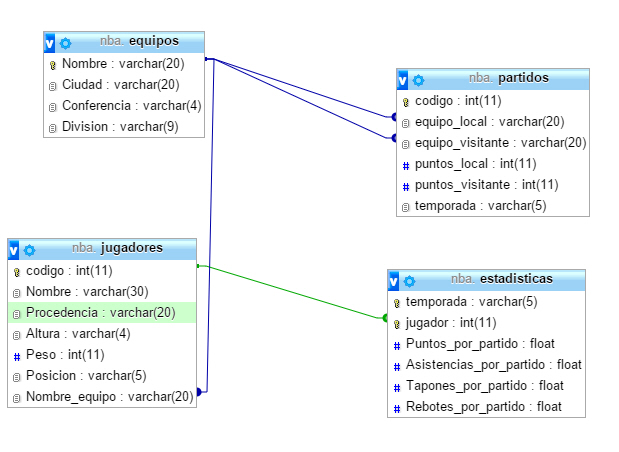
**UD6. SQL III. Agrupaciones y consultas multitabla**

**1. Agrupaciones**

Para estudiar esta parte introduciremos una nueva base de datos. La base de datos *NBA*.



Antes de entrar en materia con la teoría de *agrupaciones*, a modo de ejemplo se muestran algunas consultas resúmenes:

**#Consulta 1**

¿Cuánto pesa el jugador más pesado de la NBA?

SELECT max(peso)

FROM jugadores

**#Consulta 2**

¿Cuánto mide el jugador más bajito de la NBA?

SELECT min(altura)

FROM jugadores

**#Consulta 3**

¿Cuántos jugadores tienen los Lakers?

SELECT count(\*)

FROM jugadores

WHERE Nombre\_equipo= “Lakers”;

**#Consulta 4**

¿Cuánto pesan de media los jugadores de los Blazers?

SELECT avg(peso)

FROM jugadores

WHERE Nombre\_equipo=”Blazers”;

Con las consultas de resumen se pueden realizar agrupaciones de registros. Se denomina agrupación de registros a un conjunto de registros que cumplen que tienen una o varias columnas con el mismo valor. Por ejemplo, en la tabla **vehículos** que se muestra a continuación:

|  |  |  |
| --- | --- | --- |
| **matrícula** | **modelo** | **marca** |
| 1129FGT | Ibiza gt | seat |
| 1132GHT | Leon tdi 105cv | seat |
| 2242DYD | Corolla g6 | toyota |
| 7423FZY | Coupe | hyundai |
| 3447BYD | A3 tdi 130cv | audi |

En esta tabla hay dos registros cuya marca=’seat’. Se puede agrupar estos dos registros formando un único grupo, de tal manera que el grupo ‘seat’ tiene los modelo ibiza gt y león tdi 105cv. A este grupo de registros se le puede aplicar una función de columna para realizar determinados cálculos, por ejemplo, contarlos:

SELECT marca, count(\*)

FROM vehículos

GROUP BY marca;

La consulta devolverá el siguiente resultado:

|  |  |
| --- | --- |
| marca | count(\*) |
| audi | 1 |
| hyundai | 1 |
| seat | 2 |
| toyota | 1 |

En este caso, si se agrupa (GROUP BY) por el campo marca, salen 4 grupos (audi, hyundai, seat y toyota). La función de columna, cuando se agrupa por un campo, actúa para cada grupo. En este caso, para cada grupo se ha contado el número de registros que tiene. En el caso de seat, cuenta los 2 antes mencionados.

La sintaxis de la sentencia SELECT con GROUP BY queda como sigue:

SELECT [DISTINCT], columna(s)

FROM tabla

WHERE filtro

GROUP BY column(a)

A continuación, a modo de ejemplo, se muestran algunas consultas con grupos y funciones resumen.

#Consulta 1

¿Cuánto pesa el jugador más pesado de cada equipo?

