

Logikai lekérdező nyelv: Datalog

Rekurzió a Datalogban és az SQL-ben

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



5.3. Logika a relációkhoz: Datalog

5.4. Relációs algebra és
nem-rekurzív biztonságos Datalog
kifejező erejének összehasonlítása

10.2. Az Eljut-feladat megoldása: (monoton és lineáris)
rekurzió a Datalogban és az SQL-ben: with recursion

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.
- Ez a nyelv: a Datalog (Tankönyv 5.3-5.4)
- Rekurzió (Tankönyv 10.2)

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.

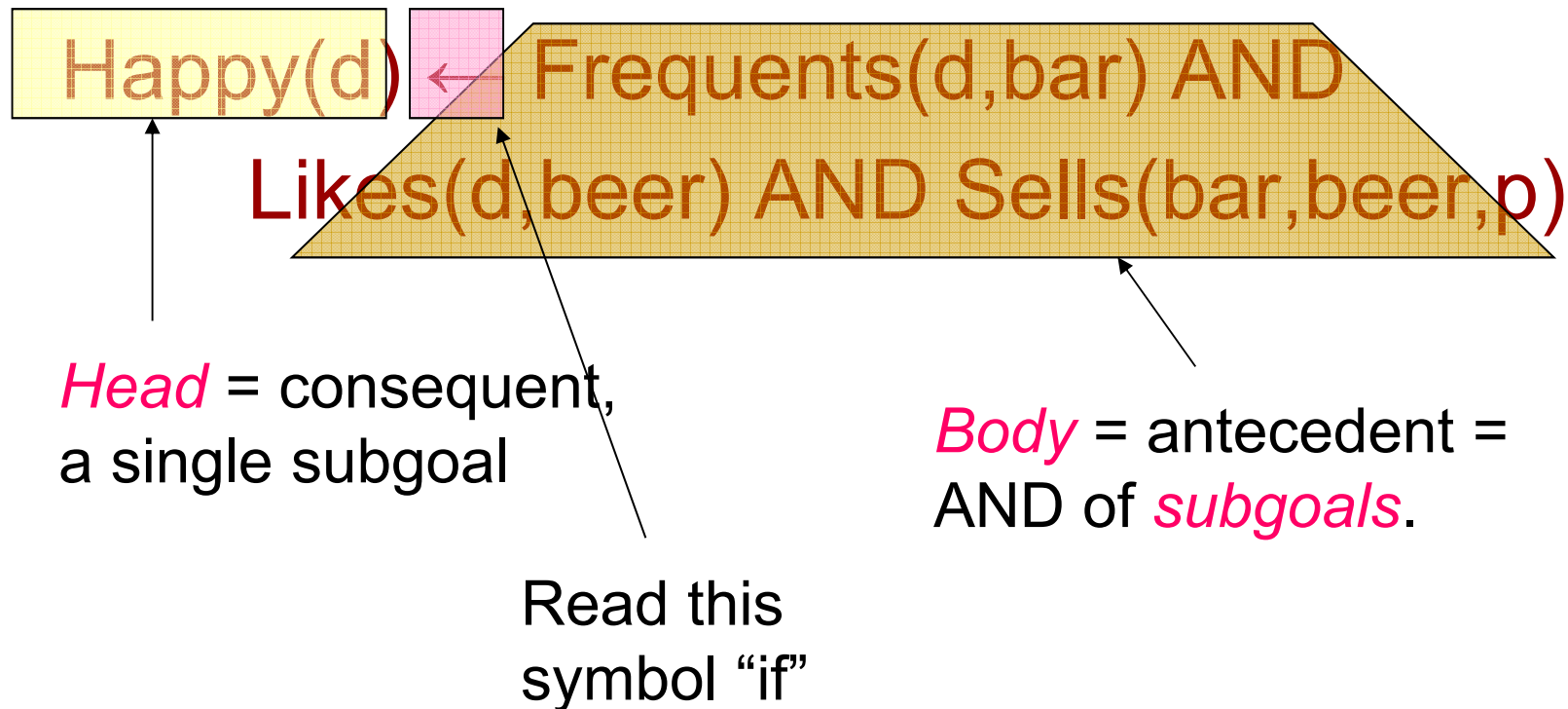
Logika, mint lekérdező nyelv

- **Abstract Query Languages:**
 - Relational Algebra (procedural → optimization)
 - Logical QL: Datalog, Rel.Calculus (declarative)
- **Datalog** = 'Data'- Database, 'log'- logic, Prolog
- **Logikai alapú nyelv, szabály alapú nyelv.** If-then logical rules have been used in many systems.
- **Nonrecursive rules** are equivalent to the core relational algebra.
