Parte
4

PESQUISA BÁSICA EM TABELAS

Agora que já sabemos criar, incluir, alterar e excluir informações nas tabelas, veja como podemos extrair informações do banco de dados. Para tanto o comando utilizado é o *SELECT*. Junto dele há uma imensa possibilidade de comandos como veremos a seguir.

4.1 SELECT

É o comando utilizado para realizar buscas/pesquisas no banco de dados. Atrás dele há uma extensão de possibilidades que vão desde a simples extração do conteúdo de todas as linhas e colunas de uma tabela até a união de diversas tabelas, cálculos, agrupamentos, ordenações e filtragem de linhas e colunas.

Sintaxe:

```
SELECT [DISTINCT | ALL] {* | Coluna [, coluna, ...]}
FROM tabela
```

Onde:

DISTINCT: Não mostra eventuais valores repetidos;
ALL: Mostra todos os valores (Propriedade *Default* do comando *select*);
*: Mostra todas as colunas da tabela;

NOTE BEM:

Para execução dos exemplos utilizaremos estrutura criada na parte 3 – DML.

Exemplos:

O exemplo mais simples de uma consulta é extrair todas as informações de uma tabela:

```
SELECT * FROM cd;
```

O exemplo a seguir demonstra como filtrar apenas algumas colunas da tabela:

```
SELECT cd_id, cd_nome FROM cd;
SELECT grav_id, grav_nome, grav_tel FROM gravadora;
```

4.1.1 Ordenando o resultado

Em muitas situações a ordem mostrada nem sempre é a que esperamos. Para que os dados sejam mostrados da maneira que melhor nos atende usa-se a cláusula ***ORDER BY*** seguida pela coluna que desejamos que seja ordenada:

Exemplo:

```
SELECT aut_id, aut_nome FROM autor
ORDER BY aut_nome;

SELECT cd_id, cd_nome FROM cd
ORDER BY cd_id;
```

Se forem especificadas mais de uma coluna a serem ordenadas, o gerenciador primeiro ordenará pela primeira coluna e em seguida pelas demais:

Exemplo:

```
SELECT grav_id, cd_nome FROM cd
ORDER BY grav_id, cd_nome;
```

4.1.2. Filtrando linhas

Para filtrar linhas em uma pesquisa, utilizamos a cláusula ***WHERE***. Assim, é definida uma expressão lógica (condição) que será validada e mostrará apenas as linhas que atenderem ao critério estabelecido.

Sintaxe:

```
SELECT [DISTINCT | ALL] {* | Coluna [, coluna, ...]}
FROM tabela
WHERE condição
```

Para um resultado satisfatório devemos saber exatamente como construir condições que satisfaçam às nossas necessidades de busca para atingir nossos objetivos. Sempre que a condição especificada for verdadeira o resultado será mostrado. Para tanto é necessário utilizar-se de alguns operadores como veremos a seguir.

4.1.2.1 OPERADORES RELACIONAIS

Estes operadores devem ser usados na definição das condições. Podemos testar igualdade, diferença, maior, menor, maior ou igual, menor ou igual. Os operadores devem ser colocados entre os argumentos que estão sendo comparados.

OPERADOR	SIGNIFICADO	EXEMPLO
=	Igual	aut_id = 2
<	Menos que	cd_preco < 10
<=	Menor ou igual a	cd_preco <= 10

>	Maior que	cd_preco > 10
>=	Maior ou igual a	cd_preco <= 10
!= ou <>	Diferente	aut_id != 2 ou aut_id <> 2

Exemplo:
```
SELECT cd_nome, cd_preco FROM cd
WHERE cd_preco > 12;
```

NOTE BEM:
- ✓ Da mesma forma que podemos comparar uma coluna com um valor, podemos comparar com outra coluna.
- ✓ Sempre quando fazemos esse tipo de comparação, devemos obedecer ao tipo de dado que estamos comparando.

4.1.2.2 OPERADORES LÓGICOS

Muitas vezes, apenas uma condição não é suficiente para determinarmos o critério de busca. Sempre que isso ocorrer, podemos utilizar operadores lógicos.

OPERADOR	SIGNIFICADO	EXEMPLO
AND	e	Condição-1 AND Condição-2
OR	ou	Condição-1 OR Condição-2
NOT ou !	não/negação	NOT Condição

AND

Indica que as duas condições devem ser verdadeiras para que seja mostrada a linha.

Exemplo:
```
SELECT cd_nome, cd_preco, grav_id FROM cd
WHERE cd_preco > 10 AND grav_id = 2;
```

OR

Utilizamos o operador OR sempre que quisermos que o resultado final seja verdadeiro.

Exemplo:
```
SELECT cd_nome, cd_preco, grav_id FROM cd
WHERE cd_preco > 11 OR grav_id = 2
```

NOTE BEM
- ✓ Não há limitação no uso e na combinação de condições usando OR e AND.
- ✓ É conveniente utilizar parênteses para determinar o que se quer comparar.

