SQL

Bancos de dados relacional com postgreSQL

SAE COMP

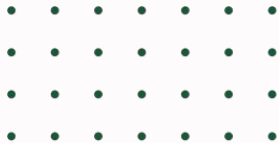
Objetivos

01 Entender o que é SQL

02 Conhecer a estrutura de um banco relacional

03 Aprender os comandos básicos de manipulação de dados

04 Praticar criação, inserção, leitura, filtragem e atualização de dados

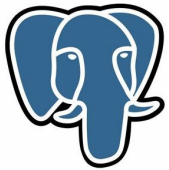


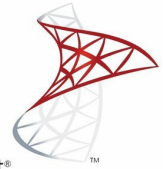
O QUE É SQL

O que é SQL?

Structured Query Language

- Linguagem padrão para bancos relacionais
- Sintaxe próxima da linguagem humana
- Usada para criar, consultar, atualizar e remover dados
- Compatível com vários bancos de dados





Instalando o PostgreSQL

Será utilizado o Postgre SQL durante essa aula. Para utilizá-lo, deve ser instalado o servidor do banco de dados e um cliente. Ambos estão disponíveis para todos os sistemas operacionais.

Servidor do Postgre SQL:
https://postgresql.org/download/

Cliente oficial (pgAdmin):
https://pgadmin.org/download/

```
1  sudo -u postgres psql
2  ALTER USER postgres with encrypted password 'postgres';
```

ESTRUTURA DE UM BANCO RELACIONAL

Banco de Dados Relacional

- Dados em tabelas (linhas × colunas)
- Cada tabela tem que ter uma chave primária em uma coluna
- Chaves estrangeiras (foreign key) podem ser adicionadas em colunas para fazer relações com outras tabelas

Table: orders

order_id	product	total	customer_id
1	Paper	500	5
2	Pen	10	2
3	Marker	120	3
4	Books	1000	1
5	Erasers	20	4

Table: customers

customer_id	first_name	last_name	phone	country
1	John	Doe	817-646-8833	USA
2	Robert	Luna	412-862-0502	USA
3	David	Robinson	208-340-7906	UK
4	John	Reinhardt	307-242-6285	UK
5	Betty	Doe	806-749-2958	UAE

Chaves de Tabelas

- Chave primária (PK) – única e obrigatória
- Chave estrangeira (FK) – referência a PK de outra tabela

Table: orders

order_id	product	total	customer_id
1	Paper	500	5
2	Pen	10	2
3	Marker	120	3
4	Books	1000	1
5	Erasers	20	4

Table: customers

customer_id	first_name	last_name	phone	country
1	John	Doe	817-646-8833	USA
2	Robert	Luna	412-862-0502	USA
3	David	Robinson	208-340-7906	UK
4	John	Reinhardt	307-242-6285	UK
5	Betty	Doe	806-749-2958	UAE

Tipos de Dados

Servidores SQL podem ter tipos diferentes entre si. Como a aula será focada no Postgre SQL, veremos os tipos principais dele. Uma lista com todos os tipos está disponível em: https://www.postgresql.org/docs/current/datatype.html

Nome	Alias	Descrição
integer	int, int4	Número inteiro (4 bytes)
bigint	int8	Número inteiro grande (8 bytes)
smallint	int2	Número inteiro pequeno (2 bytes)
serial	serial4	Inteiro auto-incrementável (4 bytes)
bigserial	serial8	Auto-incremento para bigint
real	float4	Ponto flutuante simples (4 bytes)
double precision	float8	Ponto flutuante duplo (8 bytes)
numeric(p,s)	decimal(p,s)	Número com precisão exata (p = dígitos totais, s = dígitos depois da vírgula)
boolean	bool	Valor lógico: TRUE ou FALSE
text	—	Texto de tamanho variável
varchar(n)	character varying(n)	Texto com até n caracteres
char(n)	character(n)	Texto com exatamente n caracteres
date	—	Data (ano, mês, dia)

