PARTE II

AGRUPAMIENTOS, SUBCONSULTAS Y CONSULTAS MULTITABLA SQL.

1. AGRUPAMIENTO DE REGISTROS.

Las consultas que hemos visto pueden obtener totales de todas las filas de un campo o una expresión calculada sobre uno o varios campos.

Hay ocasiones en las que utilizamos consultas (o subconsultas) de resumen nos va a interesar calcular totales parciales, es decir, agrupados según un determinado campo.

De este modo podríamos obtener de la tabla EMPLEADOS, en la que se guarda su SALARIO y su POBLACION, la media de salarios en función de la población.

Otro ejemplo podría ser para una serie de clientes y obtener el número de veces que ha realizado un pedido, etc. En todos estos casos en lugar de una única fila de resultados necesitaremos una fila por cada actividad, cada cliente, etc.

Para poder realizar estas operaciones con subtotales tenemos que utilizar una nueva cláusula que es GROUP BY.

La sintaxis es la siguiente:

```
SELECT columna1, columna2, ...
FROM tabla1, tabla2, ...
WHERE condición1, condición2, …
GROUP BY columna1, columna2, …
HAVING condición
ORDER BY ordenación;
```

En la cláusula GROUP BY se colocan las columnas por las que vamos a agrupar.

En la cláusula HAVING se especifica la condición que han de cumplir los grupos para que se realice la consulta.

Es muy importante que te fijes bien en el orden en el que se ejecutan las cláusulas:

1. **WHERE** que filtra las filas según las condiciones que pongamos.
2. **GROUP BY** que crea una tabla de grupos nueva.
3. **HAVING** filtra los grupos.
4. **ORDER BY** que ordena o clasifica la salida. Los campos de ordenación deben encontrarse en la cláusula SELECT.

Las columnas que aparecen en el SELECT y que no aparezcan en la cláusula GROUP BY deben tener una función de agrupamiento. Si esto no se hace así producirá un error. Otra opción es poner en la cláusula GROUP BY las mismas columnas que aparecen en SELECT. Veamos un par de ejemplos:

Obtener la media de los salarios agrupados por población, con dos decimales.

```
SELECT TRIM(POBLACION), ROUND(AVG(SALARIO),2)
FROM EMPLEADOS
GROUP BY TRIM(POBLACION);
```

Obtenemos la suma de los salarios agrupados por provincia, de los empleados que vivan en la provincia de Cádiz o en la provincia de Badajoz:

```
SELECT PROVINCIA, SUM (salario)
FROM EMPLEADOS
GROUP BY provincia
HAVING TRIM(PROVINCIA) LIKE 'CADIZ' OR TRIM(PROVINCIA) LIKE 'TOLEDO';
```

De esta forma podemos agrupar informaciones y podemos mostrar la información por la que estamos agrupando.

La agrupación se puede realizar por el número de campos que sea necesario, **siempre que se indiquen tanto en la cláusula SELECT, como en la cláusula GROUP BY los mismos campos.** Si los campos de agrupación son diferentes obtendremos un error. En la cláusula SELECT también tendremos que incluir la función que de agrupación que estemos utilizando a parte de los campos de agrupación.

```
SELECT TRIM(PROVINCIA), TRIM(POBLACION), SUM (salario) AS SUMA
FROM EMPLEADOS
GROUP BY TRIM(POBLACION), TRIM(PROVINCIA)
ORDER BY TRIM(PROVINCIA), SUMA;
```

Observar como el orden en que se especifican los campos en el SELECT y en el ORDER BY no es relevante.

Observa también como se realiza la ordenación.

2. SUBCONSULTAS.

Hay veces en las que los resultados obtenidos en una consulta son necesario en otra consulta, a la consulta de la que obtenemos los datos para compararlos la llamaremos subconsulta:

```
SELECT listaExpr
FROM tabla
WHERE expresión OPERADOR (SELECT listaExpr FROM tabla);
```

La subconsulta puede ir dentro de las cláusulas WHERE, HAVING o FROM.

