Estudo Básico de SQL

Estudos Sobre SQL

V1.0.0 (OUT/2023)

Sumário

1 SQL **3**

2 COMANDOS **4**

3 Tipos primitivos **5**

4 Criando tabelas **6**
4.1 Adicionando uma nova coluna . . . 6
4.2 Removendo uma coluna . . . 6
4.3 Escolhendo a posição da coluna . . . 6
4.4 Modificando Definições . . . 6
4.5 Renomeando coluna . . . 7
4.6 Adicionando mais tabelas . . . 7
4.7 Removendo uma linha . . . 7
4.8 Removendo todas as linhas . . . 7

5 MANIPULANDO REGISTROS **8**
5.1 Selecionado Intervalos . . . 8
5.2 Selecionando Valores . . . 8
5.3 Combinando Testes . . . 8
5.4 Usando o Operador Like . . . 9

6 PHPMyAdmin **10**

7 Obtendo Dados das Tabelas **11**
7.1 Selecionando Intervalos . . . 11
7.2 Selecionando Valores . . . 11
7.3 Usando o operador Like . . . 11
7.4 WILDCARDS . . . 12
7.5 Distinguindo . . . 12
7.6 Funções de Agregação . . . 12
7.7 Agrupando e Agregando . . . 12

8 Relacionando as Tabelas **14**
8.1 Adicionando a FOREIGN KEY . . . 14
8.2 Integridade Referencial . . . 14

1 SQL

SQL, ou Structured Query Language, é uma linguagem de programação projetada para gerenciar e consultar sistemas de gerenciamento de bancos de dados relacionais (RDBMS). É a linguagem padrão para interagir com a maioria dos sistemas de bancos de dados relacionais, como MySQL, PostgreSQL, Oracle, SQL Server, SQLite e muitos outros. SQL é uma linguagem poderosa que permite realizar diversas operações em bancos de dados

MySQL: É um programa utilizado para gerenciar bancos de dados.

O banco de dados é formado por 4 fases:

```
1 Basa de dados, sistema gerenciador, Linguagem de exploração e Programas
      adicionais.
```

2 COMANDOS

Para criar um banco de dados use:

```
CREATE DATABASE _cadastro_;
```

Para criar tabelas use:

```
CREATE TABLE _nome do banco_( );
```

- **Deve-se declarar a qual tipo primitivo os campos pertencem**

Para apagar banco de dados use:

```
DROP DATABASE _nome do banco_;
```

Para usar acentuação use:

```
default character set utf8
default collate utf8_general_ci;
```

