Tema 9. Consultas con cláusulas avanzadas

Contenido

1 Agrupamiento de registros

Es muy común utilizar consultas en las que se desee agrupar los datos a fin de realizar cálculos en vertical, es decir calculados a partir de datos de distintos registros. Para ello se utiliza la cláusula GROUP BY que permite indicar en base a qué registros se realiza la agrupación. Con GROUP BY la instrucción SELECT queda de esta forma:

```
SELECT listaDeExpresiones
FROM listaDeTablas
[JOIN tablasRelacionadasYCondicionesDeRelación]
[WHERE condiciones]
[GROUP BY grupos]
[HAVING condicionesDeGrupo]
[ORDER BY columnas];
```

En el apartado GROUP BY, se indican las columnas por las que se agrupa. La función de este apartado es crear un único registro por cada valor distinto en las columnas del grupo. Si por ejemplo agrupamos en base a las columnas tipo y modelo en una tabla de existencias, se creará un único registro por cada tipo y modelo distintos:

```
SELECT tipo, modelo
FROM existencias
GROUP BY tipo, modelo;
```

Si la tabla de existencias sin agrupar es:

TI	MODELO	N_ALMACEN	CANTIDAD
AR	6	1	2500
AR	6	2	5600
AR	6	3	2430
AR	9	1	250
AR	9	2	4000
AR	9	3	678
AR	15	1	5667
AR	20	3	43
BI	10	2	340
BI	10	3	23
BI	38	1	1100
BI	38	2	540
BI	38	3	

La consulta anterior creará esta salida:

TI	MODELO
AR	6
AR	9
AR	15
AR	20
BI	10
BI	38

Es decir es un resumen de los datos anteriores. Los datos n_almacen y cantidad no están disponibles directamente ya que son distintos en los registros del mismo grupo. Sólo se pueden utilizar desde funciones (como se verá ahora). Es decir esta consulta es errónea:

```
SELECT tipo, modelo, cantidad
FROM existencias
GROUP BY tipo, modelo;
```

1.1 Funciones de cálculo con grupos

Se utilizan funciones que permiten trabajar con los registros de un grupo:

Función	Significado
COUNT(*)	Cuenta los elementos de un grupo. Se utiliza el asterisco para no tener que indicar un nombre de columna concreto, el resultado es el mismo para cualquier columna
SUM(*expresión*)	Suma los valores de la expresión
AVG(*expresión*)	Calcula la media aritmética sobre la expresión indicada
MIN(*expresión*)	Mínimo valor que toma la expresión indicada
MAX(*expresión*)	Máximo valor que toma la expresión indicada
STDDEV(*expresión*)	Calcula la desviación estándar
VARIANCE(*expresión*)	Calcula la varianza

Todas las funciones de la tabla anterior se calculan para cada elemento del grupo, así la expresión:

```
SELECT tipo, modelo, SUM(Cantidad)
FROM existencias
GROUP BY tipo, modelo;
```

Obtiene este resultado:

```
TIPO               MODELO SUM(CANTIDAD)
------------------ ---------- -------------
AR                         20            43
BI                         38          1640
AR                          6         10530
BI                         10           363
AR                          9          4928
AR                         15          5667
```

1.2 Funciones de listas

Las funciones de listas trabajan sobre un grupo de columnas dentro de una misma fila. Comparan los valores de cada una de las columnas en el interior de una fila para obtener el mayor o el menor valor de la lista

Función	Propósito
GREATEST (valor1, valor2…)	Obtiene el mayor valor de la lista
LEAST (valor1, valor2, …)	Obtiene el menor valor de la lista

Ejemplo:

```
SELECT GREATEST ('ANA', 'MARIA', 'LUISA')
   "Greatest" FROM DUAL;
```

Muestra:

```
Greatest
--------
MARIA
```

```
SELECT LEAST ('ANA', 'MARIA', 'LUISA')
   "Least" FROM DUAL;
```

Muestra:

```
Least
--------
ANA
```

1.3 Condiciones HAVING

A veces se desea restringir el resultado de una expresión agrupada, por ejemplo con:

```
SELECT tipo, modelo, SUM(Cantidad)
FROM existencias
WHERE SUM(Cantidad)>500
GROUP BY tipo, modelo;
```

Pero la consulta devuelve **error** porque primero se calcula el where y después los grupos, por lo que esa condición no la puede realizar al no estar establecidos los grupos. Por ello se utiliza la cláusula HAVING, que se ejecuta una vez realizados los grupos. Se usaría de esta forma:

```
SELECT tipo, modelo, cantidad, SUM(Cantidad)
FROM existencias
GROUP BY tipo, modelo
HAVING SUM(Cantidad)>500;
```

