ACCION
Sistemas para Gestão e Inovação

amtech | Prefeitura de Maringá

software by maringá

PROGRAMA Saber TI


Módulo 2
AULA 3
PROGRAMA
Saber Ti

REVISANDO...

Agora que já entendemos sobre:

- Banco de dados
- Modelagem
- Instalação do banco de dados Postgresql
- Instalação do DBeaver

LINGUAGEM SQL

O que é?

- A linguagem **SQL** significa ***Structured Query Language*** *(Linguagem de Consulta Estruturada)*.

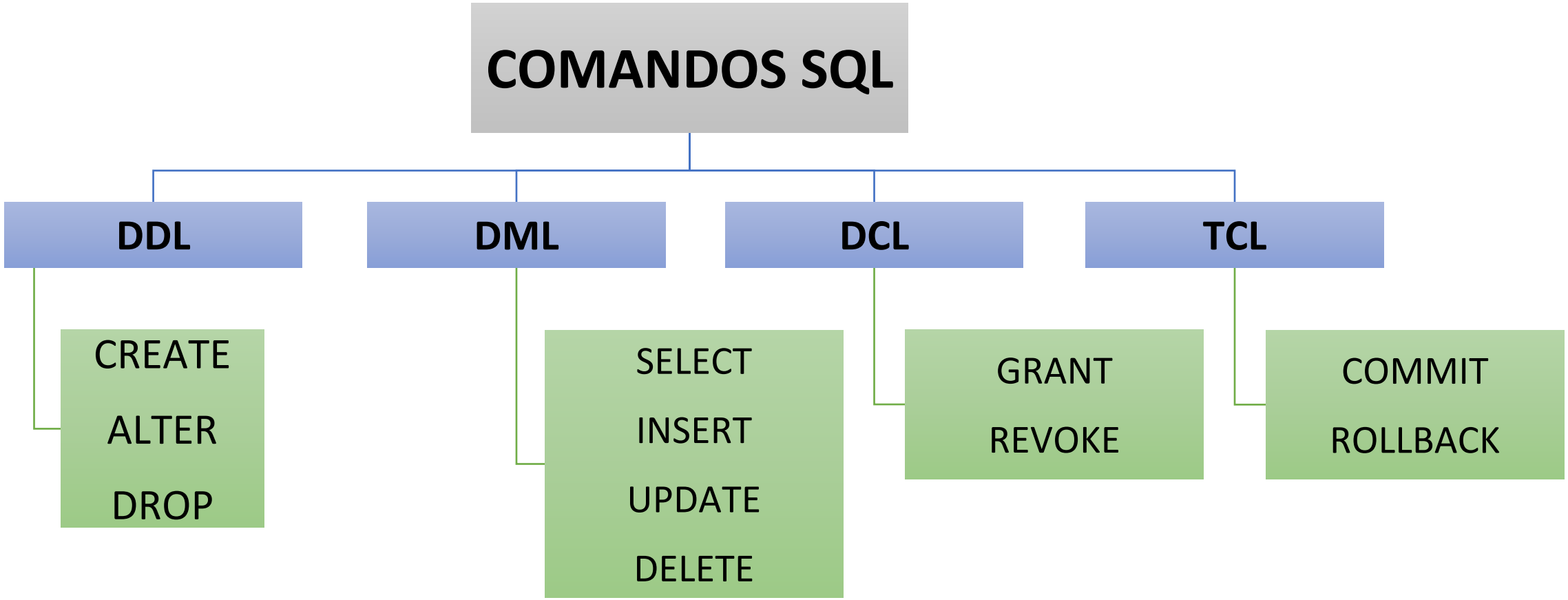
- Algumas características do SQL são baseados na álgebra relacional.

- Desenvolvido pela IBM no início dos anos 70.

- Permite ao administrador do sistema/Banco de dados uma agilidade no momento em que é feita a manipulação do dados inseridos em **Bancos de Dados Relacionais**.

LINGUAGEM SQL

Classificação dos Comandos SQL



COMANDOS DDL

LINGUAGEM SQL

Comandos DDL

- **Comando CREATE TABLE:**
  Permite a Criação de tabelas no Banco de dados.

