Unidade 3

Bancos de Dados Relacionais e Linguagem SQL

3.2 Junções e Funções de Grupo

Funções de Grupo

- Funções de grupo operam em uma tabela inteira, em um agrupamento ou conjunto de linhas para fornecer um resultado por grupo.
- **MIN:** usado com colunas que armazenam qualquer tipo de dados para retornar o valor mínimo.
- **MAX:** usado com colunas que armazenam qualquer tipo de dados para retornar o valor máximo.
- **SUM:** usado com colunas que armazenam dados numéricos para encontrar o total ou a soma dos valores.

Funções de Grupo

- **AVG:** usado com colunas que armazenam dados numéricos para calcular a média.
- **COUNT:** retorna o número de linhas contidas em determinada tabela.
- **VARIANCE:** usado com colunas que armazenam dados numéricos para calcular a distribuição dos dados acima ou abaixo da média.
- **STDDEV:** do mesmo modo que a variância, o desvio padrão mede a distribuição dos dados. Em dois conjuntos de dados com aproximadamente a mesma média, quanto maior a distribuição, maior o desvio padrão.

Funções de Grupo

- As funções de grupo **não podem** ser utilizadas na cláusula **WHERE!**
- Vamos utilizar o banco de dados World e World_x.
- Todos os bancos de exemplo estão disponíveis no link:
- https://dev.mysql.com/doc/index-other.html
- Façam o download e importem para os SGBDs de vocês.

Now the new database world_x!

Funções de Grupo

```
use world;
SELECT MIN(Percentage)
FROM countrylanguage
WHERE Language = 'English'
AND IsOfficial = 'T'
AND Percentage >0;
```

```
SELECT MAX(Percentage)
FROM countrylanguage
WHERE Language = 'English'
AND IsOfficial = 'T'
AND Percentage >0;
```

```
SELECT MIN(LifeExpectancy)
FROM country;
```

```
SELECT MAX(LifeExpectancy)
FROM country;
```

```
SELECT MIN(Name)
FROM country;
```

```
SELECT MAX(Name)
FROM country;
```

```
SELECT name
FROM country
ORDER BY name;
```

```
SELECT name
FROM country
ORDER BY name DESC;
```

Funções de Grupo

- **AS FUNÇÕES DE GRUPO NÃO LEVAM EM CONSIDERAÇÃO VALORES NULOS PARA OS CÁLCULOS.**
- **LEMBRE-SE DE QUE NULO É DIFERENTE DE 0 !**
- **ALÉM DISTO ELAS PODEM SER EXECUTADAS PARA O MESMO CONJUNTO DE DADOS GERANDO UM SÓ RESULTADO.**

Funções de Grupo

```
USE employees;
SELECT SUM(SurfaceArea) as "Área da Superfície Total do Caribe"
FROM country
WHERE Region = 'Caribbean';

SELECT ROUND(SUM(salary),2)
FROM salaries s INNER JOIN employees e   ON (s.emp_no=e.emp_no)
                INNER JOIN dept_emp de   ON (e.emp_no=de.emp_no)
                INNER JOIN departments d ON (de.dept_no= d.dept_no)
 WHERE d.dept_name = 'Marketing'
 AND de.to_date ='9999-01-01'
 AND e.hire_date > '1999-01-01';

SELECT ROUND(AVG(salary),2)
FROM salaries s INNER JOIN employees e   ON (s.emp_no=e.emp_no)
                INNER JOIN dept_emp de   ON (e.emp_no=de.emp_no)
                INNER JOIN departments d ON (de.dept_no= d.dept_no)
 WHERE d.dept_name = 'Marketing'
 AND de.to_date ='9999-01-01'
```

```
SELECT ROUND(VARIANCE(LifeExpectancy),4)
FROM country;

SELECT ROUND(STDDEV(LifeExpectancy),4)
FROM country;

SELECT MIN(LifeExpectancy), MAX(LifeExpectancy),
       AVG (LifeExpectancy)
FROM country;

SELECT COUNT(*)
FROM employees
WHERE hire_date < '1999-01-01';
```

