SQL/DDL – Gerenciamento de Usuários e de Privilégios

Prof. Dr. Ives Renê V. Pola
ivesr@utfpr.edu.br

Departamento Acadêmico de Informática – DAINF
UTFPR – Pato Branco DAINF
UTFPR
Pato Branco - PR

Apresentação dos conceitos de como gerenciar usuários e privilégios de acesso, e dos comandos correspondentes na Linguagem SQL para o SGBD PostgreSQL.

Roteiro

1. Permissões de Acesso no PostgreSQL
2. Ações alternativas - RULES
3. CREATE SCHEMA
4. CREATE SEQUENCE

Permissões de acesso no PostgreSQL

PostgreSQL gerencia permissões de acesso na base da dados através do conceito de Roles (papéis).

Roles

Um role pode ser visto como um **usuário ou um grupo de usuários**, dependendo de como é definido.

- Roles podem possuir objetos (ex: tabelas, funções) e definem privilégios que controlam quem pode ter acesso a estes objetos.
- É possível ainda transferir privilégios de um role para outro.
- Em versões anteriores à 8.1, eram definidos usuários e grupos. Mas agora ambos são Roles.
- Um role pode atuar como um usuário, um grupo, ou ambos.

Permissões de acesso no PostgreSQL

- Roles são globais dentro de um database cluster (e não individual por database).
- A criação e remoção de Roles são feitas em SQL como:

Role – Criação

```
CREATE ROLE nome [ [ WITH ] option [ ... ] ];
```

Role – Remoção

```
DROP ROLE nome;
```

Permissões de acesso no PostgreSQL

- Por conveniência, existem os comandos de criação de usuários.
- A criação e remoção de usuários são feitas em SQL como:

Usuário– Criação

```
create user nome;
```

Usuário – Remoção

```
drop user nome;
```

- Para acessar os roles existentes, é necessário acessar o catálogo:

```
SELECT * FROM pg_roles;
```

- Quando instalado, é criado por default um role chamado postgres.
- Assim, para criar outros roles, é necessário logar com este papel (superusuário).

Role – Atributos

- Um Role terá um conjunto de atributos que vão definir alguns privilégios no sistema.
- O privilégio mais básico possível é a permissão para logar no sistema.
- Isso pode ser feito de duas maneiras:

Privilégio de login

```
CREATE ROLE nome LOGIN;
CREATE USER nome;
```

- Veja que o create user é equilavente ao create role, mas assume o privilégio de LOGIN por default.

Role – Atributos

Outros atributos podem ser definidos para definir permissões:

- SUPERUSER

  Um papel de superusuário tem todas as permissões, menos a de login (cuidado).
- CREATEDB

  Permissão para criar databases.
- CREATEROLE

  Permissão para criar outros papéis.
- REPLICATION

  Permissão para iniciar replicação por streaming.
- PASSWORD

  Define uma senha para autenticação. Por exemplo: CREATE ROLE nome PASSWORD 'senha'.

Exemplos de criação de usuários e papéis

- Criar um usuário zeca com uma senha.
  CREATE USER zeca WITH PASSWORD 'jw8s0F4';
- Criar um papel miriam que pode logar e tenha sua senha definida e válida até uma data.
  CREATE ROLE miriam WITH LOGIN PASSWORD 'jw8s0F4' VALID UNTIL '2005-01-01';
- Criar um papel admin que pode criar databases e atribuir papéis.
  CREATE ROLE admin WITH CREATEDB CREATEROLE;

Permissões a Objetos

- Quando um objeto é criado, ele pertence ao papel de quem o criou, ou seja, o usuário que executou o CREATE.
- Para permitir acesso de outros usuários ou grupos de usuários, permissões devem ser atribuídas.
- Permissões são dadas através do comando GRANT, por exemplo:

Exemplo de GRANT

```
GRANT UPDATE ON vendas TO zeca;
```

Concede permissão para o zeca atualizar a tabela vendas.

- GRANT ALL concede todos privilégios referentes ao objeto.

Permissões a Objetos

- O comando GRANT possui duas variantes básicas.
  1. Concede privilégios a um objeto (tabela, coluna, view, sequence, função, esquema,...)
  2. Concede os privilégios a um outro role.

GRANT – sintaxe geral

```
GRANT permissão ON objeto TO role_spec;
```

- Se role_spec for a palavra PUBLIC, isso indica que o privilégio é passado a todos roles, inclusive aqueles criados depois !

Permissões Possíveis

- Vamos ver alguns privilégios.

