Martínez Rojas José Eduardo

Subconsultas

DEFINICIONES

Una ***subconsulta*** en SQL consiste en utilizar los resultados de una consulta dentro de otra, que se considera la principal. Esta posibilidad fue la razón original para la palabra "estructurada" en el nombre Lenguaje de Consultas Estructuradas (Structured Query Language, SQL).

Una subconsulta es una sentencia SELECT que aparece dentro de otra sentencia SELECT que llamaremos consulta principal.
Se puede encontrar en la lista de selección, en la cláusula WHERE o en la cláusula HAVING de la consulta principal

Es una sentencia SELECT que aparece en la definición de otra sentencia SELECT, a la que denominamos consulta principal.

REGLAS

Las subconsultas están sujetas a las restricciones siguientes:

- La lista de selección de una subconsulta que se especifica con un operador de comparación, solo puede incluir un nombre de expresión o columna (excepto EXISTS e IN, que operan en SELECT * o en una lista respectivamente).
- Si la cláusula WHERE de una consulta externa incluye un nombre de columna, debe ser compatible con una combinación con la columna indicada en la lista de selección de la subconsulta.
- Los tipos de datos **ntext**, **text** y **image** no están permitidos en las listas de selección de subconsultas.
- Puesto que deben devolver un solo valor, las subconsultas que se especifican con un operador de comparación sin modificar (no seguido de la palabra clave ANY o ALL) no pueden incluir las cláusulas GROUP BY y HAVING.
- La palabra clave DISTINCT no se puede usar con subconsultas que incluyan GROUP BY.
- Las cláusulas COMPUTE y INTO no se pueden especificar.
- Solo se puede especificar ORDER BY si se especifica también TOP.
- Una vista creada con una subconsulta no se puede actualizar.
- La lista de selección de una subconsulta especificada con EXISTS, por convención, tiene un asterisco (*) en lugar de un solo nombre de columna. Las reglas de una subconsulta especificada con EXISTS son idénticas a las de una lista de selección estándar, porque una subconsulta introducida por EXISTS crea una prueba de existencia y devuelve TRUE o FALSE en lugar de datos.

Formato de una subconsulta

```
??(SELECT???ALL  ???????*?????????????????????>
           ?DISTINCT?  ?columna_resultado?

  >??FROM lista_de_tablas ?????????????????????>

  >?????????????????????????????????????????????>
     ?WHERE condición_de_selección?

  >?????????????????????????????????????????????>
     ?GROUP BY lista_de_columnas_para_agrupar?

  >?????????????????????????????????????????????>)
     ?HAVING condición_de_selección?
```

donde el formato de la sentencia `SELECT` entre paréntesis tiene las siguientes diferencias con la sentencia `SELECT` de las consultas:

- - No tiene sentido la cláusula `ORDER BY` ya que los resultados de una subconsulta se utilizan internamente y no son visibles al usuario.
- - Los nombres de columna que aparecen en una subconsulta pueden referirse a columnas de la tabla de la consulta principal y se conocen como *referencias externas*.

SELECT

En la cláusula SELECT:
Generalmente es para obtener la información agregada de algún dato dentro de una tabla
Restricción: La subconsulta no puede regresar más de un valor

FROM

En la cláusula FROM:
Permite obtener una serie de filas y columnas, basadas en el resultado de la subconsulta, que podemos tratar como una tabla.

JOIN
En un join:
Permite obtener una serie de filas y columnas, basadas en el resultado de la subconsulta, que podemos tratar como una tabla para emplearla como operando según sea el caso.

WHERE/HAVING
En la cláusula WHERE/HAVING:
La subconsulta fungirá como operando para la condición. Puede aplicarse de dos formas:
Directamente
Dentro de algún operador lógico (IN, ANY, ALL, EXISTS)

CORRELACIONADA
Correlacionada:
La consulta principal y subconsultas extraen datos de la misma tabla.