Eso no implica que no se pueda usar WHERE. Ésta expresión sí es válida:

```
SELECT tipo, modelo, SUM(Cantidad)
FROM existencias
WHERE tipo!='AR'
GROUP BY tipo, modelo
HAVING SUM(Cantidad)>500;
```

En definitiva, el orden de ejecución de la consulta marca lo que se puede utilizar con WHERE y lo que se puede utilizar con HAVING:

Para evitar problemas estos podrían ser los pasos en la ejecución de una instrucción de agrupación por parte del gestor de bases de datos:

1. Seleccionar las filas deseadas utilizando WHERE. Esta cláusula eliminará filas según la condición indicada.
2. Se establecen los grupos indicados en la cláusula GROUP BY
3. Se calculan los valores de las funciones de totales (COUNT, SUM, AVG,...)
4. Se filtran los registros que cumplen la cláusula HAVING
5. El resultado se ordena en base al apartado ORDER BY.

2 Combinaciones especiales

2.1 UNION

La palabra UNION permite añadir el resultado de un SELECT a otro SELECT. Para ello ambas instrucciones tienen que utilizar el mismo número y tipo de columnas. Ejemplo:

```
SELECT nombre FROM provincias
 UNION
SELECT nombre FROM comunidades;
```

El resultado es una tabla que contendrá nombres de provincia y de comunidades. Es decir, UNION crea una sola tabla con registros que estén presentes en cualquiera de las consultas. Si están repetidas sólo aparecen una vez, para mostrar los duplicados se utiliza UNION ALL en lugar de la palabra UNION.

Es muy importante señalar que tanto ésta cláusula como el resto de combinaciones especiales, requieren en los dos SELECT que unen el **mismo tipo de columnas** (y en el **mismo orden**).

2.2 INTERSECT

De la misma forma, la palabra INTERSECT permite unir dos consultas SELECT de modo que el resultado serán las filas que estén presentes en ambas consultas.

Ejemplo; tipos y modelos de piezas que se encuentren en los almacenes 1 y 2 (sólo los que se encuentran en ambos):

```
SELECT tipo,modelo FROM existencias
WHERE n_almacen=1
INTERSECT
SELECT tipo,modelo FROM existencias
WHERE n_almacen=2;
```

2.3 MINUS

Con MINUS también se combinan dos consultas SELECT de forma que aparecerán los registros del primer SELECT que no estén presentes en el segundo.

Ejemplo; tipos y modelos de piezas que se encuentren el almacén 1 y no en el 2

```
(SELECT tipo,modelo FROM existencias
WHERE n_almacen=1)
MINUS
(SELECT tipo,modelo FROM existencias
WHERE n_almacen=2);
```

Se podrían hacer varias combinaciones anidadas (una unión cuyo resultado se intersecará con otro SELECT por ejemplo), en ese caso es conveniente utilizar paréntesis para indicar qué combinación se hace primero:

```
(SELECT....
....
UNION
SELECT....
...
)
MINUS
SELECT.... /* Primero se hace la unión y luego la diferencia*/
```

3 Consultas con ROWNUM

La función ROWNUM devuelve el número de la fila de una consulta. Por ejemplo en:

```
SELECT ROWNUM, edad, nombre FROM clientes;
```

Aparece el número de cada fila en la posición de la tabla. Esa función actualiza sus valores usando subconsultas, de modo que la consulta:

```
SELECT ROWNUM AS ranking, edad, nombre
FROM (SELECT edad, nombre FROM clientes ORDER BY edad DESC);
```

Puesto que la consulta

```
SELECT edad, nombre FROM clientes ORDER BY edad DESC;
```

obtiene una lista de los clientes ordenada por edad, el SELECT superior obtendrá esa lista pero mostrando el orden de las filas en esa consulta. Eso permite hacer consultas del tipo top-n, (los n más....).

Por ejemplo para sacar el top-10 de la edad de los clientes (los 10 clientes de más edad):

```
SELECT ROWNUM AS ranking, edad, nombre FROM clientes
FROM (SELECT edad, nombre FROM clientes ORDER BY edad DESC)
WHERE ROWNUM<=10;
```

Cuando utilizamos ROWNUM en la cláusula WHERE, sólo podemos poner como condiciones:

WHERE ROWNUM<=n donde n es un número entero positivo

Ó

WHERE ROWNUM=1