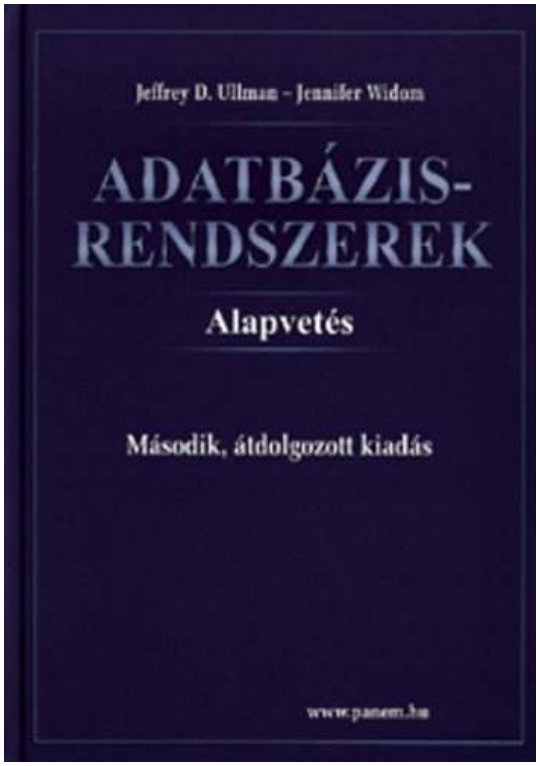
Rekurzió a Datalogban és az SQL3-ban

Tankönyv: Ullman-Widom:
Adatbázisrendszerek Alapvetés
Második, átdolgozott kiadás,
Panem, 2009



10.2. Rekurzió. Az Eljut feladat

Expressive Power of Datalog

- Without recursion, Datalog can express all and only the queries of core relational algebra.
  - The same as SQL select-from-where, without aggregation and grouping.
- But with recursion, Datalog can express more than these languages.

Milyen fontos rekurzív feladatok vannak?

I. Hierarchiák bejárása

- Leszármazottak-ősök ParentOf(parent,child)
  - Find all of Mary's ancestors
- Vállalati hierarchia felettes-beosztott

  Employee(ID,salary)

  Manager(mID,eID)

  Project(name,mgrID)
  - Find total salary cost of project 'X'
- Alkatrész struktúra (mely alkatrésznek mely alkatrész része)

Milyen fontos rekurzív feladatok vannak?

II. Gráf jellegű bejárások

- Repülőgép járatok, eljut-feladat

  Flight(orig,dest,airline,cost)
  - Find cheapest way to fly from ‘A’ to ‘B’
- Közösségi hálók

  Ki-kinek az ismerőse, Twitterben ki-kit követ

Kiegészítés a gráf adatbázisokról

- Gráfok könnyen megadhatók relációs táblával, a gráf lekérdezések egyre gyakoribb feladatok, ezek relációs megoldása hatékonysági kérdés. Vannak kimondottan gráf-adatbázisok.

A Recursive Example

- EDB: Par(c,p) = *p* is a parent of *c*.
- Generalized cousins: people with common ancestors one or more generations back:

Sib(x,y) ← Par(x,p) AND Par(y,p) AND x<>y

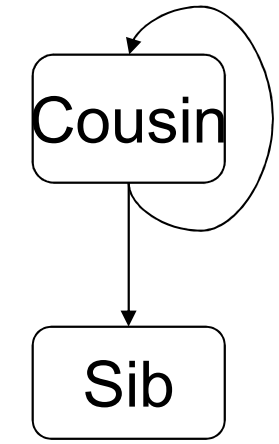
Cousin(x,y) ← Sib(x,y)

Cousin(x,y) ← Par(x,xp) AND Par(y,yp)
AND Cousin(xp,yp)

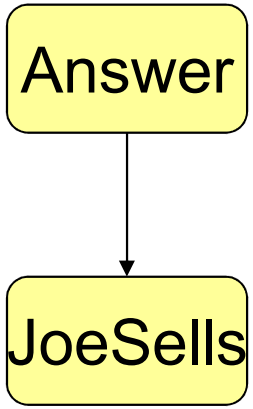
Definition of Recursion

- Megelőzési gráf (IDB relációk közötti élek)
- Form a dependency graph whose
- Nodes = IDB predicates.
- Arc *X* ->*Y* if and only if there is a rule with *X* in the head and *Y* in the body.
- Cycle = recursion; no cycle = no recursion.

