Adatbázisok elmélete

Fizikai szervezés, tárkezelés, lekérdezések optimalizálása

Katona Gyula Y.

Számítástudományi és Információelméleti Tanszék
Budapesti Műszaki és Gazdaságtudományi Egyetem

2017. október 26.

Fizikai szervezés, tárkezelés

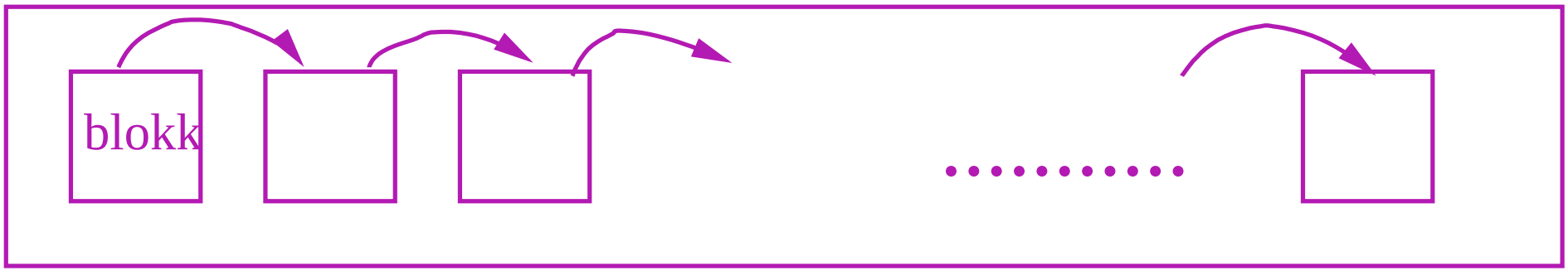
Célja: a rekordokból (egy rekord = a reláció egy sora) álló állomány kezelése úgy, hogy az adatokhoz való hozzáférés gyors legyen.

Fontos jellemzők:

- külső táras adatkezelés, mert sok az adat $\Longrightarrow$ ha valamivel dolgozni akarunk, akkor be kell hozni a belső memóriába $\Longrightarrow$ a költséget a beolvasás/kiírás jelenti $\Longrightarrow$ az I/O műveletek számára akarunk optimalizálni
- a műveletek, amiket gyorsan meg kell tudni csinálni: rekordok beillesztése, törlése, módosítása, keresése

Az állomány felépítése

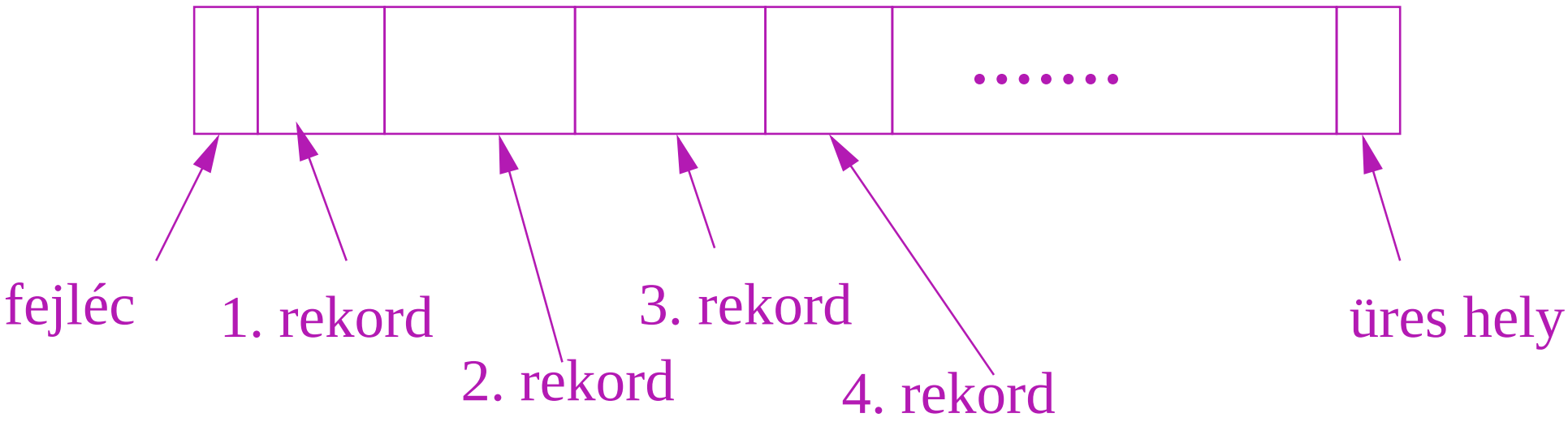
Az adatállomány a külső táron van, blokkok (lapok) elérésfolytonos sorozatán.



