Consultas Complejas

Sitio: Agencia de Aprendizaje a lo largo de la Vida
Curso: Administración de Base de Datos 1° G
Libro: Consultas Complejas

Imprimido por: Sebastian Puche
Día: domingo, 13 de octubre de 2024, 18:44

Tabla de contenidos

Introducción

Cuando hablamos de consultas complejas pensamos que son difíciles, y en consecuencia imposibles de resolver.

En verdad son complejas porque se aplica todo lo aprendido hasta este momento.

Entonces la consulta puede tener:

- Subconsultas: ubicadas tanto en el select , en el from o en el where; como una combinación de todas o alguna en particular.
- Combinación interna de join.
- Combinación externa de join.
- Mezcla de combinaciones de join.

Para resolverlas con éxito te sugiero comenzar de adentro para afuera.

En los próximos capítulos veremos ejemplos de consultas complejas.

Función de función

En el video de apertura de la semana mencionamos el caso de los meses de cumpleaños.

Volvé a ver el video haciendo clic acá .

¿Cuál es el mes que hay más cumpleaños en el grupo de estudio de la cursada?

1. Continuemos con el tema del video y observemos la siguiente imagen:

Enero	Feb.	Marzo	Abril	Mayo	Junio	Julio	Agosto	Sep	Oct	Nov	Dic
		I	III	IIII		I	II		I		

2. Supongamos que cada uno de los integrantes de la cursada mencionó su mes de cumpleaños. Luego en una lista o tabla, dividida con los nombres de los meses se apuntó lo que dijo cada uno. Al finalizar la anotación y el recuento “mayo”, resultó ser el mes de más cumpleaños. Observemos que no hay empate con otro mes.

4. Ahora veamos como lo podemos hacer con una consulta.
Suponemos que los datos del grupo están almacenados en una tabla con estas características.

ALUMNO

Legajo	Nombre	Apellido	Doc	FechaNac	OtrosDatos

5. Lo primero que debemos hacer es agrupar por mes y contar cuantas filas tiene el resultado de cada grupo. Para eso utilizamos el group by y el count( )

Pero ¿Cómo obtenemos el mes de la fecha de nacimiento?

La fecha de nacimiento es un tipo de dato date y existen funciones que operan sobre este tipo de datos.

<< En el archivo Fecha_Hora_Msql.sql se explica cómo aplicarlas>> Podés descargar el archivo en la sección Material complementario haciendo clic aquí.

6. Para obtener el mes de la fecha de nacimiento usamos la función month(……) , lo que colocamos entre los paréntesis es el argumento de la función, en nuestro caso es el atributo FechaNac.

```
select month(FechaNac), count(*)
    from alumno
      group by month(FechaNac)
```

month(FechaNac)	Count(*)
03	1
04	3
05	4
07	1
08	2
10	1

7. Ahora nos tenemos que centrar en la segunda columna y buscar el valor máximo.

En el capítulo siguiente veremos como realizar esta última tarea.

Valor máximo

Lo primero que hacemos es renombrar la columna del contador para usarla en la función max( ) y renombrar la consulta como una tabla.

month(FechaNac)	contador
03	1
04	3
05	**4**
07	1
08	2
10	1

4

```
select max(contador)
 from (select month(FechaNac), count(*) as contador
            from alumno
             group by month(FechaNac))TABLA;
```

El resultado es nuestro valor máximo.

Aún no sabemos cuál es el mes, entonces proyectamos el mes de las filas agrupadas por mes y le colocamos la condición al grupo, por eso usamos el having, esta condición se refiere al contador. Y como vemos en la gráfica ese contador debe coincidir con el máximo que es 4.

```
select month(FechaNac) as mes
    from alumno
        group by month(FechaNac)
         having count(*) = "AL VALOR MÁXIMO"
```

Que es 4 según el resultado

¿Podemos colocar directamente ese valor? La respuesta es NO. Se debe reemplazar ese valor por la consulta que permitió calcularlo.

Entonces queda de esta forma:

```
select month(FechaNac) as mes
    from alumno
        group by month(FechaNac)
         having count(*) = (select max(contador)
                            from (select month(FechaNac), count(*) as contador
                                  from alumno
                                   group by month(FechaNac))TABLA);
```

mes
05

Resultado

¡Ahora si estamos frente al resultado del mes!

Observamos que el resultado tiene una sola fila porque no encontramos empate con otro mes. Si hay empate la consulta es exactamente igual y tendrá la cantidad de filas que cumplan con la condición de filtro.

Sugerencia : Para darte cuenta cómo funciona te propongo que uses la Base “Taller” de la semana 7 y busques en que mes se realizaron más presupuestos.

¿Te diste cuenta que comenzamos a razonar la consulta desde adentro hacia afuera?

La forma de construir la consulta no es única, si no tienes experiencia es más fácil comenzar por las subconsultas para ir a la consulta principal.

Uso de combinaciones

Este nuevo caso nos invita a pensar y a aplicar lógica de resolución de problemas para determinar cómo armar las consultas y cuantas utilizar.

El caso dice “Mostrar por presupuesto el costo de mano de obra de cada reparación”.

Según el modelo de la base de datos “Taller” (semana 7) cada presupuesto se relaciona con los repuestos a través de la tabla “presurep” y tiene un atributo llamado monto que representa lo que el cliente debe pagar en concepto del arreglo de su vehículo.

Ese monto está formado por el costo mano de obra y el costo de repuestos.

¿Por qué decimos que nos invita a pensar?

Porque puede ocurrir que un presupuesto requiera de un arreglo que no utilice repuestos como, el reseteo de la computadora interna, entonces, el monto es netamente mano de obra.

¡Comencemos!

En los próximos capítulos veremos un paso a paso de esta operación.

Paso 1

Cálculo de costo de repuestos por presupuesto

Trabajamos con presurep para saber que repuestos y que cantidad corresponden a cada presupuesto y con repuesto para saber el precio de cada uno. Como es por presupuesto se debe agrupar por el número de presupuesto.

```
select npresup, sum(precio * cant) as Costo_Rep
    from presurep pr inner join repuesto r on pr.codrep = r.codrep
        group by npresup
```

Al usar el **group by** la función **sum** → suma todas las filas del grupo, pero **ATENCIÓN** antes se realiza la **multiplicación** del precio del repuesto por la cantidad

El atributo precio es de la tabla **repuesto** y el atributo cant de la tabla **presurep**

El resultado de esta consulta son dos columnas, una con el número de presupuesto y la otra con el costo de repuesto.

Paso 2

Cálculo del monto de cada presupuesto

Trabajamos con la tabla presup. Esta es la entidad que tiene toda la información de cada presupuesto, no utilizo grupo porque el número de presupuesto no se repite por ser clave primaria.

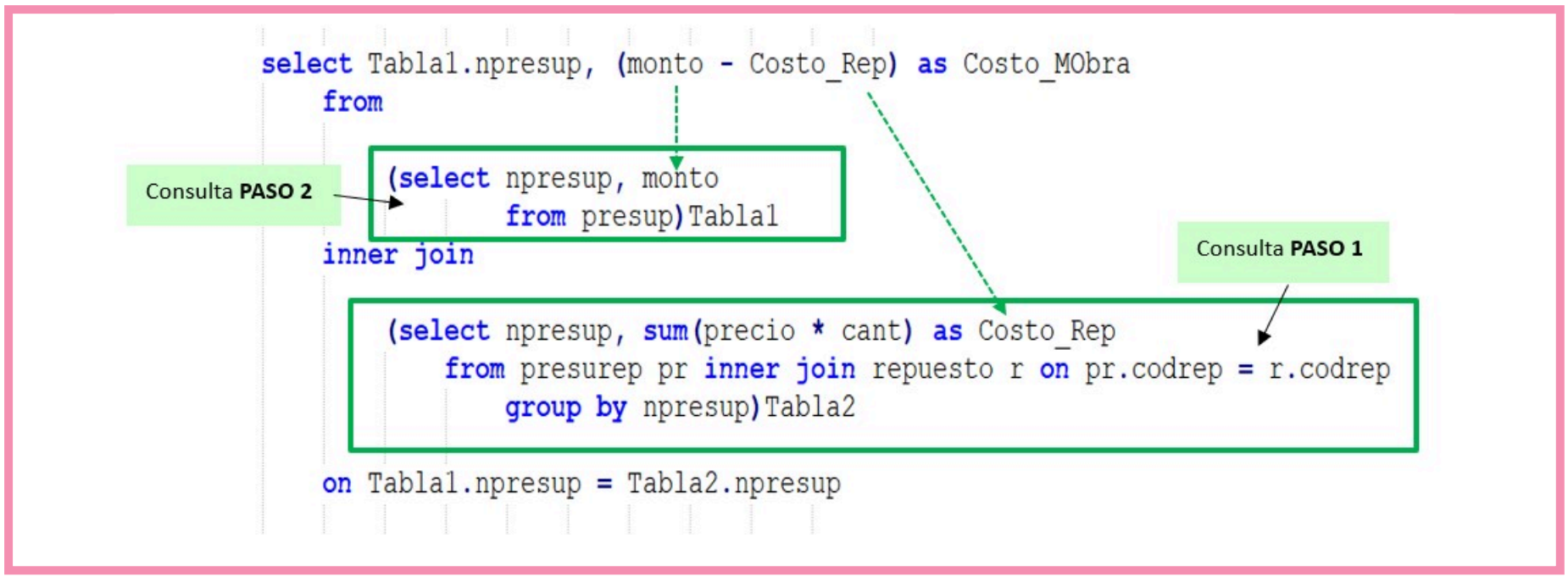
```
select npresup, monto
    from presup
```

