Greenplum 权限管理

1 关于 greenplum 权限说明

1.1 概述

用户与角色在整个数据库中都是全局性的。
在安装数据库时已指定超级管理员，系统管理员，例如超级管理员:gpadmin
每个数据库的逻辑结构对象都有一个所有者,所有者默认拥有所有的权限,不需要重新赋予。
删除和任意修改它的权利不能赋予别人，为所有者固有，不能被赋予或撤销。
可以把操作该对象的权限赋予别人。
授权和撤销授权 用命令 GRANT REVOKE

1.2 赋予权限的步骤总结

权限按如下几个层次进行管理
1、首先管理赋予在用户特殊属性上的权限
2、在数据库上的权限
3、在数据库中创建 schema 的权限
4、在模式中创建数据库对象的权限，表，索引等
5、 表的增删改查的权限
6、操作表中某些字段的权限

1.3 管理赋予在用户特殊属性上的权限

1、user 的 Superuser 与 createuser 属性不能同时拥有。
2、有 superuser 属性的用户实际可以创建库和创建用户，且 nocreateuser nocreatedb 对 superuser 属性没有约束。
3、create role 创建用户，alter role 修改用户属性。删除用户 drop role，同理删除数据库是 drop database;
4、拥有资源的用户不能被 drop，提示错误。但是资源可以被 superuser drop 掉。
5、修改用户属性用 alter role

1.4 用户对数据库对象操作权限列表

对象类型	特权
表、视图、序列	SELECT INSERT UPDATE DELETE RULE ALL
外部表	SELECT RULE ALL
数据库	CONNECT CREATE TEMPORARY | TEMP ALL
函数	EXECUTE

过程语言	USAGE
方案	CREATE USAGE ALL

1.5 用户角色列表

性	描述
SUPERUSER | NOSUPERUSER	决定角色是否为一个超级用户。要创建一个新的超级用户，用户本身必须是超级用户。NOSUPERUSER 是默认值。
CREATEDB | NOCREATEDB	决定该角色是否被允许创建数据库。NOCREATEDB 是默认值。
CREATEROLE | NOCREATEROLE	决定该角色是否被允许创建和管理其他角色。NOCREATEROLE 是默认值。
INHERIT | NOINHERIT	决定一个角色是否从它的父角色继承特权。一个带有 INHERIT 属性的角色可以自动地使用授予给其所有直接父角色以及间接父角色的任何数据库特权。INHERIT 是默认值。
LOGIN | NOLOGIN	决定一个角色是否被允许登入。一个带有 LOGIN 属性的角色可以被认为是一个用户。没有这个属性的角色对于管理数据库特权有用（组）。NOLOGIN 是默认值。
CONNECTION LIMIT*connlimit*	如果角色能够登入，这指定该角色能建立多少并发连接。-1（默认）表示没有限制。
CREATEEXTTABLE | NOCREATEEXTTABLE	决定一个角色是否被允许创建外部表。NOCREATEEXTTABLE 是默认值。对于一个带有 CREATEEXTTABLE 属性的角色，默认的外部表类型是 readable，而默认的协议是 gpfdist。注意使用 file 或 execute 协议的外部表只能由超级用户创建。
PASSWORD '*password*'	设置角色的口令。如果没有计划使用口令认证则可以省略这个选项。如果没有指定口令，口令将被设置为空并且该用户的口令认证总是会失败。也可以有选择地使用 PASSWORD NULL 显式地写入一个空口令。

ENCRYPTED | UNENCRYPTED	控制新口令是否在pg_authid系统目录中存储为一个哈希字符串。如果既没有指定ENCRYPTED也没有指定UNENCRYPTED，默认行为由password_encryption配置参数决定，这个参数默认是on。

