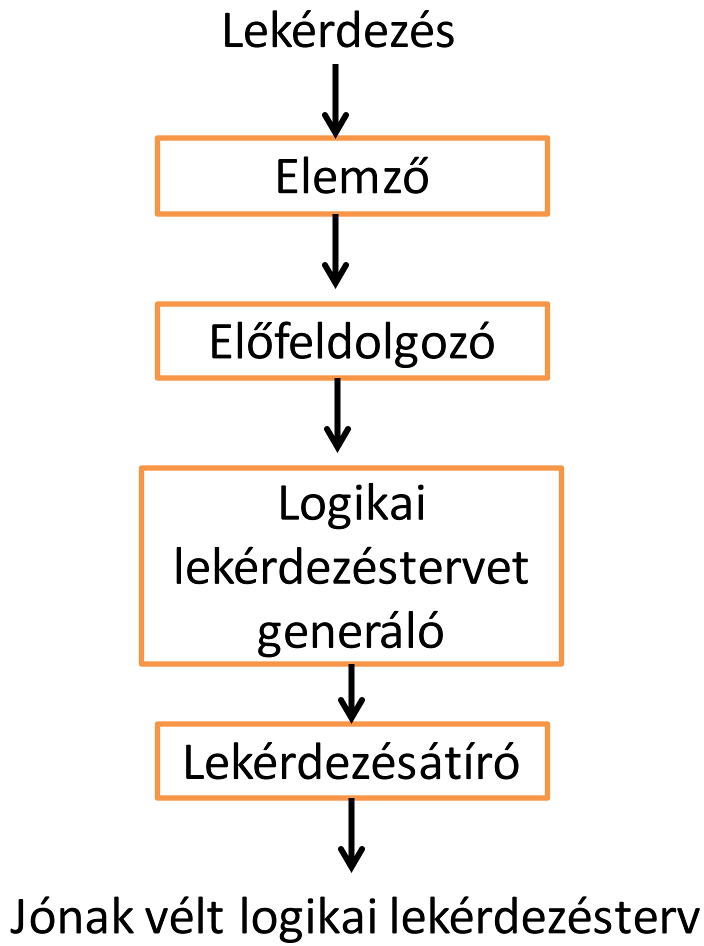
Lekérdezésfordító

Adatbázisok 2.

Elemzés


Lekérdezés
Elemző
Előfeldolgozó
Logikai lekérdezéstervet generáló
Lekérdezésátíró
Jónak vélt logikai lekérdezésterv

Leegyszerűsített nyelvtan I.

```
<Lekérdezés> ::= <SFW>
<Lekérdezés> ::= (<Lekérdezés>)

<SFW> ::=   SELECT <SelLista>
            FROM <FromLista>
            WHERE <Feltétel>

<SelLista> ::= <Attribútum>, <SelLista>
<SelLista> ::= <Attribútum>
```

Leegyszerűsített nyelvtan II.

```
<FromLista> ::= <Reláció>, <FromLista>
<FromLista> ::= <Reláció>

<Feltétel> ::= <Feltétel> AND <Feltétel>
<Feltétel> ::= <Sor> IN <Lekérdezés>
<Feltétel> ::= <Attribútum> = <Attribútum>
<Feltétel> ::= <Attribútum> = <Konstans>

<Sor> ::= <Attribútum>
```

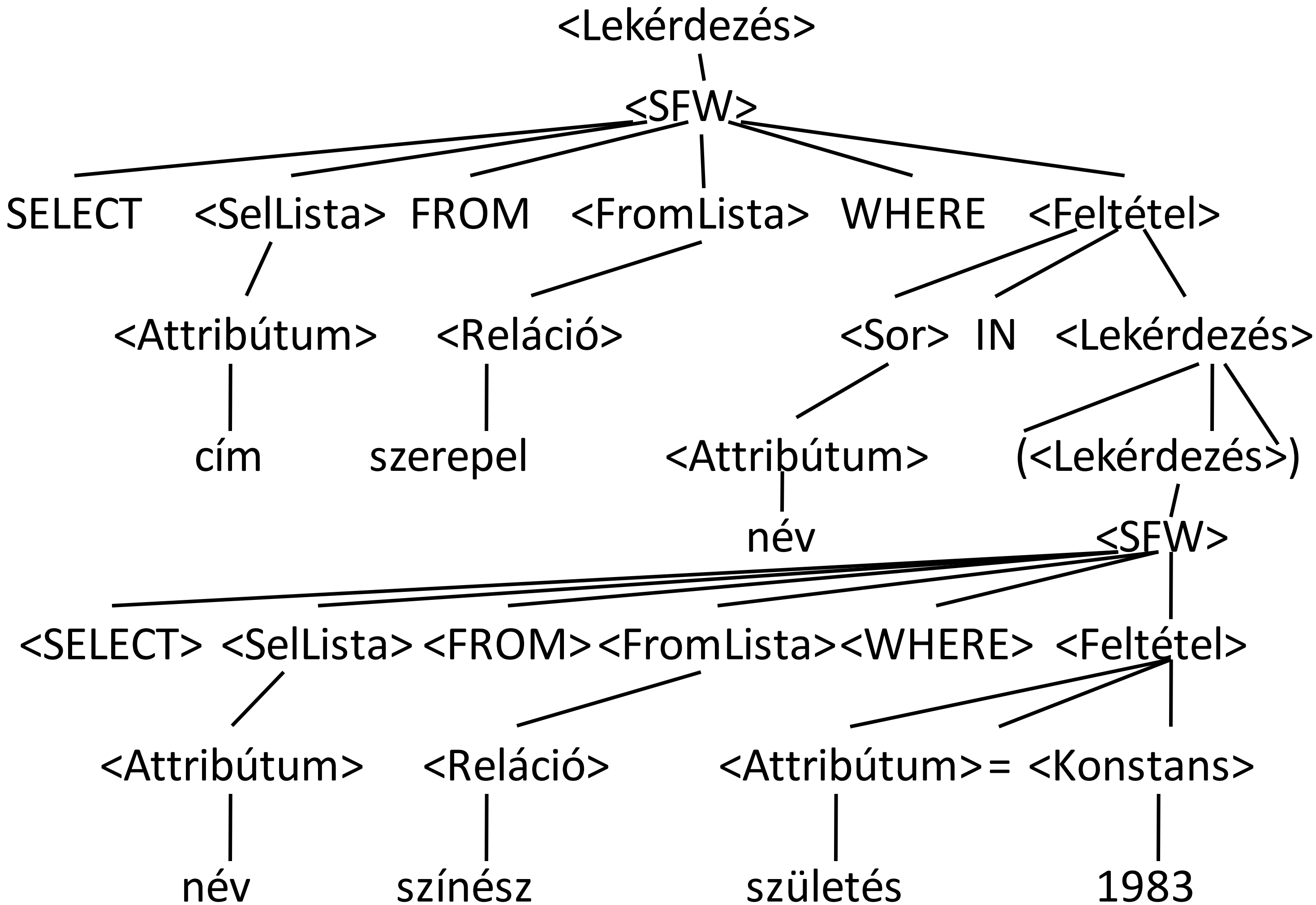
Elemzőfa

- Az elemzőfa (parse tree) csomópontjai az alábbiak lehetnek:
  - atomok: azaz attribútumok, relációk nevei, konstansok, zárójelek, operátorok (+, < stb.), lexikai elemek (SELECT).
  - Szintaktikus kategóriák.

Példa I.

- Szerepel (cím, év, név)
  Színész (név, cím, fizetés, születés)

```
SELECT cím
FROM szerepel
WHERE név IN (SELECT név
              FROM színész
              WHERE születés = 1983);
```

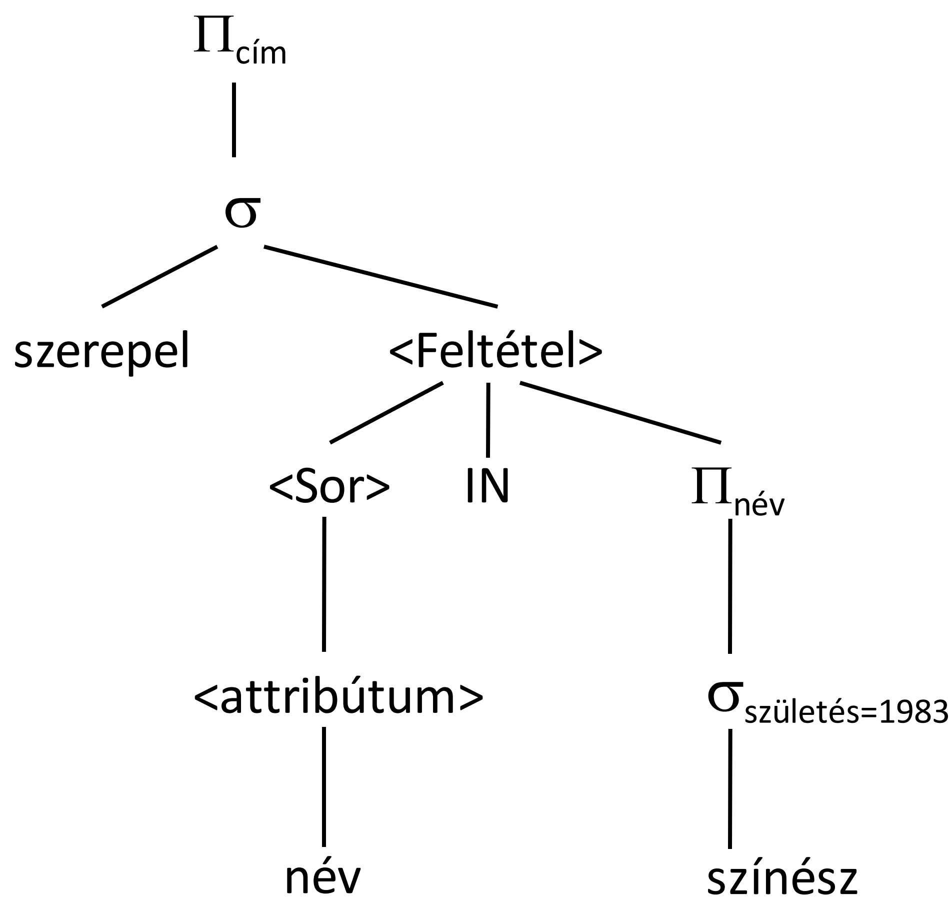

<Lekérdezés>
<SFW>
SELECT
<SelLista>
FROM
<FromLista>
WHERE
<Feltétel>
<Attribútum>
<Reláció>
<Sor>
IN
<Lekérdezés>
cím
szerepel
<Attribútum>
(<Lekérdezés>)
név
<SFW>
<SELECT>
<SelLista>
<FROM>
<FromLista>
<WHERE>
<Feltétel>
<Attribútum>
<Reláció>
<Attribútum>
=
<Konstans>
név
színész
születés
1983

Előfeldolgozó

- Ha a lekérdezésben nézettáblát használunk, akkor ezt a megfelelő elemzőfával helyettesíti.
- Szemantikus ellenőrzés (semantic checking):
  - relációk használatának ellenőrzése,
  - attribútumnevek ellenőrzése és feloldása,
  - típusellenőrzés.

Elemzőfákból logikai lekérdezésterv

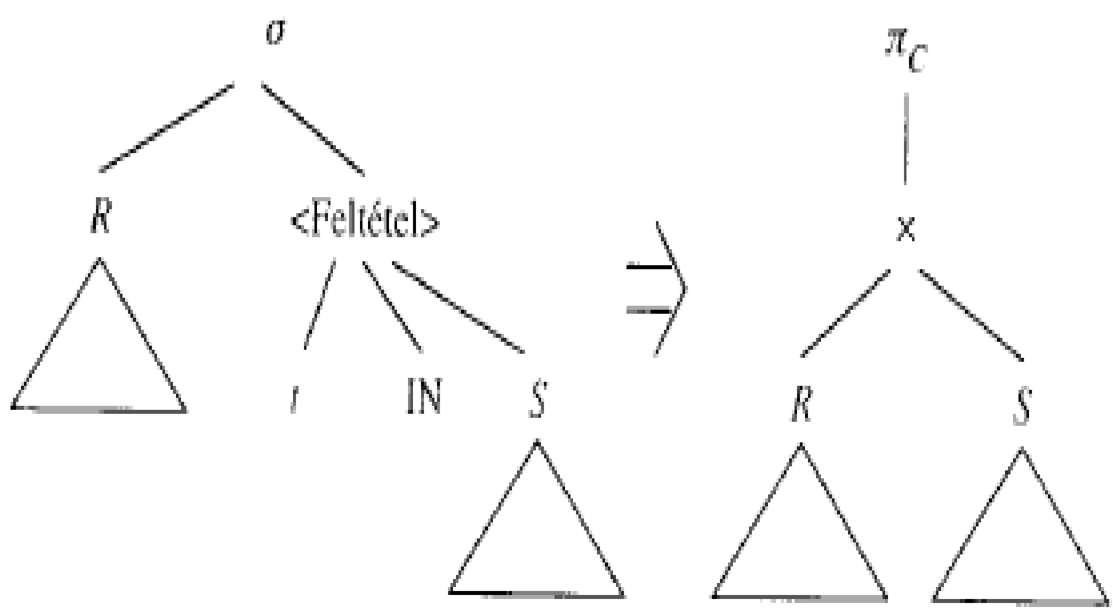
- Ha a lekérdezés nem tartalmaz alkérdést, az átírás könnyen megy.
- A <FromLista> relációinak vesszük a Descartes-szorzatát.
- A <SelLista> helyett Π_L szerepel, ahol L a <SelLista> attribútumlistája.
- A <Feltétel>-t σ_C-vel helyettesítjük, itt C értelemszerűen megadható.


$\Pi_{\text{cím}}$
σ
szerepel
<Feltétel>
<Sor>
IN
$\Pi_{\text{név}}$
<attribútum>
$\sigma_{\text{születés=1983}}$
név
színész

Kétargumentumú kiválasztás

- σ paraméterek nélkül reprezentálja a „kétargumentumú kiválasztás”-t
- A kétargumentumú szelekció az elemzőfa szintaktikus kategóriái és a relációs algebrai operátorok közt képvisel átmenetet.
- Az ***alkérdések átírásánál*** használatos.
- A baloldali gyermeke a reláció, amire a kiválasztás vonatkozik.
- A jobboldali gyermek a kiválasztó feltételt jeleníti meg.
- Mindkét argumentum ábrázolható, mint elemzőfa, mint kifejezésfa és mint a kettő keveréke.
- Szabályaink vannak, melyekkel a kétargumentumú kiválasztást "normális" kiválasztással és egy másik relációs algebrai operátorral helyettesítjük.

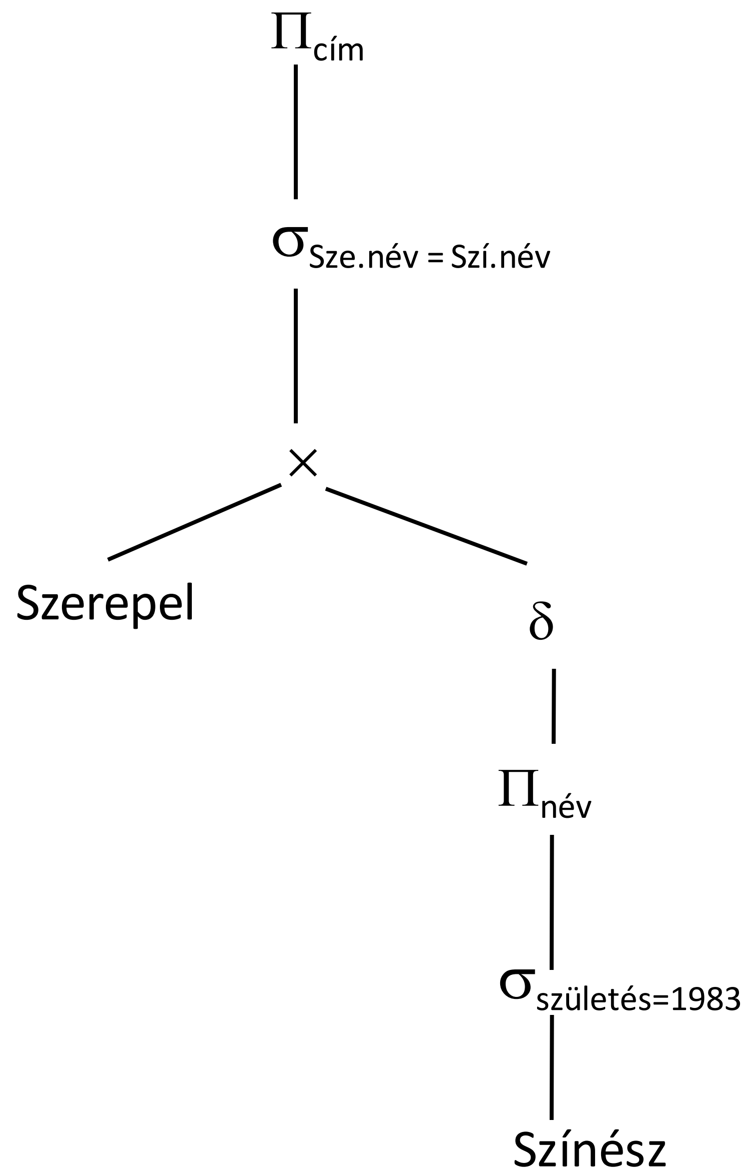
Kétargumentumú kiválasztás



7.15. ábra. IN-*t tartalmazó feltétellel rendelkező kétargumentumú kiválasztást kezelő szabály*

t IN S

- Tegyük fel, hogy a baloldalon R reláció szerepel (elő argumentum), a feltétel pedig t IN S alakú. (második argumentum).
- Ekkor:
  - a jobboldali feltételt az S-et létrehozó kifejezéssel helyettesítjük. Ha S-ben lehetnek ismétlődések, akkor ezeket ki kell küszöbölnünk.
  - R-nek és az ismétlődések kiküszöbölése után kapott relációs kifejezésnek vesszük a Descartes-szorzatát.
  - Efölött a kétargumentumú kiválasztást σ_C-vel helyettesítjük, ahol C-ben a t sor komponenseit tesszük egyenlővé S megfelelő attribútumaival.


