



ORACLE

Academy



Database Programming with SQL

9-2

Uso de las Operaciones Rollup y Cube, y Grouping Sets

ORACLE
Academy



Objetivos

- En esta lección se abordan los siguientes objetivos:
 - Utilizar ROLLUP para generar valores subtotales
 - Utilizar CUBE para generar valores de tabulación cruzada
 - Utilizar GROUPING SETS para generar un juego de resultados único
 - Utilizar la función GROUPING para identificar los valores de fila adicionales creados por una operación ROLLUP o CUBE



Objetivo

- Vamos a profundizar un poco más en el problema que se le presentó en la última lección
- Para buscar la altura media de todos los alumnos, utilice esta consulta:

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

- Si desea saber la altura media de los alumnos según sus años en el centro educativo, podría escribir una serie de sentencias SQL distintas, como esta:

```
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;
```

Objetivo

- O bien, podría simplificar el problema mediante la escritura de una sola sentencia que contenga las cláusulas GROUP BY y HAVING
- ¿Qué ocurre si, una vez que haya seleccionado sus grupos y calculado los valores agregados en esos grupos, también deseara los subtotales por grupo y una suma total de todas las filas seleccionadas?

Objetivo

- Podría importar los resultados en una aplicación de hoja de cálculo, sacar la calculadora, o bien, calcular los totales manualmente en papel con la aritmética
- Pero, mejor aún, podría utilizar algunas de las extensiones de la cláusula GROUP BY, creadas específicamente para este fin: ROLLUP, CUBE y GROUPING SETS
- Al utilizar estas extensiones se necesita menos trabajo por su parte. Además, todas ellas tienen un uso muy eficaz, desde el punto de vista de la base de datos

ROLLUP

- En las consultas con GROUP BY, a menudo se deben producir subtotales y totales, y la operación ROLLUP puede realizar esta acción por usted
- Sin utilizar el operador ROLLUP, ese tipo de requisito significaría escribir varias consultas y, a continuación, introducir los resultados, por ejemplo, en una hoja de cálculo para calcular y aplicar formato a los resultados
- ROLLUP crea subtotales que se acumulan desde el nivel más detallado hasta la suma total, siguiendo la lista de agrupamiento especificada en la cláusula GROUP BY

ROLLUP

- La acción de ROLLUP es directa: crea subtotales que se acumulan desde el nivel más detallado hasta la suma total
- ROLLUP utiliza una lista ordenada de las columnas del agrupamiento en su lista de argumentos
- En primer lugar, calcula los valores de agregación estándar especificados en la cláusula GROUP BY
- A continuación, crea subtotales de nivel superior de forma progresiva, de derecha a izquierda a través de la lista de columnas del agrupamiento
- Por último, crea una suma total

Tabla de Resultados de ROLLUP

- En la tabla de resultados siguiente, las filas resaltadas en rojo las generas la operación 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
20	MK_MAN	13000
20	MK_REP	6000
20	-	19000
-	-	23400

← Subtotal de dept_id 10

← Subtotal de dept_id 20

← Suma total del informe

Fórmula de Resultado de ROLLUP

- El número de columnas o expresiones que aparecen en la lista de argumentos de ROLLUP determina el número de agrupamientos
- La fórmula es (número de columnas) + 1, donde el número de columnas es el número de columnas que se indica en la lista de argumentos de ROLLUP



Fórmula de Resultado de ROLLUP

- En la siguiente consulta de ejemplo, se muestran dos columnas en la lista de argumentos de ROLLUP y, por lo tanto, verá que se generan tres valores automáticamente