Exemplo:
```
SELECT cd_nome, grav_id, cd_preco FROM cd
WHERE (grav_id = 2 OR grav_id = 3) AND (cd_preco >= 17.50)
```

***NOT* ou !**

É utilizado para inverter o resultado de uma expressão lógica, negando o resultado da condição. Caso a condição seja verdadeira, será retornado falso e vice-versa.

Exemplo:
```
SELECT cd_nome, cd_preco FROM cd
WHERE NOT (cd_preco > 15);
```

4.1.2.3 OPERADORES ESPECIAIS

Existem alguns operadores que são utilizados para determinar melhor as linhas que queremos filtrar. São eles: **IS *NULL*, *IS NOT NULL*, *BETWEEN*, *LIKE*** e ***IN*.**

IS NULL

Sabemos que nem todas as colunas têm valores inicializados. Logo esse comando é utilizado para saber os campos que não foram inicializados:

Exemplo:
```
SELECT * FROM gravadora
WHERE grav_tel IS NULL
```

IS NOT NULL

Compara a negação do comando anterior. Somente aqueles que tiverem conteúdo serão mostrados:

Exemplo:
```
SELECT * FROM gravadora
WHERE grav_tel IS NOT NULL
```

BETWEEN

Esse operador serve para determinar um intervalo de busca. Quando desejarmos um intervalo entre números, datas, etc, utilizaremos o *BETWEEN* para simplificar a forma de escrevermos o comando. Normalmente é utilizado em conjunto com o ***AND***.

Exemplo:
```
SELECT cd_nome, cd_dt_lancamento FROM cd
WHERE cd.cd_dt_lancamento BETWEEN '1979-01-01' AND '2000-12-31';
```

LIKE

Com esse operador podemos comparar cadeias de caracteres utilizando padrões de comparação para um ou mais caracteres. O caractere percentual (**%**) substitui zero, um ou mais caracteres e sublinha (_) substitui um caractere.

EXPRESSÃO	APLICAÇÃO
LIKE 'A%'	Todas as palavras que iniciem com a letra A
LIKE '%A'	Todas as palavras que terminem com a letra A
LIKE '%A%'	Todas as palavras que tenham a letra A em qualquer posição
LIKE 'A_'	String de dois caracteres que tenha a primeira letra A
LIKE '_A'	String de dois caracteres que tenha o último caractere letra A
LIKE '_A_'	String de três caracteres cuja segunda letra seja A
LIKE '%A_'	Todas as palavras que tenham a letra A na penúltima posição
LIKE '_A%'	Todas as palavras que tenha a letra A na segunda posição

Exemplos:

```
SELECT * FROM autor
WHERE aut_nome LIKE 'R%';

SELECT * FROM gravadora
WHERE grav_nome LIKE '_o%';
```

Um problema que pode surgir quando queremos fazer buscas utilizando os caracteres de substituição é tê-los na cadeia de caracteres que está sendo pesquisada. Neste caso devemos usar um caractere especial denominado ***ESCAPE***.

Exemplo:

```
SELECT * FROM cd
WHERE cd_nome LIKE '%\_%'ESCAPE '\';
```

IN

Permite comparar o valor de uma coluna com um conjunto de valores. Utilizamos para substituir uma série de comparações seguidas da cláusula OR.

Exemplo:

```
SELECT * FROM autor
WHERE aut_id IN (1, 3);
```

Sua maior utilização é em *subquerys* (*será visto posteriormente*).

4.2. EXERCÍCIOS

a) Liste todos os campos e linhas da tabela GRAVADORA;

b) Liste todas as linhas dos campos CD_ID, CD_NOME, CD_PRECO da tabela CD;

c) Liste todas as linhas dos campos AUT_ID, AUT_NOME da tabela AUTOR em ordem alfabética;

d) Repita o comando anterior em ordem alfabética decrescente;

e) Liste todos os CDs da gravadora 3;

f) Liste as colunas CD_NOME, CD_PRECO dos CD's cujos preços de venda sejam inferiores a 20,00 e sejam da GRAVADORA 3;

g) Liste as colunas da tabela gravadora cujo GRAV_CONTATO seja nulo;

h) Repita o comando anterior desta vez listando GRAV_CONTADO não nulo;

i) Liste os CD's cujos CD_PRECO esteja entre 15,00 e 30,00;

j) Liste todos os CD's cuja CD_DT_LANCAMENTO seja posterior ao ano 01/01/2000;

k) Liste as MUSICAS cujo nome comece com A da tabela MUSICA;

l) Liste os CD's cuja segunda letra do CD_NOME seja a letra E;

m) Liste os CD's que possuam a letra O em qualquer posição do CD_NOME;

n) Liste os CD's que possuam CD_PRECO inferior a 30,00 em ordem decrescente de CD_DT_LANCAMENTO;

o) Liste as músicas cuja MUS_ID seja 1, 3, 5;