**Curso Udemy – SQL**

**Banco de Dados** = É um conjunto organizado de dados, tipicamente armazenado em formato eletrônico. É um local de armazenamento de informações.

**Tipos de Bancos de Dados**

\*Bancos de dados de tipo plano (flat-type databes)

\*Bancos de dados hierárquicos (Hierarchial databases)

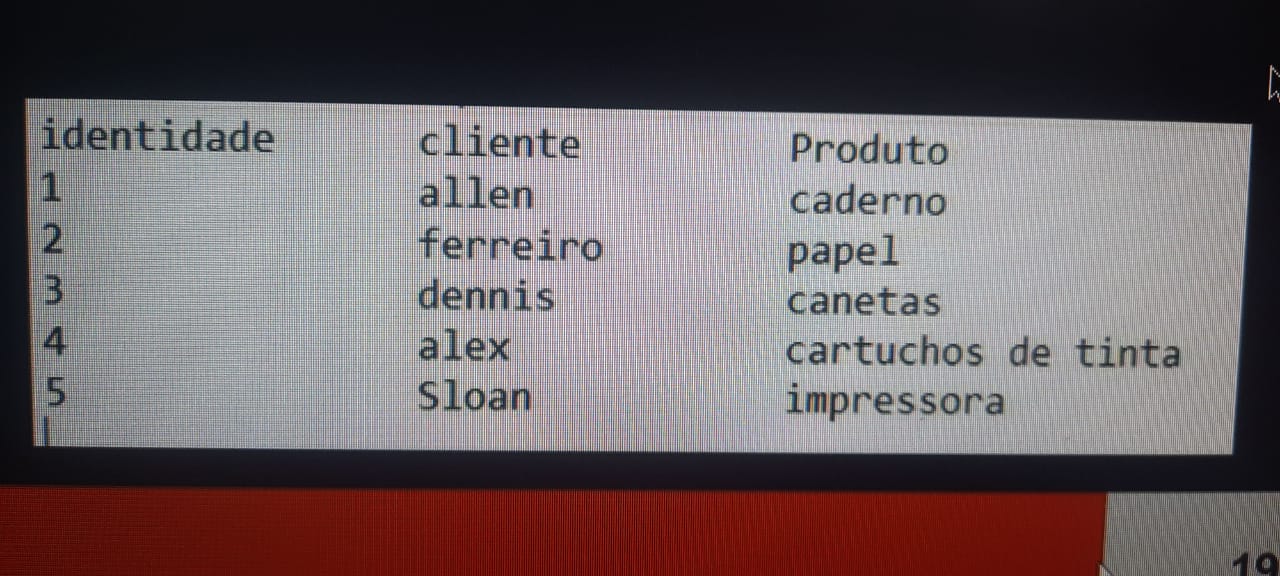
\*Bancos de dados relacionais (Relational databases)

Observação: Cada tipo de Banco possui seus próprios recursos de desing importantes.

**1.Flat-type databases**

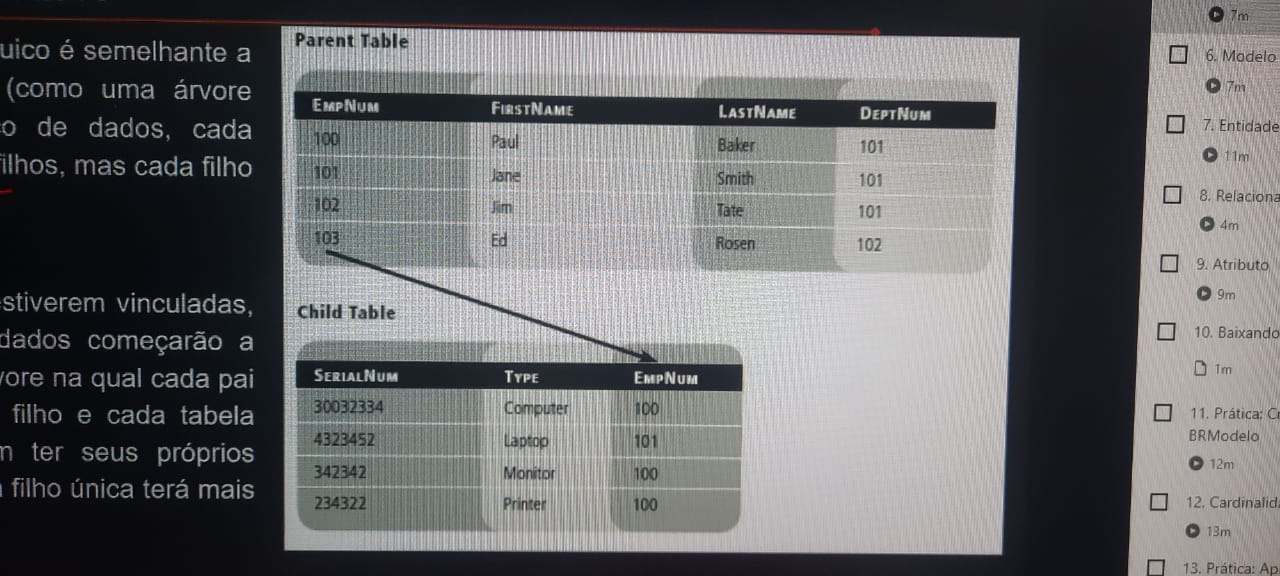
Bancos de Dados de tipo plano são simplistas em design. Eles são mais comumente usados em formatos de textos simples. Como o proposito deles é manter um registro por linha, eles tornam o acesso, o desempenho e as consultas muito rápidos.

