实验七 数据库安全性

计算机科学与技术

2021160291 李景昊

一、目的和要求

1．掌握查看、重命名及删除用户定义的数据类型。
2．掌握如何对数据库和表进行安全控制。

二、内容和步骤

1、**数据库安全控制**Database Security Control**

建立用户前先分析该用户的用途。在应用中我们经常会建立很多OpenGauss的登录用户，每个用户都有不同的用途，完成的功能也不相同，如果您是一个数据库的管理员，面对这么多数据库的登录用户是否对这些登录用户有很详细的了解，这里所说的了解是指具体的用途，比如：user_a只能读某个数据库的某个表；user_b可以读、写某个数据库的某些表等。有些管理员为了偷懒省事，直接复给它个db_owner更有甚者给它System Administrators权限,这样暂时虽然给您的操作带来了方便，但同时也给hacker们带来了方便。因此这里有一个建议：“在建立登录用户时，把它的详细用途用笔记下，然后整理，同时也为下次核查数据库的安全做了参考”，例如表5-1：

表5-1 角色及权限

用户名	对应数据库	对应表	权限	功能描述	用途
User_a	Db_a	Aa,ab,ac	Public (只读)	Select	浏览新闻
	Db_b	Ba,bb			
User_b	Db_c	*	Public, Db_owner	Update, insert	更新新闻
User_c	*	*	administrotors		数据库管理

