# 实验3 自主存取控制实验

[2.1用户实验 1](#_Toc88740393)

[2.1.1 创建用户 1](#_Toc88740394)

[语法格式 1](#_Toc88740395)

[参数说明 2](#_Toc88740396)

[示例 2](#_Toc88740397)

[2.1.2 修改用户 2](#_Toc88740398)

[语法格式 3](#_Toc88740399)

[参数说明 3](#_Toc88740400)

[示例 4](#_Toc88740401)

[2.1.3 删除用户 4](#_Toc88740402)

[语法格式 5](#_Toc88740403)

[参数说明 5](#_Toc88740404)

[示例 5](#_Toc88740405)

[2.2 角色实验 5](#_Toc88740406)

[2.2.1 创建角色 5](#_Toc88740407)

[语法格式 6](#_Toc88740408)

[参数说明 6](#_Toc88740409)

[示例 7](#_Toc88740410)

[2.2.2 修改角色 7](#_Toc88740411)

[语法格式 7](#_Toc88740412)

[参数说明 7](#_Toc88740413)

[示例 8](#_Toc88740414)

[2.2.3 删除角色 8](#_Toc88740415)

[语法格式 8](#_Toc88740416)

[参数说明 8](#_Toc88740417)

[示例 9](#_Toc88740418)

[2.3 授权实验 9](#_Toc88740419)

[2.3.1 GRANT 9](#_Toc88740420)

[语法格式 10](#_Toc88740421)

[参数说明 11](#_Toc88740422)

[示例1 12](#_Toc88740423)

[示例2 12](#_Toc88740424)

[示例3 13](#_Toc88740425)

[2.3.2 REVOKE 13](#_Toc88740426)

[语法格式 13](#_Toc88740427)

[参数说明 14](#_Toc88740428)

[示例 14](#_Toc88740429)

[2.4 使用Data Studio进行操作 15](#_Toc88740430)

[2.4.1 用户 15](#_Toc88740431)

[2.4.2 角色 18](#_Toc88740432)

# 实验3 自主存取控制实验

（1）实验目的

掌握自主存取控制权限的定义和维护方法。

（2）实验内容和要求

利用openGauss数据库，定义用户、角色，分配权限给用户、角色，回收权限，以相应的用户名登录数据库验证权限分配是否正确。选择一个应用场景，使用自主存取控制机制设计权限分配。

关于本实验：

数据库用户的主要用途是连接数据库、访问数据库对象和执行SQL语句。在连接数据库时，必须使用一个已经存在的数据库用户。因此，作为数据库管理员，需要为每一个需要连接数据库的使用者规划一个数据库用户。

一个数据库可能有多个用户需要访问，为了方便管理，可以先将权限分组，并授予给角色。每一个权限组对应一个角色。对于不同权限级别的用户，可以将对应的角色授予用户，相当于批量授予用户需要的权限，而不需要逐个授予权限。

## 2.1用户实验

### 2.1.1 创建用户

CREATE USER

功能描述：

创建一个数据库用户。

注意事项：

* 执行该语句的用户需要有CREATE USER系统权限。
* 用户名不能和数据库中已存在的用户名和角色名重复。
* 创建用户时，需要指定用户名和密码。

通过CREATE USER创建的用户，默认具有LOGIN权限。

通过CREATE USER创建用户的同时系统会在执行该命令的数据库中，为该用户创建一个同名的SCHEMA；其他数据库中，则不自动创建同名的SCHEMA；用户可使用CREATE SCHEMA命令，分别在其他数据库中，为该用户创建同名SCHEMA。

系统管理员在普通用户同名schema下创建的对象，所有者为schema的同名用户（非系统管理员）。

#### 语法格式

CREATE USER user\_name [ [ WITH ] option [ ... ] ] [ ENCRYPTED | UNENCRYPTED ] { PASSWORD | IDENTIFIED BY } { 'password' [EXPIRED] | DISABLE };

#### 参数说明

* user\_name

用户名。

取值范围：字符串，要符合标识符的命名规范。且最大长度不超过63个字符。

* password

登录密码。

密码规则如下：

◾密码默认不少于8个字符。

◾不能与用户名及用户名倒序相同。

◾至少包含大写字母（A-Z），小写字母（a-z），数字（0-9），非字母数字字符（限定为~!@#$%^&\*()-\_=+\|[{}];:,<.>/?）四类字符中的三类字符。

◾密码也可以是符合格式要求的密文字符串，这种情况主要用于用户数据导入场景，不推荐用户直接使用。如果直接使用密文密码，用户需要知道密文密码对应的明文，并且保证明文密码复杂度，数据库不会校验密文密码复杂度，直接使用密文密码的安全性由用户保证。

