

SQL Analítico - Grouping Sets, Rollup e Cube

Laboratorio de Bases de Datos

Neste boletín preséntanse un conxunto de exercicios que se deberán resolver mediante os conceptos vistos de SQL Analítico, máis concretamente, empregando *GROUPING SETS*, *ROLLUP* e *CUBE*.

1. GROUPING SETS

1. Executa a sentenza que permita obter os seguintes valores da táboa informe (Figura 1).

JOB	DEPTNO			Total
	10	20	30	
CLERK				4
ANALYST				2
MANAGER				3
SALESMAN				4
PRESIDENT				1
	3	5	6	

Figura 1: Táboa informe 1.

Solución:

```
01 | SELECT job, deptno, COUNT(*)
02 | FROM emp
03 | GROUP BY GROUPING SETS (job, deptno);
```

Agrúpase pola columna **job** e pola columna **deptno**.

2. Deseña unha consulta que devolva o número de empregados en cada traballo (*job*) de cada departamento (*deptno*) e o número de empregados que ten ao cargo cada xefe (*mgr*).

Solución:

```
01 | SELECT job, deptno, mgr, COUNT(*)
02 | FROM emp
03 | GROUP BY GROUPING SETS ((job, deptno), mgr);
```

3. Para a seguinte consulta:

```
01 | SELECT job, deptno, mgr, sal, COUNT(*)
02 | FROM emp
03 | GROUP BY (mgr, sal), GROUPING SETS (job, deptno);
```

Indica a súa equivalencia:

- Usando só *GROUP BY* (en varias consultas, non é necesario usar *UNION*).
- Usando unicamente *GROUPING SETS* (non pode ter valores fóra do *GROUPING SETS*).

Solución: Para obter o mesmo resultado usando só *GROUP BY* é necesario tres consultas:

```
01 | SELECT job, mgr, COUNT(*)
02 | FROM emp
03 | GROUP BY (job, mgr, sal);
```

```
01 | SELECT deptno, mgr, COUNT(*)
02 | FROM emp
03 | GROUP BY (deptno, mgr, sal);
```

Usando só un *GROUPING SETS*:

```
01 | SELECT job, deptno, mgr, COUNT(*)
02 | FROM emp
03 | GROUP BY GROUPING SETS ((job, mgr, sal), (deptno, mgr, sal));
```

4. Deseña unha consulta que devolva o número de empregados en cada departamento (*deptno*) e o número total de empregados.

Solución:

```
01 | SELECT deptno, COUNT(*)
02 | FROM emp
03 | GROUP BY GROUPING SETS (deptno, ());
```

2. ROLLUP e CUBE

5. Deseña as consultas que devolvan o seguinte resultado dado:

JOB	DEPTNO	MGR	COUNT(*)
CLERK	10	7782	1
CLERK	10		1
CLERK	20	7788	1
CLERK	20	7902	1
CLERK	20		2
CLERK	30	7698	1
CLERK	30		1
CLERK			4
ANALYST	20	7566	2
ANALYST	20		2
ANALYST			2
MANAGER	10	7839	1
MANAGER	10		1
MANAGER	20	7839	1
MANAGER	20		1
MANAGER	30	7839	1
MANAGER	30		1
MANAGER			3
SALESMAN	30	7698	4
SALESMAN	30		4
SALESMAN			4
PRESIDENT	10		1
PRESIDENT	10		1
PRESIDENT			1
			14

Figura 2: ROLLUP e CUBE - Resultado da consulta.

- Utilizando *ROLLUP* e/ou *CUBE*.
- A súa equivalencia usando só *GROUPING SETS*.

Solución: Utilizando *ROLLUP*:

```
01 | SELECT job, deptno, mgr, COUNT(*)
02 | FROM emp
03 | GROUP BY ROLLUP (job, deptno, mgr);
```

Utilizando *GROUPING SETS*:

```
01 | SELECT job, deptno, mgr, COUNT(*)
02 | FROM emp
03 | GROUP BY GROUPING SETS ((job, deptno, mgr), (job, deptno), job
    , ());
```

6. Deseña as consultas que devolvan o seguinte resultado dado:

- Utilizando *ROLLUP* e/ou *CUBE*.
- A súa equivalencia usando só *GROUPING SETS*.

