Aula de 16/11/2022: Introdução ao SQL

SQL – Structured Query Language

- Linguagem de Consulta Estruturada.
- Usada para armazenar, manipular e recuperar registros em um Banco de Dados relacional:
- Relacional = dados relacionados entre si em tabelas.
  - **SGBDR** – Sistema Gerenciador de Banco de Dados Relacional.
  - **RDBMS** – Relational Database Management System.
  - Sistemas que usam SQL (alguns exemplos): Microsoft Access, Microsoft SQL Server, Microsoft Azure SQL Database, MySQL, MariaDB, Oracle Database, PostgreSQL, SQLite.

DML – Data Manipulation Language:

- Linguagem de Manipulação de Dados.
- Realiza inclusões, alterações e exclusões de registros.
- Comandos: INSERT, UPDATE e DELETE.

DDL – Data Definition Language:

- Linguagem de Definição de Dados.
- Define novas tabelas e novos elementos.
- Comandos: CREATE, ALTER, DROP.

DCL – Data Control Language:

- Linguagem de Controle de Dados.
- Defini o tipo de acesso do usuário.
- Comandos: GRANT, REVOKE.

DTL – Data Transaction Language:

- Linguagem de Transação de Dados.
- Uma transação é uma propagação de uma ou mais mudanças no banco de dados.
- Comandos: BEGIN WORK, COMMIT, ROLLBACK, SAVEPOINT.

DQL – Data Query Language:

- Linguagem de Consulta de Dados.
- Realizar buscas específicas.
- Comandos: SELECT.

Cláusulas:

- Comandos: FROM, WHERE, GROUP BY, HAVING, ORDER BY, DISTINCT, UNION.

Operadores Lógicos:

- AND, OR, NOT.

Operadores Relacionais:

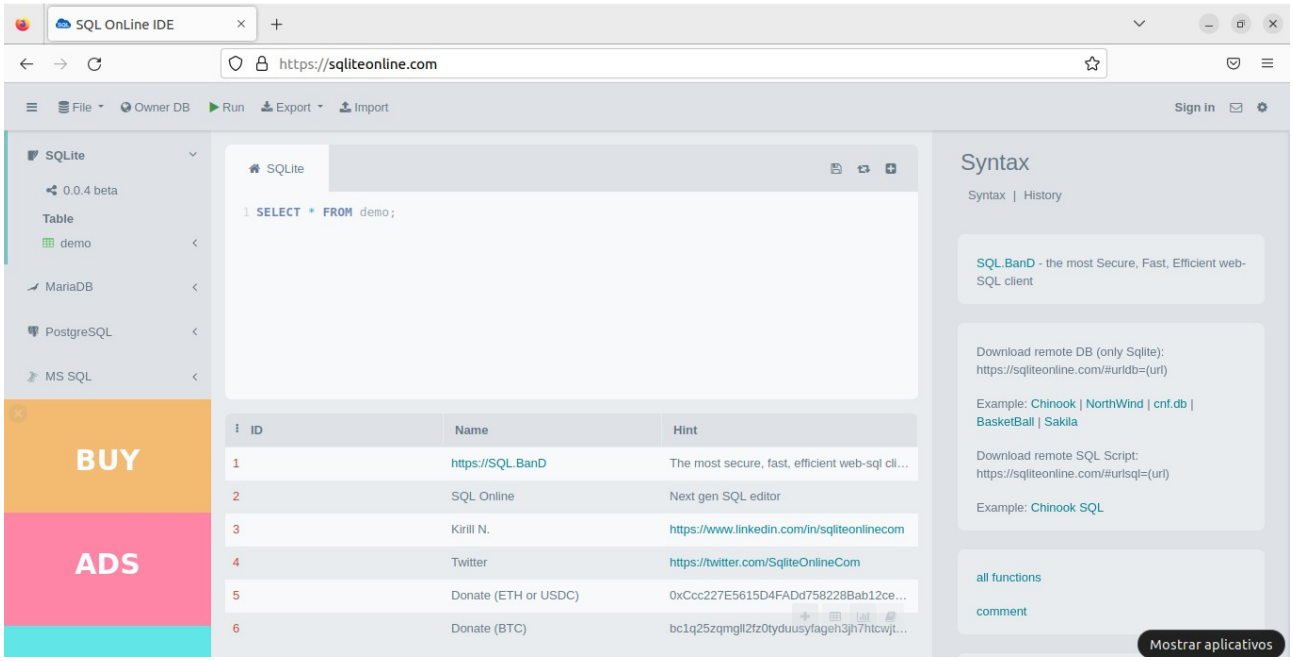
- Operadores: <, >, <=, >=, =, <>.
- Termos: BETWEEN, LIKE, IN.

