**CONCEITOS E MELHORES PRÁTICAS COM BANCOS DE DADOS POSTGRESQL.**

**Introdução ao banco de dados PostgreSQL.**

Parte 1: Fundamentos de banco de dados.

**Dados** são valores brutos, fatos brutos, observações documentadas, registros soltos, que são recolhidos e armazenados sem sofrer qualquer tipo de tratamento.

**Informações** são estruturações de dados, organização de dados. Conjunto de dados relacionados entre si que geram valor, que criam sentidos aos dados. Material de conhecimento.

Parte 2: Modelo relacional.

Modelar significa criar o modelo e explicará as características de funcionamento ou comportamento de um programa/software. O modelo de dados é uma ferramenta que permite como uma estrutura de dados está organizada e se estrutura.

Definição: modelo mais comum é o modelo relacional, que classifica e organiza as informações em tabelas com linhas e colunas. As linhas, ou tuplas, são os dados organizados, são os valores das tabelas, e as colunas são os atributos destes dados.

**Tabelas** são conjuntos de dados dispostos em colunas e linhas referentes a um objetivo comum. As colunas são consideradas como ‘campos da tabela’, como atributos da tabela. As linhas de uma tabela são chamadas também de tuplas, e é onde estão contidos os valores, os dados.

O que pode ser definido como tabelas?

- São coisas tangíveis, como elementos físicos, a exemplo do carro, produto e animal.

- São funções, como perfis de usuários e status de compra.

- São eventos ou ocorrências, como produtos de um pedido e histórico de dados.

Colunas importantes:

**- Chave primária / Primary Key / PK:** é o conjunto de um ou mais campos que nunca se repetem. Identidade da tabela. São utilizados como índice de referência na criação de relacionamento entre tabelas.

**- Chave estrangeiro / Foreign Key / FK:** valor de referência a uma PK de outra tabela ou da mesma tabela para criar um relacionamento entre tabelas.

**Sistema de gerenciamento de banco de dados** ou também chamado de sistema de gestão de base de dados ou SGBD. É um conjunto de programas ou softwares responsáveis pelo gerenciamento de um banco de dados. Programas que facilitam a administração de um banco de dados.

Parte 3: Introdução ao PostgreSQL

É um sistema de gerenciamento de banco de dados objeto relacional. Trata-se de um SGBD open source.

O PostgreSQL tem uma arquitetura multiprocessos, ou seja, vários processos serão executados no servidor, realizando diversas tarefas. O processo principal é chamado de postmaster e as cópias desses postmaster são chamados de childs. Tudo isso é reservado em um storage.

Modelo cliente/servidor: o PostgreSQL tem processos que acontecem somente na máquina cliente e processes que acontecem somente no servidor.

**Principais características do PostgreSQL**

- Open source

- Point in time recovery

- Linguagem procedural com suporte a várias linguagens de programação (perl, python, etc)

- Views, functions, procedures, triggers

- Consultas complexas e common table expressions (CTE)

- Suporte a dados geográficos (PostGIS)

- Controle de concorrência mult-versão.

**OBJETOS E TIPOS DE DADOS DE POSTGRESQL**

Parte 1: O arquivo postgresql.conf

É o arquivo onde estão definidas e armazenadas todas as configurações do servidor PostgreSQL. Alguns parâmetros só podem ser alterados com uma reinicialização de banco de dados. A view pg\_settings, acessada por dentro do banco de dados, guarda todas as configurações.

**Configurações de conexão:**

LISTEN\_ADDRESSES – Endereço(s) TCP/IP das interfaces que o servidor PostgreSQL vai escutar/liberar conexões.

PORT – A porta TCP que o servidor PostgreSQL vai ouvir. O padrão é 5432.

MAX\_CONNECTIONS – número máximo de conexões simultâneas no servidor PostgreSQL

SUPERSUSER\_RESERVED\_CONNECTIONS – número de conexões (slots) reservadas para conexões ao banco de dados de super usuários.