Example: Dependency Graphs



Recursive

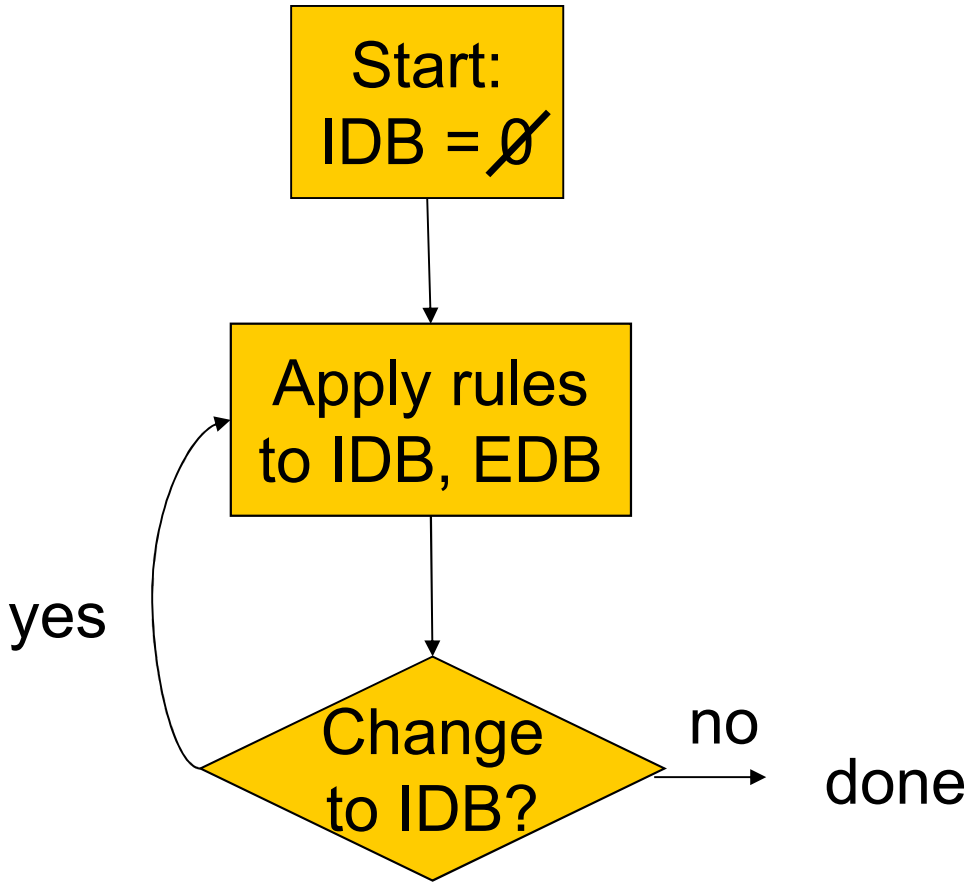


Nonrecursive

Evaluating Recursive Rules

- The following works when there is no negation:
  1. Start by assuming all IDB relations are empty.
  2. Repeatedly evaluate the rules using the EDB and the previous IDB, to get a new IDB.
  3. End when no change to IDB.

The "Naïve" Evaluation Algorithm



Seminaive Evaluation

- Since the EDB never changes, on each round we only get new IDB tuples if we use at least one IDB tuple that was obtained on the previous round.
- Saves work; lets us avoid rediscovering *most* known facts.
  - A fact could still be derived in a second way.

Example: Evaluation of Cousin

- We'll proceed in rounds to infer Sib facts and Cousin facts.