Para configuração de caractere parão use:

```
DEFAULT CHARSET = utf8;
```

3 Tipos primitivos

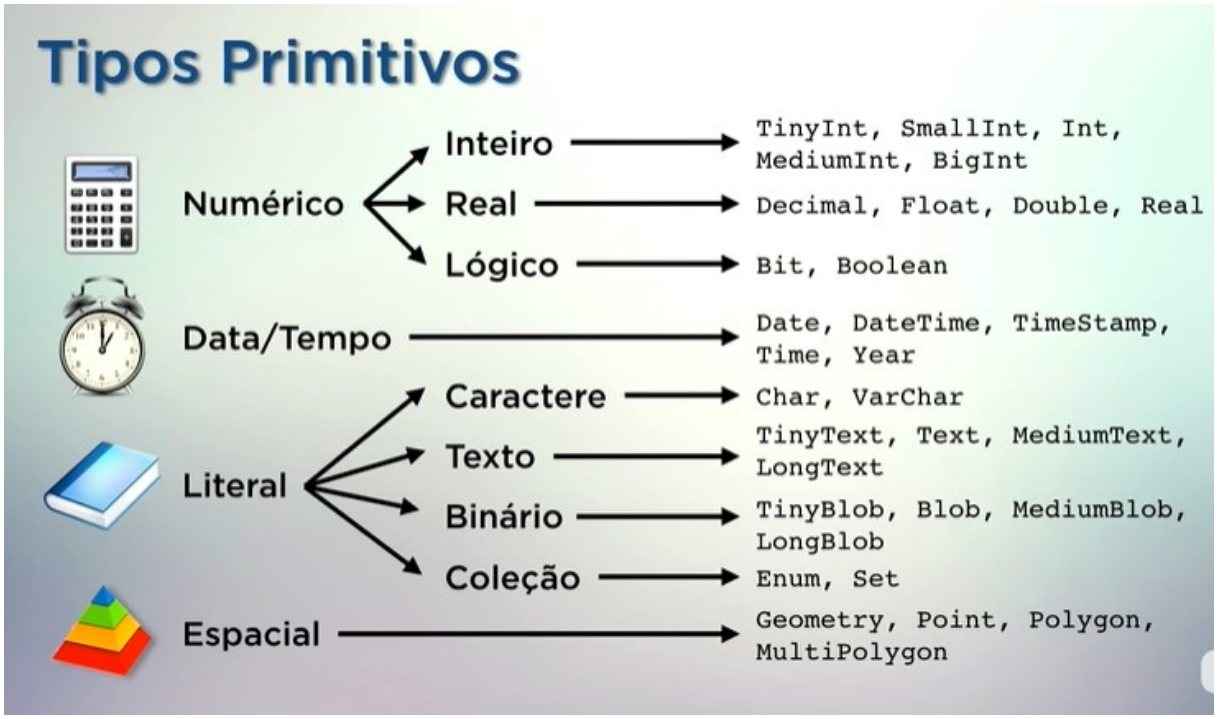

Figura 1: TiposPrimitivos

4 Criando tabelas

Ex:

CREATE TABLE ‘Nome da tabela’ (id **int NOT NULL AUTO_INCREMENT**, nome **varchar(30) NOT NULL**, nascimento **date**, sexo **enum(‘M’, ‘F’)**, peso **decimal(5,2)**, altura **decimal(3,2)**, nacionalidade **varchar(20) DEFAULT ‘Brasil’**, **PRIMARY KEY (id)**) **DEFAULT CHARSET = utf8mb3**;

4.1 Adicionando uma nova coluna

A sintaxe básica é a seguinte:

```
ALTER TABLE nome_da_tabela
ADD nome_da_coluna tipo_de_dado;
```

4.2 Removendo uma coluna

A sintaxe básica é:

```
ALTER TABLE nome_da_tabela
DROP COLUMN nome_da_coluna;
```

4.3 Escolhendo a posição da coluna

A sintaxe básica é:

```
ALTER TABLE nome_da_tabela
ADD COLUMN nome_da_coluna' varchar(10) AFTER ;

ALTER TABLE 'nome ta tabela'
ADD COLUMN 'nome da coluna' int FIRST
```

4.4 Modificando Definições

A sintaxe básica é:

```
ALTER TABLE nome_da_tabela
MODIFY nome_da_coluna NOVO_TIPO_DE_DADO (Novo_Tamanho);
```

4.5 Renomeando coluna

A sintaxe básica é:

```
ALTER TABLE nome_da_tabela
CHANGE COLUMN nome_coluna_antigo nome_coluna_novo TIPO_DE_DADO;
```

4.6 Adicionando mais tabelas

A sintaxe básica é:

```
CREATE TABLE IF NOT EXISTS nome_da_tabela (
coluna1 tipo_de_dado,
coluna2 tipo_de_dado,
...
colunaN tipo_de_dado
```

);

4.7 Removendo uma linha

```
DELETE FROM 'nome da tabela'
WHERE 'nome da coluna' = 'numero da linha';
```

4.8 Removendo todas as linhas

```
TRUNCATE TABLE 'nome da tabela';
```

5 MANIPULANDO REGISTROS

Registros manipulais referem-se à realização de operações que afetam os dados armazenados em uma tabela. As operações de manipulação de registros comuns incluem:

1. Inserir Registros **(INSERT)**: Adicionar novos registros a uma tabela.

   INSERT INTO ‘nome da tabela’ (coluna1, coluna2) VALUES (valor1, valor2);

2. Atualizar Registros **(UPDATE)**: Modifica os valores de registros existentes em uma tabela.

   UPDATE ‘nome da tabela’ SET coluna1 = ‘novo_valor’ WHERE condição;