**Configurações de autenticação:**

AUTHENTICATION\_TIMEOUT – tempo máximo de segundos par ao cliente conseguir uma conexão com o servidor.

PASSWORD\_ENCRYPTION – algoritmo de criptografia das senhas dos novos usuários criados no banco de dados.

SSL – habilita a conexão criptografada por SSL. Somente se o PostgreSQL foi compilado com suporte SSL.

**Configurações de memória:**

SHARED\_BUFFERS – tamanho da memória compartilhada do servidor PostgreSQL para cache/buffer de tabelas, índices e demais relações. O valor padrão é de 25% do valor da sua memória.

WORK\_MEM – tamanho da memória para operações de agrupamento e ordenação (ORDER BY, DISTINCT, MERGE JOINS)

MAINTENANCE\_WORK\_MEM – tamanho da memória para operações como VACUUM, INDEX, ALTER TABLE.

Parte 2: o arquivo pg\_hba.conf

É o arquivo responsável pelo controle de autenticação dos usuários no servidor PostgreSQL. É o que permite o acesso de usuário a acessar determinado dado do banco de dados, a partir de determinado IP.

**Métodos de autenticação**:

* TRUST (conexão sem requisição de senha)
* REJECT (rejeitar conexões)
* MD5 (criptografia MD5)
* PASSWORD (senha sem criptografia)
* GSS (generic security servisse application program interface)
* SSPI (security support provider interface – somente para Windows)
* KRB5 (kerberos v5)
* IDENT (utiliza o usuário do sistema operacional do cliente via ident server)
* PEER (utiliza o usuário do sistema operacional do cliente)
* LDAP (ldap server)
* RADIUS (radius server)
* CERT (autenticação via certificado ssl do cliente)
* PAM (pluggable authentication modules. O usuário precisa estar no banco)

Parte 3: o arquivo pg\_ident.conf

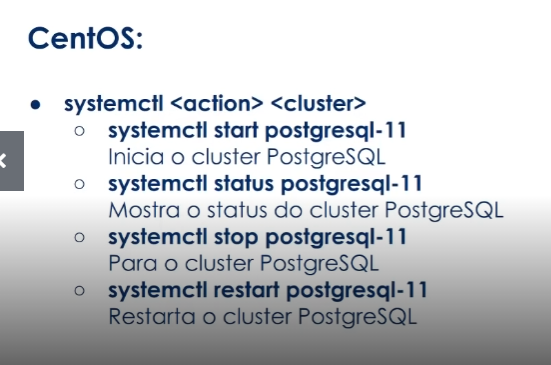
É o arquivo responsável por mapear os usuários do sistema operacional com os usuários do banco de dados. Localizado no diretório de dados PGDATA de sua instalação. A opção ident deve ser utilizada no arquivo pg\_hba.conf.

Parte 4: comandos administrativos

São os comandos para startar o banco, desligar o banco etc. Onde estão esses comandos.

UBUNTU:

* PG\_LSCLUSTERS – lista todos os clusters do PostgreSQL
* PG\_CREATECLUSTER <VERSION> <CLUSTER NAME> - cria um novo cluster PostgreSQL
* PG\_DROPCLUSTER <VERSION> <CLUSTER> - apaga um cluster PostgreSQL
* PG\_CTLCLUSTER <VERSION> <CLUSTER> <ACTION> - start, stop, status, restart de clusters PostgreSQL.



**Binários do PostgreSQL**:

* Createdb
* Createuser
* Dropdb
* Dropuser
* Initdb
* Pg\_ctl
* Pg\_basebackup
* Pg\_dump /pg\_dumpall
* Pg\_restore
* Psql
* Reindexdb
* Vacuumdb

**Arquitetura/Hierarquia**

Cluster: É uma coleção de bancos de dados que compartilham as mesmas configurações (arquivos de configuração) do PostgreSQL e do sistema operacional (porta, listen\_addresses etc). Dentro desse cluster, pode-se ter um ou mais bancos de dados.