◾创建用户时，应当使用双引号或单引号将用户密码括起来。

#### 示例

创建用户jim，登录密码为Bigdata@123。

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

CREATE ROLE

同样的下面语句也可以创建用户。

postgres=# CREATE USER kim IDENTIFIED BY 'Bigdata@123';

CREATE ROLE

如果创建有＂创建数据库＂权限的用户，则需要加CREATEDB关键字。

postgres=# CREATE USER dim CREATEDB PASSWORD 'Bigdata@123';

CREATE ROLE

### 2.1.2 修改用户

ALTER USER

修改数据库用户的属性。

**注意事项**

ALTER USER中修改的会话参数只针对指定的用户，且在下一次会话中有效。

#### 语法格式

修改用户的权限等信息。

ALTER USER user\_name [ [ WITH ] option [ ... ] ];

其中option子句为。

{ CREATEDB | NOCREATEDB }

| { CREATEROLE | NOCREATEROLE }

| { INHERIT | NOINHERIT }

| { AUDITADMIN | NOAUDITADMIN }

| { SYSADMIN | NOSYSADMIN }

| { USEFT | NOUSEFT }

| { LOGIN | NOLOGIN }

| { REPLICATION | NOREPLICATION }

| {INDEPENDENT | NOINDEPENDENT}

| {VCADMIN | NOVCADMIN}

| CONNECTION LIMIT connlimit

| [ ENCRYPTED | UNENCRYPTED ] PASSWORD { 'password' [EXPIRED] | DISABLE | EXPIRED }

| [ ENCRYPTED | UNENCRYPTED ] IDENTIFIED BY { 'password' [ REPLACE 'old\_password' | EXPIRED ] | DISABLE }

| VALID BEGIN 'timestamp'

| VALID UNTIL 'timestamp'

| RESOURCE POOL 'respool'

| PERM SPACE 'spacelimit'

| ACCOUNT { LOCK | UNLOCK }

| PGUSER

#### 参数说明

* **user\_name**

现有用户名。

取值范围：已存在的用户名。

* **new\_password**

新密码。

密码规则如下：

* + 不能与当前密码相同。
  + 密码默认不少于8个字符。
  + 不能与用户名及用户名倒序相同。
  + 至少包含大写字母（A-Z），小写字母（a-z），数字（0-9），非字母数字字符（限定为~!@#$%^&\*()-\_=+\|[{}];:,<.>/?）四类字符中的三类字符。

取值范围：字符串。

* **old\_password**

旧密码。

* **ACCOUNT LOCK | ACCOUNT UNLOCK**
  + ACCOUNT LOCK：锁定帐户，禁止登录数据库。
  + ACCOUNT UNLOCK：解锁帐户，允许登录数据库。
* **PGUSER**

当前版本不允许修改用户的PGUSER属性。

#### 示例

将用户jim的登录密码由Bigdata@123修改为Abcd@123。

postgres=# ALTER USER jim IDENTIFIED BY 'Abcd@123' REPLACE 'Bigdata@123';

ALTER ROLE

为用户jim追加CREATEROLE权限。

postgres=# ALTER USER jim CREATEROLE;

ALTER ROLE

将enable\_seqscan的值设置为on，设置成功后，在下一会话中生效。

postgres=# ALTER USER jim SET enable\_seqscan TO on;

ALTER ROLE

锁定jim帐户。

postgres=# ALTER USER jim ACCOUNT LOCK;

ALTER ROLE

### 2.1.3 删除用户

DROP USER

**功能描述**

删除用户，同时会删除同名的schema。

**注意事项**

* 须使用CASCADE级联删除依赖用户的对象（除数据库外）。当删除用户的级联对象时，如果级联对象处于锁定状态，则此级联对象无法被删除，直到对象被解锁或锁定级联对象的进程被杀死。
* 在openGauss中，存在一个配置参数enable\_kill\_query，此参数在配置文件postgresql.conf中。此参数影响级联删除用户对象的行为：
  + 当参数enable\_kill\_query为on ，且使用CASCADE模式删除用户时，会自动kill锁定用户级联对象的进程，并删除用户。
  + 当参数enable\_kill\_query为off，且使用CASCADE模式删除用户时，会等待锁定级联对象的进程结束之后再删除用户。
* 在数据库中删除用户时，如果依赖用户的对象在其他数据库中或者依赖用户的对象是其他数据库，请用户先手动删除其他数据库中的依赖对象或直接删除依赖数据库，再删除用户。即drop user不支持跨数据库进行级联删除。
* 如果该用户被DATA SOURCE对象依赖时，无法直接级联删除该用户，需要手动删除对应的DATA SOURCE对象之后再删除该用户。

#### 语法格式

DROP USER [ IF EXISTS ] user\_name [, ...] [ CASCADE | RESTRICT ];