El orden en la ejecución de las consultas es importante, primero se ejecuta la subconsulta y después se ejecuta la consulta principal, comparando con el resultado de la subconsulta.

Los tipos de datos que devuelven la subconsulta y la columna con la que se compara han de ser iguales o compatible.

Una subconsulta puede devolvernos un solo valor o puede que nos devuelva varios valores, en función de esto vamos a utilizar diferentes operadores.

Cuando una consulta devuelve un solo valor utilizamos operadores relacionales: >, <, >=, <=, != o =.

Las subconsultas deben ir entre paréntesis y a la derecha del operador.

Si la subconsulta devolviera más de un valor en entonces se produciría un error.

Pongamos un ejemplo, si queremos obtener el nombre, apellidos y el salario de los empleados que tengan un salario menor al salario de ANA.

```
SELECT TRIM(NOMBRE) || ' ' ||TRIM(APELLIDO1), SALARIO
FROM EMPLEADOS
WHERE SALARIO < (SELECT SALARIO
                 FROM EMPLEADOS
                 WHERE TRIM(NOMBRE) like 'ANA');
```

Obtendríamos el nombre de los empleados y el sueldo de aquellos que cobran menos que Ana. ¿Qué sucedería si hubiera varios empleados que se llamaran ANA?

Cuando una consulta devuelve varios valores tenemos que evaluar todos los posibles valores que nos devuelve, por lo que no podemos utilizar un operador relacional, ya que sólo puede relacionar dos valores, por lo que tendremos que utilizar operadores que me permitan evaluar varios valores como son: **ALL, ANY, IN, EXIST, …**

- ✓ **ALL**. Compara con todas las filas de la consulta. La instrucción resultará cierta si es cierta toda la comparación con los registros de la subconsulta.

  ALL debe utilizarse junto a un operador relacional:

```
SELECT nombre, apellidos1, apellido2, salario
FROM EMPLEADOS
WHERE SALARIO < ALL (SELECT SALARIO
                     FROM EMPLEADOS
                     WHERE TRIM(poblacion) LIKE 'TALAVERA DE LA REINA');
```

- ✓ **ANY** Compara el valor de una expresión con cada uno del conjunto de valores producido por una subconsulta, si alguna de las comparaciones individuales da como resultado TRUE, ANY devuelve TRUE, si la subconsulta no devuelve nada devolverá FALSE.

  ANY debe utilizarse junto a un operador relacional, igual que **ALL**:

```
SELECT nombre, apellidos1, apellido2, salario
FROM EMPLEADOS
WHERE SALARIO = ANY (SELECT SALARIO
                     FROM EMPLEADOS
                     WHERE TRIM(poblacion) LIKE 'TALAVERA DE LA REINA')
```

- ✓ **EXISTS, NOT EXISTS** Examina si una subconsulta produce alguna fila de resultados. Devuelve TRUE si la consulta devuelve filas, si no devuelve filas es FALSE.

```
SELECT nombre, apellido1, apellido2, salario
FROM EMPLEADOS
WHERE EXISTS (SELECT SALARIO
              FROM EMPLEADOS
              WHERE TRIM(poblacion) LIKE 'KUALA LUMPUR');
```

  Sólo evaluamos si la subconsulta devuelve o no algún valor, no importa el valor que devuelva. Observar que no comparamos respecto de ningún valor.

- ✓ **IN**, **NOT IN** Lo que hace es comprobar si existe un valor dentro del conjunto de valores devueltos por una subconsulta, Devuelve true si el resultado existe y false si el resultado no existe.

```
SELECT nombre, salario
FROM EMPLEADOS
WHERE TRIM(poblacion) LIKE 'FUENSALIDA'
AND salario IN (SELECT salario
                FROM EMPLEADOS
                WHERE TRIM(poblacion) LIKE 'TALAVERA DE LA REINA');
```

3. CONSULTAS MULTITABLA.

Hasta ahora las consultas que hemos realizado se referían a una sola tabla, pero también es posible hacer consultas usando varias tablas en la misma sentencia SELECT.

