Índice de la Unidad

1. Introducción

En la unidad anterior ampliamos las posibilidades a la hora de consultar datos gracias a las funciones. Con ellas podíamos procesar la información de una columna según nuestras necesidades (contar el número de palabras del título de un libro, redondear el valor de un salario, formatear apropiadamente una fecha a través de una conversión de tipos, etc.) o bien obtener información estadística de un conjunto de columnas (obtener la media de los salarios, contar el número de empleados del departamento 20, etc.).

En esta unidad veremos nuevas cláusulas que los amplían, aún más, nuestras posibilidades de consultas. Ahora, en lugar de obtener la media de salarios de un departamento, podemos obtener la media de salarios de todos los departamentos a través de una única consulta.
Adicionalmente, veremos una nueva forma de combinar las tablas en una consulta y de cómo unir/intersecar/restar los resultados de varias consultas en una única tabla resultado.

2. Agrupación de elementos: GROUP BY y HAVING

Hasta ahora, el funcionamiento de una consulta SELECT se basaba en buscar las tablas necesarias (cláusula FROM), de cada tabla sólo nos quedamos con aquellas filas que cumplan la condición especificada (cláusula WHERE), a continuación ordenábamos las filas seleccionadas según el criterio indicado (cláusula ORDER BY) y, por último, no quedamos sólo con un determinado conjunto de columnas (cláusula SELECT) de las filas seleccionadas y ordenadas. Si sobre alguna de las columnas seleccionadas hay aplicada una función de grupo, el resultado de dicha función se ejecuta al final.
Sin embargo, hay ocasiones en que después de quedarnos con las filas que cumplen ciertos criterios queremos agrupar éstas en base a cierto criterio (por ejemplo el número de departamento) y así poder conseguir cosas tan interesantes como:

- Obtener el sueldo medio por departamento.
- Obtener la edad máxima de cada departamento.
- Obtener el número de empleados de aquellos oficios con más de 5 trabajadores.
- Etc.

Para conseguir la agrupación anterior se hace uso de una cláusula que no habíamos visto hasta ahora: GROUP BY.
Para obtener el sueldo medio por departamento la consulta quedaría de la siguiente manera:

```
SELECT dept_no, AVG(salario)
FROM Emple
GROUP BY dept_no;
```

Si además queremos aplicar una condición a los grupos, podemos hacer uso de la cláusula HAVING. De esta manera podríamos obtener el sueldo medio de aquellos departamentos donde se gane más de 2000€ de media.

```
SELECT dept_no, AVG(salario)
FROM Emple
GROUP BY dept_no
HAVING AVG(salario)>2000;
```

Como se puede observar, esto no lo podíamos conseguir con los conocimientos que teníamos antes. Hubiese sido necesario realizar una consulta por cada departamento mientras que ahora lo podemos hacer todo con una sola consulta.
La sentencia SELECT con las dos nuevas cláusulas quedaría de la siguiente manera:

```
SELECT [ALL | DISTINCT]
[expre_col1, … , expre_colN | *]
FROM nom_tabla1 [, … , nom_tablaN]
[WHERE condición]
[GROUP BY expre_col1, …, expre_colM]
[HAVING condición_de_grupo]
[ORDER BY expre_col1 [DESC | ASC] [, expre_col2 [DESC | ASC]]…]
```

Si se utiliza la cláusula GROUP BY hay que tener en cuenta una serie de consideraciones importantes:

- Dentro de la cláusula SELECT sólo pueden aparecer expresiones de columna que cumplan los siguientes criterios: ser una constante, una función de grupo (SUM, COUNT, AVG, etc.) o una expresión de columna usada en la cláusula GROUP BY.
- La cláusula HAVING se utiliza para, una vez hecho los grupos, decidir cuales se van a mostrar y cuales no. Por lo tanto, en la condición sólo pueden aparecer expresiones con funciones de grupo y expresiones con las columnas por las que se agrupa.

El proceso para evaluar una consulta con estas dos nuevas cláusulas es el siguiente:

1. Se cogen las tablas indicadas en la cláusula FROM.
2. De las tablas indicadas, sólo nos quedamos con aquellas filas que cumplan con la condición especificada en la cláusula WHERE.
3. Agrupamos las filas resultantes en base a las expresiones de columnas especificadas en la cláusula GROUP BY.
4. Nos quedamos sólo con aquellos grupos que cumplen la condición indicada en la cláusula HAVING.
5. De las filas agrupadas, sólo nos quedamos con las columnas expresadas en la cláusula SELECT.
6. Si la SELECT lleva el DISTINCT, entonces eliminamos filas repetidas.
7. Por último, ordenamos las filas que se han obtenido en base a las expresiones de columna indicadas en la cláusula ORDER BY.