- egyszerre egy blokk írható ki/olvasható be
- blokk méret fix (ált. 2^{10}, 2^{12} byte)
- az operációs rendszer tartja nyilván, hogy melyik reláció rekordjai hol vannak és ő biztosítja az elérésfolytonosságot is

Blokkokról általában

Tipikus blokk



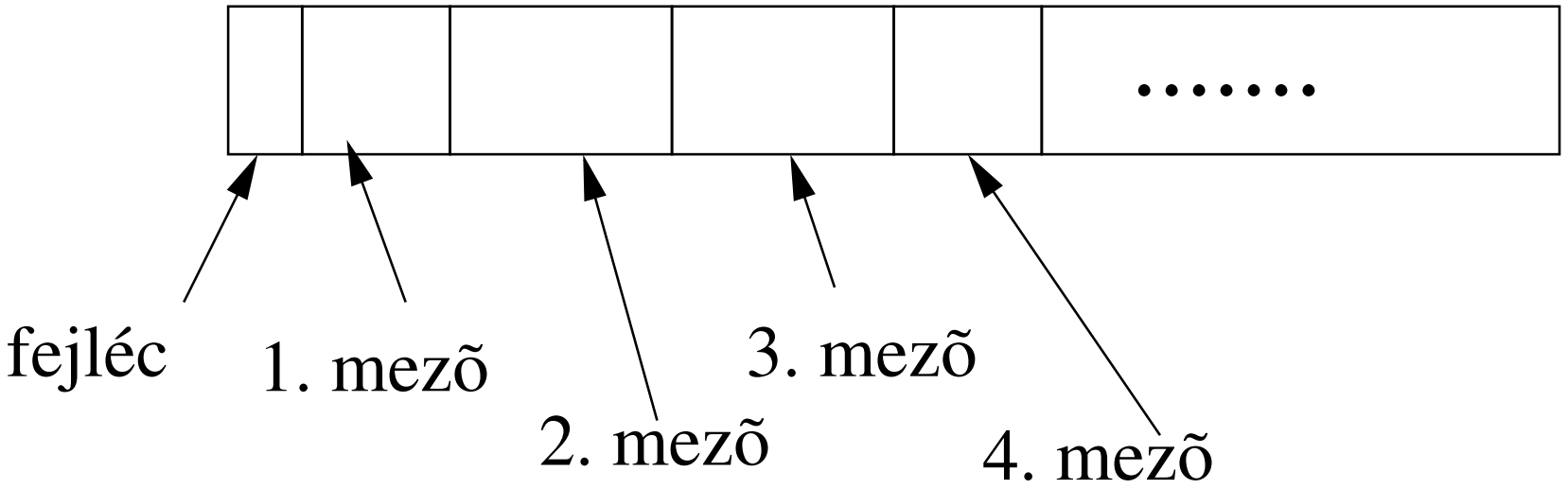
A fejléc tartalmazza a blokkra vonatkozó infókat, (pl: melyik relációhoz tartozik, mennyi a szabad hely benne, hol kezdődik); ezután jönnek a rekordok egymás után, a végén általában marad üres hely.

Fontos feltevés: rekordok blokkhatárt nem lépnek át, ezért általában van üres, fel nem használható hely a blokkok végén. (Ha nagyok a rekordok, pl. képfile-ok, és mégis át kell lépni laphatárt, akkor extra technikák kellenek, de ezzel most nem foglalkozunk.)

