6 数据库安全

6.1 实验介绍

6.1.1 关于本实验

演示数据库中对于用户以及角色的相关操作；在 GaussDB(DWS)数据库集群上执行对象权限相关操作；演示端到端构建基于角色的权限管理，并在数据库模型中加以应用。

本章实验以华为公有云 GaussDB(DWS)服务为基础平台，同时在 ECS 服务器服务器上搭建 gsql 客户端，本地通过弹性公网 IP 连接 ECS 服务器，通过 gsql 客户端连接 GaussDB(DWS)数据库。

6.1.2 实验目的

- 强化数据库用户与角色的概念。
- 掌握 GaussDB(DWS)中的用户与角色相关操作。
- 强化数据库中的对象权限的认识，同时掌握如何操作这些权限。
- 掌握数据库系统中如何构建实际的 RBAC 模型。

6.2 实验任务

6.2.1 用户、角色与权限操作

步骤 1 连接数据库

通过管理员用户 dbadmin 登入数据库 postgres

```
source gsql_env.sh
gsql -d postgres -h 159.138.228.159 -U dbadmin -p 8000 –r
```

步骤 2 创建用户 user_1

```
postgres=> create user user_1 password 'user@123';
```

结果：CREATE ROLE

创建成功

步骤 3 查看用户

分别从 pg_roles 视图和 pg_authid 系统表中查看创建的用户。

```
postgres=> select * from pg_roles where rolname = 'user_1';
postgres=> select * from pg_authid where rolname = 'user_1';
```

步骤 4　修改用户属性

修改用户 user_1 的密码

```
postgres=> alter user user_1 IDENTIFIED BY 'Gauss@123' replace 'user@123';
```

步骤 5　删除用户

```
postgres=> drop user user_1;
```

结果：DROP ROLE

删除成功

步骤 6　创建角色

创建普通角色 role_1 和带有系统权限的角色 role_2

```
postgres=> CREATE ROLE role_1 IDENTIFIED BY "role@123";
postgres=> CREATE ROLE role_2 WITH SYSADMIN IDENTIFIED BY "role@123";
```

步骤 7　查看角色

查看角色 role_1 信息，可以发现 rolcanlogin 为 f（false），说明角色 role_1 是无法登录连接数据库的。

```
postgres=> select * from pg_roles where rolname = 'role_1';
```

结果：

```
rolname | rolsuper | rolinherit | rolcreaterole | rolcreatedb | rolcatupdate | rolcanlogin | rolreplication | rolauditadmin |
rolsystemadmin | rolconnlimit |
  rolpassword | rolvalidbegin | rolvaliduntil |    rolrespool    | rolparentid | roltabspace | rolconfig |    oid    | roluseft |
rolkind | nodegroup
---------+----------+------------+---------------+-------------+--------------+-------------+----------------+---------------+-----------
-----+--------------+
-------------+---------------+---------------+--------------+-------------+-------------+-----------+-------+----------+---------+------
-----
  role_1   | f           | t            | f                | f               | f               | f               | f
| f               | f                |              -1 |
  ********      |                   |                  | default_pool |               0 |                 |               |
24681 | f           | n          |
(1 row)
```

步骤 8　修改角色属性

修改角色 role_1 的属性，增加登录属性

```
postgres=> ALTER ROLE role_1 LOGIN;
```

步骤 9　查看修改后的角色

查看角色 role_1 修改后的属性，可以发现 rolcanlogin 由 f（false）变为 t（true）。此时重新使用 role_1 进行登录验证。

```
postgres=> select * from pg_roles where rolname = 'role_1';
```

步骤 10　删除角色

```
postgres=> DROP ROLE role_1;
postgres=> DROP ROLE role_2;
```

6.2.2 对象权限操作

步骤 1　创建用户 user_1

创建用户 user_1，密码为 user@123，并用 user_1 登录数据库。

```
postgres=> create user user_1 password 'user@123';
gsql -d postgres -h 159.138.228.159 -U user_1 -p 8000  -r
```

步骤 2　创建表 table_1

```
postgres=> create table public.table_1 (id int, name text);
```

步骤 3　插入数据

```
postgres=> insert into public.table_1 values(1, 'string1'),(2, 'string2');
```

步骤 4　查看表的权限

```
postgres=> select * from INFORMATION_SCHEMA.role_table_grants where table_name = 'table_1';
```

INFORMATION_SCHEMA.role_table_grants 视图说明

列column	含义
grantor	授权者，上述步骤中table_1是user_1创建的，所以grantor为user_1自己。
grantee	被授权者，user_1创建table_1后，自己默认的权限被授权者也是user_1自己。
table_catalog	table_1属于postgres数据库
table_schema	由于create table时，没有指定table_1的模式，所以默认属于public模式。
table_name	表名table_1
privilege_type	权限类型，表对象每种权限对应role_table_grants中的一条行记录。
is_grantable	权限是否可被授予
with_hierarchy	是否允许在表继承层级上的特定操作，GaussDB A被包括在SELECT中。