	如果提供的 *password* 字符串已经是哈希过的格式，就会原样存储，而不管指定的是 ENCRYPTED 还是 UNENCRYPTED。 有关保护登录口令的额外信息请见在 Greenplum 数据库中保护口令。
VALID UNTIL '*timestamp*'	设置一个日期和时间，在此之后该角色的口令不再有效。如果省略，则口令将会永久有效。
RESOURCE QUEUE *queue_name*	为负载管理的目的将角色分配到提及的资源队列。然后该角色发出的任何语句都服从于该资源队列的限制。注意 RESOURCE QUEUE 属性不会被继承，必须在每个用户级（LOGIN）角色上设置它。
DENY {deny_interval | deny_point}	在一个间隔期间限制访问，用日或者日和时间指定。更多信息请见基于时间的认证。

2 权限操作实例

2.1 在用户(USER)特殊属性上的权限

2.1.1 权限说明

1、user 的 Superuser 与 createuser 属性不能同时拥有。

2、有 superuser 属性的用户实际可以创建库和创建用户，且 nocreateuser nocreatedb 对 superuser 属性没有约束。
3、create role 创建用户，alter role 修改用户属性。删除用户 drop role，同理删除数据库是 drop database;
4、拥有资源的用户不能被 drop，提示错误。但是资源可以被 superuser drop 掉。
5、修改用户属性用 alter role

2.1.2 权限创建实例

2.1.2.1 创建 role

$ psql -h 192.168.***.55 -U gpmon -d china***

在管理员用户上创建以下 role
create role user1 with login password '123456';
NOTICE: resource queue required -- using default resource queue "pg_default"
CREATE ROLE

2.1.2.2 检验创建的 role

$ psql -h 192.168.***.55 -U user1 -d china***
Password for user user1:
psql (8.3.23)
Type "help" for help.

china***=>

2.1.3 撤销用户的权限

2.1.3.1 删除没有授权的账户

2.1.3.1.1 撤销权限

登录到管理员账户删除刚创建的 user1
auth_test=# drop role if exists user1;
DROP ROLE

2.1.3.1.2 查看撤销效果

$ psql -h 192.168.***.55 -U user2 -d auth_test
Password for user user2:
psql: FATAL: password authentication failed for user "user2"

2.1.3.2 删除授数据库的用户

2.1.3.2.1 撤销权限

登录到管理员用户执行删除用户，需要把 user1 下的创建的相关信息全部删除掉才可删除给用户,或者使用 cascade 强制删除
auth_test=# revoke connect on database auth_test from user1;
REVOKE

auth_test=# drop user user1;
DROP ROLE

错误信息提示
auth_test=# drop user user1;
ERROR: role "user1" cannot be dropped because some objects depend on it
DETAIL: owner of append only file visibility map pg_aoseg.pg_aovisimap_2214714
owner of append only file segment listing pg_aoseg.pg_aoseg_2214714
owner of append only table schema1.test1

2.1.3.2.2 查看效果

$ psql -h 192.168.***.55 -U user1 -d auth_test
Password for user user1:
psql: FATAL: password authentication failed for user "user1"

2.2 在 DATABASE(数据库)上的权限

2.2.1 权限说明

gpadmin 创建一个数据库，依次授予某用户 CREATE CONNECT TEMPORARY TEMP 的权限。验证各选项的作用。
创建用户 user1 ,赋予对 auth_test 数据库 CREATE 权限，则可以在 auth_test 下创建 schema;

2.2.2 权限创建实例

2.2.2.1 创建数据库

在管理员的用户下创建以下数据库
CREATE DATABASE auth_test;

2.2.2.2 把数据库的 CREARE 权限赋予用户

登录到管理员账号把权限赋予 user1
$ psql -h 192.168.***.55 -U gpmon -d auth_test

auth_test=# GRANT CREATE ON DATABASE auth_test TO user1;
GRANT

2.2.2.3 在用户下创建 SCHEMA

auth_test=> create schema schema1;
CREATE SCHEMA

2.2.2.4 在 SCHEMA 下创建表

创建 test1 并插入数据
auth_test=> create table schema1.test1(id int,name varchar)
WITH (appendonly=true, compresstype=zlib, compresslevel=5)
DISTRIBUTED BY (id);
CREATE TABLE