Rekordok típusai

Kötött formátum

Ekkor a mezők száma, mérete, típusa és sorrendje fix



Fejléc:

- a rekord kezelésével kapcsolatos infók: törölt-e, melyik relációhoz tartozik
- a mezők típusa
- időbélyeg (mikor módosult utoljára)

Rekordok típusai

Változó formátum

- Mezők hossza esetleg nem fix (szöveget tartalmazó adatbázisok)
- Ismétlődő mezők lehetnek, és az ismétlések száma nem fix vagy pedig többértékű mezők (szereplők felsorolása filmnél)

Ilyenkor bonyolultabb a fejléc, kevésbé lehet gazdaságosan/előre tervezhetően tárolni a rekordokat, ezért érjük el inkább, hogy ne legyen ez az eset:
Vezessük vissza ezt az esetet a kötöttre, pl. mutatók alkalmazásával a problémás helyeken $\Longrightarrow$ Mostantól feltesszük, hogy a rekordok kötött formátumúak és hogy az egész állományon belül ugyanaz a formátum van.

Fontos fogalmak

1. **mutató:** blokk vagy rekord címét tartalmazó bejegyzés
2. **kötött blokk/rekord:** mutathat rá mutató, ezért nem mozgatható el szabadon. Ez típusszinten adott, azaz ha egy reláció rekordjaira/blokkjaira mutathat mutató, akkor még akkor is kötöttnek számít, ha éppen nem mutat egyre se semmi.
3. **szabad blokk/rekord:** nem mutathat rá mutató
4. **Kulcs, keresési kulcs** (néha csak kulcsnak hívjuk):
   - a rekordok mezőinek egy kitüntetett halmaza (a reláció attribútumainak egy részhalmaza)
   - ez alapján megy a keresés (ezeknek az értékét adjuk meg és azokat a rekordokat (sorokat a relációban) keressük, amiknél pont ezek az értékek szerepelnek)
   - a keresési kulcs nem egyezik meg feltétlenül a reláció egyik kulcsával sem (pl. név a telefonkönyvnél)
   - de az azért elvárás, hogy ne legyen nagyon sok egy-egy értékre illeszkedő rekord

Alapvető állományszervezési technikák

Milyen struktúrát hozzunk létre az adatok tárolására?

Lehetőségek:

1. Szekvenciális tárolás $\Longrightarrow$ Keresés: $O(n)$, beszúrás: $O(1)$, törlés: $O(n)$
2. Indexek Hash táblával $\Longrightarrow$ Keresés: $O(n/M)$, beszúrás: $O(n/M)$, törlés: $O(n/M)$
3. Indexek B-fával $\Longrightarrow$ Keresés: $O(\log n)$, beszúrás: $O(\log n)$, törlés: $O(\log n)$

Lekérdezések végrehajtása, „optimalizálása”

Elemzés (parsing):

- szintaktikai ellenőrzés $\Longrightarrow$ *megfelelő parancsok, megfelelő sorrendben*
- átírás elemzőfa alakra

Előfeldolgozó:

- Relációk használatának ellenőrzése $\Longrightarrow$ *van-e ilyen*
- Attribútumnevek használatának ellenőrzése $\Longrightarrow$ *pl. egyértelmű-e, melyik attribútum melyik relációban van, benne van-e egyáltalán*
- típusellenőrzések $\Longrightarrow$ *pl. LIKE használatakor csak karakterlánc lehet*

Logikai lekérdezési terv:

- Átírás (kibővített) relációs algebrai alakra
- Transzformációk $\Longrightarrow$ *több terv, gyorsítás*
- Legjobb terv kiválasztása költségbecsléssel

Fizikai terv kiválasztása:

- Algoritmusok a műveletekhez
- Pufferkezelés
- Közbülső relációk eltárolása

A relációs algebra kibővítése

Az SQL többet tud, mint a relációs algebra, de az extra dolgokat is át akarjuk írni relációs formába. Néhány különbség:

- Multihalmazok $\Longrightarrow \cap_H, \cap_M$
- Kiválasztásnál, $\overset{\bowtie}{\theta}$-nál a feltételben használhatunk aritmetikai műveleteket $\Longrightarrow \sigma_{A+B<5}(R)$, $R \overset{\bowtie}{_{A+R.B<C+S.B}} S$
- Vetítés aritmetikai műveletekkel és átnevezéssel $\Longrightarrow \pi_{A,\, B+C \to X}(R)$
- Ismétlődések kiszűrése $\Longrightarrow \delta(R)$
- Csoportosítások, aggregátumok $\Longrightarrow$ SELECT A, $MIN(B)$ AS $minB$ FROM R GROUP BY A $\Longrightarrow \gamma_{A,MIN(B)\to minB}(R)$

Fizikai végrehajtás

Leginkább az I/O műveletigény érdekes. Ha „túl nagy” a számítási igény az is baj lehet.

Soronkénti, unáris műveletek: Kiválasztás és vetítés. Egyszerre csak egy sort kell vizsgálni, az algoritmus nem függ a memória nagyságától.

Unáris, teljes relációs műveletek: Pl. $\delta(R), \gamma(R)$. Ha nem fér el a reláció a memóriában, akkor mást kell csinálni.

Bináris, teljes relációs műveletek: $\cup, \cap, \setminus, \times, \bowtie$. Sok minden függ a méretektől.

Jelölés: Az adatokat a külső tárról blokkonként olvassuk be. Az R reláció tárolásához szükséges blokkok számát $B(R)$-rel jelöljük.
A belső memória mérete szintén blokkokban mérve legyen M.
$\sigma_C(R)$ **végrehajtása:** Blokkonként beolvassuk R-et. Soronként megnézzük teljesül-e C. Ha igen, kiírjuk.
I/O műveletigény: $B(R)$
Ha $\sigma_{A='c'}(R)$-t akarjuk, és van index A-ra: sokkal gyorsabb lehet.

Fizikai végrehajtás

$R(X, Y) \bowtie S(Y, Z)$ **végrehajtása:**

- *Ha* $B(S) < M - 1$*, azaz* S *belefér a memóriába:* egymenetes algoritmus
  1. Beolvassuk S-et és hashtáblát vagy B-fát készítünk, ahol a kulcs Y attribútumai.
  2. Beolvasunk egy blokkot R-ből. Minden sorára kikeressük a passzoló S-beli sorokat. Az eredményt kiírjuk.

  I/O műveletigény: $B(S) + B(R)$
- *Ha* $B(R) > B(S) > M - 1$*:* beágyazott ciklusú algoritmus
  Beolvasunk minél több blokkot a memóriába S-ből, utána ugyanazt csináljuk mint fenn.
  I/O műveletigény: $B(S) + B(S)B(R)/(M - 1) \approx B(S)B(R)/M$
- *Ha* $B(R), B(S) \leq M^2$*:* rendezéses algoritmus
  Y kulcs szerint rendezzük R-et és S-et összefésüléses rendezéssel. Vesszük az összes y kulcsú sort a két lista elejéről és kiírjuk az összes párt. (Feltettük, hogy az összes y kulcsú sor elfér a memóriában.)
  I/O műveletigény: $5(B(S) + B(R))$

Fizikai végrehajtás

- *Ha* $\min(B(R), B(S)) \leq M^2$*:* hasheléses algoritmus
  Y kulcs szerint vödrös hashelést végzünk R-re és S-re. (Ha közben megtelik egy vödör, azt kiírjuk.) A kapott R_i, S_i vödrökkel egymenetes algoritmust végzünk.
  I/O műveletigény: $3(B(S) + B(R))$
- *Ha van index S-re Y szerint:* indexet használó algoritmus
  R-et blokkonként olvassuk be, az index alapján keressük ki a hozzá passzoló sorokat.
  Átlagos I/O műveletigény: $B(S)B(R)/V(S, Y)$, ahol $V(S, Y)$: Y értékkészletének számossága S-ben.

A többi műveletet is hasonló ötletekkel lehet végrehajtani, azokat most nem részletezzük.