2、建立**OpenGuass**用户及授权**

（1）打开功能栏上方的角色按钮，如图7-1所示：

图7-1 角色按钮

（2）选择新建角色进入创建角色界面，如图7-2所示：

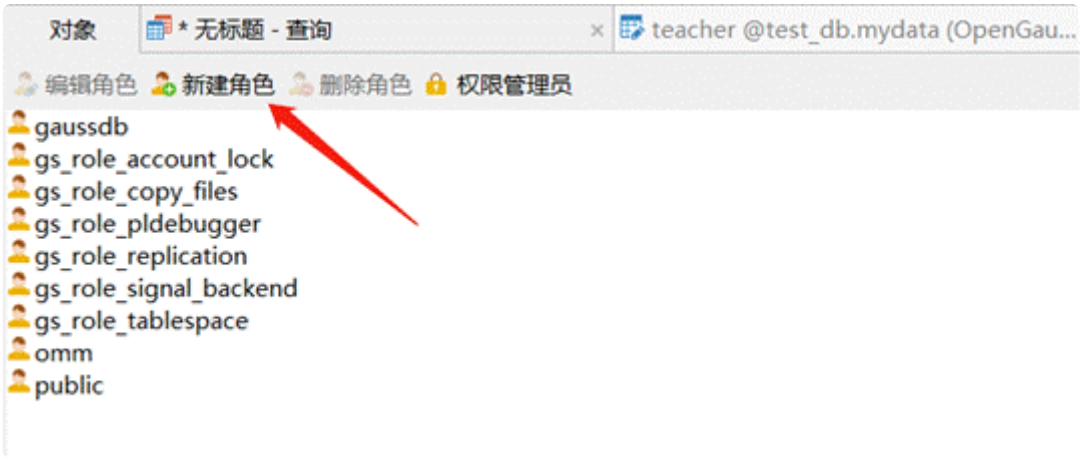


图7-2 创建角色界面

（3）进入创建角色界面后可以看到相对应的配置信息，如图7-3所示：

图7-3 角色配置

（4）创建角色界面中的常规中，我们对角色命名为”testrole”，并赋予密码，剩下的可选框是赋予该角色的权利，选择继承权限一栏。如图7-4所示：

图7-4 角色常规选项卡

（5）点击添加权限，对该角色进行权限设置，如图7-5所示：

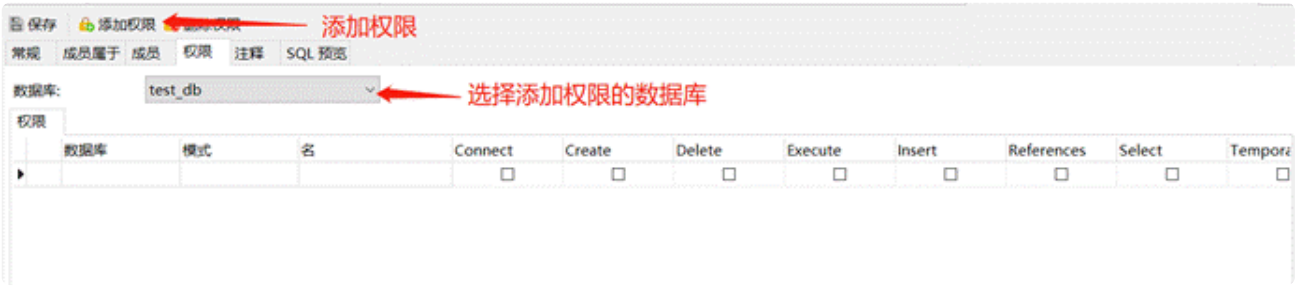


图7-5 角色权限选项卡

（6）我们将该角色定义为学生角色，因此该角色所拥有的查看成绩表的权力以及查看课程表中课程号、课程名称、书标识的权力，如图7-6，7-7所示：

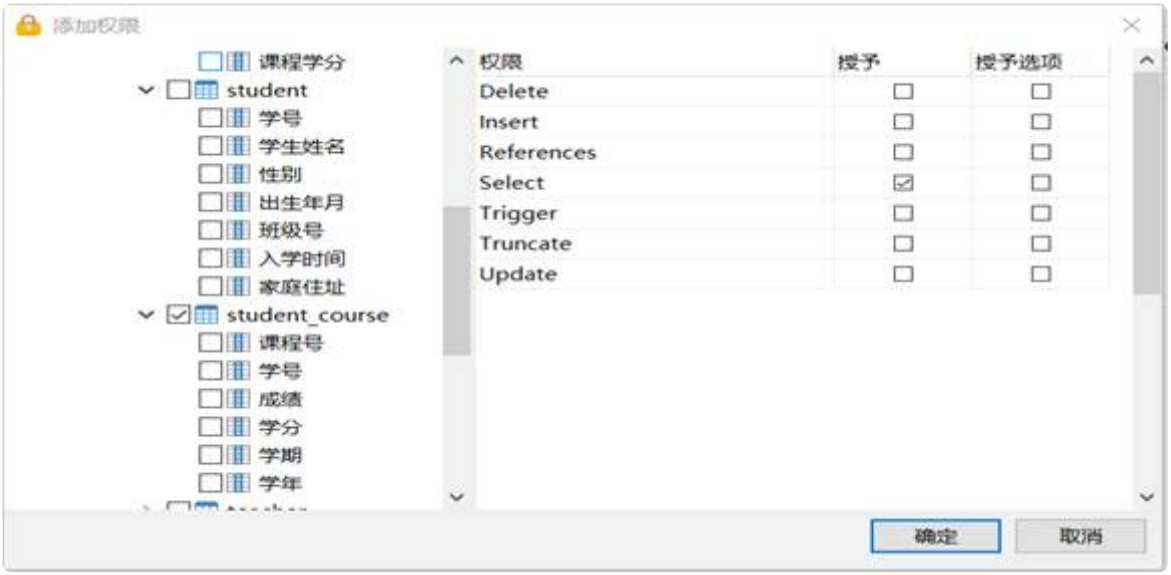


图7-6 Student角色的权限

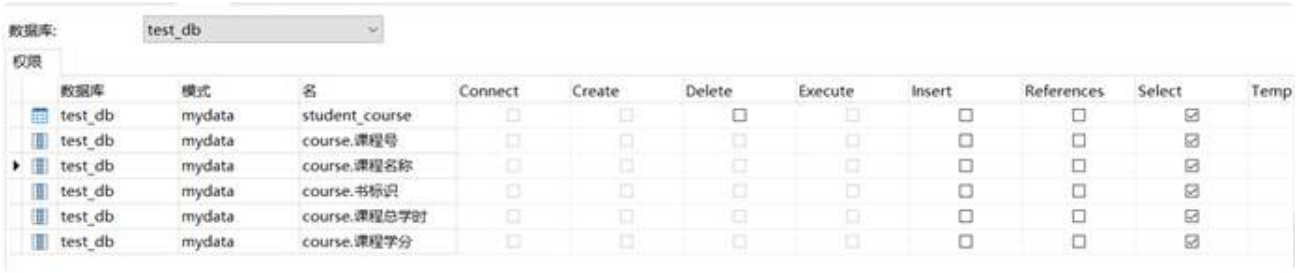


图7-7 Student角色的权限对象

此外，也可以用SQL语句对角色赋权，请注意当我们向目标角色进行赋权的时候，需要考虑当前我们使用的角色是否有相对应的权力进行赋权，如果没有则赋权失败。赋权语句如图7-8所示：

```
CREATE ROLE "testrole" NOINHERIT LOGIN PASSWORD '********';

GRANT Select ON TABLE "mydata"."student_course" TO "testrole";

GRANT Select("课程号") ON TABLE "mydata"."course" TO "testrole";

GRANT Select("课程名称") ON TABLE "mydata"."course" TO "testrole";

GRANT Select("书标识") ON TABLE "mydata"."course" TO "testrole";

GRANT Select("课程总学时") ON TABLE "mydata"."course" TO "testrole";

GRANT Select("课程学分") ON TABLE "mydata"."course" TO "testrole";
```

图7-8 SQL授权

3、自定义数据类型**

（1）CREATE DOMAIN自定义数据类型

OpenGauss可通过CREATE DOMAIN语句自定义数据类型，自定义数据类型包含CHECK，NOT NULL等约束。如

```
CREATE DOMAIN person_name AS VARCHAR NOT NULL CHECK (value!~ '\s');

```

该语句定义了一个名为 person_name的数据类型，其中包含非空且不可以有空格存在的约束条件。创建域之后，域不允许进行重命名和修改以及删除。

（2）CREATE TYPE定义复合数据类型

存储过程返回值的数据类型可以是CREATE TYPE命令定义的复合数据类型。如

```
CREATE TYPE Item_details AS (
    item_id INT,
    item_name VARCHAR,
    item_price Numeric(5,2)
);