步骤 5　创建用户 user_2

创建用户 user_2，并登录数据库

```
postgres=> create user user_2 password 'user@123';
gsql -d postgres -h 159.138.228.159 -U user_1 -p 8000 –r
```

步骤 6 访问表 table_1

```
postgres=> select * from table_1;
```

结果：ERROR: permission denied for relation table_1

访问被拒绝

步骤 7 给用户 user_2 授权

user_1 登录数据库，将 table_1 表的读权限赋给 user_2 用户。

```
gsql -d postgres -h 159.138.228.159 -U user_1 -p 8000 –r
postgres=> grant select on table public.table_1 to user_2;
```

步骤 8 user_2 访问 table_1

授权后再使用 user_2 登录数据库，并访问表 table_1。

```
gsql -d postgres -h 159.138.228.159 -U user_2 -p 8000 –r
postgres=> select * from table_1;
```

结果：

```
id |   name
----+---------
  1 | string1
  2 | string2
(2 rows)
```

步骤 9 删除用户

```
postgres=> drop user user_1 cascade;
postgres=> drop user user_2;
```

6.2.3 RBAC 关系模型

说明：一张 notices 表，Author 角色具有插入权限，Reader 角色具有读取权限，Author 角色下的所有用户都具备插入权限，同理 Reader 角色下所有用户都具有读取权限。对用户的权限分配转换成为了对角色的权限分配。

步骤 1 创建角色

管理员用户 dbadmin 创建 Author 与 Reader 角色

```
postgres=> CREATE ROLE author WITH CREATEDB CREATEROLE LOGIN PASSWORD 'author@123';
postgres=> CREATE ROLE reader WITH CREATEROLE LOGIN PASSWORD 'reader@123';
```

步骤 2 创建用户

author 角色登录数据库，并创建用户 author_1 与 author_2。

```
gsql -d postgres -h 159.138.228.159 -U author -p 8000 –r
postgres=> CREATE USER author_1 PASSWORD 'author@123';
```

```
postgres=> CREATE USER author_2 PASSWORD 'author@123';
```

步骤 3　创建表

author 角色创建 notices 表

```
postgres=> CREATE TABLE notices (id int , message text);
```

步骤 4　查看表权限

此时 author 角色具有 notices 表所有权限。

```
postgres=> select * from INFORMATION_SCHEMA.role_table_grants where table_name = 'notices';
```

步骤 5　赋权

将 author 角色的权限赋给 author_1 与 author_2，author 角色将表 notices 的 select 权限赋给 reader 角色。

```
postgres=> GRANT author TO author_1;
GRANT ROLE
postgres=> GRANT author TO author_2;
GRANT ROLE
postgres=> GRANT SELECT ON notices TO reader;
GRANT
```

步骤 6　再次查看表的权限

再次查看 notices 表的权限，发现 reader 具有了 notices 表的 SELECT 权限。

```
postgres=> select * from INFORMATION_SCHEMA.role_table_grants where table_name = 'notices';
```

步骤 7　再次创建用户

使用 reader 角色创建用户 reader_1 与 reader_2，操作步骤同 author。

```
gsql -d postgres -h 159.138.228.159 -U reader -p 8000 –r
postgres=> CREATE USER reader_1 PASSWORD 'author@123';
CREATE ROLE
postgres=> CREATE USER reader_2 PASSWORD 'author@123';
CREATE ROLE
```

步骤 8　赋权

将 eader 角色的权限赋给 reader_1 与 reader_2，操作步骤同 author。

```
postgres=> GRANT reader TO reader_1;
GRANT ROLE
postgres=> GRANT reader TO reader_2;
GRANT ROLE
```

步骤 9　author_1 发布消息

```
\q
gsql -d postgres -h 159.138.228.159 -U author_1 -p 8000 –r
postgres=> insert into notices values(1, 'message1'),(2, 'message2');
INSERT 0 2
```

步骤 10 reader_1 读取消息

```
\q
gsql -d postgres -h 159.138.228.159 -U reader_1 -p 8000  －r
postgres=> select * from notices;
```

结果：

```
id | message
----+----------
   1 | message1
   2 | message2
(2 rows)
```

6.3 清理运行环境

删除所创建的角色，用户和表。

```
postgres=> drop table notices;
DROP TABLE
postgres=> drop role author;
DROP ROLE
postgres=> drop role reader;
DROP ROLE
postgres=> drop user author_1;drop user author_2;
DROP ROLE
DROP ROLE
postgres=> drop user reader_1;drop user reader_2;
DROP ROLE
DROP ROLE
```