Banco de dados (database): É um conjunto de schemas com seus objetos/relações (tabelas, funções, views etc).

Schemas: É um conjunto de objetos/relações (tabelas, funções, views, etc). No MySQL, schema é identificado como um banco de dados, ou seja, diferente do que diz o PostgreSQL.

**CONHEÇA A FERRAMENTA PGADMIN**

Importante para conexão:

1 – liberar acesso ao cluster em postgresql.conf

2 – liberar acesso ao cluster para o usuário de banco de dados em pg\_hba.conf

3 – criar/editar usuários.

**COMO ADMINISTRAR USUÁRIOS NO BANCO DE DADOS**

Parte 1: Conceitos users/roles/groups

Roles (papeis ou funções), users (usuários) e grupo de usuários são “contas”, perfis de atuação em um banco de dados, que possuem permissões em comum ou específicas. Nas versões anteriores ao do PostgreSQL 8.1, usuários e roles tinham comportamentos diferentes. Atualmente, roles e users são alias. É possível que roles pertença a outras roles.

Parte 2: administrando users/roles/groups

CREATE ROLE name WITH option

Where option can be:

SUPERUSER | NOSUPERUSER

CREATEDB | NOCREATEDB

CREATEROLE | NOCREATEROLE

INHERIT | NOINHERIT

LOGIN | NOLOGIN

REPLICATION | NOREPLICATION

BYPASSRLS | NOBYPASSRLS

CONNECTION LIMIT connlimit

[ENCRYPTED] PASSWORD ‘password’ | PASSWORD NULL

VALID UNTIL ‘timestamp’

IN ROLE role\_name [, ...]

IN GROUP role\_name [, ...]

ROLE role\_name [, ...]

ADMIN role\_name [, ...]

USER role\_name [, ...]

SYSID uid

Associação entre roles

Quando uma role assume as permissões de outra role. Necessário a opção **INHERIT**.

No momento de criação da role:

- **IN ROLE** (passa a pertencer a role informada)

- **ROLE** (a role informada passa a pertencer a nova role)

Ou após a criação da role:

- GRANT [role a ser concedida] TO [role a assumir as permissões]

CREATE ROLE professores

NOCREATEDB

NOCREATEROLE

INHERIT

NOLOGIN

NOBYPASSRLS

CONNECTION LIMIT -1;

CREATE ROLE daniel LOGIN CONNECTION LIMIT 1 PASSWORD ‘123’ IN ROLE professores;

- A role Daniel passa a assumir as permissões da role professores.

CREATE ROLE daniel LOGIN CONNECTION LIMIT 1 PASSWORD ‘123’ ROLE professores;

- A role professores passa a fazer parte da role daniel assumindo suas permissões.

CREATE ROLE daniel LOGIN CONNECTION LIMI 1 PASSWORD ‘123’;

GRANT professores TO daniel;

Desassociar membros entre roles

REVOKE [role que será revogada] FROM [role que terá suas permissões revogadas]

REVOKE professores FROM daniel;

Alterando uma role

ALTER ROLE role\_specification [WITH] option [...]

Where option can be:

SUPERUSER | NOSUPERUSER

CREATEDB | NOCREATEDB

CREATEROLE | NOCREATEROLE

INHERIT | NOINHERIT

LOGIN | NOLOGIN

REPLICATION | NOREPLICATION

BYPASSRLS | NOBYPASSRLS

CONNECTION LIMIT connlimit

[ENCRYPTED] PASSWORD ‘password’ | PASSWORD NULL

VALID UNTIL ‘timestamp’

Parte 3: administrando acessos (GRANT)

São privilégios de acesso de determinados objetos do banco de dados.

Privilégios:

**-- tabela**

-- coluna

-- sequence

**-- database**

-- domain

-- foerign data wrapper

- foreign server

**-- function**

-- language

-- large objeto

**-- schema**

-- tablespace

-- type

**DATABASE**

GRANT { { CREATE | CONNECT | TEMPORARY | TEMPO } [, ...] | ALL [ PRIVILEGES ] }

ON DATABASE database+name [, ...]

