

1. DML. Las subconsultas

1.1 Introducción

Una subconsulta es una consulta que aparece dentro de otra consulta o subconsulta en la lista de selección, en la cláusula WHERE o HAVING, originalmente no se podían incluir en la lista de selección.

Una subconsulta se denomina también consulta o selección interna, mientras que la instrucción que contiene la subconsulta es conocida como consulta o selección externa.

Aparece siempre encerrada entre paréntesis y tiene la misma sintaxis que una sentencia SELECT normal con alguna limitación:

No puede incluir una cláusula COMPUTE o FOR BROWSE y sólo puede incluir una cláusula ORDER BY cuando se especifica también una cláusula TOP.

Una subconsulta puede anidarse en la cláusula WHERE o HAVING de una instrucción externa SELECT, INSERT, UPDATE o DELETE, o bien en otra subconsulta. Se puede disponer de hasta 32 niveles de anidamiento, aunque el límite varía dependiendo de la memoria disponible y de la complejidad del resto de las expresiones de la consulta. Hay que tener en cuenta que para cada fila de la consulta externa, se calcula la subconsulta, si anidamos varias consultas, el número de veces que se ejecutarán las subconsultas ¡puede dispararse!

Cuando la subconsulta aparece en la lista de selección de otra consulta, deberá devolver un solo valor, de lo contrario provocará un error.

Ejemplo de subconsulta: Listar los empleados cuya cuota no supere el importe vendido por el empleado.

```
SELECT nombre
FROM empleados
WHERE cuota <= (SELECT SUM(importe)
                FROM pedidos
                WHERE rep = numemp);
```

Muchas de las instrucciones Transact-SQL que incluyen subconsultas se pueden formular también utilizando composiciones de tablas. Otras preguntas se pueden formular sólo con subconsultas.

En Transact-SQL, normalmente no hay diferencias de rendimiento entre una instrucción que incluya una subconsulta y una versión semánticamente equivalente que no la incluya.

Sin embargo, en algunos casos en los que se debe comprobar la existencia de un elemento (por ejemplo con la cláusula DISTINCT), una composición produce mejores resultados, ya que se debe procesar la consulta anidada para cada resultado de la consulta externa con el fin de garantizar la eliminación de los duplicados. En tales casos, la utilización de composiciones producirá mejores resultados.

Si una tabla aparece en la subconsulta y no en la consulta externa, las columnas de esa tabla no se podrán incluir en la salida (la lista de selección de la consulta externa).

Hay tres tipos básicos de subconsultas:

- Las que aparecen en la lista de selección de la consulta externa o con un operador de comparación sin modificar y deben devolver un solo valor.
- Las que generan una lista de valores y operan con operadores IN y operadores de comparación modificados como ANY, SOME o ALL.
- Las que son utilizadas en pruebas de existencia especificadas con EXISTS, y las que da igual el número de columnas de la lista de selección.

A lo largo del tema las estudiaremos todas.

Antes de terminar con la introducción queda comentar el concepto de referencia externa muy útil en las subconsultas.

A menudo, es necesario, dentro del cuerpo de una subconsulta, hacer referencia al valor de una columna en la fila actual de la consulta externa, el nombre de la columna de la consulta externa dentro de la subconsulta recibe el nombre de referencia externa, ya que hace referencia a una columna externa.

En el ejemplo anterior *numemp* es una referencia externa, no es una columna del origen de datos de la subconsulta (*pedidos*), es una columna del origen de la consulta externa (*empleados*).

Hay que tener en cuenta de cómo se ejecuta la consulta; por cada fila de la consulta externa se calcula el resultado de la subconsulta y se evalúa la comparación.

En el ejemplo, se coge el primer empleado (*numemp*= 101, por ejemplo) y se calcula la subconsulta sustituyendo *numemp* por el valor 101, se calcula la suma de los pedidos del rep = 101, y el resultado se compara con la cuota de ese empleado, y así se repite el proceso con todas las filas de empleados.

El nombre de una columna dentro de la subconsulta se presupone del origen de datos de la subconsulta y, sólo si no se encuentra en ese origen, la considera como columna externa y la busca en el origen de la consulta externa.

Por ejemplo:

```
SELECT oficina, ciudad
FROM oficinas
WHERE objetivo > (SELECT SUM(ventas)
                  FROM empleados
                  WHERE oficina = oficina);
```

Hay un campo *oficina* en los dos orígenes (*oficinas* y *empleados*) pero esta consulta no dará error, dentro de la subconsulta se considera *oficina* el campo de la tabla *empleados* no como una referencia externa.