Optimalizálás

Triviális egyszerűsítések (főleg generált lekérdezések esetén hasznos):

- $r \cap r = r$; $r \bowtie r = r$; $r \cup \emptyset = r$; $\sigma_C(\emptyset) = \emptyset$; $\sigma_{false}(r) = \emptyset$
- $\pi_X(r \cup s) = \pi_X(r) \cup \pi_X(s)$
- $\sigma_{A=B \wedge B=C \wedge A=C}(r) = \sigma_{A=B \wedge B=C}(r)$

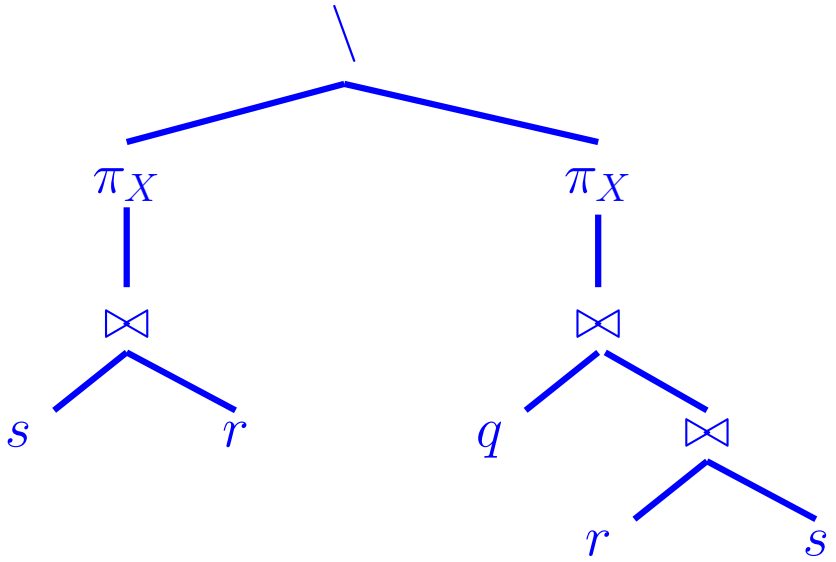
Nem teljesen trivi egyszerűsítések:

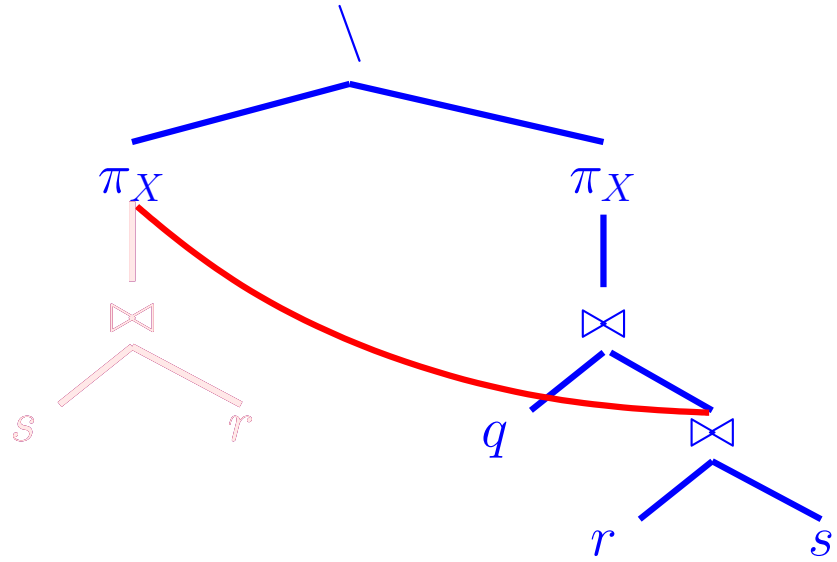
- $\sigma_{A=1}(\sigma_{B=2}(r)) = \sigma_{A=1 \wedge B=2}(r)$
- $\sigma_\theta(r \times s) = r \underset{\theta}{\bowtie} s$
- asszociativitás
- melyik indexet érdemes használni
- ha van index, akkor $2 \leq A \wedge A \leq 100$ helyett jobb A BETWEEN 2 AND 100

