Banco de Dados - Ciclo 3 - Webaula 20

FURB

WEBAULA 20

- Seleção Básica de Dados
- Referências

QUESTION BANKS

Seleção Básica de Dados

A partir de agora você vai conhecer os recursos do principal comando da SQL – o comando SELECT – e suas características básicas, como as cláusulas, os tipos e os principais operadores utilizados.

> **Dica**
>
> Recomendamos que você, à medida que avançar na disciplina, vá executando os exemplos apresentados. Para tanto, estão disponíveis os comandos de inserção dos dados nas tabelas EMP, DEPT e SALGRADE. Verifique no Ambiente de Aprendizagem o *link* para *download*.

Como já dito, o SELECT é o principal comando da SQL. Estatísticas mostram que mais de 70% das operações em um SGBD relacional correspondem a consultas aos dados. Estas, por sua vez, são realizadas com o uso deste comando. Com este comando você poderá:

- Selecionar dados de uma ou mais tabelas
- Restringir e projetar dados de uma ou mais tabelas.
- Agrupar e selecionar dados de uma ou mais tabelas.
- Gerar diferentes visões sobre os mesmos dados.

Ao longo desta e das próximas seções você irá conhecer gradualmente o comando SELECT. É possível que você encontre em algumas literaturas a informação de que este comando pertence ao subconjunto de comandos DML da SQL, mas isso não é consenso. Nós vamos estudar o comando independentemente dessa classificação, evidenciando sempre a sua importância.

O comando SELECT apresenta seis cláusulas, sendo que duas delas são obrigatórias, ou seja, sempre deverão estar presentes quando você utilizar o comando. Conheça a estrutura do comando no Quadro 46.

Quadro 46 – Estrutura do comando SELECT

```
SELECT

    { *, coluna [apelido], coluna [apelido],... }

FROM

   {tabela [apelido], tabela [apelido],...}

WHERE

   {condição, condição,...}

GROUP BY

   {coluna_agrupamento, coluna_agrupamento,...}

HAVING
```

```
  {condição_função_agrupamento, condição_função_agrupamento,...}

ORDER BY

  {coluna, coluna,...}
```

No quadro acima você pode observar que na estrutura do comando SELECT há um conjunto de palavras-chave, as quais denominamos cláusulas (SELECT, FROM, WHERE, GROUP BY, HAVING e ORDER BY). Na cláusula SELECT você deve informar o que deseja recuperar. A cláusula FROM recebe a informação de qual ou quais objetos (tabelas) devem ser envolvidos na recuperação. A cláusula WHERE é opcional, mas dificilmente não é utilizada. Nela você deve informar os critérios ou condições que os dados a serem recuperados devem atender.

A seguir temos as cláusulas GROUP BY e HAVING, que são menos utilizadas e auxiliam no agrupamento de dados, assunto que iremos abordar mais adiante. Por último e não menos importante, temos a cláusula ORDER BY, que você deve utilizar sempre que necessitar informar um ou mais critérios de ordenamento dos dados recuperados que serão apresentados. As cláusulas SELECT e FROM são obrigatórias, ou seja, toda sentença SELECT deve apresentá-las. Se não quiser ter problemas com o comando, recomendamos sempre seguir esta sequência de cláusulas. Agora vamos conhecer um pouco mais sobre as cláusulas SELECT, FROM, WHERE e ORDER BY. As cláusulas GROUP BY e HAVING você irá conhecer mais adiante.

A primeira cláusula (SELECT) que dá nome ao comando é utilizada para que você informe qual ou quais são as colunas que você deseja recuperar. A(s) coluna(s) deve(m) pertencer à tabela presente da cláusula FROM. Se você deseja selecionar todas as colunas da tabela, poderá utilizar o asterisco (*). Neste caso, as colunas serão exibidas na ordem em que foram definidas na tabela. O Quadro 47 apresenta um exemplo simples de utilização do comando SELECT.

Quadro 47 – Exemplo simples do comando SELECT

```
SELECT  *
```

```
FROM EMP;
```

No exemplo apresentado serão listadas todas as colunas e todas linhas da tabela EMP. Isso ocorre porque utilizamos o asterisco (*) e omitimos as demais cláusulas do comando, o que garante a exibição total da tabela (todas as linhas).

