Banco de Dados - Ciclo 3 - Webaula 22

FURB

WEBAULA 22

- Seleção de dados com funções de agrupamento
- Referências

QUESTION BANKS

Seleção de dados com funções de agrupamento

Vamos conhecer agora os recursos para fazer consultas de dados através de funções de agrupamento de dados. Para alguns autores, esses recursos da SQL que processam grandes conjuntos de dados e os transformam em informações sumarizadas é que tornam a linguagem diferenciada em relação a quaisquer outras linguagens de programação de sistemas existentes. Aprenderemos, inicialmente, o que são funções de agrupamento de dados ou, se preferir, funções de grupo.

Dica

Assim como nas demais seções de conteúdo, recomendamos que você vá executando as sentenças SQL para colocar em prática os conceitos apresentados.

As funções de grupo operam em conjuntos de linhas (registros) para fornecer um resultado por grupo. Os conjuntos de linhas podem ser uma tabela inteira ou uma tabela dividida em grupos menores. Para exemplificar a utilização das funções de grupo, utilizaremos um conjunto de tabelas e dados adaptados de Oracle (2017), que entendermos ser um cenário simples e com excelente capacidade de demonstração das funcionalidades pretendidas. Observe a Figura 62.

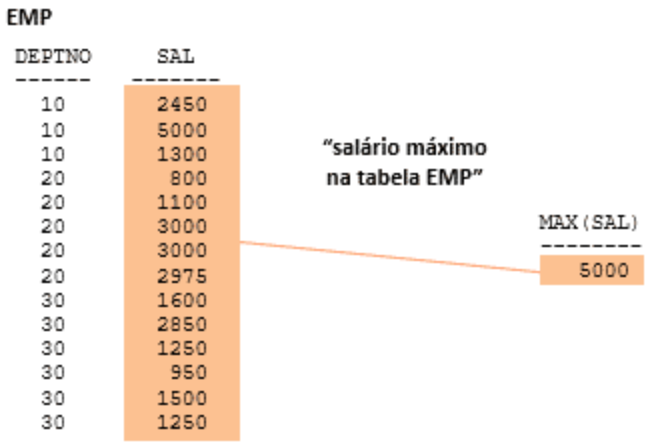

Figura 62

No exemplo acima estamos supondo que se deseja conhecer/recuperar o maior salário (coluna SAL) entre os empregados (tabela EMP). Tradicionalmente, utilizando os recursos básicos da linguagem SQL, você poderia construir uma sentença SQL ordenando pelo maior salário, logo a primeira linha estaria apresentando o maior salário, não é mesmo? Pois bem, isso nos parece simples, e de fato é – mas para este cenário, é claro. Em breve você vai perceber que em cenários mais complexos esta solução trivial não pode ser aplicada e então teremos que recorrer aos recursos que você está conhecendo agora. De volta ao exemplo, observe que ao utilizar uma função conhecida como MAX o SGBD relacional realiza o processamento e apresenta o maior valor da coluna passada como parâmetro – no exemplo, a coluna SAL. Seguiremos com o conteúdo para conhecer como usar as funções de agrupamento de dados no comando SELECT.

Observe a sintaxe de utilização da função de agrupamento de dados na cláusula SELECT, no Quadro 66.

Quadro 66 – Sintaxe de utilização de função de grupo no comando SELECT

```
SELECT    função_de_grupo(coluna)
FROM      tabela
```

```
[WHERE      condição]

[ORDER BY   coluna];
```

Mais adiante você será apresentado à cláusula GROUP BY. Mas por enquanto é importante destacar que as cláusulas WHERE e ORDER BY são utilizadas conforme a necessidade e não são fundamentais para a utilização das funções de agrupamento.

> **Dica**
>
> Fique muito atento a todas as explicações e exemplos. Conceitualmente, este é o assunto que mais exigirá de você na aprendizagem básica da linguagem SQL.

São cinco as principais funções de grupo definidas pelo padrão ASCII:

- AVG: utilizada para calcular o valor médio;
- COUNT: utilizada para contar o número de ocorrências de linhas (registros) de uma tabela;
- MAX: retorna o maior valor;
- MIN: retorna o menor valor;
- SUM: retorna a soma dos valores.

