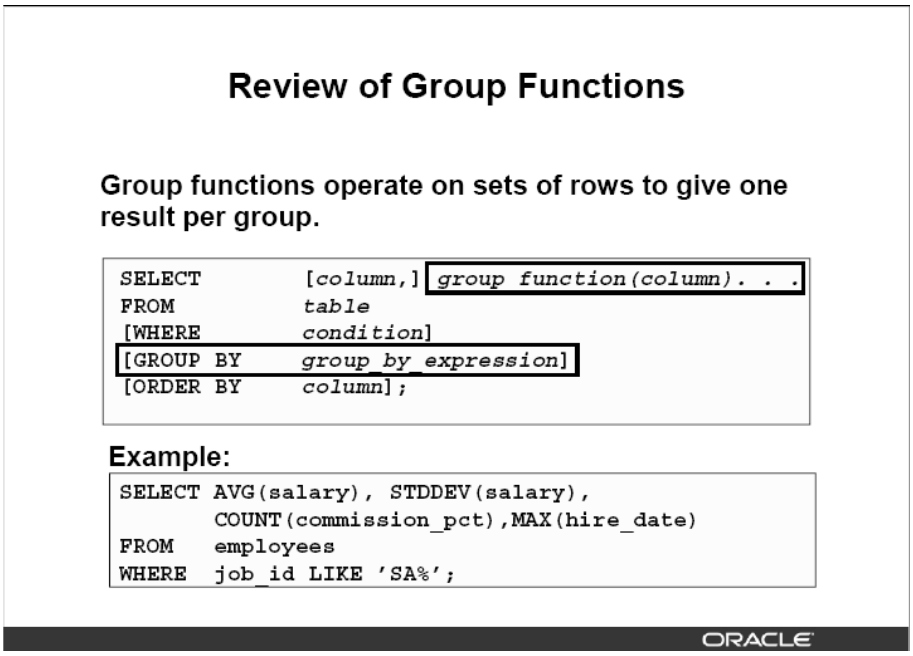
Mejoras para la cláusula GROUP BY

Objetivos

Después de completar este capítulo conocerá el uso de lo siguiente:

- Uso de la operación ROLLUP para obtener subtotales
- Uso de la operación CUBE para obtener operaciones cruzadas
- Uso de la función GROUPING para identificar los valores de las filas creadas por las operaciones ROLLUP y CUBE
- Uso de GROUPING SETS para producir un solo resultado



Funciones de Grupo

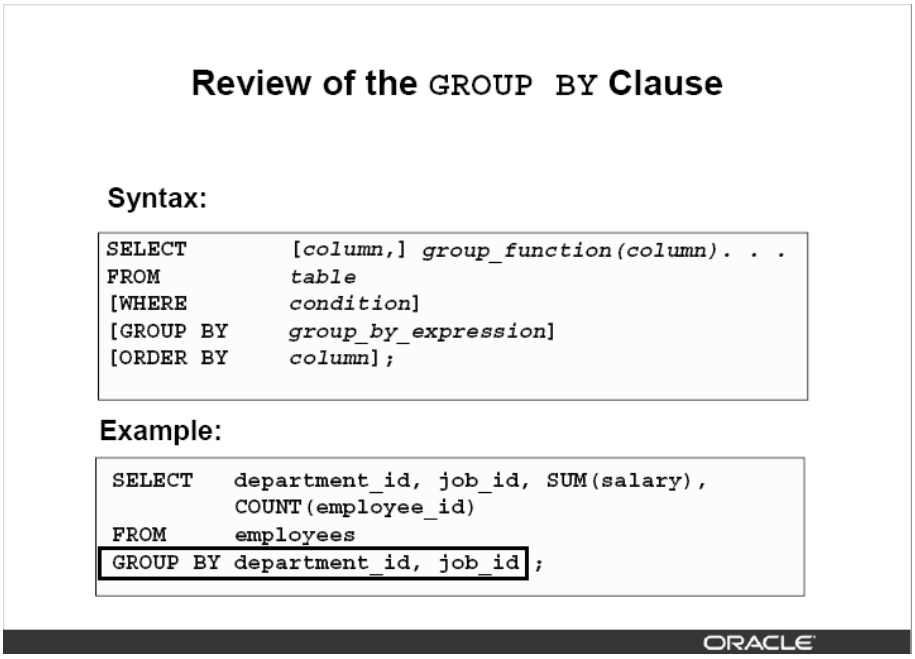
Se puede usar la cláusula GROUP BY para dividir las filas en grupos las filas de una tabla. Se pueden usar las funciones de grupo para obtener información resumida por cada grupo. Las funciones de grupo pueden estar en la selección de la lista y en las cláusulas ORDER BY y HAVING. El servidor de Oracle emplea las funciones de grupo para cada grupo de filas y obtiene un resultado por cada grupo.

Tipos de funciones de grupo

Cada función de grupo AVG, SUM, MAX, MIN, COUNT, STDDEV y VARIANCE aceptan un argumento. Las funciones AVG, SUM, STDDEV y VARIANCE trabajan solo con valores numéricos. MAX y MIN pueden trabajar en números, caracteres y fechas. COUNT obtiene el número de filas no nulas para una expresión. En el ejemplo se calcula el salario promedio, desviación estándar del salario, número de empleados que tienen una comisión y la fecha de contratación máxima para aquellos empleados que su puesto inicie con 'SA'.

Normas para el uso de funciones de grupo

- Los tipos de datos para los argumentos pueden ser CHAR, VARCHAR2, NUMBER o DATE.
- Todas las funciones de grupo con excepción de COUNT(*) ignoran valores nulos. Para sustituir un valor de valores nulos, utilice la función NVL. COUNT obtiene algún número o cero.
- El servidor de Oracle implícitamente ordena los resultados ascendentemente de acuerdo a los grupos de las columnas especificadas, cuando se usa la cláusula GROUP BY. Para evitar este ordenamiento por defecto, se puede usar DESC en una cláusula ORDER BY.