```

该语句创建一个名为Item_details的数据类型，其中规定了商品ID、商品名字、商品价格的数据类型。

（3）alter type重命名数据类型

alter type对已经定义的数据类型进行重命名，如

```
alter type Item_details rename to Idetail;
```

（4）Drop type删除数据类型：

```
1  drop type if EXISTS Idetail;
```

4、用**navicat**定义域**Domain**

（1）点击navicat功能栏中的“其它”并选择“域”，如图7-9所示：

图7-9 navicat定义域Domain

（2）选择新建域，设置与person_name相同的数据类型，如图7-10所示。

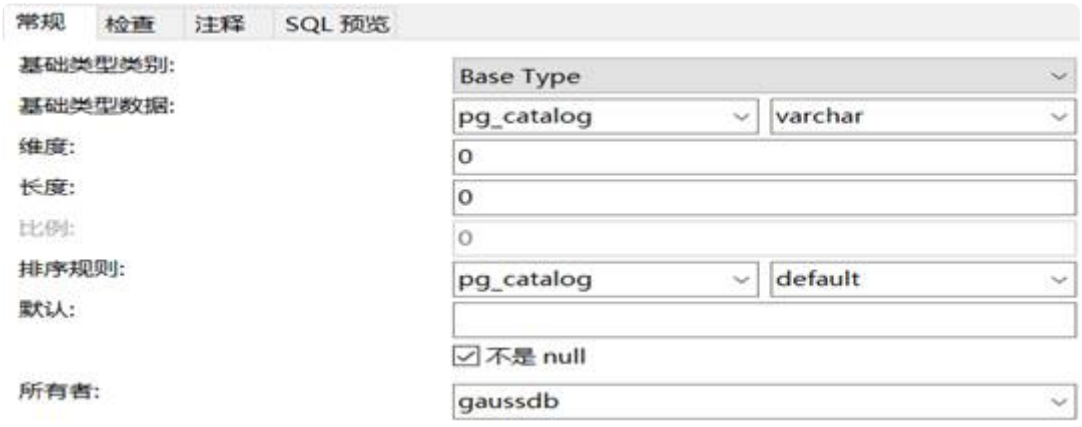


图7-10 设置域的数据类型

（3）在检查一栏中写上与person_name相关的约束，如图7-11所示

图7-11 添加域的约束

（4）保存并生成相对应的域，如图7-12.

图7-12 保存域

5、查看自定义数据类型**

查看自定义数据类型，由于Navicat与OpenGauss的版本兼容性存在一定的问题，因此这里提供在数据库系统界面直接对自定义的数据类型进行查看，输入命令“\dD“查看相对应的domain，如图7-13所示。

```
test_db=# \dT
         List of data types
 Schema |     Name      | Description
--------+---------------+-------------
 public | addtype       |
 public | item_details  |
 public | person_name   |
(3 rows)
```

图7-13 查看自定义数据类型

输入命令”\dT+”查看所有数据类型，如图7-14。

```
test_db=# \dT+;
                                  List of data types
 Schema |     Name     | Internal name | Size  | Elements | Access privileges | Description
--------+--------------+---------------+-------+----------+-------------------+-------------
 public | addtype      | addtype       | var   |          |                   |
 public | item_details | item_details  | tuple |          |                   |
 public | person_name  | person_name   | var   |          |                   |
(3 rows)

test_db=#
```

