SQL - Structured Query Language
PARTE 2

4. DML - LINGUAGEM PARA MANIPULAÇÃO DE DADOS

Entende-se como manipulação de dados a inserção (INSERT), modificação (UPDATE) e apagamento (DELETE) de dados de tabelas. Existe ainda a necessidade de efetuar consultas a esses dados armazenados utilizando o comando SELECT. Neste documento (parte 2) você irá conhecer os comandos básicos DML e sua aplicabilidade.

4.1. Inserção de dados (INSERT)

O comando INSERT tem as seguintes sintaxes:

```
INSERT INTO (tabela|vista) [(nome_col1, nome_col2, …)] VALUES (valor1, valor2, …);
ou
INSERT INTO (tabela|vista) [(nome_col1, nome_col2, …)]
Subconsulta;
```

A primeira opção insere apenas um registro de cada vez enquanto que a segunda insere um ou mais registros. Na segunda opção, os valores a inserir são retirados de uma subconsulta.

Exemplo 1 - Inserção de um registro e uso dos nomes das colunas da tabela destino:

```
INSERT INTO empregados (numero, nome, funcao, salario, premios, num_departamento)
VALUES (1839, 'Jorge Sampaio', 'Presidente', 49000, null, 10);
```

O primeiro conjunto de parêntesis serve apenas para indicar a que correspondem os valores que estão a ser inseridos. É facultativo desde que os valores mencionados a seguir à cláusula VALUE estejam na mesma ordem das colunas da tabela e desde que sejam todos mencionadas. Assim, o comando do exemplo 1 produz o mesmo resultado que o comando do exemplo 2.

Exemplo 2 – Inserção de um registro sem especificar os nomes das colunas (produz o mesmo resultado que o exemplo 1):

```
INSERT INTO empregados
VALUES (1839, 'Jorge Sampaio',  'Presidente', 49000, null, 10);
```

Exemplo 3 – Inserção de um registro sem incluir todos os dados e alterando a ordem de algumas colunas (produz o mesmo resultado que os dois exemplos anteriores):

```
INSERT INTO empregados (nome, funcao, numero, salario)
VALUES ('Jorge Sampaio',  'Presidente', 1839, 49000)
```

Dados para inserção na tabela departamentos:

```
INSERT INTO departamentos VALUES (10, 'Contabilidade', 'Condeixa');
INSERT INTO departamentos VALUES (20, 'Investigação', 'Mealhada');
INSERT INTO departamentos VALUES (30, 'Vendas', 'Coimbra');
INSERT INTO departamentos VALUES (40, 'Planejamento', 'Montemor');
```

A tabela departamento ficará assim:

num_departamento	nome	localizacao
10	Contabilidade	Condeixa
20	Investigação	Mealhada
30	Vendas	Coimbra
40	Planejamento	Montemor
NULL	NULL	NULL

Dados para inserção na tabela empregados:

```
INSERT INTO empregados VALUES(1566, 'Augusto Reis', 'Encarregado', 4509,  null, 20);
INSERT INTO empregados VALUES(1698, 'Duarte Guedes',  'Encarregado', 3808,  null, 30);
INSERT INTO empregados VALUES(1782, 'Silvia Teles',  'Encarregado', 2794,  null, 10);
INSERT INTO empregados VALUES(1788, 'Maria Dias', 'Analista', 5650,  null, 20);
INSERT INTO empregados VALUES(1902, 'Catarina Silva', 'Analista', 4350,  null, 20);
INSERT INTO empregados VALUES(1499, 'Joana Mendes', 'Vendedor', 1456, 5630, 30);
INSERT INTO empregados VALUES(1521, 'Nelson Neves', 'Vendedor', 2122, 9850, 30);
INSERT INTO empregados VALUES(1654, 'Ana Rodrigues',  'Vendedor', 2212, 8140, 30);
INSERT INTO empregados VALUES(1844, 'Manuel Madeira', 'Vendedor', 1578, 0, 30);
INSERT INTO empregados VALUES(1900, 'Tome Ribeiro', 'Continuo', 5695,  null, 30);
INSERT INTO empregados VALUES(1876, 'Rita Pereira', 'Continuo', 6510,  null, 20);
INSERT INTO empregados VALUES(1934, 'Olga Costa', 'Continuo', 6830,  null, 10);
INSERT INTO empregados VALUES(1369, 'Antonio Silva',  'Continuo', 7080,  null, 20);
```

A tabela empregados ficará assim:

numero	nome	funcao	salario	premios	num_departamento
1369	Antonio Silva	Continuo	7080.00	NULL	20
1499	Joana Mendes	Vendedor	1456.00	5630	30
1521	Nelson Neves	Vendedor	2122.00	9850	30
1566	Augusto Reis	Encarregado	4509.00	NULL	20
1654	Ana Rodrigues	Vendedor	2212.00	8140	30
1698	Duarte Guedes	Encarregado	3808.00	NULL	30
1782	Silvia Teles	Encarregado	2794.00	NULL	10
1788	Maria Dias	Analista	5650.00	NULL	20
1844	Manuel Madeira	Vendedor	1578.00	0	30
1876	Rita Pereira	Continuo	6510.00	NULL	20
1900	Tome Ribeiro	Continuo	5695.00	NULL	30
1902	Catarina Silva	Analista	4350.00	NULL	20
1934	Olga Costa	Continuo	6830.00	NULL	10
NULL	NULL	NULL	NULL	NULL	NULL

4.2. Extração de Dados – (SELECT)

O comando SELECT é o mais importante e mais complexo de todos os comandos de SQL. O seu objetivo é o de selecionar dados, podendo para tal, aplicar vários tipos de relação (restrição, projeção, produto, junção, união, intersecção e diferença) às tabelas existentes na base de dados e executar operações sobre os valores retirados das tabelas antes de mostrá-los.

O comando de ***query*** (pesquisa) básico incluí apenas as cláusulas SELECT e FROM.

a) Cláusulas SELECT e FROM

A cláusula SELECTespecifica uma lista de nomes de colunas das tabelas separadas por vírgulas. Permite ainda fazer operações aritméticas, de *strings* e de datas sobre os valores selecionados. Permite ainda renomear as colunas de dados através de pseudônimos.
A cláusula FROM determina de que tabelas se vão buscar os dados.

Exemplos:

- Query simples:

```
SELECT nome, numero, salario FROM empregados;
```

Seleciona as colunas nome, numero e salario da tabela empregados.

- Uso de expressões aritméticas:

```
SELECT nome, salario * 14
FROM empregados;
```

Seleciona a coluna nomee seleciona o resultado da multiplicação da coluna salario por 14 (o salário anual já a contar com 13º mês e subsídio de férias)

- Uso de pseudônimos (ou alias em inglês)

```
SELECT nome, salario * 14 'Remuneração Anual'
FROM empregados;
```

Seleciona as colunas nome e a multiplicação da coluna salario por 14 (o salário anual já a contar com 13º mês e subsídio de férias) e atribuí o pseudônimo de "Remuneração Anual" à segunda coluna.

- Concatenação de colunas

```
SELECT CONCAT (nome, "/", funcao) 'Nome / função'
FROM empregados;
```

Junta os valores de nome e função numa string e mostra-os numa coluna cujo nome é “Nome mais funcao”.

- Uso de constantes

```
SELECT CONCAT ('O Exmo Sr. '," ",nome ," ", ' trabalha como ' ," ",funcao)  'Descrição Formal'
FROM empregados;
```

Junta algumas constantes aos campos nome e funcao e concatena tudo numa única coluna de nome “Descricao Formal”.

NOTA: Repare que as *strings* são limitadas por plicas (‘ e ’) e os *alias* são limitados por aspas (“ e ”).