Já a próxima sentença (Quadro 48) especifica as colunas que serão recuperadas/apresentadas. Quando você faz isto, está utilizando um conceito denominado projeção de dados. A **projeção de dados** nada mais é do que indicar (projetar) explicitamente qual(is) será(ão) a(s) coluna(s) a serem exibidas, diferentemente do uso do asterisco (*), que apresenta todas as colunas.

Quadro 48 – Exemplo do comando SELECT com projeção de dados

```
SELECT ENAME, JOB, SAL

FROM EMP;
```

É importante ressaltar que os exemplos acima não restringem os registros recuperados, ou seja, em ambos todos os registros da tabela serão recuperados.

Um recurso usual na seleção de dados é a atribuição de um apelido para a(s) coluna(s). Observe no exemplo apresentado no Quadro 49 como isso pode ser feito.

Quadro 49 – Exemplo do comando SELECT com apelido nas colunas

```
SELECT ENAME as NOME,
                JOB as ATIVIDADE,
```

```
                Sal as SALARIO

FROM EMP;
```

Em alguns SGBDs relacionais, como o ORACLE, por exemplo, não é necessário utilizar a palavra reservada “as” antes do apelido atribuído. Caso se deseje atribuir nomes compostos ao apelido, será necessário utilizar aspas ("apelido composto"). Veja o exemplo apresentado no Quadro 50.

Quadro 50 – Exemplo do comando SELECT com apelido composto nas colunas

```
SELECT ENAME as NOME,
                JOB as ATIVIDADE,
                  Sal as “SALARIO FIXO”

FROM EMP;
```

Dica

Em alguns SGDBs relacionais o apelido (também conhecido como alias) atribuído pode ser utilizado na cláusula ORDER BY, e somente nela. Voltaremos a falar sobre essa cláusula mais adiante.

Aos poucos você vai perceber que raramente um comando SELECT é executado sem a presença da cláusula WHERE. Através da cláusula WHERE é possível restringir os registros que você deseja recuperar, além de estabelecer a relação entre duas ou mais tabelas. Vamos conhecer um exemplo de restrição (condição) feita através da cláusula WHERE no Quadro 51.

Quadro 51 – Exemplo do comando SELECT com cláusula WHERE

```
SELECT ENAME, JPB, SAL
FROM EMP
WHERE
        EMPNO = 7788;
```

No exemplo acima podemos observar que a restrição apresenta a sentença de recuperação de dados, ou seja, recuperar os registros onde o valor da coluna EMPNO seja igual a 7788 (EMPNO = 7788). Para adicionar mais de uma restrição (condição), é necessário utilizar um operador lógico entre elas. O Quadro 52 apresenta um exemplo.

Quadro 52 – Exemplo do comando SELECT com cláusula WHERE e condição composta

```
SELECT ENAME, JPB, SAL
FROM EMP
WHERE
        DEPTNO = 10 AND SAL > 1000;
```

Como apresentado no exemplo (Quadro 52), ao utilizarmos restrição composta, além do operador relacional em cada condição, necessitamos fazer uso de operadores lógicos entre as condições. A seguir vamos conhecer os principais operadores utilizados na cláusula WHERE.

Os operadores mais utilizados na cláusula WHERE são classificados em relacionais (ou de comparação) e lógicos (booleanos). A Figura 47 apresenta o conjunto desses operadores.

Operadores relacionais

Operador	Significado
=	Igual a
>	Maior do que
>=	Maior do que ou igual a
<	Menor do que
<=	Menor do que ou igual a
<>	Diferente de

As condições seguem as seguintes **regras de precedência**:

1ª Operadores Relacionais
2ª Operador NOT
3ª Operador AND
4ª Operador OR

Para ajustar a precedência, caso seja necessário, utilize os parênteses.

Operadores lógicos

Operador	Significado
OR	Uma das condições for verdade
AND	Ambas as condições devem ser verdade
NOT	Negação ou inverso da variável

Figura 47

Dica

Certamente você já deve ter se deparado com os tipos e operadores nas linguagens de programação, não é mesmo? Pois bem, o princípio de funcionamento é o mesmo, assim como as regras de precedência. Fique atento, pois eles serão utilizados frequentemente nas sentenças que você terá que construir.

Um importante operador utilizado para a recuperação de dados alfanuméricos (textuais) é o LIKE. O operador LIKE busca valores alfanuméricos incompletos a partir de um ou mais caracteres. Para tanto, basta utilizar os seguintes caracteres especiais associados ao valor utilizado para comparação:

"%" → corresponde a uma sequência qualquer de zero ou mais caracteres

"_" → corresponde a qualquer caractere

Observe na ilustração a seguir (Figura 48) exemplos de utilização do operador LIKE.