Resumo sobre as Funções de Grupo

- Ignoram os valores nulos;
- Não podem ser usadas na cláusula WHERE;
- MIN, MAX e COUNT podem ser usados com qualquer tipo de dados;
- SUM, AVG, STDDEV e VARIANCE podem ser usados somente com tipos de dados numéricos;

DISTINCT

- A palavra-chave DISTINCT é usada para retornar somente valores não duplicados ou combinações de valores não duplicados em uma consulta;
- Podemos utilizá-la em uma consulta que seleciona mais de uma coluna, retornando as combinações não duplicadas das colunas selecionadas;
- Podemos utilizá-la com todas as funções de grupo assim desconsiderando os valores duplicados para os cálculos.

DISTINCT

```
use world;

SELECT DISTINCT(Continent)
FROM country;

SELECT COUNT(DISTINCT(Continent))
FROM country;
```

```
use employees;

SELECT DISTINCT emp_no, dept_no
FROM dept_emp
WHERE to_date = '9999-01-01'
AND dept_no = 'd007';
```

Group by e Having

- A cláusula GROUP BY é usada para dividir as linhas em uma tabela em grupos menores.
- É possível usar as funções de grupo para retornar informações resumidas sobre cada conjunto de dados selecionado.

```
SELECT dept_no,COUNT(emp_no)
FROM dept_emp
WHERE to_date = '9999-01-01'
GROUP BY dept_no
ORDER BY dept_no
```

```
SELECT Continent,SUM(SurfaceArea)
FROM country
GROUP BY Continent
ORDER BY Continent;
```

Group by e Having

- Normalmente, incluímos a coluna *GROUP BY* na lista *SELECT*.
- As funções de grupo requerem que qualquer coluna listada na cláusula *SELECT* que não faça parte de uma função de grupo seja listada em uma cláusula *GROUP BY*.
- Pode-se usar a cláusula WHERE para excluir linhas antes que o restante seja agrupado.
- **Você não pode usar um alias de coluna na cláusula *GROUP BY.***

Group by e Having

- Às vezes, você precisa dividir grupos em subgrupos menores, assim o GROUP BY consegue oferece a visualização apenas adicionando a segunda coluna para agrupamento.

```
SELECT ct.Name , c.District, SUM(c.Population)
FROM city c INNER JOIN country ct ON (c.CountryCode=ct.Code)
GROUP BY c.CountryCode, c.District;

SELECT ct.Name , c.District, AVG(c.Population)
FROM city c INNER JOIN country ct ON (c.CountryCode=ct.Code)
GROUP BY c.CountryCode,c.District;
```

Group by e Having

- Com a cláusula *GROUP BY* é possível aplicar restrições que não conseguimos aplicar utilizando a cláusula *WHERE*.
- Para aplicarmos filtros de restrição nos grupos selecionados utiliza-se a cláusula *HAVING.*
- No plano de execução das cláusulas *GROUP BY* e *HAVING*, as linhas são primeiro agrupadas; as funções de grupo são aplicadas; e, em seguida, são exibidos somente os grupos correspondentes à cláusula *HAVING.*

Group by e Having

```
SELECT ct.Name , c.District, SUM(c.Population)
FROM city c INNER JOIN country ct ON (c.CountryCode=ct.Code)
GROUP BY c.CountryCode, c.District
HAVING COUNT(c.District) > 2;
```

```
use employees;
```

```
SELECT d.dept_name, AVG (salary)
FROM dept_emp de INNER JOIN employees e   ON (de.emp_no=e.emp_no)
                 INNER JOIN departments d ON (de.dept_no=d.dept_no)
                 INNER JOIN salaries    s ON (s.emp_no=e.emp_no)
WHERE de.to_date = '9999-01-01'
AND    s.to_date = '9999-01-01'
GROUP BY d.dept_no
HAVING AVG(salary) > 70000
ORDER BY d.dept_no;
```

```
SELECT d.dept_name, ROUND((MAX(salary)/12),2)
FROM dept_emp de INNER JOIN employees e   ON (de.emp_no=e.emp_no)
                 INNER JOIN departments d ON (de.dept_no=d.dept_no)
                 INNER JOIN salaries    s ON (s.emp_no=e.emp_no)
WHERE de.to_date = '9999-01-01'
AND    s.to_date = '9999-01-01'
GROUP BY d.dept_no
HAVING AVG(salary) > 70000
ORDER BY d.dept_no;
```

