
ORACLE
Academy

Programação de Banco de Dados com SQL

9-2
Usando as Operações Rollup e Cube e Grouping Sets

ORACLE Academy

Objetivos

Esta lição abrange os seguintes objetivos:

- Usar ROLLUP para produzir valores subtotais
- Usar CUBE para produzir valores de tabulação cruzada
- Usar GROUPING SETS para produzir um único conjunto de resultados
- Usar a função GROUPING para identificar os valores de linha extras criados pela operação ROLLUP ou CUBE

Finalidade

- Vamos desenvolver um pouco mais o problema apresentado na última lição.
- Para encontrar a média de altura de todos os alunos, você usa esta consulta:

```
SELECT AVG(height) FROM students;
```

- Se quiser saber a média de altura dos alunos com base em seu ano na escola, você poderá escrever uma série de instruções SQL diferentes:

```
SELECT AVG(height) FROM students WHERE year_in_school = 10;
SELECT AVG(height) FROM students WHERE year_in_school = 11;
SELECT AVG(height) FROM students WHERE year_in_school = 12;
```

Finalidade

- Ou poderá simplificar o problema gravando uma única instrução que contenha as cláusulas GROUP BY e HAVING.
- E se depois de ter selecionado os grupos e calculado as agregações desses grupos, você também quisesse os subtotais por grupo e o total geral de todas as linhas selecionadas?

Finalidade

- Você poderia importar os resultados para um aplicativo de planilha, pegar sua calculadora ou calcular os totais no papel usando aritmética.
- Algo melhor ainda seria usar algumas das extensões da cláusula GROUP BY criadas especificamente para esse propósito: ROLLUP, CUBE e GROUPING SETS.
- O uso desses extensões exige menos trabalho da sua parte e elas são todas muito eficientes, do ponto de vista do banco de dados.

ROLLUP

- Em consultas GROUP BY, você muitas vezes precisa produzir subtotais e totais, e a operação ROLLUP pode fazer isso.
- Sem usar a operação ROLLUP, esse tipo de necessidade levaria à gravação de várias consultas e, em seguida, à inclusão dos resultados em uma planilha, por exemplo, para calcular e formatar os resultados.
- ROLLUP cria subtotais que fazem roll-up do nível mais detalhado para um total geral, usando a lista de agrupamento especificada na cláusula GROUP BY.

ROLLUP

- A ação de ROLLUP é simples: ele cria subtotais que fazem roll-up do nível mais detalhado para um total geral
- ROLLUP usa uma lista ordenada de colunas de agrupamento em sua lista de argumentos.
- Primeiro, ele calcula os valores agregados padrão especificados na cláusula GROUP BY.
- Em seguida, cria subtotais de nível progressivamente maior, movendo-se da direita para a esquerda na lista de colunas de agrupamento.
- Por fim, ele cria um total geral.

ROLLUP: Tabela de Resultados

- Na tabela de resultados a seguir, as linhas destacadas em vermelho são geradas pela operação ROLLUP:

```
SELECT department_id, job_id, SUM(salary)
FROM employees
WHERE department_id < 50
GROUP BY ROLLUP (department_id, job_id);
```

DEPARTMENT_ID	JOB_ID	SUM(SALARY)	
10	AD_ASST	4400	
10	-	4400	← Subtotal de id_depto 10
20	MK_GER	13000	
20	MK_REP	6000	
20	-	19000	← Subtotal de id_depto 20
-	-	23400	← Total Geral para o relatório

ROLLUP: Fórmula de Resultados

- O número de colunas ou expressões que aparecem na lista de argumentos de ROLLUP determina o número de agrupamentos.
- A fórmula é (número de colunas) + 1, onde o número de colunas é aquele presente na lista de argumentos de ROLLUP.

ROLLUP: Fórmula de Resultados

- Na consulta de exemplo abaixo, duas colunas estão presentes na lista de argumentos de ROLLUP. Portanto, você verá três valores serem gerados automaticamente.

```
SELECT department_id, job_id, SUM(salary)
FROM employees
WHERE department_id < 50
GROUP BY ROLLUP (department_id, job_id);
```

Sem ROLLUP

- Se você usar GROUP BY sem ROLLUP para a mesma consulta, como ficarão os resultados?

```
SELECT department_id, job_id, SUM(salary)
FROM employees
WHERE department_id < 50
GROUP BY (department_id, job_id);
```

DEPARTMENT_ID	JOB_ID	SUM(SALARY)
20	MK_MAN	13000
10	AD_ASST	4400
20	MK_REP	6000

- Você terá que executar várias consultas para obter os subtotais que ROLLUP produz.

CUBE

- Assim como ROLLUP, CUBE é uma extensão da cláusula GROUP BY.
- Ele produz relatórios com tabulação cruzada.
- Ele pode ser aplicado a todas as funções agregadas, incluindo AVG, SUM, MIN, MAX e COUNT.
- É feita uma referência cruzada das colunas listadas na cláusula GROUP BY para criar um superconjunto de grupos.
- As funções agregadas especificadas na lista SELECT são aplicadas ao grupo para criar valores resumidos para as linhas superagregadas adicionais.

CUBE

- Toda combinação possível de linhas é agregada por CUBE.
- Se houver n colunas na cláusula GROUP BY, haverá 2n combinações superagregadas possíveis.
- Matematicamente falando, essas combinações formam um cubo com n dimensões, e foi assim que o operador recebeu o nome de CUBE (CUBO, em inglês).