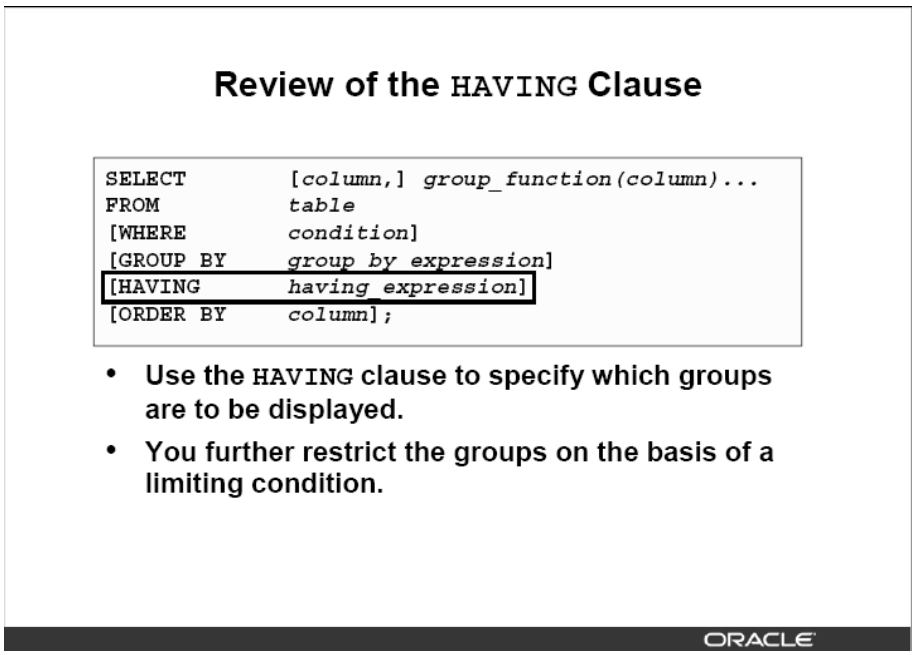
Revisando la cláusula GROUP BY

El ejemplo ilustra como evalúa el servidor de Oracle:

- La cláusula SELECT especifica que las siguientes columnas son recuperadas.
  - Las columnas DEPARTMENT_ID y JOB_ID de la tabla EMPLOYEES
  - La suma de todos los salarios y el número de empleados en cada grupo que se tiene especificado en la cláusula GROUP BY

- La cláusula GROUP BY especifica como las filas serán agrupadas. El salario total y el número de empleados es calculado por cada puesto de cada departamento. Las filas son agrupadas por departamento y agrupadas por puesto para cada departamento.

DEPARTMENT_ID	JOB_ID	SUM(SALARY)	COUNT(EMPLOYEE_ID)
10	AD_ASST	4400	1
20	MK_MAN	13000	1
20	MK_REP	6000	1
50	ST_CLERK	11700	4
...			
110	AC_ACCOUNT	8300	1
110	AC_MGR	12000	1
	SA_REP	7000	1



La cláusula HAVING

Los grupos formados y las funciones de grupo son calculados antes de que la cláusula HAVING sea aplicada a los grupos. La cláusula HAVING puede estar antes de la cláusula GROUP BY, pero es recomendado poner primero la cláusula GROUP BY puesto que es más lógico.
El servidor de Oracle ejecuta los siguientes pasos cuando se usa la cláusula HAVING:

1. Agrupa las filas
2. Aplica las funciones de grupo a los grupos y despliega los grupos que corresponden al criterio de la cláusula HAVING

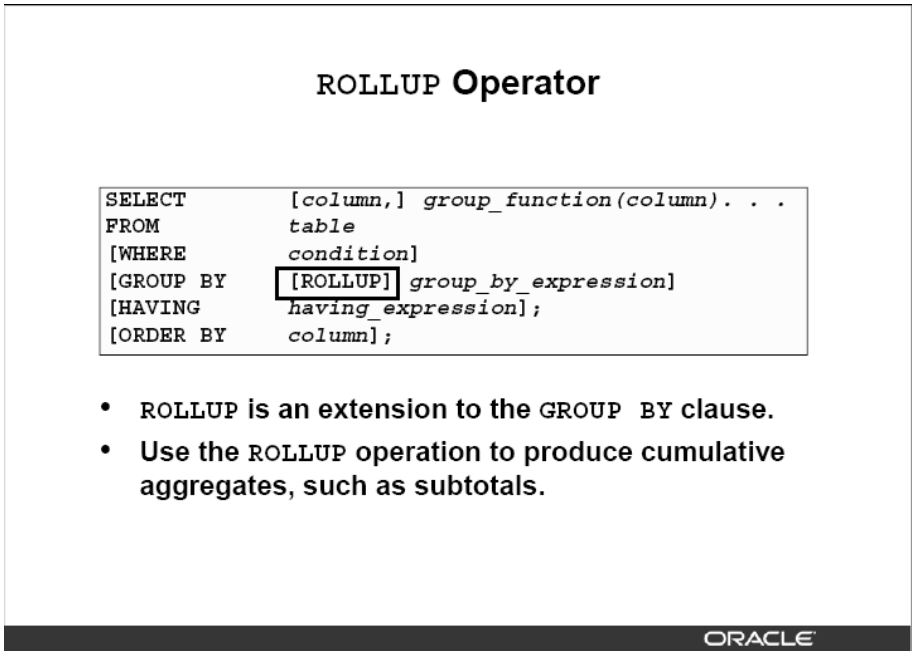
```
SELECT department_id, AVG(salary)
FROM   employees
GROUP  BY  department_id
HAVING AVG(salary) >9500;
```

DEPARTMENT_ID	AVG(SALARY)
80	10033.3333
90	19333.3333
110	10150

Agrupando con los operadores ROLLUP y CUBE

Se puede especificar los operadores ROLLUP y CUBE en la cláusula GROUP BY de una consulta. Agrupando con ROLLUP se produce un conjunto de resultados conteniendo la agrupación de filas y un subtotal de las mismas. La operación CUBE en la cláusula GROUP BY agrupa las filas seleccionadas con base en los valores de todas las posibles combinaciones de expresiones en su especificación y obtiene una fila con información del resumen por cada grupo. Se puede usar el operador CUBE para producir tabulaciones cruzadas de filas.