SELECT Nombre\_equipo, max(peso)

FROM jugadores

GROUP BY Nombre\_equipo;

|  |  |
| --- | --- |
| **Nombre\_equipo** | **max(peso)** |
| 76ers | 250 |
| Bobcats | 266 |
| Bucks | 260 |
| …… | …. |
| Warriors | 250 |
| Wizards | 263 |

#Consulta 2

¿Cuántos equipos tiene cada conferencia en la NBA?

SELECT count(\*), conferencia

FROM equipos

GROUP BY conferencia;

|  |  |
| --- | --- |
| **count(\*)** | **conferencia** |
| 15 | East |
| 15 | West |

#Consulta 3

¿Cuánto pesan de media los jugadores de España, Francia e Italia?

SELECT avg(peso), procedencia

FROM jugadores

WHERE procedencia IN (‘Spain’, ‘Italy’, ‘France’)

GROUP BY procedencia;

|  |  |
| --- | --- |
| **avg(peso)** | **procedencia** |
| 219.2500 | France |
| 250.000 | Italy |
| 208.6000 | Spain |

**IMPORTANTE:** se observa que para cada agrupación, se ha seleccionado también el nombre de la columna por la cual se agrupa. Esto no es posible si no se incluye el GROUP BY.

**Filtros de grupos**

Los filtros de grupos deben realizarse mediante el uso de la cláusula HAVING puesto que WHERE actúa antes de agrupar los registros. Es decir, si se desea filtrar resultados calculados mediante agrupaciones se debe usar la siguiente sintaxis:

SELECT [DISTINCT], columna(s)

FROM tabla

WHERE filtro

GROUP BY column(a)

HAVING filtro\_grupos

HAVING aplica los mismos filtros que la cláusula WHERE. A continuación se ilustran algunos ejemplos:

#Consulta1

Seleccionar los equipos de la nba cuyos jugadores pesen de media más de 228 libras.

SELECT Nombre\_equipo, avg(peso)

FROM jugadores

GROUP BY Nombre\_equipo

HAVING avg(peso)>228

//Dicho de otra forma. Agrupamos los jugadores por equipos y mostramos únicamente aquellos equipos cuya media de peso es mayor que 228.

El resultado de la consulta será:

|  |  |
| --- | --- |
| **Nombre\_equipo** | **avg(peso)** |
| Suns | 238.8462 |
| Wizards | 229.6923 |
| Lakers | 230.000 |
| Jazz | 230.0714 |
| Knicks | 235.4667 |

#Consulta 2

Seleccionar qué equipos de la nba tienen más de un jugador español.

SELECT Nombre\_equipo, count(\*)

FROM jugadores

WHERE procedencia= ‘Spain’ // condición sobre tuplas individuales

GROUP BY Nombre\_equipo

HAVING count(\*)>1; // condición sobre los grupos realizados

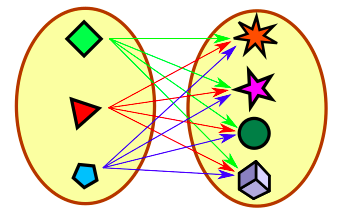
|  |  |
| --- | --- |
| **Nombre\_equipo** | **count(\*)** |
| Raptors | 2 |

**2. Consultas multitabla: composiciones internas y externas**

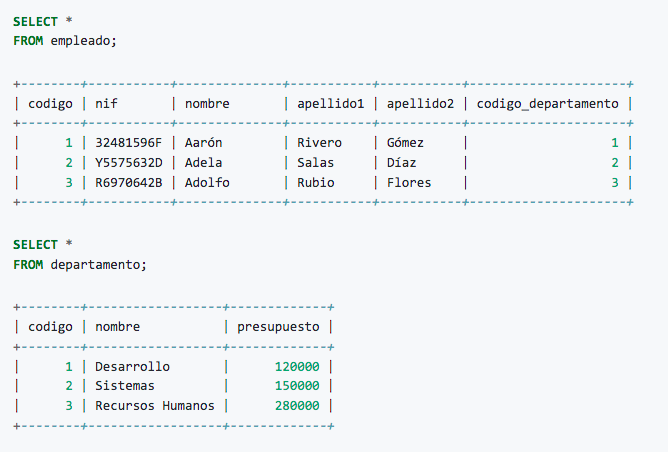
Las consultas multitabla nos permiten consultar información en más de una tabla. La única diferencia respecto a las consultas sencillas es que vamos a tener que especificar en la cláusula FROM cuáles son las tablas que vamos a usar y cómo las vamos a relacionar entre sí.

