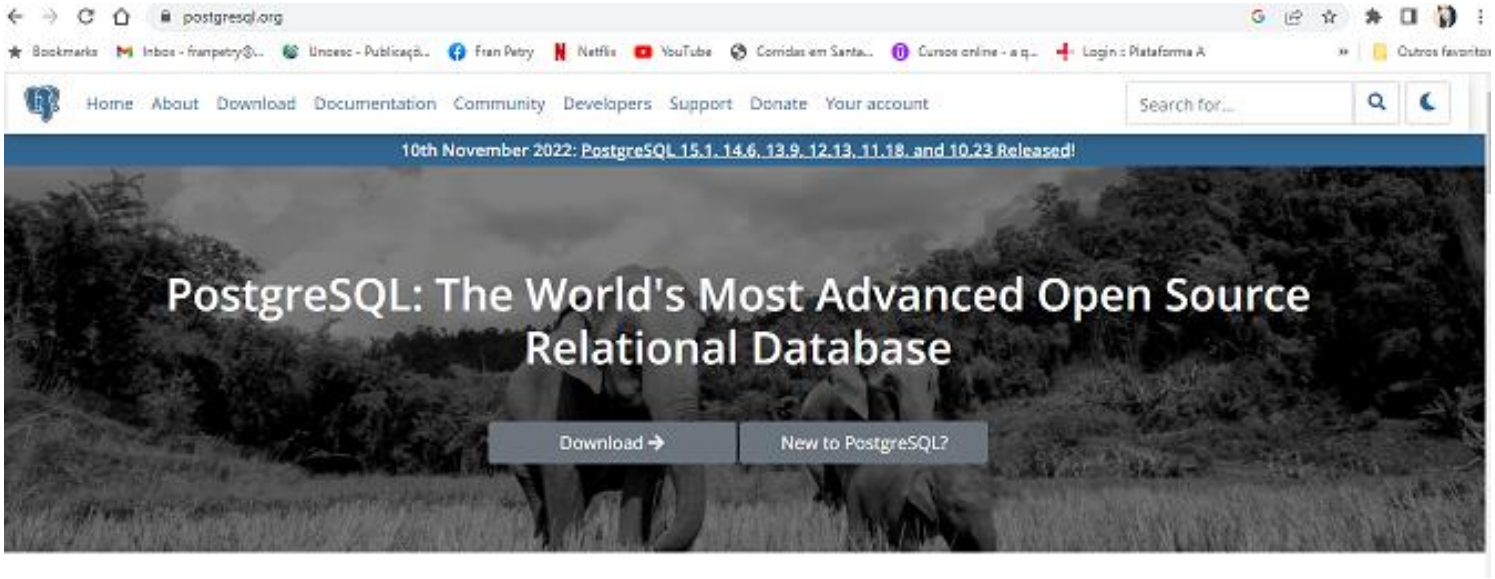
PostgreSQL

FAPESC – DESENVOLVEDORES PARA TECNOLOGIA DA INFORMAÇÃO

FRANCIELE PETRY
Franciele.petry@unoesc.edu.br

DEFINIÇÃO

https://www.postgresql.org/



DEFINIÇÃO

- O PostgreSQL é também considerado por muitos como um sistema de gerenciamento de banco de dados objeto-relacional (SGBDOR), tendo sido o pioneiro em muitos conceitos objeto-relacionais que agora estão se tornando disponíveis em alguns bancos de dados comerciais
  - O projeto POSTGRES foi iniciado em 1985, sendo uma evolução do projeto Ingres, liderado pelo Professor Michael Stonebraker (ganhador do Prêmio Turing de 2014), foi patrocinado pelas seguintes instituições: *Defense Advanced Research Projects Agency* (DARPA); *Army Research Office* (ARO); *National Science Foundation* (NSF); e ESL, Inc.
  - Ele foi lançado para uso externo em 1996 no Departamento de Ciência da Computação da Universidade da Califórnia em Berkeley
  - O PostgreSQL descende deste código original de Berkeley, possuindo o código-fonte aberto e fornecendo suporte às linguagens SQL92/SQL99, além de outras funcionalidades modernas