$\Pi_{\text{cím}}$
$\sigma_{\text{Sze.név = Szí.név}}$
$\times$
Szerepel
δ
$\Pi_{\text{név}}$
$\sigma_{\text{születés=1983}}$
Színész

Részleg

Id	Név	Város
10	PR	Eger
20	HR	Szeged

Dolgozó

Id	Név	Kereset	Részleg_id
101	Elek	150000	10
102	Dezső	170000	20
103	Ilona	230000	20

A

```
SELECT név, város
FROM részleg
WHERE id IN (SELECT részleg_id
        FROM dolgozó
        WHERE kereset > 150000);
```

```
SELECT r.név, r.város
FROM dolgozó d, részleg r
WHERE r.id = d.részleg_id AND
       kereset > 150000;
```

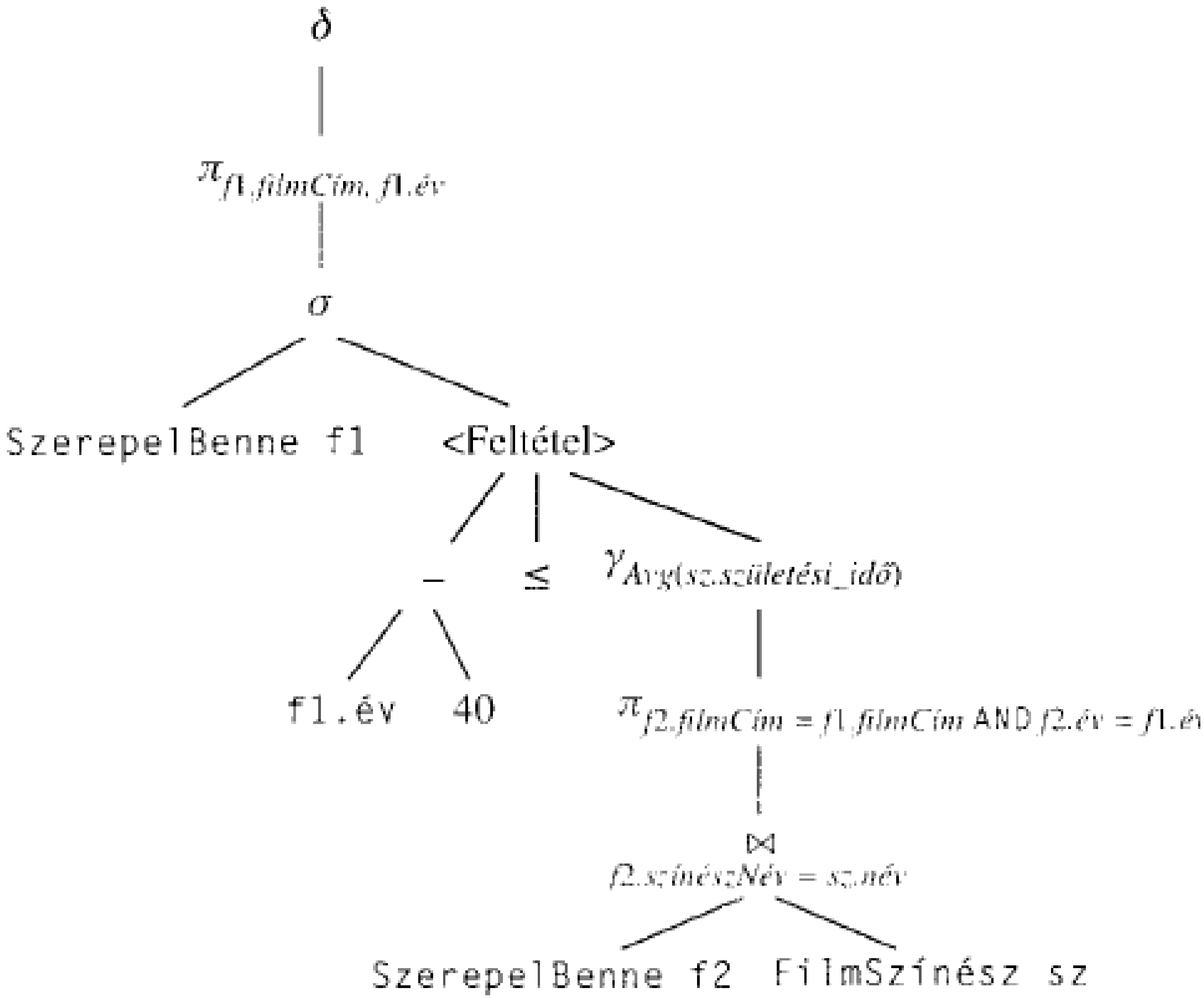
lekérdezések nem ugyanazt az eredményt adják a példa táblákra, emiatt szükséges az ismétlődések kiküszöbölése.

Érdekes eset

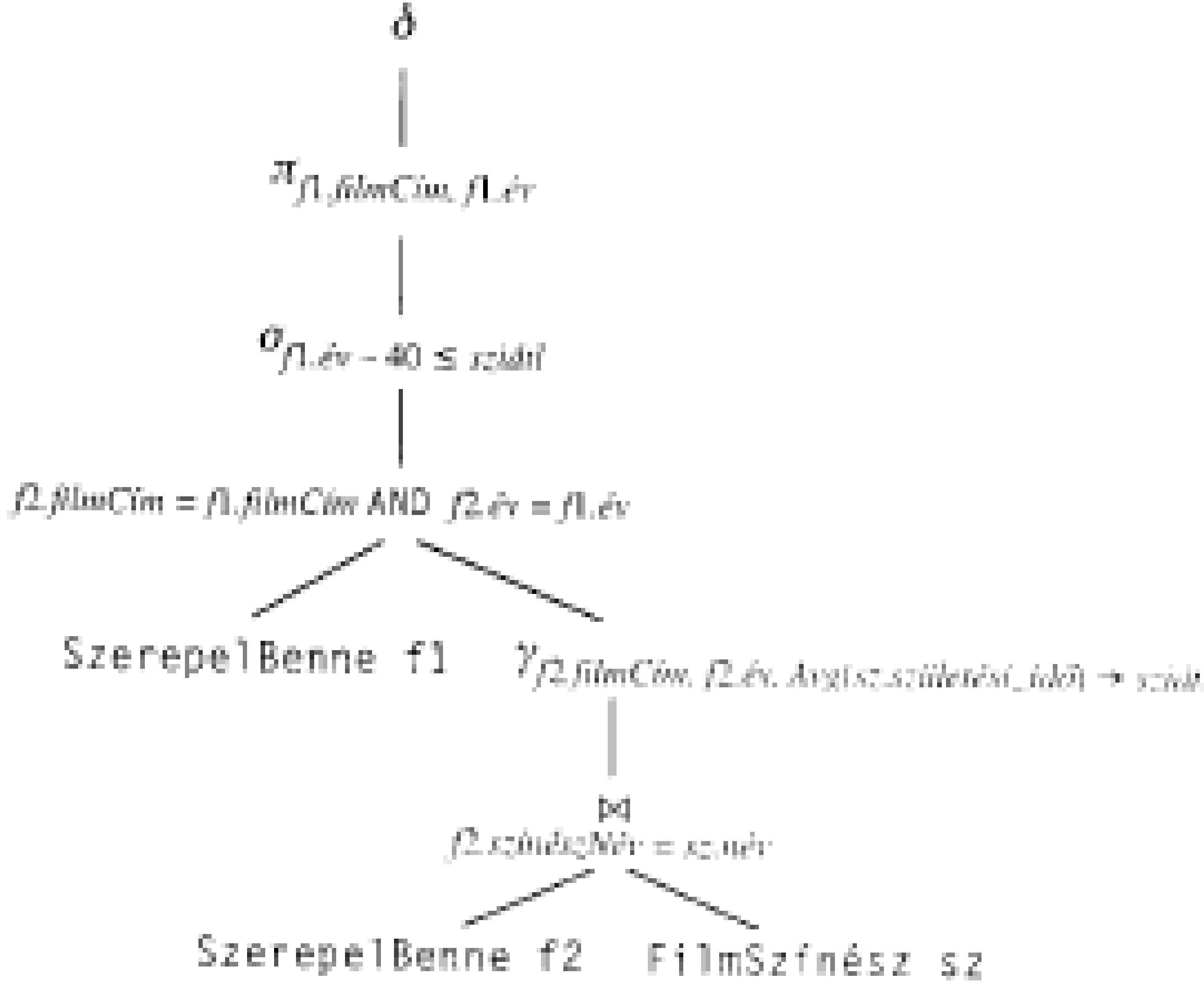
```
SELECT DISTINCT f1.filmCím, f1.év
FROM SzerepelBenne f1
WHERE f1.év - 40 <= (
            SELECT AVG(születési_idő)
            FROM SzerepelBenne f2, FilmSzínész sz
                  WHERE f1.filmCím = f2.cím AND
                        f1.év = f2.év AND
                        f2.színészNév = sz.név
                  );
```

Keressük meg az olyan filmeket, ahol színészek átlagéletkora legfeljebb 40 év volt, amikor film készült. (Értelmezés: *születési_idő := születési év;* f2.születési_idő=f2.év)

```
SzerepelBenne(filmCím, év, színészNév)
FilmSzínész(név, cím, nem, születési_idő)
```



7.18. ábra. *Részlegesen transzformált elemzőfa a 7.17. ábra lekérdezéséhez*

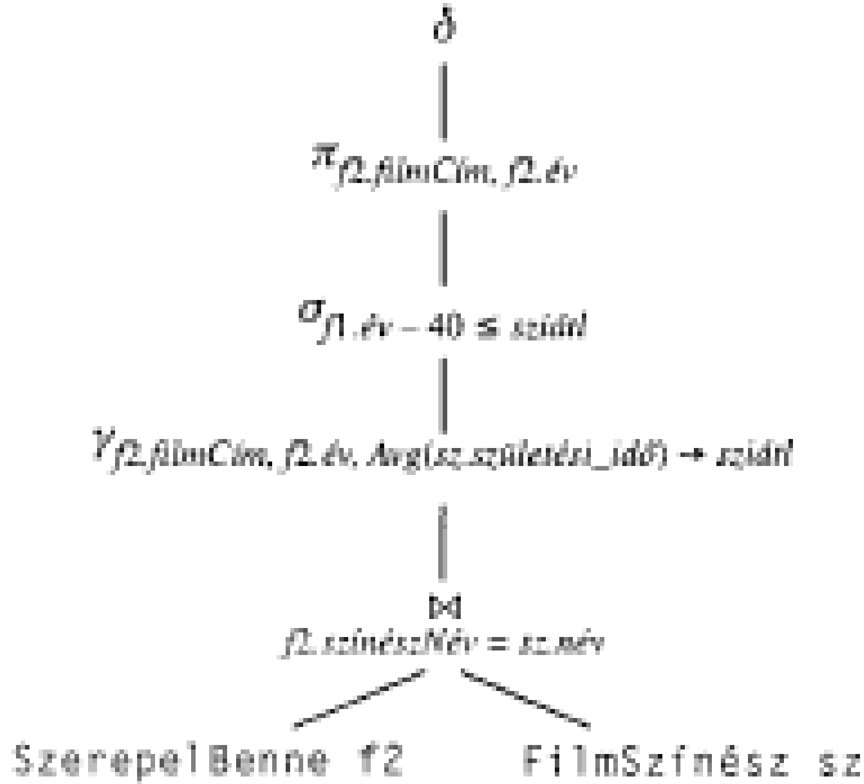


7.19. ábra. *A 7.18. ábra átalakítása egy logikai lekérdezéstervvé*

sziátl= születési idők átlaga

Feltételek egyszerűsítésre

- A teljesülő feltételek:
- Az ismétlődések megszűntetése a végén történik.
- A vetítés `SzerepelBenne f1` relációból kihagyja a színészek neveit.
- `SzerepelBenne f1` és a maradék kifejezés közti összekapcsolásegyenlővé teszi a `SzerepelBenne f1` és `SzerepelBenne f2` relációk `filmCím` és `év` attribútumait.



7.20. ábra. *A 7.19. ábra egyszerűsítése*

Szabály alapú optimalizálás

- Szeretnénk minél kevesebb lemez olvasási és írási (I/O) műveletet végrehajtani egy-egy lekérdezés végrehajtása során.
- Az legegyszerűbb megközelítés, ha igyekszünk minél kisebb méretű relációkkal dolgozni.
- Az optimalizáció során relációs algebrai azonosságokat fogunk alkalmazni. Ezek segítségével egy lekérdezésből az eredetivel ekvivalens lekérdezést készítünk, amelynek kiszámítása az esetek többségében kevesebb I/O műveletet igényel majd.
- A q, q' relációs algebrai lekérdezések (vagy tetszőleges lekérdezések) ekvivalensek, ha tetszőleges I előfordulás esetén $q(I) = q'(I)$ fennáll. Jelben: $q \equiv q'$.

Egy példa...

- A táblák legyenek:

  Film (cím, év, hossz)

  Szerepel (filmcím, év, színésznév)
- Ekkor a következő lekérdezés:

  $\Pi_{\text{cím}}(\sigma_{\text{cím=filmcím} \wedge \text{F.év=Sz.év} \wedge \text{színésznév='Edus'}}\ (F \times Sz))$

  ekvivalens a

  $\Pi_{\text{cím}}(\sigma_{\text{cím=filmcím} \wedge \text{F.év=Sz.év}}(F \times (\sigma_{\text{színésznév='Edus'}}\ (Sz))))$

  lekérdezéssel.

- Emellett az utóbbi valószínűleg gyorsabban végrehajtható.

Descartes-szorzat és összekapcsolások

- Asszociativitás:

  $(E_1\ \Theta\ E_2)\ \Theta\ E_3 \equiv E_1\ \Theta\ (E_2\ \Theta\ E_3)$, ahol $\Theta \in \{\times, \bowtie\}$ és

  $(E_1 \bowtie_{F1} E_2) \bowtie_{F2} E_3 \equiv E_1 \bowtie_{F1} (E_2 \bowtie_{F2} E_3)$, ha

  $attr(F1) \subseteq attr(E1) \cup attr(E2)$ és $attr(F2) \subseteq attr(E2) \cup attr(E3)$

- Kommutativitás:

  $E_1\ \Theta\ E_2 \equiv E_2\ \Theta\ E_1$, ahol $\Theta \in \{\times, \bowtie, \bowtie_F\}$.

Projekció és szelekció

- Projekciók sorozata:

$$\Pi_X(\Pi_Y(E)) \equiv \Pi_X(E), \text{ ha } X \subseteq Y.$$

- Kiválasztás és a feltételek konjunkciója:

$$\sigma_{F1 \wedge F2}(E) \equiv \sigma_{F1}(\sigma_{F2}(E)).$$

- Kiválasztás és a feltételek diszjunkciója:

$$\sigma_{F1 \vee F2}(E) \equiv \sigma_{F1}(E) \cup \sigma_{F2}(E).$$

- Kiválasztás elé projekció beillesztése:

$$\Pi_X(\sigma_F(E)) \equiv \Pi_X(\sigma_F(\Pi_Y(E))), \text{ ahol } Y = \mathrm{attr}(F) \cup X.$$

Kiválasztás és Descartes-szorzat/összekapcsolás

- Kiválasztás és Descartes-szorzat, összekapcsolás felcserélése:

  $\sigma_F (E_1 \Theta E_2) \equiv \sigma_F (E_1) \Theta E_2$,

  ahol attr (F) $\subseteq$ attr (E_1) és $\Theta \in \{\times, \bowtie\}$.

- Általánosabban:

  $\sigma_F (E_1 \Theta E_2) \equiv \sigma_{F1} (E_1) \Theta \sigma_{F2} (E_2)$,

  ahol attr (F_i) $\subseteq$ attr (E_i) (i = (1, 2)), $F = F_1 \wedge F_2$ és $\Theta \in \{\times, \bowtie\}$.

Kiválasztás és Descartes-szorzat/összekapcsolás

- Picit másképp:

  $\sigma_{F}(E_1 \, \Theta \, E_2) \equiv \sigma_F(E_1) \, \Theta \, \sigma_F(E_2)$,

  ahol attr $(F) \subseteq$ attr $(E_1) \cap$ attr (E_2) és $\Theta \in \{\times, \bowtie\}$.

- Ezekből levezethető:

  $\sigma_{F}(E_1 \, \Theta \, E_2) \equiv \sigma_{F2}(\sigma_{F1}(E_1) \, \Theta \, E_2)$,

  ahol attr $(F_1) \subseteq$ attr (E_1), $F = F_1 \wedge F_2$, de attr $(F_2) \subseteq$ attr (E_i) nem teljesül $(i = (1, 2))$, $\Theta \in \{\times, \bowtie\}$.

Projekció és Descartes-szorzat/összekapcsolás

- Projekció és Descartes-szorzat, összekapcsolás felcserélése:

  $\Pi_X(E_1\ \Theta\ E_2) \equiv \Pi_Y(E_1)\ \Theta\ \Pi_Z(E_2)$,

  ahol $X = Y \cup Z$, $Y \subseteq attr\,(E_1)$, $Z \subseteq attr\,(E_2)$ és $\Theta \in \{\times, \bowtie\}$.

Projekció/kiválasztás és (multi)halmazműveletek

- Kiválasztás és unió (különbség) felcserélése:

$$\sigma_F\,(E_1\ \Theta\ E_2) \equiv \sigma_F\,(E_1)\ \Theta\ \sigma_F\,(E_2),\ \text{ahol}\ \Theta \in \{\cup, -\}.$$

- Projekció unióval való felcserélése:

$$\Pi_X(E_1 \cup E_2) \equiv \Pi_X(E_1) \cup \Pi_X(E_2).$$

- Megjegyzés: nincs általános szabály a projekció különbséggel való felcserélésére.
- Multihalmaz-műveletek esetén a projekció szintén csak a multihalmaz-egyesítéssel cserélhető fel.