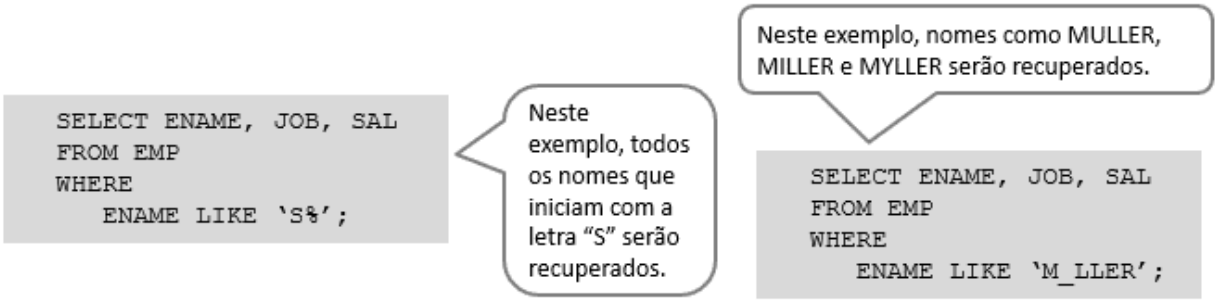

Figura 48

No exemplo apresentado acima temos duas sentenças. Na primeira utilizamos o símbolo do percentual (%) para indicar que a linha onde a coluna nome (ENAME) inicia com o caractere "S" deverá ser recuperada. Isso significa dizer que, ao utilizar o "%" à direita, como no exemplo, todos os demais caracteres são desconsiderados na comparação. Da mesma forma seria caso tivéssemos inserido o "%" à esquerda, ou seja, caso a expressão fosse "%S", seriam recuperadas as linhas cujo nome, necessariamente, terminasse em "S". Temos também a situação em que utilizamos o "%" à esquerda e à direita: "%S%". Neste caso, qualquer nome (ENAME) que apresentar um caractere "S" será recuperado, independentemente da posição da letra "S" no referido nome.

Já a segunda sentença do exemplo faz uso do símbolo do *underline* (_), que faz o papel de coringa, ou seja, substitui qualquer caractere na referida posição.

Dica

Fique atento! Em alguns SGBDs relacionais os dados alfanuméricos são sensitivos, ou seja, diferenciam maiúsculas de minúsculas. Para mais informações, consulte o manual do SGBD relacional escolhido por você ou contate o professor.

Continuando a exploração dos operadores que podemos utilizar na cláusula WHERE, chegamos aos operadores IS e IN. O operador IS é utilizado quando é necessário avaliar se uma coluna apresenta ou não ausência de valor, ou seja, é nula. Observe a Figura 49, onde é apresentado um exemplo de sentença com o operador IS.

Figura 49

O outro operador citado é o IN. O operador IN é utilizado para testar o valor da coluna em uma lista de valores. Na prática é como se escrevêssemos o comando utilizando uma sequência de restrições envolvendo a coluna e os valores listados, conjugados ao operador lógico “OR”. Observe na Figura 50 o que acabamos de apresentar.

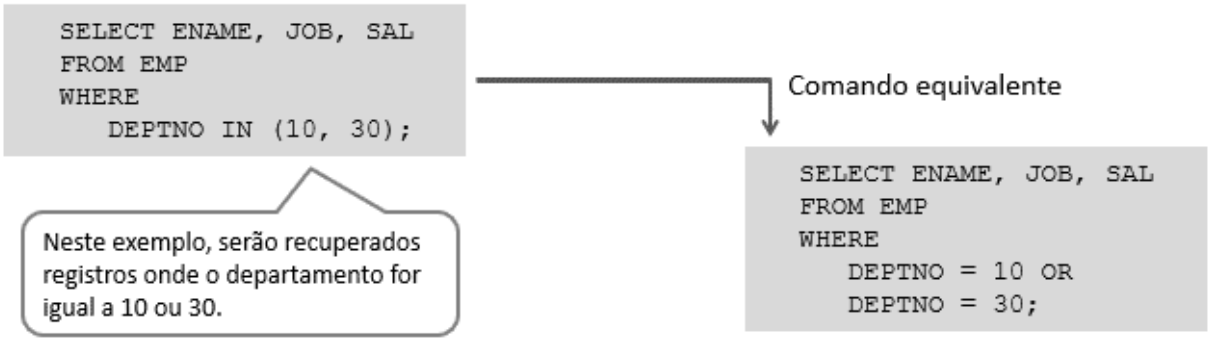

Figura 50

Outro operador não muito comum, mas interessante, é o BETWEEN, que é utilizado quando desejamos testar um intervalo de valores. Vejamos alguns exemplos apresentados na Figura 51.