Para entender cómo se realizan las consultas multitabla, primero se debe entender en qué consiste la realización del producto cartesiano.

El **producto cartesiano** de dos conjuntos, es una operación que consiste en obtener otro conjunto cuyos elementos son **todas las parejas que pueden formarse entre los dos conjuntos**. Por ejemplo, tendríamos que coger el primer elemento del primer conjunto y formar una pareja con cada uno de los elementos del segundo conjunto. Una vez hecho esto, repetimos el mismo proceso para cada uno de los elementos del primer conjunto.



Como ejemplo utilizaremos la base de datos “Gestión de empleados”. Suponemos los siguientes valores en las tablas empleado y departamento:

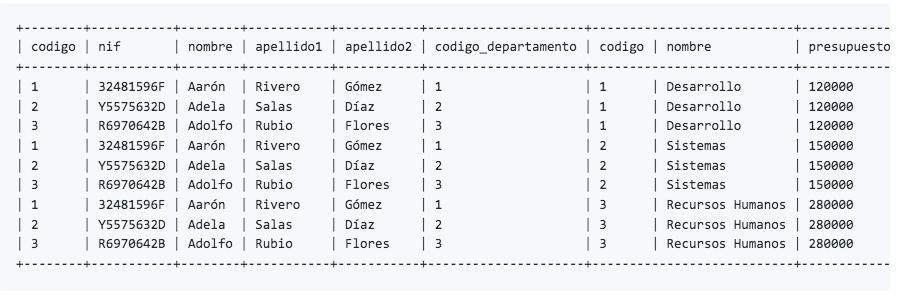


El producto cartesiano de las dos tablas se realiza con la siguiente consulta.

**SELECT** \*

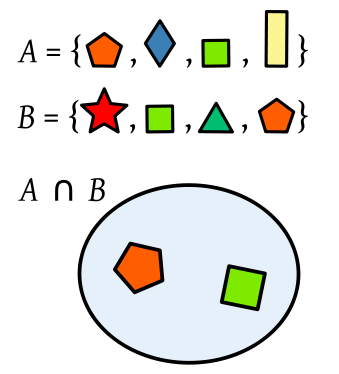
**FROM** empleado, departamento;

El resultado sería el siguiente:



Una vez que ya sabemos cómo se realiza el producto cartesiano, ahora se debe entender la operación de interseccion entre dos tablas.

La **intersección de dos conjuntos** es una operación que resulta en otro conjunto que contiene **sólo los elementos comunes** que existen en ambos conjuntos.



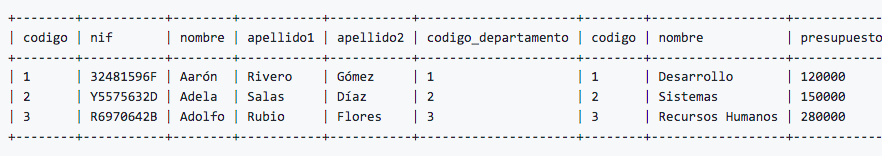
Para poder realizar una **operación de intersección** entre las dos tablas debemos utilizar la cláusula WHERE para indicar la columna con la que queremos relacionar las dos tablas. Por ejemplo, para obtener un listado de los empleados y el departamento donde trabaja cada uno podemos realizar la siguiente consulta:

**SELECT** \*

**FROM** empleado, departamento

**WHERE** empleado.codigo\_departamento = departamento.codigo;

El resultado sería el siguiente:



**Nota:** Tenga en cuenta que con la **operación de intersección** sólo obtendremos los elementos de existan en ambos conjuntos. Por lo tanto, en el ejemplo anterior puede ser que existan filas en la tabla empleado que no aparecen en el resultado porque no tienen ningún departamento asociado. (Esto quiere decir que en la columa codigo\_departamento de la tabla empleado hay un NULL). Los NULL no se relacionan.