3. Excluir Registros **(DELETE)**: Remove registros de uma tabela com base em uma condição.

   DELETE FROM ‘nome da tabela’ WHERE condição;

4. Selecionar Registros **(SELECT)** : Recupera registros de uma tabela com base em critérios específicos.

   SELECT coluna1, coluna2 FROM ‘nome da tabela’ WHERE condição;

5.1 Selecionado Intervalos

```
SELECT * FROM 'nome da tabela'
WHERE 'nome da coluna' BETWEEN 'dados da coluna' AND 'dados de outra
    coluna'
ORDER BY 'nome da coluna que vai ordenar os dados';
```

5.2 Selecionando Valores

```
SELECT 'colunas selecionadas' FROM 'tabela de onde deseja recuperar os
    dados'
WHERE 'condição para filtrar os dados'
ORDER BY 'nome da coluna na qual você deseja ordenar';
```

5.3 Combinando Testes

Para combinar condições (testes) em uma consulta SQL, você pode usar os operadores lógicos AND, ORe NOT. Isso permite que você faça consultas mais complexas que envolvem várias condições. Aqui estão os operadores e como você pode usá-los:

1. **E**: Utilize o operador **AND** para combinar duas ou mais condições, onde todas as condições precisam ser verdadeiras para que o registro seja incluído no resultado. Por exemplo:

   SELECT coluna1, coluna2 FROM ‘nome da tabela’ WHERE condição1 AND condição2;

Exemplo combinando condições com **AND**:

```
SELECT nome, idade
FROM clientes
WHERE cidade = 'São Paulo' AND idade > 30;
```

Esta consulta seleciona clientes de São Paulo com mais de 30 anos.

2. **OU**: Use o operador **OR** para combinar duas ou mais condições, onde pelo menos uma das condições precisas seja verdadeira para que o registro seja incluído no resultado. Por exemplo:

   SELECT coluna1, coluna2 FROM 'nome da tabela' WHERE condição1 OR condição2;

Exemplo combinando condições com **OR**:

```
SELECT nome, idade
FROM clientes
WHERE cidade = 'São Paulo' OR cidade = 'Rio de Janeiro';
```

Esta consulta seleciona clientes que são de São Paulo ou do Rio de Janeiro.

3. **NOT**: Utilize o operador **NOT** para negar uma condição, ou seja, selecione registros que não atendem a uma determinada condição. Por exemplo:

   SELECT coluna1, coluna2 FROM 'nome da tabela' WHERE NOT condição;

Exemplo usando **NOT**:

```
SELECT nome, idade
FROM clientes
WHERE NOT cidade = 'São Paulo';
```

Esta consulta seleciona clientes que não são de São Paulo.

5.4 Usando o Operador Like

É usado para realizar pesquisas em colunas de texto, permitindo que você encontre registros que contenham um padrão específico de caracteres. O LIKE é frequentemente usado com os caracteres curinga "%" e "_".

```
SELECT coluna1, coluna2
FROM 'nome da tabela'
WHERE 'coluna texto' LIKE padrão;
```

6 PHPMyAdmin

Navegue pelas Tabelas: Após o login, você verá uma lista de bancos de dados disponíveis no lado esquerdo da interface. Selecione o banco de dados que deseja gerenciar. Isso abrirá uma lista de tabelas no banco de dados.

Executar Tarefas: A partir daqui você pode realizar diversas tarefas, como: **Criar Tabelas**: Clique no banco de dados e, em seguida, selecione a guia "SQL" para executar comandos SQL para criar tabelas. **Inserir Dados**: Clique em uma tabela e use a guia "Inserir" para adicionar registros. **Executar Consultas**: Use um guia "SQL" para escrever e executar consultas SQL. **Gerar Backup**: Utilize o guia "Exportar" para criar backups do seu banco de dados. **Modificar Tabelas**: Você pode alterar a estrutura de tabelas existentes, adicionar ou remover colunas, etc.

7 Obtendo Dados das Tabelas

7.1 Selecionando Intervalos

Sintaxe simples:

```
SELECT coluna1, coluna2, ...
FROM 'nome da tabela'
WHERE coluna BETWEEN 'valor minimo' AND 'valor maximo';
```

Exemplo:

```
SELECT nome, idade
FROM clientes
WHERE idade BETWEEN 18 AND 30;
```

Neste exemplo, a consulta selecionará os nomes e idades dos clientes cujas idades estão entre 18 e 30 anos.