Ismétlődések kiküszöbölése

- $\delta(R) = R$ (nincsenek benne ismétlődések) például, ha
  - R-hez megadtunk egy elsődleges kulcsot,
  - R-et csoportosítás eredményeként kaptuk (γ).
- Ismétlődések kiküszöbölése és Descartes-szorzat, összekapcsolás:

  $\delta(R)\ \theta\ \delta(S) \equiv \delta(R\ \theta\ S),\ \theta \in \{\times, \bowtie, \bowtie_F\}.$
- Ismétlődések kiküszöbölése és a kiválasztás:

  $\delta(\sigma_F(R)) \equiv \sigma_F(\delta(R)).$
- Ismétlődések kiküszöbölése és multihalmaz-metszet:

  $\delta(R \cap_M S) \equiv \delta(R) \cap_M S \equiv R \cap_M \delta(S) \equiv \delta(R) \cap_M \delta(S).$
- A halmazműveletek és δ felcserélése értelmetlen.

Csoportosítás kiküszöbölése

- $\delta(\gamma_L(R)) \equiv \gamma_L(R)$.
- $\gamma_L(R) \equiv \gamma_L(\Pi_M(R))$. *M* az *R* olyan attribútumainak listája, amelyek *L*-ben előfordulnak.
- $\gamma_L(R) = \gamma_L(\delta(R))$ (ha γ_L ***ismétlődés érzéketlen*** , ha *L*-ben csak MIN és/vagy MAX összesítő fv.-k szerepelnek).

Példa optimalizálásra

- A következő két feladathoz használt táblák:

  Személy (név, kor, város, ISBN)

  Könyv (cím, író, ISBN, ár)

  Kiad (k_cím, k_író, város, ország)

- Kik azok, akik 20 évesek, és moszkvai kiadású könyvet kölcsönöztek ki?

$$\Pi_{név}(\sigma_{Sz.ISBN=Kö.ISBN \wedge cím=k\_cím \wedge író=k\_író \wedge kor=20 \wedge K.város=Moszkva}(Sz \times Kö \times K))$$

Lekérdezésfa

$$\Pi_{név}$$

$$\sigma_{Sz.ISBN=Kö.ISBN \wedge cím=k\_cím \wedge író=k\_író \wedge kor=20 \wedge K.város=Moszkva}$$

$$\times$$

Sz $\quad\quad \times$

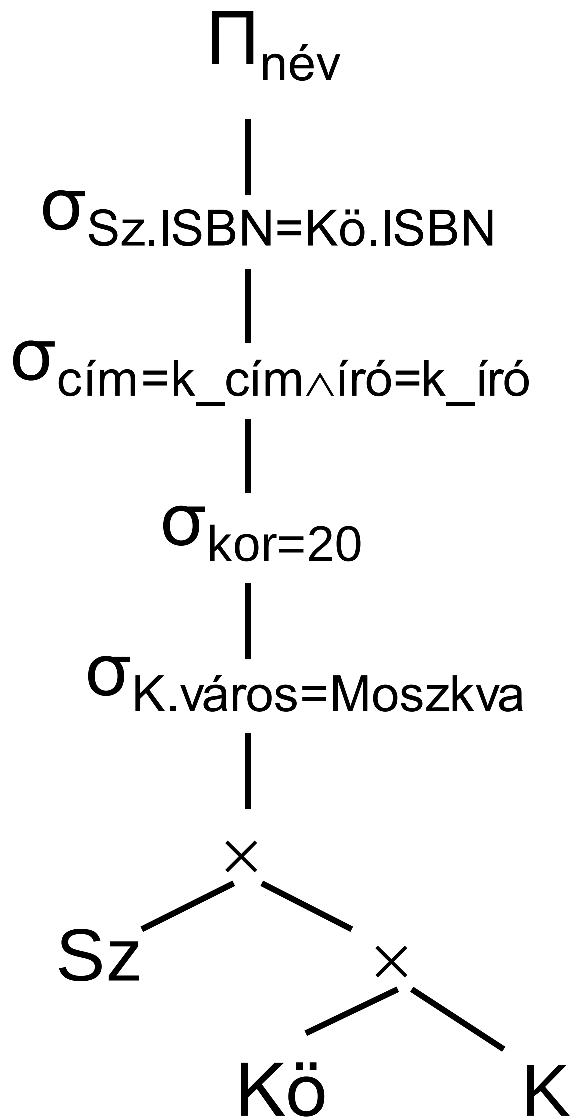
Kö $\quad\quad$ K

Kiválasztások "lejjebb csúsztatása"

- Első lépésben a kiválasztások konjunkciós feltételeit daraboljuk szét elemi feltételekké a $\sigma_{F1 \wedge F2}(E) \equiv \sigma_{F1}(\sigma_{F2}(E))$ szabály segítségével.
- Ezek után alkalmazzuk a kiválasztás halmazműveletekkel illetve Descartes-szorzattal és a természetes összekapcsolással való felcserélésének szabályait.
- Azaz: igyekszünk a kiválasztásokat minél hamarabb végrehajtani, hiszen azok jelentősen csökkenthetik a feldolgozandó köztes relációk méretét.
- A Théta-összekapcsolást itt jobb, ha egy Descartes-szorzatra és egy azt követő kiválasztásra bontjuk.

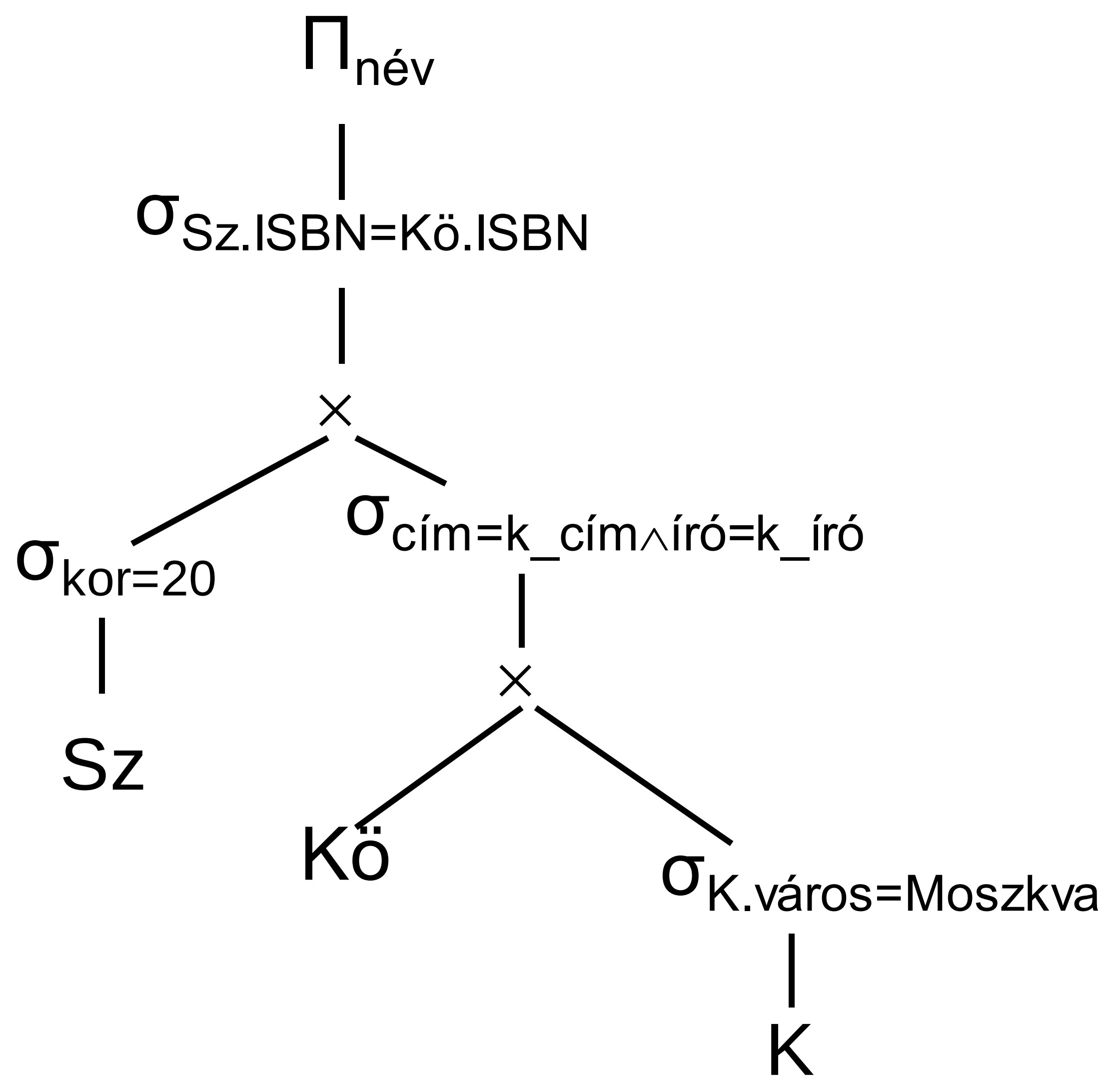
  $R \bowtie_F S \equiv \sigma_F(R \times S)$.

Darabolás



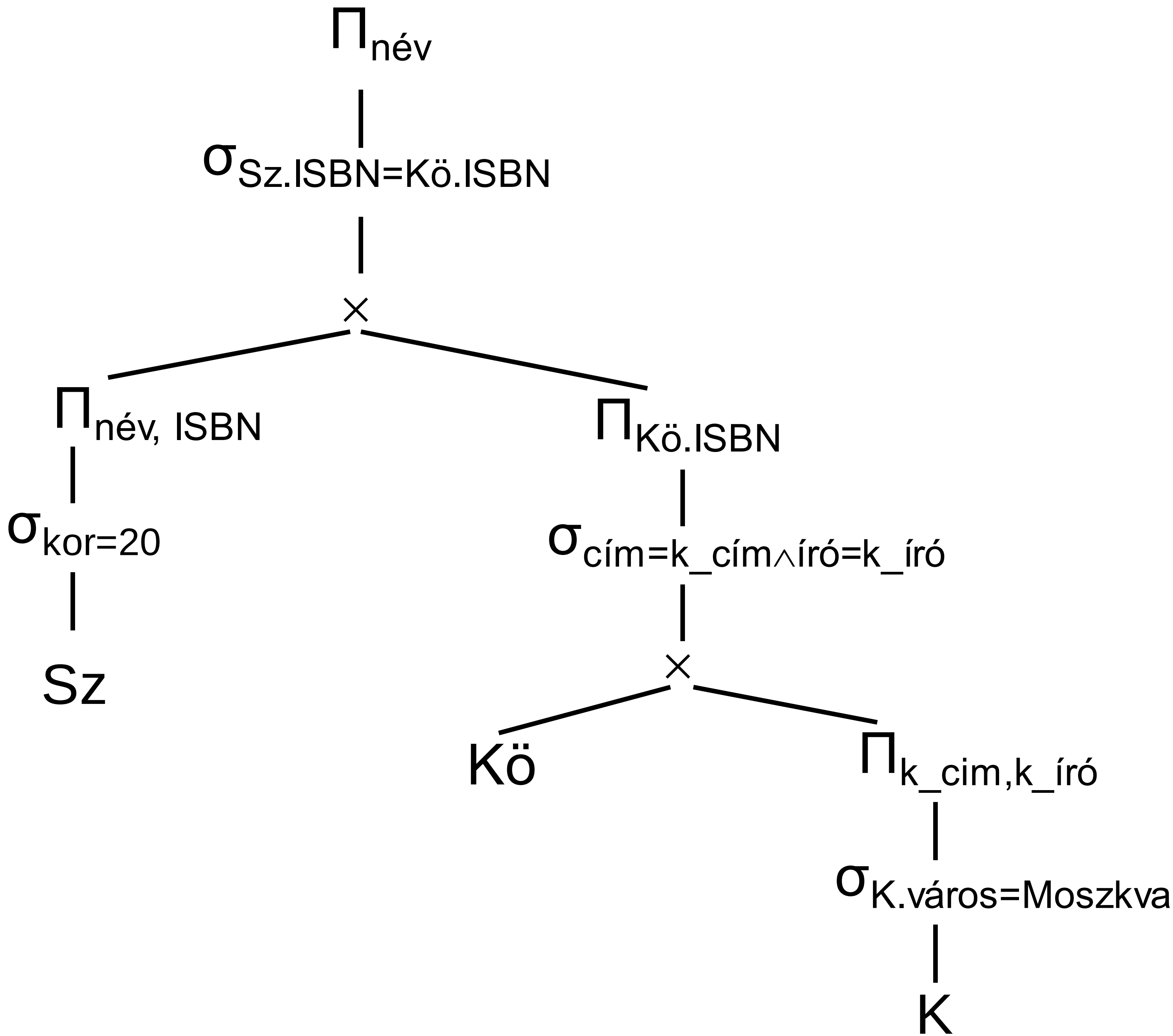
$\Pi_{név}$
$\sigma_{Sz.ISBN=Kö.ISBN}$
$\sigma_{cím=k\_cím \wedge író=k\_író}$
$\sigma_{kor=20}$
$\sigma_{K.város=Moszkva}$
$\times$
Sz
$\times$
Kö
K

Letolás


Π név
σ Sz.ISBN=Kö.ISBN
×
σ kor=20
σ cím=k_cím∧író=k_író
Sz
×
Kö
σ K.város=Moszkva
K

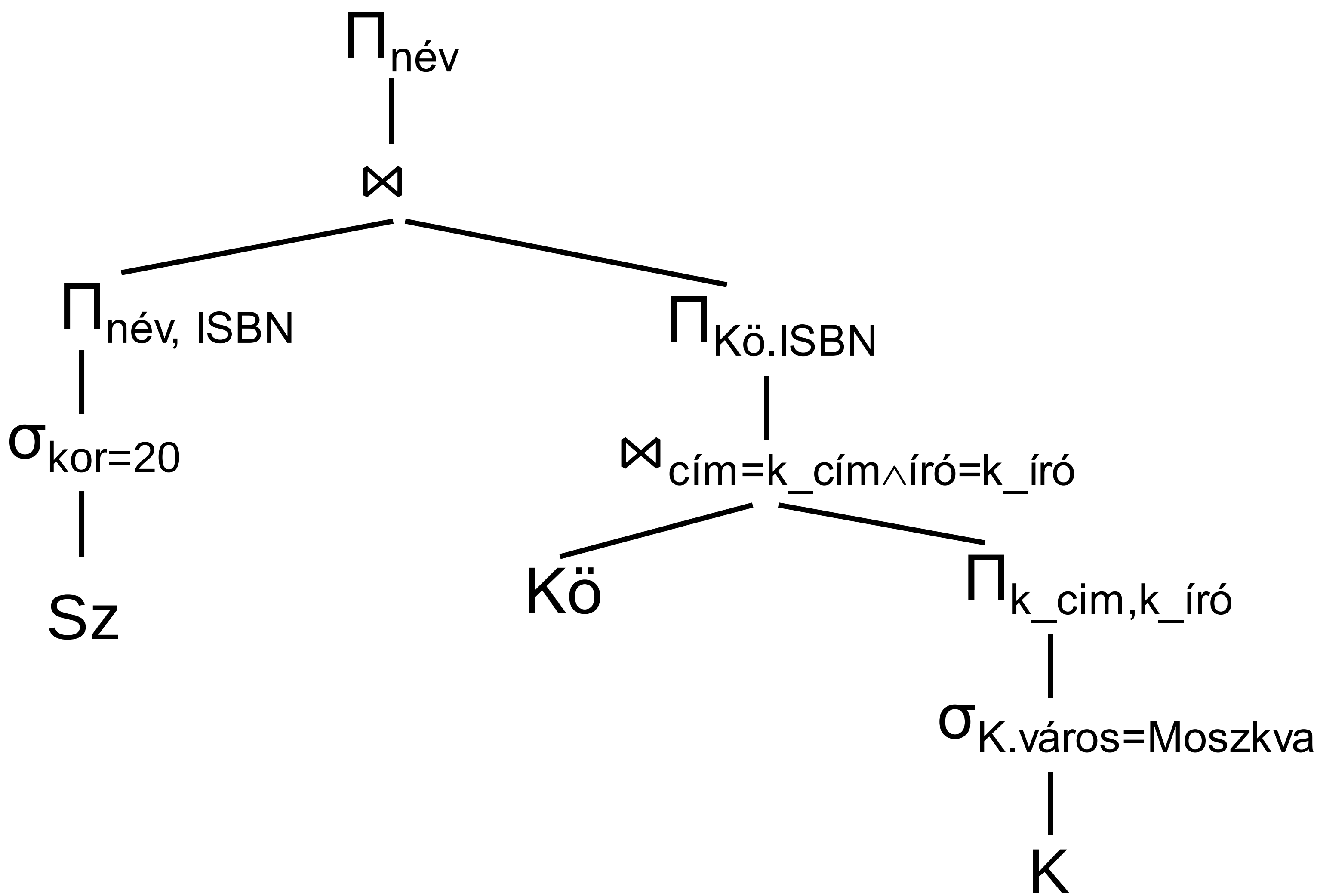
Projekciók "beírása"

- Ennél a lépésnél igyekszünk csak azokat az oszlopokat megtartani a (köztes) relációkban, amelyekre később szükség lesz.
- Általában itt nem olyan nagy a nyereség. A projekciók végrehajtása viszont időt igényel, ezért meg kell gondolni, hogy tényleg végre akarjuk-e hajtani a vetítést.
- Az átalakításoknál értelemszerűen a projekciókra vonatkozó szabályokat használjuk.