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

Sin ROLLUP

- Si utiliza GROUP BY sin ROLLUP para la misma consulta, ¿qué aspecto tendría el resultado?

```
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

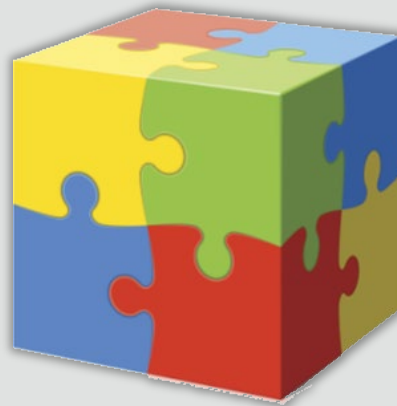
- Tendría que ejecutar varias consultas para obtener los subtotales obtenidos con ROLLUP

CUBE

- CUBE, al igual que ROLLUP, es una extensión de la cláusula GROUP BY
- Genera informes de tabulación cruzada
- Se puede aplicar a todas las funciones de agregación, incluidas AVG, SUM, MIN, MAX y COUNT
- Las columnas que se muestran en la cláusula GROUP BY están incluidas en referencias cruzadas para crear un superjuego de grupos
- Las funciones de agregación especificadas en la lista SELECT se aplican a estos grupos para crear valores de resumen para las filas superagregadas adicionales

CUBE

- Todas las combinaciones posibles de filas se agregan con CUBE
- Si tiene n columnas en la cláusula GROUP BY, habrá 2^n posibles combinaciones de superagregados
- Matemáticamente, estas combinaciones forman un cubo de n dimensiones, que es de donde procede el nombre del operador



CUBE

- CUBE se utiliza a menudo en las consultas que utilizan columnas de tablas independientes, en lugar de distintas columnas de una sola tabla
- Imagine, por ejemplo, un usuario que consulta la tabla Sales para una compañía como AMAZON.COM
- Un informe de tabulación cruzada normalmente solicitado podría incluir subtotales para todas las combinaciones posibles de las ventas de un mes, región y producto

CUBE

- En la siguiente sentencia, las filas en rojo las genera la operación CUBE:

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

Total del informe →

Subtotal de MK_MAN →

Subtotal de MK_REP →

Subtotal de AD_ASST →

Subtotal de departamento 10 →

Subtotal de departamento 20 →

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

GROUPING SETS

- GROUPING SETS es otra extensión de la cláusula GROUP BY
- Se utiliza para especificar varias agrupaciones de datos
- Esto le proporciona la funcionalidad de tener varias cláusulas GROUP BY en la misma sentencia SELECT, lo cual no está permitido en la sintaxis normal



GROUPING SETS

- Si desea ver los datos de la tabla EMPLOYEES agrupados por (department_id, job_id, manager_id)
- Pero también agrupados por (department_id, manager_id)
- Y también agrupados por (job_id, manager_id), normalmente tendría que escribir tres sentencias select distintas siendo la única diferencia entre ellas las cláusulas GROUP BY



GROUPING SETS

- Para la base de datos, esto significa recuperar los mismos datos tres veces distintas, lo que podría suponer una gran sobrecarga
- Imagine que su compañía tiene 3.000.000 de empleados
- Estaría pidiendo a la base de datos que recuperase 9 millones de filas en lugar de solo 3 millones de filas, una gran diferencia
- Por lo tanto, el uso de GROUPING SETS es mucho más eficiente al escribir informes complejos

GROUPING SETS

- En la siguiente sentencia, las filas resaltadas en color las genera la operación 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_MAN	100	13000
-	MK_MAN	201	6000
-	AD_ASST	101	4400
10	AD_ASST	-	4400
20	MK_MAN	-	13000
20	MK_REP	-	6000
10	-	101	19000
20	-	100	13000
20	-	201	6000

Funciones GROUPING

- Al utilizar ROLLUP o CUBE para crear informes con subtotales, muy a menudo también tiene que poder saber qué filas de la salida son filas reales devueltas de la base de datos y qué filas son filas de subtotal calculado resultantes de las operaciones ROLLUP o CUBE

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

Funciones GROUPING

- Si observa el informe de la derecha, ¿cómo podría distinguir entre las filas reales de la base de datos y las filas calculadas?
- ¿Cómo puede notar la diferencia entre un valor NULL almacenado devuelto por la consulta y los valores NULL creados mediante ROLLUP o CUBE

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

Funciones GROUPING

- La función GROUPING resuelve estos problemas
- Utilizando una sola columna de la consulta como argumento, la función GROUPING devolverá un 1 para una fila agregada (calculada) y un 0 para una fila no agregada (devuelta)
- La sintaxis de GROUPING es simplemente GROUPING (nombre_columna)
- Solo se utiliza en la cláusula SELECT y solo acepta una expresión de columna como argumento

Funciones GROUPING

- Ejemplo:

```
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_MAN	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_MAN	13000	0	0
20	MK_REP	6000	0	0

Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
 - CUBE
 - FUNCIÓN GROUPING
 - GROUPING SETS
 - ROLLUP

Resumen

- En esta lección, debe haber aprendido lo siguiente:
 - Utilizar ROLLUP para generar valores subtotales
 - Utilizar CUBE para generar valores de tabulación cruzada
 - Utilizar GROUPING SETS para generar un juego de resultados único
 - Utilizar la función GROUPING para identificar los valores de fila adicionales creados por una operación ROLLUP o CUBE





ORACLE

Academy