COMANDOS BÁSICOS

Criando uma Tabela

Para criar uma tabela se usa o comando CREATE TABLE. Dentro dele deve ser especificado as colunas, seus tipos e seus modificadores.

```sql
CREATE TABLE usuarios (
  id            BIGSERIAL     PRIMARY KEY,
  username      VARCHAR(50)   UNIQUE NOT NULL,
  nome          VARCHAR(100)  NOT NULL,
  idade         INTEGER       CHECK (idade ⩾ 0),
  email         TEXT          UNIQUE,
  data_cadastro DATE          NOT NULL DEFAULT CURRENT_DATE,
  ativo         BOOLEAN       DEFAULT TRUE
);
```

Modificador	Definição
PRIMARY KEY	Define uma coluna como identificador único da tabela. Não permite NULL.
UNIQUE	Garante que todos os valores da coluna sejam diferentes.
NOT NULL	Obriga a coluna a sempre ter um valor. Não pode ser NULL.
CHECK	Impõe uma condição lógica que os valores devem obedecer.
DEFAULT	Define um valor padrão para a coluna se nada for especificado.
BIGSERIAL	Tipo numérico inteiro de 8 bytes com incremento automático.

```sql
CREATE TABLE posts (
  id          BIGSERIAL     PRIMARY KEY,
  usuario_id  BIGINT        REFERENCES usuarios(id) ON DELETE CASCADE,
  titulo      VARCHAR(200)  NOT NULL,
  conteudo    TEXT          NOT NULL,
  criado_em   TIMESTAMP     DEFAULT CURRENT_TIMESTAMP
);
```

Inserindo Dados

Para inserir dados em uma tabela, é necessário usar o comando INSERT INTO. Nele deve-se especificar o nome das colunas que os dados serão inseridos e depois do comando VALUES deve ser colocado os registros dos dados.

```sql
INSERT INTO usuarios
(username, nome, idade, email)

VALUES
('ana123',   'Ana Almeida',   20, 'ana@example.com'),
('bruno22',  'Bruno Silva',   22, 'bruno@example.com');
```

Consultando Dados

Para a consulta de dados se utiliza o comando SELECT. Logo em seguida, é especificado quais colunas serão retornadas e o FROM define a tabela. É possível definir filtros após a palavra WHERE.

```sql
-- Todos os registros
SELECT * FROM usuarios;

-- Dados com filtro
SELECT username, nome
FROM   usuarios
WHERE  idade > 18;
```

Atualizando e Deletando

Para atualizar os dados se usa UPDATE com SET definindo os novos campos. Não se esqueça de colocar where para selecionar os dados que serão atualizados. Para apagar se usa a palavra DELETE FROM junto com WHERE para selecionar.

```sql
UPDATE usuarios
SET    ativo = FALSE
WHERE  username = 'ana123';

DELETE FROM usuarios
WHERE  idade < 18;
```

Ordenando e Limitando

É possível combinar o comando SELECT com ORDER BY para retornar dados fora da sua ordenação padrão. ASC define que o menor vem primeiro e DESC define que o maior vem primeiro. LIMIT limita a quantidade de dados e OFFSET pula uma certa quantidade de linhas.

```sql
SELECT * FROM usuarios
ORDER BY idade DESC, email ASC NULLS LAST
LIMIT 5 OFFSET 1;
```