Definição

- O PostgreSQL é um dos SGBDs de código aberto (licença BSD) mais avançados que existem atualmente, contando com recursos como
  - Multiplataforma: Windows, Linux, FreeBSD, OpenBSD, macOS
  - Consultas complexas
  - Chaves estrangeiras
  - Integridade transacional
  - Controle de concorrência multiversão (MVCC)
  - Tipos de dados geométricos, endereços de rede, JSON, XML e *arrays*
  - Gatilhos (*triggers*)
  - Visões (*views*)
  - Linguagem procedural em várias linguagens (PL/pgSQL, PL/Python, PL/Java, PL/Perl) para procedimentos armazenados *(stored procedures)*
  - Estrutura para guardar dados georreferenciados PostGIS
  - Criptografia
  - Criação de tabelas temporárias
- Acompanhamento de *slow query*

UNOESC

Apoiadores:

UNOESC CITEOSC INOVALE CENTRO DE INOVAÇÃO VIDEIRA DEV-TI SC TEC fapesc SANTA CATARINA

DEFINIÇÃO

- Recursos (continuação...)
  - Suporte ao modelo híbrido objeto-relacional
    - Herança de tabelas
    - Sobrecarga de funções
  - Recuperação em um ponto no tempo (PITR)
  - Transações agrupadas (*savepoints*)
  - Múltiplas transações online concorrentes entre usuários
  - Expressões regulares
  - Subconsultas
  - Suporte a *rules* (sistema de regras que reescreve diretivas SQL)
  - Sofisticado planejador de consultas (otimizador)
  - Suporta conjuntos de caracteres internacionais com codificação de caracteres multibyte, Unicode e sua ordenação por localização
  - Indexação por texto
  - Busca *full-text*

Definição

- Apresenta conformidade com o padrão ACID
  - **A**tomicidade: Suporta transações, ou todas as operações são efetuadas, ou nenhuma é
  - **C**onsistência: As regras de integridade devem ser obedecidas para não levar um banco de dados a um estado inconsistente
  - **I**solamento: Evita que transações concorrentes interfiram umas nas outras
- **D**urabilidade: Transações bem sucedidas devem persistir no banco de dados, mesmo em casos de quedas de energia, travamentos ou erros

TIPOS DE DADOS BÁSICOS

Tipos de Dados Mais Comuns

Numéricos

Tipo	Tamanho	Apelido	Faixa
SMALLINT	2 bytes	inteiro pequeno	32768 a +32767
INTEGER ou INT	4 bytes	inteiro	2147483648 até +2147483647
BIGINT	8 bytes	inteiro longo	9223372036854775808 a +9223372036854775807
NUMERIC(p,e) / DECIMAL(p,e)			tamanho variável, precisão especificada pelo usuário. Exato e sem limite e – escala (casas decimais) p – precisão (total de dígitos, inclusive escala)
REAL ou FLOAT	4 bytes	ponto flutuante	precisão variável, não exato e precisão de 6 dígitos
DOUBLE PRECISION	8 bytes	dupla precisão	precisão variável, não exato e precisão de 15 dígitos

Caracteres

Tipo	Apelido	Faixa
CHARACTER VARYING(n) ou VARCHAR(n)	caractere tamanho variável	comprimento variável, com limite
CHARACTER(n) ou CHAR(n)	caractere tamanho fixo	comprimento fixo
TEXT	comprimento variável e ilimitado	

Data/Hora

Tipo	Tamanho	Apelido	Faixa
TIMESTAMP[(p)] [without time zone]	8 bytes	data e hora sem zona	4713 AC a 5874897 DC
TIMESTAMP [(p)] [with time zone]	8 bytes	data e hora com zona	4713 AC a 5874897 DC
INTERVAL	12 bytes	intervalo de tempo	178000000 anos a 178000000 anos
DATE	4 bytes	somente data	4713 AC até 32767 DC
TIME [(p)] [without time zone]	8 bytes	somente a hora	00:00:00.00 até 23:59:59.99
TIME [(p)] [with time zone]	8 bytes	somente a hora	00:00:00.00 até 23:59:59.99