Exemplo desse Banco : Seria em arquivo .txt, .csv ou .ini.

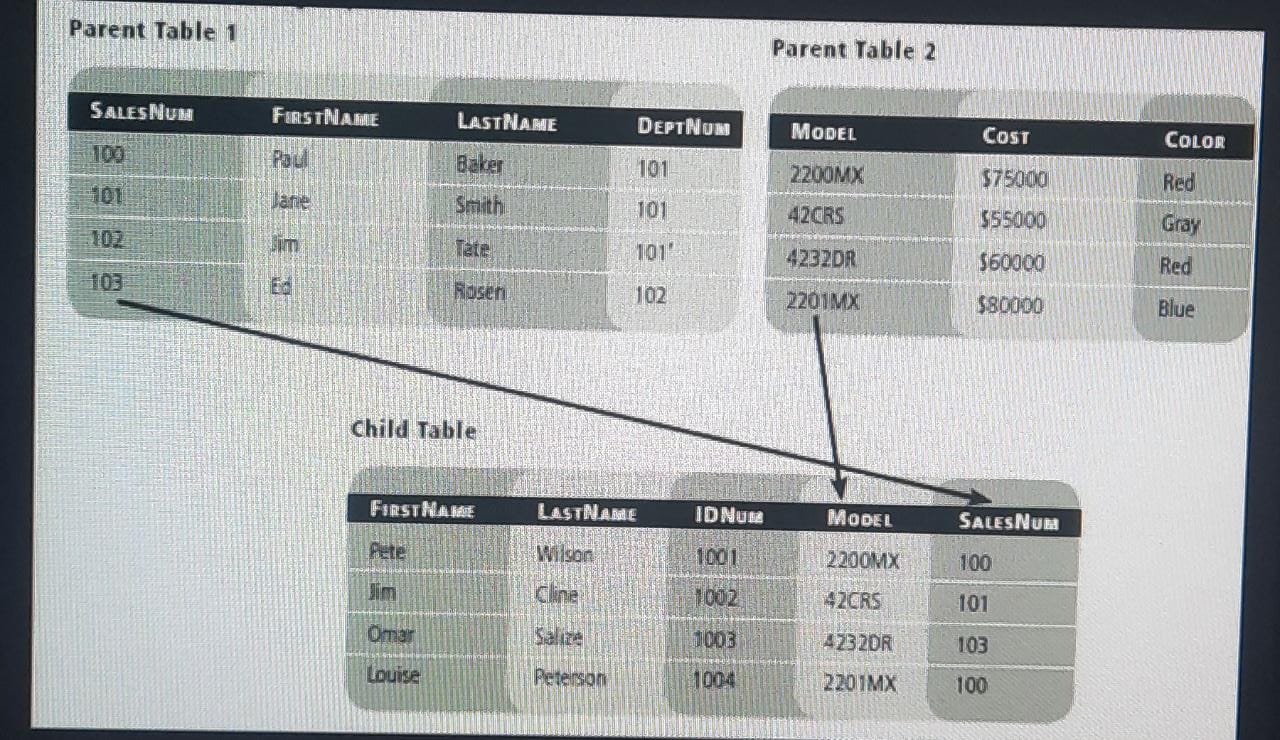
Bancos de dados desse tipo são considerados “planos” porque são tabelas bidimensionais consistindo de linhas e colunas.

**2.Hierarchical databases**

Um bancos de dados hierárquico é semelhante a uma estrutura em árvore (como uma árvore genealógica). Nesse banco de dados , cada tabela “pai” pode ter vários filhos, mas cada filho pode ter apenas um pai.



**3.Relational databases**

É um banco semelhante a um banco de dados hierárquico. Diferentemente dos bancos de dados hierárquicos, no banco relacional, uma tabela em um banco de dados relacional pode ter vários pais.

**Modelos de Dados**

Um modelo de dados mostra a estrutura lógica de um banco de dados, incluindo as relações e restrições que determinam como os dados podem ser armazenados e acessados.

**Categoria dos tipos de dados do SQL Server**

No SQL Server, cada coluna, variável, expressão e parâmetro tem um tipo de dados relacionado. O tipo de dado especifica o que pode ser colocado no objeto. A integridade do banco de dados depende decisivamente dos tipos de dados com escopo apropriado.

Os tipos de dados que o objeto pode assumir são: dados inteiros, dados de caractere, dados monetários, data e hora, cadeias de caracteres binárias etc.

