011441644429263	2016-05-24 00:00:00	50	SalesRepresentative
55	Grace	Ellis	graceellis@examplecom	011441344987668	2016-12-09 00:00:00	46	SalesRepresentative
54	Lily	Fisher	lilyfisher@examplecom	011441344498718	2016-03-30 00:00:00	46	SalesRepresentative
53	Sophia	Reynolds	sophiareynolds@examplecom	011441344478968	2016-09-20 00:00:00	46	SalesRepresentative
52	Sophie	Owens	sophieowens@examplecom	011441344345268	2016-03-24 00:00:00	46	SalesRepresentative
51	Poppy	Jordan	poppyjordan@examplecom	011441344129268	2016-09-30 00:00:00	46	SalesRepresentative
63	Phoebe	Murray	phoebemurray@examplecom	011441346129268	2016-11-11 00:00:00	48	SalesRepresentative

select * from salesman_contacts;

```

sales=> select * from salesman_contacts;
first_name | last_name | email | phone
-----+-----+-----+-----
Evie | Harrison | evieharrison@examplecom | 011441344486508
Scarlett | Gibson | scarlettgibson@examplecom | 011441345429268
Ruby | Mcdonald | rubymcdonald@examplecom | 011441345929268
Chloe | Cruz | chloecruz@examplecom | 011441345829268
Isabelle | Marshall | isabellemarshall@examplecom | 011441345729268
Daisy | Ortiz | daisyortiz@examplecom | 011441345629268
Freya | Gomez | freyagomez@examplecom | 011441345529268
Elizabeth | Dixon | elizabethdixon@examplecom | 011441644429262
Florence | Freeman | florencefreeman@examplecom | 011441346229268
Alice | Wells | alicewells@examplecom | 011441346329268
Charlotte | Webb | charlottewebb@examplecom | 011441346529268
Sienna | Simpson | siennasimpson@examplecom | 011441346629268
Matilda | Stevens | matildastevens@examplecom | 011441346729268
Evelyn | Tucker | evelyntucker@examplecom | 011441343929268
Eva | Porter | evaporter@examplecom | 011441343829268
Millie | Hunter | milliehunter@examplecom | 011441343729268
Sofia | Hicks | sofiahicks@examplecom | 011441343629268
Lucy | Crawford | lucycrawford@examplecom | 011441343529268
Elsie | Henry | elsiehenry@examplecom | 011441343329268
Imogen | Boyd | imogenboyd@examplecom | 011441644429267
Layla | Mason | laylamason@examplecom | 011441644429266
Rosie | Morales | rosiemorales@examplecom | 011441644429265
Maya | Kennedy | mayakennedy@examplecom | 011441644429264
Esme | Warren | esmewarren@examplecom | 011441644429263
Grace | Ellis | graceellis@examplecom | 011441344987668
Lily | Fisher | lilyfisher@examplecom | 011441344498718
Sophia | Reynolds | sophiareynolds@examplecom | 011441344478968
Sophie | Owens | sophieowens@examplecom | 011441344345268
Poppy | Jordan | poppyjordan@examplecom | 011441344129268
Phoebe | Murray | phoebemurray@examplecom | 011441346129268
(30 rows)

```

(5) 在当前窗口输入命令：`\c - omm` 切换到 omm 用户。

```

sales=> \c - omm
Non-SSL connection (SSL connection is recommended when requiring high-security)
You are now connected to database "sales" as user "omm".

```

(6) 创建新用户 user1。

create user user1 identified by 'user1@123';

```

sales=# create user user1 identified by 'user1@123';
CREATE ROLE

```

(7) 在当前窗口输入命令：`\c - user1` 切换到 user1 用户。

```
sales=# \c - user1
Password for user user1:
Non-SSL connection (SSL connection is recommended when requiring high-security)
You are now connected to database "sales" as user "user1".
```

(8) 发布查询命令: **select * from salesman_contacts**;观察结果。

```
sales=> select * from salesman_contacts;
ERROR:  relation "salesman_contacts" does not exist on dn_6001
LINE 1: select * from salesman_contacts;
                        ^
```

(9) 发布命令: **\c - hx** 切换到 hx 用户。

```
sales=> \c - hx
Password for user hx:
Non-SSL connection (SSL connection is recommended when requiring high-security)
You are now connected to database "sales" as user "hx".
sales=>
```

(10) 在当前 hx 用户下输入命令: **grant select on salesman_contacts**
to user1; 实现授权操作。

grant usage on schema sales to user1;

grant select on **salesman_contacts** to user1;

```
sales=> grant usage on schema sales to user1;
GRANT
sales=> grant select on salesman_contacts to user1;
ERROR:  relation "salesman_contacts" does not exist
sales=> set search_path to sales;
SET
sales=> grant select on salesman_contacts to user1;
GRANT
```

