CS145 Lecture Notes #13
SQL3 Recursion

Introduction

Example schema: ParentChild(parent, child)
Example data:

('Homer', 'Bart');
('Homer', 'Lisa');
('Marge', 'Bart');
('Marge', 'Lisa');
('Abe', 'Homer');
('Ape', 'Abe');

Example query: find all of Bart’s ancestors

“Ancestor” has a recursive definition:

SQL2 does not support recursive queries:
• Need to write PL/SQL or embedded SQL
SQL3 supports recursive queries:
• WITH statement
  — First, define AncestorDescendent(ancestor, descendent)
  — Then, find Bart’s ancestors

WITH
  RECURSIVE AncestorDescendent(ancestor, descendent) AS
    (SELECT * FROM ParentChild)
  UNION
    (SELECT ad1.ancestor, ad2.descendent
     FROM AncestorDescendent ad1, AncestorDescendent ad2
     WHERE ad1.descendent = ad2.ancestor)
  SELECT ancestor
  FROM AncestorDescendent
  WHERE descendent = 'Bart';

SQL3 only requires support of linear recursion: each RECURSIVE definition has at most one reference to a recursively-defined relation

Can we make the above query linear?
Fixed-Point Semantics

Analogy in Mathematics

If \( f : \tau \rightarrow \tau \) is a function from some type \( \tau \) to itself, a fixed point of \( f \) is a value \( x \) of type \( \tau \) such that \( f(x) = x \)

Example: what is the fixed point of \( f(x) = x/2 \)?

A numerical method to compute fixed point of \( f \):

- Start with a “seed” \( x_0 \): \( x \leftarrow x_0 \)
- Compute \( f(x) \)
  - If \( f(x) = x \) (numerically), stop; \( x \) is a fixed point of \( f \)
  - Otherwise, \( x \leftarrow f(x) \); repeat

Example: compute the fixed point of \( f(x) = x/2 \) given seed 1

Fixed Point of a Recursive Query

Think of a query \( q \) as a function that takes one table as input and computes another as output: a fixed point of \( q \) is a table \( t \) such that \( q(t) = t \)

To compute fixed point of \( q \):

- Start with an empty table: \( t \leftarrow \emptyset \)
- Evaluate the query \( q \) over the current contents of \( t \)
  - If the query result is identical to \( t \), stop; \( t \) is a fixed point
  - Otherwise, \( t \leftarrow \) the query result; repeat

Example: compute \text{AncestorDescendent} \text{ (using the linear version)}

Intuition: why does fixed-point iteration give us the right answer?

- Initially, we know nothing about ancestor-descendent relationships
- In Round 1, we deduce that parents and children are ancestors and descendents
- In each subsequent round, we use the facts deduced in previous rounds to get more ancestor-descendent relationships
- We stop when no new facts can be proven
Operational Semantics of WITH Statement

General syntax:

\[
\text{WITH}
\]

\[
\text{RECURSIVE } R_1 \text{ AS } Q_1, \ldots,
\]

\[
\text{RECURSIVE } R_n \text{ AS } Q_n
\]

\[
Q;
\]

\[
\sim \text{ Note that } Q, Q_1, \ldots, Q_n \text{ may refer to } R_1, \ldots, R_n
\]

Operational semantics:

1. \( R_1 \leftarrow \emptyset, \ldots, R_n \leftarrow \emptyset \)
2. Evaluate \( Q_1, \ldots, Q_n \) using the current contents of \( R_1, \ldots, R_n \):
\[
R_1^{\text{new}} \leftarrow Q_1, \ldots, R_n^{\text{new}} \leftarrow Q_n
\]
3. If \( R_i^{\text{new}} \neq R_i \) for some \( i \):
   3.1. \( R_1 \leftarrow R_1^{\text{new}}, \ldots, R_n \leftarrow R_n^{\text{new}} \)
   3.2. Go to 2.
4. Compute \( Q \) using the current contents of \( R_1, \ldots, R_n \), and output the result

Example: find Bart’s ancestors

Monotonicity & Recursion

Suppose that query \( Q \) is posed over table \( R \) (and perhaps other tables):

- \( Q \) is monotone with respect to \( R \) if adding tuples to \( R \) can never cause any tuple to be removed from the result of \( Q \)
- \( Q \) is not monotone with respect to \( R \) if adding tuples to \( R \) might cause some tuple to be removed from the result of \( Q \)

Example schema: Student (\( \text{SID} \), name, age, GPA)
Example data: (123, ‘Bart’, 10, 3.0), (456, ‘Lisa’, 8, 4.0)
Example: students with GPA higher than 3.9

Example: students with the lowest GPA

\[
\sim \text{ What if we insert } (987, \text{‘Nelson’}, 10, 2.0)\?
\]

“Bad mix” of nonmonotonicity and recursion cause problems

Example: reward students with GPA higher than 3.9
- Those not on Dean’s List should get a scholarship
- Those without scholarships should be on Dean’s List
WITH
  RECURSIVE Scholarship(SID) AS -- Q1
  (SELECT * FROM R) UNION SELECT * FROM Q,
RECURSIVE DeansList(SID) AS -- Q2

...  

- Q1 is not monotone with respect to DeansList
- Q2 is not monotone with respect to Scholarship

~→ Problem: minimal fixed point is not unique

~→ Problem: fixed-point iteration does not converge

**Dependency Graph**

- One node for each table
- A directed arc $R \rightarrow S$ if $R$ is defined in terms of $S$
- Label the directed arc “—” if the query defining $R$ is not monotone with respect to $S$

Requirement for legal SQL3 recursion: no cycle containing a “—” arc

Legal example: find Bart’s ancestors

Illegal example: reward students with GPA higher than 3.9

A more subtle example:

```
WITH RECURSIVE P(x) AS 
  (SELECT * FROM R) UNION (SELECT * FROM Q),
RECURSIVE Q(x) AS 
  SELECT SUM(x) FROM P
...
```
Stratified Recursion

The *stratum* of a node $R$ is the maximum number of “—” arcs on any path from $R$ in the dependency graph.

Example: find Bart’s ancestors
- Stratum of ParentChild:
- Stratum of AncestorDescendent:

Example: reward students with GPA higher than 3.9
- Stratum of Student:
- Stratum of Scholarship:
- Stratum of DeansList:

Example: find all pairs of persons with no common ancestors

WITH
RECURSIVE AncestorDescendent(ancestor, descendent) AS
(SELECT * FROM ParentChild)
UNION
(SELECT ad.ancestor, pc.child
FROM AncestorDescendent ad, ParentChild pc
WHERE ad.descendent = pc.parent),

Person(person) AS

RECURSIVE NoCommonAncestor(person1, person2) AS

SELECT * FROM NoCommonAncestor;

- Dependency graph:

- Stratum of ParentChild:
- Stratum of AncestorDescendent:
- Stratum of Person:
- Stratum of NoCommonAncestor:
A WITH statement is *stratified* if every node as a finite stratum

\[\Rightarrow\text{ Requirement for legal SQL3 recursion (rephrased): WITH is stratified}\]

**Operational Semantics of Stratified WITH Statement**

- Compute tables lowest-stratum-first
- For each stratum, use fixed-point iteration on all tables in that stratum

Example: find all pairs of persons with no common ancestors

- Stratum 0:
- Stratum 1: