

# Introduction to recursive CTE

HIERARCHICAL AND RECURSIVE QUERIES IN SQL SERVER



**Dominik Egarter**  
Data Engineering Enthusiast

# The recursive CTE

Consists of 4 parts:

```
WITH cte_name AS (  
    -- Anchor member execute only once  
    initial_query  
  
    UNION ALL  
  
    -- Recursive member execute repeatedly until no rows are returned  
    recursive_query  
)  
  
SELECT *  
    FROM cte_name
```

# Guide to use recursive CTE

- For more than 200 recursion steps, increase the number of recursion steps,
  - set `OPTION(MAXRECURSION 32767)`
- The following SQL statements are not allowed: `GROUP BY` , `HAVING` , `LEFT JOIN` , `RIGHT JOIN` , `OUTER JOIN` , `SELECT DISTINCT` , `Subqueries` , `TOP`
- The number of columns for anchor and recursive member are the same.
- The data types of anchor and recursive member are the same

# Simple recursive example

Calculating the factorial:

The factorial of `n` is defined by the product of all positive integers less than or equal to `n` :

$$3! = 1 \times 2 \times 3 = 6$$

The factorial `n!` is defined recursively as follows:

- `0! = 1` for iteration = 1
- `(n+1)! = n! * (iteration+1)` for iteration > 1

# Simple recursive example

```
WITH recursion AS
  (SELECT 1 AS iterationCounter, 1 AS factorial
   UNION ALL
   SELECT iterationCounter+1, factorial * (iterationCounter+1)
    FROM recursion
    WHERE iterationCounter < 10 )
SELECT factorial
FROM recursion
```

3628800

# Let's practice!

HIERARCHICAL AND RECURSIVE QUERIES IN SQL SERVER

# Working with recursive queries

HIERARCHICAL AND RECURSIVE QUERIES IN SQL SERVER



**Dominik Egarter**  
Data Engineering Enthusiast

# The hierarchy of an IT-organization

The organization is described by:

- `ID` - Employee ID
- `Name` of Employee
- `JobTitle` in the company
- `Department` in the company
- `Supervisor` in the company

Fields describing hierarchy:

1. `ID`
2. `Supervisor`



# The IT-organization

```
+-----+-----+-----+-----+
|ID | Name           | Position           | Department | Supervisor |
|-----|-----|-----|-----|
| 1 | Heinz Griesser  | IT Director        | IT         | 0          |
| 2 | Andreas Sitter  | Security Manager   | IT         | 1          |
| 3 | Thomas Bergman  | Innovation Manager | IT         | 1          |
| 4 | Hannes Berg     | Operation Manager  | IT         | 1          |
| 5 | Anna Krugge1    | Administrator      | IT         | 4          |
| 6 | Karin Pacher    | Developer          | IT         | 4          |
+-----+-----+-----+-----+
```

# Common tasks for hierarchical data

## Get the hierarchy of a record

- Who is your supervisor?

---

## Get the level of the hierarchy

- Get the hierarchy level of an organization

---

## Combine recursion results into one field

- Which supervisors do I have?

# Get the hierarchy

```
WITH hierarchy AS (  
    SELECT ID, Supervisor  
        FROM employee  
        WHERE supervisor = 0  
  
    UNION ALL  
  
    SELECT emp.ID, emp.Supervisor  
        FROM employee emp  
  
    JOIN employeeHierarchy  
        ON emp.Supervisor = employeeHierarchy.ID)  
  
SELECT *  
    FROM hierarchy
```

# Get the hierarchy

```
+---+-----+
|ID |Supervisor |
|---|-----|
|1  | 0          |
|2  | 1          |
|3  | 2          |
+---+-----+
```

# Get the level of the hierarchy

```
WITH hierarchy AS (  
    SELECT ID, Supervisor, 1 as LEVEL  
        FROM employee  
        WHERE Supervisor = 0  
  
    UNION ALL  
  
    SELECT emp.ID, emp.Supervisor, LEVEL + 1  
        FROM employee emp           increase every recursion step  
  
    JOIN hierarchy  
    ON emp.Supervisor = hierarchy.ID  
  
)  
SELECT *  
    FROM hierarchy
```

# Get the level of the hierarchy

```
+---+-----+-----+
|ID |Supervisor | Level |
|---|-----|-----|
|1  | 0          | 0     |
|2  | 1          | 1     |
+---+-----+-----+
```

# Combine recursion results into one field

```
WITH hierarchy AS (  
    SELECT ID, Supervisor, CAST('0' AS VARCHAR(MAX)) as PATH  
    FROM employee  
    WHERE Supervisor = 0  
  
    UNION ALL  
  
    SELECT emp.ID, emp.Supervisor, Path + '->' + CAST(emp.Supervisor AS VARCHAR(MAX))  
    FROM employee emp  
    INNER JOIN hierarchy  
    ON emp.Supervisor = hierarchy.ID  
)  
SELECT *  
FROM hierarchy
```

# Combine recursion results into one field

```
+-----+-----+
| PATH          |
|-----|
| 0 -> 1 -> 4  |
+-----+-----+
```