Funções de Agregação:

- AVG, COUNT, SUM, MAX, MIN, STDDEV, VARIANCE.

Primeiros Passos

Utilize o editor online: https://sqliteonline.com/



Vamos usar, inicialmente, a tabela `demo` (demonstração) para aprender alguns comandos básicos; essa tabela possui três colunas: `ID`, `Name` e `Hint`.

1. "*Selecione todas as colunas da tabela demo*":

```
SELECT * FROM demo;
```

Sintaxe:

```
SELECT colunas  FROM nome_da_tabela;
```

Nota:

- `SELECT` significa ***selecionar***; `FROM` significa ***a partir de*** ou ***de*** (sentido de origem).
- Se forem todas as colunas, usar "`*`"; para especificar uma coluna, coloque o nome da mesma; se for mais de 1 coluna, separe-as usando vírgula.

Lembrar que:

- O SQL **não é Case Sensitive**, ou seja, não importa se os comandos forem escritos em letras maiúsculas ou minúsculas. Ex: `select` é o mesmo que `SELECT`.
- **Por convenção**, vamos escrever todos os comandos com LETRAS MAIÚSCULAS.
- Usar ponto-e-vírgula ao final da linha de comando.

2. “*Selecione todas as colunas da tabela demo onde a ID for igual a 4*”:

```
SELECT * FROM demo WHERE ID = 4;
```

Sintaxe:

```
SELECT colunas  FROM nome_da_tabela WHERE condição;
```

Nota:

- `WHERE` significa ***onde***, ou ***no caso em que***.
- Após o `WHERE`, deve vir uma condição que deve ser satisfeita para que o comando `SELECT` apresente os registros.

Para escrever uma condição, é necessário usar operadores relacionais:

- `=` Igual
- `>` Maior
- `<` Menor
- `>=` Maior igual
- `<=` Menor igual
- `<>` Diferente (`!=`)
- `BETWEEN` (significa ***entre***): Procura um registro entre um valor mínimo e um valor máximo.
- `LIKE` (significa ***como***): Procura de acordo com um modelo.
- `IN` (significa ***dentro***): Procura um registro que seja igual a uma lista de valores aceitados.

3. “*Selecione todas as colunas da tabela demo onde Name é igual a ‘Twitter’ e a ID é menor que 5*”:

```
SELECT * FROM demo WHERE Name = ‘Twitter’ AND ID < 5;
```

Sintaxe:

```
SELECT colunas  FROM nome_da_tabela WHERE condição1 AND condição2;
```

Lembrar que:

- No caso de strings (texto), usar aspas simples: ***‘string’***.

Para escrever mais de uma condição, usamos os operadores lógicos:

- **`AND`** As condições devem ser todas verdadeiras.
- **`OR`** Ao menos uma condição deve ser verdadeira.
- **`NOT`** A condição deve ser falsa.

No caso do exemplo, como a `ID` cujo `Name` é igual a `‘Twitter’` é 4, e 4 é menor que 5, as duas condições são verdadeiras. Logo, o comando **`SELECT`** exibe o registro que satisfaz as duas condições.