Optimalizálás

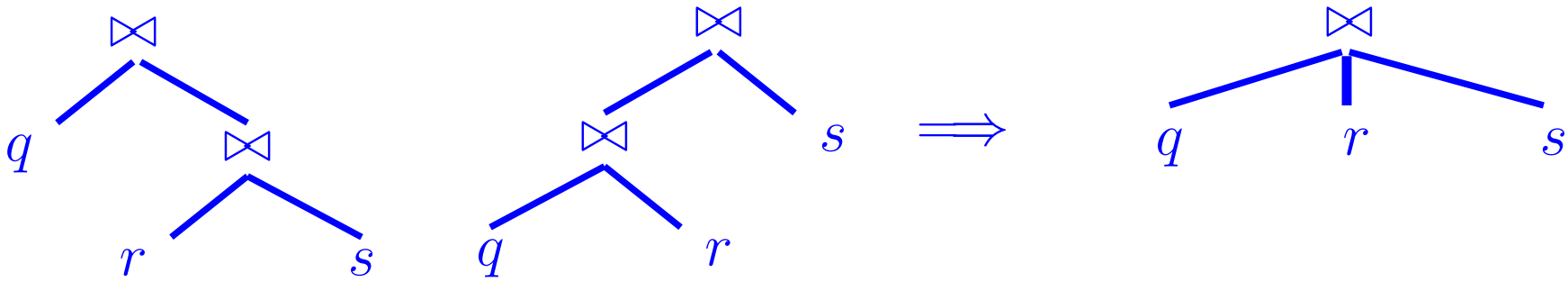
Ami többször előfordul, nem biztos, hogy érdemes mindig kiszámolni:

Pl. $\pi_X(s \bowtie r) \setminus \pi_X(q \bowtie r \bowtie s)$





Asszociativitás kihasználható:



Optimalizálás

Milyen sorrendben érdemes kiszámolni $r(A, B) \bowtie s(B, C) \bowtie q(C, D)$**-t?**
Szélsőséges esetben lehet, hogy bár r, s, q mindegyikének 1000 sora van, de $r \bowtie s$-nek csak 1 sora, és $s \bowtie q$-nak 1000000 sora.
$\Longrightarrow$ Sokkal gyorsabb $(r(A, B) \bowtie s(B, C)) \bowtie q(C, D)$ kiszámolása.
Ezt persze előre nem lehet tudni biztosan. $\Longrightarrow$ Statisztikákat vezetünk a relációk attribútumaiban előforduló értékekről
Ebből lehet becsülni a költségeket + dinamikus programozás vagy mohó algoritmus.

Igazi optimumot nehéz megtalálni:

Tétel

Annak eldöntése NP-teljes, hogy néhány reláció természetes illesztésének van-e legalább egy sora.

Tétel

Az optimum megtalálása NP-nehéz probléma.

Bizonyítás

Tétel

Annak eldöntése NP-nehéz, hogy néhány reláció természetes illesztésének van-e legalább egy sora.

Bizonyítás.

Visszavezetjük rá a 3-SZÍN problémát. Adott egy gráf, kérdés színezhető-e 3 színnel.
A gráf minden e éléhez vegyünk fel egy-egy relációt.
A reláció két attribútma az él két végpontja legyen, sorai pedig az összes lehetséges színpár. Például:

$e = \{X, Y\}$

X	Y
piros	kék
piros	sárga
kék	piros
kék	sárga
sárga	piros
sárga	kék

$e' = \{X, Z\}$

X	Z
piros	kék
piros	sárga
kék	piros
kék	sárga
sárga	piros
sárga	kék

Bizonyítás

Bizonyítás.

Ha az összes élhez tartozó reláció természetes illesztésének van sora $\Longrightarrow$ egy sor minden csúcshoz rendel egy színt. Mivel az illesztés megfelel az egyes relációknak, egy él két végpontján nem lesz ugyanolyan szín.
Ha van színezés $\Longrightarrow$ a színezésben minden élre vegyük ki a megfelelő színpárt. Ezek a sorok összeillenek, lesz sor a természetes illesztésben. $\surd$ $\square$