插入数据并查看数据
auth_test=> insert into schema1.test1(id,name) values(1,'1');
INSERT 0 1
auth_test=> select * from schema1.test1;
 id | name
----+------
  1 | 1
(1 row)

2.2.3 权限撤销实例

权限包括 CREATE,CONNECT,TEMP

2.2.3.1 撤销用户的 connect 权限

revoke connect on database auth_test from user1;
REVOKE

2.2.3.2 撤销用户的 create 权限

revoke create on database auth_test from user1;

REVOKE

2.2.3.3 查看权限撤销后的效果

auth_test=> create schema schema2;
ERROR: permission denied for database auth_test

在以上可以看出用户已失去创建 schema 的权限，但还有以下的 select 权限，同理也可以撤销其他的权限。

auth_test=> select * from schema1.test1;
 id | name
----+------
  1 | 1
(1 row)

2.2.4 总结数据库上权限

1、实际上在数据库方面控制的权限有 CREATE,CONNECT,TEMP,即使把这些权限全部取消了，仍可以有 CONNECT 和创建临时表(TEMP)的权限。
2、属性（nosuperuser,nocreatedb,nocreaterole）的用户默认情况下可以 connect 数据库。不可以创建 schema。可以创建 temporary table ，自动生成临时的 schema，在会话结束后自动销毁。可以在 public schema 中创建表。不能在 owner 为其他用户的 schema 下创建表。
3、数据库的 CREATE 权限，控制是否可以在库中创建 schema,以及是否可以在 schema 下创建表与查询表中的数据。
4、通过身份验证的用户总有 CONNECT 库的权限。即使是通过 REVOKE 撤销 CONNECT，也能正常连接数据库。

5、用户总有创建 TEMP 表的权限。即使是通过 REVOKE 撤销 TEMP，也能创建临时表。

2.3 在 SCHEMA(模式)上的权限

2.3.1 验证权限说明

创建 user2 验证对 user1 的 schema 的操作，看是否有权限，实验证明 user1 的 schema 不可分享给 user2,这也是 GP 怕多用户操作混乱

2.3.2 创建 USER2

```
# create role user2 with login password '123456';
NOTICE:    resource queue required -- using default resource queue "pg_default"
CREATE ROLE
```

2.3.3 把 USAGE(使用权限)赋予用户

登录到 user1 账号下并把使用权限赋予 user2

```
# grant usage on schema schema1 to user2;
```

2.3.4 查看权限的效果

登录到 user2 的用户下

```
$    psql -h 192.168.***.55 -U user2 -d auth_test

auth_test=> select * from schema1.test1;
ERROR:    permission denied for relation test1
```

2.3.5 把 CREATE(创建权限)赋予用户

登录到 user1 用户下执行以下命令

```
auth_test=> grant create on schema schema1 to user2;
GRANT
```

2.3.6 在用户下查询数据

在 user2 下查询 user1 的 schema 下的数据

auth_test=> select * from schema1.test1;
ERROR: permission denied for relation test1

2.3.7 总结 SCHEMA(模式)上的权限汇总

1、如果要在别人的 schema 中创建自己的表，需要用户对该 shema 有 CREATE，USAGE 权限，才可以对表和数据有足够权限。
2、用户默认无法在 owner 为别个用户的 schema 中创建表。
3、用户默认无法看到 owner 为别个用户的 schema 中的表，注意设置 search_path 。（\dt 命令查看）。
4、赋予 USAGE 权限后可以看到 owner 为别个用户的 schema 中的表，但无法在里面创建表。
5、赋予 CREATE 权限后可以在别个用户的 schema 中创建表，但如果没有 USAGE 权限，仍无法看到表，无法查询表中的数据，也无法更改表，即使 owner 也是不行。再赋予 USAGE 后可以查询自己创建的表，可以更改自己创建的表，但无法查询别人的表。
6、用户 user1 的 schema 信息无法分配给 user2 用户。