7.2 Selecionando Valores

Sintaxe simples:

```
SELECT coluna1, coluna2, ...
FROM 'nome da tabela'
ORDER BY 'coluna de ordenacao';
```

Exemplo:

```
SELECT nome, idade, cidade
FROM clientes
ORDER BY cidade ASC, idade DESC;
```

Isso classificará os resultados por “cidade” em ordem crescente e, para registros com a mesma cidade, classificará por “idade” em ordem decrescente.

7.3 Usando o operador Like

É usado para buscar registros que exigem um padrão de texto específico em uma coluna de texto (geralmente, strings). Ele é frequentemente usado quando você deseja recuperar registros com base em correspondências parciais, ou seja, quando você não precisa de uma correspondência exata.

sintaxe básica:

```
SELECT colunas
FROM 'nome da tabela'
WHERE coluna LIKE padrão;
```

7.4 WILDCARDS

Curingas (caracteres curinga) são caracteres especiais usados em conjunto com o operador LIKEem consultas SQL para representar padrões de texto flexíveis. Os dois curingas mais comuns são:

% (Porcentagem): O caractere %representa zero, um ou vários caracteres em uma consulta. Por exemplo, use 'Jo%'em uma consulta que LIKEdeverá corresponder a qualquer string que comece com "Jo" e tenha qualquer número de caracteres após "Jo". Exemplo: 'John', 'Joseph', 'Jones', etc.

_ (Sublinhado): O caractere _representa um único caractere em uma consulta. Por exemplo, use 'Sm_th'em uma consulta como LIKEcorresponderá a qualquer string de seis caracteres que começam com "Sm", tenha um caractere em qualquer posição e termine com "th". Exemplo: 'Smith', 'Smath', etc.

7.5 Distinguindo

```
SELECT DISTINCT 'nome da tabela' FROM 'nome da tabela';
```

7.6 Funções de Agregação

```
SELECT COUNT(nome da coluna) FROM 'nome da tabela'

SELECT MAX(nome da coluna) FROM 'nome da tabela'

SELECT MIN(nome da coluna) FROM 'nome da tabela'

SELECT SUM(nome da coluna) FROM 'nome da tabela'

SELECT AVG(nome da coluna) FROM 'nome da tabela'
```

7.7 Agrupando e Agregando

1. Agrupando **(GROUP BY)**: A cláusula GROUP BYé usada para agrupar registros em conjuntos com base nos valores de uma ou mais colunas. Quando você usa GROUP BY, os registros são divididos em grupos com base nos valores das colunas especificadas. Você geralmente agrupa por colunas que deseja usar como classificações de agrupamento.

Ex: SELECT departamento, COUNT(*) FROM funcionarios GROUP BY departamento;

2. Agregando **(Funções de Agregação)**: As funções de agregação (como COUNT, SUM, AVG, MAXe MIN) são usadas para calcular resumos ou totais em grupos de registros. Essas funções funcionam em um grupo de registros, retornando um único valor agregado com base nas colunas especificadas.

Ex:

```
SELECT departamento, AVG(salario)
FROM funcionarios
GROUP BY departamento;
```

8 Relacionando as Tabelas

8.1 Adicionando a FOREIGN KEY

```
1 ALTER TABLE tabela_atual
2 ADD CONSTRAINT 'nome da chave estrangeira'
3 FOREIGN KEY (colunaatual)
4 REFERENCES 'tabela referenciada' (coluna referenciada);
```

8.2 Integridade Referencial

A integridade referencial é importante porque garante que os dados no banco de dados permaneçam consistentes e evite a inserção de dados inválidos ou inconsistentes. Ela é fundamental para manter a precisão e a confiabilidade dos dados em sistemas de gerenciamento de banco de dados relacionados.

Sintaxe:

```
1 DELETE FROM 'nome da tabela'
2 WHERE nomeda coluna = 'numero da linha'
```

QUERY: É uma consulta.