Nota: Cuando se trabaja con ROLLUP y CUBE, asegúrese de que las columnas seguidas de la cláusula GROUP BY tengan significado, una correlación lógica con las otras; de otra manera los operadores obtienen información irrelevante.
Los operadores ROLLUP y CUBE están disponibles solo en versiones de Oracle8*i* y posteriores.



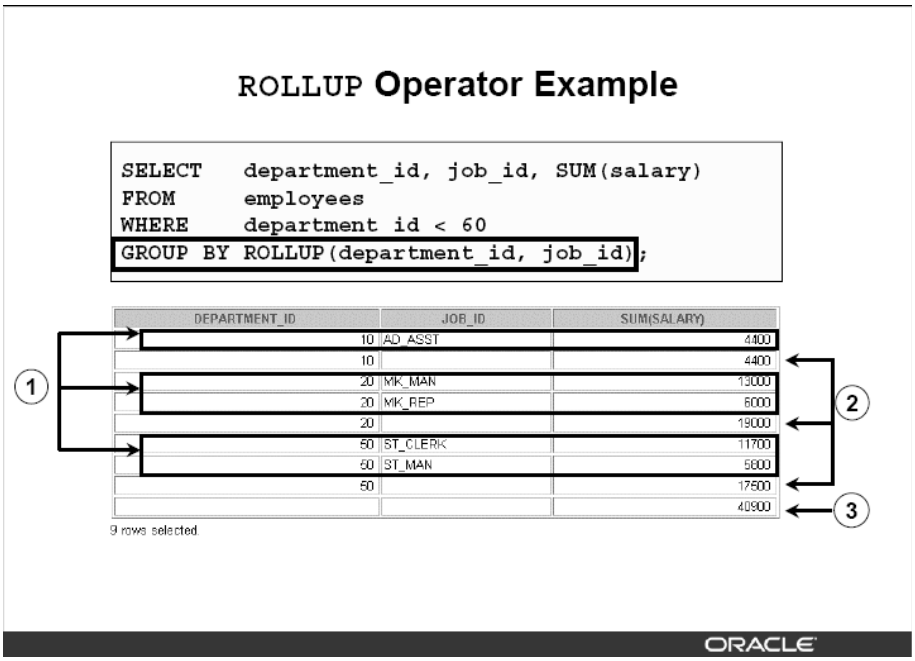
Operador ROLLUP

El operador ROLLUP entrega totales y subtotales para expresiones con una sentencia GROUP BY. El operador ROLLUP puede ser usado por reportes para extraer información estadísticas y resúmenes del conjunto de resultados. Los conjuntos acumulados pueden ser usados en reportes y gráficas.

El operador ROLLUP crea agrupaciones en una sola dirección de derecha a izquierda, a través de la lista de columnas especificadas en la cláusula GROUP BY. Si es utilizada la función acumula esas agrupaciones.

Nota: Para producir subtotales en *n* dimensiones (que es, *n* columnas en la cláusula GROUP BY) sin un operador ROLLUP, *n + 1* sentencias SELECT deben ser ligadas con la cláusula UNION ALL. Esto hace que la ejecución de la consulta sea ineficiente, puesto que cada sentencia SELECT causa un acceso a la tabla. El operador ROLLUP recoge este resultado con un solo acceso a la

tabla. El operador ROLLUP es útil si hay varias columnas involucradas en la generación de los subtotales.

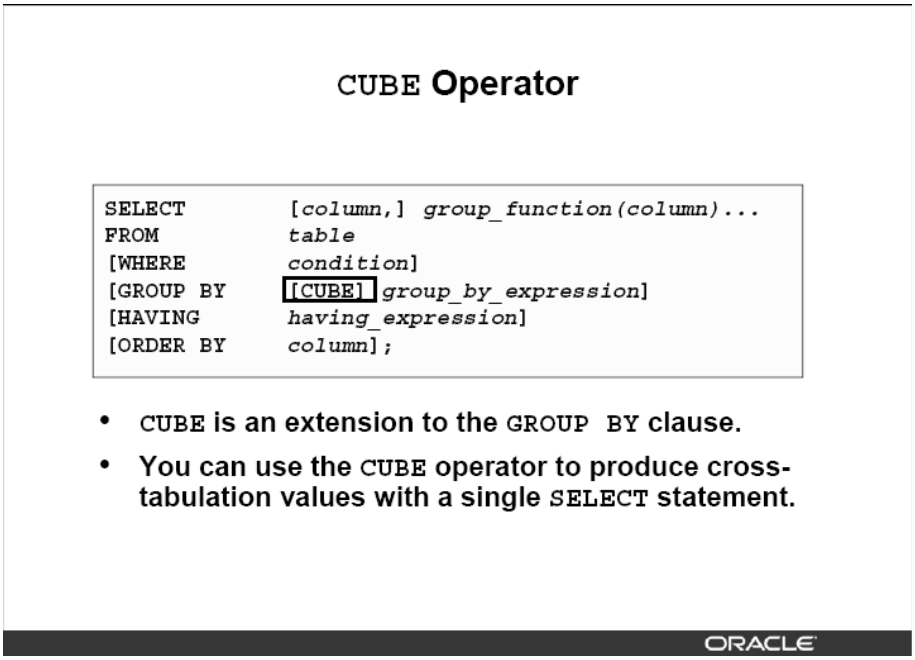


En el ejemplo anterior:

- El salario total por cada puesto en un departamento cuyo identificador del departamento es menor que 60 es desplegado por la cláusula GROUP BY (***Etiqueta 1***)
- El operador ROLLUP despliega:
  - El salario total para aquellos departamentos cuyo DEPARTMENT_ID es menor a 60 (***Etiqueta 2***)
  - El salario total para todos los departamentos cuyo DEPARTMENT_ID es menor que 60 sin tomar en cuenta el JOB_ID (***Etiqueta 3***)
- Todas las filas indicadas como **1** son filas regulares y todas las filas indicadas como **2** y **3** son filas totalizadas.