\* Os tipos de dados em SQL Server são organizados nas seguintes categorias:

- **Númericos**:

-Númericos exatos

# bigint # numeric # bit # smallint # decimal

# smallmoney # int # tinyint # money

-Númericos aproximados

# float # Real

-**String**

-Cadeias de caracteres

# char #vachar #text

-Cadeias de caracteres Unicode

# nchar #nvchar # ntext

-Cadeia de caracteres binária

# binary # varbinary # imagem

**-Data e hora**

# date # datetimeoffset # datetime2

# smalldatetime # datetime # time

**-Outros tipos de dados**

# cursor # rowversion # hierarchyid

# uniqueidentifier # sql\_variant # xml

# Tipos de geometria espacial

# Tipos de geografia espacial # Table

\***Conhecer mas os dados** = <https://docs.microsoft.com/pt-br/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-2017>

**SQL**

Structured Query Language ou linguagem de Consulta Estrutura ou SQL, é a linguagem de pesquisa declarativa padrão para banco de dados relacional (base de dados relacional). Muita das características originais do SQL forma inspiradas na álgebra relacional. Tipicamente a linguagem SQL ANSI é a linguagem de SQL padrão.

Pode ser migrada de plataforma, para plataforma sem mudanças estruturais principais.

**Variações de SQL ANSI**

SQL Server: Transact -SQL

Oracle: PL/SQL

PostegreSQL: PL/pgSQL

**Grupos de comandos do SQL**

A linguagem de SQL se subdivide em cinco grupos de comandos. Alguns autores apenas consideram 3 ou 4 grupos. Cada grupo tem seus comandos específicos, tendo cada um uma finalidade específica.

\*DDL

- USE

-CREATE

-ALTER

-DROP

-TRUNCATE

\*DML

-INSERT

-UPDATE

-DELETE

-MERGE

\*DCL

-GRENT

-REVOKE

-DENY

\*DTL

-BEGIN

TRANSACTION

-COMMIT

-ROLLBACCK

\*DQL

-SELECT

Observação:

- O comando SELECT para prova é designado como DML;

**Definição dos Comandos**

**DDL – DATA DEFINITION LANGUAGE**

O DDL é usado para criar, alterar e deletar objetos do banco de dados. Podendo ser Banco de Dados, tabelas, views, procedures, funtions, index, e etc.

Principais Comandos:

- USE = Muda o contexto da janela de consulta;

- CREATE = Criar objetos no banco de dados;

- ALTER = Altera objetos no banco;

- DROP = Apaga objetos do banco;

- TRUNCATE = Ele é DML também , Apaga todos os registros de uma tabela, reiniciando o índice das colunas autoincrementais.

**Diferenças entre os comandos SQL Server DELETE e TRUNCATE**

\*Truncar redefine os valores de identidade, enquanto a exclusão não;

\*Truncar remove todos os registros e não aciona gatilhos;

\*Truncar é mas rápido do que excluir, pois faz menos uso dos log da transação;

\*Truncar não é possível quando uma tabela é referenciada por uma Chave Estrangeira (Foreing Key) ou tabelas são usadas na replicação ou com visualizações indexadas.

**DML – DATA MANIPULATION LANGUAGE**

O DML é usado para a manipulação de registros do banco de dados.

Seus comandos são:

-INSERT = Insere registros no bancos;

-UPDATE =Atualiza registros no banco.

-DELETE = Deleta/Apaga registro no banco.

-MERGE = Executa operações de inserção, atualização ou exclusão em uma tabela de destino com base nos resultados de uma junção com uma tabela de origem.

Obs.: Para a certificação MTA Database Fundamentais o comando SELECT é caracterizado como um comando DML (Verificar o livro oficial).

**DCL – DATA COTROL LANGUAGE**

O DCL controla a segurança dos dados, definido quem tem acesso a determinado tipo de operação.

Seus comandos são:

-GRANT = Usado para a permissão a um determinado usuário para executar um procedimento.

-REVOKE = Desfaz uma permissão , seja uma GRANT ou DENY.

-DENY = Negar permissão de acessos de dados.

**DTL – DATA TRANSACTION LANGUAGE**

O DTL é usado para a manipulação de transações (inserção, alteração, remoção e consulta) no banco de dados.

Seus comandos são:

-BEGIN TRAN OU (BEGIN TRANSACTION) = Utilizado para iniciar uma transação.

-COMMIT = Utilizado para gravar as transações no banco.

-ROLLBACK = Utilizado para desfazer as alterações de uma transação no banco.

Obs1: Alguns SGBDs são auto-commit, ou seja, uma vez executado comandos do tipos DML os dados são gravados automaticamente no banco de dados.

Obs2: Ao realizar a abertura de uma transação com o BEGIN TRAN, a transação deve ser finalizada com os comandos COMMIT ou ROLLBACK.

Alguns sistemas travam quando uma transação é aberta e não é finalizada.

**DQL – DATA QUERY LANGUAGE**

O DQL é usado para a consultar dados no banco.

