DML

Linguagem de Manipulação de Dados (ou DML, de Data Manipulation Language) é uma família de linguagens de computador utilizadas para a recuperação, inclusão, remoção e modificação de informações em bancos de dados. Pode ser procedural, que especifica como os dados devem ser obtidos do banco; pode também ser declarativa (não procedural), em que os usuários não necessitam especificar o caminho de acesso, isto é, como os dados serão obtidos. O padrão SQL é não procedural. DMLs foram utilizadas inicialmente apenas por programas de computador, porém (com o surgimento da SQL) também têm sido utilizadas por pessoas.

Principais comandos

- INSERT
- UPDATE
- DELETE
- SELECT*

INSERT

Função: inserir uma linha em uma tabela.

Sintaxe: `INSERT INTO tabela (coluna1, coluna2, colunaN) VALUES (valor1, valor2, valorN);`

Exemplo 1 – Inserindo as primeiras linhas

1 – Criar uma base de dados chamada teste.

2 – Conectar na base teste.

3 – Criar uma tabela chamada usuário com os campos abaixo.

4 – Inserir um registro na tabela.

```
1 - CREATE DATABASE teste;
2 - USE teste;
3 - CREATE TABLE usuario (
        id          INT,
        email       VARCHAR(80),
        senha       VARCHAR(10),

        PRIMARY KEY (id)
);
```

```
INSERT INTO usuario (id,email,senha) VALUES (3,'trust@email.com', 'ef789');
```

Tabela Usuario

ID	Email	Senha
1	alice@email.com	ab123
2	bob@email.com	cd456
3	trust@email.com	ef789

Como saber qual id inserir?

Exemplo 2 – Usando AUTO_INCREMENT

```
CREATE TABLE usuario (
id          INT AUTO_INCREMENT,
email       VARCHAR(80),
senha       VARCHAR(60),
```

```
PRIMARY KEY (id)
);

INSERT INTO usuario (email,senha) VALUES ('alice@email.com', 'ab123');
INSERT INTO usuario (email,senha) VALUES ('bob@email.com'  , 'cd456');
INSERT INTO usuario (email,senha) VALUES ('trust@email.com', 'ef789');
```

Agora não precisamos mais nos preocupar em inserir o id, pois ele será criado e incrementado automaticamente pelo mysql. O Id será inserido automaticamente em ordem crescente.

Exemplo 3 – Tornar os campos obrigatórios com NOT NULL

É possível deixar campos em branco em nossa tabela, deixando os dados inconsistentes.

```
INSERT INTO usuario (email,senha) VALUES ('', 'ab123');
INSERT INTO usuario (email,senha) VALUES ('bob@email.com'  , '');
INSERT INTO usuario (email,senha) VALUES ('', '');
```

Solução: usar NOT NULL nas colunas que são obrigatórias.

```
DROP TABLE usuario;
CREATE TABLE usuario (
id          INT          NOT NULL AUTO_INCREMENT,
email       VARCHAR(80)  NOT NULL,
senha       VARCHAR(60)  NOT NULL,

PRIMARY KEY (id)
);
INSERT INTO usuario (email,senha) VALUES ('alice@email.com', 'ab123');
INSERT INTO usuario (email,senha) VALUES ('bob@email.com'  , 'cd456');
INSERT INTO usuario (email,senha) VALUES ('trust@email.com', 'ef789');
```

Exemplo 4 – Inserindo mais de uma linha.

Separando cada conjunto de dados através de parênteses pode-se inserir diversas linhas em um único comando.

```
INSERT INTO usuario (email,senha) VALUES
('alice@email.com','ab123'),('bob@email.com','cd456'),('trust@email.com','ef789');
```

Exemplo 5 – Inserindo sem identificar as colunas

Não identificando s colunas é obrigatório inserir dados em todas as colunas, inclusive noID.

```
INSERT INTO usuario VALUES
('1','alice@email.com','ab123'),('2', 'bob@email.com','cd456'),( '3','trust@email.com','ef789');
```

Exemplo 6 – Inserindo dados através de uma consulta

Neste caso deve-se tomar cuidado com valores UNIQUE e conferir o tipo de dados caso o `INSERT` seja para uma tabela nova. A palavra reservada `VALUES` não é usada.

```
INSERT INTO usuario2(nome,senha) SELECT nome,senhaFROM usuario
```