Rollup, Cube e Grouping Sets

- Evoluindo mais a nossa análise de dados com a utilização da cláusula *GROUP BY* e da cláusula *HAVING,* pode-se agrupar os subtotais por grupo e o total geral de todas as linhas selecionadas.
- Com o que conhecemos até agora teríamos que executar vários comandos com saídas diferentes e agrupá-los de forma manual.
- Utiliza-se extensões da cláusula *GROUP BY* criadas especificamente para esse propósito: *ROLLUP*, *CUBE* e *GROUPING SETS* o que exige menos trabalho da sua parte e garante eficiência.

Rollup, Cube e Grouping Sets

- Em consultas *GROUP BY*, para atender a necessidade de se produzir subtotais e totais a operação *ROLLUP* é ideal.
- *ROLLUP* cria subtotais que fazem roll-up do nível mais detalhado para um total geral utilizando uma lista ordenada de colunas de agrupamento em sua lista de argumentos.
- O Plano de execução calcula os valores agregados padrão especificados na cláusula *GROUP BY*, cria subtotais de nível progressivamente maior, movendo-se da direita para a esquerda na lista de colunas de agrupamento criando por fim um total geral.

Rollup, Cube e Grouping Sets

```
use world;

SELECT ct.Name , c.District, AVG(c.Population)
FROM city c INNER JOIN country ct ON (c.CountryCode=ct.Code)
GROUP BY ct.Name,c.District WITH ROLLUP;

SELECT ct.Region , c.Name, AVG(c.Population)
FROM city c INNER JOIN country ct ON (c.CountryCode=ct.Code)
GROUP BY ct.Region,c.Name WITH ROLLUP;

SELECT ct.Region , c.Name, c.District, AVG(c.Population)
FROM city c INNER JOIN country ct ON (c.CountryCode=ct.Code)
GROUP BY ct.Region,c.Name, c.District WITH ROLLUP;
```

Rollup, Cube e Grouping Sets

```
SELECT d.dept_name, ROUND((AVG(salary)/12),2) as "Média de salário mensal"
FROM dept_emp de INNER JOIN employees e   ON (de.emp_no=e.emp_no)
                 INNER JOIN departments d ON (de.dept_no=d.dept_no)
                 INNER JOIN salaries    s ON (s.emp_no=e.emp_no)
WHERE de.to_date = '9999-01-01'
AND    s.to_date = '9999-01-01'
GROUP BY d.dept_name WITH ROLLUP
ORDER BY d.dept_name;
```

Rollup, Cube e Grouping Sets

```
SELECT d.dept_name, t.title, count(de.emp_no) as "Qtde de funcionários com este cargo",
       ROUND((AVG(salary)/12),2) as "Média de salário mensal",
       ROUND((MAX(salary)/12),2) as "Maior salário mensal do departamento",
       ROUND((MIN(salary)/12),2) as "Menor salário mensal do departamento"
FROM dept_emp de INNER JOIN employees e   ON (de.emp_no=e.emp_no)
                 INNER JOIN departments d ON (de.dept_no=d.dept_no)
                 INNER JOIN salaries    s ON (s.emp_no=e.emp_no)
                 INNER JOIN titles      t ON (e.emp_no=t.emp_no)
WHERE de.to_date = '9999-01-01'
AND    s.to_date = '9999-01-01'
GROUP BY d.dept_name, t.title WITH ROLLUP
ORDER BY d.dept_name;
```