Valores de retorno de las subconsultas y condiciones de selección.

Una subconsulta siempre forma parte de la condición de selección en las cláusulas `WHERE` o `HAVING`.

El resultado de una subconsulta puede ser un valor simple o más de un valor. Según el retorno de la subconsulta, el operador de comparación que se utilice en la condición de selección del `WHERE` o `HAVING` deberá ser del tipo apropiado según la tabla siguiente:

Operador comparativo	***Retorno de la subconsulta***
De tipo aritmético	Valor simple
De tipo lógico	Más de un valor

Condición de selección con operadores aritméticos de comparación.

Se utiliza cuando la subconsulta devuelve un único valor a comparar con una expresión, por lo general formada a partir de la fila obtenida en la consulta principal. Si la comparación resulta cierta (`TRUE`), la condición de selección también lo es. Si la subconsulta no devuelve ninguna fila (`NULL`), la comparación devuelve también el valor `NULL`.

Formato para la condición de selección con operadores aritméticos de comparación

```
?? expresión  operador_aritmético de comparación  subconsulta ??>

        Operadores_aritméticos de comparación: =,<>,<,>,<=,<=
```

Ejemplos.

1. Obtener todos los empleados que tienen el mismo oficio que 'Alonso'.

```
SQL> SELECT emp_no "Nº Empleado",apellido,oficio
       FROM empleados
       WHERE oficio=(SELECT oficio FROM empleados
                     WHERE UPPER(apellido)='ALONSO');
```

```
  Nº Empleado APELLIDO OFICIO
----------- -------- ----------
       7499 ALONSO   VENDEDOR
       7654 MARTIN   VENDEDOR
       7844 CALVO    VENDEDOR
```

```
En Access:
      SELECT emp_no AS NºEmpleado, apellido, oficio FROM empleados
            WHERE oficio=(SELECT oficio FROM empleados
                          WHERE UCASE(apellido)= 'ALONSO');
```

Aquí no es una es una subconsulta relacionada, donde la consulta principal es select No empleado,apellido, oficio, from empleados where oficio y la subconsulta es Select oficio from empleados where upper(apellido='ALONSO')
Upper sirve Para tranformar a mayusculas

1. Listar, en orden alfabético, aquellos empleados que no trabajen ni en Madrid ni en Barcelona.

```
SQL> SELECT emp_no "Nº Empleado",apellido,dep_no
       "NºDepartamento"
       FROM empleados
       WHERE dep_no IN (SELECT dep_no
                        FROM departamentos
                        WHERE UPPER(localidad) NOT IN
                        ('MADRID','BARCELONA'))
       ORDER BY apellido;
```

```
Nº Empleado APELLIDO NºDepartamento
----------- -------- --------------
       7876 GIL                  20
       7900 JIMENEZ              20
```

SUBCONSULTAS

Pueden utilizarse de tres formas:

- Dentro de la lista de campos de la instrucción SELECT

```
SELECT ListaCampos, (Select Campo from TablaSubconsulta where…)
  FROM TablaPrincipal
```

- En la cláusula FROM

```
SELECT ListadeCampos, OtroCampo, UltimoCampo
  FROM (Select Campo from TablaSubconsulta where…) T
```

- En la cláusula WHERE

```
SELECT ListadeCampos, OtroCampo, UltimoCampo
  FROM Tabla
 WHERE Campo =( Select… )
```