Seu comando é:

-SELECT = Usado para consultar informações ao registro das tabelas do banco de dados.

**Dicas sobre os comandos SQL**

É importante criar um padrão de codificação e nomenclatura dos objetos do banco de dados. Esse processo facilita a leitura do código e futuras manutenções.

Dicas:

- Os tipos de dados string (texto) e data são delimitados por aspas simples.

- Ex: ‘João’

- Ex: ‘2019-01-01’

- Comentários no SQL Server

- Linha: --

- Bloco: /\*\*/

- Dependendo do idioma que o SQL Server foi configurado a data pode vir no formato ANO-MÊS-DIA ou DIA-MÊS-ANO.

- Virgula na linguagem SQL é separador de campos.

- Ex.: Campo1, campo2, campo3, ....

- Para dados números utilizar o ponto para separar os valores.

- Ex.: 149.99

- No comando SELECT o asterisco (\*) significa que todos os campos da tabela serão exibidos.

**Comandos – DDL**



- **CREATE**

- Sintaxe:

- CREATE OBJETO NOME\_OBJETO;

- Criando banco de dados

- CREATE DATABASE BD\_DDL;

- **USE**

- Carregando o banco de dados no contexto da janela de consulta do SQL Server

Sintaxe:

- USE BD\_DDL

-Criando uma Tabela

CREATE TABLE TB\_DDL(MATRICULA INT PRIMARY KEY, NOME VARCHAR(100), SEXO CHAR(1))

**-ALTER**

Sintaxe:

- ALTER OBJETO NOME\_OBJETO;

Alterando a tabela, adicionando um novo campo

Sintaxe:

- ALTER TABLE TB\_DDL ADD DTNASC DATETIME, CPF CHAR(11)

Alterando a tabela, alterando o tipo de dados campo SEXO de CHAR(1) para VACHAR (15)

Sintaxe:

- ALTER TABLE TB\_DDL

- ALTER COLUMN DTNASC DATE

Alterando a tabela, excluindo o campo SEXO.

Sintaxe:

- ALTER TABLE TB\_DDL DROP COLUMN SEXO;

**- DROP**

Sintaxe

- DROP OBJETO NOME\_OBJETO

Apagando tabela do banco de dados

Sintaxe

- DROP TABLE TB\_DDL

Obs.: Se existir ligação de chave estrangeira não será possível deletar a tabela.

Apagando Objetos do banco de dados

- DROP VIEW VW\_TESTE

- DROP FUNCTION FN\_TESTE

- DROP PROCEDURE USP\_TESTE

- DROP TRIGGER TR\_TESTE

**- TRUNCATE**

Sintaxe

- TRUNCATE TABLE NOMETABELA;

Apagando todos os registro da tabela

- TRUNCATE TABLE TB\_DDL

**Comandos – DML**



**- INSERT**

Sintaxe:

Maneira 1

- INSERT INTO TABELA (CAMPOS) VALUES (VALORES);

Maneira 2

- INSERT INTO NOMETABELA VALUES (VALORES,VALORES)

Obs.: Quando não se especifica os campos na hora do insert, deve-se assumir a ordem de criação dos mesmo na tabela.

Inserindo um registro especificando os campos

- INSERT INTO TB\_TML ( MATRICULA, NOME, DTNASC, SEXO) VALUES (1, ‘PESSOA\_A’, ‘2000-12-10’, ‘M’)

Utilizando o mesmo insert para inserir vários registros

- INSERT INTO TB\_DML (MATRICULA, NOME, DTNASC, SEXO) VALUES (2, ‘PESSOA\_B’, ‘2000-12-10’, ‘M’), (3, ‘PESSOA\_B’, ‘2000-01-01’, ‘F’)

Inserindo um registro na tabela sem especificar os campos. Lembrar da ordem de inserção das colunas no momento da criação da tabela.

Inserindo um registro nas tabelas sem especificar os campos. Lembrar da ordem de inserção das colunas no momento da criação das tabelas.

- INSERT TB\_DML VALUES (4, ‘PESSOA\_D’, ‘2000-01-10’, ‘M’)

Inserindo dados em uma tabela que foi resultado de n x n.

- INSERT INTO LISTA\_COMPRAS (COD\_PRODUTO, CODNF, QTD) VALUES(‘PRD0001’, ‘NF000001’, 3)

**UPDATE**

Sintaxe:

- UPDATE TABELA SET CAMPO = NOVO\_VALOR WHERE CONDICAO = VALOR

Obs.: Sempre utilizar filtros para utilizar este comando

Atualizando um registro de um campo de uma tabela funcionário

Sintaxe:

- UPDATE TB\_DML SET SEXO = ‘F’ WHERE NOME = ‘PESSOA\_B’

Atualizando dois campos de um registro de uma tabela

Sintaxe:

- UPDATE TB\_DML SET SEXO = ‘F’, DATNASC = ‘1997-07-01’ WHERE NOME = ‘PESSOA\_D’

**DELETE**

Sintaxe:

- DELETE FROM TABELA CONDICAO = VALOR

Obs.: Sempre utilizar filtros para utilizar este comando

Apagando registros da tabela funcionário

Sintaxe:

- DELETE FROM DB\_DML WHERE NOME = ‘PESSOA\_C’

**Comando – DQL**



**SELECT**

Sintaxe

- SELECT \* FROM TABELA ou VIEW ou FUNCTION

Selecionado todos os dados da tabela

Sintaxe

- SELECT \* FROM TB\_DML

Selecionando apenas alguns campos da tabela

Sintaxe

- SELECT NOME, DATNASC, FROM TB\_DML WHERE SEXO = ‘F’

Selecionando todos os campos da tabela filtrando por várias informações

Sintaxe:

- SELECT \* FROM TB\_DML WHERE DATNASC > ‘1999-10-25’ AND SEXO = ‘F’

**Comando – DTL**

**BEGIN TRANSACTION / COMMIT / ROLLBACK**

Sintaxe:

- BEGIN TRAN / BEGIN TRANSACTION

COMANDO SQL

COMMIT OU ROLLBACK

Abrindo uma transação no banco de dados para utilização do comando UPDATE

Sintaxe:

- BEGIN TRAN

UPDATE TB\_DML SWET SEXO = ‘M’ WHERE NOME = ‘PESSOA\_D’

COMMIT : Salvará a alteração na base de dados definitivamente e fechará a transação.

ROLLBACK : Irá desfazer a alteração realizada e fechará a transação.

**Comando – DCL**



**GRANT / REVOKE / DENY**

Sintaxe:

- GRANT / REVOKE / DENY <Privilegio>

ON OBJETO\_BD TO <Usuario>.

Dando permissão para o usuário executar o comando SELECT na tabela.

Sintaxe:

- CREATE USER USER\_DCL WITHOUT LOGIN;

-GRANT SELECT, INSERT ON DBO.TB\_DML TO USER\_DCL

Desfazendo permissão dada do comando GRANT

Sintaxe:

- REVOKE INSERT ON DBO.TB\_DML TO USER\_DCL

Negando permissão do usuário dar SELECT na tabela

Sintaxe:

- DENY SELECT ON DBO.TB\_DML TO USER\_DCL

Executando consulta com usuário criado

Sintaxe:

- EXECUTE AS USER = ‘USER\_DCL’ SELECT \* FROM TB\_DML REVERT;

Lista de Privilégios

SELECT – Capacidade de executar instruções SELECT na tabela.

INSERT – Capacidade de executar instruções INSERT na tabela.

UPDATE – Capacidade de executar instruções UPDATE na tabela.

DELETE – Capacidade de executar instruções DELETE na tabela.

REFERENCES – Capacidade de criar uma restrição que se refere à tabela.

ALTER – Capacidade de executar instruções ALTER TABLE para alterar a definição da tabela.

ALL – ALL não concede todas as permissões para a tabela. Em vez disso, concede as permissões ANSI-92 que são SELECT, INSERT, UPDATE, DELETE, e REFERENCES.

**CONTRAINTS**

**IDENTITY – EXTRA**

- Faz com o atributo selecionado seja incrementado automaticamente, ou seja, irá gerar uma numeração automática sem que seja necessário gerar o valor no insert.

- É indicado ser utilizado com campos do tipo inteiro.

- No identity você pode informar qual o inicio da operação e quando será o incremento. Caso não seja informado implicitamente o SQL SERVER aplica o valor (1,1). Iniciando o número 1 e incrementando 1.

- Criando uma tabela com o campo incremental.

Sintaxe:

CREATE TABLE tb\_identity( cod INT IDENTITY PRIMARY KEY, NOME VACHAR(50))

- Criando uma tabela começando a numeração a partir do número 1000 e incrementando de 1 em 1.

Sintaxe:

CREATE TABLE tb\_identity( cod INT IDENTITY (1000,1), NOME VACHAR(50))

- Habilitando e desligando uma Identity

Sintaxe:

SET IDENTITY\_INSERT tb\_identity ON – Desliga a identidade

SET IDENTITY\_INSERT tb\_identity OFF – Habilita a identidade

**PRIMARY KEY – PK**

- Restrição que identifica o atributo com identificador da tabela.

- Quando uma tabela possui mais de uma chave primária , é chamada de chave composta.

- Ao criar uma tabela sempre definir um campo com chave primária.

- Criando chave primária de uma tabela com cosntraint

Sintaxe:

- CREATE TABLE tb\_pk ( cod INT CONSTRAINT pk\_teste PRIMARY KEY, NOME Vachar(50))

-Criando uma chave primária no final do comando. Deve-se referenciar os campos nos parênteses.

Sintaxe:

- CREATE TABLE tb\_pk ( Cod INT, Nome VACHAR(50), PRIMARY KEY (cod)).

- Criando tabela com duas chaves primárias

Sintaxe:

- CREATE TABLE tb\_pk ( Cod INT, Cod2 INT, Nome VACHAR(50), PRIMARY KEY (cod1,cod2))

- Adicionando chave primária, após criação de uma tabela sem pk.