TO role\_specification [, ...] [ WITH GRANT OPTION ]

**SCHEMA**

GRANT { { CREATE | USAGE } [, ...] | ALL [ PRIVILEGES ] }

ON SCHEMA schema\_name [, ...]

TO role\_specification [, ...] [ WITH GRANT OPTION ]

**TABLE**

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 ]

**REVOKE**

Retira as permissões da role.

**DATABASE**

REVOKE [ GRANT OPTION FOR ]

{ { CREATE | CONNECT | TEMPORARY | TEMP } [, ...] | ALL [ PRIVILEGES ] }

ON DATABASE database\_name [, ...]

FROM { [ GROUP ] role\_name | PUBLIC } [, ...]

[ CASCADE | RESTRICT ]

**SCHEMA**

REVOKE [ GRANT OPTION FOR ]

{ { CREATE | USAGE } [, ...] | ALL [ PRIVILEGES ] }

ON SCHEMA schema\_name [, ...]

FROM { [ GROUP ] role\_name | PUBLIC } [, ...]

[ CASCADE | RESTRICT ]

**REVOGANDO TODAS AS PERMISSÕES (SIMPLIFICADO)**

REVOKE ALL ON ALL TABLES IN SCHEMA [ schema] FROM [role];

REVOKE ALL ON SCHEMA [schema] FROM [role];

REVOKE ALL ON DATABASE [database] FROM [role];

**OBJETOS E COMANDOS DO BANCO DE DADOS**

DATABASE

É o banco de dados. Grupo de schemas e seus objetos, como tables, types, views, funções, entre outros. Seus schemas e objetos não podem ser compartilhados entre si. Cada database é separado um do outro compartilhando apenas usuários/roles e configurações do cluster PostgreSQL.

SCHEMAS

É um grupo de objetos, como tabelas, types, views, funções, entre outros. É possível relacionar objetos entre diversos schemas. Por exemplo: schema public e schema curso podem ter tabelas com o mesmo nome (teste por exemplo) relacionando-se entre si.

OBJETOS

São as tabelas, views, funções, types, sequences, entre outros, pertencentes aos schemas e não podem ser compartilhados entre databases.

DATABASE

CREATE DATABASE name

[ [ WITH ] [ OWNER [=] user\_name ]

[ TEMPLATE [=] template ]

[ ENCODING [=] encoding ]

[ LC\_COLLATE [=] lc\_collate ]

[ LC\_CTUPE [=] lc\_ctype ]

[ TABLESPACE [=] tablespace\_name ]

[ ALLOW\_CONNECTIONS [=] connlimit ]

[ IS\_TEMPLATE [=] istemplate ] ]

ALTER DATABASE name RENAME TO new\_name

ALTER DATABASE name OWNER TO { new\_owner | CURRENT\_USER | SESSION\_USER }

ALTER DATABASE name SET TABLESPACE new\_tablespace

DROP DATABASE [name]

SCHEMA

CREATE SCHEMA schema\_name [AUTHORIZATION role\_specification ]

ALTER SCHEMA name RENAME TO new\_name

ALTER SCHEMA name OWNER TO { new\_owner | CURRENT\_USER | SESSION\_USER }

DROP SCHEMA [nome]

Melhores práticas:

CREATE SCHEMA IF NOT EXISTS schema\_name [AUTHORIZATION role\_specification ]

DROP SCHEMA IF EXISTIS [nome];

Parte 2: Tabelas, Colunas e Tipos de dados

Definição

Conjunto de dados dispostos em colunas e linhas referentes a um objetivo comum. As colunas são consideradas como “campos da tabela”, como atributos da tabela. As linhas de uma tabela são chamadas também de tuplas, e é onde estão contidas os valores, os dados.