SUBCONSULTAS

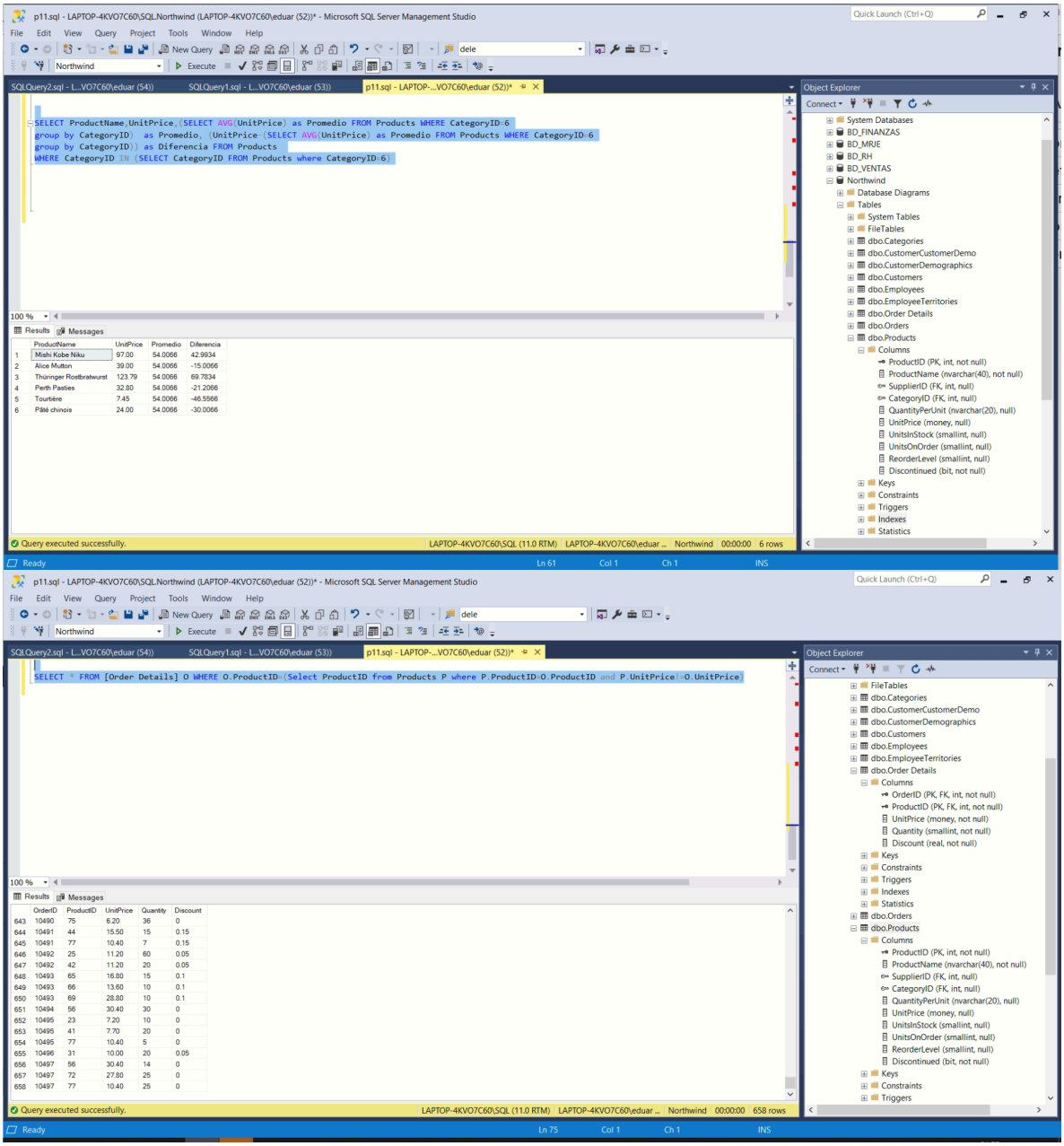
Ejem. Obtener el nombre del contacto de los Clientes que compraron en Abril de 1998 y la cuidad a la que pertenecen

```
SELECT ContactName, City
FROM Customers
WHERE CustomerID in
    (Select Distinct CustomerID
       From Orders
      Where Datename(mm,OrderDate) = 'April'
            and year(OrderDate) = 1998)
```