El operador ROLLUP crea subtotales que van desde el nivel mas detallado hasta un gran total, siguiendo la lista de agrupación especificada en la cláusula GROUP BY. Primero se calculan los valores de los grupos especificados en la cláusula GROUP BY (en el ejemplo, la suma de salarios agrupados en cada puesto dentro de un departamento). Entonces se crea progresivamente subtotales de alto nivel, moviéndose de derecha a izquierda a través de la lista de columnas agrupadas. (En el ejemplo anterior, la suma de salarios por cada departamento es calculado seguido de la suma de salarios para todos los departamentos)

- Proporcionando *n* expresiones en el operador ROLLUP de la cláusula GROUP BY, la operación obtiene $n + 1 = 2 + 1 = 3$ agrupaciones.
- Las filas basadas en los valores de las primeras *n* expresiones son llamadas filas regulares y las otras son llamadas filas totalizadas.

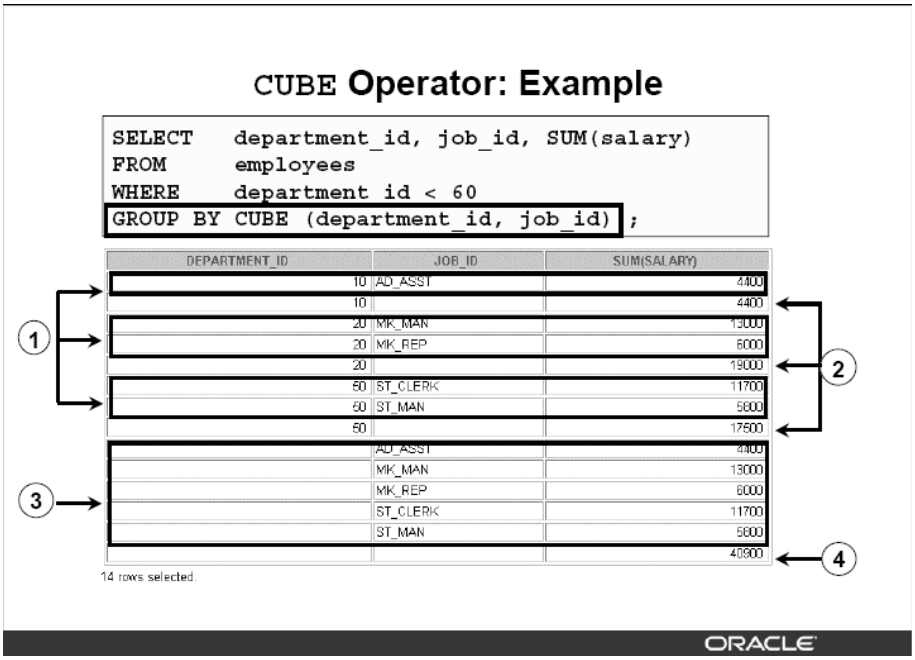


Operador CUBE

El operador CUBE es un interruptor adicional de la cláusula GROUP BY en una sentencia SELECT. El operador CUBE puede ser aplicado a todas las funciones de grupo, incluyendo AVG, SUM, MAX, MIN y COUNT. Esta es usada para producir un conjunto de resultados que típicamente son utilizados para reportes cruzados. Mientras ROLLUP produce solo una parte de posibles subtotales, CUBE produce subtotales para todas las posibles combinaciones de agrupaciones especificadas en la cláusula GROUP BY y un gran total.
El operador CUBE es usado con una función de grupo para generar filas adicionales en un conjunto de resultados. Las columnas incluidas en la cláusula GROUP BY son una referencia cruzada para producir un súper conjunto de grupos. Las funciones de grupo especificadas en la lista seleccionada son aplicadas a esos grupos para producir valores resumidos para las filas agregadas. El número de grupos extra en el conjunto de resultados es determinado por el número de columnas incluidas en la cláusula GROUP BY.

En efecto, cada posible combinación de columnas o expresiones en la cláusula GROUP BY es usada para producir nuevos grupos. Si tienes *n* columnas o expresiones en una cláusula GROUP BY, puedes tener 2^n posibles combinaciones de nuevas columnas. Matemáticamente, estas combinaciones forman un cubo de *n* dimensiones, por lo que de allí proviene su nombre.

Para usar aplicaciones o herramientas de programación, estos valores agregados pueden ser provistos en gráficas y diagramas que conduzcan los resultados y relaciones visualmente y efectivamente.



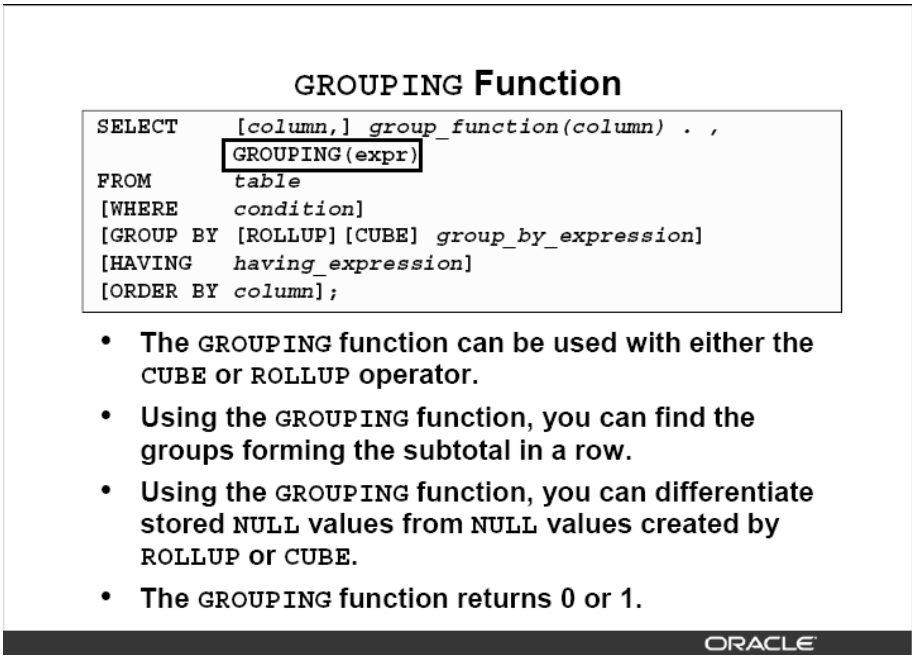
El resultado de la sentencia SELECT del ejemplo puede ser interpretado como sigue:

- El salario total para cada puesto en un departamento (para aquellos departamentos cuyo DEPARTMENT_ID sea menor a 60) es desplegado por la cláusula GROUP BY (***Etiqueta 1***)
- El salario total para aquellos departamentos cuyo DEPARTMENT_ID es menor que 60 (***Etiqueta 2***)
- El salario total para cada puesto sin tomar en cuenta el departamento (***Etiqueta 3***)
- El salario total para aquellos departamentos cuyo DEPARTMENT_ID es menor a 60 independientemente del puesto (***Etiqueta 4***)

En el ejemplo anterior, todas las filas indicadas con 1 son filas regulares, todas las filas indicadas con 2 y 4 son filas totalizadas, y todas las filas indicadas con 3 son valores cruzados.

El operador CUBE tiene también un buen desempeño como el operador ROLLUP para desplegar subtotales para aquellos departamentos cuyo DEPARTMENT_ID es menor a 60 y el salario total para aquellos departamentos cuyo DEPARTMENT_ID es menor que 60, independientemente de los puestos. Adicionalmente, el operador CUBE muestra el salario total para cada puesto independientemente del departamento.

Nota: Similar al operador ROLLUP, produciendo subtotales en *n* dimensiones (que es *n* columnas en la cláusula GROUP BY) sin un operador CUBE se requiere 2^n sentencias SELECT para ser relacionadas con UNION ALL. Así, un reporte con tres dimensiones requiere $2^3 = 8$ sentencias SELECT unidas con UNION ALL.



Funciones de Agrupación

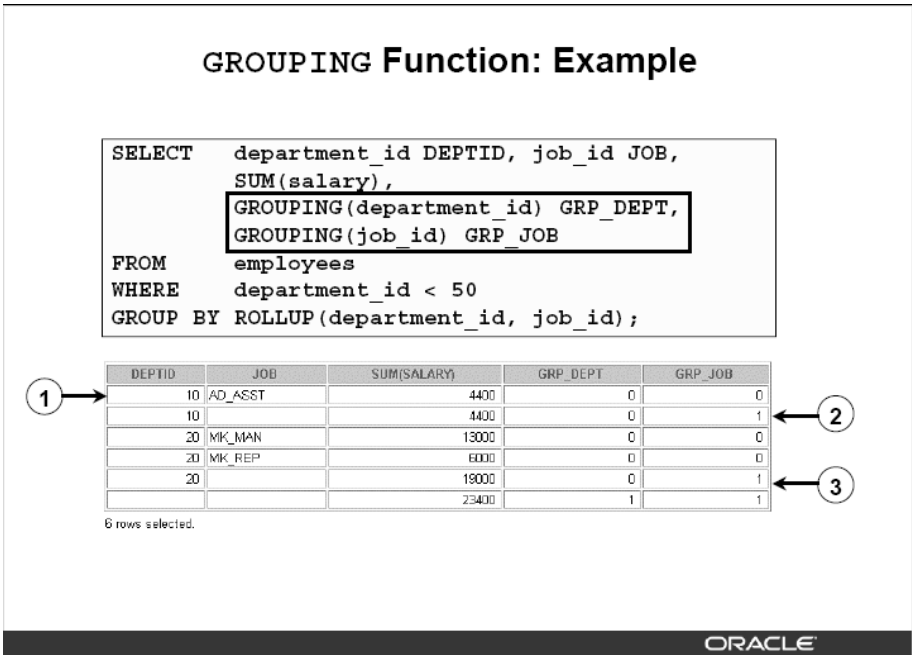
Las funciones de agrupación pueden ser usadas con ambos operadores CUBE y ROLLUP para ayudar a entender como un valor resumido ha sido obtenido.
Las funciones de agrupación usan una columna como argumento. La *expr* en una función *GROUPING* debe corresponder a una de las expresiones en la cláusula GROUP BY. La función obtiene un valor de 0 o 1.

Los valores obtenidos por la función de agrupación son usados para:

- Determinar el nivel de agregación de determinados subtotales; esto es, el grupo o grupos en el cual el subtotal es basado.
- Identificar si un valor nulo en la expresión de una fila de resultados indica:
  - Un valor nulo de la tabla base (un valor nulo almacenado)
  - Un valor nulo creado por ROLLUP/CUBE (es un resultado de una función de grupo o de una expresión)

Un valor obtenido de 0 por la función de agrupación basado en una expresión indica uno de los siguientes puntos:

- La expresión no ha sido usada para calcular un valor de grupo
- El valor nulo en la expresión es creado por ROLLUP o CUBE como resultado de la agrupación.



Ejemplo de funciones de agrupación

En el ejemplo anterior, se considera el valor 4400 en la primera fila (***Etiqueta 1***). Este valor es el salario total para el JOB_ID de "*AD_ASST*" en el departamento 10. Para calcular este valor, ambas columnas DEPARTMENT_ID y JOB_ID tiene que ser tomadas dentro de la cuenta. De esta manera un valor de 0 es obtenido por ambas expresiones *GROUPING(department_id)* y *GROUPING(job_id).*

Considere el valor 4400 en la segunda fila (***Etiqueta 2***). Este valor es el salario total para el departamento 10 y ha sido calculado considerando la columna DEPARTMENT_ID; así un valor de 0 ha sido obtenido por *GROUPING(department_id)*. Puesto que la columna JOB_ID no ha sido tomada en cuenta para calcular este valor, un valor de 1 ha sido obtenido por *GROUPING(job_id).* Se puede observar un resultado similar en la quinta fila.

