ESCOLA SUPERIOR DE REDES RNP

Administração de Bancos de Dados

Aluno: **Diego Vitor Soares dos Santos**

Cod. Turma: **DES11**

Data: **18 de novembro de 2025**

Módulo 3 – Org. lógica e física dos dados

⚙ Ambiente de Teste

- **Virtualização:** Vagrant + VirtualBox
- **Sistema Operacional:** CentOS Stream 9 (Kernel 5.x)
- **PostgreSQL:** 17.2 (estável)
- **Fonte oficial:** https://ftp.postgresql.org/pub/source/v17.2/postgresql-17.2.tar.gz

Atividade 3.a – Criar Base de Dados

1. **Conecte-se ao PostgreSQL:**

```
$ sudo su - postgres
$ psql
```

2. **Crie a base `curso`:**

```
postgres=# CREATE DATABASE curso;
```

3. **Liste as bases de dados existentes:**

```
postgres=# \l
```

Resultado Obtido

```
[postgres@vm02 ~]$ psql
psql (17.2)
Type "help" for help.

postgres=# CREATE DATABASE curso;
CREATE DATABASE

postgres=# \l

   Name    | Owner | Encoding | Locale Provider |   Collate   |    Ctype    | Locale |
Rules |   Access privile>
-----------+----------+----------+-----------------+-------------+-------------+--------+
-----+----------------->
 curso     | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |        |
|
 postgres  | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |        |
|
 template0 | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |        |
| =c/postgres     >
           |          |          |                 |             |             |        |
| postgres=CTc/pos>
 template1 | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |        |
| =c/postgres     >
```

```
                  |          |          |                  |              |              |
| postgres=CTc/pos>
        (4 rows)
```

Atividade 3.b – Criar schemas

Objetivo: Organizar a base de dados logicamente utilizando schemas.

1. **Conecte-se na base curso:**

   ```
   postgres=# \c curso
   curso=#
   ```

   Ou, de forma equivalente:

   ```
   $ psql -U postgres -d curso
   curso=#
   ```

2. **Crie os schemas:**

   ```
   curso=# CREATE SCHEMA vendas;
   curso=# CREATE SCHEMA estoque;
   ```

3. **Liste os schemas existentes:**

   ```
   curso=# \dn
   ```

Resultado Obtido

```
[postgres@vm02 ~]$ psql -U postgres -d curso
psql (17.2)
Type "help" for help.

curso=# CREATE SCHEMA vendas;
CREATE SCHEMA
curso=# CREATE SCHEMA estoque;
CREATE SCHEMA
curso=# \dn
       List of schemas
  Name   |       Owner
---------+-------------------
 estoque | postgres
 public  | pg_database_owner
 vendas  | postgres
(3 rows)

curso=#
```

Atividade 3.c – Criar tablespace

Objetivo: Criar uma área de armazenamento físico fora do diretório padrão (`pg_default`).

1. **Crie o diretório físico (No Shell do S.O.):**

   ```
   $ su - aluno
   $ sudo mkdir -p /db/data2
   $ sudo chown postgres:postgres /db/data2
   ```

2. **Crie o tablespace (No psql):**

```
curso=# CREATE TABLESPACE tbs_dados2 LOCATION '/db/data2';
```

3. **Crie a tabela e mova para o tablespace:** Primeiro, foi criado a tabela `estoque`. Em seguida, utilizou-se o `ALTER TABLE` para movê-la fisicamente.

```
curso=# CREATE TABLE estoque.produto (id int, nome varchar(50));
curso=# ALTER TABLE estoque.produto SET TABLESPACE tbs_dados2;
```

Resultado Obtido

```
[postgres@vm02 ~]$ su aluno
Password:
[aluno@vm02 postgres]$ sudo mkdir -p /db/data2
[aluno@vm02 postgres]$ sudo chown postgres:postgres /db/data2
[aluno@vm02 postgres]$ exit
exit
[postgres@vm02 ~]$ psql -U postgres -d curso
psql (17.2)
Type "help" for help.

curso=# CREATE TABLESPACE tbs_dados2 LOCATION '/db/data2';
CREATE TABLESPACE
curso=# CREATE TABLE estoque.produto (id int, nome varchar(50));
CREATE TABLE
curso=# ALTER TABLE estoque.produto SET TABLESPACE tbs_dados2;
ALTER TABLE
curso=# SELECT pg_relation_filepath('estoque.produto');
            pg_relation_filepath
-----------------------------------------------
 pg_tblspc/16391/PG_17_202406281/16388/16395
(1 row)

curso=#
```

Situação do diretório `/db/data2` após a tabela estoque.produto ser movida:

```
[root@vm02 data]# tree -d /db/data2/
/db/data2/
└── PG_17_202406281
    └── 16388

2 directories
[root@vm02 data]#
```

Atividade 3.d – SEARCH_PATH

Objetivo: Entender a precedência de resolução de nomes de objetos.