Todas as funções de grupo recebem como parâmetro uma coluna ou expressão. Os tipos de dados da coluna para o argumento podem ser de carácter fixo ou variável, número inteiro ou com casa decimal e data. Mas cada qual tem suas particularidades em relação ao parâmetro passado. Vamos conhecer isso em seguida. Já uma expressão, normalmente matemática (ex. SAL * 0.33), é pouco utilizada, porém possível. Em qualquer caso, deve retornar um dos tipos de dado previstos em sua funcionalidade. Todas as funções de grupo, exceto o COUNT, quando utilizado o parâmetro asterisco, ignoram colunas nulas, ou seja, sem valor. A seguir vamos explorar as funções por meio de exemplos, e assim conhecer um pouco mais as suas particularidades.

Você deve utilizar as funções de grupo AVG e SUM somente para dados numéricos, pois o retorno delas sempre será um valor numérico. Vejamos um exemplo (Quadro 67) onde é recuperada a média salarial e a soma de todos os salários entre os empregados que são vendedores.

Quadro 67 – Exemplo de utilização das funções de grupo AVG e SUM

```
SELECT   AVG(sal), SUM(sal)
FROM     emp
WHERE    job = 'SALESMAN';
```

Considerando o conjunto de dados utilizados como exemplo, adaptado de Oracle (2017), teremos como retorno o seguinte resultado apresentado no Quadro 68.

Quadro 68 – Retorno de sentença utilizando as funções de grupo AVG e SUM

```
AVG(SAL)    SUM(SAL)
--------------   --------------
       1400           5600
```

No exemplo apresentado acima temos que a média salarial dos empregados “vendedores” (*salesman*) é 1400. É importante destacar que em havendo algum registro onde o valor do salário não tenha sido informado (portanto é *null*), este não foi considerado para efeito de cálculo da média. Já o valor de 5600 é resultado da soma de todos os salários dos respectivos empregados.

As próximas funções, MIN e MAX, podem ser utilizadas para qualquer tipo de dado. O exemplo a seguir (Quadro 69) ilustra a sentença SQL para recuperar a data em que o primeiro empregado (mais antigo) foi contratado e a data de contratação do empregado mais recente.

Quadro 69 – Exemplo de utilização das funções de grupo MIN e MAX com data

```
SELECT MIN(hiredate), MAX(hiredate)

FROM emp;
```

Como retorno teremos o resultado apresentado no Quadro 70, onde se observam os valores do tipo data.

Quadro 70 – Retorno de sentença utilizando funções de grupo MIN e MAX com data

```
MIN(HIREDATE)   MAX(HIREDATE)
--------------------- ------------------------
17/12/80            12/01/83
```

Dica

O formato de apresentação do resultado de uma coluna do tipo data pode variar de um SGBD relacional para outro. No exemplo do Quadro 69 temos o formato dia, mês e ano (dd/mm/aa).

O próximo exemplo (Quadro 71) ilustra a sentença SQL para recuperar o menor e o maior salário dentre os empregados que são vendedores.

Quadro 71 – Exemplo de utilização das funções de grupo MIN e MAX com numérico

```
SELECT  MIN(sal), MAX(sal)
FROM    emp
WHERE  job = 'SALESMAN';
```

Ao executarmos a sentença SQL do exemplo acima teremos como resultado a apresentação dos valores do Quadro 72. Como é possível observar, o resultado corresponde a valores numéricos, o que atesta que as funções MIN e MAX podem ser utilizados para coluna(s) com qualquer tipo de dado.

Quadro 72 – Retorno de sentença utilizando funções de grupo MIN e MAX com numérico

```
MIN(SAL)     MAX(SAL)
-------------     --------------
      1250            1600
```

A seguir temos a função COUNT. A função COUNT, geralmente utilizada com o asterisco (*), retorna o número de linhas (registros) em uma tabela ou conjunto de dados restringidos pela condição da

cláusula WHERE. O exemplo do Quadro 73 exibe o número de empregados vinculados ao departamento 30.