En la última fila, considere el valor 23400. Este es el salario total para aquellos departamentos cuyo DEPARTMENT_ID es menor que 50 y todos los puestos. Para calcular este valor, ninguna de las columnas DEPARTMENT_ID y JOB_ID tiene que ser tomadas en cuenta. De esta manera un valor de 1 es obtenido para ambas expresiones *GROUPING(department_id)* y *GROUPING(job_id).*

GROUPING SETS (Conjuntos de agrupaciones)

Los *GROUPING SETS* son otras de las extensiones de la cláusula *GROUP BY* que permiten especificar múltiples agrupaciones de datos. Haciendo más fácil y eficiente las agrupaciones y facilitando el análisis de datos a través de múltiples dimensiones.

Una simple sentencia SELECT puede ahora ser escrita usando GROUPING SETS para especificar varias agrupaciones (se puede también incluir los operadores ROLLUP y CUBE), mejor que utilizar múltiples sentencias SELECT combinadas con el operador UNION ALL. Por ejemplo, se puede ver:

```
SELECT   department_id, job_id, manager_id, AVG(salary)
FROM     employees
GROUP BY GROUPING SETS
((department_id, job_id, manager_id),
(department_id, manager_id),(job_id, manager_id));
```

Esta calcula agrupaciones de tres grupos:

```
(department_id, job_id, manager_id), (department_id, manager_id)
and (job_id, manager_id)
```

Sin estas mejoras de Oracle9*i*, múltiples consultas combinadas conjuntamente con UNION ALL son requeridas para obtener el resultado de la sentencia SELECT anterior. Una consulta múltiple es ineficiente, para ello se requiere de múltiples consultas a la misma tabla.

Compare la sentencia anterior con esta alternativa:

```
SELECT   department_id, job_id, manager_id, AVG(salary)
FROM     employees
GROUP BY CUBE(department_id, job_id, manager_id);
```

La sentencia anterior calcula todas las 8 (2*2*2) agrupaciones, aunque solo los grupos (department_id, job_id, manager_id), (department_id, manager_id) y (job_id, manager_id) son de nuestro interés.

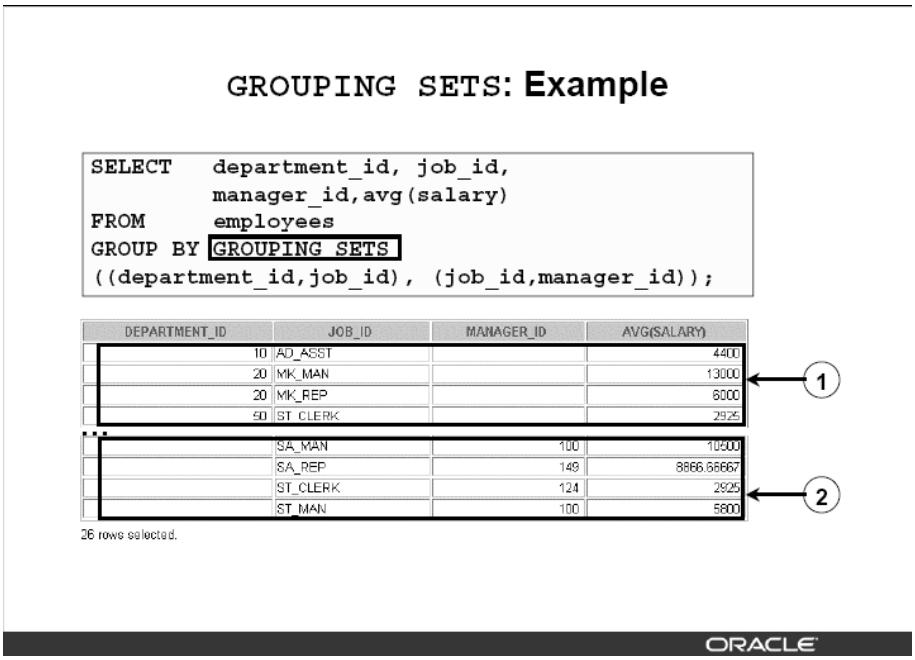
Otra alternativa es la siguiente sentencia:

```
SELECT   department_id, job_id, manager_id, AVG(salary)
FROM     employees
GROUP BY department_id, job_id, manager_id
UNION ALL
SELECT   department_id, NULL, manager_id, AVG(salary)
FROM     employees
GROUP BY department_id, manager_id
UNION ALL
SELECT   NULL, job_id, manager_id, AVG(salary)
FROM     employees
GROUP BY job_id, manager_id;
```

Esta sentencia requiere tres búsquedas a la tabla base, haciéndolo ineficiente.

CUBE y ROLLUP pueden ser si bien, conjuntos de agrupaciones con una semántica específica. Las siguientes equivalencias muestran este efecto.

CUBE (a, b, c) es equivalente a	GROUPING SETS ((a, b, c), (a, b), (a, c), (b, c), (a), (b), (c), ())
ROLLUP(a, b, c) es equivalente a	GROUPING SETS ((a, b, c), (a, b), (a), ())



Ejemplo GROUPING SETS

La consulta en el ejemplo calcula las agrupaciones para dos grupos. La tabla es dividida en los siguientes grupos:

- Department_ID, Job_ID
- Job_ID, Manager_ID

Los salarios promedio para cada uno de esos grupos son calculados. El conjunto de resultados despliega el salario promedio para cada uno de los dos grupos.

En el resultado, el grupo marcado como 1 puede ser interpretado como:

- El salario promedio de todos los empleados con el puesto *AD_ASST* en el departamento 10 es 4400
- El salario promedio de todos los empleados con el puesto *MK_MAN* en el departamento 20 es 13000
- El salario promedio de todos los empleados con el puesto *MK_REP* en el departamento 20 es 6000
- El salario promedio de todos los empleados con el puesto *ST_CLERK* en el departamento 50 es 2925, etc.

