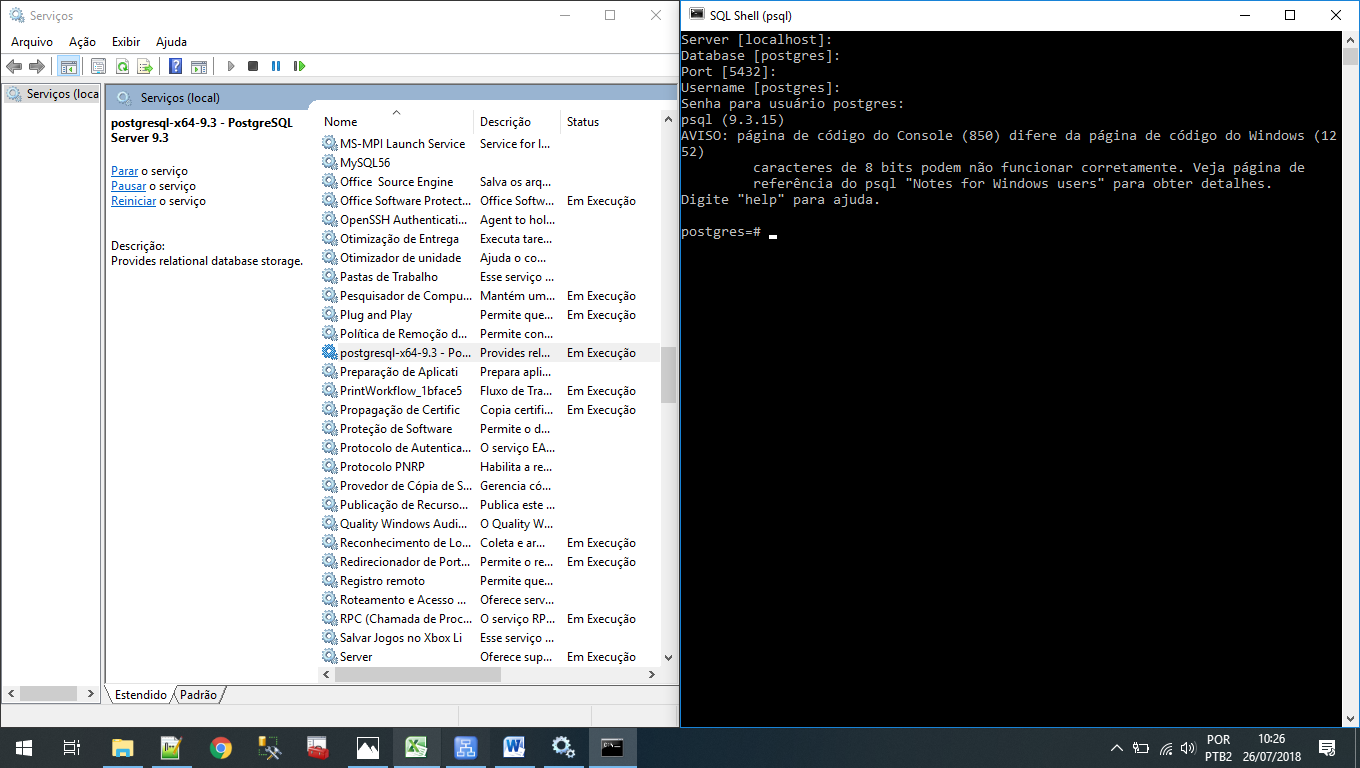
PostgreSQL

Ferramenta de banco de dados gratuita com os melhores recursos do mercado. Muito robusto, seguro e com ótimas funcionalidades de programação e administração, é recomendado para aplicações de médio e grande porte.



-- Aula5 26/07/2018

/\*

Estudo de Caso.

Uma empresa do ramo imobiliário requer controlar os dados de imóveis e

clientes que podem ser proprietários de imóveis.

Existem dois tipos de clientes, pessoa física e pessoa jurídica.

Os dados dos clientes pessoa física são: nome, email, renda, cpf e gênero.

Os dados dos clientes pessoa jurídica são: nome, email, renda, cnpj e ramo.

Cada cliente pode possuir mais de um imóvel, já um imóvel pode ter apenas um

proprietário. Pode haver tanto imóvel sem proprietário, quanto cliente sem

imóvel. Os dados dos imóveis são: tipo(Casa, Apto, Sala, Loja, etc.), área, valor e

endereço.

\*\*/

/\*

Instruções nativas de linha de comando do PostgreSQL são iniciadas com \

Comandos da linguagem SQL e suas sub-linguagens, DDL e DML, não sofrem alteração

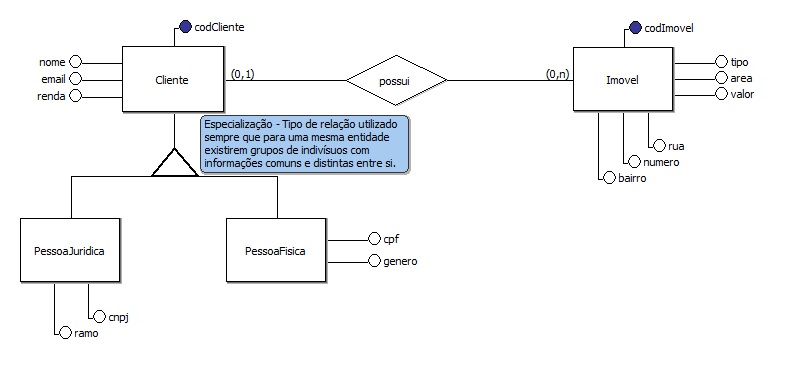
na maioria das ferramentas de banco de dados relacionais.