**Composiciones internas (JOIN y NATURAL JOIN)**

Bien, ahora es el momento de utilizar el la sentencia JOIN para realizar consultas en más de una tabla:

// La operación JOIN consiste en realizar el producto cartesiano más un filtro (la intersección).

**SELECT** \*

**FROM** empleado **JOIN** departamento **ON**

empleado.codigo\_departamento = departamento.codigo;

**NOTA:** Tenga en cuenta que **si olvidamos incluir la cláusula ON obtendremos el producto cartesiano de las dos tablas**.

//Otra posibilidad es utilizar la notación de SQL I

**SELECT** \*

**FROM** empleado, departamento

**WHERE** empleado.codigo\_departamento = departamento.codigo;

NATURAL JOIN realiza la misma función que JOIN y se utiliza únicamente cuando las columnas realizadas tienen el mismo nombre.

Veamos un ejemplo:

**SELECT** \*

**FROM** empleado **NATURAL** **JOIN** departamento;

Se presentan algunos ejemplos para que quede más claro el uso de JOIN:

Devuelve el nombre del departamento donde trabaja el empleado que tiene el nif 38382980M.

**SELECT** departamento.nombre

**FROM** empleado **JOIN** departamento **ON** empleado.codigo\_departamento = departamento.codigo

**WHERE** empleado.nif= “38382980”;

Devuelve el nombre del departamento donde trabaja el empleado Pepe Ruiz Santana.

**SELECT** departamento.nombre

**FROM** empleado **JOIN** departamento

**ON** empleado.codigo\_departamento = departamento.codigo

**WHERE** empleado.nombre LIKE “Pepe Ruiz Santana”;

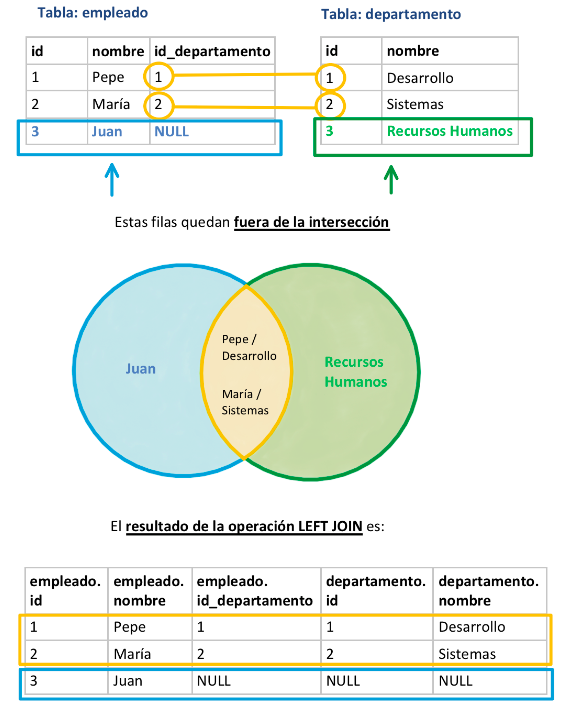
**Composiciones externas (LEFT JOIN, RIGHT JOIN y FULL OUTER JOIN)**

**Ejemplo de LEFT JOIN:**

**SELECT** \*

**FROM** empleado **LEFT** **JOIN** departamento **ON** empleado.codigo\_departamento = departamento.codigo;

Esta consulta devolverá todas las filas de la tabla que hemos colocado a la izquierda de la composición, en este caso la tabla empleado. Y relacionará las filas de la tabla de la izquierda (empleado) con las filas de la tabla de la derecha (departamento) con las que encuentre una coincidencia. Si no encuentra ninguna coincidencia, se mostrarán los valores de la fila de la tabla izquierda (empleado) y en los valores de la tabla derecha (departamento) donde no ha encontrado una coincidencia mostrará el valor NULL.

