Fundamentos DataBase

- Dados são elementos brutos
- Informações são conjuntos de dados tratados
- Já o banco de dados é um conjunto de dados relacionados e armazenados de forma estruturada.

Entidade x Relacionamento

- Atributos: campo, coluna, tipo
- Conceito A C I D

Get Started PostgreSQL

- Open-source (contribuições individuais e de empresas)
- Desenvolvido pela PostgreSQL Global Development Group
- Ferramentas: psql, pgAdmin
- Integração com diversos plugins (Diversas integrações com outras linguagens)
- Padrão SQL:2008
- Manutenção Online - non-blocking indexes
- SQL/JSON path

Criando database e iniciando pgAdmin

- Comando \l no sql shell mostra todos os bancos de dados
- Comando \du mostra os usuários
- Comando \c (nome do db) conecta no banco de dados
- Comando \dt lista as tabelas
- Dá pra usar os comandos que se usa no query do pgAdmin no SQL Shell

Introdução a PostgreSQL

```
CREATE TABLE tb_diretor(
id_diretor int NOT NULL,
nome_diretor varchar(50),
PRIMARY KEY (id_diretor)
);

SELECT * from tb_diretor;

CREATE TABLE tb_filme(
id_filme int NOT NULL,
nome_filme varchar(50),
id_diretor int NOT NULL,
id_produtora int NOT NUll,
genero varchar(25),
```

```
PRIMARY KEY (id_filme),
FOREIGn KEY (id_diretor) REFERENCES tb_diretor(id_diretor)
);
select * from tb_filme

CREATE TABLE tb_produtora (
id_produtora int NOT NULL,
nome_produtora varchar(25),
PRIMARY KEY ( id_produtora)
);

select * from tb_produtora;

ALTER TABLE tb_filme
ADD CONSTRAINT fk_id_produtora FOREIGN KEY (id_produtora) REFERENCES
tb_produtora(id_produtora);

select * from tb_filme
```

Inserindo valores nas tabelas

```
INSERT INTO tb_diretor
VALUES
(1,'Steven Spielberg'),
(2,'James Cameron'),
(3,'Quetin Tarantino');

select * from tb_diretor

INSERT INTO tb_produtora (id_produtora, nome_produtora) VALUES (1,'29th Century
Studios');
INSERT INTO tb_produtora (id_produtora, nome_produtora) VALUES (2,'Sony Picture');
INSERT INTO tb_produtora (id_produtora, nome_produtora) VALUES (3,'Paramount
Picture');

SELECT * FROM tb_produtora

INSERT INTO tb_filme
VALUES
(1, 'Django Livre',3,2,'Faroeste/Ação'),
(2, 'Avatar',2,1,'Ficção Científica'),
(3, 'O Resgate do Soldado Ryan',1,3,'Guerra');

select * from tb_filme

SELECT *
FROM tb_diretor
```

```
INNER JOIN tb_filme
ON tb_filme.id_diretor = tb_diretor.id_diretor;

select * from tb_filme
```

Dúvidas e Comentários

- LGPD (Lei geral de proteção de Dados)
- Existe comunicação entre os envolvidos em desenvolver o database, então não tem 2 pessoas manipulando um database ao mesmo tempo.

  ```
  SELECT *
      FROM tb_diretor
      INNER JOIN tb_filme
      ON tb_filme.id_diretor = tb_diretor.id_diretor;
  ```

  Aqui eu seleciono todas as colunas da tabela diretor, também selecionando as da tabela filme, mas fazendo uma relação entre elas, onde o **id_diretor** da tabela **tb_filme** é igual ao **id_diretor** da tabela **tb_diretor**, assim puxando as tuplas com o nome do diretor e seu id, juntamente aos filmes referentes a esse diretor (no caso referente a seu id que é uma chave estrangeira na tabela **tb_filme**).

  - O **Explain** pelo que deu a entender é um método de leitura que faz uma análise e mostra o jeito mais otimizado de realizar uma consulta.
    “ *O explain é uma ferramenta disponível em diversos bancos de dados que nos permite entender qual caminho o banco vai seguir para executar uma consulta que passamos a ele.* ”

    https://www.tumblr.com/tecsinapse/125165558784/entendendo-o-explain-do-postgres
  - Padrão Snake Case é bom de ser usado, enquanto o Java usa o Camel Case.
    Snake case - meu_bom_nome_de_variavel
    Camel case - meuBomNomeDeVariave

  - O curso tem uma aula de instalação no Ubuntu mas decidi não assistir