SQL – Structured Query Language (Linguagem de Consulta Estruturada)

SGDBR – Sistemas de Gerenciamento de Banco de Dados Relacionais

Grupo de Comandos:

DDL – Data Definition Language

* CREATE - Cria tabelas, visão ou outro objeto no BD.
* ALTER - Modifica um objeto existente no BD, como uma tebela.
* DROP - Exlcui uma tabela inteira, uma exibição de uma tabela ou outro objeto no BD.

DML – Data Manipulation Language

* INSERT – Cria um registro (linhas/tuplas)
* UPDATE – Modifica registros
* DELETE - Exclui registros

DCL – Data Control Language

* GRANT – Dá privilégios a um usuário
* REVOKE – Retira privilégios fornecidos

DQL – Data Query Language

* SELECT Obtém registros especificados de uma ou mais tabelas. (consulta)

SQL Constraints (Restricoes)

* Restricoes são regras aplicadas nas colunas de uma tabela.
* São usadas para limitar os tipos de dados que são inseridos.
* Podem ser especificadas no momento de criacao da tabela (CREATE) ou após a tabela ter sido criada (ALTER).

Principai Constraits:

1. NOT NULL

* Impoe a uma coluna a NÃO aceitar valores NULL.
* Obriga um campo a sempre possuir um valor.
* Não e possivel inserir um registro (ou atualizar) sem entrar com um valor nete campo.

1. UNIQUE

* Identifica de forma única cada registro em uma tabela de um banco de dados.
* UNIQUE E PRIMARY KEY garantem a unicidade em uma coluna ou conjunto de colunas.
* Uma constrint PRIMARY KEY automaticamente possui uma restricao UNIQUE definida.
* Voce pode ter varis constraits UNIQUE em uma tabela, mas apenas um Chave primaria por tabela.

1. PRIMARY KEY

* Identifica de forma única cada registro em uma tabela de BD.
* Devem conter valores unicos
* Não pode ter valores NULL
* Cada tabela deve ter uma chave primaria.

1. FOREING KEY

* Uma FOREING 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.

1. 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.

Constraint AUTO\_INCREMENT

* O auto incremento permite que um número seja gerado quando um novo registro é inserido em um a tabela.
* Em MySQL trata-se da palavra chave AUTO\_INCREMENT, cujo valor incial padrão é 1, e se incrementa em 1.
* Para que o valor da coluna se inicie em 100, por exemplo, use:

AUTO\_INCREMENT = 100

* Ao inserir valores na tabela, não é necessário especificar o valor para a coluna de auto-incremento.
* Só é permitido usar uma coluna de auto incremento por tabela, geralmente do tipo inteiro.
* Necessita também de constraint NOT NULL (configurado automaticamente).

Verificar o valor atual do auto incremento

Podemos verificar o valor de incremento mais atual armazenado em um tabela no banco de dados com o comando a seguir:

SELECT MAX (\*nome\_coluna\*)

FROM \*nome\_tabela\*;

Alterar o próximo valor no campo de Auto-Incremento

Para alterar o valor de incremento do próximo registro a ser armazenado em uma tabela, use o comando a seguir:

ALTER TABLE \*tabela\* AUTO\_INCREMENT = \*valor\*;

Tipos de Dados

* INT – Inteiros entre -2,147,483,648 e 2,147,483,647.
* TINYINT – Inteiros entre -128 a 127.
* SMALLINT – Inteiros de -32768 a 32767.
* MEDUIMINT – Inteiros de -8388608 a 8388607.
* BIGINT – Inteiros entre -9,223,372,036,854,775,808 e 9,223,372,036,854,775,807.
* DECIMAL(M,D) – Ponto decimal com M dígitos no total (precisão) e D casas decimais (Escala); o padrão é 10,0; M vai até 65 e D até 30.
* FLOAT(M,D) – Ponto flutuante com precisão M e escala D; o padrã é 10,2; D vai até 24.
* CHAR(M) – String que ocupa tamanho fixo entre 0 e 255 caracteres.
* BOLL / BOOLEAN – Valores binários (0 / 1); Na verdade, é um ‘alias’ para o tipo TINYINT(1).
* VARCHAR(M) – String de tamanho variável, até 65535 caracteres.
* BLOB / MEDIUMBLOB / TINYBLOB – Campo com tamanho máximo de 65535 caractéres binários; ‘Binary Large Objects’, são usados para armazenar grandes quantidades de dados, como imagens.
* MEDIUMTEXT – Permite armazenar até 16.777.215 caracteres.
* LOGTEXT – Permite armazenar até 4.294.967.295 caracteres.
* DATE – Uma data de 01/01/1000 a 31/12/9999, no formato YYYY-MM-DD
* DATETIME – Uma combinação de data e hora de 01/01/100 00:00:00 a 31/12/9999 23:59:59, no formato YYYY-MM-DD HH:MM:SS.
* TIME – Hora apenas, no formato HH:MM:SS
* YEAR (M) – Ano nos formatos de 2 ou 4 digitos; Se forem 2 (YEAR(2)), ano vai de 1970 a 2069; para 4 (YEAR(4)), vai de 1901 a 2155. O padrão é 4.