Operadores e Filtros

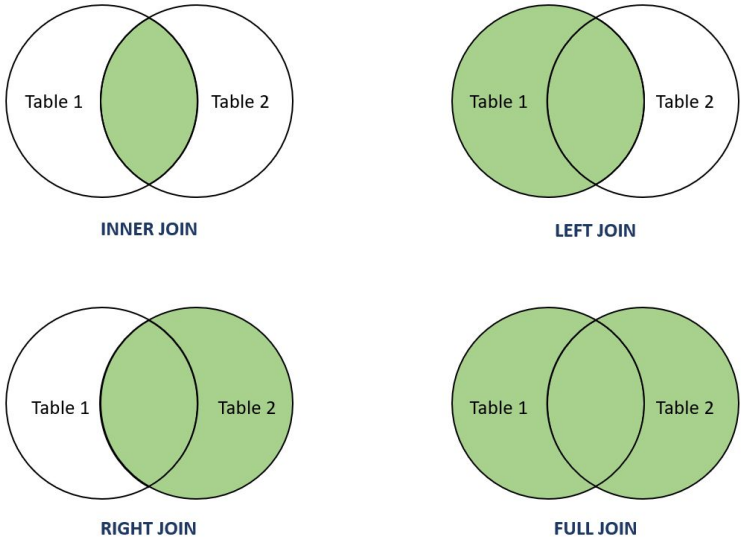
Dentro dos filtros com WHERE é possível colocar diversos operadores para que sejam criados filtros mais customizados.

```sql
-- Produtos com preço acima de R$ 1000
SELECT * FROM produtos WHERE preco > 1000;

-- Produtos com preço entre R$ 50 e R$ 2000
SELECT * FROM produtos WHERE preco BETWEEN 50 AND 2000;

-- Produtos cujo nome contenha "book" (insensível a maiúsculas)
SELECT * FROM produtos WHERE nome ILIKE '%book%';

-- Produtos da categoria "eletronicos" ou "livros"
SELECT * FROM produtos WHERE categoria IN ('eletronicos', 'livros');

-- Produtos SEM preço definido (NULL)
SELECT * FROM produtos WHERE preco IS NULL;
```

Joins

Joins servem para juntar tabelas em uma chave ou conjunto de chaves específico. Existem vários tipos de join que podem ser utilizados, como mostrado na figura.

```sql
SELECT u.nome, r.nome AS role
FROM   usuarios u
LEFT JOIN roles r ON r.id = u.role_id;
```



Sub-consultas com WITH

Sub-consultas são muito úteis quando se quer simplificar queries muito grandes. É possível fazer várias sub-consultas separando elas por vírgula e cada subconsulta pode usar o resultado da anterior.

```sql
WITH maiores AS (
  SELECT * FROM usuarios WHERE idade ⩾ 18
)
SELECT username, idade
FROM   maiores
ORDER  BY idade;
```

ATIVIDADES

Atividade 1

Criar um sistema de gerenciamento de tarefas

Tarefas:

- Crie as 3 tabelas com tipos de dados adequados.
- Relacione tarefas com usuários e categorias usando chaves estrangeiras.
- Insira pelo menos 3 usuários, 3 categorias e 5 tarefas usando inserts em batch.
- Faça uma consulta que retorne o título da tarefa, o nome do usuário e a categoria.
- Atualize o status de uma tarefa (concluída = TRUE).
- Delete todas as tarefas concluídas.
- Faça uma nova consulta mostrando apenas tarefas pendentes.

Requisitos:

- Uma tabela usuarios, que armazena informações básicas dos usuários.
- Uma tabela categorias, contendo diferentes tipos de categorias para tarefas.
- Uma tabela tarefas, que possui chave estrangeira para usuários e categorias.

Atividade 2

Fazer uma query em um banco de dados consolidado (pagila)

Crie uma consulta que mostre os 5 clientes que mais realizaram aluguéis, incluindo as seguintes informações:

- Nome completo do cliente
- E-mail
- Cidade e país onde o cliente mora
- Quantidade total de aluguéis realizados
- Quantidade de filmes diferentes alugados
- Última data em que realizou um aluguel
- Quantidade total gasta (somando amount da tabela payment)

Para baixar o pagila:

1. Baixe o repositório https://github.com/devrimgunduz/pagila
2. Crie um novo database no pgAdmin (Databases -> create -> database…)
3. Clique no novo database com o botão direito e aperte query tool
4. Abra o arquivo pagila-schema.sql e execute
5. Abra o arquivo pagila-insert-data.sql e execute

MUITO OBRIGADO


@saecomp.ec


saecomp@usp.br


saecomp.github.io


Prédio da Engenharia de Computação,
Campus 2, USP São Carlos

SAECOMP