1. **Crie a tabela no schema vendas:**

```
curso=# CREATE TABLE vendas.produto (id int, nome varchar(50));
```

2. **Defina o search_path:**

```
curso=# SET search_path = vendas, estoque;
```

3. **Insira dados (Caminho Vendas):**

```
curso=# INSERT INTO produto VALUES(10, 'margarina');
```

4. **Redefina o search_path e insira (Caminho Estoque):**

```
curso=# SET search_path = estoque;
curso=# INSERT INTO produto VALUES(10, 'margarina');
```

5. **Consulte os dados:**

```
curso=# SELECT * FROM vendas.produto;
curso=# SELECT * FROM estoque.produto;
```

Resultado Obtido

```
curso=# CREATE TABLE vendas.produto (id int, nome varchar(50));
CREATE TABLE
curso=# SET search_path = vendas, estoque;
SET
curso=# INSERT INTO produto VALUES(10, 'margarina');
INSERT 0 1
curso=# SET search_path = estoque;
SET
curso=# INSERT INTO produto VALUES(10, 'margarina');
INSERT 0 1
curso=# SELECT * FROM vendas.produto;
 id |   nome
----+-----------
 10 | margarina
(1 row)

curso=#  SELECT * FROM estoque.produto;
 id |   nome
----+-----------
 10 | margarina
(1 row)

curso=#
```

Análise dos resultados

Podemos observar na primeira inserção que:

1. O PostgreSQL procura a tabela `produto` seguindo a ordem definida no `search_path`.
2. A primeira ocorrência encontrada foi **vendas.produto**.
3. Por isso, o registro foi inserido na tabela **vendas.produto**, mesmo que exista uma tabela com o mesmo nome em `estoque`.

Já na segunda inserção podemos observar que:

1. Agora o `search_path` contém somente o schema **estoque**.
2. O PostgreSQL procurou a tabela `produto` apenas dentro desse schema.
3. Assim, o registro foi inserido na tabela **estoque.produto**.

Conclusão: A ordem do `search_path` determina a prioridade dos schemas na resolução dos objetos. Atera-lo muda diretamente para qual schema comandos SQL implícitos são direcionados.

Atividade 3.e – Localizar dados na estrutura de diretórios

Objetivo: - Acessar e listar o conteúdo do diretório de tablespaces; - Encontrar os arquivos da tabela estoque.produto nos diretórios;

1. **Acessar e listar o conteúdo do diretório de tablespaces;**

```
[root@vm02 16388]# ls -l $PGDATA/pg_tblspc
total 0
lrwxrwxrwx. 1 postgres postgres 9 Nov 18 18:09 16391 -> /db/data2
[root@vm02 16388]# tree -d -L 4 /db/data2
/db/data2
└── PG_17_202406281
    └── 16388

2 directories
[root@vm02 16388]# ls -lh /db/data2/PG_17_202406281/16388/16395
-rw-------. 1 postgres postgres 8.0K Nov 18 20:08 /db/data2/PG_17_202406281/16388/163
[root@vm02 16388]#
```

2. **Encontrar os arquivos da tabela estoque.produto nos diretórios**

```
curso=# SELECT oid, datname FROM pg_database WHERE datname = 'curso';
oid   | datname
-------+---------
16388 | curso
(1 row)
```

> Portanto, 16388 = OID do banco “curso”.

```
[root@vm02 16388]# ls -l $PGDATA/pg_tblspc
total 0
lrwxrwxrwx. 1 postgres postgres 9 Nov 18 18:09 16391 -> /db/data2
```

> Portanto, 16391 = OID do tablespace que aponta para /db/data2

```
curso=# SELECT c.oid, n.nspname, c.relname, c.relfilenode, pg_relation_filepath(c.oid
filepath
FROM pg_class c
JOIN pg_namespace n ON n.oid = c.relnamespace
WHERE c.relname = 'produto' AND c.relkind = 'r';
oid   | nspname | relname | relfilenode |                  filepath
-------+---------+---------+-------------+-------------------------------------------
16399 | vendas  | produto |       16399 | base/16388/16399
16392 | estoque | produto |       16395 | pg_tblspc/16391/PG_17_202406281/16388/16395
(2 rows)

curso=#
```

> O caminho indica que a tabela está armazenada no tablespace cujo diretório real é: /db/data2/PG_17_202406281/16388/16395

Atividade 3.f – Executar Script na Base curso

Execute o script para criar e carregar tabelas na base **curso**:

```
su aluno
sudo mkdir -p /curso/scripts

# Ajustar permissões
sudo chown postgres:postgres /curso/scripts
sudo chmod 755 /curso/scripts
exit
```