El grupo marcado como 2 es interpretado como:

- El salario promedio de todos los empleados con el puesto *MK_REP,* que reportan a su jefe con la clave *210*, es 6000

- El salario promedio de todos los empleados con el puesto *SA_MAN,* que reportan a su jefe con la clave *100*, es 10500, etc.

El ejemplo anterior puede ser escrito también de la siguiente manera:

```
SELECT   department_id, job_id, NULL as manager_id,
         AVG(salary) as AVGSAL
FROM     employees
GROUP BY department_id, job_id
UNION ALL
SELECT   NULL, job_id, manager_id, avg(salary) as AVGSAL
FROM     employees
GROUP BY job_id, manager_id;
```

En ausencia de una optimización que vea a través de bloques de consultas para generar el plan de ejecución, la consulta anterior necesitara dos búsquedas en la tabla base, EMPLOYEES. Esto puede ser muy ineficiente. Por lo tanto el uso de la sentencia GROUPING SETS es recomendado.

Columnas compuestas

Una columna compuesta es una colección de columnas que son tratadas como una unidad durante el cálculo de agrupaciones. Se especifican las columnas en paréntesis como en la siguiente sentencia:

```
ROLLUP (a, (b, c), d)
```

Aquí, (b, c) forman una columna compuesta y son tratadas como una unidad. En general, las columnas compuestas son útiles en ROLLUP, CUBE y GROUPING SETS. Por ejemplo, en CUBE o ROLLUP, la columna compuesta puede tratarse como una agrupación a través de ciertos niveles.

Esto es, `GROUP BY ROLLUP(a, (b, c))`

Es equivalente a

```
GROUP BY a, b, c UNION ALL
GROUP BY a UNION ALL
GROUP BY ()
```

Aquí, (b, c) es tratada como una unidad y ROLLUP puede no ser aplicado a través de (b, c). Esto es como si tuvieras un alias, por ejemplo *z*, para (b, c), y la expresión GROUP BY se reduce a GROUP BY ROLLUP (a, z)

Nota: GROUP BY ( ) es típicamente una sentencia SELECT con valores nulos para las columnas *a* y *b* y únicamente la función de agrupación. Esto es generalmente usado para obtener el G*ran Total*.

```
SELECT   NULL, NULL, aggregate_col
FROM     <table_name>
GROUP BY ( );
```

Compare esta con un ROLLUP normal como:

```
GROUP BY ROLLUP(a, b, c)
```

el cual puede ser

```
GROUP BY a, b, c UNION ALL
GROUP BY a, b UNION ALL
GROUP BY a UNION ALL
GROUP BY ().
```

de forma similar

```
GROUP BY CUBE((a, b), c)
```

Puede ser equivalente a

```
GROUP BY a, b, c UNION ALL
GROUP BY a, b UNION ALL
GROUP BY c UNION ALL
GROUP By ()
```

La siguiente tabla la especificación del conjunto de agrupaciones y su equivalente especificación GROUP BY

GROUPING SETS Statements	**Equivalent GROUP BY Statements**

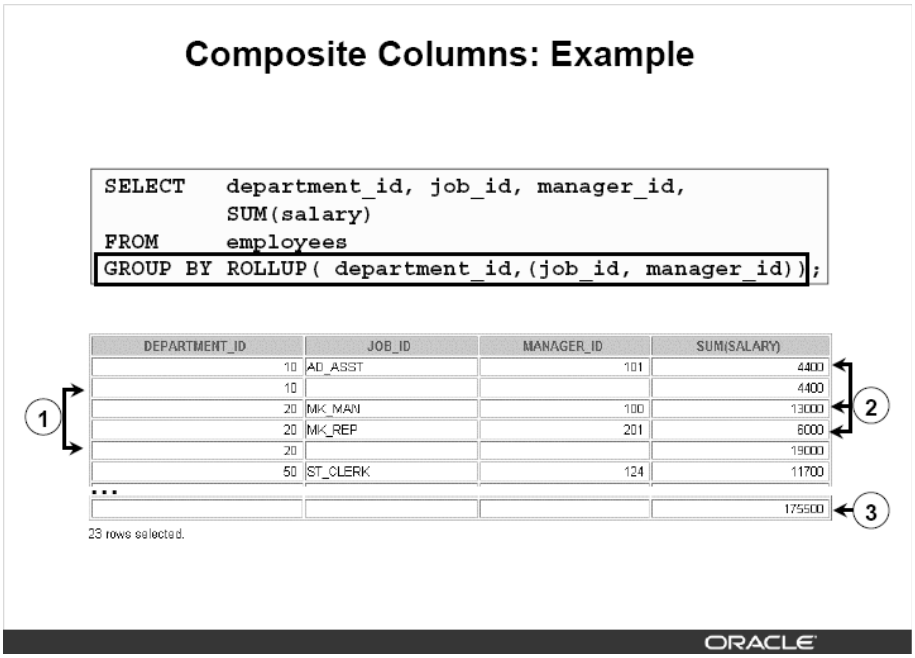
`GROUP BY GROUPING SETS(a, b, c)`	`GROUP BY a UNION ALL`
	`GROUP BY b UNION ALL`
	`GROUP BY c`

`GROUP BY GROUPING SETS(a, b,(b, c))`
(The `GROUPING SETS` expression has a composite column)
	`GROUP BY a UNION ALL`
	`GROUP BY b UNION ALL`
	`GROUP BY b, c`

`GROUP BY GROUPING SETS((a, b, c))`	`GROUP BY a, b, c`

`GROUP BY GROUPING SETS(a, (b), ())`	`GROUP BY a UNION ALL`
	`GROUP BY b UNION ALL`
	`GROUP BY ()`

`GROUP BY GROUPING SETS`
`(a,ROLLUP(b, c))`
(The `GROUPING SETS` expression has a composite column)
	`GROUP BY a UNION ALL`
	`GROUP BY ROLLUP(b, c)`



Ejemplo de Columnas Compuestas

Si estas interesado en la agrupación de las líneas (1), (3) y (4) del ejemplo anterior, no puedes limitar el cálculo para esas agrupaciones sin usar columnas compuestas. Con las columnas compuestas, esto es posible procesando las columnas JOB_ID y MANAGER_ID como una unidad en la cláusula ROLLUP.
Las columnas entre paréntesis son tratadas como una unidad mientras se calcula ROLLUP y CUBE.