Dica

Apesar de não ser muito popular, o operador é padrão ANSII. Para saber mais sobre ele, consulte o manual de utilização do SGBD relacional escolhido por você para nossa disciplina.

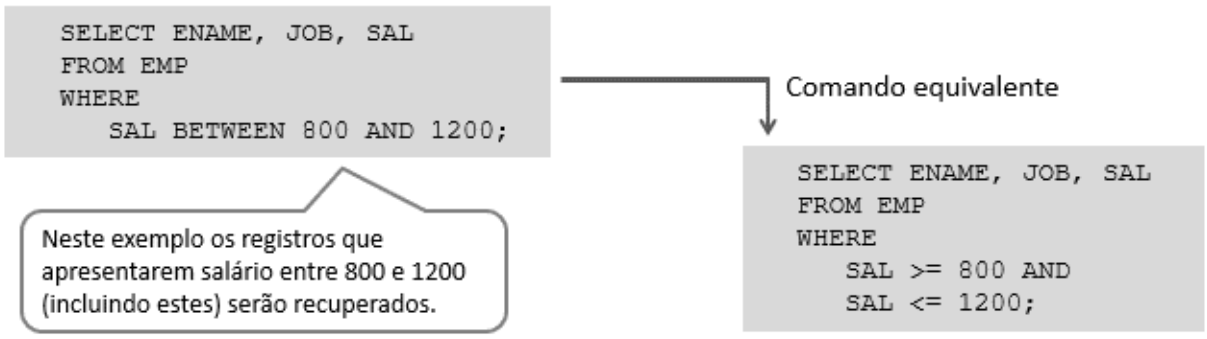

Figura 51

É importante destacar que os valores das extremidades também compõem a lista a ser analisada. Neste operador, assim como nos demais, é possível utilizar a negação da condição através do operador lógico NOT (Ex.: ... WHERE SAL NOT BETWEEN 800 AND 1200).

Dica

É possível que você tenha se perguntado por que afirmamos que o BETWEEN não é um operador muito comum. A resposta está apresentada no exemplo ao lado da utilização, ou seja, muitos programadores optam por escrever a sentença SELECT utilizando a combinação de duas restrições, em vez de utilizar o operador. É importante destacar que não há qualquer alteração em termos de *performance* na recuperação dos dados. Internamente o SGBD relacional unifica os operadores como sendo iguais.

Finalizamos a exploração dos principais operadores utilizados na cláusula WHERE. Aliás, destacamos que não são exclusivamente operadores desta cláusula WHERE – veremos mais adiante que eles poderão ser utilizados em outras cláusulas também. Agora avançaremos para a cláusula ORDER BY.

A cláusula ORDER BY é opcional no comando SELECT. Ela permite especificar a ordem em que as linhas (registros) serão apresentadas no resultado. Veja o exemplo apresentado no Quadro 53.

Quadro 53 – Exemplo do comando SELECT com cláusula ORDER BY ascendente

```
SELECT ENAME, JOB, SAL
FROM EMP
ORDER BY ENAME;
```

No exemplo acima, o resultado será ordenado pela coluna "ENAME", do menor para o maior. Esta é a opção padrão da cláusula, ou seja, a ordenação ascendente (ASC): ordenar sempre do menor para o maior quando não for especificado o critério de ordenação. Porém, é possível indicar que a sequência seja ordenada do maior para o menor. Para tanto, é necessário utilizar a palavra reservada "DESC", conforme o exemplo apresentado no Quadro 54.

Quadro 54 – Exemplo do comando SELECT com cláusula ORDER BY descendente

```
SELECT ENAME, JOB, SAL
FROM EMP
ORDER BY SAL DESC;
```

No exemplo acima, o resultado das linhas recuperadas será apresentado considerando a ordem do maior salário para o menor. Ainda sobre a cláusula ORDER BY, é importante destacar que é possível utilizar mais de uma coluna – as quais podem apresentar critérios diferentes de ordenação. Veja o exemplo apresentado no Quadro 55.