2.3.8 权限撤销实例

登录到管理员用户执行撤销权限
$ psql -h 192.168.***.55 -U gpmon -d auth_test

auth_test=# revoke select,update,delete,insert on schema1.test1 from user1;
REVOKE
auth_test=# revoke usage on schema schema1 from user1;
REVOKE
auth_test=# revoke create on schema schema1 from user1;
REVOKE

撤销时需要先从 table 到 schema

强制删除 schema
auth_test=# drop schema schema1 CASCADE;
NOTICE: drop cascades to table schema1.test1
DROP SCHEMA

2.3.9 验证撤销的权限

在 user1 上验证权限问题

auth_test=> select * from schema1.test1;
ERROR: permission denied for schema schema1

LINE 1: select * from schema1.test1;

2.3.10 把 Schema 的所属主付给制定用户

```
alter schema schema_test owner to xiaoxu;
```

schema_test ： schema 的名字
xiaoxu ： 用户的名字

2.3.11 把 schema 的所有权限给用户

```
grant all on schema schema_test to xiaoxu;
```

schema_test ： schema 的名字
xiaoxu ： 用户的名字

2.4 在 TABLE(表)上的权限

2.4.1 权限验证说明

登录到 user1 执行授权给 user2 用户的 schema1 下的某张表的查询权限，查看 user2 时候有查询的权限

2.4.2 把 SELECT(查询)权限赋予用户

登录到 user1 用户，把 schema1 下的 test1 查询权限赋予 user1
$ psql -h 192.168.***.55 -U user1 -d auth_test

auth_test=> grant select on schema1.test1 to user2;

2.4.3 查看权限效果

登录到 user2 用户，查询 user1 的 schema1 下的表
$ psql -h 192.168.***.55 -U user2 -d auth_test

auth_test=> select * from schema1.test1;
 id | name

```
----+------
   1 | 1
   2 | 2
   2 | 2
(3 rows)
```

2.4.4 赋予用户更多的权限

在用户 user1 下操作一下命令

```
$ psql -h 192.168.***.55 -U user1 -d auth_test

auth_test=> grant select,update,delete,insert on schema1.test1 to user2;
GRANT
```

2.4.5 撤销用户的权限

一下的操作全部在 user1 下操作

2.4.5.1 撤销创建权限

```
auth_test=> revoke create on schema schema1 from user2;
REVOKE
```

2.4.5.2 撤销更多的权限

```
auth_test=> revoke select,update,delete,insert on schema1.test1 from user2;
REVOKE
```

2.4.6 查看撤销后的效果

登录到 user2 用户下查询 user1 下的数据

```
$ psql -h 192.168.***.55 -U user2 -d auth_test

auth_test=> select * from schema1.test1;
ERROR:    permission denied for relation test1
```