Paso 3

Cálculo del costo de mano de obra por presupuesto

Trabajamos con las consultas del Paso 1 y del Paso 2, sabiendo que se debe restar al monto el costo de repuesto.

Consideramos que la resolución contempla solamente a los presupuestos que usan repuestos, por eso usamos inner join.



Paso 4

Cálculo del costo de mano de obra por presupuesto con y sin repuestos

En este último paso vamos a considerar el caso planteado al comienzo, la existencia de presupuestos que no utilicen repuestos.

En la resolución del Paso 3 la tabla que toma el lugar izquierdo es la Tabla1.

Tabla1 representa al select que proyecta a todos los presupuestos existentes, podemos pensar entonces que con un left outer join está solucionado, pero lamento decirte que no es posible.

¿Por qué?

Porque la proyección final de la consulta del Paso 3 tiene una columna con una operación matemática, la columna monto siempre tiene valor, pero la columna Costo_Rep puede ser null cuando la fila de Tabla 1 no tiene relación con Tabla 2.

SQL muestra entonces en ese lugar un null y nosotros necesitamos el valor del monto.

Veamos entonces como hay que transformar la consulta para obtener al costo de mano de obra de todos los presupuestos con o sin repuestos.

Pero antes hay que conocer al “IF”.

Este será el tema que trataremos en el próximo capítulo.

Uso del IF

¿Qué es?

El if es una función muy usada en programación, nos permite tomar dos caminos en relación a una condición. Por ejemplo, si decimos que todos los repuestos cuyo precio es mayor a 10000 es caro y en caso contrario es barato, podemos preguntar

¿precio >10000? SI -------- *"es CARO"*

¿precio >10000? NO -------- *"es BARATO"*

Veamos como lo representamos en una consulta SQL.

```
select nombre,precio, if(precio >10000,"es CARO", "es BARATO")as mensaje
    from repuesto;
```

NO cumple la condición

SI cumple la condición

Y por pantalla vemos:

```
MariaDB [taller]> select nombre,precio, if(precio >10000,"es CARO", "es BARATO")as mensaje
    -> from repuesto;
+---------------------------------+--------+-----------+
| nombre                          | precio | mensaje   |
+---------------------------------+--------+-----------+
| Motor                           |  65000 | es CARO   |
| Rueda                           |   1000 | es BARATO |
| Ventana                         |   1000 | es BARATO |
| Retrovisor                      |    500 | es BARATO |
| Cigüeñal                        |   3000 | es BARATO |
| Grasa de caja de cambios        |    900 | es BARATO |
| Limpiaparabrisas                |    359 | es BARATO |
| Barra caja de cambios           |   1200 | es BARATO |
| Volante                         |  14300 | es CARO   |
| Tapa de cilindro gol trend      |   9800 | es BARATO |
```

Entonces la función if tiene 3 argumentos, el primero muestra la condición precio > 10000, el segundo el valor que asignamos cuando es verdadero en nuestro ejemplo "es caro" y el tercero el valor que asignamos cuando es falso en nuestro caso "es barato".