> Criação de um arquivo sql para criar uma nova tabela chamada cliente no schema vendas

```
cat << 'EOF' > /curso/scripts/curso.sql
CREATE SCHEMA IF NOT EXISTS vendas;

CREATE TABLE IF NOT EXISTS vendas.cliente (
    id_cliente SERIAL PRIMARY KEY,
    nome       VARCHAR(100) NOT NULL,
    cpf        VARCHAR(14) UNIQUE,
    email      VARCHAR(150),
    telefone   VARCHAR(20),
    endereco   VARCHAR(200)
);
EOF
```

Resultado Obtido

```
                                                                                Table "vendas.cli
           Column    |          Type          | Collation | Nullable |                      Defaul
| Sto>
        ------------+------------------------+-----------+----------+-----------------------------
------------------+---->
         id_cliente | integer                |           | not null |
nextval('vendas.cliente_id_cliente_seq'::regclass) | pla>
         nome       | character varying(100) |           | not null |
| ext>
         cpf        | character varying(14)  |           |          |
| ext>
         email      | character varying(150) |           |          |
| ext>
         telefone   | character varying(20)  |           |          |
| ext>
         endereco   | character varying(200) |           |          |
| ext>
        Indexes:
            "cliente_pkey" PRIMARY KEY, btree (id_cliente)
            "cliente_cpf_key" UNIQUE CONSTRAINT, btree (cpf)
        Access method: heap
```

Atividade 3.g – Consultar catálogo para listar as tabelas

Objetivo: - Ecrever uma query que consulte a tabela do catálogo pg_class para listar apenas as tabelas; - Escrever outra query para listar apenas as tabelas que possuem índice.

1. **Listar apenas tabelas (`relkind = 'r'`):**

```
curso=# SELECT relname
        FROM pg_class
        WHERE relkind = 'r'
        AND relname NOT LIKE 'pg_%'
        AND relname NOT LIKE 'sql_%';
relname
---------
produto
produto
cliente
(3 rows)

curso=#
```

2. **Listar tabelas que possuem índice (`relhasindex = 't'`):**

```
curso=# SELECT relname
        FROM pg_class
        WHERE relkind = 'r'
        AND relhasindex = 't'
        AND relname NOT LIKE 'pg_%';
relname
---------
cliente
(1 row)

curso=#
```

Atividade 3.h – Listar as visões

Objetivo: Escrever uma query que liste as visões existentes.

1. **Query no catálogo (`relkind = 'v'`):**

   Exportando o resultado para um arquivo:

```
curso=# \copy (
   SELECT relname
   FROM pg_class
   WHERE relkind = 'v'
      AND relname NOT LIKE 'pg_%'
      ) TO '/tmp/views.txt';
COPY 65
curso=# /tmp/views.txt
```

Resultado Obtido

```
[aluno@vm02 postgres]$ cat /tmp/views.txt
column_column_usage
information_schema_catalog_name
check_constraints
applicable_roles
administrable_role_authorizations
```

```
attributes
collations
character_sets
check_constraint_routine_usage
column_privileges
collation_character_set_applicability
column_domain_usage
column_udt_usage
columns
constraint_column_usage
constraint_table_usage
domain_constraints
routine_table_usage
domain_udt_usage
domains
enabled_roles
routines
key_column_usage
parameters
referential_constraints
schemata
role_column_grants
routine_column_usage
routine_privileges
sequences
role_routine_grants
routine_routine_usage
routine_sequence_usage
role_table_grants
table_privileges
table_constraints
transforms
tables
triggered_update_columns
triggers
udt_privileges
_pg_foreign_data_wrappers
role_udt_grants
usage_privileges
foreign_tables
role_usage_grants
foreign_data_wrapper_options
user_defined_types
view_column_usage
view_routine_usage
foreign_data_wrappers
view_table_usage
views
_pg_foreign_servers
data_type_privileges
element_types
_pg_foreign_table_columns
_pg_user_mappings
column_options
foreign_server_options
foreign_servers
_pg_foreign_tables
foreign_table_options
user_mapping_options
user_mappings
```

Projetos

- Admin Banco de Dados DES11: Scripts, configurações e atividades relacionadas à administração de banco de dados.

Referências (Material do Curso)

- ESCOLA SUPERIOR DE REDES (RNP). **Administração de Banco de Dados DES11: Capítulo 3 - Organização Lógica e Física do PostgreSQL**. Material do curso DES11. (Arquivo: `DES11-Mod03-v02_24.pdf`).
- Hans-Jürgen Schönig (Packt). **Mastering PostgreSQL 17**. Elevate your database skills with advanced deployment, optimization, and security strategies (6th Edition).