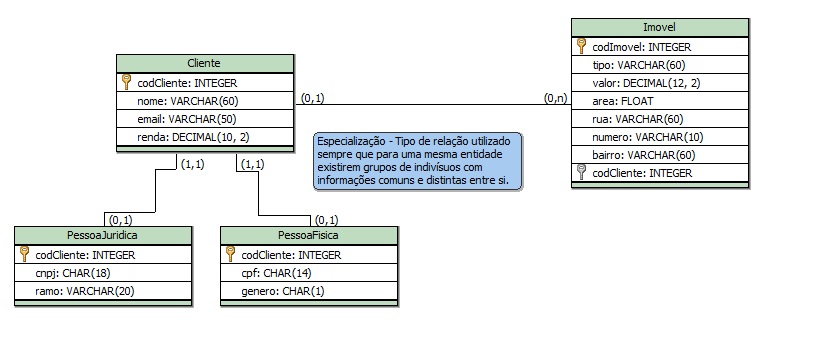
SQL(Linguagem de Busca Estruturada) - SELECT, FROM, WHERE, JOIN, ORDER BY, GROUP BY...

DDL(Linguagem de Definição de Dados) - CREATE, DROP e ALTER.

DML(Linguagem de Manipulação de Dados) - INSERT, UPDATE e DELETE.

\*\*/





-- Exibir bases de dados.

\l

DROP DATABASE aula6;

CREATE DATABASE aula6;

-- Conectar a uma base de dados

\c aula6

CREATE TABLE cliente (

codCliente INTEGER PRIMARY KEY,

nome VARCHAR(60) NOT NULL,

email VARCHAR(50) NOT NULL,

renda DECIMAL(10, 2) NOT NULL);

CREATE TABLE pessoaJuridica (

codCliente INTEGER PRIMARY KEY,

cnpj CHAR(18) NOT NULL,

ramo VARCHAR(20) NOT NULL,

FOREIGN KEY(codCliente) REFERENCES cliente (codCliente));

CREATE TABLE pessoaFisica (

codCliente INTEGER PRIMARY KEY,

cpf CHAR(14) NOT NULL,

genero CHAR(1) NOT NULL,

FOREIGN KEY(codCliente) REFERENCES cliente (codCliente));

CREATE TABLE imovel (

codImovel INTEGER PRIMARY KEY,

tipo VARCHAR(60) NOT NULL,

valor DECIMAL(12, 2) NOT NULL,

area FLOAT NOT NULL,

rua VARCHAR(60) NOT NULL,

numero VARCHAR(10) NOT NULL,

bairro VARCHAR(60) NOT NULL,

cod\_cliente INTEGER,

FOREIGN KEY(cod\_cliente) REFERENCES Cliente (codCliente));

-- Exibir objetos existentes;

\d

-- Exibir a estrutura das tabelas.

\d cliente

\d pessoaFisica

\d pessoaJuridica

\d imovel

-- SEQUENCE - Objeto utilizado para gerar velores sequenciais.

CREATE SEQUENCE seqcliente;

CREATE SEQUENCE seqimv START 101 INCREMENT BY 1;

-- Inserir dados nas tabelas.

INSERT INTO cliente VALUES(NEXTVAL('seqcliente'), 'Bia', 'bia@aol.com', 4000);

INSERT INTO cliente VALUES(NEXTVAL('seqcliente'), 'Ana', 'ana@uol.com', 6000);

INSERT INTO cliente VALUES(NEXTVAL('seqcliente'), 'Leo', 'leo@uol.com', 5500);

INSERT INTO cliente VALUES(NEXTVAL('seqcliente'), 'Nat', 'nat@aol.com', 7100);

INSERT INTO cliente VALUES(NEXTVAL('seqcliente'), 'XPTO', 'xpto@otpx.com', 9000);

INSERT INTO cliente VALUES(NEXTVAL('seqcliente'), 'COTI', 'coti@info.com', 15000);

SELECT \* FROM cliente;

INSERT INTO pessoaFisica VALUES(1, '234.432.441-16', 'f'),

(2, '111.432.441-17', 'f'),

(3, '222.432.441-18', 'm'),

(4, '333.432.441-19', 'f');

SELECT \* FROM pessoaFisica;

INSERT INTO pessoaJuridica VALUES(5, '29.432.442/0001-17', 'COM'),

(6, '13.145.523/0001-24', 'EDU');

SELECT \* FROM pessoaJuridica;

-- ALTER TABLE imovel ALTER COLUMN cod\_cliente DROP NOT NULL;