En el ejemplo anterior se calculan las siguientes agrupaciones:

1. (department_id, job_id, manager_id)
2. (department_id)
3. ( )

El ejemplo despliega lo siguiente:

- El salario total para cada departamento (***Etiqueta 1***)
- El salario total para cada departamento, puesto y jefe (***Etiqueta 2***)
- Gran Total (***Etiqueta 3***)

Este ejemplo puede ser escrito como:

```
SELECT   department_id, job_id, manager_id, SUM(salary)
FROM     employees
GROUP BY department_id,job_id, manager_id
UNION  ALL
SELECT   department_id, TO_CHAR(NULL),TO_NUMBER(NULL), SUM(salary)
FROM     employees
GROUP BY department_id
UNION ALL
SELECT   TO_NUMBER(NULL), TO_CHAR(NULL),TO_NUMBER(NULL), SUM(salary)
FROM     employees
GROUP BY ();
```

Considere el ejemplo:

```
SELECT department_id, job_id,manager_id, SUM(salary)
FROM   employees
GROUP BY ROLLUP( department_id,job_id, manager_id);
```

La consulta anterior resulta en el Servidor de Oracle las siguientes agrupaciones:

1. (department_id, job_id, manager_id)
2. (department_id, job_id)
3. (department_id)
4. ( )

En ausencia de una optimización que vea a través de bloques de consultas para generar el plan de ejecución, la consulta anterior necesitara tres búsquedas en la tabla base, EMPLOYEES. Esto puede ser muy ineficiente. Por lo tanto el uso de columnas compuestas es recomendado.

Columnas Concatenadas

Las columnas concatenadas ofrecen un camino conciso para generar combinaciones útiles de agrupaciones. Las agrupaciones concatenadas son simplemente especificadas al listar múltiples conjuntos de agrupaciones, CUBE, ROLLUP, y separando estas con comas, Aquí se muestra un ejemplo de un conjunto de agrupaciones concatenadas:

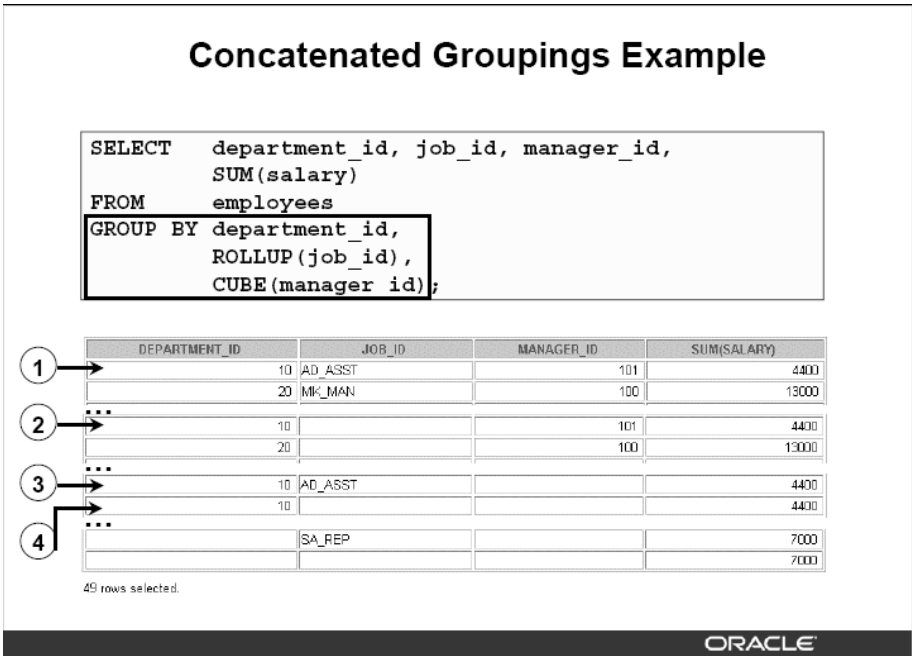
```
GROUP BY GROUPING SETS(a, b), GROUPING SETS(c, d)
```

La anterior sentencia SQL define las siguientes agrupaciones:

```
(a, c), (a, d), (b, c), (b, d)
```

La concatenación de conjuntos de agrupaciones es de mucha ayuda por las siguientes razones:

- Facilita el desarrollo de las consultas, no se necesita manualmente enumerar todas las agrupaciones
- Uso para aplicaciones: las aplicaciones OLAP con frecuencia involucran la concatenación de conjuntos de agrupaciones, con cada conjunto de agrupación definen grupos necesarios para una dimensión.



Ejemplo de Agrupaciones Concatenadas

El ejemplo anterior muestra las siguientes agrupaciones:

- (department_id, manager_id, job_id)
- (department_id, manager_id)
- (department_id, job_id)
- (department_id)

El salario total para cada uno de estos grupos es calculado.

En el ejemplo se muestra lo siguiente:

- El salario total para todos los departamentos, puestos y jefes
- El salario total para todos los departamentos y jefes
- El salario total para todos los departamentos y puestos
- El salario total para todos los departamentos

Para un fácil entendimiento los detalles del departamento 10 son marcados en la ilustración.

Resumen

En este capítulo ha aprendido como:

- Utilizar la operación ROLLUP para producir subtotales
- Utilizar la operación CUBE para producir valores cruzados
- Usar la función GROUPING para identificar los valores de las filas creadas por ROLLUP y CUBE
- Usar la sintaxis GROUPING SETS para definir múltiples agrupaciones en la misma consulta
- Usar la cláusula GROUP BY, para combinar expresiones en varia vías:
  - Columnas compuestas
  - Conjuntos de agrupaciones concatenadas