Quadro 73 – Exemplo de utilização da função de grupo COUNT

```
SELECT COUNT(*)
FROM        emp
 WHERE  deptno = 30;
```

A execução da sentença SQL do Quadro 73 terá como resultado o valor apresentado no Quadro 74, uma vez que foi somado o número de linhas (registros) onde a condição (deptno = 30) foi atendida, independentemente da existência de colunas nulas nas linhas analisadas.

Quadro 74 – Retorno da sentença de utilização da função de grupo COUNT

```
COUNT(*)
--------------
           6
```

Já no exemplo abaixo (Quadro 75) temos a função COUNT recebendo como parâmetro uma coluna. Dessa forma, apenas as linhas que apresentarem um valor para a coluna COMM serão consideradas.

Quadro 75 – Exemplo de utilização da função de grupo COUNT com parâmetro coluna

```
SELECT COUNT(comm)
```

```
FROM emp
WHERE  deptno = 30;
```

Como retorno da busca realizada pela sentença acima, temos o valor “4”, pois somente as linhas (registros) onde havia um valor para a coluna comissão (comm) foram consideradas. O Quadro 76 ilustra este retorno.

Quadro 76 – Retorno da sentença de utilização da função de grupo COUNT

```
COUNT(*)
--------------
           4
```

Dica

Como vimos, a função COUNT apresenta duas formas de utilização: com o parâmetro asterisco e com uma coluna ou expressão. Com o parâmetro asterisco ela retorna o número de linhas de uma tabela, inclusive linhas duplicadas e linhas contendo valores nulos em qualquer uma das colunas. Se uma cláusula WHERE estiver incluída na instrução SELECT, COUNT com asterisco retornará o número de linhas que satisfizerem a condição na cláusula WHERE. É o caso do primeiro exemplo apresentado (Quadro 72), onde é exibido o número de empregados vinculados ao departamento 30. Entretanto, COUNT com uma coluna passada como parâmetro retorna o número de linhas não nulas da coluna identificada. É o segundo exemplo apresentado (Quadro 74), onde temos um retorno de números de empregados do departamento 30 cuja comissão apresente um valor válido, ou seja, diferente de nulo.

Com visto, as funções de grupo ignoram valores nulos na coluna. Por isso, você deve ficar muito atento e, quando necessário, utilizar uma função para converter o valor da coluna em tempo real, a fim de evitar distorções no resultado apurado. Vejamos um exemplo para que você compreenda o que estamos dizendo. A sentença SQL abaixo (Quadro 77) tem por objetivo exibir a comissão média recebida pelos empregados.

Quadro 77 – Exemplo de cálculo da média de comissão

```
SELECT AVG(comm)

FROM    emp;
```

O Quadro 78 apresenta o valor correspondente ao retorno da execução da sentença.

Quadro 78 – Retorno da sentença de utilização da função de grupo AVG

```
AVG(COMM)
---------------------
          550.00
```

Se você realizar um teste de mesa, vai perceber que a comissão média recebida pelos empregados não é o valor exibido. Isso ocorre porque a função de grupo desprezou as linhas (registros) que não apresentam um valor válido para a coluna COMM. No contexto, a soma das comissões foi de 2200, e esse valor foi dividido por 4, pois esse é o número de empregados que apresentam um valor válido para a coluna que foi somada. A seguir vamos descobrir como contornar isso!

Para contornar o problema ilustrado é necessário utilizar uma função que transforme, em tempo real, o valor nulo da coluna que foi passada como parâmetro em um valor válido que não interfira no cálculo (neste caso, zero). No SGBD Oracle, essa função é a NVL. Veja o exemplo apresentado no Quadro 79.

Quadro 79 – Exemplo de utilização da função de grupo AVG com NVL

```
SELECT AVG(NVL(comm,0))

FROM   emp;
```

No exemplo do Quadro 79 é possível observar que a função de conversão NVL está recebendo dois parâmetros: a coluna a ser convertida e o valor a ser adotado. Neste caso, na linha onde houver *null* para a coluna COMM, a função de grupo AVG irá considerar como sendo "0" (zero) o valor e, desta forma, utilizar a referida linha para fins de cálculo da média das comissões entre os empregados. O Quadro 80 apresenta o valor médio apurado com o uso da função de conversão.