- Tratamento de valores nulos

```
SELECT nome, IFNULL(premios, 0) AS premios
FROM empregados;
```

Quando um determinado campo não tem qualquer valor atribuído, é usado um valor especial, o NULL. Um campo a NULL significa falta de valor ou falta de informação e não deve ser confundido com a string vazia, ‘’, nem com o valor zero (0). Não é possível realizar operações aritméticas ou de caracteres sobre valores nulos. Para isso, usa-se a função ifnull para substituiu o primeiro parâmetro pelo segundo no caso do primeiro ser nulo.

Exemplo 1:

```
SELECT IFNULL(nome, 'Sem nome'), salario
FROM empregados;
```

Se o empregado não tiver um nome associado, o valor dessa coluna aparece como ‘Sem nome’.

Exemplo 2:

```
SELECT nome, salario * 14 + IFNULL(premios, 0) 'Ganho Anual'
FROM empregados;
```

b) Cláusula DISTINCT

A cláusula DISTINCT elimina linhas duplicadas do resultado.

```
SELECT DISTINCT funcao 'Mostra que profissões existem'
FROM empregados;
```

O comando anterior mostra apenas uma lista das profissões existentes. Se não usássemos o DISTINCT apareceriam as funções de **todos** os trabalhadores mesmo que existissem funções

repetidas.

Exemplos:

```
SELECT funcao 'Com Repetidos'
FROM empregados;
```

Com Repetidos
Continuo
Vendedor
Vendedor
Encarregado
Vendedor
Encarregado
Encarregado
Analista
Vendedor
Continuo
Continuo
Analista
Continuo

```
SELECT DISTINCT funcao 'Sem Repetidos'
FROM empregados;
```

Sem Repetidos
Continuo
Vendedor
Encarregado
Analista

```
SELECT DISTINCT funcao, num_departamento 'Profissão e departamento'
FROM empregados
ORDER BY num_departamento, funcao;
```

funcao	Profissao e departamento
Continuo	10
Encarregado	10
Analista	20
Continuo	20
Encarregado	20
Continuo	30
Encarregado	30
Vendedor	30

O comando DISTINCT usa-se apenas uma vez por comando SELECT e a sua localização, é sempre depois da cláusula SELECT e antes da lista de colunas. Aplica-se à lista de colunas selecionadas.

c) Cláusula ORDER BY

Uma vez que a ordem das linhas na tabela é irrelevante e uma vez que a ordem do resultado das *queries* depende fortemente do algoritmo de procura usando internamente pelo SGBD, e uma vez que esse algoritmo é invisível ao usuário, a única forma de obrigar a uma ordem específica dos resultados é através do uso da cláusula ORDER BY.

Exemplo 1:

```
SELECT num_departamento, nome, funcao, salario
FROM empregados
ORDER BY num_departamento, salario DESC, nome;
```

ORDER BY, a ser usada terá que aparecer depois das cláusulas, SELECT, FROM e WHERE e aplica-se a uma lista de colunas. Os resultados são ordenados primeiro pela primeira coluna da lista referida, e em caso de empate pela segunda coluna referida e assim sucessivamente. A ordenação é ASCendente por omissão a não ser que se especifique que é DESCendente através da palavra DESC à frente da coluna respectiva.

Pode ainda especificar-se, na cláusula ORDER BY, o alias de uma coluna, ou o número de uma coluna. O número da coluna depende da ordem das colunas que aparecem na cláusula SELECT. A primeira coluna é a número 1 e não a número 0.

Exemplo 2: Ordena primeiro ascendentemente por funcao (2ª coluna), depois descendentemente por "Ganho Anual" e finalmente ascendentemente por nome.

```
SELECT DISTINCT nome, funcao, salario*14 + IFNULL(premios,0) 'Ganho Anual'
FROM empregados
ORDER BY 2 ASC, 'Ganho Anual' DESC, nome ASC;
```

nome	funcao	Ganho Anual
Catarina Silva	Analista	60900.00
Maria Dias	Analista	79100.00
Antonio Silva	Continuo	99120.00
Olga Costa	Continuo	95620.00
Rita Pereira	Continuo	91140.00
Tome Ribeiro	Continuo	79730.00
Augusto Reis	Encarregado	63126.00
Duarte Guedes	Encarregado	53312.00
Silvia Teles	Encarregado	39116.00
Ana Rodrigues	Vendedor	39108.00
Joana Mendes	Vendedor	26014.00
Manuel Madeira	Vendedor	22092.00
Nelson Neves	Vendedor	39558.00

d) Cláusula WHERE

A cláusula WHERE permite restringir linhas através de uma condição. Apenas as linhas que satisfaçam a condição são devolvidas.