Caso a condições seja alterada:

```
SELECT * FROM demo WHERE Name = ‘Twitter’ AND ID > 5;
```

o comando **`SELECT`** não retorna nenhum registro, pois a segunda condição é falsa, pois 4 não é maior que 5.

O **`NOT`** pode ser usado por apenas uma condição. Veja o exemplo:

```
SELECT * FROM demo WHERE NOT ID = 5;
```

nesse caso, esse comando é o mesmo que dizer “*Selecione todas as colunas da tabela demo onde a ID não é 5*”.

4. “*Selecione todas as colunas da tabela demo e ordene pela ID de forma ascendente*”:

```
SELECT * FROM demo ORDER BY ID ASC;
```

Sintaxe:

```
SELECT colunas  FROM nome_da_tabela ORDER BY coluna ASC/DESC;
```

Nota:

- `ORDER BY` significa ***ordene por*** .
- `ASC` significa ***ascendente***, e quer dizer “*Em ordem alfabética de A a Z, e numérica crescente*”.
- `DESC` significa ***descendente***, e quer dizer “*Em ordem alfabética invertida de Z a A, e numérica decrescente*”.

Para modificar a ordenação de mais de uma coluna, use vírgulas, como por exemplo:

```
SELECT * FROM demo ORDER BY ID ASC, Name DESC;
```

5. “*Insira na tabela demo os valores: (ID) 26, (Name) Paula, (Hint) Professora*”:

```
INSERT INTO demo VALUES(26, ‘Paula’, ‘Professora’);
```

Sintaxe:

```
INSERT INTO tabela VALUES(valor_da_coluna1, valor_da_coluna2, …, valor_da_ultima_coluna);
```

Nota:

- `INSERT` significa ***insira***, ***coloque***.
- `INTO` significa ***em***, ***dentro de***.
- `VALUES` significa ***valores***.

O comando `INSERT` cria um novo registro na tabela selecionada. A sintaxe apresentada aqui, no momento, é a mais simples, que consiste em preencher todas as colunas de uma vez e manualmente. Seria o mesmo que preencher uma nova linha numa tabela do Excel.
As outras formas de inserção serão apresentadas posteriormente.

Você pode testar se a nova linha foi registrada corretamente com o comando:

```
SELECT * FROM demo WHERE ID > 25;
```

6. *"Atualize a tabela demo colocando na coluna Hint o valor "Física" onde a ID for igual a 26"*:

```
UPDATE demo SET Hint='Física' WHERE ID = 26;
```

Sintaxe:

```
UPDATE tabela SET coluna1=valor1 WHERE condição;
```

Nota:

- `UPDATE` significa ***atualizar***, ***modificar***.
- `SET` significa ***configurar***, ***colocar***.

Lembrar:

- Não se esqueça do comando `WHERE`: sem ele, todas as linhas da coluna selecionada serão alteradas!
- Se quiser modificar mais colunas com a mesma condição, use virgulas para separar os valores:
  ```
  UPDATE tabela SET coluna1=valor1, coluna2=valor2,
  coluna3=valor3,…  WHERE condição;
  ```
- Você pode testar se a nova linha foi atualizada corretamente com o comando:
  ```
  SELECT * FROM demo WHERE ID = 26;
  ```

7. *"Apague (todas as linhas) da tabela demo onde a ID for maior que 25"*:

```
DELETE FROM demo WHERE ID > 25;
```

Sintaxe:

```
DELETE FROM tabela WHERE condição;
```

Nota:

- `DELETE` significa ***deletar***, ***apagar***.
- Não se esqueça do comando `WHERE`: sem ele, todas as linhas da tabela serão apagadas.
- Você pode testar se a nova linha foi registrada corretamente com o comando (que deve mostrar nenhum registro):
  ```
  SELECT * FROM demo WHERE ID > 25;
  ```