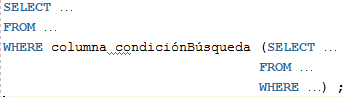
**UT\_6 CONSULTAS AVANZADAS**

* 1. **Subconsultas**
  2. **Agrupamiento de registros**
  3. **Consultas multitabla**
  4. **Unión, Intersección y Diferencia**

**6.1 Subconsultas**

A veces, para realizar alguna operación de consulta necesitamos los datos devueltos por otra consulta. Por ejemplo, si queremos obtener los datos de los empleados que tengan el mismo oficio que “PEPE”, primero hemos de averiguar el oficio de “PEPE” (primera consulta). Conociendo el resultado de la primera consulta podremos averiguar qué empleados tienen el mismo oficio que “PEPE” (segunda consulta). Este tipo de problemas se resuelven utilizando **subconsultas** que no son más que una sentencia SELECT dentro de otra SELECT.

Las subconsultas son las sentencias SELECT que forman parte de una claúsula WHERE de una SELECT anterior. El formato de la subconsulta es:



Primero se ejecutará la subconsulta y el valor obtenido se pasa a la consulta principal.

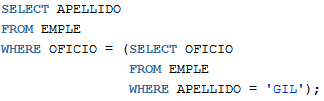
Ejemplo: obtener el APELLIDO de los empleados con el mismo oficio que ‘GIL’. Dividimos el enunciado en dos consultas y primero averiguamos el OFICIO de ‘GIL’ :

 -- Resultado ANALISTA

La segunda consulta buscará el APELLIDO de los empleados con oficio ‘ANALISTA’

-- GIL y FERNANDEZ

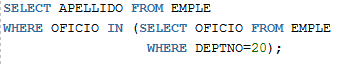
Podemos resumir ambas consultas en una sola:



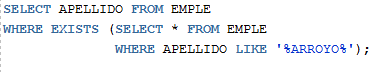
* Mostrar los datos (apellido, oficio, salario y fecha de alta) de aquellos empleados que desempeñan el mismo oficio que ‘JIMENEZ’ o que tengan un salario mayor o igual que ‘FERNANDEZ’.

Las subconsultas normalmente aparecen como parte de la condición de búsqueda de una clausula WHERE o HAVING. Las condiciones de búsqueda que nos podemos encontrar en una subconsulta son:

* **Test de comparación en subconsultas (>, <, <>, <=, >=, =)**. Compara el valor de una expresión con un valor único producido por una subconsulta. En el ejemplo anterior la subconsulta devuelve un único resultado.
* **Test de pertenencia a un conjunto devuelto por una subconsulta (IN)**. Ejemplo: obtener los apellidos de los empleados cuyo oficio sea alguno de los oficios que hay en el departamento 20.



* **Test de existencia (EXISTS, NOT EXISTS).** Examina si una subconsulta produce alguna fila de resultados. El test será TRUE si devuelve filas y FALSE en caso contrario. Visualizar los APELLIDOS de los empleados si existe alguno llamado ‘ARROYO’. Se utilizará la claúsula NOT EXISTS si queremos comprobar si la subconsulta no devuelve ninguna fila.

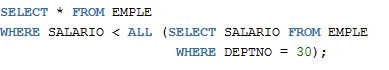


* **Test de comparación cuantificada (ANY y ALL).** Se utilizan junto a los operadores de comparación (>, <, <>, <=, >=, =).

**ANY** compara el valor de una expresión con cada uno del conjunto de valores devuelto por la subconsulta, si alguna de las comparaciones individuales da resultado true, ANY devolverá true, si la sulbconsulta devuelve false, ANY devolverá false. Ejemplo, obtener los datos de los empleados cuyo salario sea igual a algún salario de los empleados del departamento 30.

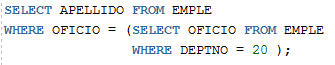


**ALL** compara el valor de una expresión con cada uno del conjunto de valores devuelto por la subconsulta, si todas las comparaciones individuales da como resultado true, ALL devuelve true, en caso contrario devuelve false. Ejemplo: obtener los datos de los empleados cuyo salario sea menor a cualquier salario de los empleados del departamento 30.



Las subconsultas pueden generar valores simples o listas de valores:

* Las que generan **valores simples** son las que devuelven una fila o un valor simple. En ellas se utilizan los operadores de comparación. Si la subconsulta obtiene más de una fila, se produce un mensaje de error. En el ejemplo siguiente se intenta obtener los apellidos de los empleados cuyo oficio coincida con algún oficio del departamento 20, pero en este departamento hay varios oficios, la subconsulta devuelve más de una fila. No se podrán utilizar los operadores de comparación.