Sintaxe:

```
CREATE TABLE NOME_DA_TABELA (
   NOME_CAMPO TIPO_DO_CAMPO);
```

LINGUAGEM SQL

Comando CREATE TABLE - Exemplos

```
CREATE TABLE CONTATO(
ID          INT PRIMARY KEY,
NOME        VARCHAR(30) NOT NULL,
NASCIMENTO DATE,
PESO        DECIMAL(10,2));

CREATE TABLE EMAIL (
ID          INTEGER PRIMARY KEY,
EMAIL       VARCHAR(60) NOT NULL,
CONTATO_FK INTEGER,
FOREIGN KEY (CONTATO_FK) REFERENCES CONTATO (ID));
```

LINGUAGEM SQL

Comando CREATE TABLE - Exemplos

```
CREATE TABLE CONTATO(
ID              INT PRIMARY KEY,
NOME            VARCHAR(30) NOT NULL,
NASCIMENTO      DATE,
PESO            DECIMAL(10,2));

CREATE TABLE EMAIL (
ID              INTEGER PRIMARY KEY,
EMAIL           VARCHAR(60) NOT NULL,
CONTATO_FK      INTEGER,
FOREIGN KEY (CONTATO_FK) REFERENCES CONTATO (ID));
```

Lembra na chave estrangeira que aprendemos na modelagem, então, é aqui onde criamos a referência para a outra tabela.

LINGUAGEM SQL

Comandos DDL

- **Comando ALTER TABLE:**
  Permite a alteração de campos em uma tabela.

Sintaxe:

```
ALTER TABLE NOME_DA_TABELA (
   ADD NOME_DO_CAMPO TIPO_DO_CAMPO);

ALTER TABLE NOME_DA_TABELA (
   DROP NOME_DO_CAMPO);
```

LINGUAGEM SQL

Comando ALTER TABLE - Exemplos

```
ALTER TABLE CONTATO ADD IDADE INTEGER;

ALTER TABLE CONTATO ADD CIDADE
VARCHAR(80);

ALTER TABLE CONTATO DROP CIDADE;
```

LINGUAGEM SQL

Comandos DDL

- **Comando DROP TABLE:**

Permite da tabela de forma completa, excluindo as colunas e linhas da tabela.

Sintaxe:

```
DROP TABLE NOME_DA_TABELA;
```

LINGUAGEM SQL

Comando DROP TABLE - Exemplos

```
DROP TABLE EMAIL;
```

```
DROP TABLE CONTATO;
```

No caso de criação de um script a ordem das tabelas importa.

COMANDOS DML

LINGUAGEM SQL

Comando INSERT

- **Comando INSERT:**

Permite a inclusão de linhas em uma tabela.

Sintaxe (INSERINDO OS CAMPOS, NA SEQUENCIA):

```
INSERT INTO NOME_TABELA VALUES (CONTEUDO_CAMPO1,
CONTEUDO_CAMPO2, CONTEUDO_CAMPO3...);
```

LINGUAGEM SQL

Comando INSERT - Exemplos

```
INSERT INTO CONTATO VALUES (1, 'Steve Jobs', '24/02/1955', 80.6);
INSERT INTO CONTATO VALUES (2, 'Mark Zuckerberg', '14/05/1984', 85);
INSERT INTO CONTATO VALUES (3, 'Bill Gates', '28/10/1955', 90);
INSERT INTO CONTATO VALUES (4, 'Elon Musk', '28/06/1971', 95);
```

LINGUAGEM SQL

Comando INSERT

- **Comando INSERT:**

Permite a inclusão de linhas em uma tabela.

Sintaxe (APONTANDO O CAMPO QUE DESEJA INSERIR):

```
INSERT INTO NOME_TABELA (CAMPO1, CAMPO2,
CAMPO3...) VALUES (CONTEUDO_CAMPO1,
CONTEUDO_CAMPO2, CONTEUDO_CAMPO3...);
```

