Extensões do SQL

CASE...WHEN...THEN…END
Filtros em Strings - LIKE
Consultas com Múltiplas Tabelas
INNER JOIN
LEFT/RIGHT JOIN (OUTER)
FULL JOIN (OUTER)
Join com Chaves Compostas

1 CASE...WHEN...THEN…END

Esta estrutura é utilizada para dados condicionais.

Ex.: Mostrar nomes em português para os nomes originais em inglês.

```
select tipocliente,
	case tipocliente
		when 1 then 'fisico'
		when 2 then 'juridico'
		else 'outros'
	end as tipo
from cliente
```

Note que não existe vírgula entre os operadores **when,** pois eles não são novas colunas e sim continuação da condição.

2. Filtros em Strings - LIKE

Para filtrar strings existe instrução própria, uma vez que números têm um valor absoluto e textos não.

A instrução **LIKE** tem como diferencial o uso dos seguintes coringas:

1) "**%**" para qualquer coisa na substituição (similar ao "*" do DOS);
2) "_" para indicar a posição de um caractere específico, sem indicá-lo;
3) "**[**" e "**]**" para definir um intervalo de valores (bases não Oracle).

Exemplos:

```
select *
from cliente
where nome = 'anton'          --nome anton é o unico que será encontrado

select *
from cliente
where nome like '[b-e]%'      --nomes iniciando entre b e e

select *
from cliente
where nome like '[bmf]%'      --nomes cujas iniciais sejam b, m ou f
```

2. Filtros em Strings – LIKE (cont.)

```
select *
from cliente
where nome like 'l[ai]%'          --iniciando em l e a segunda letra sendo a ou i

select *
from cliente
where nome like '%eti%'           --nomes contendo eti em qualquer lugar

select *
from cliente
where nome like '%eti%ass%'       --contendo eti e ass

select *
from cliente
where nome like 'Brun_'           --nomes como Bruno e Bruna retornariam
```

3. Consultas com Múltiplas Tabelas

Através da normalização os dados são separados em diferentes tabelas (relações). Para realizar o processo de inferência sobre um modelo de dados, em algumas situações, torna-se necessário envolver mais de uma relação para a busca da informação desejada.

O processo de "juntar" linhas (tuplas) de várias tabelas (relações), usando uma ou mais colunas como guia é chamado de **join**.

Existem quatro tipos diferentes de join no novo padrão:

1) **INNER JOIN**;
2) **LEFT JOIN**;
3) **RIGHT JOIN**;
4) **FULL JOIN**.

Nem todos os bancos de dados relacionais dão suporte para o uso desta sintaxe. Será necessário verificar junto ao fornecedor do SGBD.

3. Consultas com Múltiplas Tabelas

```
SQL> select * from t1;

        ID
---------
         1
         2
         3
         4

SQL> select * from t2;

        ID
---------
         3
         4
         5
         6
```

4. INNER JOIN

Este é o utilizado com mais frequência e sua característica é trazer as linhas que contenham correspondência nas tabelas envolvidas.

Deve-se lembrar de que a utilização do **INNER JOIN** pode não trazer todos os registros da tabela original, principalmente se nas outras tabelas relacionadas não existir linhas correspondentes.

```
-- INNER JOIN (REGULAR JOIN)
SQL> select t1.id, t2.id from t1
SQL> INNER JOIN t2 ON (t1.id = t2.id)
SQL> /

      ID         ID
--------- ---------
       3          3
       4          4
```

```
select f.nome, p.nome, pf.estoque
from produtoxfornecedor pf
inner join fornecedor f on pf.codfornecedor = f.codfornecedor
inner join produto p on pf.codproduto = p.codproduto
```

5. LEFT/RIGHT JOIN (OUTER)

As instruções LEFT JOIN e RIGHT JOIN são utilizadas para resolver a não correspondência de dados entre diferentes tabelas.

```
-- LEFT JOIN (SAME AS LEFT OUTER JOIN)
SQL> select t1.id, t2.id from t1
SQL> LEFT JOIN t2 ON (t1.id = t2.id)
SQL> /

        ID         ID
---------- ---------
         3          3
         4          4
         1
         2
```

-- Trazer todos os fornecedores, independente de existir correspondência nas tabelas.

```
select f.nome, p.nome, pf.estoque from fornecedor f
left join produtoxfornecedor pf on pf.codfornecedor = f.codfornecedor
left join produto p on pf.codproduto = p.codproduto
```

5. LEFT/RIGHT JOIN (OUTER - cont.)

Podemos entender o LEFT e o RIGHT designando a tabela à esquerda ou à direita da principal, não importando o fato de que a outra tabela possua ou não correspondências.

```
-- RIGHT JOIN (SAME AS RIGHT OUTER JOIN)
SQL> select t1.id, t2.id from t1
SQL> RIGHT JOIN t2 ON (t1.id = t2.id)
SQL> /

        ID         ID
---------- ----------
         3          3
         4          4
                    6
                    5
```

-- Trazer todos os produtos, independente de existir correspondência nas tabelas.

select f.nome, p.nome, pf.estoque from fornecedor f
right join produtoxfornecedor pf on pf.codfornecedor = f.codfornecedor
right join produto p on pf.codproduto = p.codproduto

6. FULL JOIN (OUTER)

Ainda existe uma situação não coberta pelo uso de LEFT/RIGHT JOIN.

```
-- FULL JOIN (SAME AS FULL OUTER JOIN)
SQL> select t1.id, t2.id from t1
SQL> FULL JOIN t2 ON (t1.id = t2.id)
SQL> /

        ID         ID
---------- ---------
         3          3
         4          4
         1
         2
                    6
                    5
```

6. FULL JOIN (OUTER – cont.)

Com o LEFT JOIN conseguimos saber os fornecedores sem produtos.

Com o RIGHT JOIN conseguimos saber os produtos sem fornecedores.

Para conseguir as duas situações em uma única consulta utiliza-se o FULL JOIN.

```
select f.nome, p.nome, pf.estoque
from fornecedor f
full join produtoxfornecedor pf
on pf.codfornecedor = f.codfornecedor
full join produto p
on pf.codproduto = p.codproduto
```

7. Join com Chaves Compostas

Em alguns casos o relacionamento entre tabelas não é feito através de uma única coluna, mas por várias.

Neste caso podemos utilizar o JOIN com o operador AND para fazer a junção composta.

Continua valendo a utilização do AND também para aplicar filtros (restrições) à consulta.

```
select d.codcurso, d.nome, d.area, t.nome, t.datainicio, t.datafim
from disciplina d
inner join turma t
on   d.codcurso = t.codcurso and
     d.coddisciplina = t.coddisciplina and
     t.datainicio between ’01-jan-2007’ and ’01-nov-2007’
```