Πnév
σSz.ISBN=Kö.ISBN
×
Πnév, ISBN
ΠKö.ISBN
σkor=20
σcím=k_cím∧író=k_író
Sz
×
Kö
Πk_cim,k_író
σK.város=Moszkva
K

Összekapcsolások

- Az utolsó lépésben $\Pi_L(\sigma_C(R \times S))$, $\sigma_C(R \times S)$ kifejezéseket helyettesítjük természetes összekapcsolással, Théta-összekapcsolással úgy, hogy az eddigivel ekvivalens lekérdezést kapjunk.


$\Pi_{név}$
$\bowtie$
$\Pi_{név,\ ISBN}$
$\sigma_{kor=20}$
Sz
$\Pi_{Kö.ISBN}$
$\bowtie_{cím=k\_cím \wedge író=k\_író}$
Kö
$\Pi_{k\_cim,k\_író}$
$\sigma_{K.város=Moszkva}$
K

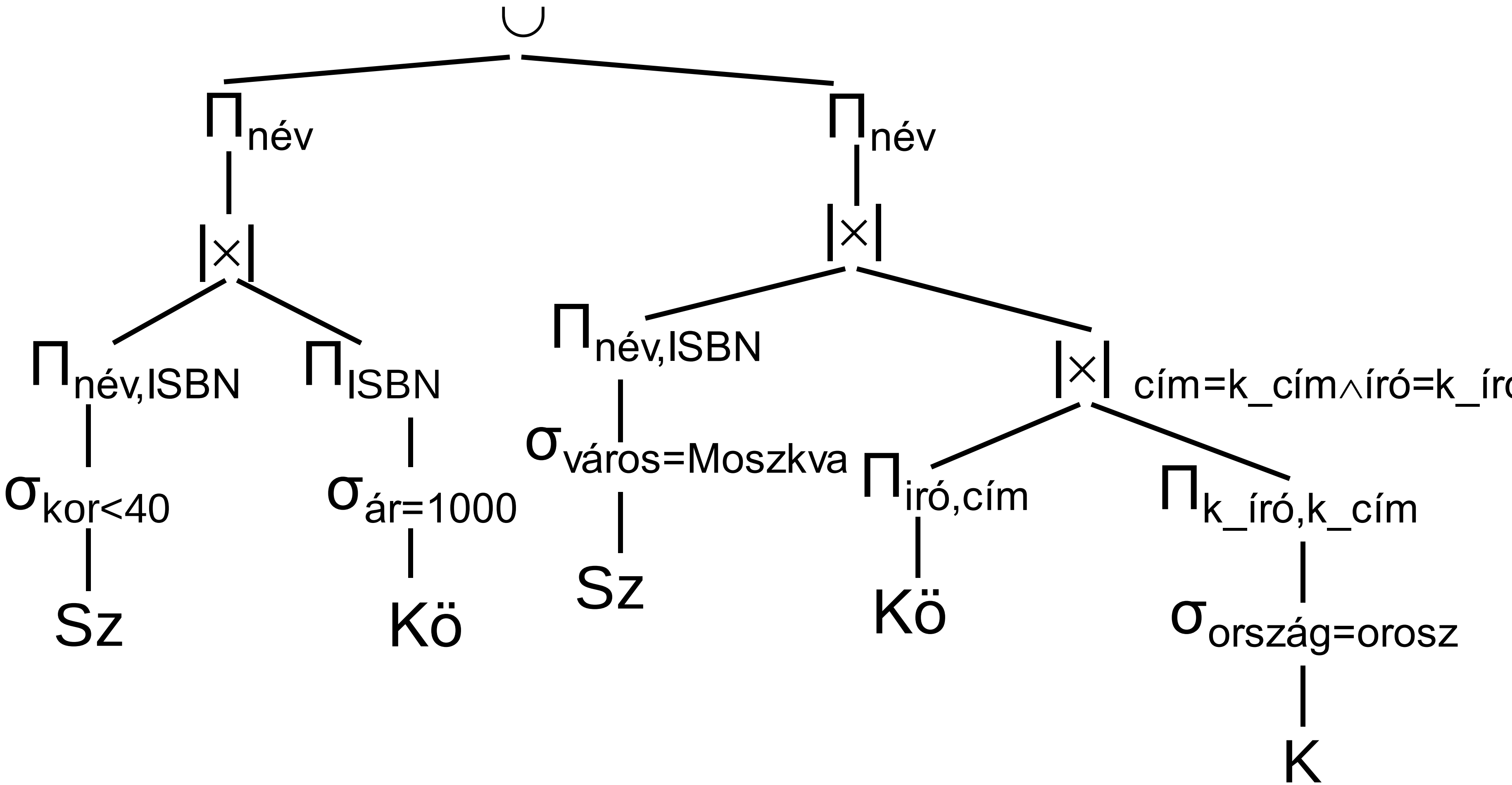
Mi történik, ha a diszjunkció is megjelenik?

- Kik azok, akik 1000 forintos könyvet vásároltak, és még nincsenek 40 évesek, vagy moszkvaiak, és orosz kiadású könyvet vettek?

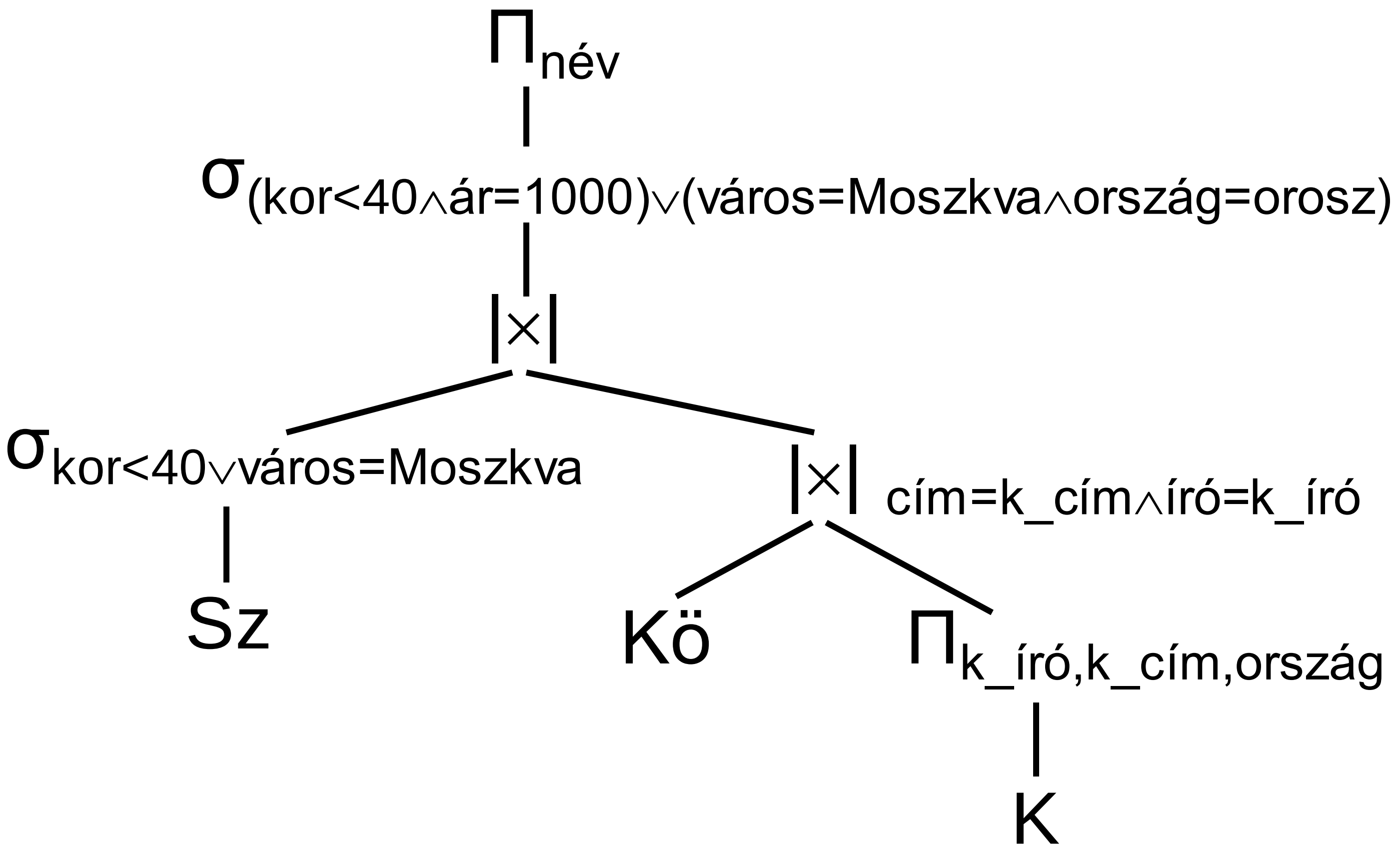
$$\Pi_N(\sigma_{C\wedge((\text{ár}=1000\wedge \text{kor}<40)\vee(\text{Sz.város}=\text{Moszkva}\wedge \text{ország}=\text{orosz}))}(Sz \times Kö \times K)).$$

- Itt C az Sz.ISBN = Kö.ISBN $\wedge$ Kö.cím = K.k_cím $\wedge$ Kö.író = K.k_író feltételt jelöli.

Megoldás I.


∪
Π név
Π név
|×|
|×|
Π név,ISBN
Π ISBN
Π név,ISBN
|×| cím=k_cím∧író=k_író
σ kor<40
σ ár=1000
σ város=Moszkva
Π iró,cím
Π k_író,k_cím
Sz
Kö
Sz
Kö
σ ország=orosz
K

Megoldás II.



$\Pi_{név}$
$\sigma_{(kor<40 \wedge ár=1000) \vee (város=Moszkva \wedge ország=orosz)}$
$|\times|$
$\sigma_{kor<40 \vee város=Moszkva}$
$|\times|$ $cím=k\_cím \wedge író=k\_író$
Sz
Kö
$\Pi_{k\_író,k\_cím,ország}$
K

Összegzés

- Ha tehát a kiválasztások feltételei diszjunkciót is tartalmaznak, a helyzet bonyolultabbá válik, és nem adható olyan egyértelmű optimalizációs algoritmus, mint konjunkciók esetén.

Kiválasztások feljebb csúsztatása

- A következő példa azt szemlélteti, amikor egy kiválasztást először felfelé kell csúsztatnunk, hogy aztán letolhassuk.
- A táblák:

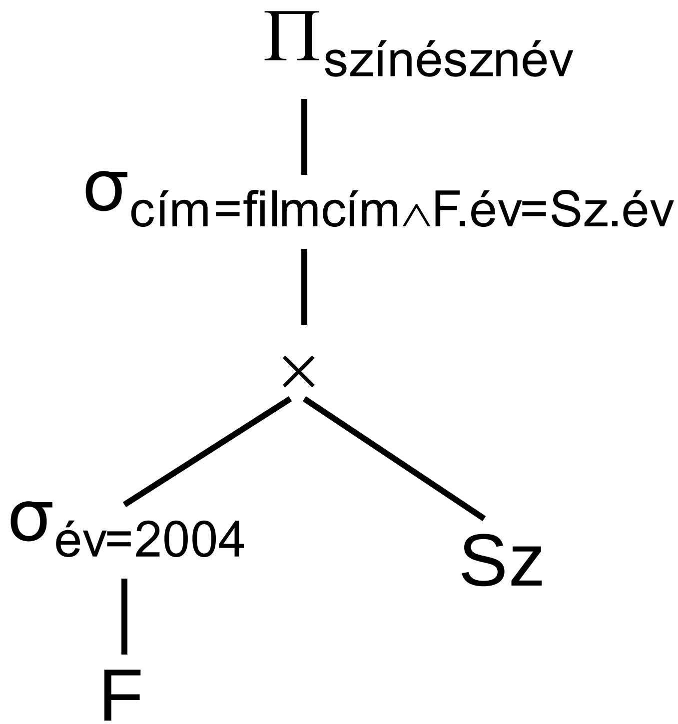
    Film (cím, év, hossz)

    Szerepel (filmcím, év, színésznév)

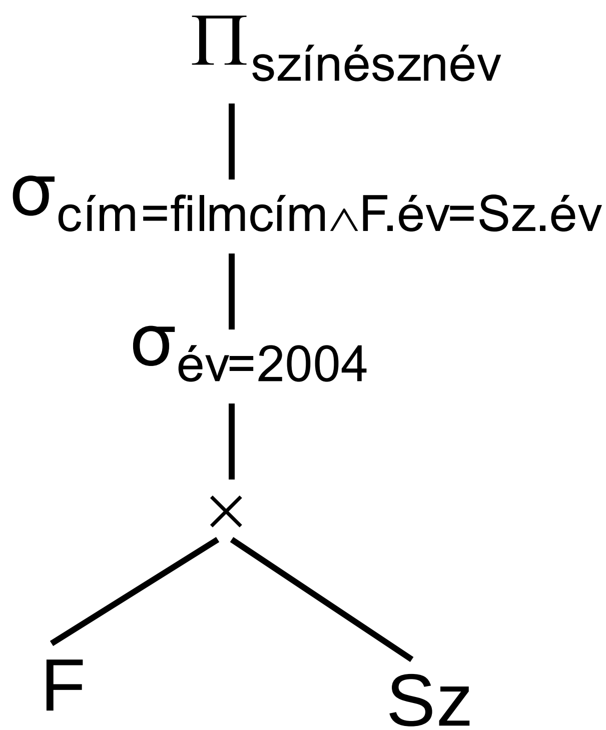
```
CREATE VIEW film04 AS
(SELECT *
FROM film
WHERE év = 2004);
```

```
SELECT színésznév
FROM film04 f, Szerepel sz
WHERE cím = filmcím AND
      f.év = sz.év;
```

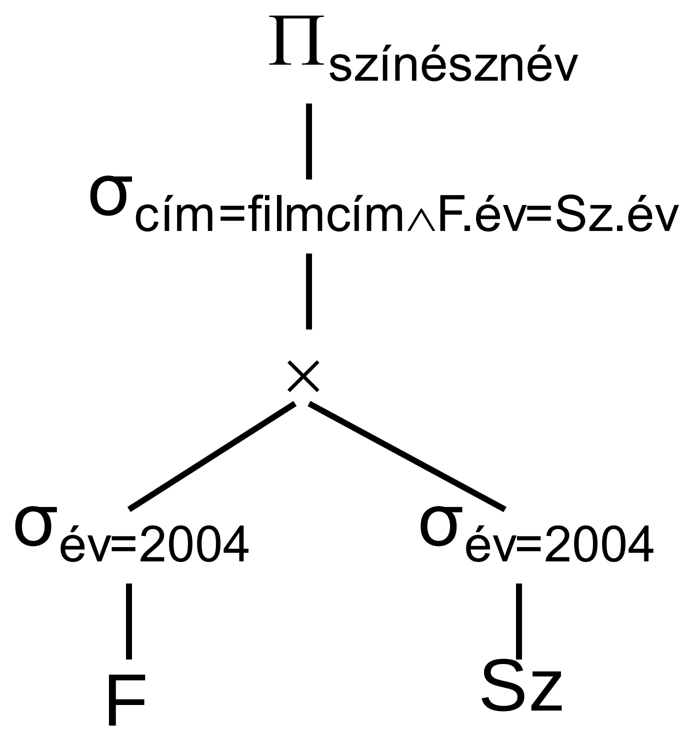
Kezdeti lekérdezésfa



Második lépés



És az eredmény…



Feladat

- A táblák legyenek:

  Szerepel (filmcím, év, színésznév)

  Színész (név, születési év, város, neme)

Keressük meg azokat a filmeket, amelyekben van olyan filmszínész, aki 1960-ban született.

