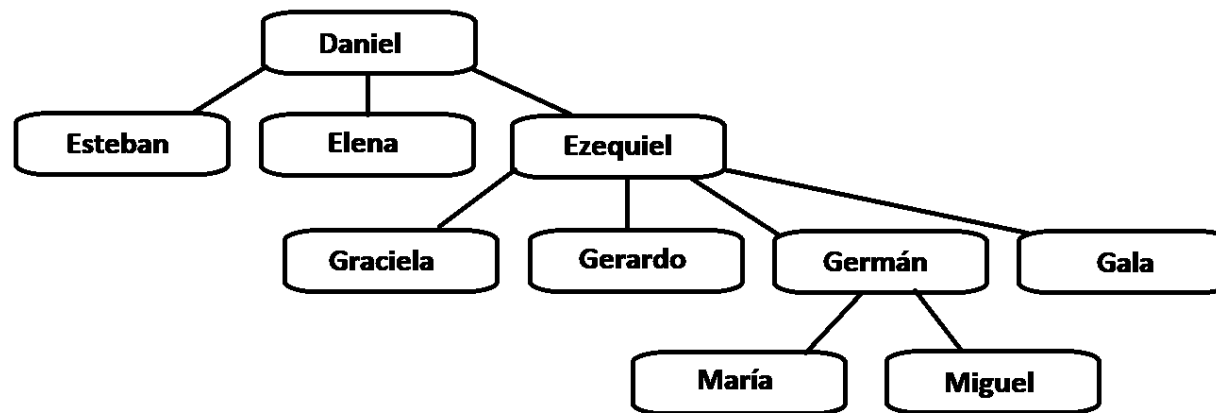


The background features abstract green geometric shapes. On the left, a solid green trapezoid points upwards. On the right, a complex arrangement of overlapping translucent green triangles and polygons creates a layered, crystalline effect. The text is centered in a clean, green, sans-serif font.

Queries recursivos

Introducción

- Deseamos reflejar la siguiente información de los niveles jerárquicos de una organización:



Introducción

- Para esto creamos la siguiente tabla e insertamos las correspondientes filas en ella:

EMPLEADO		
IDPersona	Nombre	IDJefe
1	DANIEL	NULL
2	EZEQUIEL	1
3	ESTEBAN	1
4	ELENA	1
5	GRACIELA	2
6	GERMAN	2
7	GERARDO	2
8	GALA	2
9	MIGUEL	6
10	MARIA	6

Introducción

- Cómo puedo saber el nombre del jefe/jefa de María?

EMPLEADO as E			EMPLEADO as E1		
IDEmpleado	Nombre	IDJefe	IDEmpleado	Nombre	IDJefe
10	MARIA	6	6	GERMAN	2

Introducción

- Cómo puedo saber el nombre del superior del jefe/jefa de María?

EMPLEADO as E			EMPLEADO as E1			EMPLEADO as E2		
IDEmpleado	Nombre	IDJefe	IDEmpleado	Nombre	IDJefe	IDEmpleado	Nombre	IDJefe
10	MARIA	6	6	GERMAN	2	2	EZEQUIEL	1

Introducción

- ▶ Cómo puedo saber el nombre de *todos* los superiores de María??

Queries recursivos

- ▶ Se construirá una tabla virtual (CTE) recursiva, en memoria, que se "alimentará" por la unión de dos queries:
 - ▶ 1) Query no recursivo llamado ancla (anchor).
 - ▶ 2) Query recursivo.