#### 参数说明

* **IF EXISTS**

如果指定的用户不存在，发出一个notice而不是抛出一个错误。

* **user\_name**

待删除的用户名。

取值范围：已存在的用户名。

* **CASCADE | RESTRICT**
  + CASCADE：级联删除依赖用户的对象。
  + RESTRICT：如果用户还有任何依赖的对象，则拒绝删除该用户（缺省行为）。

#### 示例

postgres=# DROP USER kim CASCADE;

DROP ROLE

postgres=# DROP USER jim CASCADE;

DROP ROLE

postgres=# DROP USER dim CASCADE;

DROP ROLE

## 2.2 角色实验

### 2.2.1 创建角色

CREATE ROLE

创建角色。

角色是拥有数据库对象和权限的实体。在不同的环境中角色可以认为是一个用户，一个组或者兼顾两者。

**注意事项**

* 在数据库中添加一个新角色，角色无登录权限。
* 创建角色的用户必须具备CREATE ROLE的权限或者是系统管理员。

#### 语法格式

CREATE ROLE role\_name [ [ WITH ] option [ ... ] ] [ ENCRYPTED | UNENCRYPTED ] { PASSWORD | IDENTIFIED BY } { 'password' [EXPIRED] | DISABLE };

#### 参数说明

**role\_name**

角色名称。

取值范围：字符串，要符合标识符的命名规范，且最多为63个字符。若超过63个字符，数据库会截断并保留前63个字符当做角色名称。在创建角色时，数据库的时候会给出提示信息。

标识符需要为字母、下划线、数字（0-9）或美元符号（$），且必须以字母（a-z）或下划线（\_）开头。

**password**

登录密码。

密码规则如下：

* + 密码默认不少于8个字符。
  + 不能与用户名及用户名倒序相同。
  + 至少包含大写字母（A-Z），小写字母（a-z），数字（0-9），非字母数字字符（限定为~!@#$%^&\*()-\_=+\|[{}];:,<.>/?）四类字符中的三类字符。
  + 密码也可以是符合格式要求的密文字符串，这种情况主要用于用户数据导入场景，不推荐用户直接使用。如果直接使用密文密码，用户需要知道密文密码对应的明文，并且保证明文密码复杂度，数据库不会校验密文密码复杂度，直接使用密文密码的安全性由用户保证。
  + 创建角色时，应当使用双引号或单引号将用户密码括起来。

取值范围：字符串。

**EXPIRED**

在创建用户时可指定EXPIRED参数，即创建密码失效用户，该用户不允许执行简单查询和扩展查询。只有在修改自身密码后才可正常执行语句。

**DISABLE**

默认情况下，用户可以更改自己的密码，除非密码被禁用。要禁用用户的密码，请指定DISABLE。禁用某个用户的密码后，将从系统中删除该密码，此类用户只能通过外部认证来连接数据库，例如：kerberos认证。只有管理员才能启用或禁用密码。普通用户不能禁用初始用户的密码。要启用密码，请运行ALTER USER并指定密码。

**ENCRYPTED | UNENCRYPTED**

控制密码存储在系统表里的口令是否加密。按照产品安全要求，密码必须加密存储，所以，UNENCRYPTED在openGauss中禁止使用。因为系统无法对指定的加密口令字符串进行解密，所以如果目前的口令字符串已经是用SHA256加密的格式，则会继续照此存放，而不管是否声明了ENCRYPTED或UNENCRYPTED。这样就允许在dump/restore的时候重新加载加密的口令。

**SYSADMIN | NOSYSADMIN**

决定一个新角色是否为“系统管理员”，具有SYSADMIN属性的角色拥有系统最高权限。

缺省为NOSYSADMIN。

#### 示例

--创建一个角色，名为manager，密码为xxxxxxxxx。

openGauss=# CREATE ROLE manager IDENTIFIED BY 'xxxxxxxxx';

--创建一个角色，从2015年1月1日开始生效，到2026年1月1日失效。

openGauss=# CREATE ROLE miriam WITH LOGIN PASSWORD 'xxxxxxxxx' VALID BEGIN '2015-01-01' VALID UNTIL '2026-01-01';

### 2.2.2 修改角色

ALTER ROLE

**功能描述**

修改角色属性。

#### 语法格式

修改角色的权限。

ALTER ROLE role\_name [ [ WITH ] option [ ... ] ];

#### 参数说明

**role\_name**

现有角色名。

取值范围：已存在的用户名。

**IN DATABASE database\_name**

表示修改角色在指定数据库上的参数。

**SET configuration\_parameter**

设置角色的参数。ALTER ROLE中修改的会话参数只针对指定的角色，且在下一次该角色启动的会话中有效。

DEFAULT：表示清除configuration\_parameter参数的值，configuration\_parameter参数的值将继承本角色新产生的SESSION的默认值。