Sintaxe:

- ALTER TABLE tb\_pk ADD PRIMARY KEY (cod);

- Adicionando chave primária , após a criação de uma tabela sem pk, informando o nome da CONSTRAINT.

Sintaxe:

- ALTER TABLE tb \_pk ADD CONSTRAINT pk\_test PRIMARY KEY (cod)

- Apagando uma CONSTRAINT PK

Sintaxe:

- ALTER TABLE tb\_pk DROP CONSTRAINT pk\_teste;

**Foreing Key – Fk**

Restrição que define que um atributo A faz referência ao atributo B que é chave de outra tabela.

A Chave Estrangeira é definida no momento da normalização.

O atributo deve ser do mesmo tipo do campo referência de sua tabela de origem.

O atributo (FK) deve ser referenciado a qual atributo da tabela de origem ele faz correlação.

Para a chave estrangeira funcionar, a tabela que contém o atributo chave deve estar criada.

Para um registro ser referenciado na tabela com a FK, o registro tem que existir na tabela de origem.

- Criando chave estrangeira na criação da tabela com constraint.

Sintaxe:

- CREATE TABLE tb\_fk\_marca ( cod INT PRIMARY KEY IDENTITY, Nome VARCHAR(50) )

- CREATE TABLE tb\_fk\_carro( Placa varchar (6) PRIMARY KEY, modelo VARCHAR(50), fk\_marca INT FOREING KEY REFERENCES test\_fk\_marca(Cod) )

- Adicionado uma chave estrangeira após criação de tabela. Supondo que o campo não foi referenciado com fk.

Sintaxe:

- ALTER TABLE tb\_fk\_carro ADD FOREING KEY (fk\_marca) REFERENCES tb\_fk\_marca (COD);

- Adicionando uma chave estrangeira com constraint após criação de tabela. Supondo que o campo não foi referenciado como fk.

Sintaxe:

- ALTER TABLE tb\_fk\_carro ADD CONSTRAINT fk\_marca\_carro FOREING KEY (fk\_marca) REFERENCES tb\_fk\_marca (COD);

- Apagando constraint

Sintaxe:

- ALTER TABLE tb\_fk\_carro DROP CONSTRAINT fk\_marca\_carro

**UNIQUE**

Restrição que garante que o valor de um determinado campo não se repita dentro de uma tabela, ou seja, os valores são únicos.

A restrição PRIMARY KEY possui uma restrição UNIQUE implícita.

Em uma tabela pode haver várias restrições Unique, porém somente uma restrição chave de PK por tabela.

- Criando tabela com a restrição UNIQUE

Sintaxe:

- CREATE TABLE tb\_unique( cod INT PRIMARY KEY IDENTITY, Nome VARCHAR (50), CPF CHAR(11) CONSTRANIT UQ\_CPF UNIQUE)

- Adicionando unique em um campo já criado

Sintaxe:

- ALTER TABLE tb\_unique ADD UNIQUE (cpf);

- Adicionando unique em um campo já criado especificando o nome da cosntraint.

Sintaxe:

- ALTER TABLE tb\_unique ADD CONSTRAINT UQ\_CPF UNIQUE (cpf);

- Apagando uma constraint

- ALTER TABLE tb\_unique DROP CONSTRAINT UQ\_CPF;

**NOT NULL**

Not Null é a restrição que impede a coluna de receber valores nulos no momento da inserção ou atualização.

O SQL Server coloca como padrão NULL quando não é informado se o campo não pode receber valores nulos.

Boa prática: Sempre explicitar se o campo é NULL ou NOT NULL na criação da tabela.

- Criando tabela com campos são nulos e campos nulos

Sintaxe:

- CREATE TABLE tb\_notnull( Cod INT PRIMARY KEY IDENTITY, Nome VARCHAR (50) NOT NULL, DT DATE NULL )

- Colocando not null em um campo já criado

Sintaxe:

- ALTER TABLE tb\_notnull ALTER COLUMN DT DATE NOT NULL

- Removendo o not null em um campo já criado

Sintaxe:

- ALTER TABLE tb\_not null ALTER COLUMN DT DATE NULL

**DEFAULT**

Restrição que coloca um valor como padrão em uma determinada coluna.

Ao realizar um Insert E o valor de um determinado campo não for informado, a restrição Default ira preencher o campo o valor padrão definido pela Constraint

- Criando tabela com a restrição default

Sintaxe

- CREATE TABLE tb\_default(Cod INT PRIMARY KEY IDENTITY, Nome VARCHAR(50), dtInsert DATE CONSTRAINT DF\_DATA DEFAULT (GETDATE())

- Adicionando default em uma coluna já criada

Sintaxe

- ALTER TABLE tb\_default ADD CONSTRAINT df\_dt\_insert DEFAULT getdate() FOR dtInsert

- Apagando constraint

Sintaxe

- ALTER TABLE tb\_default ALTER COLUMN dtInsert DROP DEFAULT

**CHECK**

A restrição CHECK é usada para limitar os valores que pode ser inseridos em uma coluna.

