8.Előadás Adatbázisok-I.

dr. Hajas Csilla (ELTE IK) http://sila.hajas.elte.hu/

Tankönyv

10.2. Az Eljut-feladat (monoton, lineáris rekurzió) megoldása
(a.) az SQL-ben WITH RECURSION utasítással

Kiegészítések a rekurzióhoz, az Eljut feladat megoldása:

(b.) Oracle megoldások/2: CONNECT BY PRIOR
(c.) Oracle megoldások/3: WITH alkérdés faktorizáció

Következik…

- Relációs algebra korlátai: bizonyos típusú lekérdezéseket nem tudunk relációs algebrával kifejezni…
- Nézzünk meg olyan logikai felépítést, amivel az ilyen rekurzív jellegű lekérdezések könnyen megoldhatók.

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.

Bevezető motiváció: Rekurzív SQL

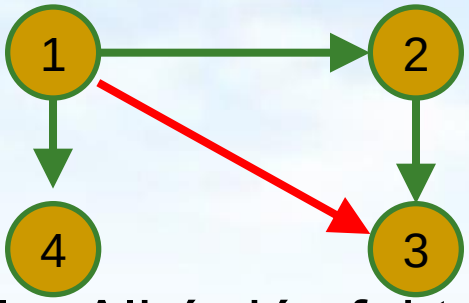
- Alkérdés-faktorizáció WITH utasítással.
- SQL-99 szabványban szerepel.
- Az Oracle 11.2 verziótól kezdve a WITH rekurzív lekérdezésekre is használható (kiegészítésekkel).
- A MS SQL Server 2005-ben implementálták a rekurziót is.
- Rögzítsük egy irányított gráf éleit, és kérdezzük le, honnan hová vezet út!
  - Minden él út is egyben, és ha van él x-ből z-be, és van út z-ből y-ba, akkor van út x-ből y-ba is.

Bevezető motiváció: Rekurzív SQL

- create table el (honnan INT, hova INT);
- insert into el values (1,2);
- insert into el values (2,3),
- insert into el values (1,4);

el:

honnan	hova
1	2
2	3
1	4



ut:

honnan	hova
1	2
2	3
1	4
1	3

- Oracle: Alkérdés faktorizáció
- **WITH** [RECURSION] ut AS
  (select * from el
  UNION ALL -- Oracle-ben
  select el.honnan, ut.hova from el, ut where el.hova=ut.honnan)
  select * from ut;

Bevezető motiváció: Rekurzív SQL

- A WITH ben definiált ut tábla átmeneti munkatábla, az utasítás végén szereplő lekérdezés futása után már nem lehet rá hivatkozni.
- A lekérdezést részekre lehet szétbontani (alkérdés faktorizáció), rekurzív lekérdezések megadhatóak.
- Az él, út példában két szabályt adtunk meg:
  - select * from el
    - Első szabály: ut(x,y)←el(x,y)
  - select el.honnan, ut.hova from el, ut where el.hova=ut.honnan
    - Második szabály: ut(x,y)←el(x,z) and ut(z,y)

Az Eljut-feladat

Tankönyv 10.2. fejezet példája (az ELJUT feladat)

- Jaratok(legitarsasag, honnan, hova, koltseg, indulas, erkezes) táblában repülőjáratok adatait tároljuk.
- A járatok táblát létrehozó script:

  http://people.inf.elte.hu/sila/eduAB/jaratok_tabla.txt
- Mely (x,y) párokra lehet eljutni x városból y városba?
- Ezt egy relációs algebrai kifejezésként nem tudjuk megadni zárt alakban, klasszikus SQL SELECT utasítással sem tudjuk kifejezni, csak azt tudjuk, hogy átszállás nélkül, egy, két, stb… átszállással:

Az Eljut-feladatnak nincs algebrai megoldása

```
select distinct honnan, hova
    from jaratok
union
select j1.honnan, j2.hova
    from jaratok j1, jaratok j2
    where j1.hova=j2.honnan
union
select j1.honnan, j3.hova
    from jaratok j1, jaratok j2, jaratok j3
    where j1.hova=j2.honnan
    and j2.hova=j3.honnan
--- union stb... Ezt így nem lehet felírni…
```

Az Eljut-feladat Datalogban

Tankönyv 10.2. fejezet példája (az ELJUT feladat)

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

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

- Datalogban felírva (lineáris rekurzió)

  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, _, _, _) --- anonimus változók
  Eljut(x, y) <- Eljut(x, z) AND Eljut(z, y) --- nem lineáris rek.