SQL query

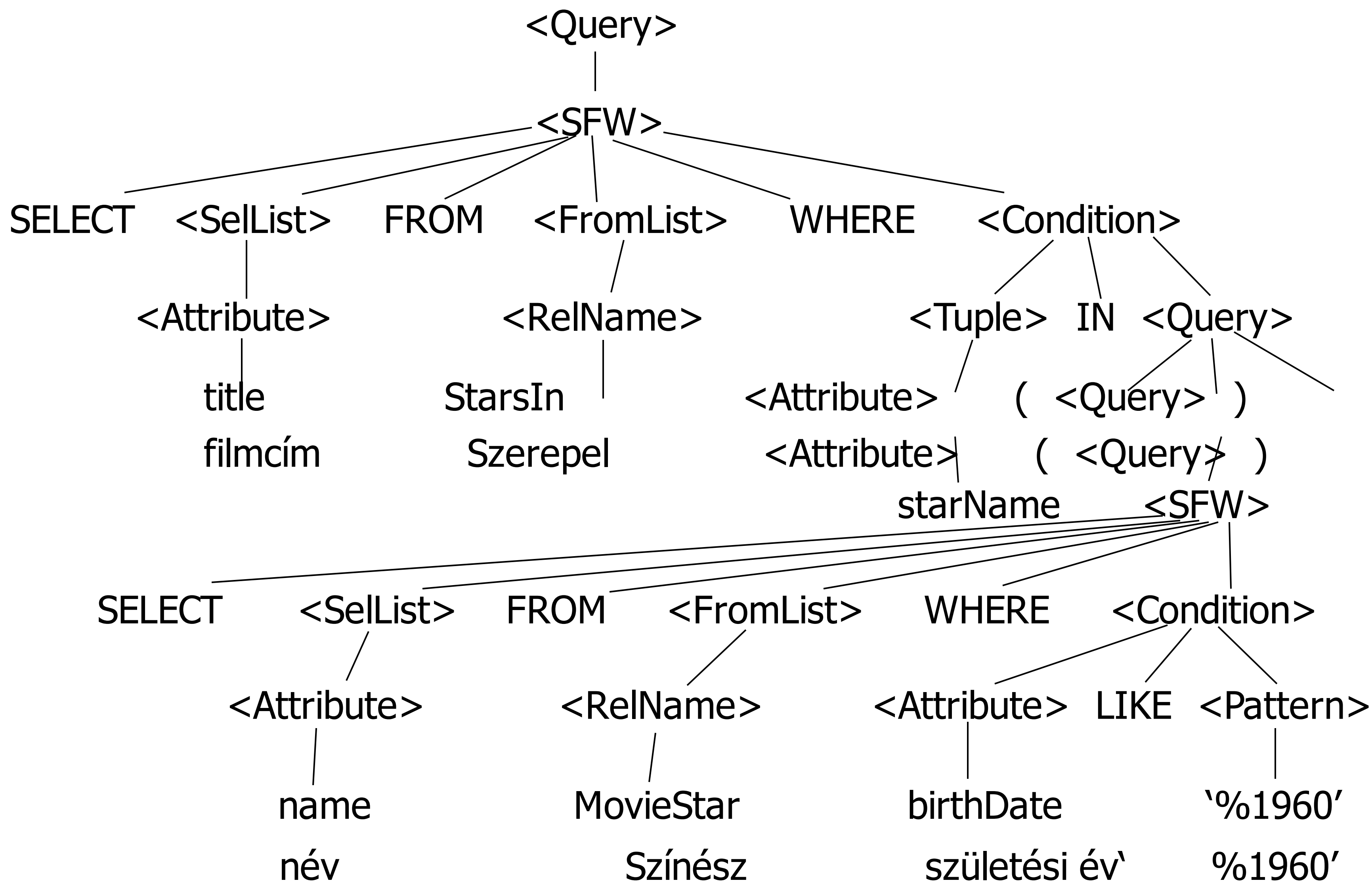
```
SELECT filmcím
FROM Szerepel
WHERE színésznév IN (
          SELECT név
          FROM Színész
          WHERE születési év LIKE
     ‘%1960’
);
```

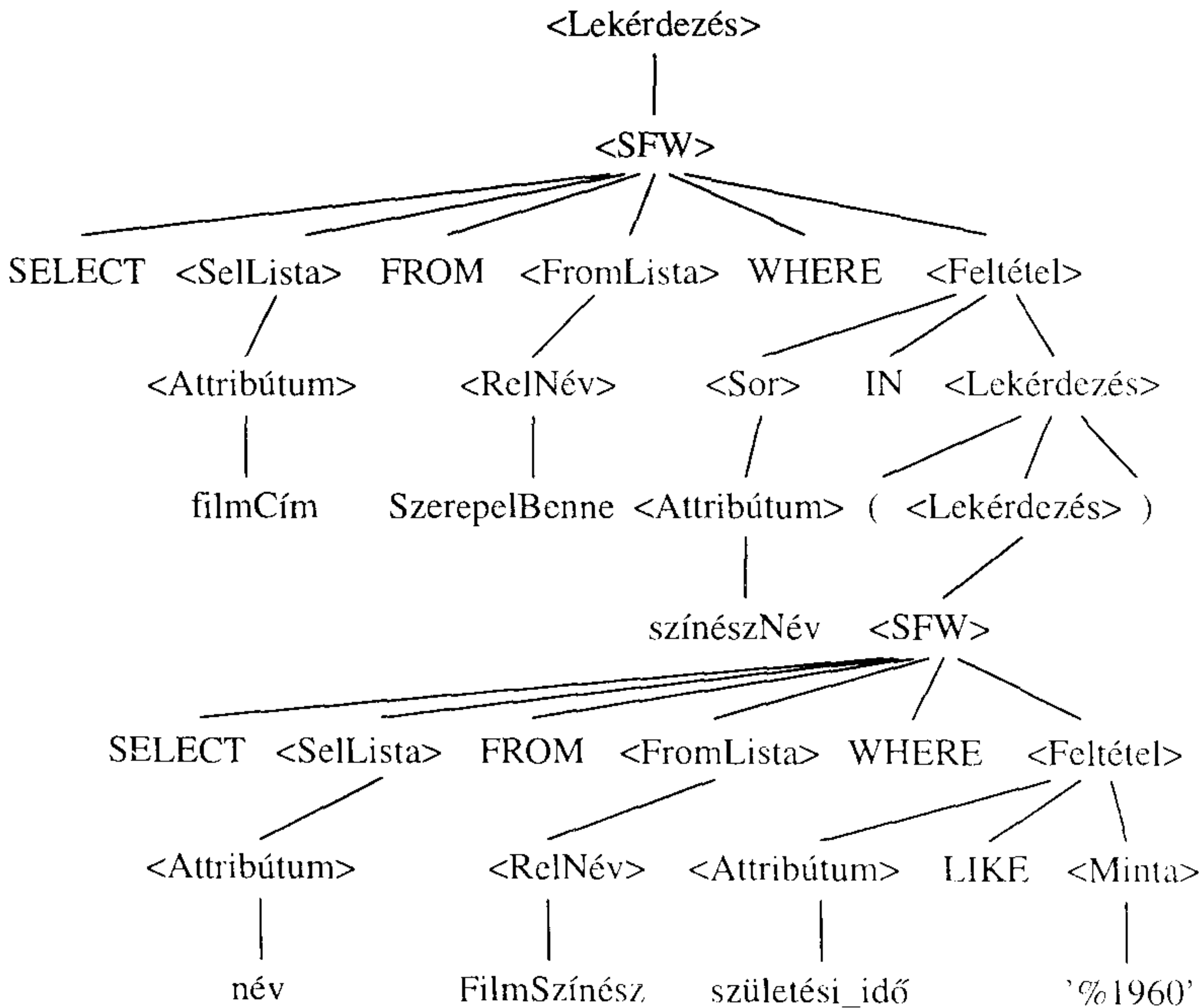
(Keressük meg azokat a filmeket, amelyekben van olyan filmszínész, aki 1960-ban született.
)

```
SELECT title
FROM StarsIn
WHERE starName IN (
          SELECT name
          FROM MovieStar
          WHERE birthdate
     LIKE ‘%1960’
);
```

(Find the movies with stars born in 1960)

Example: Parse Tree




<Lekérdezés>
<SFW>
SELECT
<SelLista>
FROM
<FromLista>
WHERE
<Feltétel>
<Attribútum>
<RelNév>
<Sor>
IN
<Lekérdezés>
filmCím
SzerepelBenne
<Attribútum>
(
<Lekérdezés>
)
színészNév
<SFW>
SELECT
<SelLista>
FROM
<FromLista>
WHERE
<Feltétel>
<Attribútum>
<RelNév>
<Attribútum>
LIKE
<Minta>
név
FilmSzínész
születési_idő
'%1960'

Example: Generating Relational Algebra

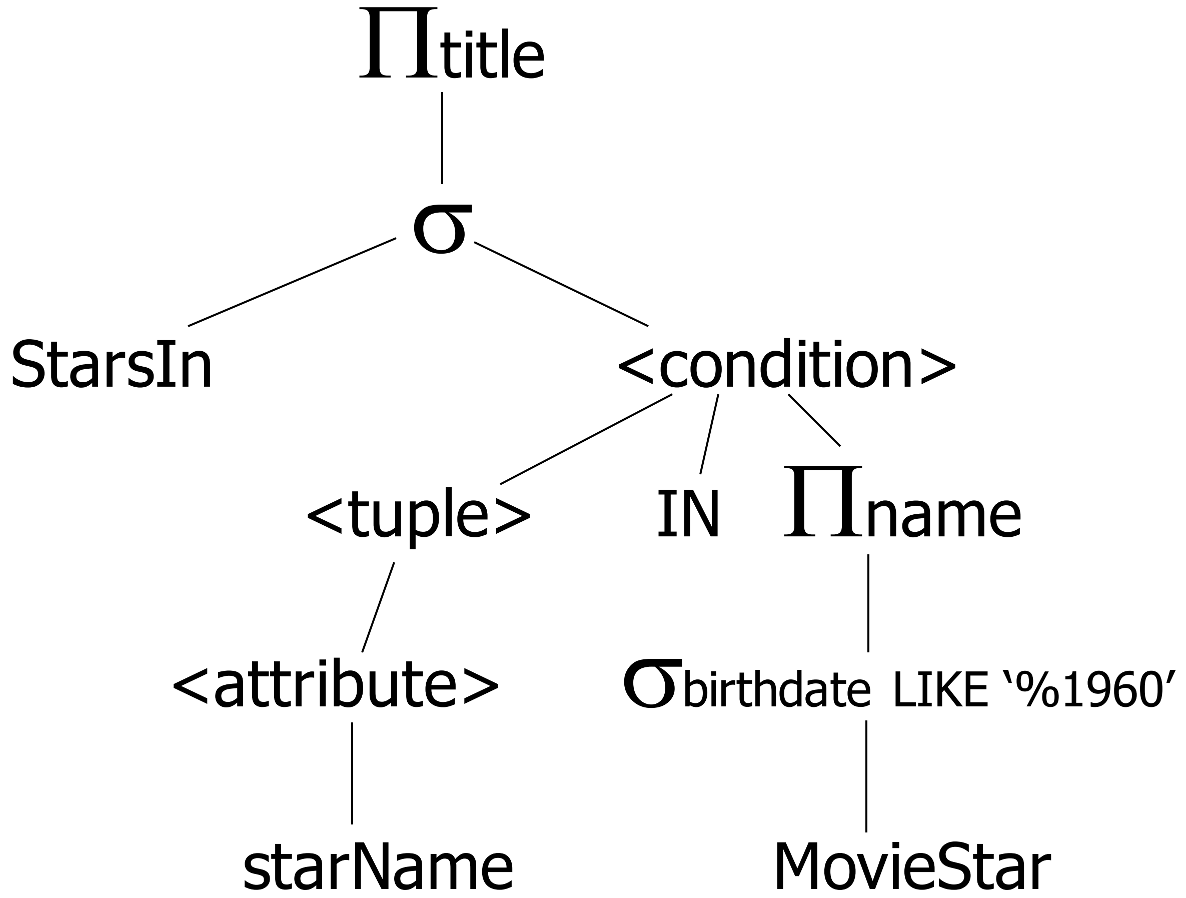
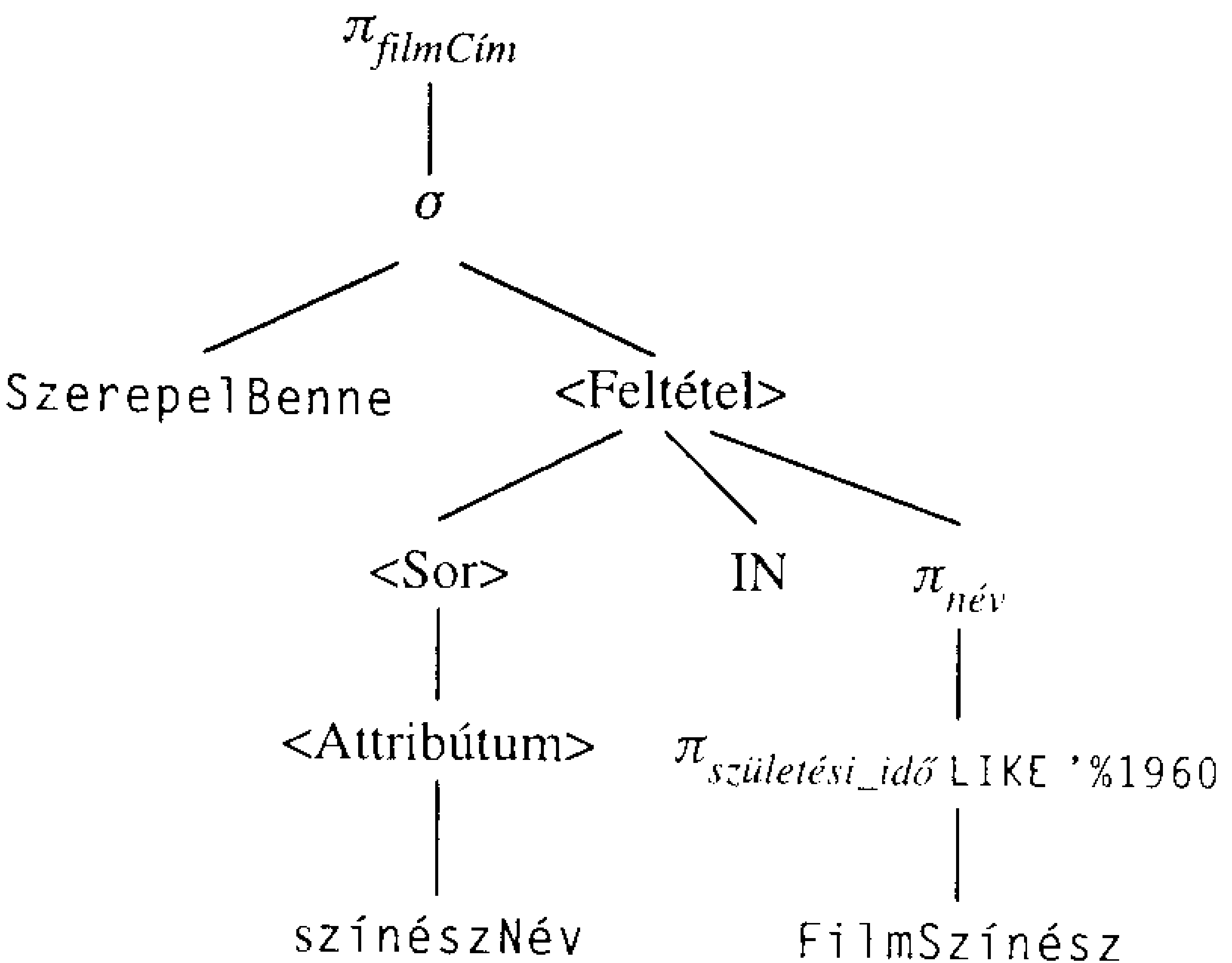


Fig. ***7.14*** (16.17): An expression using a two-argument σ, midway between a parse tree and relational algebra



$\pi_{filmCím}$
σ
SzerepelBenne
<Feltétel>
<Sor>
IN
$\pi_{név}$
<Attribútum>
$\pi_{születési\_idő}$ LIKE '%1960'
színészNév
FilmSzínész

Example: Logical Query Plan

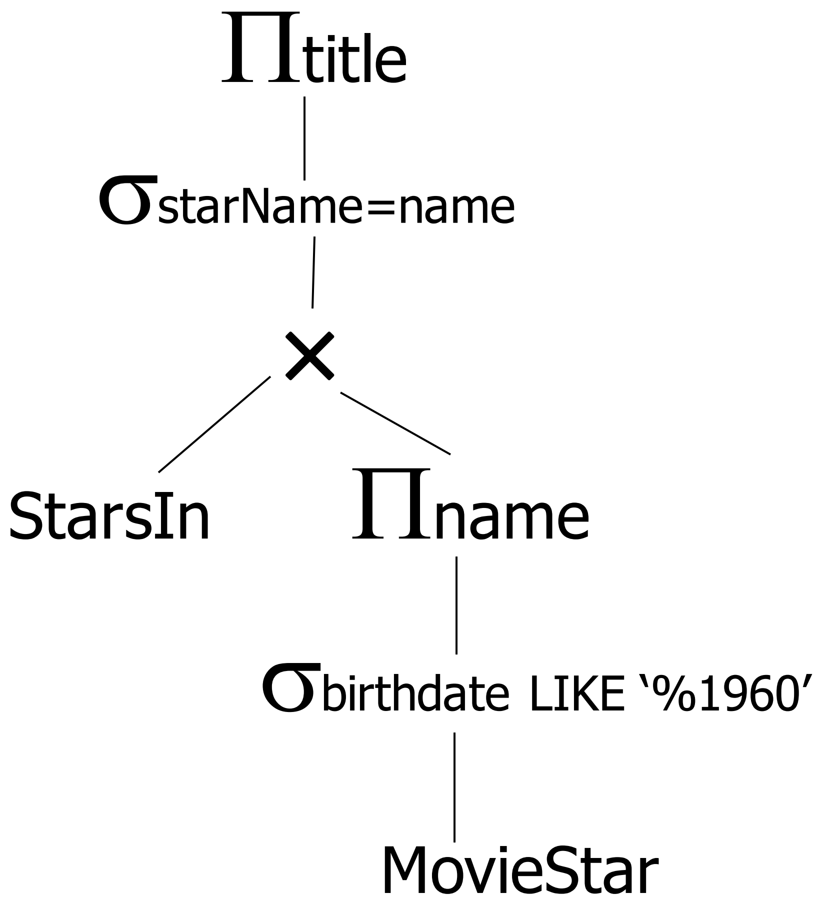
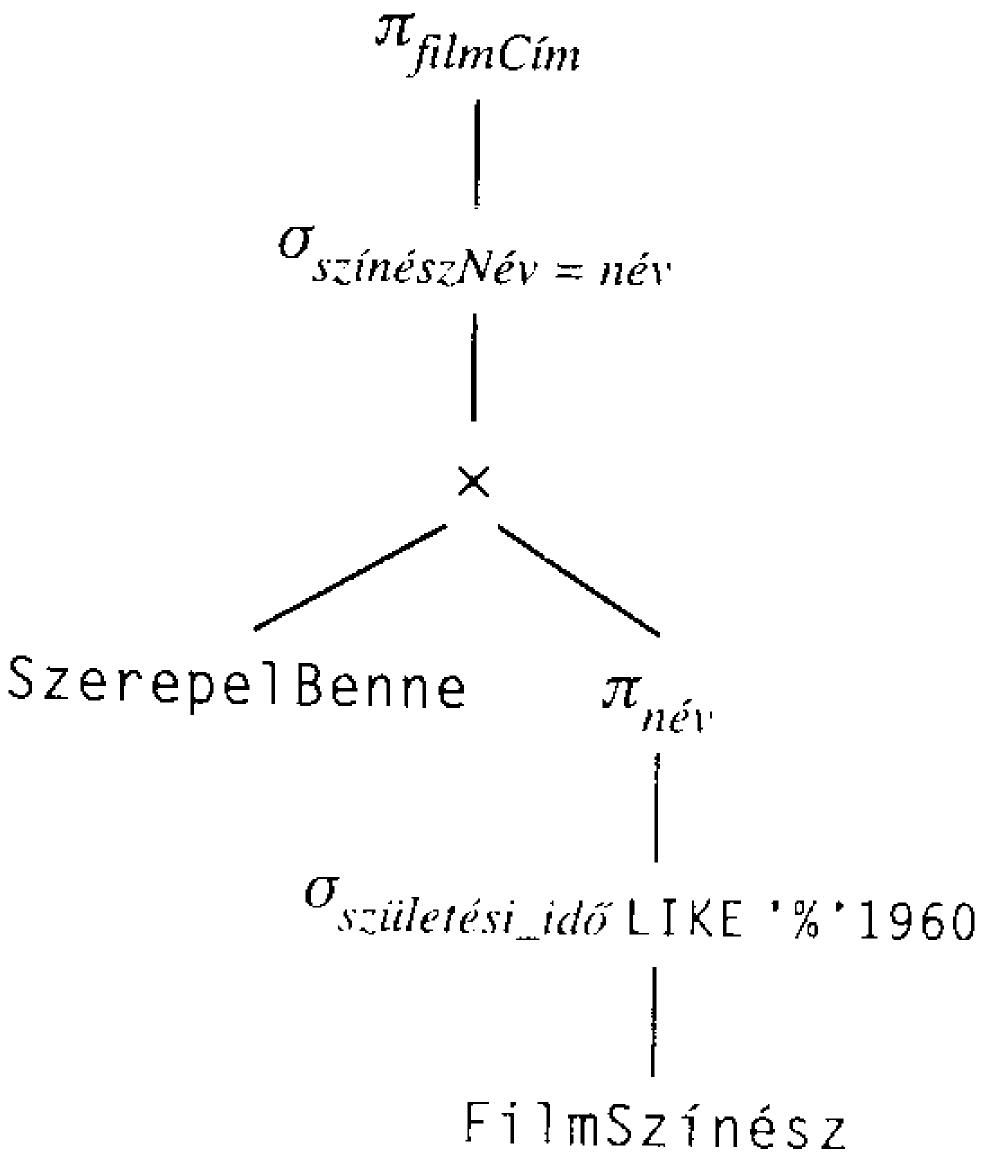