Esto permitirá realizar distintas operaciones como son:

- ✓ La composición interna.
- ✓ La composición externa.

3.1. COMPOSICIONES INTERNAS.

Cuando combinamos dos o más tablas sin ninguna restricción, el resultado será un producto cartesiano. El producto cartesiano entre dos tablas da como resultado todas las combinaciones de todas las filas de esas dos tablas.

Se indica poniendo en la cláusula FROM las tablas que queremos componer separadas por comas. Y puedes obtener el producto cartesiano de las tablas que quieras.

Como lo que se obtiene son todas las posibles combinaciones de filas, debes tener especial cuidado con las tablas que combinas. Si tienes dos tablas de 10 filas cada una, el resultado tendrá 10x10 filas, a medida que aumentemos el número de filas que contienen las tablas, mayor será el resultado final, con lo cual se puede considerar que nos encontraremos con una operación costosa.

Lo más normal es que queramos seleccionar los registros según algún criterio. Necesitaremos discriminar de alguna forma para que únicamente aparezcan filas de una tabla que estén relacionadas con la otra tabla. A esto se le llama asociar tablas (JOIN).

Para hacer una composición interna se parte de un producto cartesiano y se eliminan aquellas filas que no cumplen la condición de composición. Lo importante en las composiciones internas es emparejar los campos que han de tener valores iguales.

Las reglas para las composiciones son:

- ✓ Pueden combinarse tantas tablas como se desee.
- ✓ El criterio de combinación puede estar formado por más de una pareja de columnas.
- ✓ En la cláusula SELECT pueden citarse columnas de ambas tablas, condicionen o no, la combinación.
- ✓ Si hay columnas con el mismo nombre en las distintas tablas, **deben** identificarse especificando la tabla de procedencia o utilizando un alias de tabla. (***En el caso de que los nombres de las columnas no se sean iguales, también es conveniente anteponer el nombre de la tabla o el alias por claridad***)

  NombreTabla.NombreColumna.

Las columnas que utilizamos para unir (relacionar, emparejar, …) las tablas, se denominan columnas de emparejamiento. ***Éstas no tienen por qué estar incluidas en la lista de selección.***

Para poder trabajar con varias tablas, tendremos emparejar tablas que estén relacionadas entre sí. *Una* de las columnas de emparejamiento será clave principal en su tabla y la otra será clave ajena en la otra tabla.

Cuando emparejamos campos debemos especificar de la siguiente forma:

NombreTabla1.Camporelacionado1 = NombreTabla2.Camporelacionado2.

O bien mediante un alias para las tablas.

Veamos un ejemplo, si queremos obtener el nombre de los empleados incluyendo nombres y apellidos de los empleados y el nombre del departamento al que pertenecen, tendremos:

```
SELECT E.NOMBRE, D.NOMBRE_DPTO
FROM EMPLEADOS E, DEPARTAMENTOS D
WHERE E.DPTO_COD = D.DPTO_COD
```

Puedes combinar una tabla consigo misma, ***pero debes poner de manera obligatoria un alias a uno de los nombres de la tabla*** que vas a repetir.

Vamos a realizar una consulta en la que obtengamos los nombres de los proyectos, los nombres de los trabajadores que han participado en cada proyecto y la función que realizó (puesto) en cada proyecto

```
SELECT E.NOMBRE || ' ' || E.APELLIDO1 ||' ' || E.APELLIDO2 "NOMBRE EMPLEADO" ,
        P.NOMBRE "NOMBRE PROYECTO", R.PUESTO
FROM EMPLEADOS E, PROYECTOS P, EMP_PROY R
WHERE E.DNI = R.EMPLEADOS_DNI
                AND P.COD_PROY=R.PROYECTOS_COD_PROY;
```

Vamos a realizar una consulta en la que obtengamos los nombres de los empleados, los nombres de los departamentos en los que han participado, así como el nombre del departamento al que pertenecen.

```
select E.nombre || ' ' || E.APELLIDO1 ||' ' || E.APELLIDO2 "NOMBRE EMPLEADO" ,
    P.nombre "NOMBRE PROYECTO", D.NOMBRE_DPTO
from EMPLEADOS E, PROYECTOS P, EMP_PROy R, DEpartamentos D
where E.DNI = R.EMPLEADOS_dni AND P.COD_PROY=R.PROYECTOS_COD_PROY
      AND E.dpto_cod = D.dpto_cod;
```

Observar como en esta consulta el nombre del departamento se repite tantas veces como el trabajador haya participado en un proyecto.

