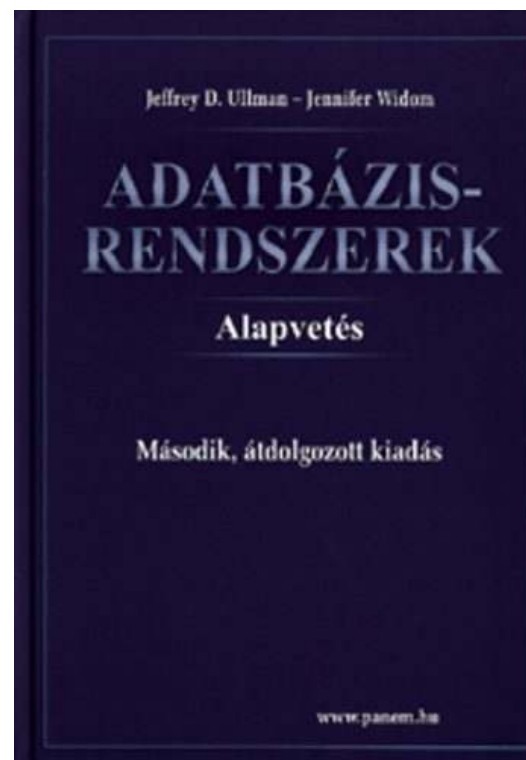


Rekurzív lekérdezések: Az „Eljut-feladat”

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



10.2. Rekurzív lekérdezések a Datalogban valamint az SQL-99 szabványban

- intiutív bevezetés a Datalogba:
- Az „Eljut”-feladat („Jaratok”- táblából honnan-hova lehet eljutni el?)

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)

$\text{Eljut}(x, y) \leftarrow \text{Jaratok}(l, x, y, k, i, e)$

$\text{Eljut}(x, y) \leftarrow \text{Eljut}(x, z) \text{ AND } \text{Jaratok}(l, z, y, k, i, e)$

- Vagy **másképp felírva** Datalogban (mi a különbség?)

$\text{Eljut}(x, y) \leftarrow \text{Jaratok}(_, x, y, _, _, _)$

$\text{Eljut}(x, y) \leftarrow \text{Eljut}(x, z) \text{ AND } \text{Eljut}(z, y)$

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';

Eljut feladat relációs algebrában (nincs mo)

- Az **Oracle SQL** a WITH RECURSIVE utasítást nem támogatja, ott másképpen oldották meg WITH utasítással (Oracle 11gR2 verziótól), illetve Oracle-ben már jóval korábban volt hierarchikus lekérdezés CONNECT BY záradékkal (lásd a gyakorlati anyagban)
- **Relációs algebrában** egy relációs algebrai kifejezéssel nem tudjuk előállítani, csak ciklusba szervezve... nézzük meg PSM-ben:

Rekurzív Eljut feladat PSM-ben ---1

- Az ELJUT feladatot a gyakorlaton oldjuk meg Oracle PL/SQL-ben, itt **csak a vázlata PSM-ben**
- A ciklus során ellenőrizni kell, hogy addig hajtsuk végre a ciklust, amíg növekszik az eredmény (Számláló)
- **DECLARE** RegiSzamlalo Integer;
UjSzamlalo Integer;
- Deklarációs rész után **BEGIN ... END;** között az utasítások, először az eljut táblának kezdeti értéket adunk (a megvalósításnál az INSERT-nél figyelni, hogy ne legyenek ismétlődő sorok: `select distinct delete from eljut; insert into eljut (SELECT distinct honnan, hova FROM jaratok);`)

Rekurzív Eljut feladat PSM-ben ---2

- Szamlalo változóknak adunk kiindulási értéket:
SET RegiSzamlalo = 0;
select count(*) into UjSzamlalo from eljut;
- **A ciklust addig kell végrehajtani**, amíg növekszik az eredmény (Szamlalo) duplikátumokra figyelni!

LOOP

insert into eljut (lásd a köv.oldalon...)

select count(*) into UjSzamlalo from eljut;

EXIT WHEN UjSzamlalo = RegiSzamlalo;

SET RegiSzamlalo = UjSzamlalo;

END LOOP;

Rekurzív Eljut feladat PSM-ben ---3

- Az eljut tábla növelése a ciklusban, figyelni kell a duplikátumokra, csak olyan várospárokat vegyünk az eredményhez, ami még nem volt!

insert into eljut

```
(select distinct eljut.honnan, jaratok.hova  
from eljut, jaratok --- *from (lineáris rekurzió)  
where eljut.hova = jaratok.honnan  
and (eljut.honnan,jaratok.hova)  
NOT IN (select * from eljut));
```

