**Curso SQL**

**Structured Query Language – SQL** **/ \*Aula 1\***

\*Linguagem de Consulta estruturada padrão para acesso a Banco de Dados;

**Funções Principais;**

\*Permite o acesso a dados em SGBDR

\*Permite definir os dados no baco de dados e manipula-los

\*Permite criar e excluir bancos de Dados e tabelas

\*Permite configurar permissões de acesso em tabelas, procedimentos e visões;

**Grupos de Comandos;**

**DDL – Data Definition Language**

Comando – Descrição

**CREATE** – Cria uma nova tabela visão ou ou outro objeto no BD

**ALTER** – Modifica um objeto existente no BD, como uma tabela

**DROP-** Exclui uma tabela inteira, uma exibição de uma tabela ou outro objeto no banco de dados

**Sintaxe = Para Excluir**

ALTER TABLE ( nome da tabela)

DROP COLUMN ( nome da coluna)

Pode-se excluir uma constraint:

ALTER TABLE (nome da tabela)

DROP COSNTRAINT (nome\_constraint)

**Sintaxe = Para Adicionar (exemplo)**

ALTER TABLE (nome da tabela)

ADD ID\_Autor SMALLINT NOT NULL

CONSTRAINT fk\_ID\_Autor FOREIGN KEY (ID\_Autor)

REFERENCES tbl\_autores

ALTER TABLE tbl\_Livro

ADD ID\_editora SMALLINT NOT NULL

CONSTRAINT fk\_id\_editora FOREIGN KEY (ID\_editora)

REFERENCES tbl\_editoras

**Sintaxe = Para Alterar Coluna – ALTER COLUMN**

ALTER TABLE (nome da tabela)

ALTER COLUMN ID\_Autor SMALLINT

**Sintaxe = Para Adicionar PK – ADD PK**

ALTER TABLE (nome da tabela)

ADD PRIMARY KEY ( ID\_Cliente)

**Sintaxe = Para Excluir Tabela – DROP TABLE**

DROP TABLE ( nome da Tabela)

**DML – Data Manipulation Language ( Manipulação de Dados )**

Comando – Descrição

INSERT – Cria um registro (linha)

UPDATE – Modifica registros

DELETE – Exclui registros

**DCL – Data Control Language ( Controle de Dados )**

GRANT – Dá privilégios(permissões) a um usuário

REVOKE – Retira privilégios(permissões) fornecidos a um usuário

**DQL -Data Query Language ( Consulta de Dados )**

SELECT – Obtém Registros especificados de uma ou mais tabelas

**SGBDR / \*Aula 2\***

\*Sistema Gerenciador de Banco de Dados Relacional -SGBDR

\*Trata-se de um sistema de gerenciamento de banco de dados baseado no modelo relacional introduzido por E.F.Cood.

**Composição de um Banco de Dados**

. Tabelas / . Campos (colunas) / . Registros (linhas)

. Tabelas=Onde são armazenados os dados , é uma coleção de entradas de dados relacionados e consiste em linha e colunas;

. Campos = Onde representam os atributos , como nome, Data Nascimento, Salário, Preço, etc. Campo é uma coluna em uma tabela que mantém informações específicas sobre cada registro.

. Registros = Linhas ou Tuplas. Cada entra individual de uma tabela, Dados de uma entidade única;

**Tipos de Dados**

Tipo / Descrição / Armazenamento

char(n) – String de caracteres de tamanho fixo, máximo de 8000 caracteres / n

varchar(n) – String de caracteres de tamanho variável, máximo de 8000 caracteres; (1 Byte por Caractere)

nchar(n) – Dados Unicode de tamanho fixo, máximo de 4000 caracteres;

nvachar(n) – Dados Unicode de tamanho variável, máximo de 4000 caracteres;

bit – 0, 1 ou nulo;