Quadro 55 – Exemplo do comando SELECT com cláusula ORDER BY composta

```
SELECT ENAME, JOB, SAL
FROM EMP
ORDER BY SAL DESC, ENAME [ASC];
```

Dica

Observe que no exemplo temos a utilização da palavra reservada ASC entre colchetes. Isso indica que ela (ASC) é opcional, isto é, se a omitirmos, implicitamente o resultado será ordenado do menor para o maior, ou seja, ascendentemente.

No exemplo acima serão apresentadas linhas ordenadas, primeiramente por ordem de maior salário (SAL) e, em segundo critério, o nome (ENAME) do empregado por ordem crescente, ou seja, do menor para o maior.

Outro recurso possível de utilizarmos na cláusula ORDER BY é a posição da coluna na cláusula SELECT como elemento de ordenação. Veja o código da sentença apresentada no Quadro 56.

Quadro 56 – Sentença SQL com ordenação pela posição da coluna

```
SELECT ENAME, JOB, SAL
FROM EMP
 ORDER BY 3 DESC, 1;
```

No exemplo acima, o resultado será ordenado pela coluna "SAL" do maior para o menor e, caso haja mais de uma linha com o mesmo salário, será apresentado o registro cujo nome (ENAME) for o menor (segundo o conjunto alfanumérico).

Os exemplos apresentados (Quadros 55 e 56) são equivalentes, porém com diferenças na cláusula ORDER BY. Qualquer coluna pertencendo à(s) tabela(s) indicada(s) na cláusula FROM pode ser utilizada para ordenação, mesmo que esta não seja apresentada no resultado.

Cada SGBD relacional apresenta um conjunto de funções de apoio à manipulação de dados alfanuméricos. É recomendável que você consulte sempre a documentação do SGBD relacional que você estiver utilizando. Mesmo assim, é importante destacar duas funções muito utilizadas e que são encontradas em todos os SGBDs relacionais: a UPPER e a LOWER. Estas funções têm o papel de converter o conjunto de dados passados como parâmetros para maiúsculo e minúsculo, respectivamente. Vejamos um exemplo da sua utilização no Quadro 57.

Quadro 57 – Sentença SQL com as funções UPPER e LOWER

```
SELECT UPPER (ENAME),
            LOWER (JOB)
FROM EMP;
```

No exemplo dado, o resultado apresentado vai conter o nome (ENAME) em caracteres maiúsculos; e a atividade (JOB), em caracteres minúsculos. Essas funções também poder ser utilizadas em outras cláusulas, como neste exemplo, na cláusula WHERE, apresentada no Quadro 58.

Quadro 58 – Funções UPPER e LOWER na cláusula WHERE

```
SELECT ENAME, JOB
FROM EMP
WHERE
    UPPER (ENAME) like 'SMIT%';
```

No exemplo do quadro acima temos a restrição do nome (ENAME) sendo comparada sempre em um conjunto maiúsculo em relação ao valor passado pelo operador LIKE.

> **Dica**
>
> É importante destacar que, por padrão, os SGBDs relacionais diferenciam conjuntos de dados (caracteres) entre maiúsculos e minúsculos. Por esta razão é interessante avaliar a possibilidade da utilização dos operadores de conversão no momento da comparação utilizada na restrição de busca.

Encerramento

O comando SELECT apresenta 6 (seis) cláusulas, sendo que até este momento aprendemos a utilizar quatro (4) delas (SELECT, FROM, WHERE e ORDER BY). Abordamos os tipos e os principais operadores utilizados na cláusula WHERE. Também aprendemos a especificar critérios para a exibição de dados. Por fim, conhecemos algumas funções que podem nos ajudar na manipulação de dados, especialmente os alfanuméricos. Nosso próximo passo é na direção da recuperação de dados provenientes de mais de uma tabela.

CONTINUE

Referências

Referências

ORACLE. Documentação de utilização do sistema (Versão 11g). Califórnia: Oracle Corp., 2017.

CARDOSO, Vírginia M. **Linguagem SQL**: fundamentos e práticas. São Paulo: Saraiva, 2009.

Referências de Imagens

Divisão de Modalidades de ensino (DME), Fundação Universidade Regional de Blumenau (FURB), 2019.