TEMA 4.6.- CLÁUSULAS AVANZADAS DE SELECCIÓN

En esta unidad nos ocuparemos de una nueva cláusula que acompaña a la sentencia SELECT y que permite hacer las consultas más complejas. Veremos las ordenes que nos permiten agrupar filas según una condición, o sin condición, para obtener algún resultado de ese grupo de filas agrupadas.

También veremos otro tipo de combinación de tablas: la que nos permite seleccionar algunas filas de una tabla aunque éstas no tengan su correspondencia con alguna tabla.

1.- AGRUPACIÓN DE ELEMENTOS. GROUP BY Y HAVING

A veces nos interesa consultar los datos según grupos determinados. Así, para saber cuál es el salario medio de cada departamento de la tabla EMPLE, las cláusulas que conocemos hasta ahora no son suficientes. Necesitamos realizar un agrupamiento por departamento, para lo que usaremos la cláusula GROUP BY:

SELECT dept_no, AVG(salario) FROM emple GROUP BY dept_no;

La sentencia SELECT posibilida agrupar uno o más conjuntos de filas. El agrupamiento se lleva a cabo mediante la cláusula GROUP BY por las columnas especificadas y en el orden especificado. La fórmula es la siguiente:

```
SELECT…
FROM…
GROUP BY columna1, columna2, columna3,…
HAVING condición
ORDER BY
```

Los datos seleccionados en la sentencia SELECT que lleva el GROUP BY deben ser: una constante, una función de grupo (SUM, COUNT, AVG,…), una columna expresada en el GROUP BY.

1.1.- GROUP BY

Sirve para calcular propiedades de una o más conjuntos de filas. Además, si se selecciona más de un conjunto de filas, controla que las filas de la tabla original sean

agrupadas en una tabla temporal. Del mismo modo que existe la condición de búsqueda WHERE para filas individuales, también hay una condición de búsqueda para grupos de filas: HAVING. La cláusula HAVING se emplea para controlar cuál de los conjuntos de filas se visualiza. Se evalúa sobre la tabla que devuelve el GROUP BY. No puede existir son GROUP BY.

Ejercicio 1 relación complementaria 3

1.2.- HAVING

Similar a la cláusula WHERE, pero trabaja con grupos de filas; pregunta por una característica de grupo, es decir, pregunta por los resultados de las funciones de grupo, lo cual WHERE no puede hacer.

Ejemplo Visualiza los departamentos con más de 4 empleados ordenando la salida descendentemente por número de empleados:

```
SELECT dept_no, COUNT(*) FROM emple
GROUP BY dept_no
HAVING COUNT(*) > 4
ORDER BY COUNT(*) DESC;
```

Cuando usamos la orden ORDER BY con columnas y funciones de grupos hemos de tener en cuenta que ésta se ejecuta detrás de las clausulas WHERE, GROUP BY y HAVING. En ORDER BY podemos especificar funciones de grupo, columnas de GROUP BY o su combinación.

La explicación de las cláusulas en tiempo de ejecución se efectúa en el siguiente orden:

WHERE	Selecciona las filas
GROUP BY	Agrupa estas filas
HAVING	Filtra los grupos. Selecciona y elimina los grupos
ORDER BY	Clasifica la salida. Ordena los grupos

Ejercicio 2 relación complementaria 3

2.- COMBINACIÓN EXTERNA (OUTER JOIN)

Permite seleccionar algunas filas de una tabla aunque éstas no tengan correspondencia con las filas de la otra tabla con la que se combinan

```
SELECT tabla1.colum1, tabla1.colum2, tabla2.colum1
FROM tabla1, tabla2
WHERE tabla1.colum1 = tabla2.colum1(+)
```

Esta SELECT seleccionará todas la filas de la tabla1, aunque no tengan correspondencia con las filas de la tabla tabla2; se denota con (+) detrás de la columna tabla2 (que es la tabla donde no se encuentran las filas) en la cláusula WHERE. Se obtendrá una fila con las columnas de la tabla1 y el resto de columnas de la tabla2 se rellenan con NULL. Veamos un ejemplo:

Sean las tablas EMPLE y DEPART. Si comprobamos su contenido vemos que en EMPLE el departamento 40 no existe pero si en DEPART. Queremos una consulta donde se visualice el número de departamento, el nombre y el número de empleados que tiene. Combinemos ambas tablas:

```
SELECT d.dept_no, dnombre, count(e.emp_no)
FROM emple e, depart d
WHERE e.dept_no = d.dept_no
GROUP BY d.dept_no, dnombre;
```

DEPT_NO	DNOMBRE	COUNT(E.EMP_NO)
30	VENTAS	6
10	CONTABILIDAD	3
20	INVESTIGACION	5

Observamos que con esta sentencia solo obtenemos los departamentos que tienen empleados, el 40 se pierde. Para que este departamento que no se corresponde con ninguno de la tabla EMPLE, usamos la combinación externa especificando con un (+) a continuación del nombre de la columna de la tabla en la que no aparece.

```
SELECT d.dept_no, dnombre, count(e.emp_no)
FROM emple e, depart d
WHERE e.dept_no (+)= d.dept_no
GROUP BY d.dept_no, dnombre;
```

DEPT_NO	DNOMBRE	COUNT(E.EMP_NO)
30	VENTAS	6
10	CONTABILIDAD	3
40	PRODUCCION	0
20	INVESTIGACION	5

3.- OPERADORES UNION, INTERSECT Y MINUS

Estos son operadores de conjuntos. Los conjuntos son las filas resultantes de cualquier sentencia SELECT válida que permiten combinar los resultados de varias SELECT para obtener un único resultado.

Supongamos que tenemos dos listas de centros de enseñanza de una ciudad y que queremos enviar a esos centros una serie de paquetes de libros. Dependiendo de ciertas características de los centros, podemos enviar libros a todos los centros de ambas listas (UNION), a los centros que estén en las dos listas (INTERSECT), o a los centros que están en una pero no en la otra (MINUS). El formato de SELECT con estos operadores es el siguiente:

```
SELECT  . . . FROM . . .WHERE . . .
Operador_de_conjunto
SELECT  . . . FROM . . .WHERE . . .
```

3.1.- OPERADOR UNION

Combina los resultados de dos consultas. Las filas duplicadas se reducen a una fila única.

```
SELECT col1, col2, . . . FROM tabla1 .WHERE condición
UNION
SELECT col1, col2, . . . FROM tabla2 .WHERE condición
```

Ejercicio 3 relación complementaria 3

UNION ALL combina los resultados de dos consultas. Cualquier duplicación de filas que se dé en el resultado final aparecerá en la consulta. En el ejercicio complementario anterior, usando UNION ALL aparecen los nombre duplicados

3.2.- OPERADOR INTERSECT

Devuelve las filas que son iguales en ambas tablas. Todas las filas duplicadas serán eliminadas antes de la generación del resultado final.

```
SELECT col1, col2, . . . FROM tabla1 .WHERE condición
INTERSECT
SELECT col1, col2, . . . FROM tabla2 .WHERE condición
```

Ejercicio 4 relación complementaria 3

3.3.- OPERADOR MINUS

Devuelve las filas que están en el primer SELECT y no en el segundo. Las filas duplicadas del primer conjunto se reducirán a una fila única antes de que empiece la comparación con el otro conjunto. Su formato es:

```
SELECT col1, col2, . . . FROM tabla1 .WHERE condición
MINUS
SELECT col1, col2, . . . FROM tabla2 .WHERE condición
```

Ejercicio 5 relación complementaria 3

3.4.- REGLAS PARA LA UTILIZACIÓN DE OPERADORES DE CONJUNTOS

Los operadores de conjunto pueden encadenarse. Los conjuntos se evalúan de izquierda a derecha; para forzar precedencia se pueden utilizar paréntesis.

Estos operadores se pueden manejar con consultas de diferentes tablas, siempre que se apliquen las siguientes reglas:

- Las columnas de las dos consultas se relacionan en orden, izquierda a derecha.
- Los nombres de columnas de la primera sentencia SELECT no tienen por qué ser los mismos que los nombres de columnas de la segunda.
- Las SELECT necesitan tener el mismo número de columnas
- Los tipos de datos deben coincidir, aunque la longitud no tiene que ser la misma.