tinyint – Números inteiros de 0 a 255 / 1 byte

smallint – Números inteiros de -32768 a 32767 / 2 bytes

int – Números inteiros entre -2,147,483,648 e 2,147,483,647 / 4 bytes

bigint – Número entre -2,223,372,036,854,775,808 e 9,223,372,036,854,775,807 / 8 bytes

real – Números de flutuante entre -3.4 x 1038 e 3.4x1038 / 4 bytes

datetime – De 01/01/1753 a 31/12/9999, com uma precisão de 3.33 milisegundos / 8 bytes

smalldatetime – De 01/01/1900 a 06/06/2079, com uma precisão de 1 minuto / 4 bytes

date – Data apenas. De 01/01/2001 a 31/12/9999 / 3 bytes

time – Hora apenas. Precisão de até 100 nanossegundos 3-5 bytes

text – Cadeia de caracteres de tamanho variável. Até 2GB de dados.

money – Dados monetários de -922,337,203,685,477.5808 até 922,337,203,685,477.5807 / 8 bytes

**Criação de um Banco de Dados**

**Criar nos comandos :**

CREATE DATABASE db\_Bibilioteca ( Nome do Banco)

ON PRIMARY (arquivos primários) (

NAME= db,Biblioteca,

FILENAME (nome do arquivo) = ´C:\SQL\db\_Biblioteca.MDF´,

SIZE (tamanho) = 6MB,

MAXSIZE (tamanho máximo) = 15MB,

FILEGROWT ( crescimento do arquivo) = 10%

)

**Comandos**

**USE =** O comando USE instrui o SGBDR a utilizar o banco de dados especificado para rodar os comandos.

Sintaxe: USE banco\_de\_dados

**sp\_heldb =** informa o tamanho, taxa de crescimento, e local do banco de dados.

Sintaxe: sp\_helpdb BANCO\_DADOS

**sp\_help (Nome Tabela)** = Informa os índices, chaves e campos, atributos tipos de atributos de uma tabela ( stored procedures, comandos pré-programados)

**--** em sequência = Significa comentar uma linha

**AUTO INCREMENT - IDENTITY=** Significa Auto incremento Informa automaticamente um código e não precisa digitar.

\*O auto incremento permite que um número único seja gerado quando um novo registro é inserido em uma tabela.

\*Em SQL SEVER trata-se da palavra chave IDENTITY (identidade), cujo valor inicial padrão é 1, e se incrementa em 1.

\*Para que o valor de IDENTITY inicie em 100 e se incremente de 2 em 1, use IDENTITY(100,2)

\*Ao inserir valores na tabela, não é necessário especificar o valor para a coluna de auto-incremento.

\*Não é possível alterar uma coluna existente para configurar IDENTITY.

\*Se necessário, crie uma nova tabela com IDENTITY e exclua a atual.

\*Só é permitido usar uma coluna de identidade por tabela.

**Inserir dados em uma tabela**

Sintaxe:

**INSERT INTO** nome tabela (coluna1,coluna2,...)

**VALUES** (valor1, valor2,...)

**TRUCATE TABLE – Apagar Dados da Tabela.**

\*Remove todas as linhas de uma tabela sem registrar as exclusões de linhas individuais.

\*TRUNCATE TABLE é como a instrução **DELETE** sem usar cláusula **WHERE**.

\*Entretanto, TRUCANTE TABLE é mais rápida e utiliza menos recursos de sistema e log de transações.

**Sintaxe=**

Consultar Antes = SELECT COUNT (\*) AS (nome tabela)\_Antes FROM (nome da Tabela)

TRUNCATE TABLE (nome da tabela)

Consultar Depois = SELECT COUNT (\*) AS (nome tabela)\_Depois FROM (nome da Tabela)

**SELECT - Consultar Dados (Consulta Simples)**

Sintaxe:

SELECT (nome coluna) FROM (nome Tabela)

Exemplo:

SELECT Nome\_Autor FROM tbl\_Autores ( Dados da Coluna e Tabela)

SELECT \* FROM tbl\_autores ( Tabela completa)

SELECT Nome\_Livro, ID\_Autor FROM tbl\_Livro ( Consultar mas de uma coluna)

SELECT Nome\_livro, ID\_Autor FROM tbl\_Livro ORDER BY Nome\_Livro ( Neste caso a coluna Livro Virá em ascendente ou descendente usando o comando ORDER BY )

**ORDER BY**

\*A palavra-chave ORDER BY é usada para ordenar o conjunto-resultado de registros.