Az Eljut feladat SQL-99 szabványban

- Datalog LINEÁRIS, 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';
```

SQL-99 szabvány: Rekurzív lekérdezés

- A WITH utasítás több ideiglenes relációra vonatkozó definíciója:

WITH [RECURSIVE] R_1 AS <R_1 definíciója>
[RECURSIVE] R_2 AS <R_2 definíciója>
...
[RECURSIVE] R_n AS <R_n definíciója>
< R_1,R_2,...,R_n relációkat tartalmazó lekérdezés>

Másik példa: Rekurzív Datalog

- A testvérek (féltestvérek) gyerekei első unokatestvérek, az első unokatestvérek gyerekei másod-unokatestvérek, és így tovább. Hívjuk egyszerűen unokatestvéreknek, akik valamilyen szinten unokatestvérek. A rokonok azok, akik közös ősnek leszármazottjai.
- Milyen Datalog program írja ezt le?

testvér(x,y) ←gyerek(x,z),gyerek(y,z),x ≠y

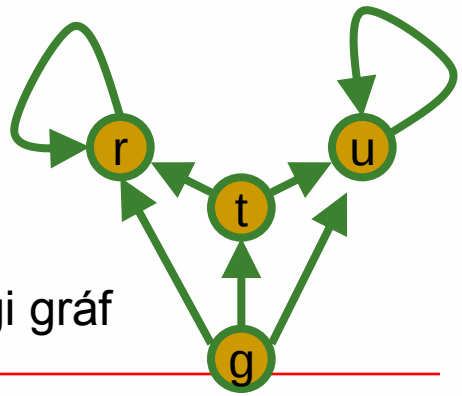
unokatestvér(x,y) ←gyerek(x,z),gyerek(y,v),testvér(z,v)

unokatestvér(x,y) ←gyerek(x,z),gyerek(y,v),unokatestvér(z,v)

rokon(x,y) ←testvér(x,y)

rokon(x,y) ←rokon(x,z),gyerek(y,z)

rokon(x,y) ←rokon(z,y),gyerek(x,z)



Függőségi gráf

Másik példa: Rekurzív Datalog

- Mik a Datalog egyenletek?

testvér(x,y) ←gyerek(x,z),gyerek(y,z),x ≠y
unokatestvér(x,y) ←gyerek(x,z),gyerek(y,v),testvér(z,v)
unokatestvér(x,y) ←gyerek(x,z),gyerek(y,v),unokatestvér(z,v)
rokon(x,y) ←testvér(x,y)
rokon(x,y) ←rokon(x,z),gyerek(y,z)
rokon(x,y) ←rokon(z,y),gyerek(x,z)

- A megfelelő relációk legyenek T,U,R,G a testvér, unokatestvér, rokon, illetve gyerek esetén.
- $T(x,y)=\Pi_{x,y}(\sigma_{x\neq y}(G(x,z)|><|G(y,z)))$

$U(x,y)=\Pi_{x,y}(G(x,z)|><|G(y,v)|><|T(z,v))\cup$
$\Pi_{x,y}(G(x,z)|><|G(y,v)|><|U(z,v))$

$R(x,y)=T(x,y)\cup$
$\Pi_{x,y}(R(x,z)|><|G(y,z))\cup$
$\Pi_{x,y}(R(z,y)|><|G(x,z))$

Másik példa: Rekurzív Datalog

- Milyen SQL utasítás írja le az IDB táblákat?
- $T(x,y)=\Pi_{x,y}(\sigma_{x\neq y}(G(x,z)|><|G(y,z)))$

  $U(x,y)=\Pi_{x,y}(G(x,z)|><|G(y,v)|><|T(z,v))\cup$
  $\Pi_{x,y}(G(x,z)|><|G(y,v)|><|U(z,v))$

  $R(x,y)=T(x,y)\cup$
  $\Pi_{x,y}(R(x,z)|><|G(y,z))\cup$
  $\Pi_{x,y}(R(z,y)|><|G(x,z))$
- Tegyük fel, hogy a séma: G(u,w).
- with

```
T as (select G1.u x, G2.w y from G G1,G G2
                                  where G1.w=G2.u and G1.u<>G2.u),
U as (select G1.u x, G2.u y from G G1, G G2, T
                                  where T.x=G1.w and T.y=G2.u) union all
     (select G1.u x, G2.u y from G G1, G G2, U
                                  where U.x=G1.w and U.y=G2.u),
R as (select * from T) union all
     (select R.x x,G.u y from R,G where R.y=G.w) union all
     (select G.u x, R.y y from R,G where R.x=G.w)
(select T.x, T.y, 'T' from T union all
 select U.x, U.y, 'U' from U union all
 select R.x, R.y, 'R' from R);
```

Rekurzív lekérdezések

- Datalog rekurzió segít megérteni az SQL-99 szabványban bevezetett rekurzív lekérdezések WITH RECURSIVE záradékát.
- A BSc-n csak MONOTON rekurziót vesszük, vagyis nem használjuk nem-monoton különbség műveletet, nincs csoportosítás-aggregálás (ugyanis az olyan lekérdezések, amelyek nem-monotonok, megengedik a negációt és aggregálást az olyan különös hatással van a rekurzióra, ezt csak MSc kurzusokon vesszük).
- Gyakorlaton a rekurzív Eljut-feladatnak az Oracle CONNECT BY záradékkal való gépes-megoldásait is megnézzük (ezt csak a gyakorlaton próbáljuk ki).

Oracle megoldások: with utasítással

- Az Oracle SQL a WITH RECURSIVE utasítást (UNION) nem támogatja, ott másképpen oldották meg WITH utasítással (Oracle 11gR2 verziótól használható)

```
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;
```

Oracle megoldások: connect by

- SELECT DISTINCT hova FROM jaratok
  WHERE HOVA <> 'DAL'
  START WITH honnan = 'DAL'
  CONNECT BY NOCYCLE PRIOR hova = honnan;

- SELECT LPAD(' ', 4*level) || honnan, hova,
  level-1 Atszallasok,
  sys_connect_by_path(honnan||'->'||hova, '/'),
  connect_by_isleaf, connect_by_iscycle
  FROM jaratok
  START WITH honnan = 'SF'
  CONNECT BY NOCYCLE PRIOR hova = honnan;