Producirá el siguiente error:



* Las subconsultas que generan **listas de valores** son las que devuelven más de una fila o más de un valor. En este caso se utilizará el operador **IN** en la cláusula WHERE. Corregimos el error anterior:

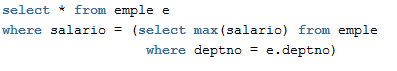


* **Consultar los datos de los empleados que trabajen en ‘MADRID’ o ‘BARCELONA’**
* **Mostrar los apellidos y oficios de todos los empleados del departamento 20 cuyo oficio sea igual al de cualquiera de los departamentos de ‘VENTAS’.**
* **Obtener el apellido de los empleados con el mismo oficio y salario que ‘GIL’ .**

La **subconsulta puede devolver varios campos**, en este caso, la cláusula WHERE debe contener también varios campos encerrados entre paréntesis y han de coincidir en número y tipo de datos con los de la subconsulta. Habrá una correspondencia entre los campos especificados en WHERE y los que devuelve la subconsulta.

* **Mostrar los apellidos y oficios de los empleads que tienen el mismo trabajo que ‘JIMENEZ’.**
* **Mostrar apellido, oficio y salario de los empleados del departamento de ‘FERNANDEZ’ que tengan el mismo salario.**

Una **subconsulta correlacionada** es aquella que hace referencia a una columna o varias de la consulta más externa. Aveces 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, obtener los datos de los empleados cuyo salario sea el máximo de su departamento.



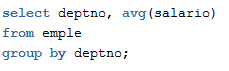
La subconsulta devuelve para cada fila que se recupere de la consulta más externa el máximo salario del departamento que se está recuperando en la consulta externa. Para referenciar a dicho departamento se necesita un alias **e** usado en la tabla de la consulta externa.

* **Mostrar los datos de los empleados cuyo salario supera la media de su departamento.**

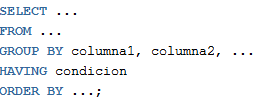
**6.2 Agrupamiento de registros. GROUP BY y HAVING**

Hasta ahora hemos utilizado la orden SELECT para recuperar filas de una tabla y WHERE para seleccionar qué filas se van a recuperar. Con las funciones de grupo se puede trabajar con conjuntos de filas, por ejemplo, el salario medio o la suma de todos los salarios.

Pero, a veces, nos interesa consultar los datos según grupos determinados. Por ejemplo, para averiguar el salario medio de cada departamento necesitaremos realizar un agrupamiento por departamento y utilizaremos la cláusula **GROUP BY**. La consulta quedaría de la siguiente forma:



El formato de la orden SELECT con la cláusula GROUP BY queda de la siguiente manera:



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

La cláusula GROUP BY sirve para calcular propiedades de uno o más conjuntos de filas. Además si se selecciona más de un conjunto de filas, GROUP BY controla que las filas de la tabla original sean agrupadas en una temporal. Del mismo modo que se puede especificar una cláusula WHERE para filas individuales, también hay una condición de búsqueda para grupos de filas: **HAVING** . Esta cláusula se usa para controlar cuál de los conjuntos de filas se visualiza. Se evalúa la condición sobre la tabla que devuelve el GROUP BY y siempre existe junto a GROUP BY.

* **Mostrar a partir de la tabla EMPLE el número de empleados que hay en cada departamento. Será necesario agrupar por departamento y utilizar la función COUNT.**
* **Mostrar el codigo de departamento, el número de empleados que hay en cada uno para aquellos departamentos en los que trabajan más de 4 empleados. Se especificará la cláusula HAVING para comprobar la condición. (HAVING COUNT (\* ) > 4)**
* **Visualiza los departamentos en los que el salario medio es mayor o igual que la media de todos los salarios.**

Podemos incluir una cláusula ORDER BY al final de la orden y en ella podremos especificar funciones de grupo, columnas de GROUP BY o su combinación.

La evaluación de las cláusulas en tiempo de ejecución se realiza en el siguiente orden:

|  |
| --- |
|  |

* **Obtener la suma de salarios, el salario máximo y el mínimo para cada departamento. Los resultados deben estar formateados (con parte decimal, separador de miles, etc .)**
* **Calcular el número de empleados que realizan cada oficio en cada departamento. Los datos que se viualizarán serán: departamento, oficio y número de empleados. Será necesario agrupar por departamento y dentro de cada departamento, por oficio:**
* **Buscar el número máximo de empleados que hay en algún departamento.**
* **Obtener el nombre de los departamentos que tengan más de 4 personas trabajando. (El resultado será ‘INVESTIGACIÓN’ y ‘VENTAS’)**

**6.3 Consultas multitabla**

Hasta ahora las consultas realizadas sólo hacían referencia a una tabla, pero la sentencia SELECT permite consultar datos de varias tablas en una sola orden. En este caso, los nombres de las tablas se indicarán a continuación de la palabra FROM separados por comas.