# Let's query the IT- organization

HIERARCHICAL AND RECURSIVE QUERIES IN SQL SERVER

# Analyze the family tree

HIERARCHICAL AND RECURSIVE QUERIES IN SQL SERVER



**Dominik Egarter**  
Data Engineering Enthusiast

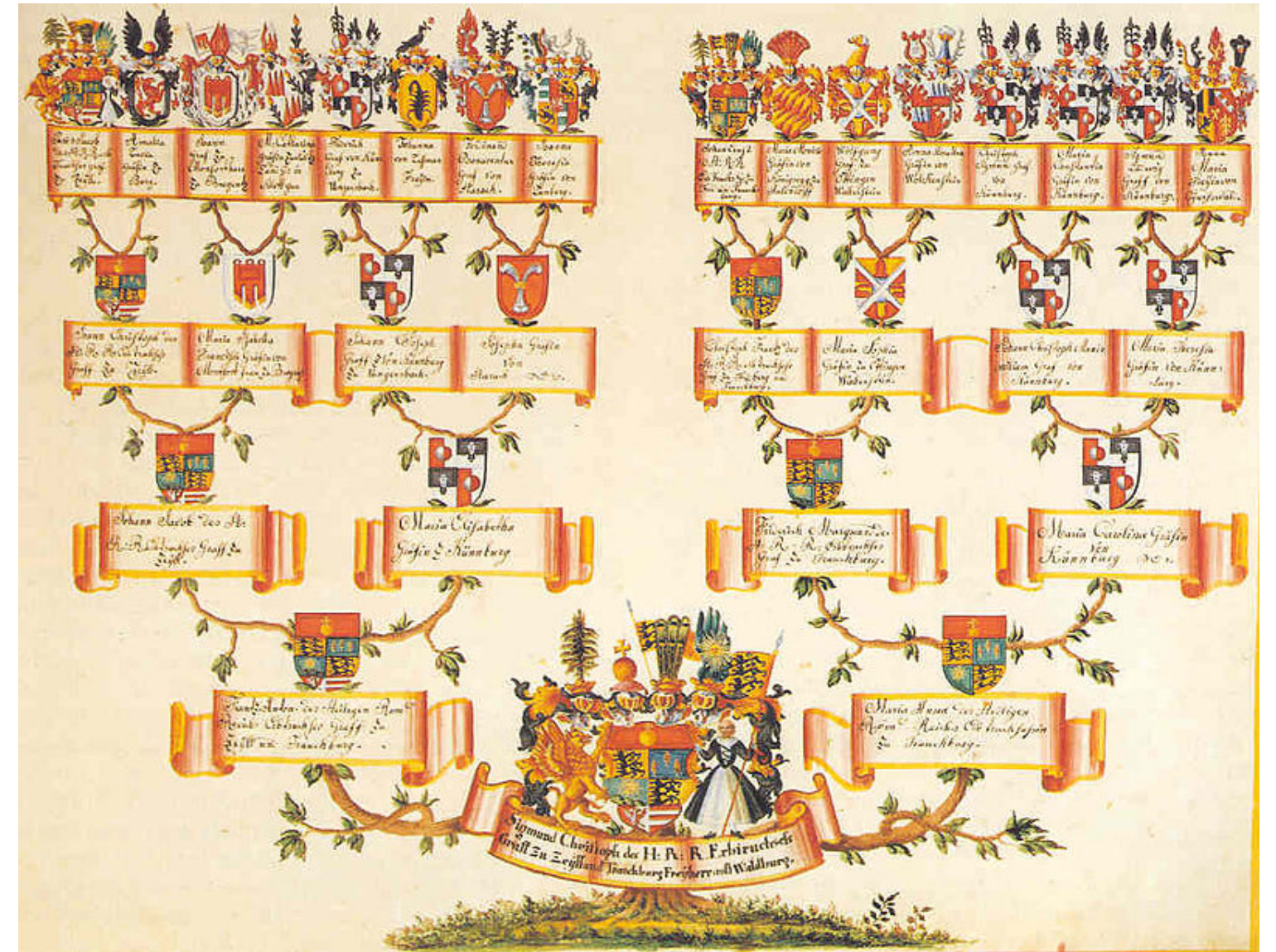
# The family tree

The family tree is described by:

- ID of the person
- Name of the person
- parentID the ID of the parent

The elements describing the hierarchy:

- ID
- parentID



# Putting it all together

Remember the following principles about recursive CTEs:

- Initialize the recursion in the anchor member
- Implement the recursion function in the recursion member
- Define a termination condition

Remember the following working principles:

- Get the level of recursion
- Combine the recursion function into one field

# Questions about the family tree

Get the number of generations

- Define the `LEVEL`

```
-- Anchor member  
0 as LEVEL  
-- Recursive member  
LEVEL + 1
```

- Count the number of LEVELS to get generations `COUNT(LEVEL)`

```
Generations:  
100
```

# Questions about the family tree

Get all possible parents in one field

- Combine recursion results into one field

```
-- Anchor member
CAST(ID AS VARCHAR(MAX)) as Parent
-- Recursive member
Parent + ' -> ' + CAST(parentID AS VARCHAR(MAX))
```

```
+-----+
| Name          | Parent          |
+-----+-----+
|Dominik Egarter | 100 -> 101 -> 102 ->103 |
+-----+-----+
```

# Let's check the family tree

HIERARCHICAL AND RECURSIVE QUERIES IN SQL SERVER