Primary Key / Chave Primária / PK

No conceito de modelo de dados relacional e obedecendo as regras de normalização, uma PJ é um conjunto de um ou mais campos que nunca se repetem em uma tabela e que seus valores garantem a integridade do dado único e a utilização do mesmo como referência para o relacionamento entre demais tabelas.

* Não pode haver duas ocorrências de uma mesma entidade com o mesmo conteúdo na PK.
* A chave primária não pode ser composta por atributo opcional, ou seja, atributo que aceite nulo.
* Os atributos identificadores devem ser o conjunto mínimo que pode identificar cada instância de uma entidade.
* Não devem ser usadas chaves externas.
* Não deve conter informação volátil.

Foreign Key / Chave Estrangeira / FK

Campo ou conjunto de campos que são referências de chaves primárias de outras tabelas ou da mesma tabela. Sua principal função é garantir a integridade referencial entre tabelas.

Tipos de dados

**Numeric types**

Monetary types

**Character Types**

Binary Data Types

**Date/Time Types**

**Boolean Type**

Enumerated Types

Geometric Types

Network Address Types

Bit String Types

Text Search Types

UUID Type

XML Type

JSON Types

Arrays Composite Types

Range Types

Domain Types

Object Identifier Types

Pg\_lsn Type

Pseudo-Types

Numéricos:

Smallint

Integer

Bigint

Decimal

Numeric

Real

Double precision

Smallserial

Serial

Bigserial

Caracteres

Chacater varying(n), varchar(n)

Character(n), char(n)

Text

Datas

Timestamp [ (p) ] [ without time zone ]

Timestamp [ (p) ] [ with time zone ]

Date

Time [ (p) ] [ without time zone ]

Time [ (p) ] [ with time zone ]

Interval [ fields ] [ (p) ]

Booleanos

Booelan true or false 0 ou 1.

Parte 3: DML e DDL

DML: Data Manipulation Language. É a linguagem e manipulação de dados. INSERT, UPDATE, DELETE, SELECT.

DDL: Data Definition Language. É a linguagem de definição de dados. CREATE, ALTER, DROP.

CREATE / ALTER / DROP

CREATE [objeto] [nome do objeto] [opções];

ALTER [objeto] [nome do objeto] [opções];

DROP [objeto] [nome do objeto] [opções];

CREATE DATABASe dadosbancarios;

ALTER DATABASE dadosbancarios OWNER TO direitoria;

DROP DATABASE dados bancariso;

CREATE SCHEMA IF NOT EXISTS bancos;

ALTER SCHEMA bancos OWNER TO diretoria;

DROP SCHEMA IF EXISTS bancos;

CREATE TABLE [IF NOT EXISTS] [nome da tabela] (

[nome do campo] [tipo] [regras] [opções],

[nome do campo] [tipo] [regras] [opções],

[nome do campo] [tipo] [regras] [opções]

);

ALTER TABLE [nome da tabela] [opções];

DROP TABLE [nome da table];

EXAMPLES:



INSERT

INSERT INTO [nome da tabela] ([campos da tabela,])

VALUES ([valores de acordo com a ordem dos campos acima,]);

INSERT INTO [nome da tabela] ([campos da tabela,])

SELECT ([valores de acordo com a ordem dos campos acima,]);

INSERT INTO BANCO (código, nome, data\_criacao)

VALUES (100, ‘Banco do Brasil’, now());

INSERT INTO banco (código, nome, data\_criacao)

SELECT 100, ‘Banco do Brasil’, now());

UPDATE

UPDATE [nome da tabela] SET

[campo1] = [novo valor do campo1]

[campo2] = [novo valor do campo2]

...

[WHERE + condições]

**ATENÇÃO: muito cuidado com os updates. Sempre utilize-os com condição.**

UPDATE banco SET

codigo = 500

WHERE codigo = 100;

UPDATE banco SET

data\_criacao = now()

WHERE data\_criacao IS NULL;

DELETE

DELETE FROM [nome da tabela]

[WHERE + condições]

**ATENÇÃO: muito cuidado com os deletes. Sempre utilize-os com condição.**

DELETE FROM banco