\*ASC – Ordem Ascendente

\*DESC – Ordem Descendente ( inversa)

Sintaxe:

SELECT \* FROM (nome da Tabela) ORDER BY ( nome da Coluna) (ASC/DESC)

Exemplo: SELECT \* FROM tbl\_Livro ORDER BY Nome\_livro ASC

**SELECT DISTINCT**

\*Algumas Colunas podem conter valores duplicados. Para exibir valores diferentes (“distintos”), use a palavra-chave DISTINCT:

\*Esse comando mostra valores únicos

Sintaxe:

SELECT DISTINCT (nome colunas) FROM (nome da tabela)

Exemplo: SELECT DISTINCT ID\_Autor FROM tbl\_Livro

**WHERE**

\*Permite filtrar registros em uma consulta.

Sintaxe:

SELECT (nome da coluna) FROM (nome da tabela) WHERE coluna = valor

Exemplo:

SELECT \* FROM tbl\_Livro WHERE ID\_Autor = ‘1’

SELECT ID\_Autor FROM tbl\_Autores WHERE Sobrenome\_Autor = ‘Stanek’

**OPERADORES AND e OR**

\*Usados para filtrar registros baseados em mais de uma condição.

\*O operador AND mostra um registro se ambas condições forem verdadeiras.

\*O operador OR mostra um registro se pelo menos uma das condições for verdadeira.

Sintaxe:

SELECT \* FROM (nome da tabela) WHERE (nome da coluna) > (número) AND/OR (nome da coluna) < (número)

SELECT \* FROM tbl\_Livro WHERE ID\_Livro > 101 AND ID\_Autor < 3

SELECT \* FROM tbl\_Livro WHERE ID\_Livro > 102 OR ID\_Autor < 4

**UPDATE= Trocar de Dados de Uma Tabela**

OBS: Usar WHERE se não vai mudar todos os dados de uma coluna

Sintaxe:

UPDATE (nome da tabela) SET (nome da Coluna) = (Valor)

WHERE

Exemplo:

UPDATE tbl\_autores SET Nome\_Autor = ‘Matheus’ WHERE ID\_Autor = 4;

Exemplo 2 : Trocando dados em mas de uma coluna de uma vez;

UPDATE tbl\_Livro SET Preco\_Livro = 65.45,

Data\_Pub = 2000-11-30 WHERE ID\_Livro = 102

**SELECT TOP**

\*Usado para especificar o número de registros a retornar.

\*Útil para tabelas com muitos registros.

Sintaxe:

SELECT TOP número/percentual (nome coluna) FROM (nome tabela)

Exemplo: (Seleciona o percentual)

SELECT TOP 10 PERCENT Nome\_Livro FROM tbl\_Livro

Exemplo 2 : ( Seleciona os números)

SELECT TOP 3 Nome\_Livro FROM tbl\_Livro

**ALIAS COM “AS” – SQL**

\*Pode-se dar um nome diferente a uma coluna ou tabela em uma consulta.

Sintaxe:

SELECT (nome da coluna) AS (nome\_alias escolhido) FROM (nome da tabela)

Exemplo:

SELECT Nome\_Livro AS Livro FROM tbl\_Livro – Mudar em uma coluna só

SELECT Nome\_Livro AS Livro, ID\_Autor AS Autor FROM tbl\_Livro – Mudar mas de uma coluna de uma vez só

**Observação**: A virgula é de suma importância em alguns comandos, para se poder realizar mas de um comando por vez.

**OPERADOR UNION**

\*Permite combinar duas ou mais declarações SELECT.