SELECT, UPDATE, INSERT, DELETE

```
GRANT SELECT ON minha_tabela TO PUBLIC;
GRANT SELECT, UPDATE, DELETE ON minha_tabela TO juca;
GRANT SELECT (col1), UPDATE (col1) ON minha_tabela TO zeca;
```

Repassar Privilégios

```
GRANT admins TO juca;
```

Permissões Possíveis

- Vamos ver alguns privilégios.

TRUNCATE

GRANT TRUNCATE ON tab1 TO zeca;

REFERENCES

Permite criar restrições de foreign key. Precisa ter este privilégio em ambas colunas referenciadas (ou nas duas tabelas).
GRANT REFERENCES ON tab1, tab2 TO desenvolvedores;

TRIGGER

Permite criar triggers na tabela especificada.
GRANT TRIGGER ON tab1 TO desenvolvedores;

Permissões Possíveis

- Vamos ver alguns privilégios.

TEMPORARY

Permite criar tabelas temporárias na database especificada.
GRANT TEMPORARY ON DATABASE nome_database TO desenvolvedores;

EXECUTE

GRANT EXECUTE ON FUNCTION nome_funcao TO desenvolvedores;

RULES

- Não devemos confundir ROLE com RULE.
- ROLE é um papel, com privilégios vistos até então.
- RULE são ações (regras) associadas aos operadores.

RULE

```
CREATE [OR REPLACE] RULE nome AS
ON evento TO tabela [WHERE condição]
DO [ALSO | INSTEAD]  NOTHING | comando
```

evento: SELECT, INSERT, UPDATE ou DELETE

- ALSO: Ação é executada após operação, em adicional.
- INSTEAD: Ação é executada no lugar da operação.

RULES – Exemplos

INSTEAD

–RULES com INSTEAD são usadas para implementar visões (views)

```
CREATE RULE "_RETURN" AS
ON SELECT TO minha_visão
DO INSTEAD SELECT * from minha_tabela;
```

ALSO

```
CREATE RULE "replica" AS
ON INSERT TO aluno
DO ALSO INSERT INTO aluno_log VALUES (NEW.RA, NEW.nome,
NEW.idade, current_user, current_timestamp);
```

aluno_log (ra integer, nome varchar, idade integer, log_quem varchar, log_quando timestamp);

RULES – Exemplos

- Uma maneira de proteger as relações temporariamente para alguma manuntenção é usar RULES.

NOTHING

```
CREATE RULE aluno_ins AS ON INSERT TO aluno
DO INSTEAD NOTHING;

CREATE RULE aluno_upd AS ON UPDATE TO aluno
DO INSTEAD NOTHING;

CREATE RULE aluno_del AS ON DELETE TO aluno
DO INSTEAD NOTHING;
```

SCHEMAS

- Um esquema (SCHEMA) pode ser visto como um namespace. Um qualificador dentro da base de dados para referenciar tabelas.
- o nome do esquema deve ser diferente dos existentes na base de dados.
- Desta forma nomes de objetos podem ser os mesmos, desde que estejam em esquemas diferentes.
- Um exemplo pode ser visto a seguir:

```
CREATE SCHEMA IF NOT EXISTS empresa AUTHORIZATION zeca;
```

- Deste modo, uma tabela pode ser referenciada como: empresa.aluno.
- E criada como:

```
CREATE TABLE
empresa.vendas(ID integer, data date, produto integer);
```

SEQUENCE

- Sequências geram valores sequenciais definidos. Por exemplo, o comando:

```
CREATE SEQUENCE id_seq
START 1
INCREMENT 1
NO MAXVALUE
CACHE 1;
```

- Cria uma sequência numérica, iniciando em 1, incrementando em 1, sem valor máximo, pré-calculadas 1 valor no cache.
- Para utilizar, em um comando insert:
  `INSERT INTO tabela VALUES (nextval('id_seq'),.....);`

Campos auto-increment

- É possível definir campos que incrementam automaticamente.
- Basta definir os tipos de dados:
  1. **smallserial**: 1 a 32767
  2. **serial**: 1 a 2147483647
  3. **bigserial**: 1 a 9223372036854775807

  - Por exemplo:
    `CREATE TABLE venda ( ID serial, valor numeric, produto integer);`
  - A inserção neste caso deve ser feita:
    `INSERT INTO venda (valor, produto) VALUES (500, 32);`
  - Sequência exata não garantida, pois transações podem abortar, pois isso é implementado como sequences no sistema.

Roteiro

SQL/DDL – Gerenciamento de Usuários e de Privilégios

Prof. Dr. Ives Renê V. Pola
ivesr@utfpr.edu.br

Departamento Acadêmico de Informática – DAINF
UTFPR – Pato Branco DAINF
UTFPR
Pato Branco - PR

FIM