



Recursion in SQL

Nonlinear and Mutual Recursion

SQL With Recursive Statement

With Recursive

R1 AS (query-1),

R2 AS (query-2),



...

Rn AS (query-n)

<query involving R1,...,Rn (and other tables)>

SQL With Recursive Statement

With Recursive

R As (**base query** 
Union
recursive query) 

<query involving **R** (and other tables)>

Linear Recursion

With Recursive

R As (base query

Union

recursive query)

<query involving R (and other tables)>

one reference
to R

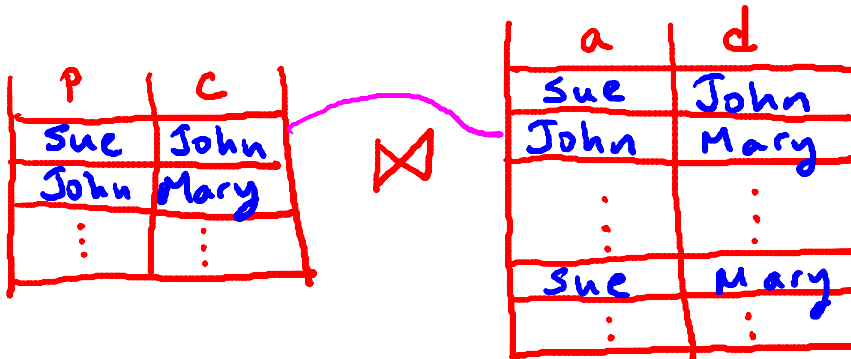
Example: Ancestors

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';
```

P	C
Sue	John
John	Mary
⋮	⋮

a	d
Sue	John
John	Mary
⋮	⋮
Sue	Mary

Nonlinear

Example: Ancestors

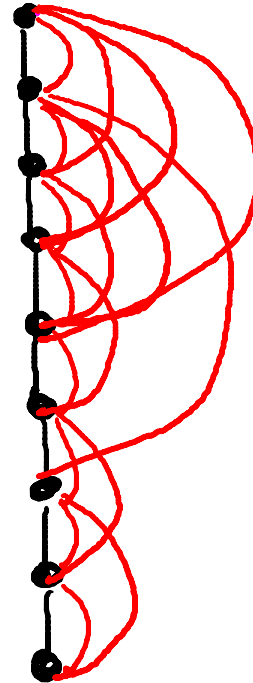
ParentOf(parent, child)

Find all of Mary's ancestors

❖ Nonlinear (versus linear)

- + Query looks cleaner
- + Converges faster
- Harder to implement

SQL standard only requires linear



SQL With Recursive Statement

With Recursive

R1 AS (**query-1**),

R2 AS (**query-2**),



...

Rn AS (**query-n**)

<query involving **R1**,...,**Rn** (and other tables)>

Mutual Recursion

With Recursive

— **R1** AS (**query-1**),  **R2**
— **R2** AS (**query-2**),  **R1**

...

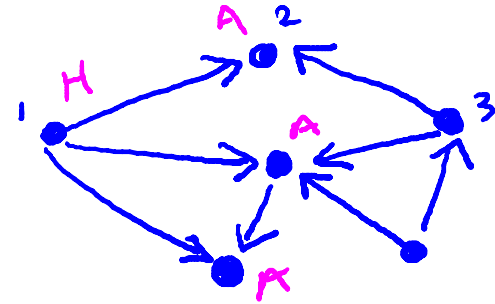
Rn AS (**query-n**)

<query involving **R1**, ..., **Rn** (and other tables)>

Example: Hubs & Authorities

Link(src,dest)

HubStart(node) AuthStart(node)



Hub points to ≥ 3 Authority

Authority pointed to ≥ 3 Hub

Example: Hubs & Authorities

Link(src,dest) ←

HubStart(node) AuthStart(node)

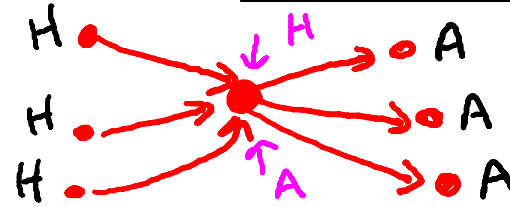
```

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;
  
```

point
to
≥ 3
Auth.

pointed
to
≥ 3
Hubs

Example: Hubs & Authorities



```

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;
  
```

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;
```

$R: P, \text{sum}(P)$

$P: 1, 2$

$R: 1, 2, \cancel{3}, \cancel{6} 9$

SQL With Recursive Statement

With Recursive

R1 AS (query-1), ← R1
↺ R2 AS (query-2), ↻

...

Rn AS (query-n)

<query involving R1, ..., Rn (and other tables)>

Extends expressiveness of SQL

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