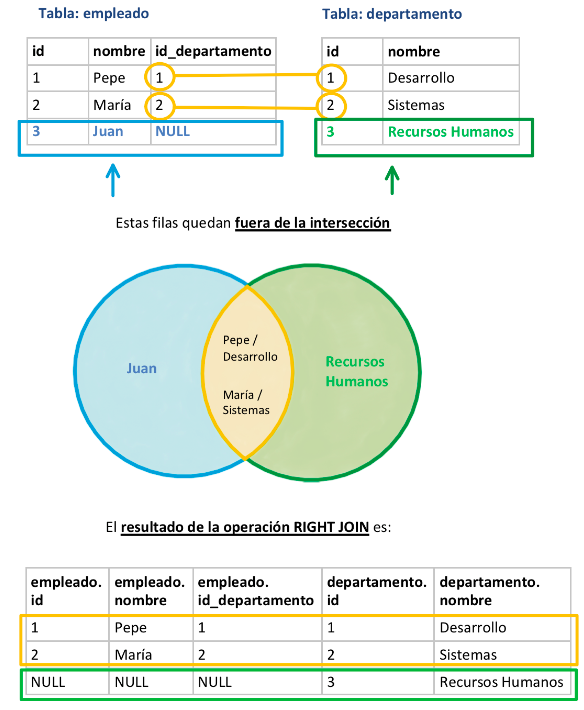


**Ejemplo de RIGHT JOIN:**

**SELECT** \*

**FROM** empleado **RIGHT** **JOIN** departamento **ON** empleado.codigo\_departamento = departamento.codigo

Esta consulta devolverá todas las filas de la tabla que hemos colocado a la derecha de la composición, en este caso la tabla departamento. Y relacionará las filas de la tabla de la derecha (departamento) con las filas de la tabla de la izquierda (empleado) con las que encuentre una coincidencia. Si no encuentra ninguna coincidencia, se mostrarán los valores de la fila de la tabla derecha (departamento) y en los valores de la tabla izquierda (empleado) donde no ha encontrado una coincidencia mostrará el valor NULL.



**Ejemplo de FULL OUTER JOIN:**

La composición FULL OUTER JOIN **no está implementada en MySQL**, por lo tanto para poder simular esta operación será necesario hacer uso del operador UNION, que nos realiza la union del resultado de dos consultas.

El resultado esperado de una composición de tipo FULL OUTER JOIN es obtener la intersección de las dos tablas, junto las filas de ambas tablas que no se puedan combinar. Dicho con otras palabras, el resultado sería el equivalente a realizar la union de una consulta de tipo LEFT JOIN y una consultas de tipo RIGHT JOIN sobre las mismas tablas.

**SELECT** \*

**FROM** empleado **LEFT** **JOIN** departamento **ON** empleado.codigo\_departamento = departamento.codigo

**UNION**

**SELECT** \*

**FROM** empleado **RIGHT** **JOIN** departamento **ON** empleado.codigo\_departamento = departamento.codigo;

**Consideraciones**

**El orden en las tablas no afecta al resultado final**

Estas dos consultas devuelven el mismo resultado:

**SELECT** \*

**FROM** empleado **INNER JOIN** departamento **ON** empleado.codigo\_departamento = departamento.codigo;

**SELECT** \*

**FROM** departamento **INNER** **JOIN** empleado **ON** empleado.codigo\_departamento = departamento.codigo;

Podemos usar alias en las tablas

**SELECT** \*

**FROM** empleado **AS** e **INNER** **JOIN** departamento **AS** d **ON** e.codigo\_departamento = d.codigo;

**SELECT** \*

**FROM** empleado e **INNER** **JOIN** departamento d **ON** e.codigo\_departamento = d.codigo;