FROM CURRENT：取当前会话中的值设置为configuration\_parameter参数的值。

**RESET configuration\_parameter/ALL**

清除configuration\_parameter参数的值。与SET configuration\_parameter TO DEFAULT的效果相同。

取值范围：ALL表示清除所有参数的值。

**ACCOUNT LOCK | ACCOUNT UNLOCK**

ACCOUNT LOCK：锁定帐户，禁止登录数据库。

ACCOUNT UNLOCK：解锁帐户，允许登录数据库。

**PGUSER**

当前版本不允许修改角色的PGUSER属性

**PASSWORD/IDENTIFIED BY 'password'**

重置或修改用户密码。除了初始用户外其他管理员或普通用户修改自己的密码需要输入正确的旧密码。只有初始用户、系统管理员（sysadmin）或拥有创建用户（CREATEROLE）权限的用户才可以重置普通用户密码，无需输入旧密码。初始用户可以重置系统管理员的密码，系统管理员不允许重置其他系统管理员的密码。

**EXPIRED**

设置密码失效。只有初始用户、系统管理员（sysadmin）或拥有创建用户（CREATEROLE）权限的用户才可以设置用户密码失效，其中系统管理员也可以设置自己或其他系统管理员密码失效。不允许设置初始用户密码失效。

密码失效的用户可以登录数据库但不能执行查询操作，只有修改密码或由管理员重置密码后才可以恢复正常查询操作。

#### 示例

--修改角色manager的密码为abcd@123。

openGauss=# ALTER ROLE manager IDENTIFIED BY 'abcd@123' REPLACE 'xxxxxxxxx';

--修改角色manager为系统管理员。

openGauss=# ALTER ROLE manager SYSADMIN;

### 2.2.3 删除角色

**DROP ROLE**

**功能描述**

删除指定的角色。

#### 语法格式

DROP ROLE [ IF EXISTS ] role\_name [, ...];

#### 参数说明

* **IF EXISTS**

如果指定的角色不存在，则发出一个notice而不是抛出一个错误。

* **role\_name**

要删除的角色名称。

取值范围：已存在的角色。

#### 示例

--删除角色manager。

openGauss=# DROP ROLE manager;

--删除角色miriam。

openGauss=# DROP ROLE miriam

## 2.3 授权实验

### 2.3.1 GRANT

GRANT

**功能描述**

对角色和用户进行授权操作。

使用GRANT命令进行用户授权包括以下三种场景：

**•将系统权限授权给角色或用户**

系统权限又称为用户属性，包括SYSADMIN、CREATEDB、CREATEROLE、AUDITADMIN、MONADMIN、OPRADMIN、POLADMIN、INHERIT、REPLICATION、VCADMIN和LOGIN等。

系统权限一般通过CREATE/ALTER ROLE语法来指定。其中，SYSADMIN权限可以通过GRANT/REVOKE ALL PRIVILEGE授予或撤销。但系统权限无法通过ROLE和USER的权限被继承，也无法授予PUBLIC。

**•将数据库对象授权给角色或用户**

将数据库对象（表和视图、指定字段、数据库、函数、模式、表空间等）的相关权限授予特定角色或用户；

GRANT命令将数据库对象的特定权限授予一个或多个角色。这些权限会追加到已有的权限上。

关键字PUBLIC表示该权限要赋予所有角色，包括以后创建的用户。PUBLIC可以看做是一个隐含定义好的组，它总是包括所有角色。任何角色或用户都将拥有通过GRANT直接赋予的权限和所属的权限，再加上PUBLIC的权限。

如果声明了WITH GRANT OPTION，则被授权的用户也可以将此权限赋予他人，否则就不能授权给他人。这个选项不能赋予PUBLIC，这是openGauss特有的属性。

openGauss会将某些类型的对象上的权限授予PUBLIC。默认情况下，对表、表字段、序列、外部数据源、外部服务器、模式或表空间对象的权限不会授予PUBLIC，而以下这些对象的权限会授予PUBLIC：数据库的CONNECT权限和CREATE TEMP TABLE权限、函数的EXECUTE特权、语言和数据类型（包括域）的USAGE特权。当然，对象拥有者可以撤销默认授予PUBLIC的权限并专门授予权限给其他用户。为了更安全，建议在同一个事务中创建对象并设置权限，这样其他用户就没有时间窗口使用该对象。另外可参考安全加固指南的权限控制章节，对PUBLIC用户组的权限进行限制。这些初始的默认权限可以使用ALTER DEFAULT PRIVILEGES命令修改。

对象的所有者缺省具有该对象上的所有权限，出于安全考虑所有者可以舍弃部分权限，但ALTER、DROP、COMMENT、INDEX、VACUUM以及对象的可再授予权限属于所有者固有的权限，隐式拥有。