UPDATE

Função: Atualizar uma ou mais colunas de uma ou mais linhas de uma tabela.
Sintaxe:

```
UPDATE TABELA SET
       coluna1='novo_valor',
       coluna2='novo_valor'
WHERE CONDIÇÃO
```

Exemplo 1 – ATUALIZAR "Sem querer querendo"

```
UPDATE usuario SET
       email='alice_novo@email.com',
       senha='zxy987';
```

```
mysql> select * from usuario;
+----+-----------------+--------+
| id | email           | senha  |
+----+-----------------+--------+
|  1 | alice@email.com | ab123  |
|  2 | bob@email.com   | cd456  |
|  3 | trust@email.com | ef789  |
+----+-----------------+--------+
3 rows in set (0.00 sec)

mysql> UPDATE usuario SET
    ->        email='alice_novo@email.com',
    ->        senha='zxy987';
Query OK, 3 rows affected (0.03 sec)
Rows matched: 3  Changed: 3  Warnings: 0

mysql> select * from usuario;
+----+---------------------+--------+
| id | email               | senha  |
+----+---------------------+--------+
|  1 | alice_novo@email.com | zxy987 |
|  2 | alice_novo@email.com | zxy987 |
|  3 | alice_novo@email.com | zxy987 |
+----+---------------------+--------+
3 rows in set (0.00 sec)
```

Como não definimos a condição para atualizar, o comando UPDATE atualizará toda a tabela.

Exemplo 2 – ATUALIZAR "Sem querer querendo 2"

```
INSERT INTO usuario (email,senha) VALUES ('alice@email.com', 'ab123');
INSERT INTO usuario (email,senha) VALUES ('bob@email.com'  , 'cd456');
INSERT INTO usuario (email,senha) VALUES ('trust@email.com', 'ef789');
```

Agora quero alterar apenas a senha da Alice, deixando o e-mail o mesmo.

```
UPDATE usuario SET
       email='',
       senha='zxy987'
WHERE id=1;
```

```
mysql> select * from usuario;
+----+-----------------+-------+
| id | email           | senha |
+----+-----------------+-------+
|  1 | alice@email.com | ab123 |
|  2 | bob@email.com   | cd456 |
|  3 | trust@email.com | ef789 |
+----+-----------------+-------+
3 rows in set (0.00 sec)

mysql> UPDATE usuario SET
    ->        email='',
    ->        senha='zxy987'
    -> WHERE id=1;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from usuario;
+----+-----------------+--------+
| id | email           | senha  |
+----+-----------------+--------+
|  1 |                 | zxy987 |
|  2 | bob@email.com   | cd456  |
|  3 | trust@email.com | ef789  |
+----+-----------------+--------+
3 rows in set (0.00 sec)
```

Acontece que quando colocamos uma coluna na cláusula *WHERE* ela **SERÁ** alterada. Para deixá-la inalterada apenas não a coloque no *WHERE*.

Exemplo 3 – ATUALIZANDO CORRETAMENTE ATRAVÉS DO ID

```
INSERT INTO usuario (email,senha) VALUES ('alice@email.com', 'ab123');
INSERT INTO usuario (email,senha) VALUES ('bob@email.com'  , 'cd456');
INSERT INTO usuario (email,senha) VALUES ('trust@email.com', 'ef789');

UPDATE usuario SET
       senha='zxy987'
WHERE id=1;
```

Pronto. Agora será atualizado somente o registro com id=1. Como a coluna id é única apenas um e somente um registro será alterado.

```
mysql> select * from usuario;
+----+-----------------+-------+
| id | email           | senha |
+----+-----------------+-------+
|  1 | alice@email.com | ab123 |
|  2 | bob@email.com   | cd456 |
|  3 | trust@email.com | ef789 |
+----+-----------------+-------+
3 rows in set (0.00 sec)

mysql> DELETE FROM usuario WHERE id=2;
Query OK, 1 row affected (0.04 sec)

mysql> select * from usuario;
+----+-----------------+-------+
| id | email           | senha |
+----+-----------------+-------+
|  1 | alice@email.com | ab123 |
|  3 | trust@email.com | ef789 |
+----+-----------------+-------+
2 rows in set (0.00 sec)

mysql>
```