  Remember the rules:

  Sib(x,y) ← Par(x,p) AND Par(y,p) AND x<>y

  Cousin(x,y) ← Sib(x,y)

  Cousin(x,y) ← Par(x,xp) AND Par(y,yp)
  AND Cousin(xp,yp)

Par Data: Parent Above Child

Sib(x,y) ← Par(x,p) AND Par(y,p) AND x<>y

Cousin(x,y) ← Par(x,xp) AND Par(y,yp) AND Cousin(xp,yp)

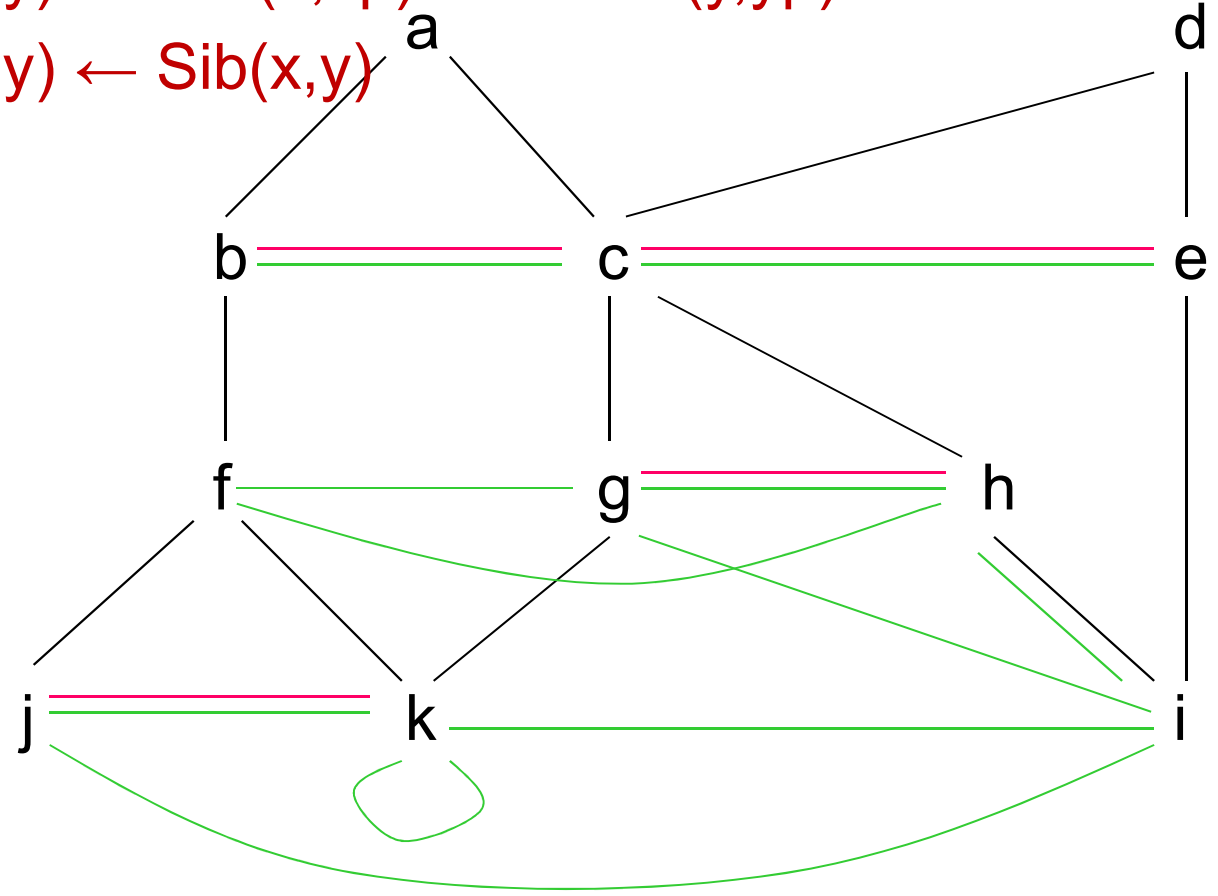
Cousin(x,y) ← Sib(x,y)

Round 1

Round 2

Round 3

Round 4



SQL-99 Recursion

- Datalog recursion has inspired the addition of recursion to the SQL-99 standard.
- Tricky, because SQL allows negation grouping-and-aggregation, which interact with recursion in strange ways.