**•将角色或用户的权限授权给其他角色或用户**

将一个角色或用户的权限授予一个或多个其他角色或用户。在这种情况下，每个角色或用户都可视为拥有一个或多个数据库权限的集合。

当声明了WITH ADMIN OPTION，被授权的用户可以将该权限再次授予其他角色或用户，以及撤销所有由该角色或用户继承到的权限。当授权的角色或用户发生变更或被撤销时，所有继承该角色或用户权限的用户拥有的权限都会随之发生变更。

数据库系统管理员可以给任何角色或用户授予/撤销任何权限。拥有CREATEROLE权限的角色可以赋予或者撤销任何非系统管理员角色的权限。

#### 语法格式

•将表或视图的访问权限赋予指定的用户或角色。

GRANT { { SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES | ALTER | DROP | COMMENT | INDEX | VACUUM } [, ...]

| ALL [ PRIVILEGES ] }

ON { [ TABLE ] table\_name [, ...]

| ALL TABLES IN SCHEMA schema\_name [, ...] }

TO { [ GROUP ] role\_name | PUBLIC } [, ...]

[ WITH GRANT OPTION ];

•将表中字段的访问权限赋予指定的用户或角色。

GRANT { {{ SELECT | INSERT | UPDATE | REFERENCES | COMMENT } ( column\_name [, ...] )} [, ...]

| ALL [ PRIVILEGES ] ( column\_name [, ...] ) }

ON [ TABLE ] table\_name [, ...]

TO { [ GROUP ] role\_name | PUBLIC } [, ...]

[ WITH GRANT OPTION ];

•将序列的访问权限赋予指定的用户或角色，LARGE字段属性可选，赋权语句不区分序列是否为LARGE。

GRANT { { SELECT | UPDATE | USAGE | ALTER | DROP | COMMENT } [, ...]

| ALL [ PRIVILEGES ] }

ON { [ SEQUENCE ] sequence\_name [, ...]

| ALL SEQUENCES IN SCHEMA schema\_name [, ...] }

TO { [ GROUP ] role\_name | PUBLIC } [, ...]

[ WITH GRANT OPTION ];

•将数据库的访问权限赋予指定的用户或角色。

GRANT { { CREATE | CONNECT | TEMPORARY | TEMP | ALTER | DROP | COMMENT } [, ...]

| ALL [ PRIVILEGES ] }

ON DATABASE database\_name [, ...]

TO { [ GROUP ] role\_name | PUBLIC } [, ...]

[ WITH GRANT OPTION ];

•将域的访问权限赋予指定的用户或角色。

GRANT { USAGE | ALL [ PRIVILEGES ] }

ON DOMAIN domain\_name [, ...]

TO { [ GROUP ] role\_name | PUBLIC } [, ...]

[ WITH GRANT OPTION ];

#### 参数说明

* system\_priviege\_name

系统权限名。详细见文档末尾的附表。

* role\_name

角色名。将角色授权给用户/角色，表示被授权的用户/角色继承role角色的所有系统权限。将角色授权给其他的角色时，不能循环授权。如：将role1授权给role2，将role2授权给role3，将role3授权给role1。

* ALL [ PRIVILEGES ]

表示所有系统权限， PRIVILEGES可省略。

* WITH ADMIN OPTION

此选项表示的含义如下：

– grantee拥有将此系统权限/角色授权给其他用户或者角色的权限（即grant权限的传递），权限回收时，不会级联回收此用户授予其他用户的权限。

– grantee拥有删除自身拥有的该系统权限或role的权限。

– 此选项一旦设置后，只能通过REVOKE语句将整个权限回收。不能单独回收此选项而保留权限。

* object\_privilege\_name：对象权限名称。不同的对象的权限集合不同。目前支持的对象类型有：表、视图、序列、存储过程、函数、触发器、系统高级包。
* object\_type

对象类型名称。当不指定object\_type时，默认对象查找优先级顺序为：表、视图、序列、高级包、存储过程、函数。

* grantee

受权者。可以是用户或者角色，一次可以指定多个用户或者角色，但不能超过63个。

* WITH GRANT OPTION

– 使用with grant option。被授权用户可将所获得的权限再次授予其它用户或角色，并且权限的取消是级联的。

– 此选项一旦设置后，只能通过REVOKE语句将整个权限回收。不能单独回收此选项而保留权限。

#### 示例1

**将系统权限授权给用户或者角色。**

创建名为joe的用户，并将sysadmin权限授权给他。

openGauss=# CREATE USER joe PASSWORD 'xxxxxxxxx';

openGauss=# GRANT ALL PRIVILEGES TO joe;

