Comandos Fundamentais para Bancos de Dados

CREATE DATABASE

Esse comando serve para criarmos um banco de dados. A sintaxe básica é:

> *CREATE DATABASE* **<database_name>** *–opcionais (geralmente iniciado com WITH)*;

> Existem parâmetros adicionais como:

1. **ENCODING**: Está relacionado com a definição do conjunto de caracteres que serão usados no banco de dados. O padrão geralmente é UTF-8.
2. **OWNER**: Usuário do SGBD que está criando o banco e é o usuário administrador principal.

CREATE TABLE

Esse comando serve para criarmos uma tabela no banco criado. A sintaxe básica é:

> *CREATE TABLE* **<table_name>** *(column_name <data_type> -- optional_params)*

> Ao criarmos uma tabela é necessário declarar suas colunas onde serão armazenados os dados. Em cada coluna é necessário declarar o tipo do dado que irá ser armazenado ali além de alguns outros parâmetros opcionais. Vamos ver agora os tipos de dados principais que podem ser utilizados nas colunas nos SGBDs MySQL e PostgreSQL.

Tipos de Dados

- **Dados de caractere**

– **VARCHAR**: O tipo VARCHAR recebe uma string (texto) com até 255 posições. Para inferir o tamanho máximo que se espera dentro de uma coluna varchar é necessário usar a sintaxe *VARCHAR(<tamanho_da_string>)*

– **CHAR**: O tipo CHAR funciona parecido com o VARCHAR, mas ao declarar o tamanho através da sintaxe *CHAR(<tamanho_da_string>)* o banco de dados espera receber uma string com o tamanho exato entre parênteses. Se coloco 10 como tamanho default no CHAR, ao inserir uma string de tamanho < 10 ou tamanho >10 um erro ocorrerá.

– **TEXT**: Utilizado para textos, strings de grande posições. Este tipo armazena de 1 a 4GB na memória.

– **BLOB**: Utiliza-se o tipo de dados BLOB para armazenar dados binários, como imagens, vídeos e arquivos de documentos. Nesse sentido, armazena valores de grande porte e pode armazenar até 65.535 bytes de dados.

- **Dados de texto**

– **TINYTEXT**: Textos pequenos com tamanho máximo de 255 caracteres (posições)

– **TEXT**: Para textos de até 65535 caracteres (posições)

– **MEDIUMTEXT**: Para textos de até 16777215 caracteres (posições)

– **LONGTEXT**: Para textos de até 4294967295 caracteres (posições)

- **Dados numéricos**

– **INT**: Usado para declarar valores do tipo inteiro (1, 2, 3, 4, 5, …, 650, …, …n).

– **BIGINT**: Semelhante ao tipo INT, mas pode armazenar valores inteiros maiores, adequado para armazenar valores inteiros grandes e nesse sentido, utilizado em tabelas com muitas linhas e colunas.

– **DECIMAL**: Usado para armazenar valores numéricos com uma precisão especificada, adequado para armazenar valores monetários e financeiros.

– **FLOAT**: O tipo de dados FLOAT é usado para armazenar valores numéricos com uma precisão específica. Assim, armazenar valores de números decimais

de até 6 dígitos.

– **DOUBLE**: Semelhante ao tipo FLOAT, mas pode armazenar valores numéricos com uma precisão maior, até 15-16 dígitos. Dessa forma, armazena valores decimais, como valores financeiros e científicos.

> Existem diversos outros tipos, como para cadastrar datas temos DATE, DATETIME, TIME, TIMESTAMP. Porém, nas referências existem artigos que irão aprofundar melhor os tipos de dados possíveis em um banco de dados.

ALTER TABLE

Utilizado para atualizar colunas em uma tabela. Com ele podemos adicionar uma coluna com o comando:

> *ALTER TABLE* **<table_name>** *ADD* **<column_name column_type>**;

Também podemos remover colunas com o comando “DROP COLUMN”:

> *ALTER TABLE* **<table_name>** *DROP COLUMN* **<column_name>**;

Também podemos inserir restrições (Constraints) a uma tabela com o comando “ADD CONSTRAINT”:

> *ALTER TABLE* **<table_name>** *ADD CONSTRAINT* **<constraint_name>** *<column_used_in_constraint>*;

Através do comando ALTER TABLE também podemos inserir chaves primárias (primary keys) e chaves estrangeiras (foreign keys), além de podermos colocar todos os tipos de constraint como UNIQUE, CHECK, DEFAULT e INDEXes.

INSERT

O método insert serve para inserirmos dados em uma tabela. A sintaxe básica é:

> *INSERT INTO* **<table_name> (column1, column2, column3, …, columnN)** *VALUES* **(value1, value2, value3, …, valueN)**;

UPDATE

Este comando é executado a fim de atualizar um registro em uma determinada tabela.

> *UPDATE* **table_name** *SET* **column1 = value1, column2 = value2, …, columnN = valueN** *WHERE* **<condition_clause>**;

Importante: A cláusula de condição é extremamente importante no UPDATE para que você atualize somente as linhas (registros) que você precisa que sejam atualizadas!!!

DELETE

Este comando é executado a fim de deletar um registro em uma determinada tabela.

> *DELETE FROM* **<table_name>** *WHERE* **<condition_clause>**;

Importante: A cláusula de condição é extremamente importante no DELETE para que você delete do banco de dados somente as linhas (registros) que você precisa que sejam deletadas!!!

SELECT

Este comando é utilizado para retornar registros de uma ou mais tabelas.

> *SELECT* **<column1, column2, …, columnN>** *FROM* **<table_name>**;

Referências

> https://4linux.com.br/comandos-basicos-linguagem-sql/ (Comandos básicos do SQL)
> https://www.freecodecamp.org/portuguese/news/comandos-basicos-em-sql-a-lista-de-consultas-e-instrucoes-de-banco-de-dados-que-voce-deve-conhecer/ (Comandos básicos em SQL)

> https://www.w3schools.com/sql/sql_create_db.asp (SQL CREATE DATABASE Statement)
> https://www.homehost.com.br/blog/tutoriais/mysql/tipos-de-dados-do-mysql/ (Tipos de dados do MySQL)
> https://www.w3schools.com/sql/sql_alter.asp (SQL Alter Table Statement)