#CrecimientoProfesional
#AprendeConLosPioneros

Online | DMC Perú

DMC Perú | CAPACITACIÓN PROFESIONAL

SESIÓN VII

Docente: Victor Gutierrez
Data Architect

Agenda

Manejo Avanzado sobre Tablas

- Pivoteo de tablas (pivot, unpivot).
- Manejo de subconsultas, tablas derivadas.
- Instrucciones tipo query jerárquica y correlaciónales.
- Uso de GROUP BY, GROUPING, SET, HAVING.
- Manejo de expresiones de tablas (Croos Apply, outer apply).

Pivoteo de tablas (pívot, unpivot).

Convertir filas en columnas ***PIVOT*** y columnas en filas ***UNPIVOT***

anio	Name	SubTotal
2011	Southeast	20565,6206
2011	Southeast	1294,2529
2011	Southwest	419,4589
2011	Northwest	24432,6088
2011	Northwest	14352,7713
2011	Southwest	5056,4896
2011	Central	6107,082
2011	Northwest	714,7043

PIVOT

Name	2011	2012	2013	2014
Northwest	1248473,5306	2937972,7031	3584751,866	1596395,5334
Northeast	875823,8318	3375456,8947	3985374,8995	1057247,3786
Central	1311627,2918	4317306,5741	3396776,2674	1040093,4071
Southwest	2117312,6152	6129119,2246	6498550,9672	2049030,1735
Southeast	1521289,1881	2674436,3518	2188082,7813	787204,4289

UNPIVOT

anio	Name	SubTotal
2011	Southeast	20565,6206
2011	Southeast	1294,2529
2011	Southwest	419,4589
2011	Northwest	24432,6088
2011	Northwest	14352,7713
2011	Southwest	5056,4896
2011	Central	6107,082
2011	Northwest	714,7043

Pivoteo de tablas (pívot, unpivot).

Convertir filas en columnas ***PIVOT*** y columnas en filas ***UNPIVOT***

```
SELECT <non-pivoted column>,
    [first pivoted column] AS <column name>,
    [second pivoted column] AS <column name>,
    ...
    [last pivoted column] AS <column name>
FROM
    (<SELECT query that produces the data>)
    AS <alias for the source query>
PIVOT
(
    <aggregation function>(<column being aggregated>)
FOR
[<column that contains the values
       that will become column headers>]
    IN ( [first pivoted column], [second pivoted column],
    ... [last pivoted column])
) AS <alias for the pivot table>
<optional ORDER BY clause>;
```

Manejo de subconsultas, tablas derivadas

Sub Consultas: no usa valores de la consulta externa
Tablas Derivadas: usa valores de la consulta externa.

Como parte de un where

```
select *
from Employees e
left join [dbo].[EmployeeTerritories] et on et.[EmployeeID] = e.[EmployeeID]
where et.TerritoryID in (select TerritoryID
                         from [dbo].[Territories] where RegionID = 3)
```

Como una columna

```
select f.*,
 (select sum(d.UnitPrice*Quantity)
  from [dbo].[Order Details] as d
  where f.[OrderID]=d.[OrderID]) as total
  from [dbo].[Orders] f
```

Manejo de subconsultas, tablas derivadas

Sub Consultas: no usa valores de la consulta externa
Tablas Derivadas: usa valores de la consulta externa.

Obtener los clientes que han hecho por lo menos un pedido

```
select c.CustomerID, c.CompanyName, c.ContactName
from Customers c
where c.CustomerID in (select CustomerID from Orders )
```

Otra forma de escribirse es utilizando la clausula ***EXISTS***

```
select c.CustomerID, c.CompanyName, c.ContactName
from Customers c
where exists (select CustomerID from Orders o
                where o.CustomerID = c.CustomerID)
```

Variables

DECLARE

- Asignar un nombre. El nombre debe tener un único @ como primer carácter.
- Asignar un tipo de datos suministrado por el sistema o definido por el usuario y una longitud.
- Para las variables numéricas, se asignan también una precisión y una escala.
- Establecer el valor a NULL.

```
DECLARE @LastName nvarchar(30), @FirstName nvarchar(20), @StateProvince nchar(2);
```