Quadro 80 – Retorno da utilização da função de grupo COUNT com NVL

```
AVG(NVL(COMM,0))
----------------------------
          157.142857
```

Dica

A função de conversão *null* para outro valor não é padrão ANSI. Assim, é necessário ficar atento ao SGBD relacional que você vai utilizar. Alguns, como o MySQL, correspondem à função IFNULL. Por isso recomendamos sempre uma consulta ao manual de utilização de cada SGBD relacional.

Até este momento todas as funções de grupo trataram a tabela como um grande grupo de dados. Às vezes é necessário dividir a tabela de dados em grupos menores. Isto pode ser feito através da cláusula GROUP BY. Observe a Figura 63, onde temos um exemplo envolvendo a tabela EMP. No cenário, o objetivo é criar grupos de dados por departamento para exibir a média salarial dos empregados vinculados a cada um deles (departamentos).

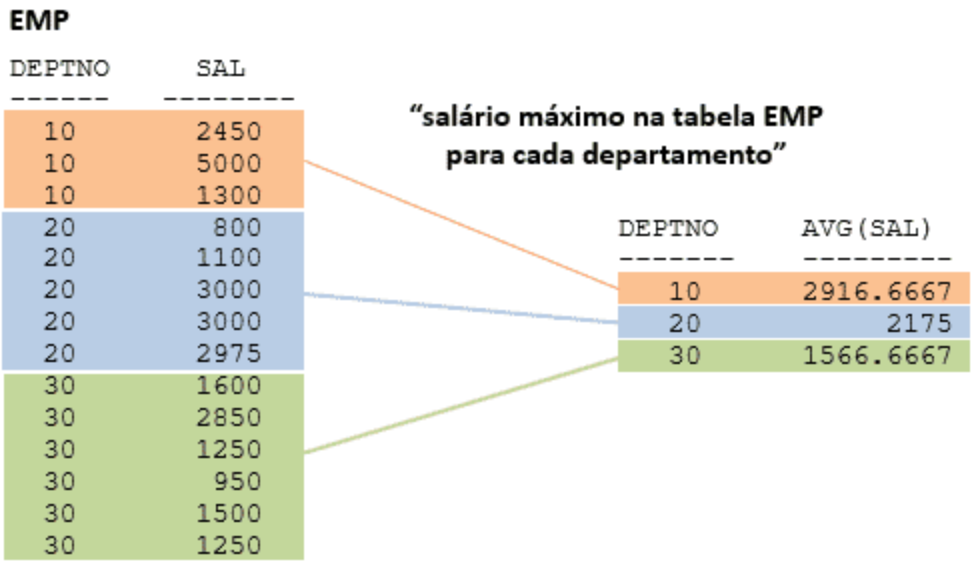

Figura 63
Fonte: Adaptado de Oracle (2017)

Até então a média apurada era considerada por todos os empregados, ou seja, todos os registros da tabela EMP. A partir de agora você vai conhecer como agrupar dados em conjuntos menores através da cláusula GROUP BY. O Quadro 81 ilustra a sintaxe da utilização da cláusula GROUP BY.

Quadro 81 – Sintaxe da utilização da cláusula GROUP BY

```
SELECT        colunaX, função_de_grupo (coluna)

FROM          tabela

[WHERE        condição]

GROUP BY      colunaX

[ORDER BY     coluna];
```

Ao observarmos a sintaxe apresentada, é importante destacar que a coluna que acompanha a função de grupo (colunaX) é opcional, mas, se acrescentada, a cláusula GROUP BY deverá necessariamente ser utilizada. Já as cláusulas WHERE e ORDER BY são utilizadas conforme a necessidade e não são fundamentais para a utilização das funções de agrupamento.

Algumas diretrizes que você deve considerar: se você incluir uma função de grupo em uma cláusula SELECT, não poderá selecionar resultados individuais, a menos que a coluna individual apareça na cláusula GROUP BY; se você não conseguir incluir a lista de colunas, uma mensagem de erro será exibida; ao usar uma cláusula WHERE você pode excluir linhas com antecedência, antes mesmo de dividi-las em grupos.

Dica

Não é possível usar o apelido de coluna na cláusula GROUP BY; por *default* as linhas são classificadas por ordem crescente das colunas incluídas na lista GROUP BY. Isto pode ser modificado através da cláusula ORDER BY.

Como dito, todas as colunas na lista SELECT que não estejam em funções de grupo devem estar na cláusula GROUP BY. Veja o exemplo do Quadro 82, que corresponde ao comando ilustrado no cenário recente (Figura 63) onde se deseja exibir o valor médio dos empregados por departamento de lotação

Quadro 82 – Exemplo da utilização da cláusula GROUP BY

```
SELECT      deptno, AVG(sal)

FROM        emp

GROUP BY    deptno
```

Como retorno da execução da sentença apresentada (Quadro 82), temos o seguinte resultado, ilustrado no Quadro 83, com todos os valores dos salários somados considerando-se o departamento de lotação. A partir disso, uma média aritmética simples foi realizada (papel da função de grupo AVG).

Quadro 83 – Resultado da utilização da cláusula GROUP BY

```
DEPTNO      AVG(SAL)
----------  ----------
        10  2916.6667
        20  2175
        30  1566.6667
```

Quando utilizamos a cláusula GROUP BY, é necessário certificar-se de que todas as colunas na lista SELECT que não estejam nas funções de grupo estejam incluídas na cláusula GROUP BY. O exemplo

exibe o número de departamento do salário médio para cada departamento. Essa instrução SELECT que contém uma cláusula GROUP BY é avaliada da seguinte forma: (i) a cláusula SELECT especifica as colunas a serem recuperadas, a coluna de número de departamentos na tabela EMP, a média de todos os salários no grupo que você especificou na cláusula GROUP BY; (ii) a cláusula FROM especifica a tabela que o banco de dados deve acessar – no caso, a tabela EMP; (iii) a cláusula WHERE especifica as linhas a serem recuperadas: já que não há uma clausula WHERE, todas as linhas serão recuperadas por *default*; (iv) a cláusula GROUP BY especifica como as linhas devem ser agrupadas, ou seja, as linhas são agrupadas pelo número de departamento, de forma que a função AVG que esteja sendo aplicada à coluna de salários calcule o salário médio para cada departamento.

A coluna inserida na cláusula GROUP BY não precisa estar na lista de colunas da cláusula SELECT. Porém, se estiver na cláusula SELECT junto a uma função de grupo, esta, por sua vez, deve estar na cláusula GROUP BY. Veja um exemplo onde ocorre o agrupamento, com a coluna utilizada para agrupar ausente na cláusula SELECT (Quadro 84).

Quadro 84 – Exemplo da utilização da cláusula GROUP BY sem exibir a coluna agrupadora

```
SELECT      AVG(sal)

FROM        emp

GROUP BY    deptno;
```

Como resultado teremos os dados apresentados sem a coluna utilizada para o agrupamento dos dados. Isso pode ser constatado no Quadro 85.

Quadro 85 – Resultado da utilização da cláusula GROUP BY sem a coluna agrupadora

```
AVG(SAL)
-------------
2916.6667
      2175
1566.6667
```

Como podemos observar, sem informações dos departamentos que serviram de critério para o agrupamento dos dados apresentados, os resultados não parecem significativos. Casos como esses são raros, mas podem ser necessários.

Há também situações em que desejamos agrupar dados considerando um número maior de colunas, ou seja, temos a necessidade de ver os resultados para grupos dentro de grupos. A Figura 64 mostra um cenário onde temos os dados originais da tabela EMP e ao lado um relatório que exibe o salário total sendo pago a cada cargo dentro de cada departamento.