LINGUAGEM SQL

Comando INSERT - Exemplos

```
INSERT INTO EMAIL (ID, EMAIL, CONTATO_FK) VALUES (1, 'jobs@gmail.com', 1);

INSERT INTO EMAIL (ID, EMAIL, CONTATO_FK) VALUES (2, 'esteve.jobs@hotmail.com', 1);

INSERT INTO EMAIL (ID, EMAIL, CONTATO_FK) VALUES (3, 'mark@facebook.com', 2);

INSERT INTO EMAIL (ID, EMAIL, CONTATO_FK) VALUES (4, 'zuck@gmail.com', 2);

INSERT INTO EMAIL (ID, EMAIL, CONTATO_FK) VALUES (5, 'bill@microsoft.com', 3);

INSERT INTO EMAIL (ID, EMAIL, CONTATO_FK) VALUES (6, 'gates@gmail.com', 3);

INSERT INTO EMAIL (ID, EMAIL, CONTATO_FK) VALUES (7, 'musk@gmail.com', 4);
```

LINGUAGEM SQL

Comando SELECT

- **SELECT**

Permite realizar a busca de dados em um banco de dados, podendo retornar todas as colunas e colunas específicas, além de permitir a criação de condições para que a consulta fique refinada.

Sintaxe:

```
SELECT NOME_DOS_CAMPOS FROM NOME_DA_TABELA;
```

*1) * -> Quando inserido após o select irá retornar todas as colunas da tabela.*

2) A leitura inicia pela pela clausulá FROM e depois vai para a cláusula SELECT.

LINGUAGEM SQL

Comando SELECT - Exemplos

```
SELECT * FROM CONTATO;
```

```
SELECT ID, NOME, NASCIMENTO, PESO FROM CONTATO;
```

Pode ser escrito em blocos:

```
SELECT
ID,
NOME,
NASCIMENTO,
PESO
FROM CONTATO;
```

LINGUAGEM SQL

Comando SELECT - Cláusula WHERE

- **WHERE**

A cláusula WHERE permite inserir condições nos comandos DML, onde os resultados ficam limitados a condição especificada. Pode usar em: SELECT, UPDATE, DELETE.

SINTAXE:

WHERE *CONDIÇÃO*

LINGUAGEM SQL

Cláusula WHERE - Exemplos

```
SELECT *
FROM CONTATO
WHERE IDADE > 50;

SELECT NOME
FROM CONTATO
WHERE NASCIMENTO >= '07/03/2022';

SELECT ID
FROM CONTATO
WHERE NOME = 'JOSÉ';
```

LINGUAGEM SQL

Cláusula WHERE - Exemplos Adicionando mais de uma Cláusula

UTILIZANDO AND:

```
SELECT NOME
FROM CONTATO
WHERE NASCIMENTO >= '07/03/2022'
AND IDADE >= 50;
```

UTILIZANDO OR:

```
SELECT NOME
FROM CONTATO
WHERE NASCIMENTO >= '07/03/2022'
OR IDADE >= 50;
```

LINGUAGEM SQL

Cláusulas

Cláusulas	Descrição
GROUP BY	Utilizada para separar os registros selecionados em grupos específicos.
HAVING	Utilizada para expressar a condição que deve satisfazer cada grupo.
ORDER BY	Utilizada para ordenar os registros selecionados com uma ordem especifica.
DISTINCT	Utilizada para selecionar dados sem repetição.

LINGUAGEM SQL

Funções de Agregação

Funções	Descrição
AVG	Utilizada para calcular a média dos valores de um campo determinado.
COUNT	Utilizada para devolver o número de registros da seleção.
SUM	Utilizada para devolver a soma de todos os valores de um campo determinado.
MAX	Utilizada para devolver o valor mais alto de um campo especificado.
MIN	Utilizada para devolver o valor mais baixo de um campo especificado.

LINGUAGEM SQL

Cláusula WHERE - Algumas funções:

Operador	Descrição
=	Igual
>	Maior que
<	Menor que
>=	Maior ou igual
<=	Menor ou igual
<> or !=	Diferente
AND	Operador Lógico AND
OR	Operador Lógico OR
IN	Retorna verdadeiro se um valor corresponder a qualquer valor em uma lista
BETWEEN	Retorna verdadeiro se um valor estiver entre um intervalo de valores
LIKE	Retorna verdadeiro se um valor corresponder a um padrão
IS NULL	Retorna verdadeiro se um valor for NULL
NOT	Negar o resultado de outros operadores

LINGUAGEM SQL

Comando DELETE

- **DELETE:**

Permite a exclusão das linhas de uma tabela.