Form of SQL Recursive Queries

WITH

<stuff that looks like Datalog rules>

<a SQL query about EDB, IDB>

“Datalog rule” =

[RECURSIVE] <name>(<arguments>)

AS <query>

Example: SQL Recursion ---1

- Find Sally's cousins, using SQL like the recursive Datalog example.
- Par(child,parent) is the EDB.

```
WITH Sib(x,y) AS
    SELECT p1.child, p2.child
    FROM Par p1, Par p2
    WHERE p1.parent = p2.parent AND
    p1.child <> p2.child;
```

Like Sib(x,y) ← Par(x,p) AND Par(y,p) AND x <> y

Example: SQL Recursion ---2

Required – Cousin is recursive

Reflects Cousin(x,y) ← Sib(x,y)

Reflects Cousin(x,y) ← Par(x,xp) AND Par(y,yp) AND Cousin(xp,yp)

```
WITH RECURSIVE Cousin(x,y) AS
   (SELECT * FROM Sib)
        UNION
   (SELECT p1.child, p2.child
    FROM Par p1, Par p2, Cousin
    WHERE p1.parent = Cousin.x AND
    p2.parent = Cousin.y)
SELECT y FROM Cousin WHERE x =
  'Sally';
```

Plan to Explain Legal SQL Recursion

1. Define “monotone” recursions.
2. Define a “stratum graph” to represent the connections among subqueries.
3. Define proper SQL recursions in terms of the stratum graph.

Recursion in the SQL-99 standard

- SQL-99 szabvány csak az ún. „monoton” rekurziót támogatja (viszont Datalogban megengedett a negáció és rekurzió együtt, de igen bonyolulttá válik a lekérdezés értelmezése, mint például a rétegzés),
- SQL-99 szabvány lineáris rekurziót enged meg, vagyis az értékadás kifejezésben egyetlen rekurzív relációt használhatunk.

Az „Eljut-feladat” Datalogban

Tankönyv 10.2. fejezet példája (az ELJUT feladat)

- Jaratok(legitarsasag, honnan, hova, koltseg, indulas, erkezes) táblában repülőjáratok adatait tároljuk.

  Mely (x,y) párokra lehet eljutni x városból y városba?

- Datalogban felírva (intuitív bevezetés a Datalogba)

  Eljut(x, y) <- Jaratok(l, x, y, k, i, e)
  Eljut(x, y) <- Eljut(x, z) AND Jaratok(l, z, y, k, i, e)

- Vagy másképp felírva Datalogban (mi a különbség?)

  Eljut(x, y) <- Jaratok(_, x, y, _, _, _)
  Eljut(x, y) <- Eljut(x, z) AND Eljut(z, y)

Az „Eljut feladat” SQL-99 szabványban

- Datalog lineáris és nem-monoton rekurzió átírható:
  Eljut(x, y) <- Jaratok(l, x, y, k, i, e)
  Eljut(x, y) <- Eljut(x, z) AND Jaratok(l, z, y, k, i, e)

- Hova, mely városokba tudunk eljutni Budapestről?

```
WITH RECURSIVE Eljut AS
        (SELECT honnan, hova FROM Jaratok
    UNION
        SELECT Eljut.honnan, Jaratok.hova
        FROM Eljut, Jaratok
        WHERE Eljut.hova = Jaratok.honnan)
SELECT hova FROM Eljut WHERE honnan='Bp';
```

Oracle megoldások

- Az Oracle SQL a WITH RECURSIVE utasítást nem támogatja, ott másképpen oldották meg WITH utasítással (Oracle 11gR2 verziótól)
- with eljut (honnan, hova) as
  (select honnan, hova from jaratok
  union all
  select jaratok.honnan, eljut.hova
  from jaratok, eljut
  where jaratok.hova=eljut.honnan
)
  SEARCH DEPTH FIRST BY honnan SET SORTING
  CYCLE honnan SET is_cycle TO 1 DEFAULT 0
  select distinct honnan, hova from eljut order by honnan;