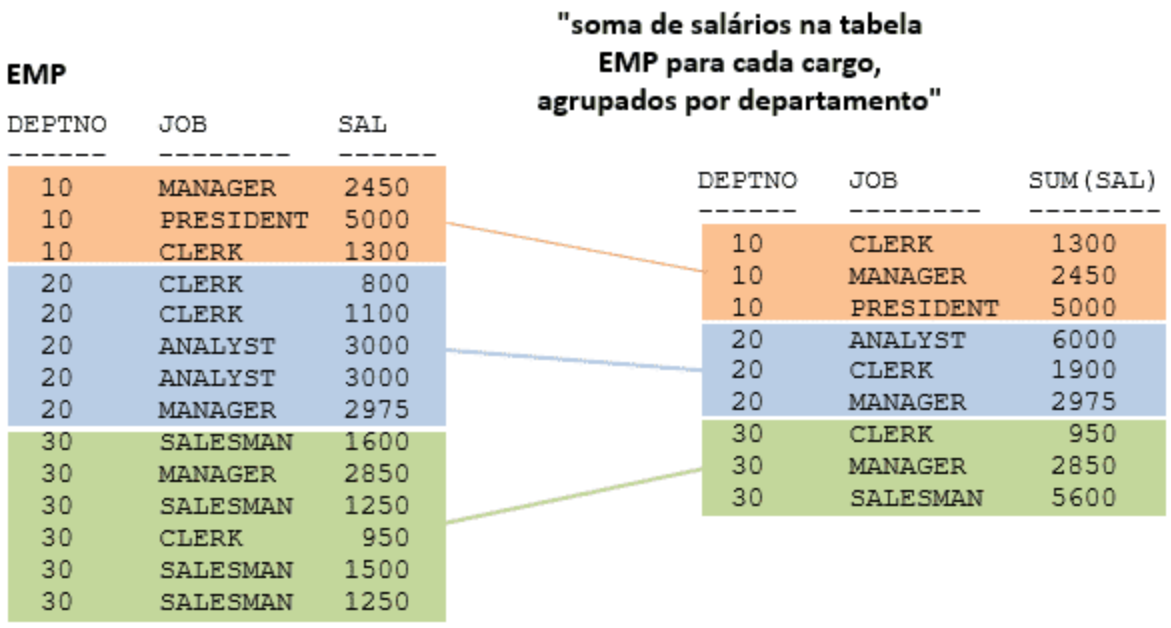

Figura 64
Fonte: Adaptado de Oracle (2017)

No cenário temos que a tabela EMP é agrupada primeiro pelo número do departamento (DEPTNO) e dentro deste agrupamento é agrupada pelo cargo (JOB). Podemos observar, por exemplo, que os dois escriturários (“clerk”) do departamento 20 estão agrupados em um único valor representado pelo salário total (1900), resultado da soma de todos os vendedores dentro do grupo de cargo.

O exemplo a seguir (Quadro 86) corresponde à sentença SQL para o cenário apresentado anteriormente (Figura 64). Observe a presença das duas colunas na cláusula SELECT e também na cláusula GROUP BY.

Quadro 86 – Exemplo da utilização da cláusula GROUP BY com mais de uma coluna agrupadora

```
SELECT deptno, job, sum(sal)

FROM emp

GROUP BY deptno, job;
```

O resultado proveniente da execução da sentença acima é apresentado no Quadro 87.

Quadro 87 – Resultado da utilização da cláusula GROUP BY com mais de uma coluna agrupadora

```
DEPTNO   JOB        SUM(SAL)
---------- ---------      -------------
    10       CLERK            1300
    10    MANAGER         2450
    10 PRESIDENT          5000
```

```
20 ANALYST          6000
... 9 linhas recuperadas
```

No resultado do exemplo acima foram omitidas as demais linhas resultantes, porém o resultado seria certamente o conjunto de dados ilustrado na Figura 64 (relatório à direita).

