Recursion in SQL

Nonlinear and Mutual Recursion
SQL With Recursive Statement

With Recursive

\[ R_1 \text{ As (query-1)}, \]
\[ R_2 \text{ As (query-2)}, \]
\[ \ldots \]
\[ R_n \text{ As (query-n)} \]

<query involving \( R_1, \ldots, R_n \) (and other tables)>

Nonlinear & Mutual Recursion
SQL With Recursive Statement

With Recursive

\[ R \text{ As } ( \text{base query Union recursive query} ) \]

<query involving R (and other tables)>
Linear Recursion

With Recursive

\[ R \text{ As ( base query Union recursive query )} \]

\(<\text{query involving } R \text{ (and other tables)}>\)
Example: Ancestors

ParentOf(parent, child)  

Find all of Mary’s ancestors

with recursive
    Ancestor(a,d) as (select parent as a, child as d from ParentOf
    union
    select Ancestor.a, ParentOf.child as d
    from Ancestor, ParentOf
    where Ancestor.d = ParentOf.parent)

select a from Ancestor where d = 'Mary';
Example: Ancestors

ParentOf(parent, child)  Find all of Mary’s ancestors

with recursive
  Ancestor(a,d) as (select parent as a, child as d from ParentOf
  union
  select A1.a, A2.d
  from Ancestor A1, Ancestor A2
  where A1.d = A2.a)

select a from Ancestor where d = 'Mary';
Example: Ancestors

ParentOf(parent, child)

- Nonlinear (versus linear)
  - Query looks cleaner
  - Converges faster
  - Harder to implement

Find all of Mary's ancestors
SQL With Recursive Statement

With Recursive

\[ \text{R}_1 \text{ As (query-1)}, \]
\[ \text{R}_2 \text{ As (query-2)}, \]
\[ \ldots \]
\[ \text{R}_n \text{ As (query-n)} \]

<query involving \text{R}_1,\ldots,\text{R}_n \text{ (and other tables)>
Mutual Recursion

With Recursive

- $R_1$ As (query-1),
- $R_2$ As (query-2),

... 

$R_n$ As (query-n)

<query involving $R_1, ..., R_n$ (and other tables)>
Example: Hubs & Authorities

**Link**(src, dest)  
**HubStart**(node)  **AuthStart**(node)

Hub points to $\geq 3$ Authority

Authority pointed to $\geq 3$ Hub
Example: Hubs & Authorities

Link(src,dest)
HubStart(node) AuthStart(node)

```sql
with recursive
    Hub(node) as (select node from HubStart
                   union
                   select src as node from Link L
                   where dest in (select node from Auth
                                   group by src having count(*) >= 3)),
    Auth(node) as (select node from AuthStart
                   union
                   select dest as node from Link L
                   where src in (select node from Hub
                                  group by dest having count(*) >= 3))
select * from Hub;
```
Example: Hubs & Authorities

```sql
with recursive
    Hub(node) as
        (select node from HubStart
         union
         select src as node from Link L
         where src not in (select node from Auth)
         and dest in (select node from Auth)
         group by src having count(*) >= 3),
    Auth(node) as
        (select node from AuthStart
         union
         select dest as node from Link L
         where dest not in (select node from Hub)
         and src in (select node from Hub)
         group by dest having count(*) >= 3)
select * from Hub;
```

(Nonlinear & Mutual Recursion)

- Depends negatively on other relation.
Example: Recursion with Aggregation

P(x) ←

```
WITH recursive
  R(x) AS (SELECT x FROM P
            UNION
            SELECT sum(x) FROM R)
SELECT * FROM R;
```

R: P, sum(P)

P: 1, 2
R: 1, 2, 3, 6, 9
SQL With Recursive Statement

With Recursive

- \( R_1 \) As (query-1),
- \( R_2 \) As (query-2),
- \( \ldots \)
- \( R_n \) As (query-n)

<query involving \( R_1, \ldots, R_n \) (and other tables)>

Extends expressiveness of SQL

- Basic functionality: linear recursion
- Extended functionality: nonlinear recursion, mutual recursion
- Disallowed: recursive subqueries (negative), aggregation