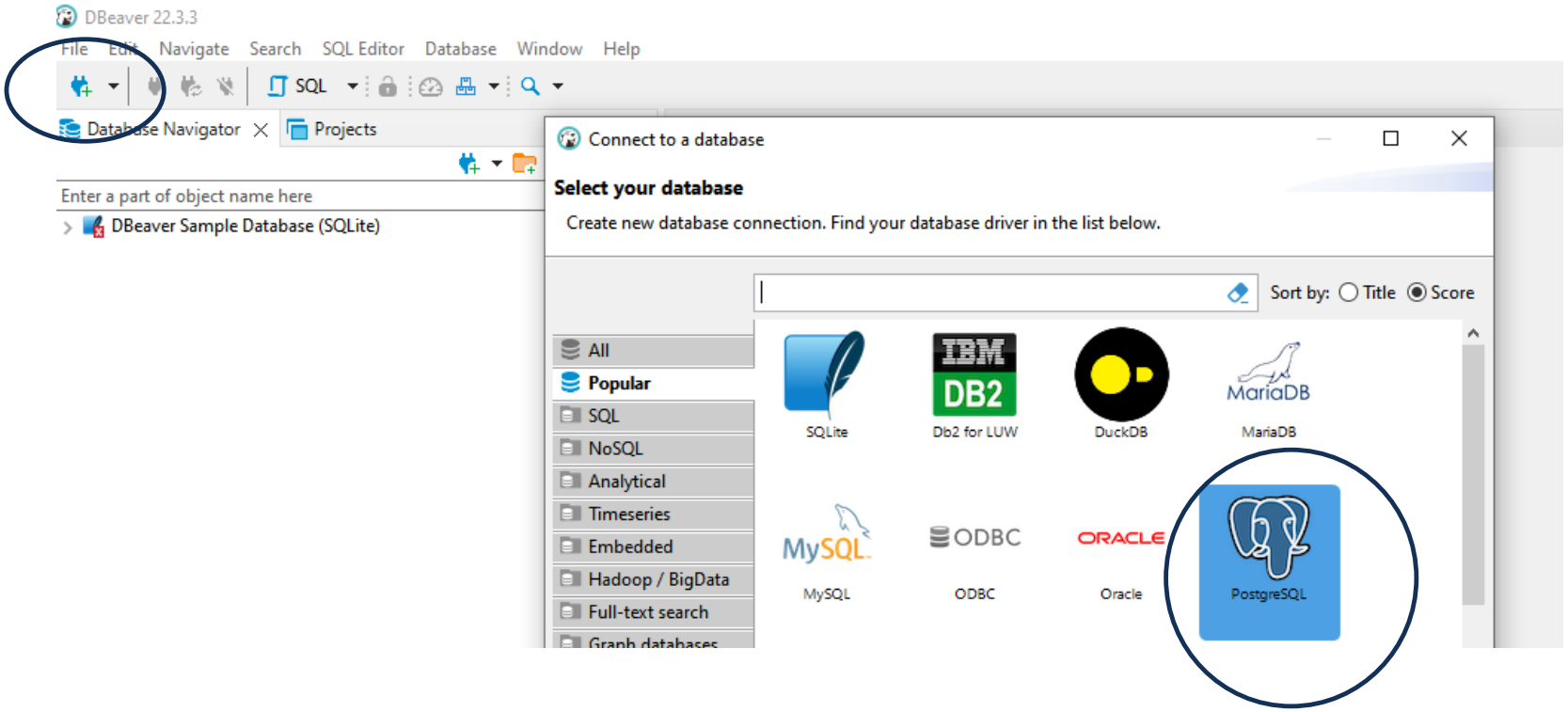
[(p)] é a precisão, que varia de 0 a 6 e o default é 2.