Exemplo 4 – ATUALIZANDO MAIS DE UM REGISTRO

```
INSERT INTO usuario (email,senha) VALUES ('alice@email.com', 'ab123');
INSERT INTO usuario (email,senha) VALUES ('bob@email.com'  , 'cd456');
```

```sql
INSERT INTO usuario (email,senha) VALUES ('trust@email.com', 'ef789');
INSERT INTO usuario (email,senha) VALUES ('aline@email.com', 'gh012');

UPDATE usuario SET
        senha='123'
WHERE email like 'a%';
```

```
mysql> select * from usuario;
+----+------------------+-------+
| id | email            | senha |
+----+------------------+-------+
|  1 | alice@email.com  | ab123 |
|  2 | bob@email.com    | cd456 |
|  3 | trust@email.com  | ef789 |
|  4 | aline@email.com  | gh012 |
+----+------------------+-------+
4 rows in set (0.00 sec)

mysql> UPDATE usuario SET
    ->         senha='123'
    -> WHERE email like 'a%';
Query OK, 2 rows affected (0.04 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> select * from usuario;
+----+------------------+-------+
| id | email            | senha |
+----+------------------+-------+
|  1 | alice@email.com  | 123   |
|  2 | bob@email.com    | cd456 |
|  3 | trust@email.com  | ef789 |
|  4 | aline@email.com  | 123   |
+----+------------------+-------+
4 rows in set (0.00 sec)
```

DELETE

Função: Deletar uma linha ou mais linhas de uma tabela.
Sintaxe: `DELETE FROM tabela WHERE CONDIÇÃO`

Exemplo 1 – DELETANDO "Sem querer querendo"

```
DELETE FROM usuario
```

```
mysql> DELETE FROM usuario;
Query OK, 3 rows affected (0.06 sec)

mysql> _
```

Como não definimos nenhuma condição a instrução DELETE apagou todas as linhas da tabela. Para que isso não aconteceça devemos **SEMPRE** especificar uma condição através da cláusula WHERE.

Exemplo 2 – DELETANDO WHERE id=?

```
INSERT INTO usuario (email,senha) VALUES ('alice@email.com', 'ab123');
INSERT INTO usuario (email,senha) VALUES ('bob@email.com'  , 'cd456');
INSERT INTO usuario (email,senha) VALUES ('trust@email.com', 'ef789');

DELETE FROM usuario WHERE id=2;
```

Pronto. Agora será deletado somente o registro com id=2. Como a coluna id é única apenas um e somente um registro será deletado.

```
mysql> select * from usuario;
+----+-----------------+-------+
| id | email           | senha |
+----+-----------------+-------+
|  1 | alice@email.com | ab123 |
|  2 | bob@email.com   | cd456 |
|  3 | trust@email.com | ef789 |
+----+-----------------+-------+
3 rows in set (0.00 sec)

mysql> DELETE FROM usuario WHERE id=2;
Query OK, 1 row affected (0.04 sec)

mysql> select * from usuario;
+----+-----------------+-------+
| id | email           | senha |
+----+-----------------+-------+
|  1 | alice@email.com | ab123 |
|  3 | trust@email.com | ef789 |
+----+-----------------+-------+
2 rows in set (0.00 sec)

mysql>
```

Exemplo 3 – DELETANDO MAIS DE UM REGISTRO

```
INSERT INTO usuario (email,senha) VALUES ('alice@email.com', 'ab123');
```

```
INSERT INTO usuario (email,senha) VALUES ('bob@email.com'  , 'cd456');
INSERT INTO usuario (email,senha) VALUES ('trust@email.com', 'ef789');
INSERT INTO usuario (email,senha) VALUES ('aline@email.com', 'gh012');


DELETE FROM usuario WHERE email like 'a%';
```

Agora serão deletados todos os registros em que começarem com a letra a no campo e-mail.

```
mysql> select * from usuario;
+----+------------------+--------+
| id | email            | senha |
+----+------------------+--------+
|  1 | alice@email.com | ab123 |
|  2 | bob@email.com    | cd456 |
|  3 | trust@email.com | ef789 |
|  4 | aline@email.com | gh012 |
+----+------------------+--------+
4 rows in set (0.00 sec)

mysql> DELETE FROM usuario WHERE email like 'a%';
Query OK, 2 rows affected (0.04 sec)

mysql> select * from usuario;
+----+------------------+--------+
| id | email            | senha |
+----+------------------+--------+
|  2 | bob@email.com    | cd456 |
|  3 | trust@email.com | ef789 |
+----+------------------+--------+
2 rows in set (0.00 sec)
```