Learning Objectives

- After this segment, students will be able to
  - Describe RECURSIVE statement in SQL3
  - Use it to query Graphs
Querying Graphs: Overview

- Relational Algebra
  - Can not express transitive closure queries

- Two ways to extend SQL to support graphs
  1. Abstract Data Types
  2. Custom Statements
     - SQL2 - CONNECT clause(s) in SELECT statement
     - SQL3 - WITH RECURSIVE statement
WITH RECURSIVE: Input, Output

- **Input:**
  - (a) Edges of a directed graph G
  - (b) Sub-queries to
    - Initialize results
    - Recursively grow results
    - Additional constraints

- **Output:** Transitive closure of G
  - Ex. Predecessors of a node
  - Ex. Successors of a node
Syntax of WITH RECURSIVE Statement

WITH RECURSIVE X(source, dest) AS
    (SELECT source, dest FROM R)
UNION
    (SELECT R.source, X.dest FROM R, X
     WHERE R.dest = X.source)

Description of Result Table

Initialization Query

Recursive Query to grow result
WITH RECURSIVE X(source,dest)
AS (SELECT source,dest FROM R)
UNION
(SELECT R.source, X.dest FROM R, X WHERE R.dest=X.source)

(a) Graph G

(b) Relation form

<table>
<thead>
<tr>
<th>SOURCE</th>
<th>DEST</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>

(c) Transitive closure (G) = Graph G
(d) Transitive closure in relation form
**SQL3 Recursion Example - Meaning**

- Initialize $X$ by
  
  $$\text{SELECT source,dest FROM } R$$

- Recursively grow $X$ by
  
  $$\text{SELECT } R\.source, X\.dest
  \text{ FROM } R, X
  \text{ WHERE } R\.dest=X\.source$$

- Infer $X(a,c)$ from $R(a,b), X(b,c)$
  - Infer $X(1,3)$ from $R(1,2), X(2,3)$
  - Infer $X(2,4)$ from $R(2,3), X(3,4)$
  - Infer $X(5,4)$ from $R(5,3), X(3,4)$
  - Infer $X(1,4)$ from $R(1,5), X(5,4)$