Ejemplo 1:
Visualiza, a partir de la tabla Emple, el número de empleados de cada departamento:

```
SELECT dept_no, COUNT(*)
FROM Emple
GROUP BY dept_no;
```

Vamos a modificar la consulta anterior para mostrar el número de empleados de aquellos departamentos con más de 4 empleados ordenando los resultados de manera descendente por el número de empleados:

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

Si en lugar de mostrar el número de departamento quisiéramos mostrar el nombre de cada departamento, tendríamos que modificar la consulta anterior de la siguiente manera:

```
SELECT D.dnombre, COUNT(*)
FROM Emple E, Depart D
WHERE E.dept_no = D.dept_no
GROUP BY D.dnombre
HAVING COUNT(*) > 4
ORDER BY COUNT(*) DESC;
```

Ejemplo 2:
A partir de la tabla Emple, obtén la suma de salarios, el salario máximo y el salario mínimo de cada departamento. La salida debe estar formateada atendiendo al siguiente criterio: Las unidades de millar tienen que estar separadas por un punto y sólo deben mostrarse dos decimales:

```
SELECT dept_no "Nºdept",
TO_CHAR(SUM(salario), '99G999D99') "Suma",
TO_CHAR(MAX(salario), '99G999D99') "Máximo",
TO_CHAR(MIN(salario), '99G999D99') "Mínimo"
FROM Emple
GROUP BY dept_no;
```

Ejemplo 3:
De la tabla Emple, muestra, para cada número de departamento, cuántas personas desempeñan cada oficio. La salida tendrá, por tanto, tres columnas: número de departamento, oficio y número de empleados. Ordenar los resultados por el número de departamento y de manera ascendente:

```
SELECT dept_no, oficio, COUNT(*) "Nºpersonas"
FROM Emple
GROUP BY dept_no, oficio
ORDER BY dept_no;
```

Si en lugar de mostrar el número de personas que desempeñan cada oficio en todos los departamentos sólo lo quisiéramos del departamento 20 lo haríamos de la siguiente manera:

```
SELECT dept_no, oficio, COUNT(*) "Nºpersonas"
FROM Emple
WHERE dept_no = 20
GROUP BY dept_no, oficio;
```

3. Combinación externa: OUTER JOIN

En la Unidad 9 (consultas simples) ya se estudió cómo realizar consultas sobre varias tablas, combinándolas para generar una tabla los resultados deseados. Ahora vamos a añadir un nueva variedad de combinación de tablas llamada OUTER JOIN, la cual nos 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:

```
SELECT t1.col1, t1.col2, t2.col1
FROM tabla1 t1, tabla2 t2
WHERE t1.col1 = t2.col1(+)
```

Si no hubiéramos añadido el símbolo (+), la SELECT anterior funcionaría tal y como la hemos estudiado: seleccionaría todas las filas de la tabla1 que tuvieran correspondencia con filas de la tabla2 en base a la condición que `t1.col1=t2.col1`.
Las filas de la tabla1 que no tengan ninguna correspondencia con filas de la tabla2 simplemente no se seleccionan.

Sin embargo, si añadimos el símbolo (+), aquellas filas de la tabla1 que no tengan correspondencia con filas de la tabla2 sí que se añadirían. Se obtendría una fila con todas las columnas de la tabla1 y el resto de columnas de la tabla2 se rellenarían con valores nulos.

Ejemplo 4:
Dada las tablas Depart y Emple, obtener el número de cada departamento, su nombre y el número de empleados que trabajan en dicho departamento:

```
SELECT D.dept_no, D.dnombre, COUNT(E.emp_no)
FROM Depart D, Emple E
WHERE D.dept_no=E.dept_no
GROUP BY D.dept_no, D.dnombre
ORDER BY dept_no;
```

Como podemos observar, como no hay empleados en el departamento 40, éste no se muestra en la consulta anterior. Si lo que pretendemos es mostrar TODOS los departamentos con el número de empleados de cada uno, tendremos que modificar la consulta anterior.

Haciendo uso del OUTER JOIN, aquellas filas de la tabla Depart que no tengan correspondencia con alguna fila de la tabla Emple SÍ que seseleccionarían:

```
SELECT D.dept_no, D.dnombre, COUNT(E.emp_no)
```

```
FROM Depart D, Emple E
WHERE D.dept_no=E.dept_no(+)
GROUP BY D.dept_no, D.dnombre
ORDER BY dept_no;
```

Hay que fijarse que la cosa funciona porque se ha puesto un COUNT(E.emp_no) en lugar de un COUNT(*). Recordemos que si en la función COUNT en lugar de poner un * ponemos una expresión, contará aquellas filas donde la expresión no sea nula.

4. Operadores de conjuntos

Las consultas SQL devuelven una nueva tabla compuesta por un conjunto de filas. Cada fila, a su vez, está formada por una serie de columnas. Luego en definitiva, las consultas SQL nos devuelven un conjunto de datos.
Hay ocasiones en que el conjunto de datos que se nos pide obtener podría calcularse de manera sencilla a través de una operación de conjuntos entre dos consultas simples.
Supongamos, por ejemplo, que tenemos ya una consulta que nos devuelve un listado de centros de la ESO, y otra consulta que nos devuelve el listado de centros de ESO y Ciclos Formativos. Imaginemos que somos una editorial que queremos mandar información a los centros. Pues bien, es posible que queramos mandar
propaganda a todos los centros (UNION), a los centros sólo de la ESO (INTERSECT) o a los centros sólo de Ciclos Formativos (MINUS).