Como vimos, é possível retornar resultados sumarizados para grupos e subgrupos listando mais de uma coluna na cláusula GROUP BY. Você pode determinar a ordem de classificação padrão dos resultados pela ordem em que indicar as colunas na cláusula GROUP BY. A instrução SELECT no exemplo contém uma cláusula GROUP BY que é avaliada da seguinte forma: (i) a cláusula SELECT especifica a coluna a ser recuperada – no caso, o número do departamento (EMPNO), o cargo (JOB) e a soma de todos os salários (SUM) que foi especificada, ou seja, agrupada a partir da cláusula GROUP BY; (ii) A cláusula FROM especifica a tabela que o banco de dados deve acessar – no caso, a tabela EMP; (iii) a cláusula GROUP BY especifica o que deve agrupar as linhas, ou seja, primeiro as linhas são agrupadas pelo número do departamento (DEPTNO), em seguida dentro dos grupos de número de departamentos as linhas são agrupadas pelo cargo (JOB). Dessa forma a função SUM é aplicada à coluna de salários (SAL) para todos os cargos dentro de cada grupo de número departamento.

Como já mencionado em outros momentos, qualquer coluna ou expressão na lista SELECT que não seja uma função de grupo deve estar na cláusula GROUP BY. Veja um exemplo (Quadro 88) que apresenta uma sentença SQL seguida de uma mensagem de erro comum quando omitimos a cláusula GROUP BY.

Quadro 88 – Erro ao omitir a cláusula GROUP BY com função de grupo

```
SELECT   deptno, COUNT (ename)

FROM    emp;
```

```
SELECT deptno, COUNT(ename)
*
ERRO na linha 1:Nenhuma função de grupo único
```

No exemplo, observe que há a presença de uma função de grupo (COUNT) na cláusula SELECT juntamente com uma coluna (DEPTNO). A indicação desta coluna sinaliza para o SGBD relacional a intenção de exibir a contagem de empregados (COUNT) considerando cada departamento existente. O erro, porém, ocorre, pois omitimos a cláusula GROUP BY.

Outro aspecto que devemos observar está relacionado à utilização de funções de grupo para restringir valores. Isso porque não é possível usar a cláusula WHERE para restringir grupos. Neste caso, você deve utilizar a cláusula HAVING. Antes, porém, veja um exemplo de uma tentativa de utilizar a função de grupo na cláusula WHERE e o respectivo erro, conforme exibido no Quadro 89.

Quadro 89 – Erro ao utilizar restrição com função de grupo na cláusula WHERE

```
SELECT      deptno, AVG(sal)
FROM        emp
WHERE       AVG(sal) > 2000
GROUP BY    deptno;



Erro na linha 3: A função de grupo não é permitida aqui
```

Portanto, concluímos que uma função de grupo não pode ser utilizada na cláusula WHERE. Então como podemos restringir os grupos a partir dos resultados obtidos pela função de grupo? É

exatamente o que veremos em seguida. Para isso, observe a Figura 65.

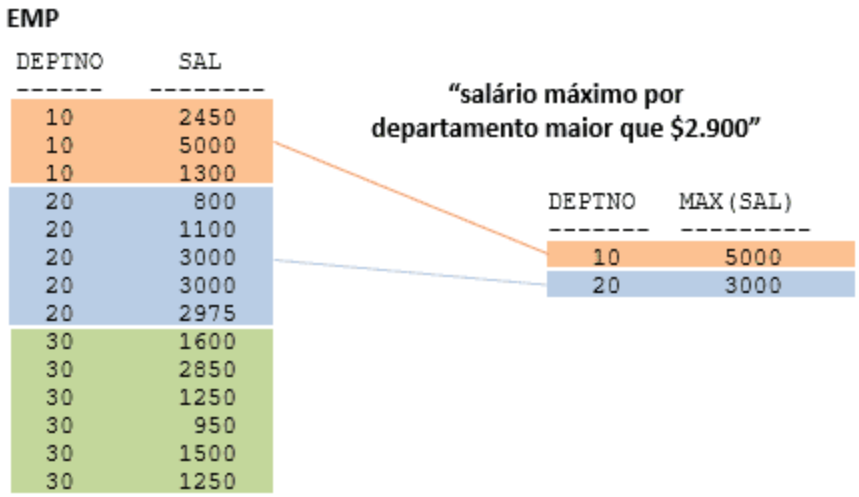

Figura 65
Fonte: Adaptado de Oracle (2017)

Na ilustração acima observe que à direita temos o conjunto de dados completo (tabela EMP) e à esquerda vemos que não está sendo exibido o resultado do agrupamento de dados do departamento 30, uma vez que ele foi desprezado no momento da recuperação. Da mesma forma que você usa a cláusula WHERE para restringir as linhas que seleciona, podemos utilizar a cláusula HAVING para restringir os grupos.