- Recursive rules extend relational algebra and appear in SQL-99.

Datalog szabályok és lekérdezések

- Datalog: logikai alapú, szabály-alapú nyelv
szabály: IF feltétel THEN eredmény, ahol a feltétel relációkkal legyen megadható és az eredmény az output tábla sorait eredményezze
- Our first example of a rule uses the relations
Frequents(drinker, bar),
Likes(drinker, beer),
Sells(bar, beer, price).
- The rule is a query asking for “happy” drinkers
--- those that frequent a bar that serves a beer
that they like.

Datalog szabályok felépítése

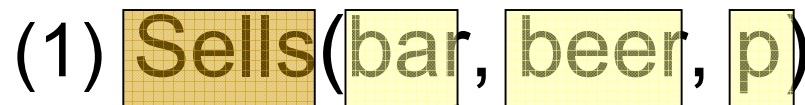


Atomi formulák-1: Relációs formulák

- An *atom* is a *predicate*, or relation name with variables or constants as arguments.
- The head is an atom; the body is the AND of one or more atoms.
- **Convention:** Predicates begin with a capital, variables begin with lower-case.

Példa relációs atomi formulára

(1) **Sells**(bar, beer, p)



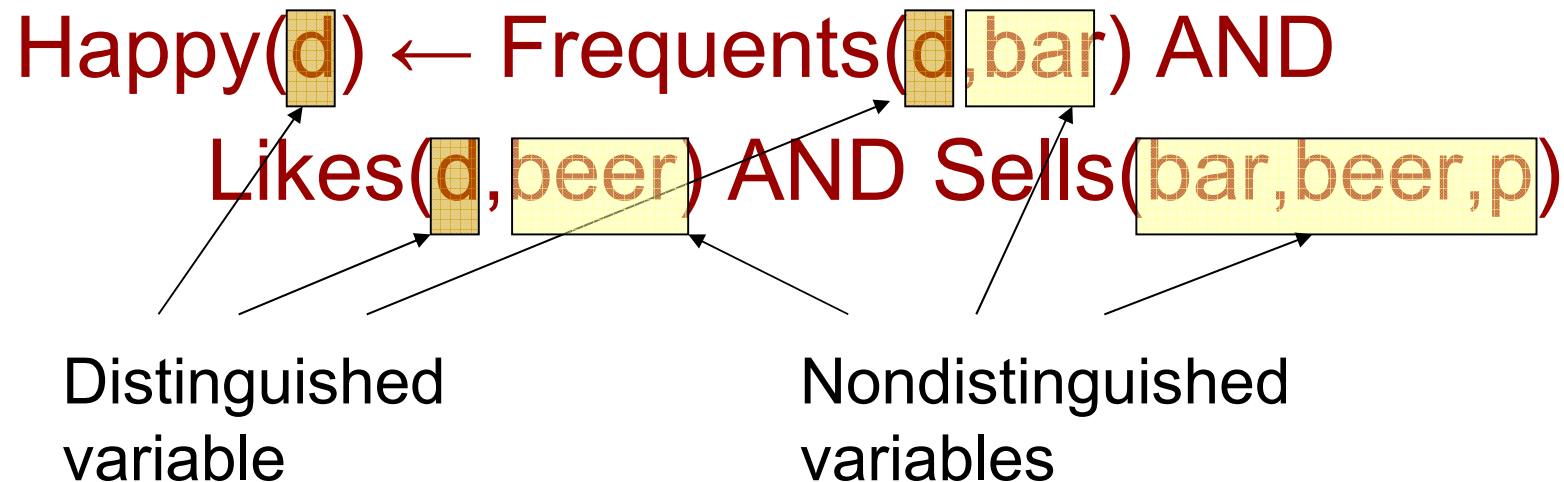
The predicate
= name of a
relation

Arguments are
variables (or constants).

