ELTE | IK

INFORMATIKAI KAR

# Adatbázisok 2

## Fizikai operátorok

# Motiváció

- Vegyünk egy lekérdezést.
- Ekvivalens relációs algebrai kifejezések:

$$\pi_{dnev,fizetes}(\sigma_{dnev="KING"}(Dolgozo))$$

$$\sigma_{dnev="KING"}(\pi_{dnev,fizetes}(Dolgozo))$$

- Rajzoljuk fel a kifejezés fákat.

```
SELECT dnev, fizetes
FROM dolgozo
WHERE dnev=„KING”;
```

$$\boldsymbol{\pi_{dnev,fizetes}}$$

$$\boldsymbol{\sigma_{dnev="KING"}}$$

**DOLGOZO**

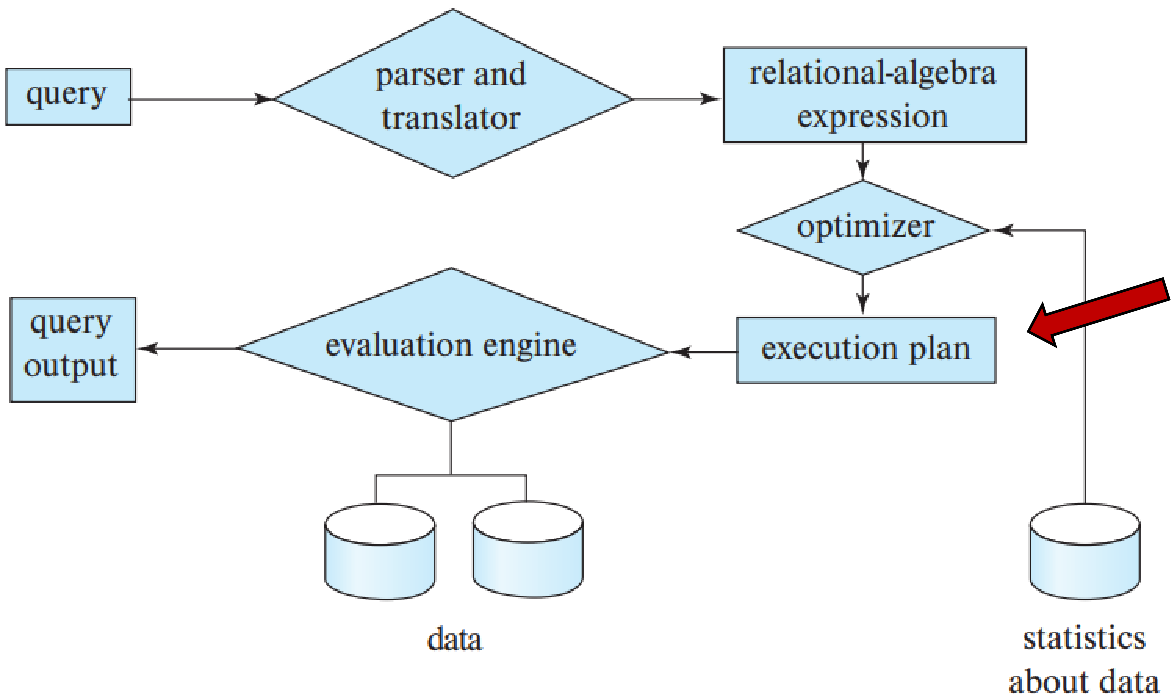
$$\boldsymbol{\sigma_{dnev="KING"}}$$

$$\boldsymbol{\pi_{dnev,fizetes}}$$

**DOLGOZO**

# Lekérdezésfeldolgozás lépései

- Parser and translator – logikai lekérdezéstervet készít (kifejezés fa).
- Optimizer – relációs algebrai (szabály alapú) és költség alapú optimalizálás.
- Evaluation engine – végrehajtja az elkészült fizikai tervet.




aktuális előadás

# Motiváció

- A adat lentről felfelé „folyik”.
- A gyökér kimenete adja meg a lekérdezés eredményét.
- Az egyes operátorokhoz fizikai operátorokat rendelünk.
- Itt is több lehetőségünk van, pl.
  - Használjunk indexet vagy sem?
  - Több indexünk is lehet. Melyik indexet használjuk?

```
SELECT dnev, fizetes
FROM dolgozo
WHERE dnev=„KING”;
```