Booleanos

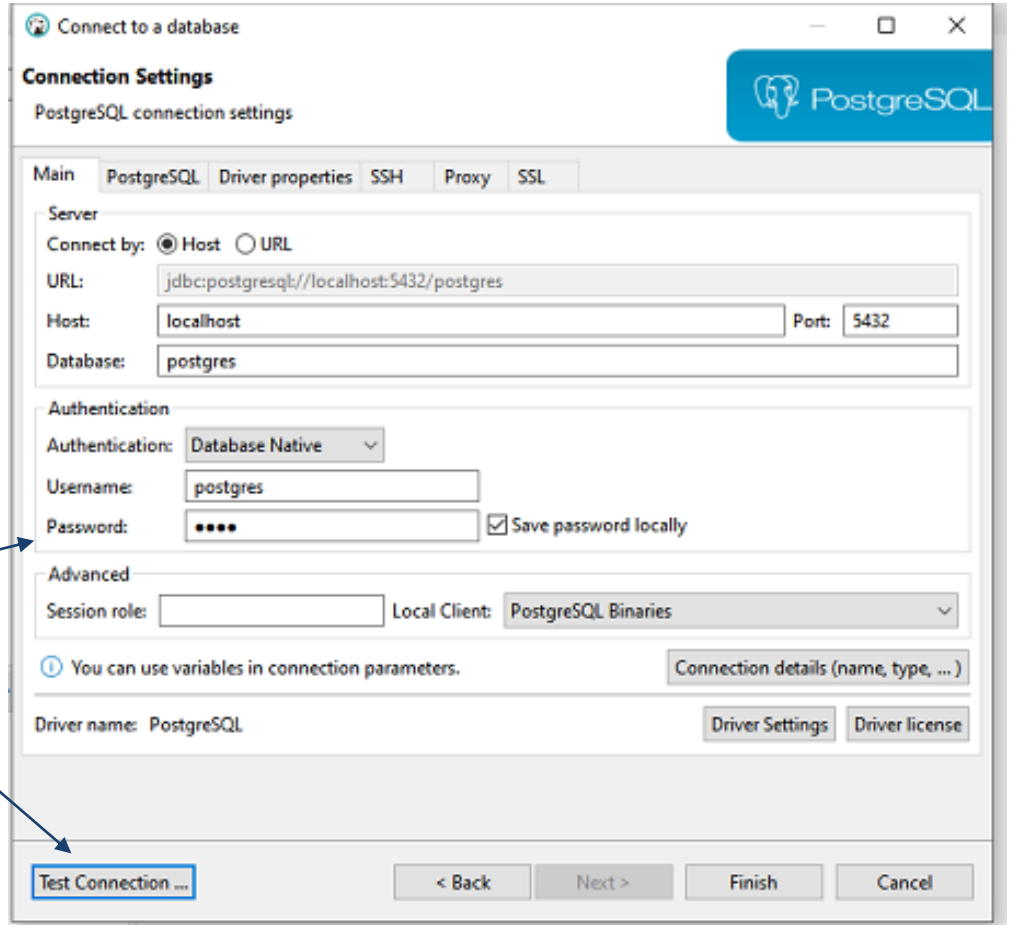
Tipo	Tamanho	Apelido	Faixa
TRUE			't', 'true', 'y', 'yes' e '1'
FALSE			'f', 'false', 'n', 'no', '0'

Apenas um dos dois estados. O terceiro estado, desconhecido, é representado pelo NULL.

DBeaver + PostgreSQL

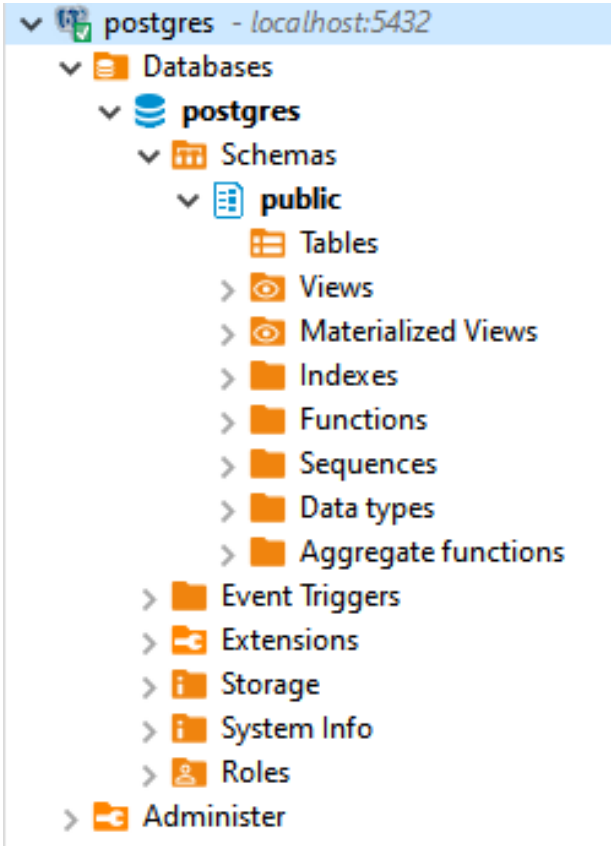


DBEAVER + POSTGRESQL



Apoiadores:

DBEAVER + POSTGRESQL



CREATE DATABASE

```sql
create database nome_do_banco_de_dados;

create database floricultura;
```

CREATE TABLE

```sql
CREATE TABLE departamentos (
    id_departamento SERIAL PRIMARY KEY,
    nome VARCHAR(50) NOT NULL
);

CREATE TABLE cargos (
    id_cargo SERIAL PRIMARY KEY,
    id_departamento INTEGER NOT NULL,
    nome VARCHAR(50) NOT NULL,
    FOREIGN KEY (id_departamento) REFERENCES departamentos(id_departamento)
);

CREATE TABLE funcionarios (
    id SERIAL PRIMARY KEY,
    nome VARCHAR(60) NOT NULL,
    escolaridade tipo_escolaridade,
    id_cargo INTEGER REFERENCES cargos(id_cargo),
    salario DECIMAL NOT NULL
);
```

INSERINDO DADOS

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

```
INSERT INTO departamentos VALUES(1, 'TI');
INSERT INTO cargos VALUES(1, 1, 'Desenvolvedor');
INSERT INTO funcionarios VALUES(1, 'André', 'Especialização', 1, 1000);
```

Selecionando Dados

Cláusula	Finalidade
SELECT	Determina quais colunas serão incluídas no conjunto resultado da consulta. * significa todos os campos.
FROM	Identifica a(s) tabela(s) de onde serão retirados os dados e como as tabelas deverão ser unidas.
WHERE	Filtra os resultados.
GROUP BY	Agrupa registros por meio de valores comuns dos campos.
HAVING	Filtra os resultados dos grupos.
ORDER BY	Ordena as linhas do resultado final usando um ou mais campos.

```
SELECT * FROM funcionarios ORDER BY nome;
SELECT * FROM funcionarios ORDER BY nome ASC;
SELECT * FROM funcionarios ORDER BY nome DESC;
```

EXERCÍCIOS

1. Utilize o comando SELECT para listar os registros por ordem de salário, dos mais altos para os mais baixos.

2. Use as funções SUM, MAX, MIN e AVG para descobrir, respectivamente, qual a soma, o maior, o menor e a média dos salários.

EXERCÍCIOS

O parâmetro DISTINCT evita que valores repetidos sejam mostrados em uma consulta.

Exemplo: SELECT DISTINCT escolaridade FROM funcionarios;

Exercícios:

3. Utilize o parâmetro DISTINCT para listar todos os cargos da tabela funcionarios.
4. Utilize o parâmetro DISTINCT para listar todos os departamentos da tabela cargos.

EXERCÍCIOS

O parâmetro GROUP BY realiza operações COUNT, SUM, AVG, MAX e MIN baseadas em agrupamentos.

Exemplo: Calcular a média de salário de cada função registrada na tabela funcionarios.

SELECT id_cargo, AVG(salario) FROM funcionarios GROUP BY id_cargo;

EXERCÍCIOS

O parâmetro COUNT realiza a contagem dos registros de uma tabela, podendo ainda atender a um critério de filtragem.

Exemplo: SELECT COUNT(*) FROM funcionarios;

Exercícios:

5. Utilize o parâmetro COUNT para descobrir quantos funcionários possuem escolaridade 'Especialização'.

6.Utilize o parâmetro COUNT para descobrir quantos funcionários ganham acima de R$ 1000,00.

EXERCÍCIOS

O parâmetro HAVING é parecido com o parâmetro WHERE. A diferença é que ele é usado para filtrar os dados do argumento GROUP BY.

Exemplo: Listar apenas os cargos cujos salários médios sejam acima de R$ 1000,00

SELECT id_cargo, AVG(salario) FROM funcionarios GROUP BY id_cargo HAVING AVG(salario)>1000;

UPDATE E DELETE

UPDATE nome_da_tabela SET nome_da_coluna1 = valor_da_coluna1, nome_da_coluna2 = valor_da_coluna2 WHERE condição;

UPDATE cargos **SET** nome = 'Analista' **WHERE** id_cargo='1';

DELETE

DELETE FROM *table_name* WHERE *condition*;

DELETE FROM funcionarios **WHERE** nome = 'Fran';