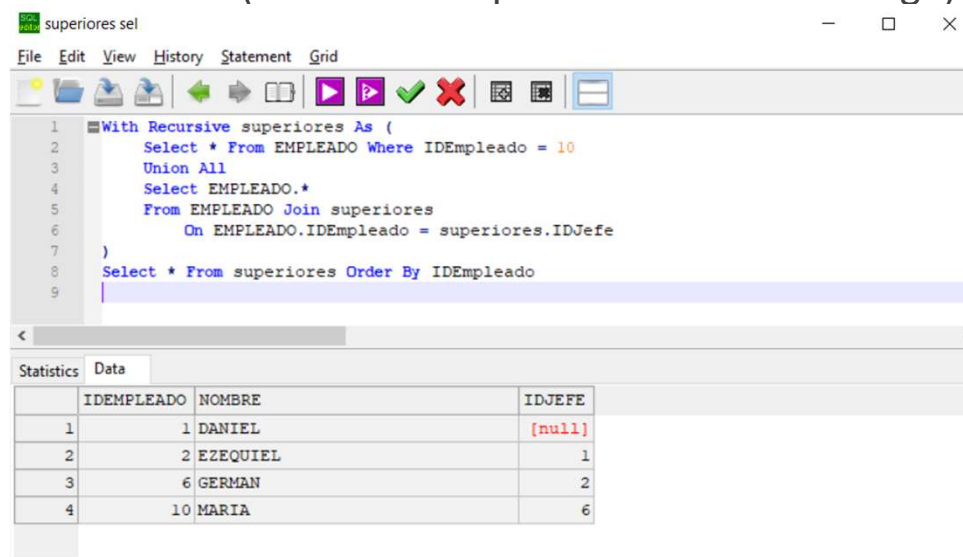
Queries recursivos

- Listar todos los jefes de María (comenzando por los de más alto rango):

```
with recursive superiores as (  
    select * from EMPLEADO where IDEmpleado = 10  
    union all  
    select EMPLEADO.*  
    from EMPLEADO join superiores  
        on EMPLEADO.IDEmpleado = superiores.IDJefe  
)  
select * from superiores order by IDEmpleado
```

Queries recursivos

- Listar todos los jefes de María (comenzando por los de más alto rango):



The screenshot shows a SQL query editor window titled "superiores sel". The query is a recursive Common Table Expression (CTE) designed to find all managers of Maria, starting from the highest rank. The query is as follows:

```
1 With Recursive superiores As (  
2   Select * From EMPLEADO Where IDEmpleado = 10  
3   Union All  
4   Select EMPLEADO.*  
5   From EMPLEADO Join superiores  
6     On EMPLEADO.IDEmpleado = superiores.IDJefe  
7 )  
8 Select * From superiores Order By IDEmpleado  
9
```

Below the query editor, the results are displayed in a table with columns: IDEMPLEADO, NOMBRE, and IDJEFE. The results show the hierarchy of managers for Maria (ID 10).