Asignación desde valores

```
-- Declare two variables.
DECLARE @FirstNameVariable nvarchar(50),
   @PostalCodeVariable nvarchar(15);

-- Set their values.
SET @FirstNameVariable = N'Amy';
SET @PostalCodeVariable = N'BA5 3HX';
```

Asignación desde una instrucción SQL

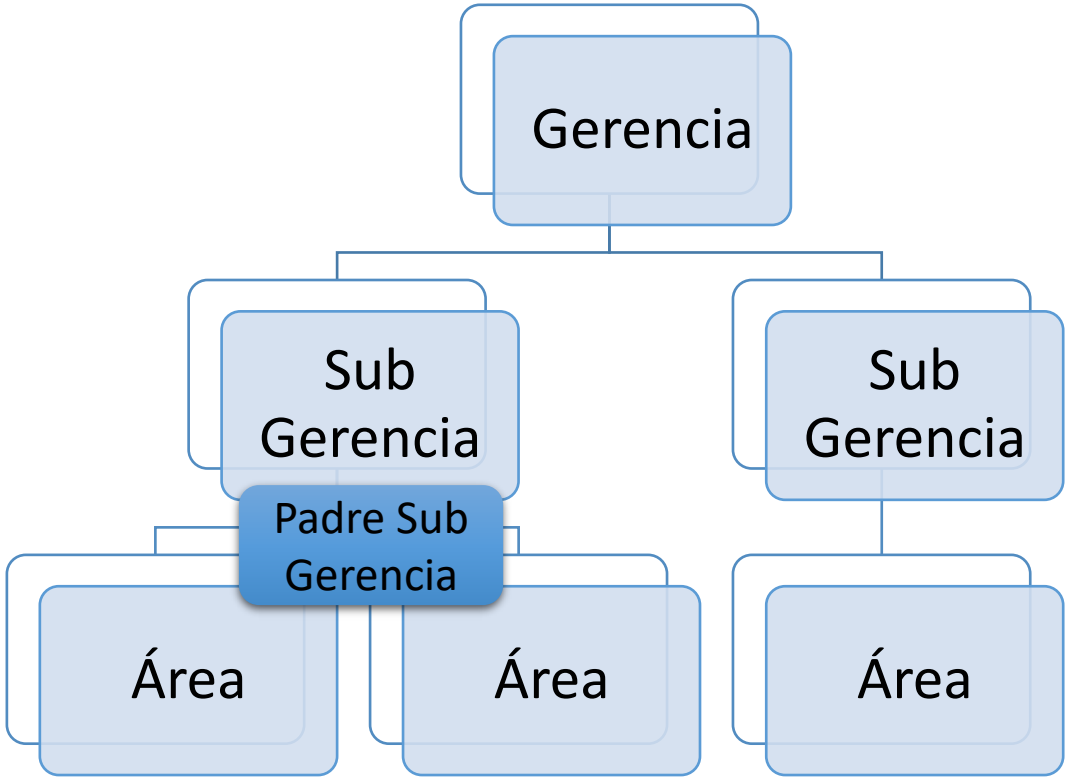
```
DECLARE @EmpIDVariable int;

SELECT @EmpIDVariable = MAX(EmployeeID)
FROM HumanResources.Employee;
```

Instrucciones tipo query jerárquica y recursivas.

Las jerarquías en los sistemas de información, se utilizan para organizar elementos por orden de importancia

Sus elementos exceptuando al elemento raíz tienen un padre y cero o más hijos



Instrucciones tipo query jerárquica y recursivas.