Solución:

Utilizando *CUBE*:

```
01 | SELECT job, mgr, COUNT(*)
02 | FROM emp
03 | GROUP BY CUBE (job, mgr);
```

JOB	MGR	COUNT(*)
		1
		14
	7566	2
	7698	5
	7782	1
	7788	1
	7839	3
	7902	1
CLERK		4
CLERK	7698	1
CLERK	7782	1
CLERK	7788	1
CLERK	7902	1
ANALYST		2
ANALYST	7566	2
MANAGER		3
MANAGER	7839	3
SALESMAN		4
SALESMAN	7698	4
PRESIDENT		1
PRESIDENT		1

Figura 3: ROLLUP e CUBE - Resultado da consulta 2.

Utilizando *GROUPING SETS*:

```
01 | SELECT job, mgr, COUNT(*)
02 | FROM emp
03 | GROUP BY GROUPING SETS ((job, mgr), job, mgr, ());
```

3. Utilizando o almacén de datos sobre vendas

Nos seguintes exercicios utilizaranse un almacén de datos que representa as vendas de produtos. Esta base de datos está composta polas seguintes táboas:

- Táboa de feitos
 - **dwsales**: Por cada venda realizada, almacena a cantidade de produtos vendidos (*quantity_sold*) e o importe total (*amount_sold*). Estas dúas columnas representan os feitos.
- Táboas de dimensións
 - **dwchannels**: As diferentes canles polas que se poden realizar as vendas.
 - **dwcustomers**: Información sobre cada un dos clientes da tenda.
 - **dwproducts**: Información sobre os produtos que están á venda na tenda, como a descrición, o prezo de venda, etc. Os produtos clasifícanse en categorías e subcategorías
 - **dwtimes**: Dimensión temporal por días, meses, anos, etc.
 - **dwpromotions**: Promocións que se poden aplicar a unha venda. A promoción con identificador 999 indica que non se aplicou ningún tipo de promoción.
 - **dwcountries**: Dimensión de localización. Esta utilízase para asignar un país ós clientes (táboa *dwcustomers*).

Antes de comezar cos exercicios recoméndase consultar cada táboa para coñecer as columnas polas que están formadas e entender o que se está a representar en cada unha.

7. Mostra a media da cantidade vendida e o importe total vendido por cada produto, por cada canle, e para tódolos produtos. Ordena o resultado polo identificador do produto e pola canle.

Solución:

```
01 | SELECT prod_id, channel_id,  
02 |       AVG(quantity_sold), SUM(amount_sold)  
03 | FROM   dwsales  
04 | GROUP BY GROUPING SETS(prod_id, channel_id, ())  
05 | ORDER BY prod_id, channel_id
```

8. Obtén o mínimo, o máximo, a media e a suma total do importe das vendas por día (1998-01-01, 1998-01-02, etc), por día do mes (1, 2, 3, ..., 31), por cada mes (1, 2, 3, 4, ..., 12) e por cada trimestre (*quarter*) (1, 2, 3 e 4). Só se terán en conta as vendas feitas os 6 primeiros meses do ano.

Solución:

```

01 | SELECT    s.time_id AS "Dia",
02 |           day_number_in_month AS "Dia do mes",
03 |           calendar_month_number AS "Mes",
04 |           calendar_quarter_number AS "Trimestre",
05 |           min(amount_sold) AS "Min", max(amount_sold) AS "Max",
06 |           avg(amount_sold) AS "Avg", SUM(amount_sold) AS "Tot."
07 | FROM      dwsales s JOIN dwtimes t
08 |           ON s.time_id = t.time_id
09 | WHERE     calendar_month_number BETWEEN 1 AND 6
10 | GROUP BY GROUPING SETS(s.time_id,
11 |                         day_number_in_month,
12 |                         calendar_month_number,
13 |                         calendar_quarter_number)
14 | ORDER BY s.time_id, day_number_in_month,
15 |          calendar_month_number, calendar_quarter_number

```

9. Quérese analizar en que días se compra e se gasta máis por **internet** (canle con identificador 4). Obtén o número de vendas e o importe gastado por cada día da semana (Monday, Tuesday, Wednesday, etc.), por cada día do mes (1, 2, ..., 30, 31.), e por cada día da semana e día do mes (Monday 1, Monday 2, ..., Sunday 1, Sunday 2, etc.). Mostra tamén o número de vendas e o importe total gastado. Ordena o resultado polo importe total, o máis alto primeiro.