Se pueden realizar dos tipos de operaciones al consultar datos de varias tablas: composiciones internas y externas.

**Composición interna**

Si se combinan dos o más tablas sin ninguna restricción, el resultado será el producto cartesiano de dichas tablas, es decir, se obtienen todas las combinaciones entre las filas de esas tablas. Si combinamos de esta forma dos tablas de 10 filas cada una, obtendremos 10 x 10 filas en el resultado. Cada fila de la primera tabla se combina con todas las de la segunda y así sucesivamente, tengan o no relación las filas.

De alguna manera debemos especificar un criterio que discrimine qué filas de una tabla están relacionadas con filas de la otra. A esta operación se le llama **composicion, asociación o join de tablas.**

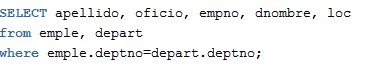
Para realizar una composicion interna se parte de un producto cartesiano y se eliminan las filas que no cumplen la condición de composición. En esta operación se combinan campos que tienen el mismo valor en las tablas



Las reglas que hay que seguir son las siguientes:

* Pueden combinarse tantas tablas como se desee.
* En la orden SELECT se pueden citar columnas de todas las tablas.
* Si hay columnas con el mismo nombre en distintas tablas, será necesario especificar a qué tabla pertenecen de la forma: nombreTabla.nombreColumna.
* El criterio que se siga para la composición, se especificará en la cláusula WHERE, si se omite, el resultado será el producto cartesiano.

Ejemplo: A partir de las tablas EMPLE y DEPART obtener apellido, oficio, código de empleado, nombre del departamento y localidad. Ambas tablas tienen en común el campo DEPTNO por el que se pueden combinar.



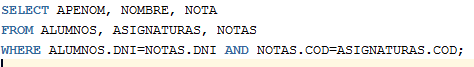
Todos los empleados del departamento 10 serán emparejados con los datos, nombre y localidad, del departamento 10, los del 20 con el 20 y así sucesivamento. Si se omite la cláusula WHERE se obtiene el producto cartesiano, cada fila de EMPLE se emparejará con cada fila de DEPART, se obtendrán 14\*4 = 56 filas.

Ejemplo de combinación de 3 tablas: partimos de las tablas ALUMNOS, ASIGNATURAS y NOTAS.

ALUMNOS1 contiene los datos de los alumnos , ASIGNATURAS1 contiene los nombres de las asignaturas con sus códigos correspondientes y NOTAS1 contiene las notas de cada alumno en cada asignatura. Esta última tabla se relaciona con la tabla ALUMNOS1 por la columna DNI y con la tabla ASIGNATURAS1 con la columna COD.

|  |  |  |
| --- | --- | --- |
|  |  |  |

Realizar una consulta para obtener el nombre de un alumno, el nombre de sus asignaturas y su nota en ellas.



Realizar una consulta para obtener el nombre de los alumnos matriculados en ‘FOL’.



* **Visualizar los nombres de los alumnos que tengan una nota entre 7 y 8 en la asignatura de ‘FOL’.**
* **Visualizar los nombres de las asignaturas que no tengan suspensos.**

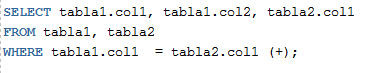
Se puede combinar una tabla consigo misma pero será necesario poner un alias a uno de los nombres de la tabla que se van a repetir.

Ejemplo: Visualizar el nombre de cada empleado y el nombre de su director.



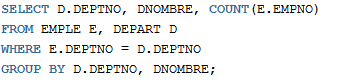
**Composición externa**

Existe una variedad de combinación de tablas que se conoce como **composición interna u OUTER JOIN** y que permite seleccionar algunas filas de una tabla aunque éstas no tengan correspondencia con las filas de la otra tabla con la que se combina. El formato es el siguiente:

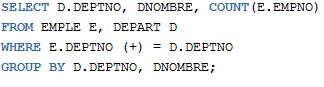


Se seleccionan todas las filas de la tabla1 aunque no tengan correspondencia con las filas de tal tabla2. Se coloca el símbolo (+) detrás de la columna de la tabla2 (que es la tabla donde no se encuentran las filas) en la cláusula WHERE. Se obtendrá una fila con las columnas de tabla1 y el resto de columnas de la tabla2 se rellena con NULL.