CUBE

- CUBE é muitas vezes usado em consultas que utilizam colunas de tabelas separadas, em vez de colunas separadas de uma única tabela.
- Imagine, por exemplo, um usuário consultando a tabela de vendas de uma empresa como a AMAZON.COM.
- Um relatório de tabulação cruzada, que é algo que se pede muito comumente, poderia incluir os subtotais de todas as combinações possíveis de vendas em um Mês, Região e Produto.

CUBE

- Na instrução a seguir, as linhas em vermelho são geradas pela operação CUBE:

```
SELECT department_id, job_id, SUM(salary)
FROM employees
WHERE department_id < 50
GROUP BY CUBE (department_id, job_id);
```

Total para o relatório →
Subtotal de MK_GER →
Subtotal de MK_REP →
Subtotal de AD_ASST →
Subtotal de depto 10 →

Subtotal de depto 20 →

DEPARTMENT_ID	JOB_ID	SUM(SALARY)
-	-	23400
-	MK_GER	13000
-	MK_REP	6000
-	AD_ASST	4400
10	-	4400
10	AD_ASST	4400
20	-	19000
20	MK_GER	13000
20	MK_REP	6000

GROUPING SETS

- GROUPING SETS é outra extensão da cláusula GROUP BY.
- Ele é usado para especificar vários agrupamentos de dados.
- Ele oferece a possibilidade de ter várias cláusulas GROUP BY na mesma instrução SELECT, algo que não é permitido na sintaxe normal.

GROUPING SETS

- Se quisesse ver os dados da tabela FUNCIONÁRIOS agrupados por (id_departamento, id_cargo, id_gerente),
- mas também agrupados por (id_departamento, id_gerente)
- e por (id_cargo, id_gerente), você normalmente teria que gravar três instruções SELECT separadas, sendo que a única diferença entre elas seriam as cláusulas GROUP BY.

GROUPING SETS

- Para o banco de dados, isso significaria recuperar os mesmos dados três vezes, e isso poderia causar um grande overhead.
- Imagine se a sua empresa tivesse 3.000.000 de funcionários.
- Você estaria pedindo para o banco de dados recuperar 9 milhões de linhas, em vez de apenas 3 milhões. É uma diferença grande.
- Portanto, GROUPING SETS são muito mais eficientes quando se está escrevendo relatórios complexos.

GROUPING SETS

- Na instrução a seguir, as linhas coloridas são geradas pela operação GROUPING SETS:

```
SELECT department_id, job_id, manager_id, SUM(salary)
FROM employees
WHERE department_id < 50
GROUP BY GROUPING SETS
((job_id, manager_id), (department_id, job_id), (department_id,
manager_id));
```

DEPARTMENT_ID	JOB_ID	MANAGER_ID	SUM(SALARY)
-	MK_GER	100	13000
-	MK_GER	201	6000
-	AD_ASST	101	4400
10	AD_ASST	-	4400
20	MK_GER	-	13000
20	MK_REP	-	6000
10	-	101	19000
20	-	100	13000
20	-	201	6000

Funções GROUPING

- Ao usar ROLLUP ou CUBE para criar relatórios com subtotais, você muitas vezes também precisa ser capaz de distinguir quais linhas na saída retornaram do banco de dados e quais linhas são de subtotais calculados pelas operações de ROLLUP e CUBE.

DEPARTMENT_ID	JOB_ID	SUM(SALARY)
-	-	23400
-	MK_GER	13000
-	MK_REP	6000
-	AD_ASST	4400
10	-	4400
10	AD_ASST	4400
20	-	19000
20	MK_GER	13000
20	MK_REP	6000

Funções GROUPING

- Se observar o relatório à direita, como você será capaz de diferenciar as linhas do banco de dados das linhas calculadas?
- Como saberá a diferença entre um valor nulo armazenado retornado pela consulta e os valores nulos criados por ROLLUP ou CUBE?

DEPARTMENT_ID	JOB_ID	SUM(SALARY)
-	-	23400
-	MK_GER	13000
-	MK_REP	6000
-	AD_ASST	4400
10	-	4400
10	AD_ASST	4400
20	-	19000
20	MK_GER	13000
20	MK_REP	6000

Funções GROUPING

- A função GROUPING lida com esses problemas.
- Usando uma coluna da consulta como argumento, a função GROUPING retornará 1 para uma linha agregada (calculada) e 0 para uma linha não agregada (retornada).
- A sintaxe dessa função é simplesmente GROUPING (nome_coluna).
- Ela é usada somente na cláusula SELECT e seleciona apenas uma expressão de coluna como argumento.

Funções GROUPING

- Exemplo:

```
SELECT department_id, job_id, SUM(salary),
 GROUPING(department_id) AS "Dept sub total",
 GROUPING(job_id) AS "Job sub total"
FROM employees
WHERE department_id < 50
GROUP BY CUBE (department_id, job_id);
```

DEPARTMENT_ID	JOB_ID	SUM(SALARY)	Dept sub total	Job sub total
-	-	23400	1	1
-	MK_GER	13000	1	0
-	MK_REP	6000	1	0
-	AD_ASST	4400	1	0
10	-	4400	0	1
10	AD_ASST	4400	0	0
20	-	19000	0	1
20	MK_GER	13000	0	0
20	MK_REP	6000	0	0

Terminologia

Estes são os principais termos usados nesta lição:

- CUBE
- Função GROUPING
- GROUPING SETS
- ROLLUP

Resumo

Nesta lição, você deverá ter aprendido a:

- Usar ROLLUP para produzir valores subtotais
- Usar CUBE para produzir valores de tabulação cruzada
- Usar GROUPING SETS para produzir um único conjunto de resultados
- Usar a função GROUPING para identificar os valores de linha extras criados pela operação ROLLUP ou CUBE


ORACLE
Academy