El formato de cualquiera de estos operadores de conjunto es el siguiente:

```
SELECT … FROM … WHERE … ← Consulta1
Operador_conjunto
SELECT … FROM … WHERE … ← Consulta2
```

Es importante resaltar que los operadores de conjuntos exigen que ambas conjuntos (es decir, el resultado de nuestras consultas) cumplan una serie de condiciones:

1. Deben tener el mismo número de columnas.
2. Las columnas deben tener tipos compatibles entre sí.
3. Como nombre de columnas del resultado, se cogen los nombres de columna establecidos en la Consulta1.

Aunque ya se ha comentado, es conveniente resaltar que los operadores de conjuntos aparecen como una ayuda para hacer más simples y legibles nuestras consultas. En muchas ocasiones no es estrictamente necesario utilizarlos pero, si lo hacemos, conseguimos dividir una consulta compleja en una operación simple
entre dos consultas sencillas.

4.1. Operador UNION

Es el operador UNION une dos conjuntos en un conjunto más grande (conjunto1 È conjunto2). Su formato es:

```
SELECT … FROM … WHERE …
UNION
SELECT … FROM … WHERE …
```

Es importante comentar que el operador UNIÓN no genera filas repetidas. A fin de cuentas, esto tiene sentido ya que el objetivo del operador es un nuevo conjunto que contenga los elementos de los dos conjunto unidos. Los elementos repetidos sólo se añaden una vez.
Si se desea que UNION genere filas repetidas entonces hay que usar el operador UNION ALL.

Ejemplo 5:
Vamos a ver un ejemplo en el que tenemos tres tablas con información de alumnos. La tabla Alum contiene los datos de los alumnos matriculados este año en el centro. La tabla Nuevos al-

macena los datos de los alumnos que han reservado plaza para el próximo curso. Por último, la tabla Antiguos contiene los nombres de antiguos alumnos del centro. La estructura de las tablas es la misma para las tres.

A partir de esta información, queremos obtener el nombre de los alumnos que, hasta la fecha, han pasado por el centro:

```
SELECT nombre FROM Alum
UNION
SELECT nombre FROM Antiguos;
```

Como era lógico, lo que se nos está pidiendo es la unión entre los antiguos alumnos y los alumnos actuales.

4.2. Operador INTERSECT

El operador INTERSECT calcula la intersección entre dos conjuntos. Es decir, busca aquellas filas que se encuentran en el resultado de ambas consultas (conjunto1 INTERSECT conjunto2). Su sintaxis es como sigue:

```
SELECT … FROM … WHERE …
INTERSECT
SELECT … FROM … WHERE …
```

Ejemplo 6:
Obtener los nombres de aquellos alumnos que estuvieron matriculados hace un tiempo en el centro y que han decidido inscribirse para el curso que viene:

```
SELECT nombre FROM Antiguos
INTERSECT
SELECT nombre FROM Nuevos;
```

4.3. Operador MINUS

El operador MINUS da como resultado aquellas filas devueltas por la consulta1 que no se encuentras en la consulta2 (consulta1 - consulta2). Las filas que hubieran duplicadas del primer conjunto se eliminarían antes de que empiece la comparación con el otro conjunto. Su sintaxis es la siguiente:

```
SELECT … FROM … WHERE …
MINUS
SELECT … FROM … WHERE …
```

Ejemplo 7:
Obtener el nombre y localidad de aquellos alumnos que están actualmente en el centro y que nunca estuvieron anteriormente en él:

```
SELECT nombre, localidad FROM Alum
MINUS
SELECT nombre, localidad FROM Antiguos;
```

Nota: Algunos de los ejemplos vistos pueden realizarse sin necesidad de los operadores de conjuntos. Podéis probar a intentar hacer las consultas haciendo uso del operador IN y valorar si la comodidad y utilidad de los operadores de conjuntos.

4.4. Reglas para la utilización de operadores de conjuntos

Al principio de este punto ya se especificaron las condiciones para que se pudieran utilizar los operadores de conjuntos.
Sin embargo, ahora vamos a comentar algunos detalles adicionales que nos ayudarán a entender mejor su funcionamiento:

1. Los operadores de conjuntos pueden encadenarse.

2. Los conjuntos se evalúan de izquierda a derecha. Para forzar precedencia se pueden utilizar paréntesis.
3. Los nombres de columna de la primera consulta no tienen que coincidir con los de la segunda consulta. Eso sí, se tomará como nombre de columnas los nombres de la primera consulta.

Ejemplo 7:
Queremos seleccionar el nombre de aquellos alumnos que estén matriculados este año, que se vayan a matricular el año que viene pero que nunca hayan estado antes en este centro:

```
SELECT nombre FROM Alum INTERSECT SELECT nombre FROM Nuevos
MINUS
SELECT nombre FROM Antiguos;
```