Sintaxe:

```
DELETE FROM NOME_DA_TABELA;
```

ATENÇÃO:

NUNCA RODE DELETE SEM WHERE.

LINGUAGEM SQL

Comando DELETE - Exemplos

```
DELETE FROM CONTATO;
```

```
DELETE FROM CONTATO
WHERE NASCIMENTO >= '07/03/2022';
```

```
DELETE FROM CONTATO
WHERE NASCIMENTO >= '07/03/2022'
AND CIDADE = 'MARINGÁ';
```

LINGUAGEM SQL

Comando UPDATE

- **UPDATE:**

Permite a alteração do conteúdo dos campos das linhas de uma tabela.

Sintaxe:

```
UPDATE NOME_DA_TABELA
SET CAMPO_MODIFICADO = NOVO_CONTEUDO
WHERE CONDIÇÃO
```

ATENÇÃO:

NUNCA RODE UPDATE SEM WHERE.

LINGUAGEM SQL

Comando UPDATE - Exemplos

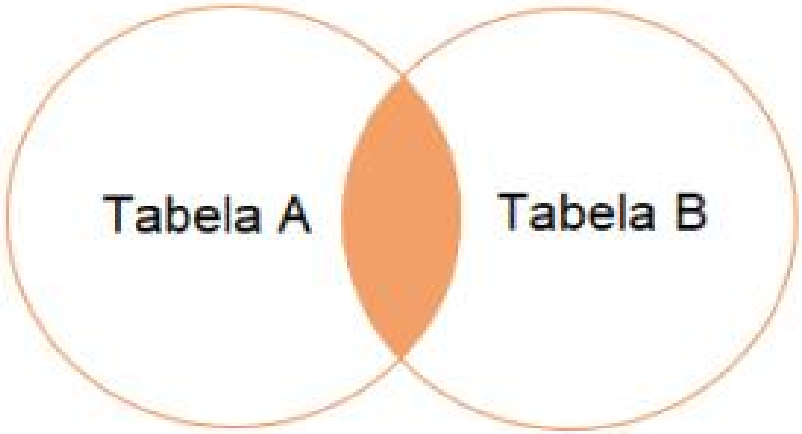
```
UPDATE CONTATO SET NOME = 'PEDRO' WHERE ID = 1;

UPDATE CONTATO
SET PESO = 85.5
WHERE ID = 2

UPDATE CONTATO
SET NOME = 'PEDRO',
    IDADE = 45
WHERE ID = 1;
```

LINGUAGEM SQL

JOIN



INNER JOIN ou JOIN

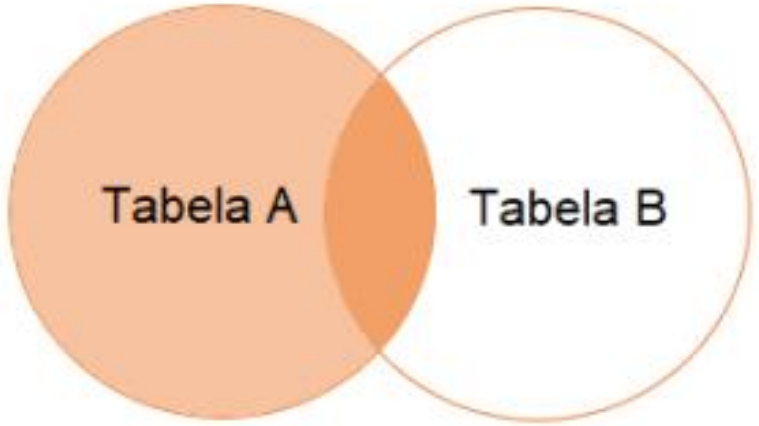
A cláusula INNER JOIN ou somente JOIN, é utilizada para cruzarmos registro trazendo apenas aqueles que existirem em ambas as tabelas.