3 权限参数详解

```
GRANT { { SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES | TRIGGER }
    [, ...] | ALL [ PRIVILEGES ] }
    ON { [ TABLE ] table_name [, ...]
         | ALL TABLES IN SCHEMA schema_name [, ...] }
    TO role_specification [, ...] [ WITH GRANT OPTION ]

GRANT { { SELECT | INSERT | UPDATE | REFERENCES } ( column_name [, ...] )
    [, ...] | ALL [ PRIVILEGES ] ( column_name [, ...] ) }
    ON [ TABLE ] table_name [, ...]
    TO role_specification [, ...] [ WITH GRANT OPTION ]

GRANT { { USAGE | SELECT | UPDATE }
    [, ...] | ALL [ PRIVILEGES ] }
    ON { SEQUENCE sequence_name [, ...]
         | ALL SEQUENCES IN SCHEMA schema_name [, ...] }
    TO role_specification [, ...] [ WITH GRANT OPTION ]

GRANT { { CREATE | CONNECT | TEMPORARY | TEMP } [, ...] | ALL [ PRIVILEGES ] }
    ON DATABASE database_name [, ...]
    TO role_specification [, ...] [ WITH GRANT OPTION ]

GRANT { USAGE | ALL [ PRIVILEGES ] }
    ON DOMAIN domain_name [, ...]
    TO role_specification [, ...] [ WITH GRANT OPTION ]

GRANT { USAGE | ALL [ PRIVILEGES ] }
    ON FOREIGN DATA WRAPPER fdw_name [, ...]
    TO role_specification [, ...] [ WITH GRANT OPTION ]

GRANT { USAGE | ALL [ PRIVILEGES ] }
    ON FOREIGN SERVER server_name [, ...]
    TO role_specification [, ...] [ WITH GRANT OPTION ]

GRANT { EXECUTE | ALL [ PRIVILEGES ] }
    ON { FUNCTION function_name ( [ [ argmode ] [ arg_name ] arg_type [, ...] ] ) [, ...]
         | ALL FUNCTIONS IN SCHEMA schema_name [, ...] }
    TO role_specification [, ...] [ WITH GRANT OPTION ]

GRANT { USAGE | ALL [ PRIVILEGES ] }
    ON LANGUAGE lang_name [, ...]
    TO role_specification [, ...] [ WITH GRANT OPTION ]
```

```
GRANT { { SELECT | UPDATE } [, ...] | ALL [ PRIVILEGES ] }
    ON LARGE OBJECT loid [, ...]
    TO role_specification [, ...] [ WITH GRANT OPTION ]

GRANT { { CREATE | USAGE } [, ...] | ALL [ PRIVILEGES ] }
    ON SCHEMA schema_name [, ...]
    TO role_specification [, ...] [ WITH GRANT OPTION ]

GRANT { CREATE | ALL [ PRIVILEGES ] }
    ON TABLESPACE tablespace_name [, ...]
    TO role_specification [, ...] [ WITH GRANT OPTION ]

GRANT { USAGE | ALL [ PRIVILEGES ] }
    ON TYPE type_name [, ...]
    TO role_specification [, ...] [ WITH GRANT OPTION ]

where role_specification can be:

    [ GROUP ] role_name
  | PUBLIC
  | CURRENT_USER
  | SESSION_USER

GRANT role_name [, ...] TO role_name [, ...] [ WITH ADMIN OPTION ]
```

4 用户批量赋权脚本

对单个表授权，可以使用 GRANT ALL ON TABLE {tablename} TO {username}，如果需要批量操作某个 schema 下所有表请使用以下函数进行赋权。

4.1 创建函数

```
create or replace function grant_on_all_tables(schema text, usr text)
returns setof text as $$
declare
   r record ;
   grantstmt text;
begin
   for r in select * from pg_class c, pg_namespace nsp
       where c.relnamespace = nsp.oid AND c.relkind='r' AND nspname = schema
```

```
   loop
     grantstmt = 'GRANT SELECT,UPDATE,DELETE,INSERT ON "' || quote_ident(schema) ||
'"."' ||
quote_ident(r.relname) || '" to "' || quote_ident(usr) || '"';
     EXECUTE grantstmt;
     return next grantstmt;
   end loop;
end;
$$ language plpgsql;
```

schema text : 需要授权的 schema 名称。
usr text : 需要授权的 role 名称。
然后代码会遍历参数 schema 下的所有表，轮询的去做授权操作。

4.2 调用函数

```
select grant_on_all_tables('schema_name','user_name');
```

schema text : 需要授权的 schema 名称。
usr text : 需要授权的 role 名称。

5 创建用户与修改用户密码方法

5.1 使用 role 方式创建

在管理员用户上创建以下 role

```
# create role user1 with login password '123456';
NOTICE:  resource queue required -- using default resource queue "pg_default"
CREATE ROLE
```