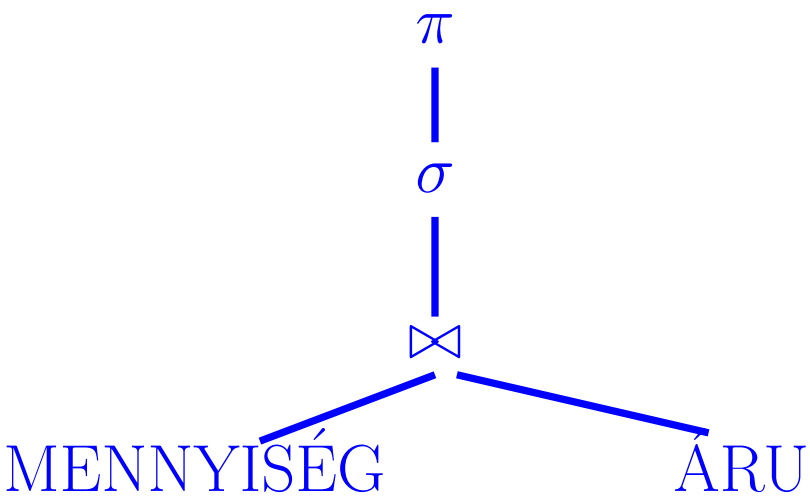
Kiválasztás tologatása

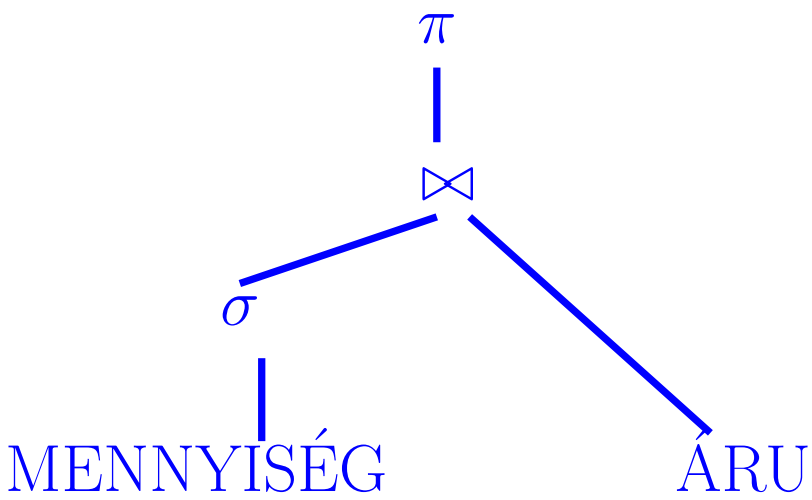
ÁRU(ÁRUKÓD, ÁRUNÉV, EGYSÉGÁR)
MENNYISÉG(DÁTUM, ÁRUKÓD, DB)

Hány darabot adtak el 2002. jan. 15-én az A123 kódú áruból, mi a neve és az ára?

$$\pi_{\text{DB, ÁRUNÉV, EGYSÉGÁR}}\left(\sigma_{\text{ÁRUKÓD='A123'} \wedge \text{DÁTUM='2002-01-15'}}\left(\text{MENNYISÉG} \bowtie \text{ÁRU}\right)\right) \Longrightarrow$$

$$\pi_{\text{DB, ÁRUNÉV, EGYSÉGÁR}}\left(\sigma_{\text{ÁRUKÓD='A123'} \wedge \text{DÁTUM='2002-01-15'}}\left(\text{MENNYISÉG}\right) \bowtie \text{ÁRU}\right)$$





Felhasznált azonosság:

$\sigma_C(R \bowtie S) = \sigma_C(R) \bowtie S$, ha minden C-beli attribútum szerepel R-ben.

Hasonló azonosságok:

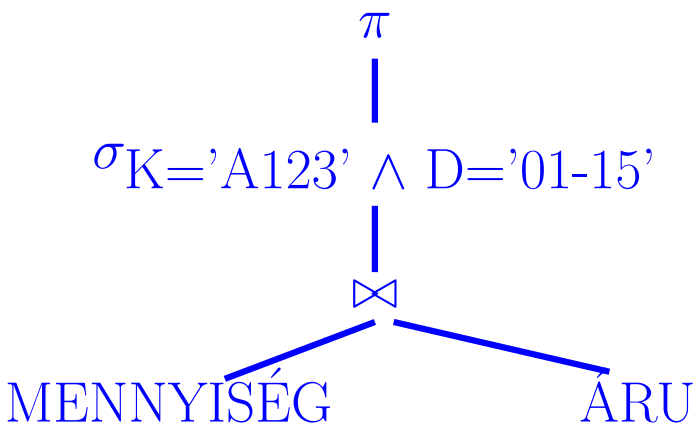
$\sigma_C(R \cup S) = \sigma_C(R) \cup \sigma_C(S)$,