3.2. COMPOSICIONES EXTERNAS.

Puede que sea necesario seleccionar algunas filas de una tabla, aunque éstas no tengan correspondencia con las filas de la otra tabla.

Imagina que tenemos dos tablas, una la información de los empleados (*Cod_empleado, Nombre, Apellidos, salario y Cod_dpto*) y otra con los departamentos (*Codigo_dep, Nombre*). Recientemente se han creado dos nuevos departamentos, a los que no se les ha asignado ningún empleado. Si tuviéramos que obtener un informe con los datos de los empleados por departamento, seguro que deben aparecer esos departamentos aunque no tengan empleados. Para poder hacer esta combinación usaremos las composiciones externas.

Para poder hacer esta consulta correctamente añadiremos un signo más entre paréntesis (+) en la igualdad entre campos que ponemos en la cláusula WHERE. El carácter (+) irá detrás del nombre de la tabla en la que deseamos aceptar valores nulos.

Por ejemplo:

WHERE es Cod_dpto (+)= Codigo_dep

Obtener un listado con los nombres de los distintos departamentos y sus trabajadores. Ten en cuenta que deben aparecer todos los departamentos, aunque no tengan asignado ningún trabajador.

Resultado:

```
select D.nombre_dpto, E.nombre
from EMPLEADOS E, DEPARTAMENTOS D
where E.dpto_cod(+)= D.dpto_cod;

select D.nombre_dpto,  E.nombre
from EMPLEADOS E, DEPARTAMENTOS D
where E.dpto_cod= D.dpto_cod;
```

Probar las dos consultas y observar los resultados obtenidos en ambas.

3.3. COMPOSICIONES EN LA VERSIÓN SQL99.

En la versión SQL de 1999 se especifica una nueva sintaxis para consultar varias tablas que Oracle incorpora. La razón de esta nueva sintaxis era separar las condiciones de asociación respecto a las condiciones de selección de registros.