Podem usar-se no WHERE condições sobre uma ou mais colunas de uma ou mais tabelas ou vistas* desde que as tabelas ou vistas apareçam na cláusula FROM.

Pode comparar-se valores de colunas, expressões aritméticas e constantes. Podem aparecer na cláusula WHERE nomes de colunas que não apareçam na cláusula SELECT. Não se podem usar pseudônimos de colunas.

Para além de fazer restrições simples sobre uma tabela, o uso mais comum do WHERE é o de permitir relacionar colunas de várias tabelas ou vistas.

A cláusula WHERE tem 3 elementos:

- nome de uma coluna
- um operador de comparação
- um nome de uma coluna, uma constante ou uma lista de valores

A cláusula WHERE, se usada, terá de aparecer depois da FROM. Os operadores lógicos permitidos são os seguintes:

=	igual
<	menor que
<=	menor ou igual
>	maior
>=	maior ou igual
!=	diferente
<>	diferente

Existem ainda os seguintes operadores SQL:

BETWEEN…AND…	entre dois valores (inclusive)
IN (lista)	corresponde a qualquer elemento da lista
LIKE	cadeia de caracteres

IS [NOT] NULL	se (não) é um valor nulo
NOT (condição)	a negação de uma condição

Para testar mais do que uma condição pode fazer-se uso de ANDs e ORs. Note que a prioridade dos ANDs é maior. Pode-se usar parêntesis para alterar a ordem de execução das comparações.

Exemplos:

- Uso de "="

```
SELECT * FROM empregados
WHERE num_departamento = 10;
```

Devolve todos os empregados do departamento 10.

- Uso de AND e NOT (e diferente)

```
SELECT * FROM empregados
WHERE num_departamento = 10 AND NOT funcao = 'Encarregado';
```

A última linha do SELECT poderia ter sido escrita de qualquer uma das seguintes maneiras:

funcao != 'Encarregado';

ou

funcao <> 'Encarregado';

- Uso de OR

```
SELECT * FROM empregados
WHERE num_departamento = 10 AND (NOT funcao = 'Encarregado'
OR num_departamento = 20);
```

- Uso de BETWEEN … AND …

```
SELECT nome, salario
FROM empregados
WHERE salario BETWEEN 10000 AND 20000;
```

- Uso de IN (lista)

```
SELECT nome, numero
FROM empregados
WHERE nome IN ('Jorge Sampaio', 'Augusto Reis', 'Duarte Guedes');
```

- Uso de LIKE <cadeia_de_caracteres>

```
SELECT nome, numero
FROM empregados
WHERE nome LIKE 'A%' OR nome LIKE '_O%';
```

O caracter especial '%' representa qualquer cadeia de caracteres. O caracter especial '_' representa um caracter com qualquer valor. Assim, o comando anterior devolve as linhas dos empregados com cujo nome começa por 'A' ou em que a segunda letra do nome é um 'O'. Só se usa o operador LIKE quando se pretende fazer comparações com caracteres e strings que usem os caracteres especiais de comparação '%' ou '_'.

- Uso de IS NULL

```
SELECT nome 'Sem Premios'
FROM empregados
WHERE premios IS NULL;
```

Sem Premios
Antonio Silva
Augusto Reis
Duarte Guedes
Silvia Teles
Maria Dias
Rita Pereira
Tome Ribeiro
Catarina Silva
Olga Costa

- Uso de IS NOT NULL

```
SELECT nome 'Com Premios'
FROM empregados
WHERE premios IS NOT NULL;
```

Com Premios
Joana Mendes
Nelson Neves
Ana Rodrigues
Manuel Madeira

Pode-se também ser usado NOT IS NULL em vez de IS NOT NULL.