Alterar Tabelas

* É possível alterar a estrutura de uma tabela após ter sido criada, acrescentando ou excluindo atributos (campos).
* Usamos para isso o comando ALTER TABLE.

ALTER TABLE – DROP

ALTER TABLE \*tabela\*

DROP COLUMN \*coluna\*;

Inserir dados

Inserir dados em uma tabela:

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

VALUES (valor1, valor2, ...)

Consultas Simples

SELECT \*coluna\* FROM \*tabela\*;

Consultas com Ordenação

* A palavra-chave ORDER BY é usada para ordernar o conjunto-resultado de registros.
* ASC – Ordem ascendente
* DESC – Ordem descendente (inversa)
* Exemplo:

SELECT \* FROM tbl\_Livro

ORDER BY Nome\_Livro ASC;

SELECT Nome\_Livro, Preco\_Livro FROM tbl\_livro ORDER BY Preco\_Livro DESC;

Índices (Muito útil para otimização para consultas de colunas que são bastante acessadas).

* São empregados em consultas para ajudar a encontrar registros com um valor específico em uma coluna de forma rápida – ou seja, aumentar o desempenho na execução de consultas.
* Com índices, o MySQL vai direto a uma linha em vez de buscar toda a tabela até encontrar os registros que importam.

Por padrão, o MySQL cria índices automaticamente para campos de:

* + Chave Primária
  + Chave Estrangeira
  + Constraint UNIQUE

Além disso, podemos criar índices para outras colunas usadas com frequência em buscas ou junções.

Índices Clusterizados

* Alteram a forma como os dados são armazenados em um banco de dados, pois ele classica as linhas de acordo com a coluna que possui o índice.
* Uma tabela só pode ter um índice clusterizado. Geralmente está na coluna que é chave primária da tabela ou, em sua ausência, em uma coluna UNIQUE.
* Se uma tabela não possir índice clusterizado, suas linhas são armazenadas em uma estrutura não-ordenada chamada de heap.

Índices Não-Clusterizados

* Em um índice não clusterizado a forma como os dados são armazenados não é alterada, e um objeto separado é criado na tabela, apontando para as linhas da tabela original após a busca.
* Baseia-se em valores chave.
* Uma tabela pode ter vários índices não-clusterizados.

Criar índices

Declaração CREATE INDEX em uma tabela já existente:

Sintaxe:

CREATE [UNIQUE] INDEX nome\_índice

ON nome\_tabela (

Coluna1 [ASC | DESC],

[coluna2 [ASC | DESC]] …

);

Declaracao ALTER TABLE / ADD INDEX em uma tabela já existente:

Sintaxe:

ALTER TABLE nome\_tabela ADD INDEX nome\_índice (colunas)

Cláusula WHERE

Permite filtrar registros em uma consulta.

Sintaxe:

SELECT colunas FROM tabela WHERE coluna = valor;

Ex:

SELECT Nome\_Livro, Data\_Pub FROM tbl\_Livro

Where ID\_Autor = 1;

SELECT ID\_Autor, Nome\_Autor FROM tbl\_Autores

WHERE Sobrenome\_Autor = ‘Stanek’;

Operadores AND, OR e NOT

* Usados para filtrar registros baseados em mais de uma condição.
* O operador AND mostra um registro se ambas as condições forem verdadeiras.
* O operador OR mostra um registro se pelo menos umas das condições for verdadeira.
* O operador NOT é a negação de uma expressão.

Exemplos:

SELECT \* FROM tbl\_Livro

WHERE ID\_Livro > 2 AND ID\_Autor < 3;

SELECT \* FROM tbl\_Livro

WHERE ID\_Livro > 2 OR ID\_Autor < 3;

SELECT \* FROM tbl\_Livro

WHERE ID\_Livro > 2 AND NOT ID\_Autor < 3;

Excluir Registros

Sintaxe:

DELETE FROM tabela WHERE coluna = valor;

Ex:

DELETE FROM tbl\_autores WHERE ID\_Autor = 2;

Obs.: Sempre use a cláusula WHERE para evitar a perda de dados da tabela!

Truncate Table

* Remove todas as linhas de uma tabela sem registrar as exclusões de linhas individuais.
* TRUNCATE TABLE é como a instrução DELETE sem usar a cçáusula WHERE.
* Entretanto, TRUNCATE TABLE é mais rápida e utiliza menos recursos de sistema e log de transações.

Ex: Excluir todos os registros da tabela tbl\_teste\_incremeto:

TRUNCATE TABLE tbl\_teste\_incremento;

SQL Alias

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

SELECT colunas

AS alias\_coluna

FROM tabela AS alias\_tabela;

SELECT Nome\_Livro

As Livro

FROM tbl\_Livro;