授权成功后，用户joe会拥有sysadmin的所有权限。

#### 示例2

**将对象权限授权给用户或者角色**。

1. 撤销joe用户的sysadmin权限，然后将模式tpcds的使用权限和表tpcds.reason的所有权限授权给用户joe。

openGauss=# REVOKE ALL PRIVILEGES FROM joe;

openGauss=# GRANT USAGE ON SCHEMA tpcds TO joe;

openGauss=# GRANT ALL PRIVILEGES ON tpcds.reason TO joe;

授权成功后，joe用户就拥有了tpcds.reason表的所有权限，包括增删改查等权限。

1. 将tpcds.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。

openGauss=# GRANT select (r\_reason\_sk,r\_reason\_id,r\_reason\_desc),update (r\_reason\_desc) ON tpcds.reason TO joe;

授权成功后，用户joe对tpcds.reason表中r\_reason\_sk，r\_reason\_id的查询权限会立即生效。如果joe用户需要拥有将这些权限授权给其他用户的权限，可以通过以下语法对joe用户进行授权。

openGauss=# GRANT select (r\_reason\_sk, r\_reason\_id) ON tpcds.reason TO joe WITH GRANT OPTION;

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

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

创建角色tpcds\_manager，将模式tpcds的访问权限授权给角色tpcds\_manager，并授予该角色在tpcds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他人。

openGauss=# CREATE ROLE tpcds\_manager PASSWORD 'xxxxxxxxx';

openGauss=# GRANT USAGE,CREATE ON SCHEMA tpcds TO tpcds\_manager;

将表空间tpcds\_tbspc的所有权限授权给用户joe，但用户joe无法将权限继续授予其他用户。

openGauss=# CREATE TABLESPACE tpcds\_tbspc RELATIVE LOCATION 'tablespace/tablespace\_1';

openGauss=# GRANT ALL ON TABLESPACE tpcds\_tbspc TO joe;

#### 示例3

**用户或者角色的权限授权给其他用户或角色。**

创建角色manager，将joe的权限授权给manager，并允许该角色将权限授权给其他人。

openGauss=# CREATE ROLE manager PASSWORD 'xxxxxxxxx';

openGauss=# GRANT joe TO manager WITH ADMIN OPTION;

创建用户senior\_manager，将用户manager的权限授权给该用户。

openGauss=# CREATE ROLE senior\_manager PASSWORD 'xxxxxxxxx';

openGauss=# GRANT manager TO senior\_manager;

### 2.3.2 REVOKE

功能描述：

REVOKE用于撤销一个或多个角色的权限。

**注意事项**

非对象所有者试图在对象上REVOKE权限，命令按照以下规则执行：

* 如果授权用户没有该对象上的权限，则命令立即失败。
* 如果授权用户有部分权限，则只撤销那些有授权选项的权限。
* 如果授权用户没有授权选项，REVOKE ALL PRIVILEGES形式将发出一个错误信息，而对于其他形式的命令而言，如果是命令中指定名称的权限没有相应的授权选项，该命令将发出一个警告。
* 不允许对表分区进行REVOKE操作，对分区表进行REVOKE操作会引起告警。

#### 语法格式

* **回收指定表或视图上权限。**

REVOKE [ GRANT OPTION FOR ]

{ { SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES | ALTER | DROP | COMMENT | INDEX | VACUUM }[, ...]

| ALL [ PRIVILEGES ] }

ON { [ TABLE ] table\_name [, ...]

| ALL TABLES IN SCHEMA schema\_name [, ...] }

FROM { [ GROUP ] role\_name | PUBLIC } [, ...]

[ CASCADE | RESTRICT ];

* **回收表上指定字段权限。**

REVOKE [ GRANT OPTION FOR ]

{ {{ SELECT | INSERT | UPDATE | REFERENCES | COMMENT } ( column\_name [, ...] )}[, ...]

| ALL [ PRIVILEGES ] ( column\_name [, ...] ) }

ON [ TABLE ] table\_name [, ...]

FROM { [ GROUP ] role\_name | PUBLIC } [, ...]

[ CASCADE | RESTRICT ];

* **回收指定数据库上权限。**

REVOKE [ GRANT OPTION FOR ]

{ { CREATE | CONNECT | TEMPORARY | TEMP | ALTER | DROP | COMMENT } [, ...]

| ALL [ PRIVILEGES ] }

ON DATABASE database\_name [, ...]

FROM { [ GROUP ] role\_name | PUBLIC } [, ...]