Rollup, Cube e Grouping Sets

- Assim como ROLLUP, CUBE é uma extensão da cláusula *GROUP BY* que produz relatórios com tabulação cruzada com aplicação em todas as funções de grupo (agregadas), incluindo AVG, SUM, MIN, MAX e COUNT. O MySQL não oferece este recurso!
- Toda combinação possível de linhas é agregada por CUBE houver n colunas na cláusula *GROUP BY*, haverá 2n combinações super agregadas possíveis. Estas combinações formam um cubo com n dimensões, e foi assim que o operador recebeu o nome de CUBE

Rollup, Cube e Grouping Sets

- *GROUPING SETS* é outra extensão da cláusula *GROUP BY* que pode ser usado para especificar vários agrupamentos de dados com possibilidade de ter várias cláusulas *GROUP BY* na mesma instrução SELECT, algo que não é permitido na sintaxe normal.
- Em uma situação normal três agrupamentos diferentes geram três cláusulas SQL diferentes com uma única diferença que é a cláusula *GROUP BY.*
- O MySQL não oferece este recurso!

Rollup, Cube e Grouping Sets

- Para melhorar o nosso relatório podemos utilizar a função *GROUPING* que retorna 1 para uma linha agregada e 0 para uma linha não agregada.
- É utilizada na cláusula SELECT e seleciona apenas uma coluna como argumento.
- Geralmente é utilizada com a expressão condicional para melhorar a visibilidade.

Rollup, Cube e Grouping Sets

```
SELECT IF(GROUPING(d.dept_name),"Todos os Departamentos",d.dept_name) as "Departamentos",
       IF(GROUPING(t.title),"Todos os cargos",t.title) as "Cargos",
       count(de.emp_no) as "Qtde de funcionários com este cargo",
       ROUND((AVG(salary)/12),2) as "Média de salário mensal",
       ROUND((MAX(salary)/12),2) as "Maior salário mensal do departamento",
       ROUND((MIN(salary)/12),2) as "Menor salário mensal do departamento"
FROM dept_emp de INNER JOIN employees e   ON (de.emp_no=e.emp_no)
                 INNER JOIN departments d ON (de.dept_no=d.dept_no)
                 INNER JOIN salaries    s ON (s.emp_no=e.emp_no)
                 INNER JOIN titles      t ON (e.emp_no=t.emp_no)
WHERE de.to_date = '9999-01-01'
AND    s.to_date = '9999-01-01'
GROUP BY d.dept_name, t.title  WITH ROLLUP
ORDER BY d.dept_name, Cargos;
```

Rollup, Cube e Grouping Sets

- Os operadores de conjunto são usados para combinar os resultados de instruções SELECT diferentes em uma única saída criando uma única saída de mais de uma tabela.
- Caso junte as tabelas, as linhas que corresponderem aos critérios da junção serão retornados e, além disto, eles conseguem retornar as linhas encontradas em várias instruções SELECT, as linhas que estão em uma tabela, mas não em outra ou ainda as linhas comuns a ambas as instruções.

Operadores de conjunto

- **UNION:** retorna todas as linhas de ambas as tabelas, após eliminar as duplicatas.
- **UNION ALL:** retorna todas as linhas de ambas as tabelas, sem eliminar as duplicatas.
- **INTERSECT:** retorna todas as linhas comuns a ambas as tabelas
- **MINUS:** retorna todas as linhas encontradas em uma tabela, mas não na outra.

Operadores de conjunto

- O *MySQL* não disponibiliza todos os operadores de grupo demonstrados mas é possível representa-los através dos tipos de junções.

```
SELECT emp_no
FROM employees e
UNION ALL
SELECT emp_no
FROM dept_emp;
```

```
SELECT emp_no
FROM employees e
UNION DISTINCT
SELECT emp_no
FROM dept_emp;
```

Referências bibliográficas

ELMASRI, R., NAVATHE, S. B., Sistemas de Banco de Dados: Fundamentos e Aplicações. 3ª Ed., Editora LTC, 2002.

MANNINO, Michael V. Projeto, Desenvolvimento de Aplicações e Administração de Banco de Dados. 3ª. Ed. Porto Alegre. Bookman. 2008.