Exemplo:

```
SELECT FUNC.NOMEFUNCIONARIO,
       MAIL.DESCRICAO
FROM FUNCIONARIO FUNC
INNER JOIN EMAIL MAIL
  ON FUNC.CODFUNC = MAIL.CODFUNC;
```

LINGUAGEM SQL

JOIN



LEFT JOIN

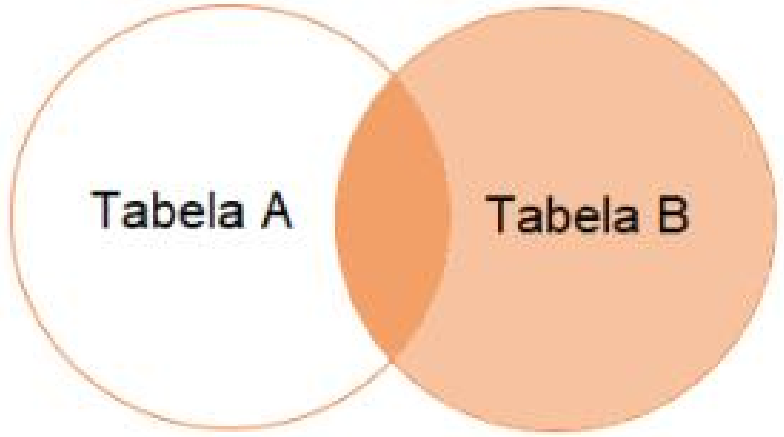
A cláusula LEFT JOIN, é utilizada para cruzarmos registro trazendo todos os da tabela declarada a esquerda e aqueles que existirem em ambas as tabelas.

Exemplo:

```
SELECT FUNC.NOMEFUNCIONARIO,
       MAIL.DESCRICAO
FROM FUNCIONARIO FUNC
LEFT JOIN EMAIL MAIL
  ON FUNC.CODFUNC = MAIL.CODFUNC;
```

LINGUAGEM SQL

JOIN



RIGHT JOIN

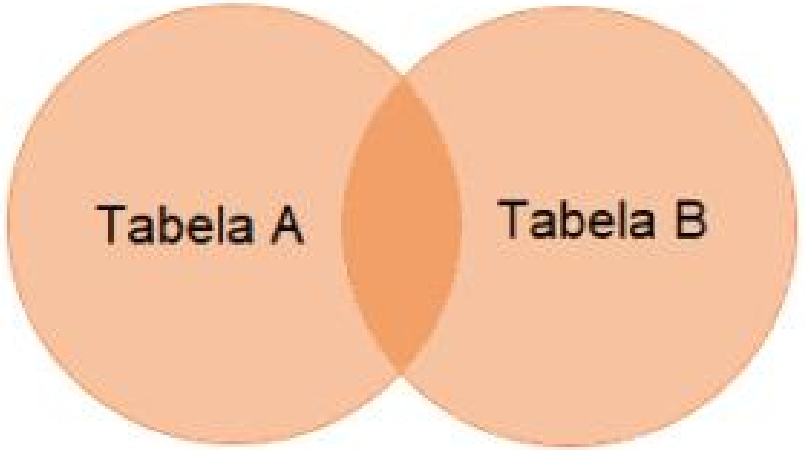
A cláusula RIGHT JOIN, é utilizada para cruzarmos registro trazendo todos os da tabela declarada a direita e aqueles que existirem em ambas as tabelas.

Exemplo:

```
SELECT FUNC.NOMEFUNCIONARIO,
       MAIL.DESCRICAO
FROM FUNCIONARIO FUNC
RIGTH JOIN EMAIL MAIL
  ON FUNC.CODFUNC = MAIL.CODFUNC;
```

LINGUAGEM SQL

JOIN



FULL JOIN

A cláusula FULL JOIN, é utilizada para cruzarmos registro trazendo os registros de ambas as tabelas, possuindo cruzamento ou não.

Exemplo:

```
SELECT FUNC.NOMEFUNCIONARIO,
       MAIL.DESCRICAO
FROM FUNCIONARIO FUNC
FULL JOIN EMAIL MAIL
  ON FUNC.CODFUNC = MAIL.CODFUNC;
```

LINGUAGEM SQL

DCL / TCL

- **DCL (Linguagem de Controle de Dados):**
  - **GRANT:** Comando que permite conceder e revogar privilégios em um banco de dados.
  - **REVOKE:** Comando utilizado para revogar (cancelar) privilégios em um banco de dados.

