Banco de Dados

AULA 6 - FUNÇÕES DE AGREGAÇÃO E AGRUPAMENTO

Introdução

- Em aplicações reais é comum haver a necessidade de totalizar valores e agrupar informações.
- O processo de totalização, agrupamento pode ser feito diretamente no SGBD, desta forma, somente os resultados retornam para a aplicação.
- Esta estratégia garante um melhor desempenho, uma vez que um conjunto menor de dados trafegam na rede.

Funções de agrupamento e cálculo sobre campos

- Diversas funções podem ser usadas para resumir informações sobre tabelas e campos:

```
SUM (Campo)
AVG (Campo)
MAX (Campo)
MIN (Campo)
COUNT (Campo/*)
```

- Usando **`DISTINCT`** antes do campo, somente os valores não duplicados são resumidos.

Exemplos

- Exibir o total de funcionários que estão lotados no departamento 80.

```
SELECT COUNT (*) AS "Total"
FROM employee
WHERE department_id = 80;
```

Exemplos

- Exibir a soma de todos os salários dos funcionários contratados a partir de 2014.

```
SELECT SUM (e.salary) AS "Soma salários"
FROM employee e
WHERE DATE_PART ('year', e.hire) >= 2014;
```

Funções de agregação

- Nos exemplos, se não for indicado um critério de seleção, a consulta totaliza ou conta todos os registros.
- A função **`COUNT`** pode ser aplicada sobre qualquer campo ou sobre *, que indica todos os campos.
- A função **`SUM`** ou **`AVG`**, somente pode ser aplicada corretamente em campos numéricos.

Funções de agregação

- Funções **MAX**, **MIN** podem ser aplicadas sobre qualquer valor, porque é possível obter o maior ou menor valor, independente do tipo de dado. Por exemplo, mostrar o funcionário mais antigo:

```
SELECT MIN (hire)

FROM employee
```

- Será mostrado um único valor, portanto o retorno será uma linha apenas, o que inviabiliza mostrar campos que retornem múltiplos registros.

Funções de agregação

- Não é possível exibir dados agregados e outros campos sobre os quais não é aplicada uma função.
- Considerando que um campo no **`SELECT`** indica diversos valores para este campo, é incompatível exibir um resultado agregado e outro não agregado.

Agrupamento

- A cláusula para agrupamento **`GROUP BY`**, pode ser usada juntamente com funções para resumir valores de campos.
- Quando ocorre um agrupamento, a função é executada para cada valor que ocorre no(s) campo(s) indicado na cláusula **`GROUP BY`**.
- **`GROUP BY`** é utilizado após a cláusula **`WHERE`**.

Agrupamento

- Por exemplo: mostrar a soma dos salários de cada departamento.

```
SELECT department_id, SUM (salary)
FROM employee
GROUP BY department_id;
```

- Ocorre um agrupamento para o campo department_id. Para cada valor diferente em department_id será produzido um resultado de soma do campo salary.

Agrupamento

- Neste exemplo, se for necessário mostrar também o nome do departamento, deverá ocorrer um agrupamento pelo código e pelo nome.
- Embora para cada código, o nome seja sempre o mesmo, o processador de consultas entende que existe um campo sobre o qual existe um agrupamento (**`employee.department_id`**) e outro sobre o qual não há agrupamento (**`department.name`**), nem aplicação de função.
- Desta forma, o campo **`name`** não pode aparecer na cláusula **`SELECT`**.
- Para resolver existem duas alternativas possíveis.

Agrupamento

- 1 - Colocar a descrição na cláusula **`GROUP BY`**

```
GROUP BY employee.department_id, department.name
```

- 2 - Aplicar uma função **`MAX`**, **`MIN`** sobre o campo:

```
SELECT employee.department_id, MAX
(department.name)
```

Agrupamento

- Versão final da consulta:

```
SELECT e.department_id, d.name, SUM (e.salary)
FROM employee e, department d
WHERE e.department_id = d.id
GROUP BY e.department_id, d.name;
```

Agrupamento

- É possível agrupar por diversos campos.
- O agrupamento será feito pelo primeiro e dentro dos valores repetidos para este, pelo segundo e assim por diante.
- Somente se houverem valores repetidos dentro do primeiro campo, haverá algum efeito em um agrupamento com diversos campos.

Valores nulos

- Todas as funções de grupo ignoram os valores nulos da coluna;
- Entretanto, a função **COALESCE** força as funções de grupo a incluírem os valores nulos;
- No exemplo abaixo, todas os registros com comissões nulas passam a valer 0 e são contabilizados na média.

```
SELECT AVG (COALESCE (commission, 0))
FROM employee;
```

Registros selecionados pelo valor retornado por funções

- A cláusula **`WHERE`** serve para selecionar valores de campos armazenados em determinados registros, portanto **`WHERE`** opera individualmente sobre cada registro e não sobre resultados de um agrupamento;
- A tentativa de usar na consulta anterior uma seleção do tipo: **`WHERE SUM (e.salary) > 20000`**, iria gerar um erro de execução;
- O **`WHERE`** tenta operar sobre um campo que não existe ou sobre resultado de uma soma ainda não realizada.

Seleção sobre resultados de uma função

- Para aplicar uma seleção sobre resultados de uma função existe o operador/cláusula **`HAVING`**.
- **`HAVING`** opera sobre resultados de uma função em totalizações, média, máximo, etc.
- A operação é realizada quando os resultados e agrupamentos já foram completados.
- Os critérios possíveis para esta cláusula são semelhantes à **`WHERE`**, porém não limitam os registros que serão processados e sim os resultados a serem mostrados.

Seleção sobre resultados de uma função

- Exemplo: mostre o nome dos empregos nos quais os funcionários possuem, na média, um salário maior ou igual a R$5.000,00

```
SELECT j.name

FROM employee e, job j

WHERE e.job_id = j.id

GROUP BY j.name

HAVING AVG (e.salary) > 5000

ORDER BY j.name DESC;
```

Ordem padrão das operações:

- 1 - Será realizado o produto cartesiano das tabelas (**`FROM`**).
- 2 - As linhas a serem consideradas permanecem (**`WHERE`**).
- 3 - São realizados os agrupamentos (**`GROUP BY`**) e projetados os campos (**`SELECT`**), incluindo o cálculo da soma.
- 4 - São selecionados somente os resultados de soma que atendem ao critério de **`HAVING`**.
- 5 - Os resultados finais são ordenados de acordo com a definição em **`ORDER BY`**.

Exercícios

Ver Lista 4