Transformación de la consulta

Usamos el IF

Ahora si estamos listos para transformar nuestra consulta.

La proyección de la consulta del Paso 3 muestra:

```
select Tabla1.npresup, (monto - Costo_Rep) as Costo_MObra
```

Al usar left outer join la columna "Costo_Rep" puede ser null, entonces colocamos la función if.

La condición debe ser Costo_Rep is null, para indica que si se cumple se le asigna el valor cero, y si no se cumple debe mantener el valor de la columna Costo_Rep.

Veamos cómo queda:

```
select Tabla1.npresup, monto - if(Costo_Rep is null,0,Costo_Rep) as Costo_MObra
```

Finalmente, la consulta queda así:

```
select Tabla1.npresup, monto - if(Costo_Rep is null,0,Costo_Rep) as Costo_MObra
from
    (select npresup, monto
        from presup)Tabla1
LEFT OUTER join
    (select npresup, sum(precio * cant) as Costo_Rep
        from presurep pr inner join repuesto r on pr.codrep = r.codrep
            group by npresup)Tabla2
on Tabla1.npresup = Tabla2.npresup;
```

si

no

Verificación del resultado

Comprobamos si el razonamiento es correcto ¿Cómo lo hacemos?

1. Para realizar esta verificación usamos los datos almacenados en la base hasta este momento.

2. Luego ejecutamos las consultas y verificamos:

Tomamos de ejemplo las siguientes imágenes

```
MariaDB [taller]> select npresup, monto
    -> from presup;
+---------+--------+
| npresup | monto  |
+---------+--------+
|   70100 | 150000 |
|   70101 |   8750 |
|   70102 |  14000 |
|   70103 |  18500 |
|   70104 |   9700 |
+---------+--------+
5 rows in set (0.00 sec)
```

Sin repuesto

```
MariaDB [taller]> select npresup, codrep, cant
    -> from presurep;
+---------+--------+------+
| npresup | codrep | cant |
+---------+--------+------+
|   70100 |     60 |    1 |
|   70101 |     66 |    2 |
|   70101 |     80 |    1 |
|   70102 |    115 |    1 |
|   70102 |    118 |    1 |
|   70104 |     90 |    1 |
+---------+--------+------+
6 rows in set (0.00 sec)
```

```
MariaDB [taller]> select npresup, sum(precio * cant) as Costo_Rep
    -> from presurep pr inner join repuesto r on pr.codrep = r.codrep
    -> group by npresup;
+---------+-----------+
| npresup | Costo_Rep |
+---------+-----------+
|   70100 |     65000 |
|   70101 |      3000 |
|   70102 |     10700 |
|   70104 |      5600 |
+---------+-----------+
4 rows in set (0.00 sec)
```

```
MariaDB [taller]> select npresup, monto
    -> from presup;
+---------+--------+
| npresup | monto  |
+---------+--------+
|   70100 | 150000 |
|   70101 |   8750 |
|   70102 |  14000 |
|   70103 |  18500 |
|   70104 |   9700 |
+---------+--------+
5 rows in set (0.00 sec)
```

NPresup → MONTO – COSTP_REP

70100→ 150000 – 65000 = 85000
70101→ 8750 – 3000 = 5750
70102 → 14000 – 10700 = 3300
70103 → 18500 – 0 (sin repuesto) = 18500
70104 → 9700 – 5600 = 4100

```
MariaDB [taller]> select Tabla1.npresup, monto - if(Costo_Rep is null,0,Costo_Rep) as Costo_MObra
    ->
    -> from
    -> (select npresup, monto
    -> from presup)Tabla1
    ->
    -> LEFT OUTER join
    ->
    -> (select npresup, sum(precio * cant) as Costo_Rep
    -> from presurep pr inner join repuesto r on pr.codrep = r.codrep
    -> group by npresup)Tabla2
    ->
    -> on Tabla1.npresup = Tabla2.npresup;
+---------+-------------+
| npresup | Costo_MObra |
+---------+-------------+
|   70100 |       85000 |
|   70101 |        5750 |
|   70102 |        3300 |
|   70103 |       18500 |
|   70104 |        4100 |
+---------+-------------+
5 rows in set (0.00 sec)
```

¡Llegamos al resultado correcto!