\*Cada declaração SELECT deve ter o mesmo número de colunas, tipos de dados e ordem das colunas.

Sintaxe:

SELECT (nome da coluna) FROM (nome da 1 tabela) UNION SELECT (mesmo tipo de dados da outra nome da coluna) FROM (nome da 2 tabela)

SELECT ID\_Autor FROM tbl\_autores UNION SELECT ID\_Autor FROM tbl\_Livro

**SELECT INTO**

\*Seleciona dados de uma ou mais tabelas e os insere em uma tabela diferente.

\*Pode ser usada para criar cópias de backup de tabelas.

Sintaxe:

SELECT (nome qualquer coluna) INTO (nome da nova tabela) FROM ( tabela atual).

Exemplo:

SELECT Nome\_Livro,ISBN INTO Livro\_ISBN FROM tbl\_Livro WHERE ID\_Livro > 2

Exemplo: Criar tabela de Backup com todas colunas e campos

SELECT \* INTO ( nome da tabela nova) FROM ( nome da tabela atual)

SELECT \* INTO tbl\_Livro\_backup FROM tbl\_Livro

**BETWEEN – Seleção de Intervalos**

Sintaxe:

SELECT (nome da coluna) FROM (nome da tabela) WHERE (nome da coluna BETWEEN (valor1) AND (valor2)

Exemplo:

SELECT \* FROM tbl\_Livro WHERE Data\_Pub BETWEEN ‘20040517’ AND ‘20100517’

**OPERADORES LIKE E NOT LIKE**

\*Determina se uma cadeia de caracteres específica corresponde a um padrão especificado. Um padrão pode incluir caracteres normais e curingas.

\*NOT LIKE inverte a comparação, verificando se a cadeia de caracteres NÃO corresponde ao padrão especificado.

\*Usado junto com WHERE:

Sintaxe:

SELECT (nome colunar) FROM (nome tabela) WHERE (nome Coluna) LIKE padrão (nomes ou letras repetidos)

Observação : A ordem da consulta é determinado no lugar onde o padrão irá ficar;

LIKE – Padrões específicos

‘%’ – Qualquer cadeia de 0 ou mais caracteres

‘\_’ – Sublinhado: qualquer caracter único

‘[]’ – Qualquer caracter único no intervalo ou conjunto especificado ([a-hh]); [aeiou]), mais de uma consulta;

‘[^]’ – Qualquer caracter único que não esteja no intervalo ou conjunto especificado ([a-h]; [aeiou])

**JOINS ( SELECIONAR DADOS DE DUAS OU MAIS TABELAS)**

\*A palavra chave JOIN é usada para obter dados provenientes de duas ou mais tabelas, baseado em um relacionamento entre colunas nestas tabelas.

JOINS – Tipos

**- INNER JOIN**: Retorna linhas quando houver pelo menos uma correspondência em ambas as tabelas.

**- OUTER JOIN**: Retorna linhas mesmo quando não houver pelo menos uma correspondência em uma das tabelas (ou ambas). OUTER JOIN divide-se em LEFT JOIN, RIGHT JOIN e FULL JOIN.

**INNER JOIN**

Sintaxe:

SELECT (nome colunas) FROM (nome tabela1) INNER JOIN (nome tabela2) ON (nome tabela1).(nome coluna)= (nometabela2).(nomecoluna)

Exemplo:

SELECT \* FROM tbl\_Livro INNER JOIN tbl\_autores ON tbl\_livro.ID\_Autor = tbl\_autores.ID\_Autor

**Exemplo 2 :**

SELECT tbl\_Livro.Nome\_Livro, tbl\_Livro.ISBN,tbl\_autores.Nome\_Autor

FROM tbl\_Livro INNER JOIN tbl\_autores ON tbl\_Livro.ID\_Autor = tbl\_autores.ID\_Autor.