	IDEMPLEADO	NOMBRE	IDJEFE
1	1	DANIEL	[null]
2	2	EZEQUIEL	1
3	6	GERMAN	2
4	10	MARIA	6

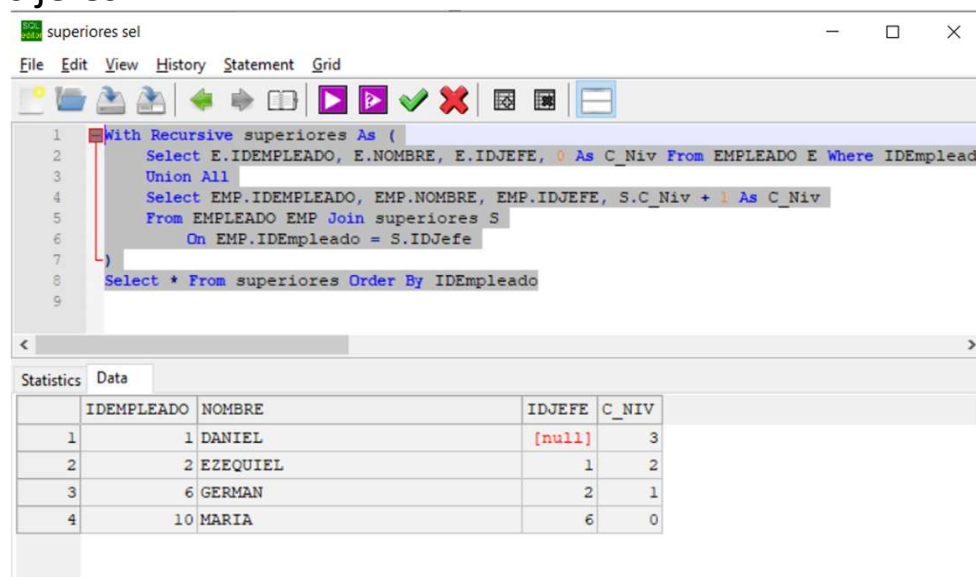
Queries recursivos

- Repetir la consulta anterior incluyendo la cantidad de niveles jerárquicos que separan a María de sus jefes.

```
with recursive superiores as (  
    select E.IDEMPLEADO, E.NOMBRE, E.IDJEFE, 0 as C_Niv from EMPLEADO E  
    where IDEmpleado = 10  
  
    union all  
  
    select EMP.IDEMPLEADO, EMP.NOMBRE, EMP.IDJEFE, S.C_Niv + 1 as C_Niv  
    from EMPLEADO EMP join superiores S  
        on EMP.IDEmpleado = S.IDJefe  
  
)  
select * from superiores order by IDEmpleado
```

Queries recursivos

- Repetir la consulta anterior incluyendo la cantidad de niveles jerárquicos que separan a María de sus jefes.



The screenshot shows a SQL query editor window titled "superiores sel". The query is a recursive query using a Common Table Expression (CTE) to find the hierarchy of employees. The query is as follows:

```
1 With Recursive superiores As (  
2     Select E.IDEMPLEADO, E.NOMBRE, E.IDJEFE, 0 As C_Niv From EMPLEADO E Where IDEmpleado  
3     Union All  
4     Select EMP.IDEMPLEADO, EMP.NOMBRE, EMP.IDJEFE, S.C_Niv + 1 As C_Niv  
5     From EMPLEADO EMP Join superiores S  
6         On EMP.IDEmpleado = S.IDJefe  
7 )  
8 Select * From superiores Order By IDEmpleado  
9
```

The results are displayed in a table with the following columns: IDEMPLEADO, NOMBRE, IDJEFE, and C_NIV. The data is as follows:

IDEMPLEADO	NOMBRE	IDJEFE	C_NIV
1	DANIEL	[null]	3
2	EZEQUIEL	1	2
3	GERMAN	2	1
4	MARIA	6	0

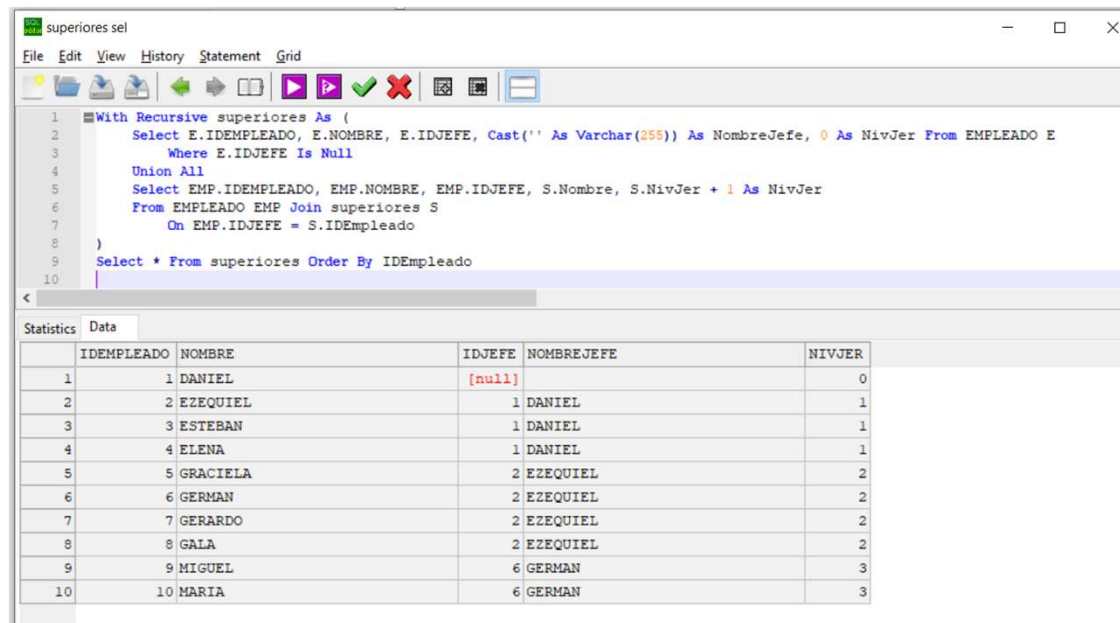
Queries recursivos

- Listar todos los empleados con su jefe inmediato.

```
with recursive superiores as (  
    select E.IDEMPLEADO, E.NOMBRE, E.IDJEFE, CAST('' AS VARCHAR(255)) AS  
    NombreJefe, 0 as NivJer from EMPLEADO E  
    where E.IDJEFE is null  
    union all  
    select EMP.IDEMPLEADO, EMP.NOMBRE, EMP.IDJEFE, S.Nombre, S.NivJer + 1 as  
    NivJer  
    from EMPLEADO EMP join superiores S  
    on EMP.IDJEFE = S.IDEmpleado  
)  
select * from superiores order by IDEmpleado
```

Queries recursivos

- Listar todos los empleados con su jefe inmediato.



The screenshot shows a SQL query editor window titled "superiores sel". The query is a recursive query using a Common Table Expression (CTE) to list all employees along with their immediate supervisors. The query is as follows:

```
1 With Recursive superiores As (  
2   Select E.IDEMPLEADO, E.NOMBRE, E.IDJEFE, Cast('' As Varchar(255)) As NombreJefe, 0 As NivJer From EMPLEADO E  
3   Where E.IDJEFE Is Null  
4   Union All  
5   Select EMP.IDEMPLEADO, EMP.NOMBRE, EMP.IDJEFE, S.Nombre, S.NivJer + 1 As NivJer  
6   From EMPLEADO EMP Join superiores S  
7   On EMP.IDJEFE = S.IDEmpleado  
8 )  
9 Select * From superiores Order By IDEmpleado  
10
```

The results are displayed in a table with the following columns: IDEMPLEADO, NOMBRE, IDJEFE, NOMBREJEFE, and NIVJER. The table contains 10 rows of data, showing the hierarchy of employees and their supervisors.

	IDEMPLEADO	NOMBRE	IDJEFE	NOMBREJEFE	NIVJER
1	1	DANIEL	[null]		0
2	2	EZEQUIEL	1	DANIEL	1
3	3	ESTEBAN	1	DANIEL	1
4	4	ELENA	1	DANIEL	1
5	5	GRACIELA	2	EZEQUIEL	2
6	6	GERMAN	2	EZEQUIEL	2
7	7	GERARDO	2	EZEQUIEL	2
8	8	GALA	2	EZEQUIEL	2
9	9	MIGUEL	6	GERMAN	3
10	10	MARIA	6	GERMAN	3