[ CASCADE | RESTRICT ];

#### 参数说明

关键字PUBLIC表示一个隐式定义的拥有所有角色的组。

任何特定角色拥有的特权包括直接授予该角色的特权、从该角色作为其成员的角色中得到的权限以及授予给PUBLIC的权限。因此，从PUBLIC收回SELECT特权并不一定会意味着所有角色都会失去在该对象上的SELECT特权，那些直接被授予的或者通过另一个角色被授予的角色仍然会拥有它。类似地，从一个用户收回SELECT后，如果PUBLIC仍有SELECT权限，该用户还是可以使用SELECT。指定GRANT OPTION FOR时，只撤销对该权限授权的权力，而不撤销该权限本身。

如用户A拥有某个表的UPDATE权限，及WITH GRANT OPTION选项，同时A把这个权限赋予了用户B，则用户B持有的权限称为依赖性权限。当用户A持有的权限或者授权选项被撤销时，必须声明CASCADE，将所有依赖性权限都撤销。

一个用户只能撤销由它自己直接赋予的权限。例如，如果用户A被指定授权（WITH ADMIN OPTION）选项，且把一个权限赋予了用户B，然后用户B又赋予了用户C，则用户A不能直接将C的权限撤销。但是，用户A可以撤销用户B的授权选项，并且使用CASCADE。这样，用户C的权限就会自动被撤销。另外一个例子：如果A和B都赋予了C同样的权限，则A可以撤销他自己的授权选项，但是不能撤销B的，因此C仍然拥有该权限。

如果执行REVOKE的角色持有的权限是通过多层成员关系获得的，则具体是哪个包含的角色执行的该命令是不确定的。在这种场合下，最好的方法是使用SET ROLE成为特定角色，然后执行REVOKE，否则可能导致删除了不想删除的权限，或者是任何权限都没有删除。

#### 示例

将系统权限从用户joe收回。

步骤1

创建用户jack

**CREATE** **USER** jack **IDENTIFIED** **BY** "database\_123";

步骤2

将数据库postgres的连接权限授权给用户jack予其在postgres中创建schema的权限，而且允许jack限授权给其他用户。

**GRANT** **create**,connect **on** **database** postgres **TO** jack **WITH** **GRANT** **OPTION**;

步骤3

从受权用户jack回收所有权限

**REVOKE** **ALL** **PRIVILEGES** **FROM** jack;

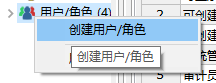
删除用户

**DROP** **USER** joe **CASCADE**;

## 2.4 使用Data Studio进行操作

### 2.4.1 用户

右键用户，选择创建用户：



在弹出的窗口中填入用户名和密码和想要添加的权限点击下一步

**注意密码规则！**

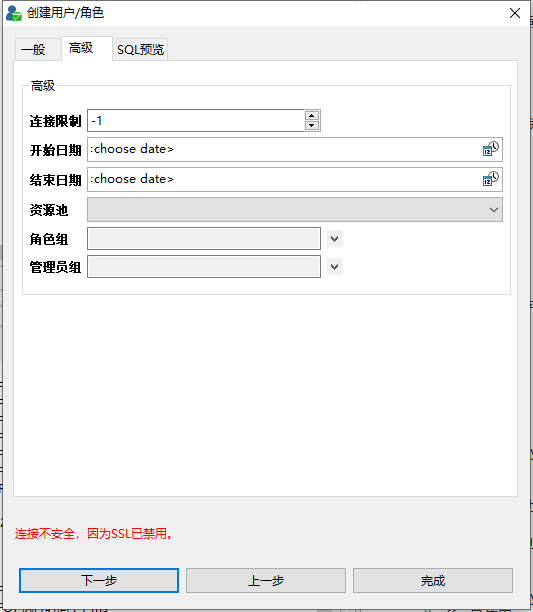
◾密码默认不少于8个字符。

◾不能与用户名及用户名倒序相同。

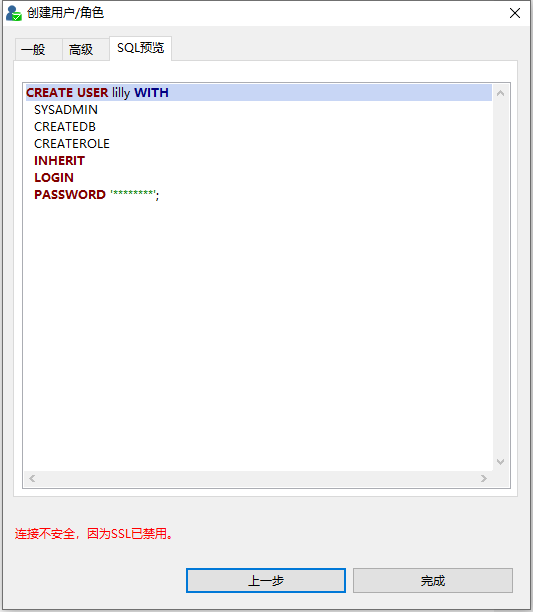
◾至少包含大写字母（A-Z），小写字母（a-z），数字（0-9），非字母数字字符（限定为~!@#$%^&\*()-\_=+\|[{}];:,<.>/?）四类字符中的三类字符。