- Eficiente en cuanto a almacenamiento y flexibilidad
- No tiene limitaciones en cuanto a la profundidad del árbol
- No tiene limitaciones en la cantidad de hijos que puede tener cada nodo.

```
SELECT [EmployeeID]
      ,[LastName]
      ,[FirstName]
      ,[Title]
      ,[ReportsTo]
  FROM [dbo].[Employees]
order by ReportsTo
```

```
[ WITH <common_table_expression> [ ,...n ] ]

<common_table_expression>::=
    expression_name [ ( column_name [ ,...n ] ) ]
    AS
    ( CTE_query_definition )
```

	EmployeeID	LastName	FirstName	Title	ReportsTo
1	2	Fuller	Andrew	Vice President, Sales	NULL
2	3	Leverling	Janet	Sales Representative	2
3	4	Peacock	Margaret	Sales Representative	2
4	5	Buchanan	Steven	Sales Manager	2
5	8	Callahan	Laura	Inside Sales Coordinator	2
6	1	Davolio	Nancy	Sales Representative	2
7	9	Dodsworth	Anne	Sales Representative	5
8	6	Suyama	Michael	Sales Representative	5
9	7	King	Robert	Sales Representative	5

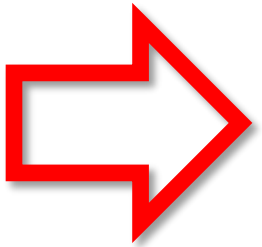
Instrucciones tipo query jerárquica y recursivas.

Cuando una tabla hace referencia a si misma, entonces nos encontramos ante una recursividad, para ello requiere escribir

- Una expresión de tabla
- Unión

Luego de unir el resultado de la tabla 1, esta se une con el resultado de la misma

```
With ListaNros AS
(
SELECT fila = 1
UNION ALL
SELECT fila + 1
FROM ListaNros
WHERE fila + 1 <= 10
)
select fila
from ListaNros
```



fila
1
2
3
4
5
6
7
8
9
10

Instrucciones tipo query jerárquica y recursivas.

En los casos de tablas que representan jerarquías, estas se simplifican al aplicar recursividad.

```
With
cteReports (EmpID, FirstName, LastName, MgrID,
EmpLevel)
as (
select employee_id, first_name, last_name, manager_id,
1
from employees
where Manager_ID is null
union all
select e.employee_id, e.first_name, e.last_name,
e.manager_id, r.EmpLevel + 1
from Employees e
join cteReports r ON e.manager_id = r.EmpId )
select
FirstName + ' ' + LastName as FullName,
EmpLevel,
( select first_name + ' ' + last_name from employees
  where employee_id = cteReports.MgrID) as Manager
from cteReports
Order by EmpLevel, MgrID
```

	EmployeeID	LastName	FirstName	Title	ReportsTo
1	2	Fuller	Andrew	Vice President, Sales	NULL
2	3	Leverling	Janet	Sales Representative	2
3	4	Peacock	Margaret	Sales Representative	2
4	5	Buchanan	Steven	Sales Manager	2
5	8	Callahan	Laura	Inside Sales Coordinator	2
6	1	Davolio	Nancy	Sales Representative	2
7	9	Dodsworth	Anne	Sales Representative	5
8	6	Suyama	Michael	Sales Representative	5
9	7	King	Robert	Sales Representative	5

Uso de ROLLUP, CUBE, GROUPING SET

Si deseamos generan resúmenes de los datos de tipo OLAP con subtotales y totales, tenemos funciones que facilitan estas operaciones

GROUP BY crea agrupaciones que están definidas por un conjunto de expresiones. Cada fila tiene una combinación única, para usar funciones agregadas como **COUNT** o **SUM**

Sin embargo, para agrupar datos por múltiples combinaciones para ello utilice GROUPING SETS junto con la cláusula GROUP BY y definir cada grupo de agrupación dentro de una sola consulta.

```
GROUP BY {
      column-expression
    | ROLLUP ( <group_by_expression> [ ,...n ] )
    | CUBE ( <group_by_expression> [ ,...n ] )
    | GROUPING SETS ( <grouping_set> [ ,...n ]  )
    | () --calculates the grand total
} [ ,...n ]
```

Clausulas que permiten realizar múltiples agregaciones en una única query; con las columnas especificadas en la clausula ***Group BY***

Uso de ROLLUP, CUBE, GROUPING SET

ROLLUP, creara una nueva fila con la suma de cada grupo, además de una fila del gran total