- Megjegyzés: PSM-ben a **nem-lineáris rekurzió** is megengedett: **from eljut e1, eljut e2 ---*from-ban**

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: $\text{Par}(c,p) = p$ is a parent of c .
- Generalized cousins: people with common ancestors one or more generations back:

$\text{Sib}(x,y) \leftarrow \text{Par}(x,p) \text{ AND } \text{Par}(y,p) \text{ AND } x \neq y$

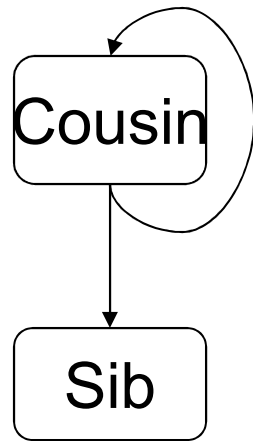
$\text{Cousin}(x,y) \leftarrow \text{Sib}(x,y)$

$\text{Cousin}(x,y) \leftarrow \text{Par}(x,xp) \text{ AND } \text{Par}(y,yp)$
 $\text{AND } \text{Cousin}(xp,yp)$

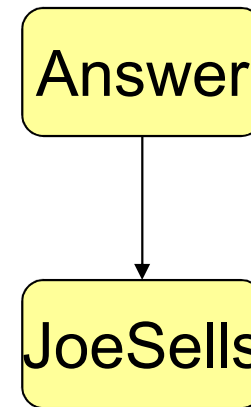
Definition of Recursion

- Megelőzési gráf (IDB relációk közötti élek)
- Form a **dependency graph** whose
- Nodes = IDB predicates.
- Arc $X \rightarrow Y$ 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

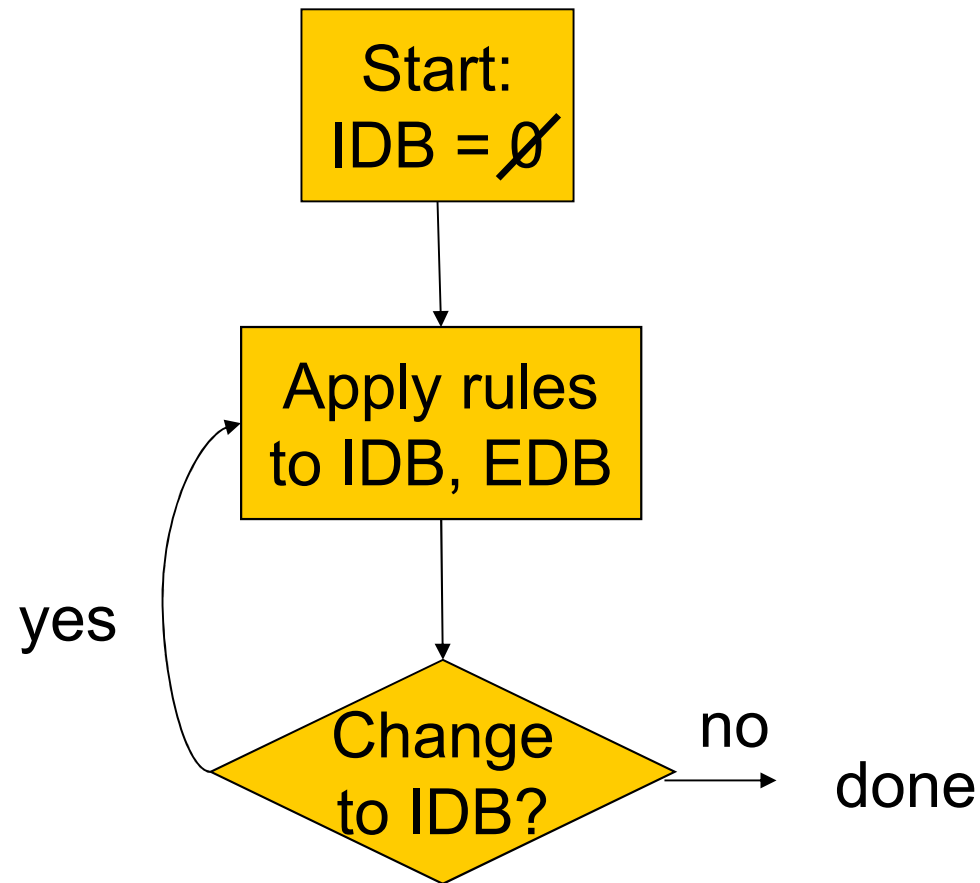


Nonrecursive

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.

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;