(2) Proba(x, x, y, 5, 'alma')



Example: Interpretation



Interpretation: drinker d is happy if there exist a bar, a beer, and a price p such that d frequents the bar, likes the beer, and the bar sells the beer at price p .

Datalog szabályok kiértékelése ---1

- **Approach 1:** consider all combinations of values of the variables.
- If all subgoals are true, then evaluate the head.
- The resulting head is a tuple in the result.

Example: Rule Evaluation

Happy(d) \leftarrow Frequent(d,bar) AND
Likes(d,beer) AND Sells(bar,beer,p)

FOR (each d, bar, beer, p)

IF (Frequent(d,bar), Likes(d,beer), and
Sells(bar,beer,p) are all true)

add Happy(d) to the result

- **Note:** set semantics so add only once.
- Set semantics vice versa bag semantics

Datalog szabályok kiértékelése ---2

- **Approach 2:** For each subgoal, consider all tuples that make the subgoal true.
- If a selection of tuples define a single value for each variable, then add the head to the result.

Happy(d) \leftarrow Frequents(d,bar) AND

Likes(d,beer) AND Sells(bar,beer,p)

FOR (each f in Frequents, i in Likes,
and s in Sells)

IF (f[1]=i[1] and f[2]=s[1] and i[2]=s[2])
add Happy(f[1]) to the result

Milyen problémák merülnek fel? (erre visszatérünk a biztonságosságnál)

- Relations are finite sets.
- We want rule evaluations to be finite and lead to finite results.
- “Unsafe” rules like $P(x) \leftarrow Q(y)$ have infinite results, even if Q is finite.
- Even $P(x) \leftarrow Q(x)$ requires examining an infinity of x -values.

Atomi formulák-2: Aritmetikai formula

- In addition to relations as predicates, a predicate for a subgoal of the body can be an arithmetic comparison.
- We write arithmetic subgoals in the usual way, e.g., $x < y$.

Példa: Aritmetikai részcélok

- A beer is “cheap” if there are at least two bars that sell it for under \$2.

Cheap(beer) \leftarrow Sells(bar1,beer,p1) AND
Sells(bar2,beer,p2) AND $p1 < 2.00$
AND $p2 < 2.00$ AND $bar1 \neq bar2$

Negált részcélok

- NOT in front of a subgoal negates its meaning.
- **Example:** Think of $\text{Arc}(a,b)$ as arcs in a graph.
 - $S(x,y)$ says the graph is not transitive from x to y ; i.e., there is a path of length 2 from x to y , but no arc from x to y .

$S(x,y) \leftarrow \text{Arc}(x,z) \text{ AND } \text{Arc}(z,y)$
 $\text{AND NOT } \text{Arc}(x,y)$

Biztonságossági elvárás

- A szabályok kiértékelhetőek legyenek, ehhez:
- A szabályban szereplő minden változónak elő kell fordulnia a törzsben nem-negált relációs atomban

Biztonságos szabályok

- A rule is *safe* if:
 1. Each distinguished variable,
 2. Each variable in an arithmetic subgoal, and
 3. Each variable in a negated subgoal,
also appears in a nonnegated,
relational subgoal, amivel az x korlátozott:
 - $\text{pred}(x, y, \dots)$ argumentuma (értéke a táblából)
 - vagy $x=c$ (konstans)
 - vagy $x=y$ (ahol y korlátozott)
- Safe rules prevent infinite results.

Példa: Nem biztonságos szabályokra

- Each of the following is unsafe and not allowed:
 1. $S(x) \leftarrow R(y)$
 2. $S(x) \leftarrow R(y) \text{ AND } x < y$
 3. $S(x) \leftarrow R(y) \text{ AND NOT } R(x)$
- In each case, an infinity of x 's can satisfy the rule, even if R is a finite relation.