$\boldsymbol{\pi_{dnev,fizetes}}$

|

$\boldsymbol{\sigma_{dnev="KING"}}$

|

**DOLGOZO**

$\boldsymbol{\pi_{dnev,fizetes}}$

|

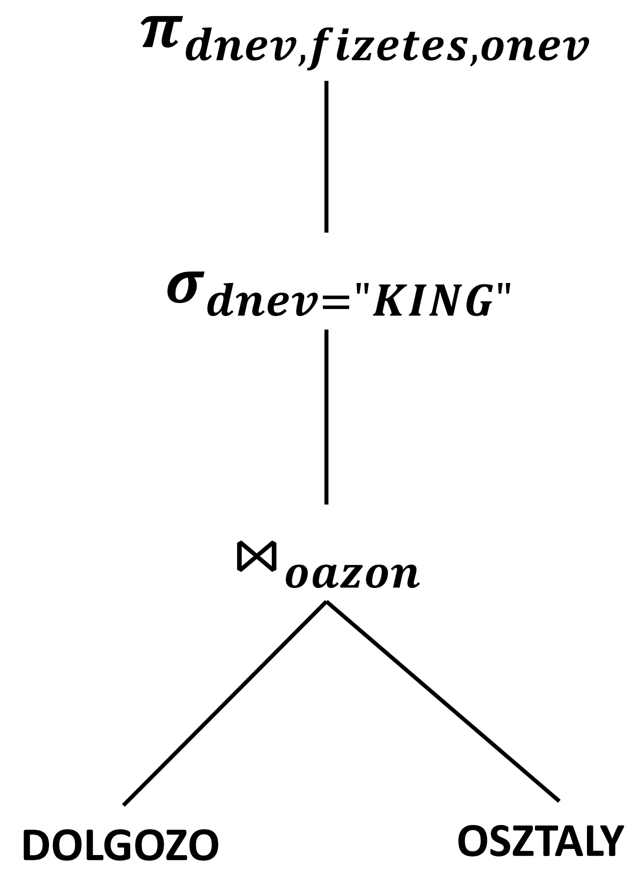
$\boldsymbol{\sigma_{dnev="KING"}}$ index használatával

|

**DOLGOZO**

# Milyen műveleteket vizsgáljunk?

- **Alap műveletek:**
  - Kiválasztás: $\sigma$
  - Vetítés: $\pi$
  - Unió: $\cup$
  - Különbség: $-$
  - Szorzat: $\times$
  - Összekapcsolás: $\bowtie$
- **Kiterjesztett relációs algebra:**
  - Ismétlődések megszüntetése: $\delta$
  - Csoportosítás és aggregáció: $\gamma$
  - Rendezés: $\tau$



# Költségek

- Mit vegyünk figyelembe?
  - **Lemez IO**
  - Ha a másodlagos tároló SSD, akkor az IO műveletek nem dominálják teljesen a költséget, ezért manapság a rendszerek az CPU időt is figyelembe veszik.
- **Két féle költség**: műveleti költség és output méret.
- Elhanyagoljuk a végeredmény kiírásának a költségét.
- Ha ismerjük a kifejezésfa egyes csúcsaiban a költségeket, akkor az összköltséget is meg tudjuk határozni.
- **Célunk**: ismerjük meg az egyes fizikai operátorok működését és költségeit (műveleti költség és output méret).

# Jelölések

- $\boldsymbol{N_R}$ - a rekordok (sorok) száma az R relációban (alternatív jelölés: $T(R)$)
- $\boldsymbol{L_R}$- egy rekord mérete az R relációban (alternatív jelölés: $L(R)$)
- $\boldsymbol{bf_R}$- blokkolási faktor, azaz az R reláció hány rekordja fér el egy blokkban (alternatív jelölés: $bf(R)$)
- $\boldsymbol{B_R}$- az R reláció tárolásához szükséges blokkok száma (alternatív: $B(R)$)

# Jelölések

- $\boldsymbol{V(R, A)}$ – az R reláció A oszlopában lévő különböző értékek száma (alternatív jelölés: $I_A(R)$)
- $\boldsymbol{SC(R, A)}$ – az A oszlop kiválasztási számossága (szelektivitás).
  - Ha A kulcs: $SC(R, A) = 1$
  - Ha A nem kulcs: $SC(R, A) = T_R / V(R, A)$ (egyenletességi feltétel esetén)
- $\boldsymbol{HT_I}$ - az I index szintjeinek a száma (fa magassága)
- Megjegyzés: a nem egész számokat felfelé kerekítjük.