**Exemplo 3 : (Usando Alisases)**

SLECT L.Nome\_Livro, E.Nome\_editorra FROM tbl\_livro AS L INNER JOIN tbl\_editoras AS E ON L.ID\_editora = E.ID\_editora.

**OUTER JOINS**

- LEFT JOIN: Retorna todas as linhas da tabela à esquerda, mesmo se não houver nenhuma correspondência na tabela à direita.

Sintaxe:

Ex1.:

SELECT (coluna) FROM (tabela\_esquerda) LEFT (OUTER) JOIN (tabela\_direita ) ON tabela\_esquerda.coluna = tabela\_direita.coluna

Ex2.: - “Excluindo correspondências”

Sintaxe:

SELECT (coluna) FROM (tabela\_esquerda) LEFT (OUTER) JOIN (tabela\_direita ) ON tabela\_esquerda.coluna = tabela\_direita.coluna

WHERE tabela\_direita.coluna IS NULL

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

- RIGHT JOIN: Retorna todas as linhas da tabela à direita, mesmo se não houver nenhuma correspondência na tabela à esquerda.

- FULL JOIN: Retorna linhas quando houver uma correspondência em qualquer um das tabelas. É uma combinação de LEFT e RIGHT JOINS.

**FULL JOIN**

Combinação de RIGHT JOIN com LEFT JOIN, retornando registros que não possuam correspondência em ambas as tabelas.

Sintaxe:

SELECT colunas

FROM tabela1

FULL(OUTER) JOIN tabela2

ON Tabela1.colunas = Tabela2.coluna

**IN: Filtro de Múltiplas Condições**

Determina se um determinado valor corresponde a qualquer valor em uma subconsulta ou lista.

Retorna sempre True ou False

Pode substituir OR em queries com mas de uma condição.

Sintaxe:

Expressão\_teste (NOT) IN (subconsulta/Lista)

SELECT \* FROM (nomedatabela)

WHERE (campo) IN (resultado que você quer obter)

**Campos Calculados**

Cria campos que são automaticamente calculados quando se colocar valores em outros campos referentes a o campo que você deseja obter resultado.

Sintaxe:

CREATE TABLE (nomedatabela) ( campos com seus dados + campo calculável AS ( campo1 (operador) campo2) ) ;

**Funções Agregadas**

\*Usa-se essa funções para uma consulta no BD, então sempre usar o comando SELECT;

\*Uso em Cálculos;

MIN = Valor Mínimo

MAX = Valor Máximo

AVG = Média Aritmética

SUM = Total (Soma)

COUNT = Contar quantidade de itens

Sintaxe:

SELECT (nome da função) (nome da coluna em () ) FROM (nome da tabela)

**SQL Constraints ( Restrições)**

\*Restrições são regras aplicadas nas colunas de uma tabela.

\*São usadas para limitar os tipos de dados que são inseridos.

\*Podem ser especificados no momento de criação da tabela (CREATE) ou após a tabela ter sido criada (ALTER)

Principais:

**NOT NULL =**

\*A constraint NOT NULL impõe a uma coluna a NÃO aceitar valores NULL.

\* A constraint NOT NULL obriga um campo a sempre possuir um valor.

\* Deste modo, não é possível inserir um registro ( ou atualizar) sem entrar com um valor este campo.

**UNIQUE =**

\*A restrição UNIQUE identifica de forma única cada registro em uma tabela de um banco de dados.

\*As constraints UNIQUE e PRIMARY KEY garantem a unicidade em uma única coluna ou conjunto de colunas.

\*Uma constraint PRIMARY KEY automaticamente possui uma restrição UNIQUE definida.

\*Você pode ter várias constraints UNIQUE em uma tabela, mas apenas uma Chave Primária por tabela.

**PRIMARY KEY=**

\*A PRIMARY KEY identifica de forma única cada registro em uma tabela de banco de dados.

\*Chaves Primárias devem conter valores únicos.