Seafood	Inlagd Sill	14542.60	0
Seafood	Jack's New England Clam...	9098.10	0
Seafood	Konbu	5234.40	0
Seafood	Nord-Ost Matjeshering	14775.54	0
Seafood	Röd Kaviar	4200.00	0
Seafood	Rogede sild	4740.50	0
Seafood	Spegesild	6144.00	0
Seafood	NULL	141623.09	1
NULL	NULL	1354458...	3

CUBE, creara una nueva fila con el nivel de agregación interna. Adiciona un gran total. Y adiciona un total por el nivel de agregación por cada grupo

149	Condiments	Vegie-spread	17696.30	0
150	NULL	Vegie-spread	17696.30	2
151	Grains/Cereals	Wimmers gute S...	23009.00	0
152	NULL	Wimmers gute S...	23009.00	2
153	Confections	Zaanse koeken	4358.60	0
154	NULL	Zaanse koeken	4358.60	2
155	NULL	NULL	135445...	3
156	Beverages	NULL	286526...	1
157	Condiments	NULL	113694...	1
158	Confections	NULL	177099...	1
159	Dairy Products	NULL	251330...	1
160	Grains/Cereals	NULL	100726...	1
161	Meat/Poultry	NULL	178188...	1
162	Produce	NULL	105268...	1
163	Seafood	NULL	141623...	1

Uso de ROLLUP, CUBE, GROUPING SET

GROUPING SETS, creara totales según la agrupación que desee aplicar.

GROUPING_ID, crea una columna lógica que identifica el grupo generado por la cláusula *ROLLUP, CUBE o GROUPING SETS*

```
select grouping_id ( categoria, subcategoria,
       producto) as GrpId,
categoria, subcategoria, producto, sum(qty) qty
from ordenes
group by grouping sets (
  (categoria, subcategoria, producto),
  (categoria, subcategoria),
  (categoria),
  ()
)
order by GrpId, categoria, subcategoria, producto
```

GrpId	categoria	subcategoria	producto	qty
0	Cate1	SubCat1	Producto1	100
0	Cate2	SubCat2	Producto2	200
0	Cate3	SubCat3	Producto3	300
0	Cate4	SubCat4	Producto4	400
1	Cate1	SubCat1	NULL	100
1	Cate2	SubCat2	NULL	200
1	Cate3	SubCat3	NULL	300
1	Cate4	SubCat4	NULL	400
3	Cate1	NULL	NULL	100
3	Cate2	NULL	NULL	200
3	Cate3	NULL	NULL	300
3	Cate4	NULL	NULL	400
7	NULL	NULL	NULL	1000

Manejo de expresiones de tablas (Croos Apply, outer apply).

Las funciones de tipo tabla, son la evolución de las funciones escalares + vistas; al permitir trabajar en bloques y recibir parámetros

- Hay dos tipos de funciones: Aquellas que devuelven un valor escalar (Scalar-valued) y aquellas que devuelven un valor de tipo tabla (Table-valued)
- Las funciones escalares trabajan fila x fila, degradando el performance de la BD
- Las funciones escalares no permiten aplicar paralelismo al trabajar fila a fila.
- Creamos funciones para obtener datos de varias bases de datos.
- Esta función de tipo tabla con un parámetro, le permitirá hacer joins, wheres y orders
- Permitiendo a la BD trabajar en bloques RDBM, y por ello las funciones de este tipo han sido desarrolladas por el fabricante
- A diferencia de las vistas, las funciones de tipo tabla permiten la utilización de parámetros.

Referencias

Pivot
https://docs.microsoft.com/en-us/sql/t-sql/queries/from-using-pivot-and-unpivot?view=sql-server-ver15

Sub Consultas
https://docs.microsoft.com/es-es/sql/relational-databases/performance/subqueries?view=sql-server-ver15

Grouping
https://docs.microsoft.com/en-us/sql/t-sql/queries/select-group-by-transact-sql?view=sql-server-ver15

FROM clause plus JOIN, APPLY, PIVOT (Transact-SQL)
https://docs.microsoft.com/en-us/sql/t-sql/queries/from-transact-sql?view=sql-server-ver16

DIIC Perú | CAPACITACIÓN **PROFESIONAL**

Online | DIIC Perú