- **TCL (Linguagem de Controle de Transações)**
  - **COMMIT:** Comando utilizado para tornar permanente alguma alteração no banco de dados.
  - **ROLLBACK:** Comando utilizado realizar reversão, ou seja, é uma operação que retorna o banco de dados a algum estado anterior.

LINGUAGEM SQL

VAMOS TRABALHAR?

```
CREATE TABLE CATEGORIA_PRODUTO (
CATEGORIAPRODUTOID   INTEGER NOT NULL,
DS_CATEGORIA_PRODUTO VARCHAR(50) NOT NULL,
PRIMARY KEY(CATEGORIAPRODUTOID));

CREATE TABLE CLIENTE(
CLIENTEID          INTEGER NOT NULL,
TIPO_CLIENTE       CHAR(1) NOT NULL,
CPF_CNPJ_CLIENTE VARCHAR(18) NOT NULL,
NOME_CLIENTE       VARCHAR(100) NOT NULL,
PRIMARY KEY(CLIENTEID));
```

LINGUAGEM SQL

VAMOS TRABALHAR?

```
CREATE TABLE PRODUTO (
PRODUTOID                INTEGER NOT NULL,
CATEGORIAPRODUTOID       INTEGER NOT NULL,
DS_PRODUTO               VARCHAR(50) NOT NULL,
OBS_PRODUTO              VARCHAR(300) NULL,
VL_VENDA_PRODUTO         NUMERIC(15,2) NOT NULL,
DT_CADASTRO_PRODUTO      TIMESTAMP NOT NULL,
STATUS_PRODUTO           VARCHAR(10) NOT NULL,
PRIMARY KEY(PRODUTOID),
FOREIGN KEY (CATEGORIAPRODUTOID) REFERENCES
CATEGORIA_PRODUTO (CATEGORIAPRODUTOID));
```

LINGUAGEM SQL

VAMOS TRABALHAR?

```
CREATE TABLE ORCAMENTO(
ORCAMENTOID                 INTEGER NOT NULL,
CLIENTEID                   INTEGER NOT NULL,
DT_ORCAMENTO                TIMESTAMP NOT NULL,
DT_VALIDADE_ORCAMENTO TIMESTAMP NOT NULL,
VL_TOTAL_ORCAMENTO     NUMERIC(15,2) NOT NULL,
PRIMARY KEY(ORCAMENTOID),
FOREIGN KEY(CLIENTEID) REFERENCES CLIENTE(CLIENTEID));
```

LINGUAGEM SQL

VAMOS TRABALHAR?

```
CREATE TABLE ORCAMENTO_ITEM(
ORCAMENTOITEMID INTEGER NOT NULL,
ORCAMENTOID     INTEGER NOT NULL,
PRODUTOID       INTEGER NOT NULL,
QT_PRODUTO      NUMERIC(15,2) NOT NULL,
VL_UNITARIO     NUMERIC(15,2) NOT NULL,
VL_TOTAL        NUMERIC(15,2) NOT NULL,
PRIMARY KEY(ORCAMENTOITEMID),
FOREIGN KEY(ORCAMENTOID) REFERENCES ORCAMENTO
(ORCAMENTOID),
FOREIGN KEY(PRODUTOID) REFERENCES PRODUTO(PRODUTOID));
```

LINGUAGEM SQL

VAMOS TRABALHAR? SE DER TEMPO.....

```
CREATE TABLE USUARIOS(
ID              SERIAL PRIMARY KEY,
USUARIO         VARCHAR(30),
NOME_COMPLETO VARCHAR(60),
SENHA           VARCHAR(50));
```

LINGUAGEM SQL

VAMOS TRABALHAR?