(11) 依次重复步骤 (7) 和 (8), 比较两次查询的结果。

```
sales=> \c - user1
Password for user user1:
Non-SSL connection (SSL connection is recommended when requiring high-security)
You are now connected to database "sales" as user "user1".
sales=>
```

```

sales=> select * from salesman_contacts;
first_name | last_name | email | phone
-----+-----+-----+-----
Evie | Harrison | evieharrison@examplecom | 011441344486508
Scarlett | Gibson | scarlettgibson@examplecom | 011441345429268
Ruby | Mcdonald | rubymcdonald@examplecom | 011441345929268
Chloe | Cruz | chloecruz@examplecom | 011441345829268
Isabelle | Marshall | isabellemarshall@examplecom | 011441345729268
Daisy | Ortiz | daisyortiz@examplecom | 011441345629268
Freya | Gomez | freyagomez@examplecom | 011441345529268
Elizabeth | Dixon | elizabethdixon@examplecom | 011441644429262
Florence | Freeman | florencefreeman@examplecom | 011441346229268
Alice | Wells | alicewells@examplecom | 011441346329268
Charlotte | Webb | charlottewebb@examplecom | 011441346529268
Sienna | Simpson | siennasimpson@examplecom | 011441346629268
Matilda | Stevens | matildastevens@examplecom | 011441346729268
Evelyn | Tucker | evelyntucker@examplecom | 011441343929268
Eva | Porter | evaporter@examplecom | 011441343829268
Millie | Hunter | milliehunter@examplecom | 011441343729268
Sofia | Hicks | sofiahicks@examplecom | 011441343629268
Lucy | Crawford | lucycrawford@examplecom | 011441343529268
Elsie | Henry | elsiehenry@examplecom | 011441343329268
Imogen | Boyd | imogenboyd@examplecom | 011441644429267
Layla | Mason | laylamason@examplecom | 011441644429266
Rosie | Morales | rosiemorales@examplecom | 011441644429265
Maya | Kennedy | mayakennedy@examplecom | 011441644429264
Esme | Warren | esmewarren@examplecom | 011441644429263
Grace | Ellis | graceellis@examplecom | 011441344987668
Lily | Fisher | lilyfisher@examplecom | 011441344498718
Sophia | Reynolds | sophiareynolds@examplecom | 011441344478968
Sophie | Owens | sophieowens@examplecom | 011441344345268
Poppy | Jordan | poppyjordan@examplecom | 011441344129268
Phoebe | Murray | phoebemurray@examplecom | 011441346129268
(30 rows)

```

/*说明：步骤（2）-（11）的主要目的是用于验证视图的作用：被授权用户只能查询在权限范围内的数据，范围外的数据不可访问*/

（12）查看与角色、权限相关的系统表和系统视图：pg_roles, pg_authid。

```

select * from pg_roles;

select * from pg_authid;

```

```
postgres=# grant insert on table SC to u5 with grant option;  
GRANT
```

[例 4.6]

```
postgres=# grant insert on table SC to u6 with grant option;  
GRANT
```

[例 4.7]

```
postgres=# grant insert on table SC to u7;  
GRANT
```

[例 4.8]把用户 U4 修改学生学号的权限收回

```
postgres=# revoke update(Sno) on table Student from u4;  
REVOKE
```

[例 4.9]收回所有用户对表 SC 的查询权限。

```
postgres=# revoke select on table SC from public;  
REVOKE
```

[例 4.10]把用户 U5 对 SC 表的 INSERT 权限收回。

```
postgres=# revoke insert on table SC from u5 cascade;  
REVOKE
```

[例 4.11]

```
postgres=# create role R1 identified by 'R1@12345';  
CREATE ROLE  
postgres=# grant select,update,insert on table Student to R1;  
GRANT
```

```
postgres=# grant R1 to u1,u2,u3;  
GRANT ROLE  
postgres=# revoke R1 from u1;  
REVOKE ROLE
```

[例 4.12]

```
postgres=# grant delete on table Student to R1;  
GRANT
```

[例 4.13]

```
postgres=# revoke select on table Student from R1;  
REVOKE
```

3. 实验总结

3.1 完成的工作

设计正确的 SQL 语句并完成了所有数据安全性操作修改等操作。

3.2 对实验的认识

(1) 具有什么权限才能创建新用户?

具有管理权限的用户才能创建新用户，因为只有管理员才能对系统的用户管理功能进行控制和设置。

(2) 角色的作用是什么?

角色的作用是为用户分配权限。具体来说，通过为用户分配角色，系统可以更轻松地控制每个用户可以执行哪些任务，而无需逐个设置每个用户的权限（会很繁琐），同时在撤销权限的时候也可以更方便的撤销。

(3) 如何实现角色所含权限的修改，请设计样例验证之。

具体过程见实验项(13)，实验项重做了教材中的[例 4.1-例 4.13]，其中包含了示例。

(4) 收获

这次实验我掌握了授权，撤回权限，查看权限的基本语法，理解了数据库的安全性是如何保障的，我认为通过用户标识和鉴别，存取控制，设置视图，审计，数据加密等方式保障安全性很有必要。

3.3 遇到的困难及解决方法

无。