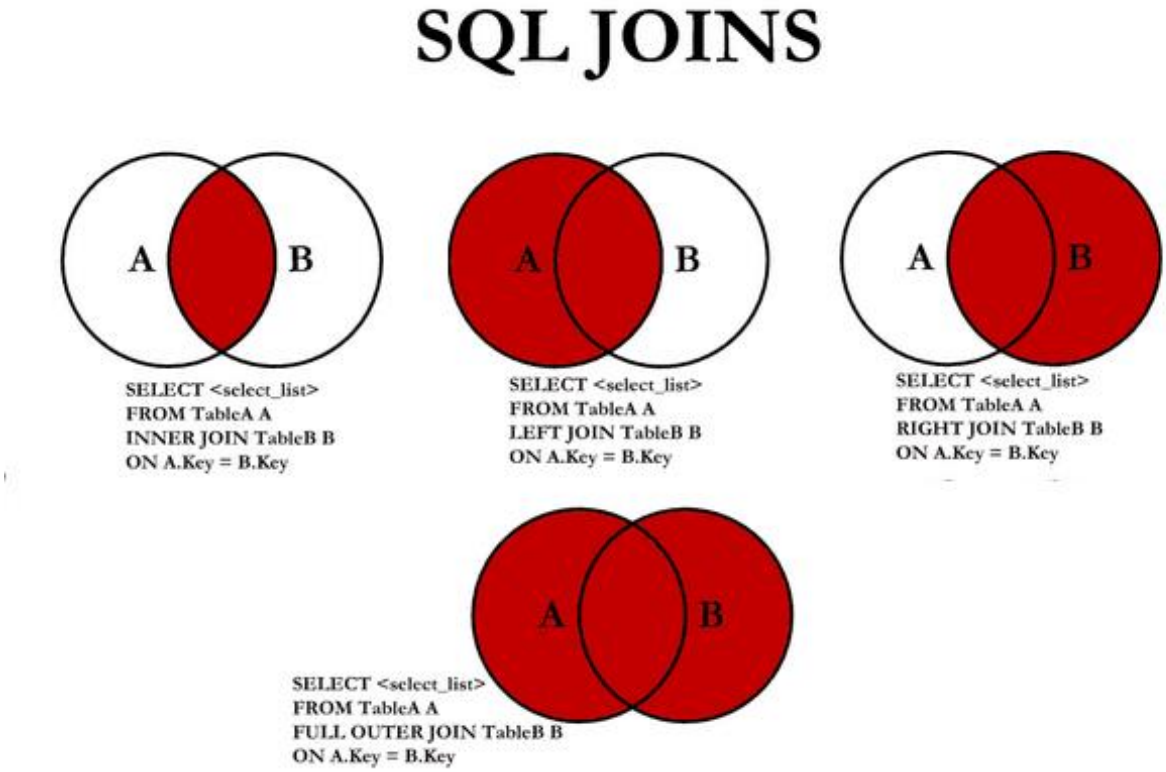
La sintaxis es la siguiente:

```
SELECT tabla1.columna1, tabla1.columna2, …, tabla2.columna1, tabla2.columna2, …
FROM tabla1
[CROSS JOIN tabla2] |
[NATURAL JOIN tabla2] |
```

[JOIN tabla2 USING (columna) |
[JOIN tabla2 ON (tabla1.columna=tabla2.columna)] |
[LEFT | RIGTH | FULL OUTER JOIN tabla2 ON (tabla1.columna=tabla2.columna)]

Su funcionamiento es el siguiente:

- ✓ **[INNER] JOIN:** se utiliza para unir tablas en la que los nombres de columna no coinciden en ambas tablas o se necesita establecer asociaciones más complicadas. Es una composición interna.
- ✓ **[OUTER] JOIN**: se puede eliminar el uso del signo (+) para composiciones externas utilizando un OUTER JOIN, de este modo resultará más fácil de entender. Es una composición externa.
- ✓ **LEFT [OUTER] JOIN:** es una composición externa izquierda, todas las filas de la tabla de la izquierda se devuelven, aunque no haya ninguna columna correspondiente en las tablas combinadas.
- ✓ **RIGTH [OUTER] JOIN:** es una composición externa derecha, todas las filas de la tabla de la derecha se devuelven, aunque no haya ninguna columna correspondiente en las tablas combinadas.
- ✓ **FULL [OUTER] JOIN:** es una composición externa en la que se devolverán todas las filas de los campos no relacionados de ambas tablas.



Podríamos transformar algunas de las consultas con las que hemos estado trabajando:

Vamos a utilizar los ejemplos que hemos visto anteriormente, pero ahora utilizando JOIN:

1. **En una consulta de composición interna utilizamos JOIN:**

   Podemos realizar utilizando los nombres de las tablas

```
SELECT DEPARTAMENTOS.NOMBRE_DPTO, EMPLEADOS.NOMBRE
FROM EMPLEADOS JOIN DEPARTAMENTOS ON (EMPLEADOS.DPTO_COD= DEPARTAMENTOS.DPTO_COD);
```

Como coinciden los nombres de la clave ajena y la clave principal, podíamos utilizar la cláusula USING:

```
SELECT D.NOMBRE_DPTO, E.NOMBRE
FROM EMPLEADOS E  JOIN DEPARTAMENTOS D ON (E.DPTO_COD= D.DPTO_COD);
```

2. **En una consulta de composición externa utilizamos LEFT JOIN o RIGHT JOIN:**

```
SELECT D.NOMBRE_DPTO,  E.NOMBRE
FROM EMPLEADOS E RIGHT JOIN DEPARTAMENTOS D ON (E.DPTO_COD = D.DPTO_COD);
```

Al ejecutar esta consulta observamos que nos aparece el departamento de ADMINISTRACION que anteriormente no aparecía.