Solución:

```

01 | SELECT    day_name AS "Dia semana",
02 |           day_number_in_month AS "Dia mes",
03 |           COUNT(*) AS "N vendas",
04 |           SUM(amount_sold) AS "Importe total"
05 | FROM      dwsales s JOIN dwtimes t
06 |           ON s.time_id = t.time_id
07 | WHERE     channel_id = 4
08 | GROUP BY CUBE(day_name, day_number_in_month)
09 | ORDER BY SUM(amount_sold) DESC

```

10. Modifica a consulta anterior pero agora o resultado debe estar ordenado por día da semana (Monday, Tuesday, Wednesday, etc) e despois por día do mes.

Solución:

```

01 | SELECT    day_name AS "Dia semana",
02 |           day_number_in_month AS "Dia mes",
03 |           COUNT(*) AS "N vendas",
04 |           SUM(amount_sold) AS "Importe total"
05 | FROM      dwsales s JOIN dwtimes t
06 |           ON s.time_id = t.time_id
07 | WHERE     channel_id = 4
08 | GROUP BY CUBE((day_name, day_number_in_week),
09 |              day_number_in_month)
09 | ORDER BY day_number_in_week, day_number_in_month

```

11. Obtén a cantidade mínima, máxima e a media das vendas para cada categoría (do produto), cada sub-categoría (do produto) e para tódolos produtos. Mostrar só cando o importe total (dunha categoría ou subcategoría)

sexa igual ou maior a 1.000.000. Ordena de tal forma que as subcategorías dunha categoría aparezan xuntas.

Solución:

```
01 | SELECT    prod_category, prod_subcategory,
02 |           min(quantity_sold) AS "min",
03 |           max(quantity_sold) AS "max",
04 |           avg(quantity_sold) AS "avg"
05 | FROM      dwsales s JOIN dwproducts t
06 |           ON s.prod_id = t.prod_id
07 | GROUP BY  ROLLUP(prod_category, prod_subcategory)
08 | HAVING    SUM(amount_sold) >= 1000000
09 | ORDER BY  prod_category, prod_subcategory
```

12. Para cada cliente, quérese coñecer cal foi o importe total gastado en cada unha das canles e tamén o importe total gastado en cada produto. Só se quere analizar as vendas feitas nos primeiros 15 días de cada mes. O resultado final debe estar ordenado por cliente, produto e canle.

Solución:

```
01 | SELECT    cust_id, prod_id, channel_id, SUM(amount_sold)
02 | FROM      dwsales s JOIN dwtimes t
03 |           ON s.time_id = t.time_id
04 | WHERE     day_number_in_month BETWEEN 1 AND 15
05 | GROUP BY  cust_id, GROUPING SETS(prod_id, channel_id)
06 | ORDER BY  cust_id, prod_id, channel_id
```

13. Quérese analizar o uso das distintas promocións ao longo do tempo na venda dos produtos. Para iso, mostra para cada promoción, o número total de unidades vendidas, o desglose por ano e por mes e ano. Ordena o resultado polo identificador da promoción, ano e mes. *Nota: NON se quere mostrar o número total de unidades vendidas por tódolos produtos.*

Solución:

```
01 | SELECT    promo_id,
02 |           t.calendar_year AS ano,
03 |           t.calendar_month_number AS mes,
04 |           SUM(s.quantity_sold) AS unidades
05 | FROM      dwsales s JOIN dwtimes t
06 |           ON s.time_id = t.time_id
07 | GROUP BY  promo_id, ROLLUP(t.calendar_year, t.
08 |           calendar_month_number)
08 | ORDER BY  promo_id, t.calendar_year, t.calendar_month_number
```

14. Modifica a consulta anterior para que mostre o nome do mes (January, February, March, etc.) e non so o número do mes. Debe seguir respectando a orde, e dicir, primeiro mostrará os do mes de xaneiro, despois febreiro, etc.