Fig. ***7.16***: Applying the rule for IN conditions


$\pi_{filmCím}$
$\sigma_{színészNév = név}$
$\times$
SzerepelBenne
$\pi_{név}$
$\sigma_{születési\_idő}$ LIKE '%'1960'
FilmSzínész

Example: Improved Logical Query Plan

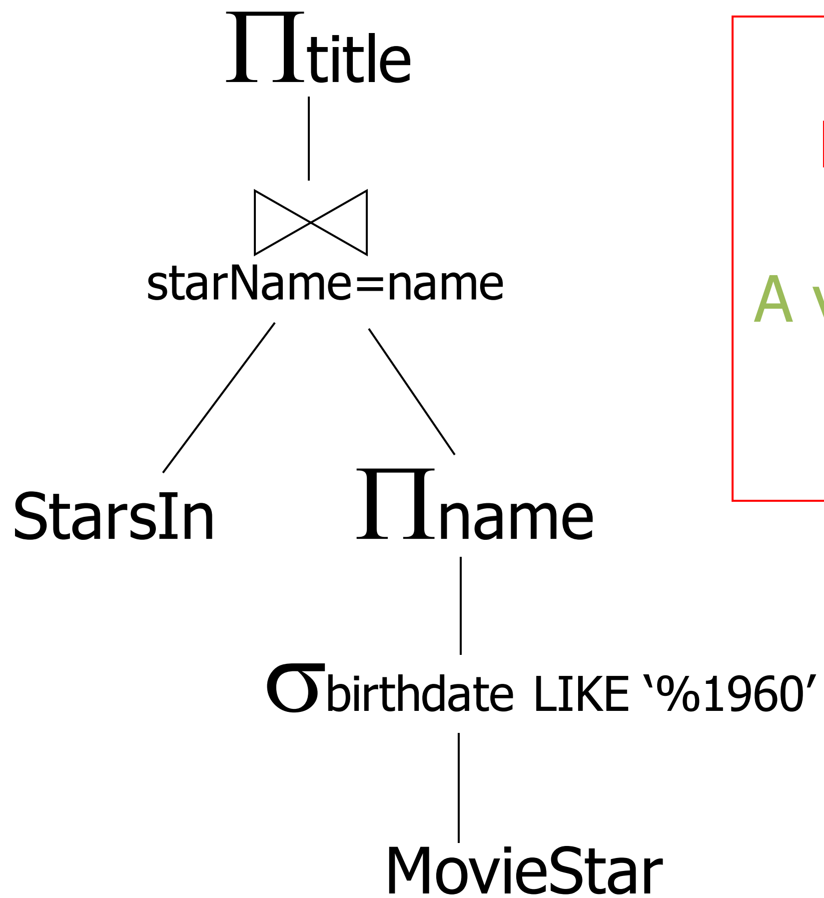
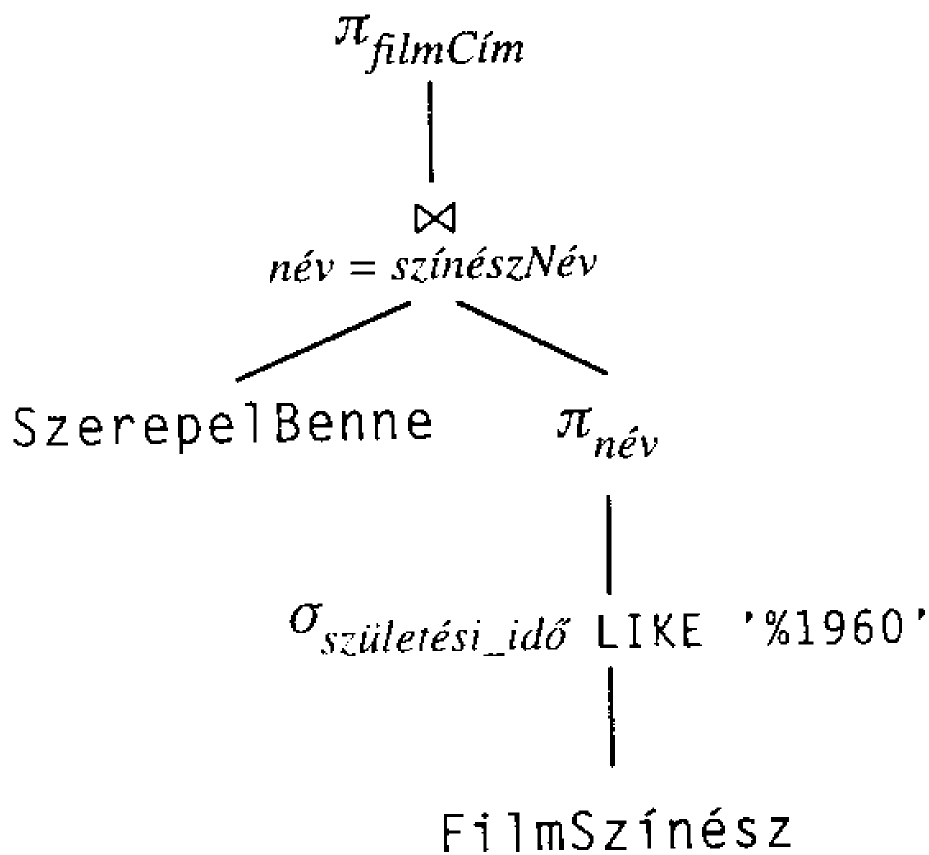


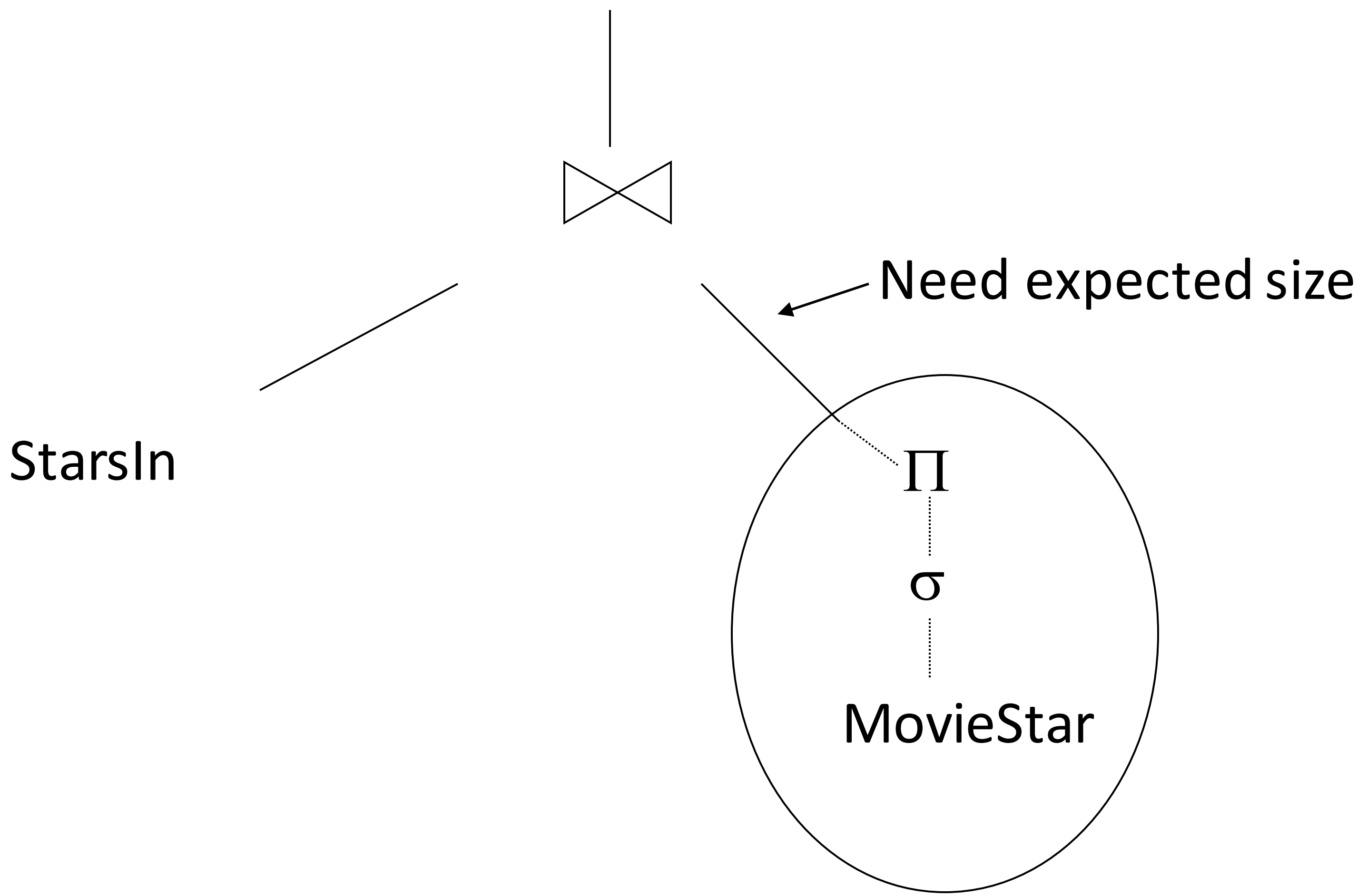
Fig. 7.22: An improvement on fig. 7.16.

Question:
Push project to StarsIn?
A vetítést a fa jobb ágába visszük

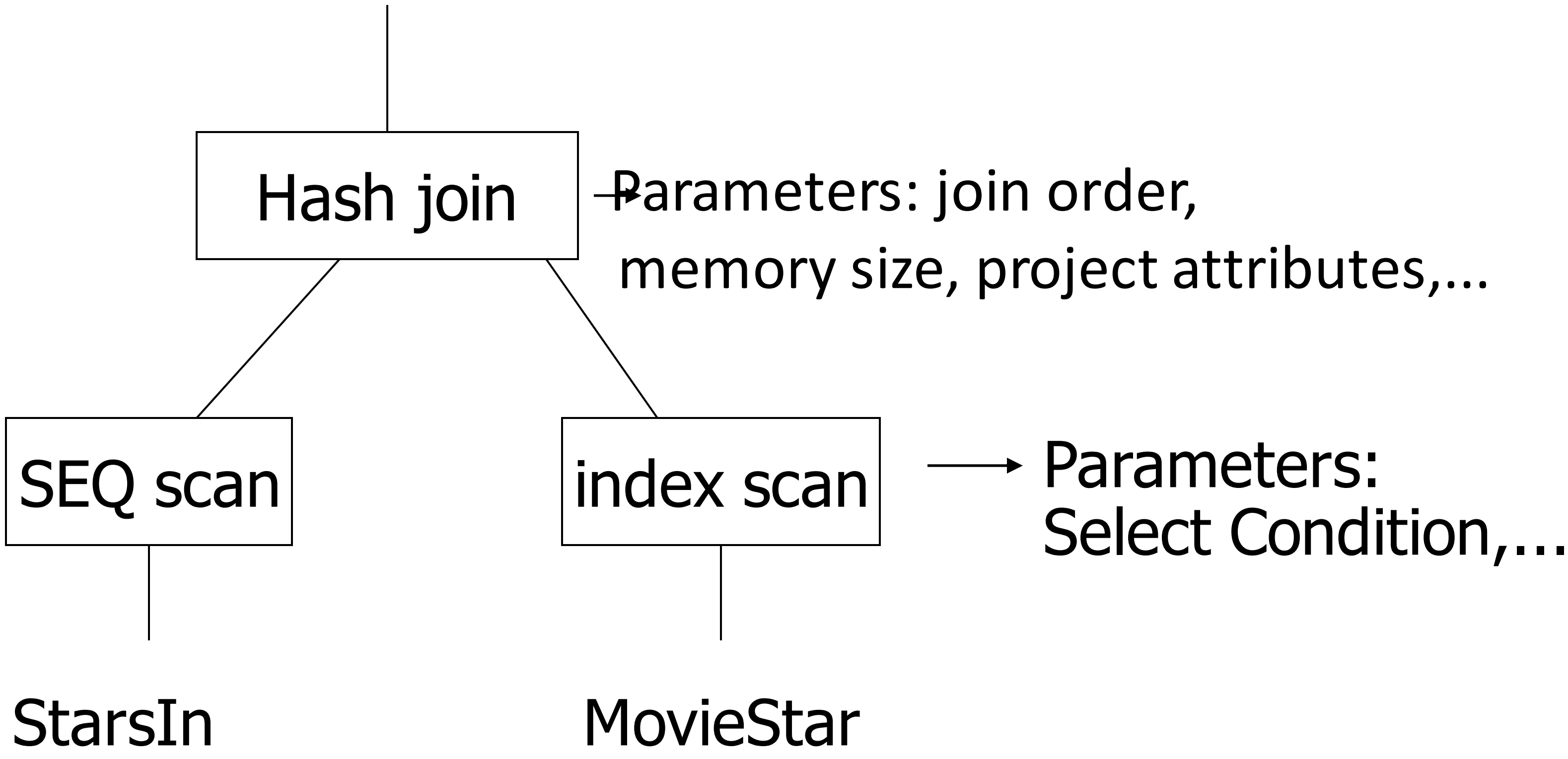


$\pi_{filmCím}$
$\bowtie_{név = színészNév}$
SzerepelBenne
$\pi_{név}$
$\sigma_{születési\_idő}$ LIKE '%1960'
FilmSzínész

Example: Estimate Result Sizes



Example: One Physical Plan



Költség alapú optimalizálás

- A kiválasztott logikai lekérdezéstervet fizikai lekérdezéstervvé alakítjuk át.
- Különböző fizikai lekérdezéstervek közt válogathatunk.
- A legjobbat a költségek becslésével választjuk ki.
- Egy-egy fizikai terv magában foglalja:
  - sorrend és csoportosítás megadását asszociatív és kommutatív műveletek esetén,
  - a megfelelő algoritmus kiválasztását minden egyes operátorhoz.
  - további műveletek megadását, amelyek a logikai tervben nem voltak jelen (beolvasás, rendezés, "extra" projekciók stb.),
  - annak módját, ahogy egy operátor továbbadja az argumentumokat egy másiknak.

A költség

- A legfőbb költségtényező továbbra is az I/O műveletek száma.
- A közbülső relációk általában nyalábolt an helyezkednek el, és ritka kivételektől eltekintve nem tartozik hozzájuk index.
- Emiatt a köztes relációk esetén az I/O műveletek száma csak a reláció méretétől függ.
- Egy-egy sor mérete jól becsülhető (átlag), vagy pontosan megadható.
- Így a legfőbb kérdés: a közbülső reláció hány sort tartalmaz.

Milyen becslések kellenek?

- Könnyen kiszámíthatóak.
- Elég pontos becslést adnak.
- Logikailag konzisztensek. Azaz: például az összekapcsolások esetén a becslés nem függ a relációk összekapcsolásának sorrendjétől.

- A vetítések mérete sok esetben pontosan, vagy majdnem pontosan megadható, hiszen a sorok számát ismerjük, csak az egyes sorok méretét kell becsülnünk.
- Ugyanakkor előfordulhat, hogy a vetítés során a sorok mérete nő. (attribútumok kombinációja, új komponensek)

Kiválasztás méretének becslése I.

- $S = \sigma_{A=c}(R)$ esetén:

$$T(S) = T(R)/V(R,A)$$

ez Zipfian-eloszlás esetén is jó, ha a kiválasztás feltételében a konstanst véletlenszerűen választjuk. (400. o., 7.4.3)

- $S = \sigma_{A<c}(R)$ esetén:

$$T(S) = T(R)/3.$$

- $S = \sigma_{A\neq a}(R)$ esetén: $T(S) = T(R)$, vagy:

$$T(S) = T(R)(V(R,a) - 1)/V(R,a).$$

Kiválasztás méretének becslése II.

- $S = \sigma_{C1 \wedge C2}(R)$ esetén a szelektivitási tényezőkkel szorzunk.
- Az előbbiek értelmében a szelektivitási tényező: $\neq$ esetén 1, $=c$ esetén $1/V(R,A)$, $<$ esetén $1/3$.