Datalog programok

- **Datalog program** = collection of rules.
- In a program, predicates can be either
 1. EDB relációk = **Extensional Database** = stored table (csak a törzsben szereplő relációk)
 2. IDB relációk = **Intensional Database** = relation defined by rules (szerepel fej-ben)
- Never both! No EDB in heads.

Datalog programok kiértékelése

- As long as there is no recursion, we can pick an order to evaluate the IDB predicates, so that all the predicates in the body of its rules have already been evaluated.
- If an IDB predicate has more than one rule, each rule contributes tuples to its relation.

Példa: Datalog program

- Using EDB **Sells(bar, beer, price)** and **Beers(name, manf)**, find the manufacturers of beers Joe doesn't sell.

JoeSells(b) ← Sells('Joe's Bar', b, p)

Answer(m) ← Beers(b,m)

AND NOT JoeSells(b)

Példa: Kiértékelése

- Step 1: Examine all **Sells** tuples with first component 'Joe's Bar'.
 - Add the second component to **JoeSells**.
- Step 2: Examine all **Beers** tuples (b,m).
 - If b is not in **JoeSells**, add m to Answer.

Datalog kifejező ereje

- 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.

Relációs algebrai kifejezések átírása

- Mi a leggyakrabban előforduló típus, amiből építkeznek? $\Pi_{\text{Lista}}(\sigma_{\text{Felt}}(\mathbf{R} \bowtie \mathbf{S} \bowtie \dots))$

Ezt a komponenst támogatja legerősebben az SQL is: **SELECT lista**

FROM táblák összekapcsolása

WHERE felt

Ez felel meg egy Datalog szabálynak.

- **Halmazműveletek:** kezdjük ezzel, hogyan lehet a metszetet és különbséget Datalog szabállyal, és az egyesítést Datalog programmal kifejezni.

Relációs algebra és Datalog ---1

Rel.algebrai műveletek hogyan néznek ki Datalogban?

Halmazműveletek: T.f.h $R(x_1, \dots, x_n)$, $S(x_1, \dots, x_n)$

predikátumokhoz tartozó reláció $R(A_1, \dots, A_n)$, $S(A_1, \dots, A_n)$

➤ $R \cap S$ metszetnek megfelelő szabály:

$Válasz(x_1, \dots, x_n) \leftarrow R(x_1, \dots, x_n) \text{ AND } S(x_1, \dots, x_n)$

➤ $R - S$ különbségnek megfelelő szabály:

$Válasz(x_1, \dots, x_n) \leftarrow R(x_1, \dots, x_n) \text{ AND NOT } S(x_1, \dots, x_n)$

➤ $R \cup S$ unió műveletet egyetlen szabállyal nem tudom felírni, mert a törzsben csak AND lehet, OR nem. Ehhez több szabályból álló Datalog program kell:

$Válasz(x_1, \dots, x_n) \leftarrow R(x_1, \dots, x_n)$

$Válasz(x_1, \dots, x_n) \leftarrow S(x_1, \dots, x_n)$

Relációs algebra és Datalog ---2

Kiválasztás:

- $\sigma_{x_i \theta x_j}(R)$ kifejezésnek megfelelő szabály :
 $Válasz(x_1, \dots, x_n) \leftarrow R(x_1, \dots, x_n) \text{ AND } x_i \theta x_j$
- $\sigma_{x_i \theta c}(E1)$ kifejezésnek megfelelő szabály:
 $Válasz(x_1, \dots, x_n) \leftarrow R(x_1, \dots, x_n) \text{ AND } x_i \theta c$

Vetítés:

- $\Pi_{A_{i1}, \dots, A_{ik}}(R)$ kifejezésnek megfelelő szabály:
 $Válasz(x_{i1}, \dots, x_{ik}) \leftarrow R(x_1, \dots, x_n)$

Megjegyzés: név nélküli anónymus változók, amelyek csak egyszer szerepelnek és mindegy a nevük azt aláhúzás helyettesítheti. Például:

$HosszúFilm(c, é) \leftarrow Film(c, é, h, _, _, _) \text{ AND } h \geq 100$

Relációs algebra és Datalog ---3

Természetes összekapcsolás: Tegyük fel, hogy

$R(A_1, \dots, A_n, C_1, \dots, C_k)$ és $S(B_1, \dots, B_m, C_1, \dots, C_k)$

- $R \bowtie S$ kifejezésnek megfelelő szabály:

$Válasz(x_1, \dots, x_n, y_1, \dots, y_m, z_1, \dots, z_k) \leftarrow$

$\leftarrow R(x_1, \dots, x_n, z_1, \dots, z_k) \text{ AND } S(y_1, \dots, y_m, z_1, \dots, z_k)$

- A felírt szabályok biztonságosak.
- Minden Q relációs algebrai kifejezéshez van nem rekurzív, biztonságos, negációt is tartalmazó Datalog program, amelyben egy kitüntetett IDB predikátumhoz tartozó kifejezés ekvivalens a Q lekérdezéssel.
- A nem rekurzív, biztonságos, negációt is tartalmazó Datalog kifejezőerő tekintetében EKVIVALENS a relációs algebrával.

Kérdés/Válasz --- 1

- **Kérdés/Válasz?**
- **Gyakorlás:** Tk. Termékek és Csatahajók feladatainak megoldásában szereplő relációs algebrai kifejezések átírása Datalog programokká és SQL SELECT-re.
- **Házi feladat:** Oracle Példatár 3.fejezet feladatai, összekapcsolások és alkérdések használatával:

<http://people.inf.elte.hu/sila/eduAB/Feladatok.pdf>

Következik: Rekurzió Datalogban és SQL SELECT-ben
WITH RECURSION záradék

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, \dots, R_n relációkat tartalmazó lekérdezés>

Egy másik példa rekurzióra

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

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

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

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

- Feladat: Datalog program átírása SQL-99 szabvány SELECT utasításra WITH RECURSIVE záradékkal:

Példa: átírás rekurzív lekérdezésre ---1

- Find Sally's cousins, using SQL like the recursive Datalog example. **Par(child,parent)** is the EDB.
- Megoldás (két oldalon fért el)

WITH Sib(x,y) AS --- ez itt az (1)

SELECT p1.child x, p2.child y

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

WITH RECURSIVE Cousin(x,y) AS --- itt jön a (2) ...
--- **SELECT** ...köv.oldalon ...

SELECT y FROM Cousin WHERE x = 'Sally' ;

Példa: átírás rekurzív lekérdezésre ---2

Required – Cousin
is recursive

WITH RECURSIVE Cousin(x,y) AS

(SELECT * FROM Sib)

UNION

(SELECT p1.child x, p2.child y
FROM Par p1, Par p2, Cousin
WHERE p1.parent = Cousin.x AND
p2.parent = Cousin.y)

Reflects Cousin(x,y) ←
Sib(x,y)

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

SELECT y FROM Cousin WHERE x = 'Sally' ;

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 gépes-megoldásait** is megnézzük, ami nem lesz majd vizsgán, 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

- Oracle sokkal korábban bevezette a **hierarchikus lekérdezéseket**, és ezt bővítette ki a rekurzióra is:

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

- Oracle-ben további hasznos függvények is használhatók:

```
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;
```

Kérdés/Válasz

- Köszönöm a figyelmet! Kérdés/Válasz?
- **Vizsgára csak** az Eljut feladatot kell ismerni és az SQL-99 szabvány WITH RECURSION utasítását, de csak papíron. Feladat: Datalog program átírása SELECT-re WITH RECURSION záradékkal (SQL-99)
- **Házi feladat: Gyakorlás** Oracle Példatár 3.fejezete tartalmaz hierarchikus lekérdezésekre feladatokat:
<http://people.inf.elte.hu/sila/eduAB/Feladatok.pdf>
- Oracle gépes-megoldások nem lesznek a vizsgán, csak a gyakorlaton próbáljuk ki: az Eljut-feladathoz a Jaratok táblát létrehozó script: [create jaratok tabla.txt](#)