O Quadro 90 apresenta a sintaxe do comando SELECT com a inserção da cláusula HAVING, utilizada para restringir grupos de dados, ou seja, dados obtidos a partir de funções de grupo.

Quadro 90 – Sintaxe do SELECT com a cláusula HAVING

```
SELECT    [colunaX,] função_de_grupo(coluna)

FROM      tabela
```

```
[WHERE      condição]

[GROUP BY   colunaX]

[HAVING     condição_de_grupo]
```

Use a cláusula HAVING para especificar quais grupos de dados serão exibidos. Na sintaxe temos a cláusula HAVING com a condição de grupo. A condição de grupo restringe os grupos de linhas retornados aos grupos para os quais a condição especificada seja verdadeira. Em geral o SGBD relacional executa as seguintes etapas quando você usa cláusula HAVING: as linhas são agrupadas, a função de grupo é aplicada ao grupo, os grupos que correspondem aos critérios na cláusula HAVING são exibidos.

> **Dica**
>
> A cláusula HAVING pode preceder a cláusula GROUP BY, mas recomendamos que você coloque a cláusula GROUP BY primeiro por ser mais lógico. Os grupos são formados e as funções de grupos são calculadas antes de a cláusula HAVING ser aplicada aos grupos na lista de colunas do SELECT.

O exemplo a seguir (Quadro 91) corresponde ao cenário apresentado anteriormente (Figura 65). Observe a presença da cláusula HAVING para restringir o grupo a ser exibido.

Quadro 91 – Exemplo de SELECT com a cláusula HAVING

```
SELECT      deptno, max(sal)
FROM        emp
```

```
GROUP BY deptno
HAVING     max(sal) > 2900;
```

Considerando o conjunto de dados já apresentado para a tabela EMP, como retorno à execução do comando acima teremos o que está ilustrado no Quadro 92.

Quadro 92 – Resultado do exemplo de SELECT com a cláusula HAVING

```
 DEPTNO    MAX(SAL)
-------------   --------------
         10            5000
         20            3000
```

O exemplo apresentado exibe números de departamentos e o salário máximo para os departamentos onde o salário máximo seja maior que 2.900. Você pode usar a cláusula GROUP BY sem usar uma função de grupo na SELECT. Se você restringe linhas com base no resultado de uma função de grupo, deve ter a cláusula GROUP BY, assim como a cláusula HAVING.

Com a apresentação da cláusula HAVING, completamos o comando SELECT com todas as cláusulas. Observe no Quadro 93 a presença de todas as cláusulas no comando por meio de um exemplo.

Quadro 93 – Comando SELECT com todas as cláusulas

```
SELECT     job, SUM(sal) TOTAL
FROM       emp
WHERE      job NOT LIKE 'SALES%'
```

```
GROUP BY job
HAVING      SUM(sal) > 5000
ORDER BY  SUM(sal);
```

Como resultado da execução da sentença acima, teremos o seguinte retorno (Quadro 94). O resultado do exemplo exibe o cargo e o salário mensal total para cada cargo, com uma folha de pagamento total excedendo $5.000. O exemplo exclui vendedores e classifica a lista pelo salário mensal total.

Quadro 94 – Resultado de sentença SELECT com todas as cláusulas

```
JOB              TOTAL
---------        -----------
ANALYST          6000
MANAGER          8275
```

Encerramento

Nesta seção conhecemos as funções de agrupamento de dados: AVG, COUNT, MAX, MIN e SUM. Também aprendemos a criar grupos de dados e restringi-los. Por fim, as últimas duas cláusulas do comando SELECT foram apresentadas – GROUP BY e HAVING –, e agora ele está completo. A próxima seção irá tratar de subconsultas, ou seja, sentenças SELECT com a inserção de outras sentenças SELECT em suas cláusulas.

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.