

ORACLE

SQL – PL/SQL

Paulo Henrique Santos da Silva

2021

INTRODUÇÃO

Este artigo visa ser uma introdução ao aprendizado de SQL no banco de dados Oracle, PL/SQL e administração de bancos de dados Oracle. Aprendizado este, obtido nas aulas sobre o Oracle na [Alura](#) ministradas pelo instrutor [Victorino Vila](#), e conteúdos da documentação [Oracle](#) e do site [Oracle Tutorial](#). Ao final do artigo, nas referências bibliográficas, listarei alguns livros para se aprofundar no assunto, incluindo obras que abordam conceitos de banco de dados em geral.

O primeiro capítulo cobre o básico e essencial do SQL, como consulta de dados, ordenação e agrupamento de dados, filtros, *joins*, subconsultas, *views* entre outros comandos que fazem parte da DML (*Data Manipulation Language*). No segundo capítulo é abordado consultas mais inteligentes no Oracle com a PL/SQL, permeando assuntos como *procedures*, *functions*, *exceptions* e *packages*. E por fim, o terceiro capítulo é referente a administração de banco de dados Oracle, no qual exponho algumas atividades de um DBA Oracle, como criação e gerenciamento do banco, segurança e otimização do banco, análise do ambiente e otimização de consultas.

1 - SQL

A *Structured Query Language*, conhecida como SQL, surgiu na década de 1970 como uma linguagem projetada para gerenciar os dados estruturados e mantidos nos sistemas de administração de banco de dados (*Data Base Management System* - DBMS), tornando-se um padrão ISO em 1987.

Os comandos SQL são divididos em *Data Definition Language* (DDL), *Data Query Language* (DQL), *Data Manipulation Language* (DML) e *Data Control Language* (DCL). A maioria das operações que envolvem esses comandos SQL serão abordados no artigo. Iniciando pela DQL e DML, que permitem a fazer a consulta e manipulação de dados nas tabelas, com os principais comandos sendo o SELECT, INSERT, UPDATE, DELETE e MERGE.

1.1 - Consultando dados

Nesta seção será abordado o comando SELECT para a consulta de dados na base de dados Oracle. O comando SELECT, sendo uma especificação do SQL, tende ter a mesma sintaxe para todos os bancos de dados relacionais, com pequenas variações em cada implementação, como a estudada neste artigo, sendo voltada para a base de dados Oracle.

Para construir a consulta com o comando SELECT sempre pergunte a si mesmo “O que eu quero buscar?” e “De onde eu quero buscar?”. Com isso em mente, podemos começar a definir a sintaxe básica do comando, que é SELECT o que? FROM de onde? E sendo mais específico temos a seguinte sintaxe:

```
SELECT coluna_1, coluna_2... FROM nome_da_tabela;
```

Digamos que nossa base de dados possua uma tabela “Funcionários”, e desta tabela queremos saber o nome, sobrenome e data de nascimento de todos os funcionários. Para isso, podemos definir o seguinte comando com SELECT:

```
SELECT nome, sobrenome, dta_nascimento FROM funcionarios;
```

Também é possível retornar todos os dados da tabela (não recomendável) usando o operador “*” da seguinte forma:

```
SELECT * FROM funcionarios;
```

A consulta (com o primeiro exemplo) retornará o seguinte conjunto de dados:

NOME	SOBRENOME	DTA_NASCIMENTO
Letícia	Matos Cardoso	16/12/1993
Ana	Laura Lemes	28/08/1998
Alexandre	de Moraes Vasconcelos	05/02/1992
...

Podemos fazer a pergunta de “Como?” esses dados serão retornados ao usar outros comandos SQL, mas por enquanto observe que foi definido o que queríamos “nome, sobrenome e data de nascimento” e de onde queríamos “tabela Funcionários”.

As consultas com o comando SELECT podem ser mais complexas, sendo apresentado nesses exemplos a sua forma básica. Em seguida veremos outras cláusulas que podem enriquecer o comando SELECT e tornar as consultas mais sofisticadas e elegantes.