- Példa: $S = \sigma_{A=10 \wedge B<20}(R)$ esetén hány sora lesz S-nek a becslésünk szerint?

  $T(R) = 10\,000$, $V(R,A) = 50$.
- $T(R)/(V(R,A)\times 3)$, $[T(R)/(V(R,A)]\times 1/3)$,

- És $S = \sigma_{A=10 \wedge A>20}(R)$ esetén?

Kiválasztás méretének becslése III.

- $S = \sigma_{C1 \vee C2}(R)$ esetén tegyük fel, hogy R n sorából m_1 sor teljesíti C_1-t és m_2 sor C_2-t. Ekkor a

$T(S) = n(1 – (1 – m_1/n)(1 – m_2/n))$.

$S = \sigma_{A=10} \vee_{B<20}(R)$

$T(R) = 10\,000$, $V(R,A) = 50$.

$T(R)/ V(R,A)$, $T(R) \times 1/3$

$T(S) = n(1 – (1 – m_1/n)(1 – m_2/n))$.

$m_1 = T(R)/ V(R,A)$, $m_2 = T(R) \times 1/3$

Összekapcsolások méretének becslése I.

- Théta-összekapcsolás esetén az $R \bowtie_C S \equiv \sigma_C (R \times S)$ szabály használható.
- Equi-join esetén a természetes összekapcsolásra vonatkozó becslések alkalmazhatóak.
- A természetes összekapcsolás méretének becsléséhez tegyük fel, hogy R(X, Y), S(Y, Z) relációknál az Y egyetlen attribútumot tartalmaz.
- Ekkor $T(R \bowtie S)$ lehet:
  - 0, ha nincs közös Y értéke a két reláció Y attribútumának.
  - T(R), ha Y S kulcsa, R-nek pedig idegen kulcsa.
  - T(R)T(S), *majdnem* az összes sor ugyanazokat az Y értéket tartalmazza R-ben és S-ben is.

Összekapcsolások méretének becslése II.

- Két egyszerűsítő feltételezés, amelyek azonban sok esetben teljesülnek:
  1. értékhalmazok tartalmazása: *Y* egy attribútum, több relációban is előfordul, mindig egy rögzített $y_1, \ldots, y_k$ lista elejéről kap értéket. Így: $V(R, Y) \leq V(S, Y)$ esetén, R minden *Y* értéke S-nek is értéke.
  2. Értékhalmazok megőrzése: ha *A* nem összekapcsolási attribútum, akkor:

     $V(R \bowtie S, A) = V(R, A)$.
- Mindkét feltétel teljesül például (1;2), ha *Y* S-ben kulcs, R-ben idegen kulcs.
- A második feltétel (2.) csak akkor nem teljesülhet, ha R-ben vannak lógó sorok, de még ekkor is fennállhat ez a feltétel.

Összekapcsolások méretének becslése II.

- Tegyük fel, hogy $V(R, Y) \leq V(S, Y)$.
- Rögzítsük R egy r sorát.
- Ekkor $1/V(S, Y)$ az esélye annak, hogy $r \in R$, $s \in S$ egy sorához kapcsolódik.
- A kapcsolódó sorok várható száma tehát: $T(S)/V(S, Y)$.
- Így az összekapcsolás mérete:

$$\frac{T(R)T(S)}{\max(V(R,Y),V(S,Y))}$$

Összekapcsolások méretének becslése III.

- Hasonló gondolatmenettel belátható, hogy
  $R(X, Y_1, Y_2) \bowtie S(Y_1, Y_2, Z)$ esetén az összekapcsolás mérete:

$V(S, Y_1) \leq V(R, Y_1)$, értékhalmazok tartalmazása

$s \in S$, $r \in R$, 1/ $V(R, y_1)$ az esély (val.szeg.), s és r , Y_1, értéke u.a.,

$$\frac{T(R)T(S)}{\max(V(R,Y_1),V(S,Y_1))\max(V(R,Y_2),V(S,Y_2))}.$$

Összekapcsolások méretének becslése IV.

- $S = R_1 \bowtie \ldots \bowtie R_n$ esetén tegyük fel, hogy az A attribútum $R_1, \ldots, R_k$-ban fordul elő.
- Tegyük fel továbbá, hogy: $V(R_1, A) \leq \ldots \leq V(R_k, A)$.
- Válasszunk ki $R_1, \ldots, R_k$-ból egy-egy sort.
- Ekkor a t_i sor $1/V(R_i, A)$ valószínűséggel egyezik meg t_1-gyel.
- Emiatt T(S) becsléséhez az egyes relációkban szereplő sorok számának szorzatát, minden egyes A attribútum esetén, ami legalább két relációban előfordul, a legkisebb kivételével osszuk el az összes V(R, A) értékkel.

Példa

R (A, B, C)	S (B, C, D)	U (B, E)
T(R) = 1000	T(S) = 2000	T(U) = 5000
V(R, A) = 100	V(S, B) = 50	V(U, B) = 200
V(R, B) = 20	V(S, C) = 100	V(U, E) = 500
V(R, C) = 200	V(S, D) = 400	

Mekkora lesz $R \bowtie S \bowtie U$ mérete?

Halmazműveletek

- Multihalmaz-egyesítés esetén a két reláció méretének összege adja az eredmény méretét.
- Halmazegyesítésnél pedig: T(R) + ((T(R) + T(S)) /2)-vel becsülhetünk (T(R) > T(S)).
- Multihalmaz-metszet a természetes összekapcsolás egy speciális esete, tehát az ott adott becslés itt is használható.
- Vagy: a kisebbik reláció sorszámának a fele.

$$\frac{T(R)T(S)}{\max(V(R,Y),V(S,Y))}$$

A	B
0	1
0	1

A	B
0	1
0	1
0	1

A multihalmaz-metszet méretének becslésére itt milyen értékeket kapunk?

Ismétlődések kiküszöbölése és csoportosítás

- $\delta(R\ (A_1, \ldots, A_n))$ mérete pontosan $V(R, [A_1, \ldots, A_n])$, ez az érték azonban sokszor nem áll rendelkezésünkre.
- Egy felső korlát a méretre a $V(R, A_i)$ értékek szorzata. Jelöljük ezt V-vel.
- Elfogadható becslés: $T(R)/2$ és V közül a kisebbik.
- $\gamma_L\ (R)$ esetén a méret pontosan $V(R, [B_1, \ldots, B_k])$, ha L-ben a $B_1, \ldots, B_k$ csoportosító attribútumok szerepelnek.
- Itt is felső korlát a csoportok számára a $V(R, B_i)$ értékek szorzata. Legyen ez W.
- Szintén elfogadható becslés, ha $T(R)/2$ és W közül a kisebbiket választjuk.

Hisztogrammok

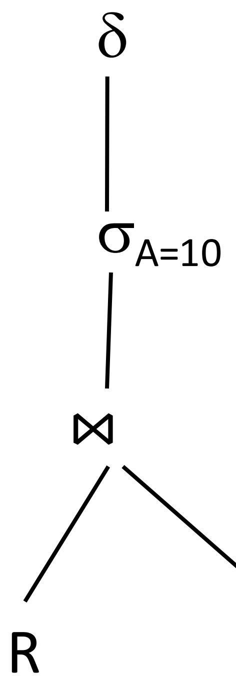
- Az R(A, B) ⋈ S(B, C) méretére szeretnénk becslést adni, V(R, B) = 14 és V(S, B) = 13.
- Rendelkezésünkre állnak a következő hisztogrammok:
  - R.B: 0: 150, 1: 200, *5*: 100, egyéb: 550 ~ 1000 sor
  - R.S: 0: 100, 1: 80, *2*: 70, egyéb: 250. ~ 500 sor
- R.B egyéb értékeinél átlagosan 50 sor egy értékre, S.B esetén 25. ([550/(V(R, B)-3)]), ([250/(V(S, B) -3)])
- A 0 és 1 értékű sorok összesen: 150 x 100 + 200 x 80 = 31000.
- Élünk a korábbi feltételezéssel, miszerint S minden B értéke R B attribútumában is előfordul.
- A *2* és *5* értékű sorok összesen: 50 x 70 + 100 x 25 = 6000.
- A maradék sorok: *9* x (50 x 25) = 11250. *V(R, B) -4-1*; *V(S, B) -4*
- Összesen: 31000 + 6000 + 11250 = 48250.
- 1000×500/14=35714

$$\frac{T(R)T(S)}{\max\left(V(R,Y),V(S,Y)\right)}$$

Statisztikák növekményes kiszámítása

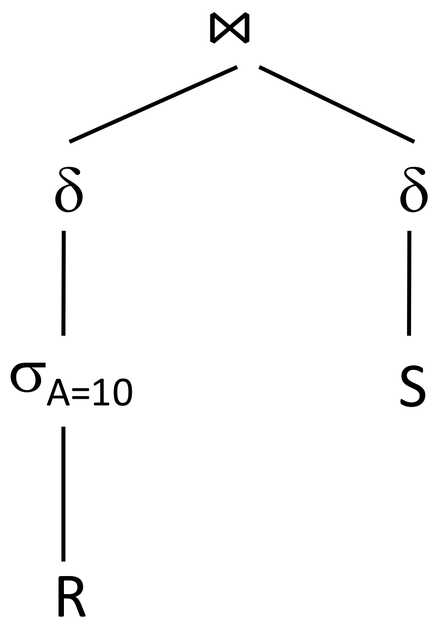
- A lekérdezés-optimalizáció a statisztikákat bizonyos időközönként frissíti, mert ezek nem szoktak rövid idő alatt radikálisan megváltozni.
- Ennek egy alternatívája a növekményes kiértékelés (incremental evaluation), azaz a rendszer minden táblamódosítás után napra készen tartja az érintett statisztikákat.
  - Törlésnél, beszúrásnál T(R)-ből kivonunk vagy hozzáadunk egyet.
  - Ha van B-fa indexünk R egy attribútumára, akkor a levelek száma is használható T(R) becslésére. Feltesszük, hogy minden blokk ¾ részéig van tele, s becsüljük, hogy ekkora helyen hány kulcs-mutató pár fér el. Ebben az esetben csak akkor kell módosítani a becslést, ha a levelek száma változik.
  - Az A attribútumra vonatkozó index segítségével a V(R, A) is könnyen karbantartható.
  - Ha az A kulcs, akkor V(R, A) = T(R).

Példa költségek becslésére I.



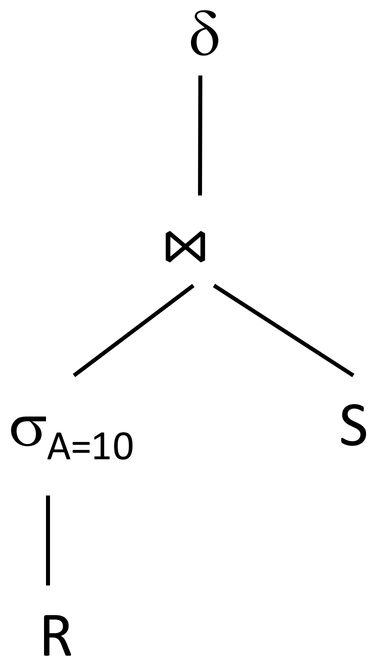
R (A, B)	S (B, C)
T(R) = 5000	T(S) = 2000
V(R, A) = 50	V(S, B) = 200
V(R, B) = 100	V(S, C) = 100

Példa költségek becslésére II.

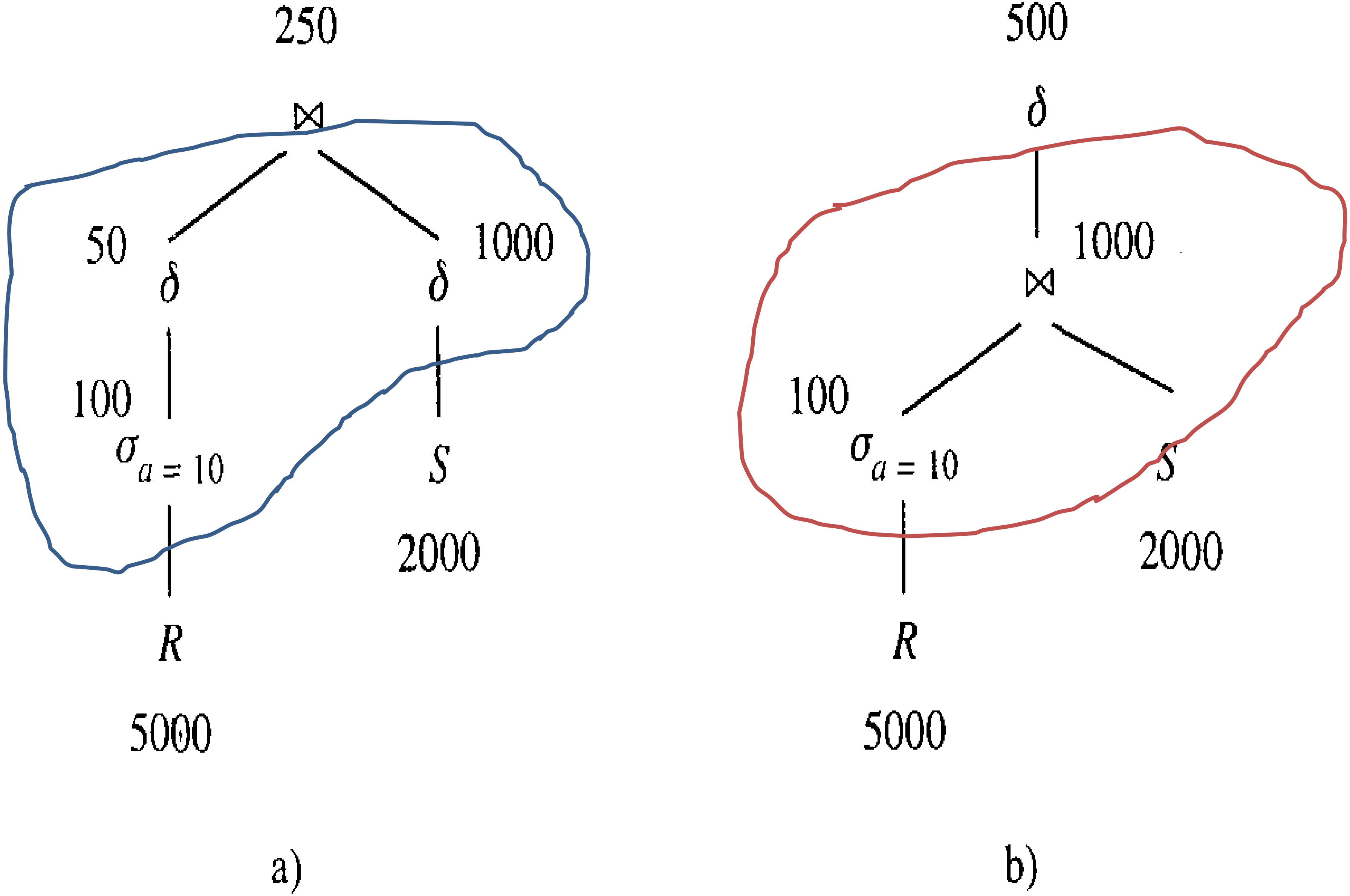


Mi a terv végrehajtási költségének a becslése?

Példa költségek becslésére III.



És itt?

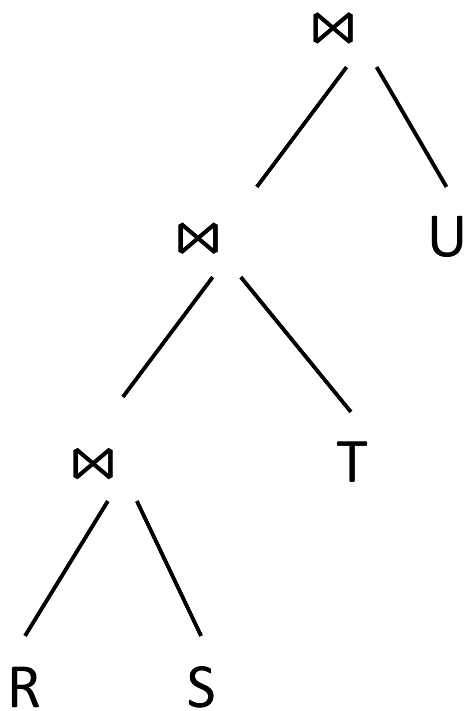


7.27. ábra. *Két jelölt a legjobb logikai lekérdezéstervre*

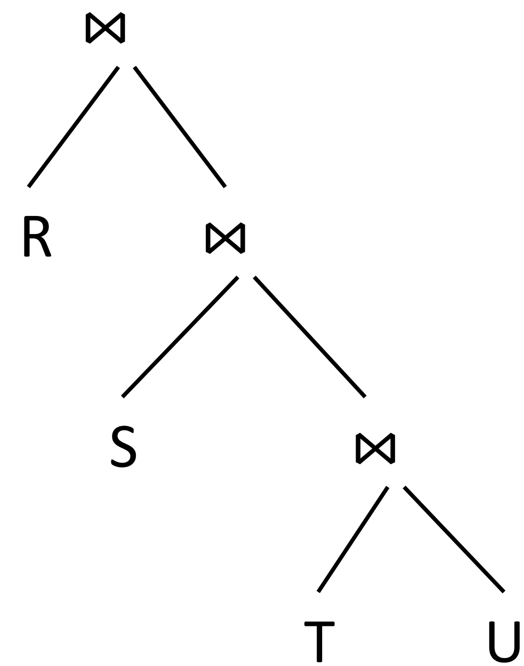
Összekapcsolások sorrendje

- Már két reláció összekapcsolásánál is sok esetben számít, hogy melyik reláció áll a baloldalon és melyik a jobbon.
- Egymenetes összekapcsolásnál feltételezzük, hogy a kisebbik relációt olvassuk be a memóriába teljesen.
- Beágyazott ciklusú összekapcsolásnál szintén a kisebbik reláció szerepel a baloldalon, ami külső ciklus relációja.
- Indexes összekapcsolás esetén pedig a jobb argumentum rendelkezik indexszel.