给用户限制连接数

```
alter user user1 CONNECTION LIMIT 20;
```

user1:用户信息

5.2 使用 user 方式创建

```
create user user_name;
```

```
alter user user_name with password '';

alter user user_namewith CONNECTION LIMIT  20;#连接数限制
```

5.3 修改用户密码

```
alter role username with password 'password';

username : 用户信息
'password' : 用户密码
```

6 函数权限管理

6.1 通用语句

```
GRANT { EXECUTE | ALL [ PRIVILEGES ] }
    ON FUNCTION funcname ([type, ...]) [, ...]
    TO { username | GROUP groupname | PUBLIC } [, ...] [ WITH GRANT OPTION ]
```

6.2 付给用户权限

```
grant all on FUNCTION ods.sp_t_job_employ_jobinfo_etl(_date text) to jiang****;
```

7 生成自动赋权的语句

以下语句主要是查询出指定 schema 下的表在用户 rolename 没有权限的信息，会生成 grant 的授权语句，如果想自动赋权请定时刷新即可。

```
select 'grant select on'||nspname||'.'||relname||' to rolename ;' as grantsql
from pg_class a,pg_namespace b where relname not like '%_1—prt%'
and relkind = 'r'
and has_table_privilege('rolename',a.oid,'select')='f' and a.relnamespace=b.oid
and nspname in ('pg_catalog','schemaname') and nspname not like '%pg_%' and relname
like 'tablename';
```

或者使用以下语句

```
select 'grant '||privilege_type||' on  schemaname.tablename  to '||grantee||';'
from
information_schema.table_privileges
where table_schema='schemaname' and table_name='tablename'
group by grantee,privilege_type;
```

rolename: 角色的名字
schemaname : 制定需要查看 schema 的信息
tablename : 表的匹配信息

8 pg_hba.conf 客户端登录权限

通过修改 pg_hba.conf 文件可以控制客户端登录数据库的权限

```
vi $MASTER_DATA_DIRECTORY/pg_hba.conf
```

pg_hba.conf 文件配置参数为：

```
# local      DATABASE  USER  METHOD  [OPTIONS]
# host       DATABASE  USER  CIDR-ADDRESS  METHOD  [OPTIONS]
# hostssl    DATABASE  USER  CIDR-ADDRESS  METHOD  [OPTIONS]
# hostnossl  DATABASE  USER  CIDR-ADDRESS  METHOD  [OPTIONS]
```

例如:

```
host     all    all 0.0.0.0/0   md5
```

host : 链接 host
all : 表示链接所有的数据库
all : 表示链接的所有的用户
0.0.0.0/0 : 允许所有的 IP 登录数据库
md5 : 允许登录的方式是 md5 加密方式

例如:

```
host     test_db  all 192.168.253.3/32  md5
```

表示 192.168.253.3 地址的所有用户通过 md5 加密方式登录 test_db 数据库

使用 gpstop -u 生效

9 删除集群中赋权的用户

9.1 撤销用户在数据库上的权限

-- 移除数据库的权限
revoke all on database databasename from username;

databasename ：数据库的名字
username ：角色的名字

9.2 撤销用户在 schema 上的权限

-- 移除 schema 的权限
revoke all on schema schema1,schema2 from username;

schema1,schema2 : schema 的集合，以逗号分开
username ：角色的名字

9.3 撤销用户在 table 上的权限

select 'revoke all on '||table_schema||'.'||table_name||' from username cascade; ' from
information_schema.table_privileges
where grantee='username';

username ：角色的名字

用此语句查询出 revoke 的语句，去执行即可

9.4 撤销用户在 function 上的权限

-- 查询该用户的所属的函数
select * from information_schema.routine_privileges where grantee='username';

-- 移除权限
revoke all on function schemaname.functionname from username;

username ：角色的名字

使用第一个语句把该角色关于函数的语句查询出来，使用第二个语句撤销语句即可

9.5 删除角色

drop role if exists username;

username ：角色的名字