Ejemplo: partiendo de las tablas EMPLE y DEPART, podemos comprobar que en la tabla EMPLE el departamento 40 no existe ya que no tiene empleados, pero en la tabla DEPART si se encuentra. Queremos realizar una consulta donde se visualice el número de departamento, el nombre y número de empleados que tiene. Para hacer la consulta combinamos ambas tablas:

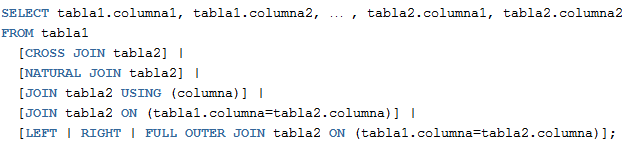


En el resultado no aparece el departamento 40, se pierde. Para que aparezca, será necesario realizar un OUTER JOIN indicando el operador (+) en la condición WHERE, a continuación de la columna de la tabla en la que no aparece el departamento 40 (en la tabla EMPLE).



* **Comprobar que ocurre si en lugar de COUNT (E.EMPNO) colocamos COUNT (\*) en la sentencia select anterior.**
* **¿Y si en la sentencia SELECT en lugar de D.DEPTNO ponemos E.DEPTNO ?**

En la versión **SQL de 1999** se especifica una nueva sinaxis para consultar varias tablas. La sintaxis de esta combinación es la siguiente:



CROSS JOIN: crea un producto cartesiano de las filas de ambas tablas .

NATURAL JOIN: detecta automaticamente las claves de unión, basándose en el nombre de la columna que coincide en ambas tablas. Esta característica funciona aunque no estén definidas las claves primarias o ajenas.

JOIN USING: las tablas pueden tener más de un campo para relacionar y no siempre queremos que se relacionen por todos. Esta cláusula permite establecer relaciones indicando qué campo o campos comunes se quieren utilizar.

JOIN ON: se utiliza para unir tablas en las que los nombres de las columnas no coinciden.

FULL OUTER JOIN: permite omitir el símbolo anterior (+) que se utilizaba en la 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.

RIGHT 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.

**6.4 Unión, Intersección y Diferencia**

Los operadores relacionales **UNION, INTERSECT y MINUS** son operadores de conjuntos. Los conjuntos son las filas resultantes de cualquiere sentencia SELECT válida que permita combinar los resultados de varias SELECT para obtener un único resultado.

Los operadores de conjuntos pueden encadenarse. Los conjuntos se evaluan de izquierda a derecha y para forzar precedencia se pueden utilizar paréntesis.

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

* Las columnas de las dos consultas se relacionan en orden, de izquierda a derecha.
* Los nombres de columna de la primera sentencia SELECT no tienen por qué ser los mismos que los nombres de columna 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.

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 que están en una lista y no en la otra (MINUS). El formato de la orden SELECT con estos operadores es:



**Operador UNION**

Este operador combina resultados de dos consultas. Permite añadir el resultado de un SELECT a otro SELECT. Las filas duplicadas que aparecen se reducen a una única fila. El formato es:



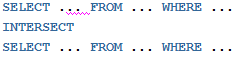
Ejemplo:



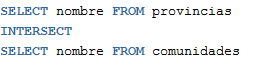
El resultado será una tabla que contendrá nombres de provincias y de comunidades. Se crea una sola tabla con registros incluidos en cualquiera de las consultas. Se pueden mostrar los registros duplicados especificando **UNION ALL**

**Operador INTERSECT**

Devuelve las filas que son iguales en ambas consultas. Todas las filas duplicadas serán eliminadas antes de la generación del resultado final. Su formato es:



En el ejemplo siguiente se mostrarán los nombres coincidentes en provincias y comunidades.



Ejemplo: visualizar los tipos y modelos de piezas que se encuentran en los almacenes 1 y 2.



**Operador MINUS**

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

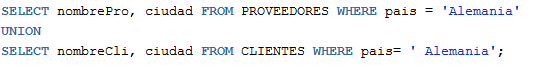


Ejemplo: tipos y modelos de piezas que se encuentren en el almacén 1 y no en el 2.

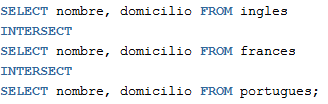


**Ejemplos:**

Obtener los nombres y ciudades de todos los proveedores y clientes de Alemania



Una academia de idiomas da clases de inglés, francés y portugués y guarda los datos de los alumnos en tres tablas cuyo nombre coincide con el idioma. Averiguar el nombre y domicilio de todos los alumnos que cursan los tres idiomas para enviarles información sobre los exámenes.



Averiguar los nombres y domicilios de los alumnos que cursan inglés y no portugués.