Solución:

```

01 | SELECT promo_id,
02 |         t.calendar_year AS ano,
03 |         t.calendar_month_name AS mes,
04 |         SUM(s.quantity_sold) AS unidades
05 | FROM   dwsales s JOIN dwtimes t
06 |         ON s.time_id = t.time_id
07 | GROUP BY promo_id, ROLLUP(t.calendar_year,
08 |                          (t.calendar_month_number, t.calendar_month_name))
09 | ORDER BY promo_id, t.calendar_year, t.calendar_month_number
10 |

```

15. Para coñecer o éxito (ou fracaso) das ofertas ofrecidas pola tenda, quérese analizar a tendencia de uso de promocións na venda dos produtos ao longo do tempo. Para iso, diseña unha consulta que devolva o número de vendas nas que se usou algunha promoción por cada ano. Ademais, os datos anuais deben desglósase en mensuais e, á súa vez, estes en días do mes. Ordenar os datos para que se mostren os máis antigos primeiro.

Solución:

```

01 | SELECT calendar_year as ano,
02 |         calendar_month_number as mes,
03 |         day_number_in_month as dia,
04 |         COUNT(*) as "Vendas totais"
05 | FROM   dwsales JOIN dwtimes
06 |         ON dwsales.time_id = dwtimes.time_id
07 | WHERE  promo_id <> 999
08 | GROUP BY ROLLUP(calendar_year, calendar_month_number,
09 |                day_number_in_month)
09 | ORDER BY calendar_year, calendar_month_number,
        day_number_in_month

```

16. Modifica a consulta anterior para que mostre a mesma información pero para cada un dos países da rexión de Europa (con *country_region_id* igual a 52803). No resultado debe aparecer o nome do país. Os datos deben ordenarse polo nome do país e despois de forma temporal.

Solución: Podemos supoñer que dous países non se poden chamar igual, noutro caso debería utilizarse tamén *country_id*

```

01 | SELECT country_name as pais,
02 |         calendar_year as ano,
03 |         calendar_month_number as mes,
04 |         day_number_in_month as dia,
05 |         COUNT(*) AS "Vendas totais"
06 | FROM   dwsales JOIN dwtimes
07 |         ON dwsales.time_id = dwtimes.time_id
08 |         JOIN dcustomers
09 |         ON dwsales.cust_id = dcustomers.cust_id
10 |         JOIN dwcountries
11 |         ON dcustomers.country_id = dwcountries.
        country_id
12 | WHERE  promo_id <> 999 AND country_region_id = 52803
13 | GROUP BY country_name,
14 |         ROLLUP(calendar_year, calendar_month_number,
        day_number_in_month)

```



```
15 | ORDER BY country_name, calendar_year,
16 |          calendar_month_number, day_number_in_month
```

17. Modifica a consulta anterior para que nas vendas diarias só mostre as correspondentes aos días 1, 15 e 30 de cada mes, xa que se cre que son os máis importantes. A información de cada mes e ano debe mostrarse como nas consultas anteriores, é dicir, debe ter en conta tódolos días do mes.

Solución:

```
01 | SELECT country_name as pais,
02 |        calendar_year as ano,
03 |        calendar_month_number as mes,
04 |        day_number_in_month as dia,
05 |        COUNT(*) AS "Vendas totais"
06 | FROM   dwsales s JOIN dwtimes t
07 |        ON s.time_id = t.time_id
08 |        JOIN dcustomers c
09 |        ON s.cust_id = c.cust_id
10 |        JOIN dwcountries co
11 |        ON c.country_id = co.country_id
12 | WHERE  promo_id <> 999 AND country_region_id = 52803
13 | GROUP BY country_name,
14 |        ROLLUP(calendar_year, calendar_month_number,
15 |               day_number_in_month)
16 | HAVING day_number_in_month IS NULL
17 |        OR day_number_in_month IN (1, 15 , 30)
18 | ORDER BY country_name, calendar_year,
19 |        calendar_month_number, day_number_in_month
```