TABELA: CATEGORIA_PRODUTO

INSERT INTO CATEGORIA_PRODUTO (CATEGORIAPRODUTOID, DS_CATEGORIA_PRODUTO) **VALUES** (1,'BASIC');
INSERT INTO CATEGORIA_PRODUTO (CATEGORIAPRODUTOID, DS_CATEGORIA_PRODUTO) **VALUES** (2,'PLATINUM');
INSERT INTO CATEGORIA_PRODUTO (CATEGORIAPRODUTOID, DS_CATEGORIA_PRODUTO) **VALUES** (3,'PREMIUM');
INSERT INTO CATEGORIA_PRODUTO (CATEGORIAPRODUTOID, DS_CATEGORIA_PRODUTO) **VALUES** (4,'ELITE');

TABELA: CLIENTE

INSERT INTO CLIENTE (CLIENTEID, CPF_CNPJ_CLIENTE, NOME_CLIENTE) **VALUES** (1,'943.051.830-56','JOSÉ DA SILVA');
INSERT INTO CLIENTE (CLIENTEID, CPF_CNPJ_CLIENTE, NOME_CLIENTE) **VALUES** (2,'802.202.930-07','PEDRO DE OLIVEIRA');
INSERT INTO CLIENTE (CLIENTEID, CPF_CNPJ_CLIENTE, NOME_CLIENTE) **VALUES** (3,'637.502.500-14','MARIA EDUARDA MEIRELES');
INSERT INTO CLIENTE (CLIENTEID, CPF_CNPJ_CLIENTE, NOME_CLIENTE) **VALUES** (4,'997.192.890-66','SANDRA GOMES');

LINGUAGEM SQL

VAMOS TRABALHAR?

TABELA: PRODUTO

INSERT INTO PRODUTO (PRODUTOID, CATEGORIAPRODUTOID , DS_PRODUTO, OBS_PRODUTO, VL_VENDA_PRODUTO, DT_CADASTRO_PRODUTO, STATUS_PRODUTO) **VALUES** (1, 1, 'CHAPA METALICA', 'CHAPA GENERICA', 106.22, '18/02/2022', 'ATIVO');
INSERT INTO PRODUTO (PRODUTOID, CATEGORIAPRODUTOID , DS_PRODUTO, OBS_PRODUTO, VL_VENDA_PRODUTO, DT_CADASTRO_PRODUTO, STATUS_PRODUTO) **VALUES** (2, 1, 'FOLHA METALICA', 'FOLHA GENERICA', 10.88, '18/02/2022', 'ATIVO');
INSERT INTO PRODUTO (PRODUTOID, CATEGORIAPRODUTOID , DS_PRODUTO, OBS_PRODUTO, VL_VENDA_PRODUTO, DT_CADASTRO_PRODUTO, STATUS_PRODUTO) **VALUES** (3, 3, 'CHAPA DOURADA', 'CHAPA ESPECIFICA', 158.88, '18/02/2022', 'ATIVO');
INSERT INTO PRODUTO (PRODUTOID, CATEGORIAPRODUTOID , DS_PRODUTO, OBS_PRODUTO, VL_VENDA_PRODUTO, DT_CADASTRO_PRODUTO, STATUS_PRODUTO) **VALUES** (4, 4, 'FOLHA DIAMANTE', 'FOLHA UNICA', 665.33, '18/02/2022', 'ATIVO');

LINGUAGEM SQL

VAMOS TRABALHAR?

TABELA: ORCAMENTO

```
INSERT INTO ORCAMENTO (ORCAMENTOID, CLIENTEID, DT_ORCAMENTO, DT_VALIDADE_ORCAMENTO, VL_TOTAL_ORCAMENTO) VALUES (1, 1, '18/02/2022', '19/02/2022', 10700.00);
INSERT INTO ORCAMENTO (ORCAMENTOID, CLIENTEID, DT_ORCAMENTO, DT_VALIDADE_ORCAMENTO, VL_TOTAL_ORCAMENTO) VALUES (2, 3, '17/02/2022', '19/02/2022', 2700.00);
INSERT INTO ORCAMENTO (ORCAMENTOID, CLIENTEID, DT_ORCAMENTO, DT_VALIDADE_ORCAMENTO, VL_TOTAL_ORCAMENTO) VALUES (3, 4, '18/02/2022', '19/02/2022', 1200.00);
INSERT INTO ORCAMENTO (ORCAMENTOID, CLIENTEID, DT_ORCAMENTO, DT_VALIDADE_ORCAMENTO, VL_TOTAL_ORCAMENTO) VALUES (4, 2, '17/02/2022', '19/02/2022', 8000.00);
```

LINGUAGEM SQL

VAMOS TRABALHAR?