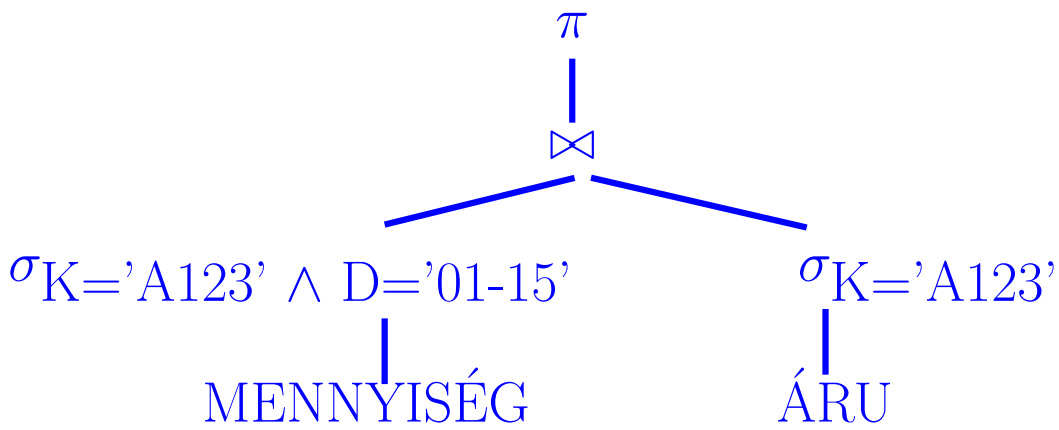
$\sigma_C(R \times S) = \sigma_C(R) \times S$, ha minden C-beli attribútum szerepel R-ben.

$\sigma_C(R \bowtie S) = \sigma_C(R) \bowtie \sigma_C(S)$, ha minden C-beli attribútum szerepel R-ben és S-ben is.

Kiválasztás tologatása

Összetett C szétszedhető:
$\sigma_{C_1 \wedge C_2}(R) = \sigma_{C_1}(\sigma_{C_2}(R))$,
$\sigma_{C_1 \vee C_2}(R) = \sigma_{C_1}(R) \cup_H \sigma_{C_2}(R)$, ha R nem multihalmaz.





Lehet, hogy érdemes előbb feltolni, aztán le.

Más műveletekre vonatkozó szabályok

Hasonló szabályok projekcióra (π), duplikációk kiszűrésére (δ) és aggregációra (γ) is vannak, de ezek nem annyira csökkentik a műveletigényt. De csak olyan attribútumot lehet eltüntetni, amire nem hivatkozunk feljebb.

- $\pi_L(R \bowtie S) = \pi_L(\pi_M(R) \bowtie \pi_N(R))$, ahol M az R olyan attribútumai, hogy vagy összekapcsolási attribútum, vagy L-beli, N pedig ...
- $\delta(R \bowtie S) = \delta(R) \bowtie \delta(S)$
- $\delta(\sigma_C(R)) = \sigma_C(\delta(R))$
- $\delta(\gamma_L(R)) = \gamma_L(R)$

De pl. δ nem tolható át $\cup_M, \pi$-n.
Még egy fontos kérdés az alkérdések kezelése, de erről most nem szólunk.

Összefoglalás

Sok ilyen szabály alkalmazásával többféle logikai terv előállítható.
Ezeknek megbecsüljük a költségét és választunk egyet. Ehhez készítünk fizikai tervet.
Jobb rendszerekben ez automatikus. Ilyenekben kérdéses, hogy a lekérdezés beírásakor mire kell figyelni.
Inkább olyan egyszerűsítéseket érdemes csak elvégezni, ami a függések következménye, mert ezeket nehezebben lehet automatizálni.
Általában van rá mód, hogy megnézzük mi a logikai és fizikai terv és meg lehet adni, hogy pontosan mit csináljon.