\*Uma coluna de chave primária não pode conter valores NULL

\* Cada tabela deve ter uma chave primária e apenas uma chave primária

**FOREIGN KEY=**

\*Uma FOREIGN KEY (Chave Estrangeira) em uma tabela é um campo que aponta para uma chave primária em outra tabela.

CONSTRAINT fk\_ID\_Autor FOREIGN KEY (ID\_Autor)

REFERENCES tbl\_autores (ID\_Autor)

Neste exemplo a chave primária está na tabela tbl\_autores e uma chave estrangeira de nome ID\_Autor foi criada na tabela atual, usando o nome fk\_ID\_Autor

**CHECK=**

\*A constraint CHECK é usada para limitar uma faixa de valores que podem ser colocados em uma coluna.

\*Se uma constraint CHECK for definida em uma única coluna ela permitirá apenas determinados valores para coluna.

\*Se a constraint CHECK for definida para a tabela, ela poderá limitar os valores em algumas colunas com base nos valores de outras colunas do registro.

**DEFAULT=**

\*A restrição DEFAULT é usada para inserir um valor padrão em uma coluna.

\*O valor padrão será adicionado a todos os novos registros caso nenhum outro valor seja especificado.

**JOINS**

**OUTER JOINS**

- LEFT JOIN: Retorna todas as linhas da tabela à esquerda, mesmo se não houver nenhuma correspondência na tabela à direita.

- RIGHT JOIN: Retorna todas as linhas da tabela à direita, mesmo se não houver nenhuma correspondência na tabela à esquerda.

- FULL JOIN: Retorna linhas quando houver uma correspondência em qualquer um das tabelas. É uma combinação de LEFT e RIGHT JOINS.

**Stored Procedures**

Definição = São lotes (batches) de declarações SQL que podem ser executadas como uma sub-rotina.

Permitem centralizar o lógica de acesso aos dados em um único local, facilitando a manutenção e otimização de código.

Também é possível ajustar permissões de acesso aos usuários, definindo quem pode ou não executá-las.

**Criar um procedimento Armazenado**

Sintaxe:

CREATE PROCEDURE nome\_procedimento

(@Parâmetro Tipo\_dados)

AS

Bloco de códigos

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

Exemplo 1

CREATE PROCEDURE teste

AS

SELECT ‘Bóson Treinamentos’ AS Nome

Para executar:

EXEC(UTE) teste

Obs.: Se o procedimento armazenado for o primeiro comando de um batch, não é necessário usar a palavra EXEC.

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

Exemplo 2

CREATE PROCEDURE p\_LivroValor

AS

SELECT Nome\_Livro, Preco\_livro FROM tbl\_Livro

**Visualizar conteúdo de SP**

Use o procedimento armazenado sp\_helptext para extrair o conteúdo de texto de uma stored procedure:

EXEC sp\_helptext nome\_procedimento

Ex.:

EXEC sp\_helptext p\_LivroValor

**Criptografar Stored Procedure**

CREATE PROCEDURE p\_LivroISBN

WITH ENCRYPTION

AS

SELECT Nome\_Livro, ISBN

FROM tb\_Livro

Agora tente visualizar seu contéudo com sp\_helptext:

EXEC sp\_helptext p\_LivroISBN

**Modificar Stored Procedure**

ALTER PROCEDURE

ALTER PROCEDURE <nomeprocedimento> bloco de código da sp

**Parâmetros de Entrada**

ALTER PROCEDURE teste (@par1 AS int)

AS

SELECT @part

Executar passando um parâmetro:

EXEC teste 22 –22 é o valor do parâmetro passado

OBSERVAÇÕES EM RELAÇÃO AO BANCO DE DADOS

\*Sempre que for criar uma tabela , o interessante é começar pela que não tem FK, pois se não dará erro no DB.

\*Sempre em Datas e nomes usar ‘aspas simples’, e valores numéricos não usar.

\*O sinal \*, ele inserido seleciona todas as colunas de uma tabela.