TABELA: ORCAMENTO_ITEM

INSERT INTO ORCAMENTO_ITEM (ORCAMENTOITEMID, ORCAMENTOID, PRODUTOID, QT_PRODUTO, VL_UNITARIO, VL_TOTAL) **VALUES** (1, 1, 1, 10, 100.00, 1000.00);
INSERT INTO ORCAMENTO_ITEM (ORCAMENTOITEMID, ORCAMENTOID, PRODUTOID, QT_PRODUTO, VL_UNITARIO, VL_TOTAL) **VALUES** (2, 1, 2, 20, 10.00, 200.00);
INSERT INTO ORCAMENTO_ITEM (ORCAMENTOITEMID, ORCAMENTOID, PRODUTOID, QT_PRODUTO, VL_UNITARIO, VL_TOTAL) **VALUES** (3, 1, 4, 30, 500.00, 1500.00);
INSERT INTO ORCAMENTO_ITEM (ORCAMENTOITEMID, ORCAMENTOID, PRODUTOID, QT_PRODUTO, VL_UNITARIO, VL_TOTAL) **VALUES** (4, 1, 3, 40, 200.00, 8000.00);

Continua...

LINGUAGEM SQL

VAMOS TRABALHAR?

TABELA: ORCAMENTO_ITEM

INSERT INTO ORCAMENTO_ITEM (ORCAMENTOITEMID, ORCAMENTOID, PRODUTOID, QT_PRODUTO, VL_UNITARIO, VL_TOTAL) **VALUES** (5, 2, 1, 10, 100.00, 1000.00);
INSERT INTO ORCAMENTO_ITEM (ORCAMENTOITEMID, ORCAMENTOID, PRODUTOID, QT_PRODUTO, VL_UNITARIO, VL_TOTAL) **VALUES** (6, 2, 2, 20, 10.00, 200.00);
INSERT INTO ORCAMENTO_ITEM (ORCAMENTOITEMID, ORCAMENTOID, PRODUTOID, QT_PRODUTO, VL_UNITARIO, VL_TOTAL) **VALUES** (7, 2, 4, 30, 500.00, 1500.00);
INSERT INTO ORCAMENTO_ITEM (ORCAMENTOITEMID, ORCAMENTOID, PRODUTOID, QT_PRODUTO, VL_UNITARIO, VL_TOTAL) **VALUES** (8, 3, 1, 10, 100.00, 1000.00);
INSERT INTO ORCAMENTO_ITEM (ORCAMENTOITEMID, ORCAMENTOID, PRODUTOID, QT_PRODUTO, VL_UNITARIO, VL_TOTAL) **VALUES** (9, 3, 2, 20, 10.00, 200.00);
INSERT INTO ORCAMENTO_ITEM (ORCAMENTOITEMID, ORCAMENTOID, PRODUTOID, QT_PRODUTO, VL_UNITARIO, VL_TOTAL) **VALUES** (10, 4, 3, 40, 200.00, 8000.00);

LINGUAGEM SQL

VAMOS TRABALHAR?

Agora, vá no link abaixo:

http://dontpad.com/saberti

Atividade 1:

Por meio da (DDL) Linguagem de Definição de Dados, vamos criar tabelas no banco (**DBeaver**).

Atividade 2:

Fazendo uso da (DML) Linguagem de Manipulação de Dados vamos alimentar nossas tabelas.

LINGUAGEM SQL

ATIVIDADES DE FIXAÇÃO

Vamos realizar as seguintes consultas:

1) Dados de produto com status ATIVO.
2) Descrição do produto + descrição da categoria do produto.
3) Dados de produtos onde o valor de venda é maior do que 50,00.
4) Dados de orçamento + nome do cliente.
5) Dados de item do orçamento + nome do produto + nome da categoria.
6) Orçamentos feitos entre 01/02/2022 e 18/02/2022.
7) Orçamentos vencidos (utilizando data atual CURRENT_DATE).
8) Maior valor de item orçamento (por produto).
9) Valor do item mais caro de todos os orçamentos.
10) Soma do valor de todos orçamentos.
11) Quantidade total de itens (por orçamento).

REFERÊNCIAS

- www.postgresql.org

- http://paulohcc.com/joins-sql-vamos-aprender/