Si cambiamos el orden de las tablas podemos utilizar un LEFT JOIN, aunque el resultado sigue siendo el mismo

```
SELECT D.NOMBRE_DPTO,  E.NOMBRE
FROM  DEPARTAMENTOS D LEFT JOIN EMPLEADOS E ON (E.DPTO_COD = D.DPTO_COD);
```

3.4. SUBCONSULTAS CORRELACIONADAS

Una subconsulta correlacionada es una subconsulta que hace referencia a una columna o varias de la consulta más externa.

A veces la subconsulta hace uso de columnas que tienen el mismo nombre que las columnas de las tablas usadas en la consulta más externa, si la subconsulta necesita acceder a esas columnas deberá definirse un alias en la tabla más externa.

Por ejemplo, deseamos obtener los datos de los empleados cuyo salario sea el máximo salario de su departamento:

```
SELECT *
FROM EMPLEADOS E
WHERE SALARIO = (SELECT MAX(SALARIO)
      FROM EMPLEADOS
      WHERE DPTO_COD = E.DPTO_COD);
```

La subconsulta devuelve para cada fila que se recupere de la consulta más externa el máximo salario del departamento que se está recuperando en la consulta externa; para referenciar a dicho departamento se necesita el alias **E** usado en la tabla de la consulta externa

4. OTRAS CONSULTAS MULTITABLA: UNIÓN, INTERSECCIÓN Y DIFERENCIA DE CONSULTAS.

Cuando tengamos varias tablas (o consultas) con los mismos campos, es decir, con la misma estructura y con distintos datos, podemos realizar diferentes operaciones sobre ellos (unirlos, ver las diferencias o los que son iguales).

Las operaciones que podemos realizar son:

- ✓ **UNION:** combina las filas de un primer SELECT con las filas de otro SELECT, desapareciendo las filas duplicadas.
- ✓ **INTERSECT:** examina las filas de dos SELECT y devolverá aquellas que aparezcan en ambos conjuntos. Las filas duplicadas se eliminarán.
- ✓ **MINUS:** devuelve aquellas filas que están en el primer SELECT pero no en el segundo. Las filas duplicadas del primer SELECT se reducirán a una antes de comenzar la comparación.

Para estas tres operaciones es muy importante que utilices en los dos SELECT el mismo número y tipo de columnas y en el mismo orden. Estas operaciones se pueden combinar anidadas, pero es conveniente utilizar paréntesis para indicar que operación quieres que se haga primero.

Veamos un ejemplo de cada una de ellas.

- ✓ **UNION:** Obtener los nombres y ciudades de todos los proveedores y clientes de Alemania.

```
SELECT NombreCia, Ciudad FROM PROVEEDORES WHERE Pais = 'Alemania'
UNION
SELECT NombreCia, Ciudad FROM CLIENTES WHERE Pais = 'Alemania';
```

- ✓ **INTERSECT:** Una academia de idiomas da clases de inglés, francés y portugués; almacena los datos de los alumnos en tres tablas distintas, una llamada "inglés", en una tabla denominada "francés" y los que aprenden portugués en la tabla "portugués". La academia necesita el nombre y domicilio de todos los alumnos que cursan los tres idiomas para enviarles información sobre los exámenes.

```
SELECT nombre, domicilio FROM ingles
INTERSECT
SELECT nombre, domicilio FROM frances
INTERSECT
SELECT nombre, domicilio FROM portugues;
```

- ✓ **MINUS:** Ahora la academia necesita el nombre y domicilio solo de todos los alumnos que cursan inglés (no quiere a los que ya cursan portugués pues va a enviar publicidad referente al curso de portugués).

```
SELECT nombre, domicilio FROM INGLES
MINUS
SELECT nombre, domicilio FROM PORTUGUES;
```