Se você definir uma restrição CHECK em uma única coluna, ela permitirá apenas inserir determinados valores que atendam a lógica de checagem.

- Criando tabela com o CHECK

Sintaxe

- CREATE TABLE tb\_check ( Cod INT PRIMARY KEY IDENTITY, NOME VARCHAR(50) NOT NULL, VALOR NUMERIC (6,2) CHECK (VALOR >0), LETRA CHAR(1) CHECK ( LETRA IN (‘A’,’B’,’C’))

)

- Adicionando check a um campo já criado

Sintaxe

- ALTER TABLE tb\_check ADD CHECK (valor>0)

- Adicionando check atribuindo nome a constraint

Sintaxe

- ALTER TABLE tb\_check ADD CONSTRAINT CHEK\_VALOR(valor>0)

- Apagando uma constraint check

Sintaxe

- ALTER TABLE tb\_check DROP CONSTRAINT CHECK\_VALOR

**Desenvolvendo Consultas no SQL**

**Comando SELECT**

O comando SELECT possui uma estrutura e é executado em uma ordem lógica para retornar as informações.

Cláusulas

- SELECT – Seleciona a lista de resultados

- FROM – Aonde vou buscar meus dados

- WHERE – O que eu quero filtrar dos dados

- GROUP BY – Agrupa os dados

- HAVING – Filtra o valor do agrupamento

- ORDEN BY – Ordena o resultado

- Como escreve uma consulta simples

Sintaxe

- SELECT <\* / campos\_da\_tabela> FROM <tabela/view/functions\_tablevalue> WHERE <Filtros> ORDER BY <campos> ASC/DESC

- Uma consulta é executada logicamente na ordem como é mostrada abaixo.

1: FROM < table souce>

2: WHERE < search condition>

3: GROUP BY <group by list>

4: HAVING < search condition>

5: SELECT <select list>

6: ORDER BY <order by list>

- ORDER BY : Maneira em que os dados são ordenados após a realização de uma consulta a uma tabela. Quando não for informado como se quer ordenar, o SQL coloca como padrão o ASC.

- ACS – Ascendente – Crescente – (menor para o maior)

Sintaxe

- SELECT \* FROM TB\_FUNC ORDER BY NOME

Ex.: 2

- SELECT \* FROM TB\_FUNC ORDER BY NOME,SALARIO

- DESC – Descendente – Decrescente – (maior para o menor)

Sintaxe

- SELECT \* FROM TB\_FUNC ORDER BY NOME DESC

Ex.:2

- SELECT \* FROM TB\_FUNC ORDER BY NOME ASC, SALARIO DESC

**Operadores Lógicos**

São utilizados quando queremos trabalhar com o relacionamento de duas ou mais condições ao mesmo tempo.

Operadores lógicos

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Operador - Descrição**

AND - E lógico

OR - OU lógico

NOT - Negação

----------------------------------------

- **AND:** E lógico. Avalia as condições e devolve um valor verdadeiro caso ambos sejam corretos. Ambos os valores devem ser verdadeiro para que o operador AND traga o resultado.

Sintaxe

- SELECT MATRICULA, NOME FROM TB\_FUN WHERE DATAEMISSAO IS NULL NA STATU\_FUNC = ‘ATIVO’

- **OUR:** OU lógico. Avalia as condições e devolve um valor verdadeiro se algum for correto.

- Uma das condições pode ser verdadeira ou não.

Sintaxe

- SELECT MATRICULA, NOME FROM TB\_FUNC WHERE DATADEMISSAO IS NULL OR STATU\_FUNC = ‘ATIVO’

- **NOT:** Negação Lógica. Devolve o valor contrario de expressão, ou seja, a negação do filtro. O NOT inverte o resultado lógico de uma condição, ou seja se a condição for verdadeira o resultado é falso, e se a condição for falsa o resultado é verdadeiro.

- Negando a condição do filtro utilizado

Sintaxe

- SELECT MATRICULA, NOME, SALARIO FROM TB\_FUNC WHERE NOT STATUS\_FUNC = ‘ATIVO’

**Operadores Aritméticos**

São responsáveis pelas operações e pelos cálculos matemáticos simples obtidos com T-SQL.

Tabela de operadores aritméticos

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Operador - Descrição

Adição - +

Subtração - -

Multiplicação - \*

Divisão - /

Resto da divisão - %

--------------------------------------

- Utilizando operadores para realizar operações com os campos da tabela

Sintaxe

- SELECT NOME, SALARIO + 50 AS ADICAO, SALARIO – AS SUBTRAÇÃO, SALARIO \* 1.2 AS AJUSTE\_20\_PCENTO, SALARIO / 2 AS DIVISAO FROM TB\_FUNC WHERE STATUS\_FUNC = ‘ATIVO’

**Operadores Relacionais**

São responsáveis pelo estabelecimento de ações de comparação, são utilizadas para a definição de condições em que há necessidade de comparação entre dois ou mais valores.