Si quisiéramos indicar que la oficina del empleado sea igual a la oficina de oficinas habría que escribirlo así:

```
SELECT oficina, ciudad
FROM oficinas
WHERE objetivo > (SELECT SUM(ventas)
                  FROM empleados
                  WHERE oficina = oficinas.oficina);
```

1.2 Subconsultas de resultado único

Existen subconsultas que deben obligatoriamente devolver un único valor, son las que aparecen en la lista de selección de la consulta externa o las que aparecen en WHERE o HAVING combinadas con un operador de comparación sin modificar.

Los operadores de comparación sin modificar son los operadores de comparación que vimos con la cláusula WHERE.

Sintaxis:

```
<expresion> {=|<|!|=|>|=|!>|<|<=|!<} <expresion>
```

En este caso la segunda expresión será una subconsulta, con una sola columna en la lista de selección y deberá devolver una única fila como mucho.

Ese valor único será el que se compare con el resultado de la primera expresión.

Si la subconsulta no devuelve ninguna fila, la comparación opera como si la segunda expresión fuese nula.

Si la subconsulta devuelve más de una fila, da error.

1.3 Subconsultas de lista de valores

Otro tipo de subconsultas son las que devuelven una lista de valores en forma de una columna y cero, una o varias filas.

Estas consultas aparecen en las cláusulas WHERE o HAVING combinadas con los operadores (IN) o las comparaciones modificadas.

1.3.1 El operador IN con subconsulta

```
<expresion> IN subconsulta
```

IN examina si el valor de la expresion es uno de los valores incluidos en la lista de valores generados por la subconsulta.

La subconsulta tiene que generar valores de un tipo compatible con la expresión.

Ejemplo:

```
SELECT *
FROM empleados
WHERE oficina IN (SELECT oficina
                  FROM oficinas
                  WHERE region = 'Este');
```

Obtiene los datos de los empleados cuyo número de oficina sea un número de oficina de las oficinas del Este. Es decir los empleados de oficinas del Este.

101	Antonio Viguer	45	12	representante	1986-10-20	104	30000,00	30500,00
103	Juan Rovira	29	12	representante	1987-03-01	104	27500,00	28600,00
104	José González	33	12	dir ventas	1987-05-19	106	20000,00	14300,00
105	Vicente Pantalla	37	13	representante	1988-02-12	104	35000,00	36800,00
106	Luis Antonio	52	11	dir general	1988-06-14	NULL	27500,00	29900,00

Si la subconsulta no devuelve ninguna fila:

```
SELECT *
FROM empleados
WHERE oficina IN (SELECT oficina
                  FROM oficinas
                  WHERE region = 'Otro');
```

La condición no se cumple y en este caso no devuelve ningún empleado.

Muchas veces la misma pregunta se puede resolver mediante una composición de tablas.

```
SELECT empleados.*
FROM Empleados INNER JOIN oficinas ON empleados.oficina = oficinas.oficina
WHERE region = 'Este';
```

Esta sentencia es equivalente.

Si combinamos el operador IN con NOT obtenemos el operador NOT IN.

```
<expresion> NOT IN subconsulta
```

Devuelve TRUE si el valor de la expresión no está en la lista de valores devueltos por la subconsulta.

```
SELECT *
FROM empleados
WHERE oficina NOT IN (SELECT oficina
                      FROM oficinas
                      WHERE region = 'Este');
```

Devuelve los empleados cuya oficina no esté en la lista generada por la subconsulta, es decir empleados que trabajan en oficinas que no son del Este.

Si la subconsulta no devuelve ninguna fila, la condición se cumplirá para todos los empleados que tengan una oficina.

A diferencia de IN, NOT IN no siempre puede resolverse con una composición:

```
SELECT numemp AS [IN]
FROM empleados
WHERE numemp IN (SELECT rep
                FROM pedidos
                WHERE fab = 'ACI');
```

Se puede resolver con una composición:

```
SELECT DISTINCT empleados.numemp AS [=]
FROM Empleados INNER JOIN pedidos ON numemp = rep
WHERE fab = 'ACI';
```

En este caso, como un empleado puede tener varios pedidos hay que añadir DISTINCT para eliminar las repeticiones de empleados (si un empleado tiene varios pedidos de ACI aparecería varias veces).