1.2 Consultando e filtrando dados

Agora veremos como filtrar os dados das consultas usando as cláusulas **WHERE**, **DISTINCT**, **AND**, **OR**, **IN**, **LIKE**, **BETWEEN**, **FETCH** e **IS NULL**.

Quando fazemos a consulta dos dados com o comando SELECT, é retornado um conjunto de linhas para as colunas indicadas. Com o WHERE podemos especificar condições para os dados que devem ser retornados da consulta, ou seja, podemos dizer “Como?” esse conjunto de dados deve ser retornado.

Então comecemos a fazer novamente as perguntas, “O que deve ser retornado?”, “De onde?” e agora “Como devem ser retornados?”. Aplicando essas perguntas a sintaxe, teremos algo como SELECT o que? FROM de onde? WHERE como? E sendo mais específico:

```
SELECT coluna_1, coluna_2  
FROM nome_da_tabela
```

WHERE condicao;

Nos exemplos anteriores usamos a tabela de funcionários, mas agora queremos consultar os dados de alguns produtos. Primeiro vamos definir “o que?” e “de onde?” vamos consultar, depois dizemos “como?” será feita a consulta, filtrando as linhas que serão retornadas. Queremos saber o nome, o preço e a quantidade (o que?) de produtos (de onde?) que estão com menos de 300 unidades em estoque (como?). Agora podemos construir a consulta da seguinte maneira:

```
SELECT nome, preco, quantidade
FROM produtos
WHERE quantidade < 300;
```

Essa consulta retorna todos os produtos que tenham menos de 300 unidades disponíveis em estoque, como exibido na Tabela 2:

NOME	PRECO	QUANTIDADE
Resma de Papel A4	14.99	128
Pincel Redondo	9.70	287
Giz de Cera	25.50	96
...

Observe que após o WHERE foi usado um dos operadores de desigualdade (!=, < >) junto a coluna de “quantidade” para definir o filtro da consulta, indicando que deveria ser retornado apenas os produtos cuja quantidade é menor < que 300. Existem vários outros operadores, como os de comparação que podem compor a consulta, dizendo como as linhas devem ser retornadas. Na Tabela 3 é exibido os operadores disponíveis para o banco Oracle.

OPERADOR	DESCRIÇÃO
=	Igualdade
!=, < >	Desigualdade
>	Maior que
<	Menor que
>=	Maior ou igual
<=	Menor ou igual

IN	Igual a qualquer valor da lista de valores especificados
ANY/SOME/ALL	Compara com o valor de uma lista de valores ou de uma subconsulta
NOT IN	Que não seja igual a qualquer valor da lista de valores especificados

O NOT pode ser usado com outros operadores como negação, entretanto veremos os demais operadores depois de entendermos como funciona cada uma das cláusulas.

O WHERE pode ser usado em conjunto com outras cláusulas, como o AND e OR que são operadores lógicos, sendo conhecidos em algumas linguagens de programação como && (AND) e || (OR).

Apesar de ser um exemplo bem simples, veremos como usar o AND na construção da *query*, para isso digamos que alguém pediu para conferir os produtos com estoque entre 300 e 600 unidades. Uma das maneiras de se fazer essa consulta é usando o WHERE com o AND:

```
SELECT nome, preco, quantidade
FROM produtos
WHERE quantidade < 300 AND preco < 15.00;
```

Nessa consulta primeiro é verificado todos os produtos que tenham a quantidade em estoque menor que 300 unidades (como visto anteriormente), depois é selecionado aqueles que tenham o seu preço menor que R\$15,00 validando a condição imposta e retornando o seguinte resultado:

NOME	PRECO	QUANTIDADE
Resma de Papel A4	14.99	128
Pincel Redondo	9.70	287
...

Para o OR muda que não é necessário ambas as condições serem verdadeiras, apenas uma já valida a consulta. Então se nessa mesma consulta o AND for trocado pelo OR, “todos” os dados avaliados, podem ser retornados. E o retorno será de todos os produtos que tenham menos de 300 unidades (mesmo que custe mais que R\$15,00) em estoque ou o preço seja menor que o especificado (mesmo que tenha mais de 300 unidades em estoque), pois como já

referido ao menos uma das condições deve ser verdadeira. E essa substituição resultaria no seguinte retorno:

NOME	PRECO	QUANTIDADE
Resma de Papel A4	14.99	128
Pincel Redondo	9.70	287
Giz de Cera	25.50	96
Lápis N2	2.75	421
Massa de Modelar Atóxica	18.90	152
...