Összekapcsolási fák



bal-mély összekapcsolási fa



jobb-mély összekapcsolási fa

Miért a bal-mély összekapcsolási fák?

- Sok esetben csak (vagy majdnem csak) a bal-mély összekapcsolási fákat veszik figyelembe az összekapcsolások sorrendjének meghatározásánál.
- Egyrészt így nagyban csökken a lehetőségek száma.
- Másrészt a bal-mély fák jól együttműködnek a szokásos összekapcsolási algoritmusokkal: különösen az egymenetes és a beágyazott ciklusú összekapcsolásokkal.

Összekapcsolási sorrendek száma

- Jelölje T(n) az n relációhoz tartozó faformák számát. Ekkor:

$$T(1) = 1, T(n) = \sum_{i=1}^{n-1} T(i)T(n-i).$$

- T(2) = 1, T(3) = 2, T(4) = 5, T(5) = 14, T(6) = 42.
- Azaz hat reláció összekapcsolásakor $42 \times 6! = 30240$ lehetőséget kellene figyelembe venni. 6! A relációk összes lehetséges sorrendje
- A bal-mély fák esetén ezzel szemben csupán 6! = 720 lehetőség kínálkozik.

Egymenetes összekapcsolás bal-mély fa

- Tegyük fel, hogy $R \bowtie S \bowtie T \bowtie U$-t szeretnénk kiszámítani, s ezt megtehetjük egymenetes összekapcsolásokkal.
- $R \bowtie S$ kiszámításához R relációt kell a memóriában tartani, majd $R \bowtie S$-et is. Tehát $B(R) + B(R \bowtie S)$ memóriapufferre van szükség.
- $(R \bowtie S) \bowtie T$ kiszámításánál az R által elfoglalt memóripufferekre már nincs szükség.
- Hasonlóképpen az U-val való összekapcsolás során az $R \bowtie S$ memóriapufferei szabadíthatók fel.
- Így a szükséges helymennyiség:

  $B(R) + B(R \bowtie S)$ illetve $B(R \bowtie S) + B((R \bowtie S) \bowtie T)$.

Egymenetes összekapcsolás jobb-mély fa

- Itt elsőként az R relációt kell betölteni a központi memóriába, majd létre kell hozni az $S \bowtie (T \bowtie U)$ relációt.
- Ehhez S-et szintén be kell olvasni a memóriába, $T \bowtie U$ kiszámításánál pedig U-t.
- Összesen tehát $B(R) + B(S) + B(T)$ helyre van szükség.
- Mivel sok esetben legalább az egyik reláció kisebb, mert tartalmaz egy szűrést, emiatt $B(R) + B(S) + B(T)$ általában nagyobb, mint $B(R) + B(R \bowtie S)$ vagy

  $B(R \bowtie S) + B((R \bowtie S) \bowtie T)$.
- Emiatt a bal-mély összekapcsolási fák használata általában előnyösebb ebben az esetben.

Beágyazott ciklusú összekapcsolás bal-mély fa

- Tegyük fel ismét, hogy R ⋈ S⋈ T ⋈ U-t szeretnénk kiszámítani. Ld.
- Tegyük fel még, hogy R, S, T, U tárolt relációk.
- Ekkor R ⋈ S⋈ T egy megfelelő darabját tartjuk a memóriába, majd a maradék területen U blokkjait vesszük sorra.
- U-t itt azonban nem kell előállítani, csupán be kell olvasni.
- R ⋈ S⋈ T ezen darabjának kiszámításához, R ⋈ S egy darabját kell előállítani, majd T blokkjait kell beolvasni, viszont T-t nem kell külön előállítani.
- Végül R ⋈ S szóban forgó darabjának előállításához R-nek egy megfelelő darabját kell beolvasni és S blokkjait kell végigolvasni.
- Összességében: köztes eredményt sosem kell eltárolni többszöri beolvasás céljából.

Beágyazott ciklusú összekapcsolás jobb-mély fa

- Ezzel szemben jobb-mély fák esetén R egy megfelelő darabját olvassuk be a memóriába, majd meg kell konstruálnunk $S \bowtie T \bowtie U$-t.
- Amikor R következő darabját olvassuk be, az iménti relációt újfent elő kell állítani.
- Stb.

Dinamikus programozás I.

- A dinamikus programozás mögött az az alapötlet áll, hogy kitöltünk egy táblázatot a részfeladatok megoldásának költségeiről úgy, hogy a költség mellett csak a megoldás kivitelezéséhez szükséges minimális információt őrizzük meg.
- $R_1, \ldots, R_n$ összekapcsolása esetén ebben a táblázatban a következők szerepelnek az egyes $R_{i1}, \ldots, R_{ik}$ részhalmazokhoz:
  - $R_{i1} \bowtie \ldots \bowtie R_{ik}$ becsült mérete.
  - Az összekapcsoláshoz szükséges legkisebb költség. Először a legegyszerűbb becslést használjuk majd, azaz a köztes relációk mérete adja majd a költséget.
  - Az a kifejezés, ami ezt a legkisebb költséget adja.

Dinamikus programozás II.

- Egyetlen R reláció esetén a táblázat egy bejegyzése (1) R méretét, (2)0 költséget és (3) R-et „leíró relációs algebrai” kifejezést.
- $\{R_i, R_j\}$ pár esetén a méretre a korábbi becslést használjuk. A költség (azaz közbülső relációk mérete) 0, baloldalon a kisebbik méretű reláció szerepel , a formula $R_i \bowtie R_j$.

$$\frac{T(R)T(S)}{\max\left(V(R,Y),V(S,Y)\right)}$$

- Indukciós lépés $\{R_1, \ldots, R_k\}$:
  - bal-mély fák esetén vesszük azt a $\{R_{i1}, \ldots, R_{ik-1}\}$ bejegyzést, amire a költség + méret érték a legkisebb. Legyen az ehhez tartozó kifejezés P. Ekkor az új bejegyzés $P \bowtie R$ lesz, ahol R jelöli a kimaradt relációt. A költséget az $\{R_{i1}, \ldots, R_{ik-1}\}$ bejegyzéshez tartozó költség + méret összeg adja. A méret a szokásos módon becsülhető.
  - Általános esetben P_1, P_2 kételemű partícióit vesszük $\{R_1, \ldots, R_k\}$-nek. Keressük azt a partíciót, amire P_1, P_2 költségeinek és méreteinek összege minimális.

Példa

R (A, B)	S (B, C)	T (C, D)	U (D, A)
V(R, A) = 100			V(U, A) = 50
V(R, B) = 200	V(S, B) = 100		
	V(S, C) = 500	V(T, C) = 20	
		V(T, D) = 50	V(U, D) = 1000

T(R)= T(S)= T(T)= T(U) =1000

Első lépés

	{R}	{S}	{T}	{U}
méret	1000	1000	1000	1000
költség	0	0	0	0
kifejezés	R	S	T	U

Második lépés

	{R, S}	{R, T}	{R, U}	{S, T}	{S, U}	{T, U}
méret	5000	1M	10000	2000	1M	1000
költség	0	0	0	0	0	0
kifejezés	$R \bowtie S$	$R \bowtie T$	$R \bowtie U$	$S \bowtie T$	$S \bowtie U$	$T \bowtie U$

Itt 1M 1 milliót jelöl.

Harmadik lépés

	{R, S, T}	{R, S, U}	{R, T, U}	{S, T, U}
méret	10000	50000	10000	2000
költség	2000	5000	1000	1000
kifejezés	$(S \bowtie T) \bowtie R$	$(R \bowtie S) \bowtie U$	$(T \bowtie U) \bowtie R$	$(T \bowtie U) \bowtie S$

Költség : a lehetséges közbülső relációk [az elsőként kiválasztott két reláció összekapcsolásának] a mérete.

Utolsó lépés

Csoportosítás	Költség
((S ⋈ T) ⋈ R) ⋈ U	12000
((R ⋈ S) ⋈ U) ⋈ T	55000
((T ⋈ U) ⋈ R) ⋈ S	11000
((T ⋈ U) ⋈ S) ⋈ T	3000
(T ⋈ U) ⋈ (R ⋈ S)	6000
(R ⋈ T) ⋈ (S ⋈ U)	2000000
(S ⋈ T) ⋈ (R ⋈ U)	12000

Pontosabb költségfüggvények

- Költségként eddig a közbülső relációk méretét használtuk, ez azonban sok esetben túlságosan durva becslést ad.
- Példa: R(A, B), S(B, C) összekapcsolása, ahol R egyetlen sorból áll.
- Ekkor $R \bowtie S$ kiszámítása messze nem ugyanolyan költségű akkor, ha S B attribútumára létezik egy indexünk, vagy nem létezik.
- Az általunk alkalmazott költségbecslés azonban ezt a különbséget nem veszi figyelembe.

Mohó algoritmus

- Ha az összekapcsolandó relációk száma nagy, a dinamikus programozás túlságosan is lassúvá válhat (hiszen az algoritmus exponenciális lépésszámú).
- Ekkor mohó algoritmust használhatunk, ahol soha nem lépünk vissza, vagy vizsgáljuk felül az egyszer már meghozott döntéseket.
- Mohó algoritmus bal-mély fák esetén:
  - kezdjük azzal a relációpárra, amelyek összekapcsolásának a mérete a legkisebb.
  - Indukció: a még nem szereplő relációk közül keressük meg azt, amelyiket az aktuális fával összekapcsolva a legkisebb becsült méretű relációt kapjuk. Az új aktuális fa bal argumentuma a régi aktuális fa, jobb argumentuma pedig a kiválasztott reláció lesz.
- Az előző példa esetén hogyan működik ez az algoritmus?

Mérföldkő

- Hol tartunk?
  - Az eredeti lekérdezésből elkészítettük a logikai lekérdezéstervet.
  - Ezt optimalizáltuk.
  - Felsoroltunk több fizikai/logikai lekérdezéstervet, megbecsültük a költségüket és kiválasztottuk a legjobbnak ígérkezőt.
  - Az asszociatív és kommutatív műveleteknek meghatároztuk egy végrehajtási sorrendjét.
- Mi hiányzik még?
  - Ha még nem tettük meg az egyes műveletekhez hozzá kell rendelni a műveletet megvalósító algoritmust.
  - El kell dönteni, hogy a köztes eredményeket materializáljuk vagy futószalagosítjuk.

Második lépés

	{R, S}	{R, T}	{R, U}	{S, T}	{S, U}	{T, U}
méret	5000	1M	10000	2000	1M	1000
költség	0	0	0	0	0	0
kifejezés	R ⋈ S	R ⋈ T	R ⋈ U	S ⋈ T	S ⋈ U	T ⋈ U

Itt 1M 1 milliót jelöl.

Harmadik lépés

	{R, S, T}	{R, S, U}	{R, T, U}	{S, T, U}
méret	10000	50000	10000	2000
költség	2000	5000	1000	1000
kifejezés	$(S \bowtie T) \bowtie R$	$(R \bowtie S) \bowtie U$	$(T \bowtie U) \bowtie R$	$(T \bowtie U) \bowtie S$

Költség : a lehetséges közbülső relációk [az elsőként kiválasztott két reláció összekapcsolásának] a mérete.

Utolsó lépés

Csoportosítás	Költség
((S ⋈ T) ⋈ R) ⋈ U	12000
((R ⋈ S) ⋈ U) ⋈ T	55000
((T ⋈ U) ⋈ R) ⋈ S	11000
((T ⋈ U) ⋈ S) ⋈ T	3000
(T ⋈ U) ⋈ (R ⋈ S)	6000
(R ⋈ T) ⋈ (S ⋈ U)	2000000
(S ⋈ T) ⋈ (R ⋈ U)	12000

Kiválasztási eljárás megvalósítása

- Az R(A, B, C) reláció fölött tekintsük a $\sigma_{A=1 \wedge B=2 \wedge C<5}(R)$ kiválasztást, ahol T(R) = 5000, B(R) = 200, V(R, A) = 100, V(R, B) = 500. Tegyük fel még, hogy R nyalábolt, az A, B, C attribútumokra létezik index, ezek közül a C-hez tartozó nyalábolt.
- Megvalósítás:
  1. táblaolvasás és közben szűrés: 200 I/O művelet, B(R), R nyalábolt.
  2. Használjuk az A-ra vonatkozó indexet és indexet olvasó operátort, majd szűrjük az így kapott sorokat a további feltételek szerint. Mivel az index nem nyalábolt, ez T(R) / V(R, A) = 50 I/O művelet.
  3. Ugyanez a B-re vonatkozó index esetében: T(R) / V(R, B) = 10 I/O művelet, nem nyalábolt index.
  4. Ugyanez a C-re vonatkozó index esetében: B(R) /3 = 67 I/O művelet, mivel itt az index nyalábolt.
- Az indexek olvasásának költségét itt nem számoltuk.

Az összekapcsolási eljárás megválasztása I.

- Ha nem vagyunk biztosak abban, mennyi memória áll rendelkezésre és néhány fontos statisztikát sem ismerünk, néhány alapelv még mindig a rendelkezésünkre áll.
- Ha valamelyik reláció várhatóan befér a memóriába, egymenetes algoritmust alkalmazunk, ha tévedünk, legfeljebb a beágyazott ciklusú megvalósítást használjuk bízva abban, hogy a külső ciklust nem kell sokszor lefuttatni.
- Rendezésen alapuló összekapcsolás akkor jó választás, ha
  - egyik vagy mindkét argumentum már rendezett a megfelelő attribútumokat tekintve,
  - később következik egy csoportosítás vagy rendezés művelet az összekapcsoló attribútumokra,
  - több relációt kapcsolunk össze „ugyanazon” attribútum alapján:

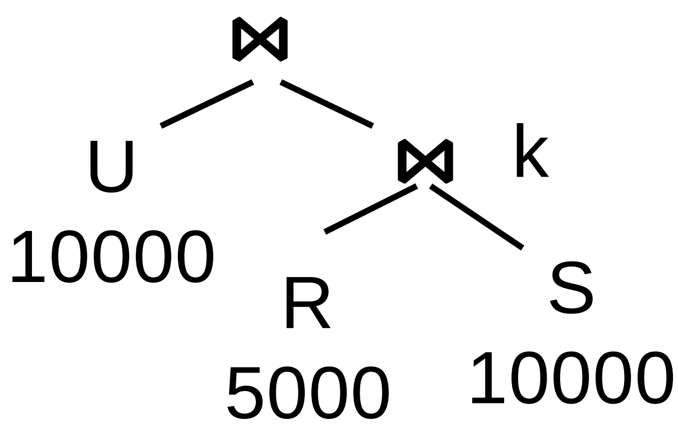
    $(S(A, B) \bowtie T(A, C)) \bowtie R(A, D)$.

Az összekapcsolási eljárás megválasztása II.

- Ha $S(A, B) \bowtie T(A, C)$ esetén S várhatóan kicsi – például, mert megelőzi egy S-re vonatkozó kiválasztás az összekapcsolást –, T pedig rendelkezik az A-ra vonatkozó indexszel, akkor az indexes összekapcsolást válasszuk.
- Ha többmenetes algoritmusra van szükség, nem áll rendelkezésünkre index az összekapcsoló attribútumokra, később nem tudjuk kihasználni az összekapcsoló attribútumok alapján való rendezettséget, akkor a tördeléses összekapcsolás általában a legjobb választás, hiszen itt a szükséges menetek száma csupán a kisebbik reláció méretétől függ.

Összekapcsolás futószalagosítása I.

- Az (R(w, x) ⋈ S(x, y)) ⋈ U(y, z) összekapcsolásokat szeretnénk megvalósítani, ahol a központi memória 101 pufferből áll, mindkét összekapcsolást tördelő algoritmussal valósítjuk meg. (*k* közbülső eredmény, által elfoglalt terület blokkban)



- R (kisebb) kosarai száma 100 korlátozzuk, így 50 blokkos kosarakra van szükség.
- R és S összekapcsolásához 51 puffer marad.(Második menet)
- Ha $k < 50$, R⋈S-t bent tartjuk a memóriában, a maradék puffer segítségével összekapcsoljuk U-val.
- R⋈ S 49 blokkot foglal, és futószalagosítható, 1 blokk U-ra
- 45e+10e=55000 I/O művelet összesen.

3(B(R)+B(S)) összekapcsolás tördeléssel

Összekapcsolás futószalagosítása II.

- Ha $k > 49$, de $k < 5001$, akkor R és S összekapcsolásának eredményét még mindig futószalagosíthatjuk. Az összekapcsolás során szabadon maradó 50 puffert arra használjuk, hogy az eredményt 50 kosárba tördeljük. A kosarakat a háttértárolón tároljuk.
- U-t is 50 kosárba tördeljük. $R \bowtie S$ Két menetes tördeléses összekapcsolás (51 kosár)
- Mivel R és S összekapcsoltjának kosarai legfeljebb 100 blokkosak, 51 kosárban (ha a hasító-függvény egyenletesen osztja szét az értékeket), így ezeket az U kosaraival egy menetben összekapcsolhatjuk.
- A műveletigény összesen: $45000 + 30000 + 2k = 75000 + 2k$.
- Ha $k > 5000$, akkor a legjobb, ha R és S összekapcsoltját materializáljuk, azaz a háttértárolón ideiglenesen eltároljuk.
- A műveletigény ebben az esetben: $75000 + 4k$.

Az ehhez a futószalagosított összekapcsoláshoz tartozó lemez I/O-műveletek száma:

a) az U beolvasásához és sorainak kosarakba történő írásához: 20 000,
b) az $R \bowtie S$ kétmenetes tördelő összekapcsolás végrehajtásához: 45 000,
c) az $R \bowtie S$ kosarainak kiírásához: k.
d) az $R \bowtie S$ és az U kosarainak beolvasásához a végső összekapcsolásban: $k + 10\,000$.

75 000+2k

K 49 → 50, egy menetesről kétmenetesre