Sin embargo esta sentencia con NOT IN, queremos los empleados que no tienen pedidos de ACI:

```
SELECT numemp AS [NOT IN]
FROM empleados
WHERE numemp NOT IN (SELECT rep
                    FROM pedidos
                    WHERE fab = 'ACI');
```

No se puede resolver con una composición:

```
SELECT DISTINCT empleados.numemp AS [<>]
FROM Empleados INNER JOIN pedidos ON numemp = rep
WHERE fab <> 'ACI';
```

Esta consulta devuelve los empleados que tienen pedidos que no son de ACI, pero un empleado puede tener pedidos de ACI y otros de otros fabricantes y por estos saldría en el resultado cuando tiene pedidos de ACI y no debería salir.

Hay que tener mucho cuidado con este tipo de preguntas.

Para terminar, sólo queda comentar que en cualquiera de los casos, si la expresión delante del operador genera el valor nulo, la condición da como resultado NULL y por lo tanto no se cumple. En todos los ejemplos con oficinas, los empleados que no tienen oficina asignada (*oficina* Null) no salen.

1.3.2 La comparación modificada

Los operadores de comparación que presentan una subconsulta se pueden modificar mediante las palabras clave ALL, ANY o SOME. SOME es un equivalente del estándar de SQL-92 de ANY.

Se utiliza este tipo de comparación cuando queremos comparar el resultado de la expresión con una lista de valores y actuar en función del modificador empleado.

1.3.2.1 El test ANY

```
<expresion> {=<>|!=|>|=|!>|<|=|!<} {ANY|SOME} subconsulta
```

ANY significa que, para que una fila de la consulta externa satisfaga la condición especificada, la comparación se debe cumplir para al menos un valor de los devueltos por la subconsulta.

Por cada fila de la consulta externa se evalúa la comparación con cada uno de los valores devueltos por la subconsulta y si la comparación es True para alguno de los valores ANY es verdadero, si la comparación no se cumple con ninguno de los valores de la consulta, ANY da False a no ser que todos los valores devueltos por la subconsulta sean nulos en tal caso ANY dará NULL.

Si la subconsulta no devuelve filas ANY da False incluso si expresion es nula.

Ejemplo:

```
SELECT *
FROM empleados
WHERE cuota > ANY (SELECT cuota
                   FROM empleados empleados2
                   where empleados.oficina = empleados2.oficina);
```

Obtenemos los empleados que tengan una cuota superior a la cuota de alguno de sus compañeros de oficina, es decir los empleados que no tengan la menor cuota de su oficina.

En este caso hemos tenido un alias de tabla en la subconsulta (*empleados2*) para poder utilizar una referencia externa.

1.3.2.2 El test ALL

```
<expresion> {=<>|!=|>|=|!>|<|=|!<} ALL subconsulta
```

Con el modificador ALL, para que se cumpla la condición, la comparación se debe cumplir con cada uno de los valores devueltos por la subconsulta.

Si la subconsulta no devuelve ninguna fila ALL da True.

```
WHERE cuota > ALL (SELECT cuota
                  FROM empleados empleados2
                  WHERE empleados.oficina = empleados2.oficina);
```

En el ejemplo anterior obtenemos los empleados que tengan una cuota superior a todas las cuotas de la oficina del empleado. Podríamos pensar que obtenemos el empleado de mayor cuota de su oficina pero no lo es, aquí tenemos un problema, la cuota del empleado aparece en el resultado de subconsulta por lo tanto > no se cumplirá para todos los valores y sólo saldrán los empleados que no tengan oficina (para los que la subconsulta no devuelve filas).

Para salvar el problema tendríamos que quitar del resultado de la subconsulta la cuota del empleado modificando el WHERE:

```
WHERE empleados.oficina = empleados2.oficina
      AND empleados.numemp <> empleados2.numemp);
```

De esta forma saldrían los empleados que tienen una cuota mayor que cualquier otro empleado de su misma oficina.

O bien

```
WHERE empleados.oficina = empleados2.oficina
      AND empleados.cuota <> empleados2.cuota);
```

Para no considerar los empleados que tengan la misma cuota que el empleado. En este caso saldrían los empleados con la mayor cuota de sus oficina, pero si dos empleados tienen la misma cuota superior, saldrían, hecho que no sucedería con la otra versión.