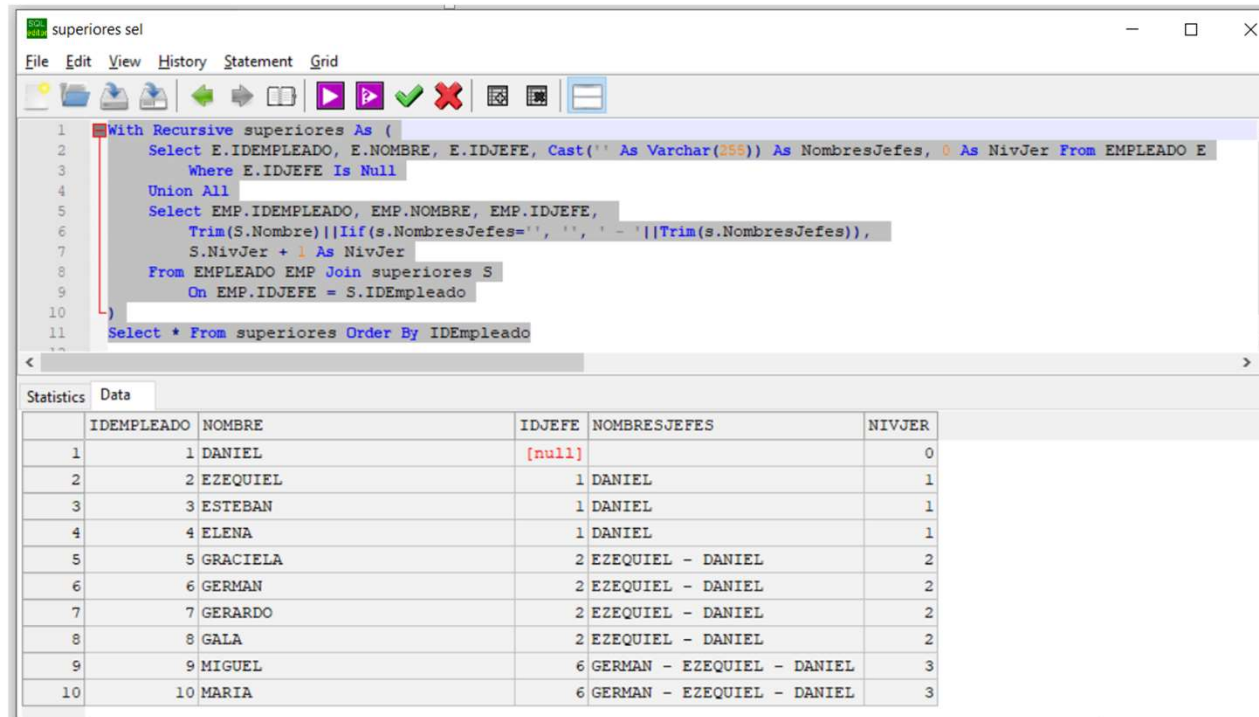
Queries recursivos

- Listar cada empleado con los nombres de todos sus jefes.

```
with recursive superiores as (  
    select E.IDEMPLEADO, E.NOMBRE, E.IDJEFE, CAST('' AS VARCHAR(255)) AS NombresJefes, 0  
    as NivJer from EMPLEADO E  
        where E.IDJEFE is null  
    union all  
    select EMP.IDEMPLEADO, EMP.NOMBRE, EMP.IDJEFE,  
        trim(S.Nombre)||iif(s.NombresJefes='', '', ' - '||trim(s.NombresJefes)),  
        S.NivJer + 1 as NivJer  
    from EMPLEADO EMP join superiores S  
        on EMP.IDJEFE = S.IDEmpleado  
)  
select * from superiores order by IDEmpleado
```

Queries recursivos

- Listar cada empleado con los nombres de todos sus jefes.



The screenshot shows a SQL query editor window titled "superiores sel". The query is a recursive query using a Common Table Expression (CTE) to list employees along with the names of all their superiors. The query is as follows:

```
1 With Recursive superiores As (  
2   Select E.IDEMPLEADO, E.NOMBRE, E.IDJEFE, Cast('' As Varchar(255)) As NombresJefes, 0 As NivJer From EMPLEADO E  
3   Where E.IDJEFE Is Null  
4   Union All  
5   Select EMP.IDEMPLEADO, EMP.NOMBRE, EMP.IDJEFE,  
6     Trim(S.NOMBRE)||Iif(s.NombresJefes='', ' ', ' - '||Trim(s.NombresJefes)),  
7     S.NivJer + 1 As NivJer  
8   From EMPLEADO EMP Join superiores S  
9     On EMP.IDJEFE = S.IDEmpleado  
10 )  
11 Select * From superiores Order By IDEmpleado
```

The results are displayed in a table with the following columns: IDEMPLEADO, NOMBRE, IDJEFE, NOMBRESJEFES, and NIVJER. The table shows 10 rows of data, representing the hierarchy of employees.