A partir do OR podemos fazer uma associação com o IN em sua sintaxe, que diz basicamente o seguinte na consulta: retorne todas as linhas que tenham algum desses valores. E como isso se relaciona com o OR? Quando usamos o OR dizemos na consulta: eu quero que retorne isso ou aquilo, pode ser qualquer um do que estou especificando. Consegue notar a semelhança? A principal diferença é vista no IN, que aceita uma “lista” de valores. A sintaxe é a seguinte:

```
SELECT coluna_1, coluna_2...  
FROM nome_da_tabela  
WHERE coluna_1 IN ('valor1', 'valor2', 'valor3'...);
```

Agora foi informado o nome de alguns produtos e nos pediram para verificar o preço de cada um. Podemos fazer isso usando o IN, com a seguinte consulta:

```
SELECT nome, preco  
FROM produtos  
WHERE nome IN ('Giz de Cera', 'Mapa Mundi para colorir',  
              'Cola');
```

Sendo o retorno da consulta o seguinte resultado:

NOME	PRECO
Giz de Cera	25.50
Mapa-Múndi para colorir	56.80
Cola	3.70

E se quiséssemos fazer consulta usando, por exemplo, uma palavra que está na descrição de um produto ou as iniciais do sobrenome de um cliente como filtro? Para isso usamos o LIKE.

O LIKE permite fazer uma consulta com base em um padrão de caracteres como filtro. Pode ser as letras no início ou final de uma palavra e até mesmo a palavra inteira em um texto. Para filtrar palavras que coincidam com alguns caracteres que estejam no fim é necessário usar o operador “%” seguido pelos caracteres de filtro, o contrário pode ser feito para consultar palavras que comecem com os caracteres de filtro, ou seja, primeiro informe os caracteres e em seguida use o operador de percentagem.

Então, para consultar pelo início, use o operador no final, e para consultar pelo final use o operador no início. E para fazer uma consulta *insensitive case* (que ignore se está em maiúsculo ou minúsculo) pode usar as funções LOWER([nome_da_coluna]) e UPPER([nome_da_coluna]).

A seguir é apresentado a sintaxe de uma consulta com o LIKE.

```
SELECT coluna_1, coluna_2...  
FROM nome_da_tabela  
WHERE UPPER(nome_da_coluna) LIKE '%CARACTERES%';
```

Agora queremos saber o nome dos clientes que tem ‘Silva’ como sobrenome. Nessa consulta podemos fazer da seguinte maneira:

```
SELECT nome FROM clientes WHERE UPPER(nome) LIKE  
'%SILVA%';
```

E como resultado da consulta é retornado os seguintes clientes:

NOME
Abel Silva Castro
Valdeci da Silva

Observe que na construção dessa *query* foi usado a função UPPER() e em seguida o sobrenome também foi colocado em upper case, isso é necessário pois como citado anteriormente essa função coloca todos os caracteres da coluna passada como parâmetro em

upper case, logo a “*string*” a ser buscada também deve estar em upper case para que possa ser possível realizar a comparação, e o mesmo ocorre ao usar a função LOWER().

No LIKE foi usado % no início e no fim da palavra, isso basicamente diz: “Busque todos os nomes que tenham ‘Silva’, não importa se é no começo, meio ou fim”. Isso não acontece se usar o símbolo de percentagem no início ou fim. Caso use no início, dirá na sua *query*: “Busque apenas os nomes que tenham ‘Silva’ no final”.

```
SELECT nome FROM clientes WHERE LOWER(nome) LIKE  
'%silva';
```

NOME
Valdeci da Silva

Agora se trocarmos a posição do operador “%”, movendo-o para o final, estamos dizendo que o retorno da consulta deve ser o nome das pessoas que começa com ‘Silva’, mas como é um sobrenome, neste caso, não irá retornar nenhuma linha.