Cuando la comparación es una igualdad, = ANY es equivalente a IN y <> ALL es equivalente a NOT IN.

1.4 Subconsultas en pruebas de existencia EXISTS

Existe otro operador de subconsulta con el que la subconsulta puede devolver más de una columna, el operador EXISTS.

En este caso la sintaxis es algo diferente:

```
WHERE [NOT] EXISTS subconsulta
```

No se realiza ninguna comparación con los valores devueltos por la subconsulta, simplemente se evalúa si la subconsulta devuelve alguna fila, en este caso EXISTS será True y si la subconsulta no devuelve ninguna fila, EXISTS será False.

Ejemplo:

```
SELECT *
FROM empleados
WHERE EXISTS (SELECT *
              FROM pedidos
              WHERE numemp = rep and fab = 'ACI');
```

Obtenemos los empleados que tengan un pedido del fabricante ACI. Por cada empleado, se calcula la subconsulta (los pedidos de ese empleado y cuyo fabricante sea ACI), si existe alguna fila, el empleado sale en el resultado, sino, no sale.

Cuando se utiliza el operador EXISTS es muy importante añadir una referencia externa, no es obligatorio pero en la mayoría de los casos será necesario. Veámoslo con ese mismo ejemplo, si quitamos la referencia externa:

```
SELECT *
FROM empleados
WHERE EXISTS (SELECT *
              FROM pedidos
              WHERE fab = 'ACI');
```

Sea el empleado que sea, la subconsulta siempre devolverá filas (si existe algún pedido cuyo fabricante sea ACI) o nunca, indistintamente del empleado que sea, por lo que se obtendrán todos los empleados o ninguno.

Otra cosa a tener en cuenta es que la lista de selección de una subconsulta que se especifica con EXISTS casi siempre consta de un asterisco (*). No hay razón para enumerar los nombres de las columnas porque no se van a utilizar y supone un trabajo extra para el sistema.

Si utilizamos NOT EXISTS el resultado será el contrario.

```
SELECT *
FROM empleados
WHERE NOT EXISTS (SELECT *
                  FROM pedidos
                  WHERE fab = 'ACI');
```

Devuelve los empleados que no tienen ningún pedido de ACI.

1.5 Orden de ejecución de la SELECT.

Para terminar, ahora que conocemos las cláusulas de la sentencia SELECT, veamos cómo, en qué orden se ejecuta internamente una SELECT con sus distintas cláusulas:

Paso 1. Si la sentencia es una UNION de sentencias SELECT, aplicar los pasos 2 hasta 7 a cada una de las sentencias para generar sus resultados de consulta individuales.

Paso 2. Se calcula el origen de datos definido en la cláusula FROM.

Paso 3. Si hay una cláusula WHERE, se aplica su condición de búsqueda a cada fila del origen, reteniendo aquellas filas para las cuales la condición tiene valor TRUE. Si la cláusula WHERE contiene una subconsulta, la subconsulta se calcula para cada fila conforme es examinada.

Paso 4. Si hay una cláusula GROUP BY, se disponen las filas obtenidas en paso 3 en grupos de filas, de modo que las filas de cada grupo tengan valores idénticos en todas las columnas de agrupación.

Paso 5. Si hay una cláusula HAVING, se aplica la condición de búsqueda a cada grupo de filas, reteniendo aquellos grupos para los cuales la condición es TRUE. Si la cláusula HAVING contiene una subconsulta, la subconsulta se efectúa para cada fila conforme es examinada.

Paso 6. Para cada fila (o grupo de filas) restante, se calcula el valor de cada elemento en la lista de selección para obtener una única fila de resultados. Para una referencia de columna simple, se utiliza el valor de la columna en la fila (o grupo de filas) actual. Para una función de columna, se utiliza como argumento el grupo de filas actuales si se especificó GROUP BY; en caso contrario, se utiliza el conjunto entero de filas obtenido en el paso 3.

Paso 7. Si se especifica SELECT DISTINCT, se elimina del resultado las filas duplicadas que se hubieran producido.

Paso 8. Si la sentencia es una UNION, se mezclan los resultados de las consultas en una única tabla de resultados, eliminando las filas duplicadas a menos que se haya especificado UNION ALL.

Paso 9. Si hay una cláusula ORDER BY, se ordenan las filas de la consulta según se haya especificado.

Las filas generadas por este procedimiento forman el resultado de la consulta.