Tabela de operadores Relacionais

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Operador - Descrição

> Maior que

< Menor que

= Igual a

>= Maior ou igual a

<= Menor ou igual a

<> ou != Diferente de

------------------------------------------------

Ex.:

Sintaxe

- SELECT \* FROM TB\_FUNC WHERE STATUS\_FUNC = ‘Ativo’

- SELECT \* FROM TB\_FUNC WHERE DATANASCIMENTO >= ‘1999-01-01’

- SELECT \* FROM TB\_FUNC WHERE SALARIO >= 2000 AND SALARIO <= 4000

- SELECT \* FROM TB\_FUNC WHERE STATUS\_FUNC <> ‘Ativo’

**Operadores Auxiliares**

São operadores que facilitam a definição de condições a serem utilizados com o comando DQL

Tabela de operadores auxiliares

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Operador - Descrição

IS NULL Verifica se o campo é Nulo

IS NOT NULL Verifica se o campo não está nulo

BETWEEN Verifica se o valor esta numa faixa de valores

IN Verifica se i valor existe em uma lista de valores

LIKE Verifica um valor buscando seus semelhantes

**IS NULL e IS NOT NULL**

IS NULL: Verifica se o campo em questão tem valores nulos.

ISD NOT NULL: verifica se o campo em questão não possui valores nulos.

Vazio diferente de nulo. O campo vazio tem informação enquanto o nulo não.

Um campo tem o valo nulo em um SGBD quando em uma determinada coluna é exibida o valor NULL.

Ex.:

Sintaxe

- SELECT \* FROM TB\_FUNC WHERE DATAEMISSAO IS NULL

- SELECT \* FROM TB\_FUNC WHERE DATAEMISSAO IS NOT NULL

**BETWEEN**

Utilizado para especificar um intervalo de valores.

Ex.:

Sintaxe

- SELECT \* FROM TB\_FUNC WHERE DATAADMISSAO BETWEEN ‘2012-01-01’ AND ‘2018-12-20’

- SELECT \* FROM WHERE DATAADMISSAO NOT BETWEEN ‘2017-01-01’ AND ‘2017-12-20’

- SELECT \* FROM TB\_FUNC WHERE CODDEPTO BETWEEN 5 NA 15

**IN**

Utilizado para verificar se o valor procurado está dentro de uma lista.

Ex.:

Sintaxe

- SELECT \* FROM TB\_FUNC WHERE STATUS\_FUNC IN(‘ATIVO’,’FÉRIAS’)

- SELECT \* FROM TB\_FUNC WHERE MONTH(DATAADMISSAO) IN (1,2)

- SELECT \* FROM TB\_FUNC WHERE MONTH(DATAADMISSAO) NOT IN(10,11)

**LIKE**

Utilizado para fazer busca em campos com valores texto.

Podemos utilizar a porcentagem para ajudar na busca. É considerado um caractere coringa no LIKE.

Palavra + % - trás todos registros que começam com o nome da palavra passada

% + - trás todos os registros que terminam com a palavra passada.

% + palavra +% - trás todos os registros que contenham a palavra no inicio, meio ou final.

EX.:

Sintaxe

- SELECT \* FROM TB\_FUNC WHERE NOME LIKE’JOSE’

- SELECT \* FROM TB\_FUNC WHERE NOME LIKE ‘JOSE%’

- SELECT \* FROM TB\_FUNC WHERE NOME ‘%JOSE’

- SELECT \* FROM TB\_FUNC WHERE NOME LIKE ‘%JOSE%’

**Extra – Clausulas que podem ser utilizadas em um comando SELECT**

Clausulas Extra

- ALIAS (Apelidos para tabelas e campos)

- CASE

- DISTINCT

- TOP

ALIAS = São apelidos utilizados para renomear tabelas e colunas a nível de consulta, ou seja, somente no momento do SELECT. Os apelidos ficam ativos durante a consulta assumindo o valor do campo ou tabela.

Caso for utilizar ALIAS de tabela, o mesmo deve ser utilizado durante toda a construção da consulta.

\* Facilitam a leitura da Query quando utilizamos JOINS.

Em casos de ALIAS de coluna não se torna obrigatório, pois são utilizados para exibir um nome amigável para a coluna em questão.

É possível realizar ALIAS de coluna de três maneiras diferentes

Sintaxe :

- Coluna AS Apelido

NOME AS ‘NOME FUNCIONARIO’

- Apelido = Coluna

‘NOME AS ‘NOME FUNCIONARIO’

- Coluna Apelido

NOME ‘NOME FUNCIONARIO’

Utilizado alias de coluna e tabela numa consulta.

Sintaxe:

SELECT F.NOME AS ‘NOME FUNCIONARIO’,’DATA DE ADMISSÃO’ = F.DATAADMISSAO,F.SALARIO ‘SALÁRIO FUNCIONÁRIO FROM TB\_FUNC AS F