IDEMPLEADO	NOMBRE	IDJEFE	NOMBRESJEFES	NIVJER
1	DANIEL	[null]		0
2	EZEQUIEL	1	DANIEL	1
3	ESTEBAN	1	DANIEL	1
4	ELENA	1	DANIEL	1
5	GRACIELA	2	EZEQUIEL - DANIEL	2
6	GERMAN	2	EZEQUIEL - DANIEL	2
7	GERARDO	2	EZEQUIEL - DANIEL	2
8	GALA	2	EZEQUIEL - DANIEL	2
9	MIGUEL	6	GERMAN - EZEQUIEL - DANIEL	3
10	MARIA	6	GERMAN - EZEQUIEL - DANIEL	3

Queries recursivos

- Listar cada empleado con los nombres de todos sus jefes, pero esta vez mediante un stored procedure seleccionable.

```
► CREATE PROCEDURE P_NombresJefes returns (x varchar(1000))  
  
► AS BEGIN  
  
► for with recursive superiores as (  
  
►     select E.IDEMPLEADO, E.NOMBRE, E.IDJEFE, CAST('' AS VARCHAR(255)) AS NombresJefes, 0 as  
NivJer from EMPLEADO E  
  
►     where E.IDJEFE is null union all  
  
►     select EMP.IDEMPLEADO, EMP.NOMBRE, EMP.IDJEFE,  
  
►         trim(S.Nombre)||iif(s.NombresJefes='', '', ' - '||trim(s.NombresJefes)), S.NivJer + 1  
as NivJer  
  
►     from EMPLEADO EMP join superiores S  
  
►     on EMP.IDJEFE = S.IDEmpleado  
  
► )  
  
► select nombresjefes from superiores order by nombresjefes  
  
► into :x do suspend;
```

Queries recursivos

SQL editor: sel P_NombresJefes

File Edit View History Statement Grid

1 Select * From P_NombresJefes

< >

Statistics Data

	X
1	
2	DANIEL
3	DANIEL
4	DANIEL
5	EZEQUIEL - DANIEL
6	EZEQUIEL - DANIEL
7	EZEQUIEL - DANIEL
8	EZEQUIEL - DANIEL
9	GERMAN - EZEQUIEL - DANIEL
10	GERMAN - EZEQUIEL - DANIEL

Queries recursivos

- ▶ Finalmente replicar este mismo listado a través de una vista.

```
▶ CREATE VIEW V_NombresJefes AS

▶ with recursive superiores as (

▶     select E.IDEMPLEADO, E.NOMBRE, E.IDJEFE, CAST('' AS VARCHAR(255)) AS
NombresJefes, 0 as NivJer from EMPLEADO E

▶     where E.IDJEFE is null

▶     union all

▶     select EMP.IDEMPLEADO, EMP.NOMBRE, EMP.IDJEFE,

▶         trim(S.Nombre)||iif(s.NombresJefes='', '', ' - '||trim(s.NombresJefes)),

▶         S.NivJer + 1 as NivJer

▶     from EMPLEADO EMP join superiores S

▶         on EMP.IDJEFE = S.IDEmpleado

▶ ) select * from superiores order by nombresjefes;
```

Queries recursivos

SQL Editor: sel V_NombresJefes

File Edit View History Statement Grid

1 Select * From V_NombresJefes

Statistics Data

	IDEMPLEADO	NOMBRE	IDJEFE	NOMBRESJEFES	NIVJER
1	1	DANIEL	[null]		0
2	4	ELENA	1	DANIEL	1
3	3	ESTEBAN	1	DANIEL	1
4	2	EZEQUIEL	1	DANIEL	1
5	8	GALA	2	EZEQUIEL - DANIEL	2
6	7	GERARDO	2	EZEQUIEL - DANIEL	2
7	6	GERMAN	2	EZEQUIEL - DANIEL	2
8	5	GRACIELA	2	EZEQUIEL - DANIEL	2
9	10	MARIA	6	GERMAN - EZEQUIEL - DANIEL	3
10	9	MIGUEL	6	GERMAN - EZEQUIEL - DANIEL	3

Queries recursivos

► Fuentes:

- Usando recursividad con CTE - Ojeda Valiente, Walter - <https://firebird21.wordpress.com/2015/08/22/usando-recursividad-con-cte/>
- Firebird 2.1 Language Reference Update - <https://firebirdsql.org/refdocs/langrefupd21.html>
- Managing recursive, tree-like data structures with Firebird - <http://www.firebirdsql.org/file/community/ppts/fbcon11/FBTrees2011.pdf>