INSERT INTO imovel VALUES(NEXTVAL('seqimv'), 'Apto', 360000, 80, 'Rua a', '352', 'Bairro q', 1),

(NEXTVAL('seqimv'), 'Apto', 430000, 115, 'Rua s', '52', 'Bairro w', 4),

(NEXTVAL('seqimv'), 'Casa', 520000, 180, 'Rua d', '32', 'Bairro e', 3),

(NEXTVAL('seqimv'), 'Apto', 315000, 70, 'Rua f', '49', 'Bairro e', 4),

(NEXTVAL('seqimv'), 'Sala', 190000, 40, 'Rua g', '25B', 'Bairro w', 5),

(NEXTVAL('seqimv'), 'Sala', 210000, 30, 'Rua h', '1035', 'Bairro t', 6),

(NEXTVAL('seqimv'), 'Apto', 295000, 30, 'Rua k', '17', 'Bairro p', NULL),

(NEXTVAL('seqimv'), 'Loja', 600000, 200, 'Rua j', '975', 'Bairro q', 6);

SELECT \* FROM imovel;

-- Pesquisas.

SELECT nome, email, cpf, genero, renda

FROM cliente c INNER JOIN pessoaFisica pf

ON c.codCliente = pf.codCliente;

SELECT nome, email, cnpj, ramo, renda

FROM cliente c INNER JOIN pessoaJuridica pj

ON c.codCliente = pj.codCliente;

SELECT nome, email, renda, tipo, area, valor

FROM cliente INNER JOIN imovel

ON codCliente = cod\_cliente;

SELECT nome, email, renda, tipo, area, valor

FROM cliente LEFT JOIN imovel

ON codCliente = cod\_cliente;

SELECT nome, email, renda, tipo, area, valor

FROM cliente RIGHT JOIN imovel

ON codCliente = cod\_cliente;

SELECT nome, email, renda, tipo, area, valor

FROM cliente FULL JOIN imovel

ON codCliente = cod\_cliente;

SELECT nome, email, renda, tipo, area, valor

FROM cliente FULL JOIN imovel

ON codCliente = cod\_cliente

WHERE tipo = 'Apto' AND valor < 400000

ORDER BY valor;

-- Programação no Banco de Dados.

-- Função para executar SQL

CREATE FUNCTION dadosCliente(cod INT)

RETURNS VARCHAR(60) AS $$

SELECT 'Nome: ' || nome || ' / Email: ' || email

FROM cliente WHERE codCliente = cod;

$$

LANGUAGE SQL;

SELECT dadosCliente(6);

SELECT dadosCliente(2);

-- Função utilizando PLPGSQL.

CREATE OR REPLACE FUNCTION calcDesconto(valor FLOAT, porcent FLOAT)

RETURNS FLOAT AS $$

DECLARE

resultado FLOAT;

BEGIN

resultado = valor - (valor \* (porcent / 100));

RETURN resultado;

END;

$$

LANGUAGE PLPGSQL;

SELECT calcDesconto(100, 5);

SELECT tipo, valor, calcDesconto(valor, 7.5) AS ValorDesconto

FROM imovel;

UPDATE cliente SET renda = calcDesconto(renda, 0.75) WHERE codCliente = 2;

SELECT \* FROM cliente;

-- Função para realizar transação.

CREATE OR REPLACE FUNCTION cadPF(pnome VARCHAR(60), pemail VARCHAR(60), prenda FLOAT,

pcpf CHAR(14), sexo CHAR)

RETURNS VARCHAR(100) AS $$

DECLARE

fk\_cliente INT;

msg VARCHAR(100);

BEGIN

INSERT INTO cliente VALUES(NEXTVAL('seqcliente'), pnome, pemail, prenda);

fk\_cliente = CURRVAL('seqcliente');

INSERT INTO pessoaFisica VALUES(fk\_cliente, pcpf, sexo);

msg = pnome || ' cadastrado como Pessoa Fisica.';

RETURN msg;

END;

$$

LANGUAGE PLPGSQL;

SELECT cadPF('Beto', 'beto@uol.com', 4500, '344.111.222-44', 'm');

SELECT \* FROM cliente;

SELECT \* FROM pessoaFisica;

-- Criar View

CREATE VIEW relProprietarios AS

SELECT nome, email, renda, tipo, area, valor

FROM cliente FULL JOIN imovel

ON codCliente = cod\_cliente;

-- Exportar dados para arquivos.

\copy imovel TO 'c:/bancodedados/imovel.txt'

\copy (SELECT \* FROM relProprietarios) TO 'c:/bancodedados/proprietarios.txt'

\copy (SELECT \* FROM relProprietarios) TO 'c:/bancodedados/proprietarios.csv' DELIMITER ';' CSV HEADER

-- Criar cópia da tabela sem os dados.

CREATE TABLE copiaImv (LIKE imovel);

\d copiaImv

SELECT \* FROM copiaImv;

-- Importar dados de um arquivo.

\copy copiaImv FROM 'c:/bancodedados/imovel.txt'

SELECT \* FROM copiaImv;