Results | Messages

	ContactName	City
44	Liz Nixon	Portland
45	Liu Wong	Butte
46	Palle Ibsen	Árhus
47	Rita Müller	Stuttgart
48	Pirkko Koskitalo	Oulu
49	Karl Jablonski	Seattle
50	Matti Karttunen	Helsinki
51	Zbyszek Piestrz...	Warszawa

Query executed successfully.


p11.sql - LAPTOP-4KVO7C60\SQLNorthwind (LAPTOP-4KVO7C60\eduar (52))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
Execute
Northwind
SQLQuery2.sql - L...VO7C60\eduar (54))
SQLQuery1.sql - L...VO7C60\eduar (53))
p11.sql - LAPTOP-...VO7C60\eduar (52))*
SELECT ProductName,UnitPrice,(SELECT AVG(UnitPrice) as Promedio FROM Products WHERE CategoryID=6
group by CategoryID) as Promedio, (UnitPrice -(SELECT AVG(UnitPrice) as Promedio FROM Products WHERE CategoryID=6
group by CategoryID)) as Diferencia FROM Products
WHERE CategoryID IN (SELECT CategoryID FROM Products where CategoryID=6)
Results
Messages
ProductName UnitPrice Promedio Diferencia
1 Mishi Kobe Niku 97.00 54.0066 42.9934
2 Alice Mutton 39.00 54.0066 -15.0066
3 Thüringer Rostbratwurst 123.79 54.0066 69.7834
4 Perth Pasties 32.80 54.0066 -21.2066
5 Tourtière 7.45 54.0066 -46.5566
6 Pâté chinois 24.00 54.0066 -30.0066
Query executed successfully.
LAPTOP-4KVO7C60\SQL (11.0 RTM)
LAPTOP-4KVO7C60\eduar ...
Northwind
00:00:00
6 rows
Ready
Ln 61
Col 1
Ch 1
INS
Object Explorer
Connect
System Databases
BD_FINANZAS
BD_MRJE
BD_RH
BD_VENTAS
Northwind
Database Diagrams
Tables
System Tables
FileTables
dbo.Categories
dbo.CustomerCustomerDemo
dbo.CustomerDemographics
dbo.Customers
dbo.Employees
dbo.EmployeeTerritories
dbo.Order Details
dbo.Orders
dbo.Products
Columns
ProductID (PK, int, not null)
ProductName (nvarchar(40), not null)
SupplierID (FK, int, null)
CategoryID (FK, int, null)
QuantityPerUnit (nvarchar(20), null)
UnitPrice (money, null)
UnitsInStock (smallint, null)
UnitsOnOrder (smallint, null)
ReorderLevel (smallint, null)
Discontinued (bit, not null)
Keys
Constraints
Triggers
Indexes
Statistics
p11.sql - LAPTOP-4KVO7C60\SQLNorthwind (LAPTOP-4KVO7C60\eduar (52))* - Microsoft SQL Server Management Studio
SELECT * FROM [Order Details] O WHERE O.ProductID=(Select ProductID from Products P where P.ProductID=O.ProductID and P.UnitPrice=O.UnitPrice)
OrderID ProductID UnitPrice Quantity Discount
643 10490 75 6.20 36 0
644 10491 44 15.50 15 0.15
645 10491 77 10.40 7 0.15
646 10492 25 11.20 60 0.05
647 10492 42 11.20 20 0.05
648 10493 65 16.80 15 0.1
649 10493 66 13.60 10 0.1
650 10493 69 28.80 10 0.1
651 10494 56 30.40 30 0
652 10495 23 7.20 10 0
653 10495 41 7.70 20 0
654 10495 77 10.40 5 0
655 10496 31 10.00 20 0.05
656 10497 56 30.40 14 0
657 10497 72 27.80 25 0
658 10497 77 10.40 25 0
Query executed successfully.
LAPTOP-4KVO7C60\SQL (11.0 RTM)
LAPTOP-4KVO7C60\eduar ...
Northwind
00:00:00
658 rows
Ready
Ln 75
Col 1
Ch 1
INS