NOTA: Note que não se pode usar premios = NULL nem premios <> NULL. Como NULL não é um número, não pode ser comparado com o valor de premios. O resultado de qualquer das duas condições anteriores é sempre falso independentemente do valor de premios.
Exemplo 1:

```
SELECT nome 'Sem Premios'
FROM empregados
WHERE premios = NULL;
```

Exemplo 2:

```
SELECT nome 'Com Premios'
FROM empregados
WHERE premios <> NULL;
```

4.3. Modificação de Dados (UPDATE)

A instrução que permite modificar dados é a UPDATE. A sintaxe do comando UPDATE é a seguinte:

```
UPDATE (tabela|vista)
SET col1 = val1, col2 = val2, …,
(col101, col102, …) = (subconsulta)
[WHERE (condição)];
```

A cláusula UPDATE determina sobre que tabela ou vista se vão executar as alterações.
A cláusula SET indica pares de colunas e valores a atribuir a colunas. Os pares estão separados por vírgulas. Além de ou invés dos pares de colunas pode ainda incluir-se atribuições através de subconsultas. Nesse caso, as colunas a terem valores atribuídos deverão aparecer entre parêntesis e separadas por vírgulas.
A cláusula WHERE indica sobre que registros da tabela ou vista serão executadas as alterações. A condição pode ser também uma subconsulta.

Exemplo 1 – Aumenta o salário de todos os empregados em 10%:

```
UPDATE empregados
SET salario = salario * 1.1;
```

Exemplo 2 – Aumenta o salário de todos os empregados do departamento 20 em 10%:

```
UPDATE empregados
SET salario = salario * 1.1
WHERE num_departamento = 20;
```

Exemplo 3 – Muda o salário e a função do 'Jorge Sampaio':
```
UPDATE empregados
SET salario = salario * 2, funcao = 'CHEFAO'
WHERE nome = 'Jorge Sampaio';
```

4.4. Remoção de Dados (DELETE)

Usa-se o comando DELETE para remover registros das tabelas ou vistas. O comando permite apagar apenas registros inteiros. Não é possível apagar um campo com o DELETE. No máximo poder-se-ia colocar o valor NULL nesse campo através de um comando UPDATE. A sintaxe do comando DELETE é a seguinte:

```
DELETE FROM tabela|vista
WHERE condições
```

Como de costume, as condições na cláusula WHERE podem incluir subconsultas e as vistas na cláusula FROM podem ser definidas através de subconsultas.

Exemplo 1 – Apagamento de todos os registros da tabela empregado.
```
DELETE FROM empregados;
```

Exemplo 2 – Apagamento dos registros da tabela empregado que correspondem aos empregados com salários menores que 2000.
```
DELETE FROM empregados
WHERE salario < 2000;
```

Exemplo 3 – Apagamento dos registros da tabela empregado que correspondem aos empregados com salários menores que 2000 e que pertençam ao departamento 20.
```
DELETE FROM empregados
WHERE salario < 2000 AND num_departamento = 20;
```

REFERÊNCIAS

- BIZARRO, Pedro. *Apostila Introdução à Base de Dados Oracle*, Universidade de Coimbra, 2000.
- MANZANO, José Augusto N. G. MySQL 5.5 Interativo: guia essencial de orientação e desenvolvimento. 1. ed. - são Paulo: Érica, 2011.
- OLIVEIRA, Celso Henrique Poderoso. *SQL – Curso Prático*. São Paulo: Novatec, 2002.
- SILBERSCHATZ, Abraham, KORTH, Henry F., SUDARSHAN, S. *Sistema de banco de dados*. 5ª ed. Rio de Janeiro: Elsevier, 2006.