图7-14 查看所有数据类型

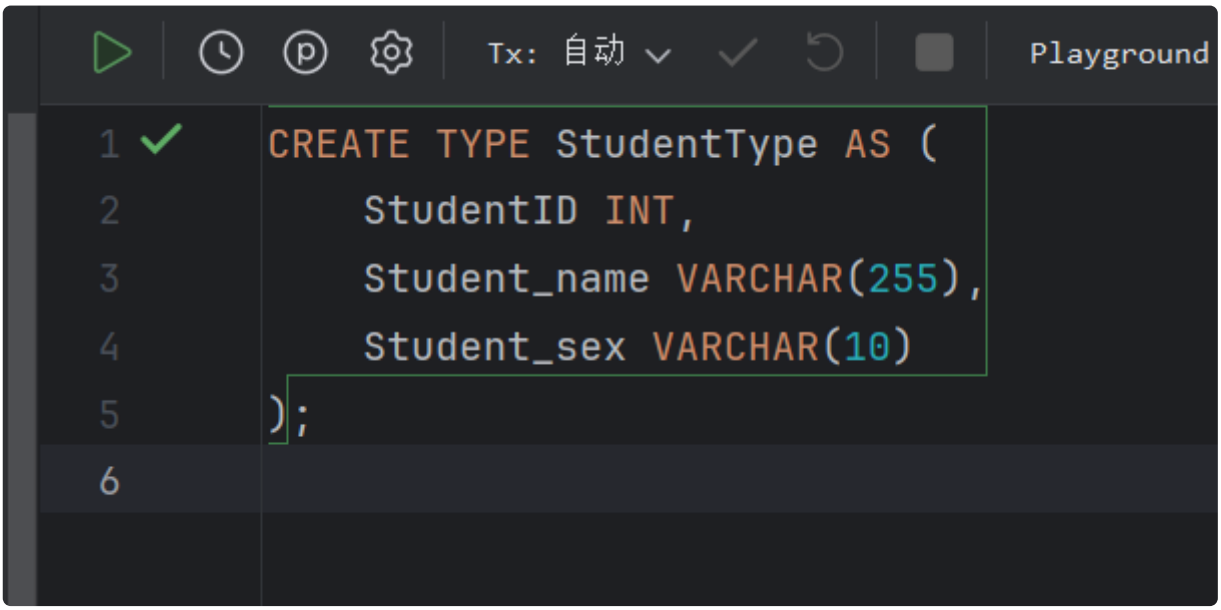
三、实验内容

练习1：创建用户自定义的数据类型

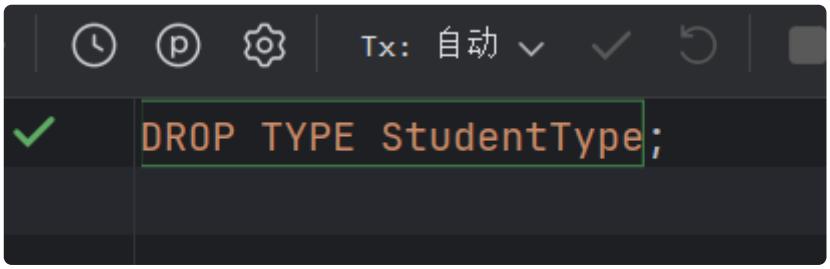
(1) 连接对象资源管理器，用guassdb录

(2) 在数据库列表中单击test数据库

(3) 用SQL语句自定义一个Student复合数据类型，包括StudentID INT, Student_name Varchar, Stduent_sex varchar。 执行该语句后到对象资源管理器中查看该数据类型。



(4) 删除(3)中创建的数据类型。



练习2**：在数据库 School 中建立三个用户 USER1，USER2 和 USER3，它们在数据库中的角色是 PUBLIC。请按以下要求，分别以管理员身份或这三个用户的身份等录到数据库中，进行操作。

（1）授予所有用户对表 COURSES 的查询权限。

```
GRANT SELECT ON COURSES TO PUBLIC;
```

（2）授予用户 USER1 对表 STUDENTS 插入和更新的权限，但不授予删除权限，并且授予用户 USER1 传播这两个权限的权利。

```
GRANT INSERT, UPDATE ON STUDENTS TO USER1 WITH GRANT OPTION;
```

（3）允许用户 USER2 在表 CHOICE 中插入元组，更新的 SCORE 列，可以选取除了SID 以外的所有列。

```
GRANT INSERT, UPDATE, SELECT (no, t_id,cid,score) ON CHOICE TO USER2;
```

（4）用户 USER1 授予用户 USER2 对表 STUDENTS 插入和更新的权限，并且授予用户 USER2 传播插入操作的权利。

```
GRANT INSERT, UPDATE ON STUDENTS TO USER2 WITH GRANT OPTION;
```

（5）收回对用户 USER1 对表 COURSES 查询权限的授权。

```
REVOKE SELECT ON COURSES FROM USER1;
```

（6）由上面（2）和（4）的授权，再由用户 USER2 对用户 USER3 授予表 STUDENTS

```
GRANT INSERT, UPDATE ON STUDENTS TO USER3 WITH GRANT OPTION;
```

插入和更新的权限，并且授予用户 USER3 传播插入操作的权利。这时候，如果由 USER3对 USER1 授予表 STUDENTS 的插入和更新权限是否能得到成功?

如果数据库允许传播权限，那么 USER1 将获得这些权限

如果能够成功，那么如果有用户 USER2 取消 USER3 的权限， USER1 会有什么影响？如果再由 DBA 取消 USER1对的权限，对 USER2 有什么影响?

USER1 和USER2的权限可能会受到影响，具体取决于数据库系统如何处理这种情况