在第二个选项卡中可以点击图标进行高级设置：



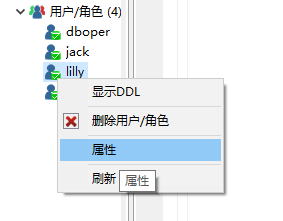
在第三个选项卡中可以预览操作生成的SQL代码



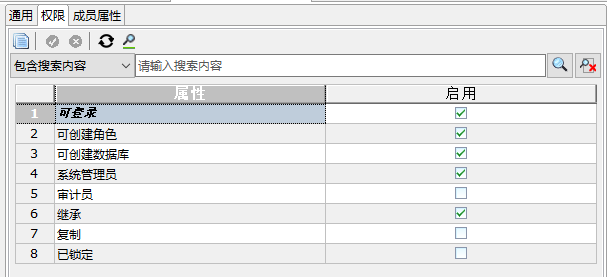
点击完成，可以看到用户/角色栏已经添加了jack用户



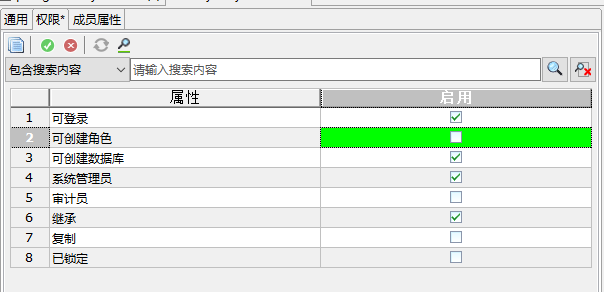
编辑用户权限，对一个已经创建的用户右键选择属性可以查看和编辑权限



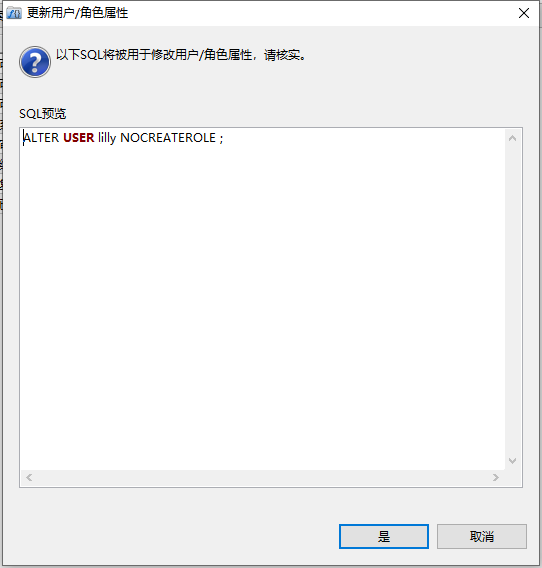
右上角选择权限栏



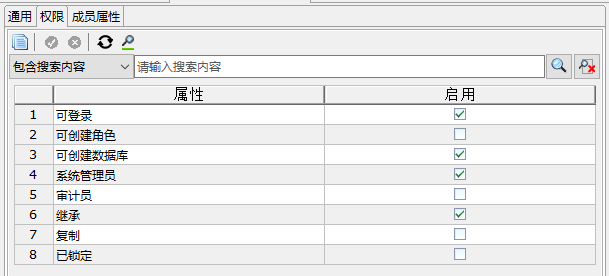
双击改变状态



点击绿色对勾，弹出SQL预览

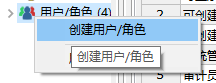


点击是按钮，可以看到用户状态已经改变

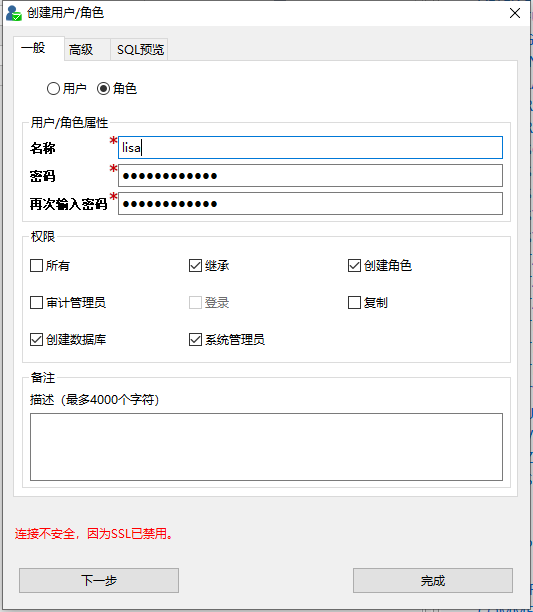


### 2.4.2 角色

右键角色，点击创建角色：



选择角色，填入角色名和密码：



其他操作于用户操作相同。