ELTE | IK

# Kiválasztás: $\sigma$

- **Lineáris keresés (index nélkül)**
  - Egyedi értékek esetén az átlagos eset: $B_R/2$
  - Ismétlődő értékek esetén: $B_R$
- **Rendezett mező**
  - Egyedi értékek esetén: $log_2(B_R)$
  - Ismétlődő értékekkel: $log_2(B_R) + m$
    - $m$ további blokkot kell beolvasni.
    - $m = \lceil SC(A, R)/bf_R \rceil - 1$

# Kiválasztás: $\sigma$

- **Elsődleges (klaszter) index (B+ fa):**
  - Egyedi értékek esetén: $HT_I + 1$
  - Ismétlődő értékekkel: $HT_I + \lceil SC(A,R)/bf_R \rceil$
- **Másodlagos index (kupac fájlszervezés):**
  - Egyedi értékek esetén: $HT_I + 1$
  - Ismétlődő értékeknél előfordulhat, hogy az azonos értékek esetén minden rekord különböző blokkban helyezkedik el (legrosszabb eset): $HT_I + SC(A,R)$
  - Nagyon kedvezőtlen eseteknél előfordulhat, hogy a lineáris keresés jobban megéri.

# Összetett kiválasztás: $\sigma_{kif}$

- Konjukciós kiválasztás: $\sigma_{\theta_1 \wedge \theta_2 \ldots \wedge \theta_n}$
- Több féle kiértékelési mód létezik.
- **Index nélküli vagy egy index használatával:**
  - Külön-külön nézzük meg mennyi lenne a kiválasztások költsége (de ne hajtsuk még végre őket).
  - Válasszuk ki azt, amelyiknek a legkisebb a költsége és azt alkalmazzuk először.
  - Eközben a memóriában vizsgáljuk meg a többi feltételt a kiválasztott rekordokra.
  - **Költség**: a kiválasztás költsége a választott feltételre.

# Összetett kiválasztás: $\sigma_{kif}$

- Konjukciós kiválasztás: $\sigma_{\theta_1 \wedge \theta_2 \ldots \wedge \theta_n}$
- **Összetett (composite/concatenated) index**
    - Először azokra a mezőkre végezzük el a kiválasztást, amelyekre létezik összetett index.
    - Ezután a memóriában vizsgáljuk a többi feltételt.
    - **Költség**: a kiválasztás költsége a választott feltételekre.

# Összetett kiválasztás: $\sigma_{kif}$

- Konjukciós kiválasztás: $\sigma_{\theta_1 \wedge \theta_2 \ldots \wedge \theta_n}$
- **Több index használatával**
  - Vegyük a feltételekben szereplő oszlopokhoz tartozó indexeket.
  - Keressük (egyszerre) több indexben.
  - Képezzük a indexekből kapott sorazonosítók (ROWID) metszetét.
  - Ha nincs az összes oszlophoz indexünk, a memóriában vizsgáljuk a többi feltételt.
  - **Költség**: az indexet használó keresések összege + rekordok beolvasása.

# Összetett kiválasztás: $\sigma_{kif}$

- Diszjunkciós kiválasztás: $\sigma_{\theta_1 \vee \theta_2 \dots \vee \theta_n}$
- **Több index esetén:**
  - Ha minden feltételhez tudunk indexet használni, akkor keressük meg a megfelelő rekordokat és vegyük a sorazonosítók unióját.
- **Összetett index esetén:**
  - Ha van olyan összetett indexünk, amely minden feltételt lefed, akkor használhatjuk a kereséshez.
- **Lineáris keresés:**
  - Ha nincs megfelelő összetett index vagy nincs minden feltételhez megfelelő index, akkor a lineáris keresést kell alkalmaznunk.

# Kiválasztás - méretbecslés

- Egyszerű egyenlőségi feltétel: $\sigma_{A=v}(R)$
  - Sorok száma: $SC(A,R)$
  - Blokkok száma: $SC(A,R)/bf_R$
- Tartományra (intervallumra) vonatkozó feltétel: $\sigma_{A\leq v}(R)$
  - Sorok száma: $N_R * \frac{v-\min(A,R)}{\max(A,R)-\min(A,R)}$
  - Blokkok száma: sorok száma$/bf_R$

# Kiválasztás - méretbecslés

- Összetett konjukciós kiválasztás: $\sigma_{\theta_1 \wedge \theta_2 \ldots \wedge \theta_n}$
  - Tegyük fel, hogy függetlenek a feltételek (így felső becslést kapunk).
  - Jelölje $s_i$ az i-edik feltétel szelektivitását, azaz hány sor felel meg a feltételnek (lásd egyenlőségi és tartományra vonatkozó feltétel).
  - Nézzük meg a valószínűségét annak, hogy egy sor kielégít egy feltételt:

$$s_i / N_R$$

  - Ha a feltételek függetlenek, akkor annak a valószínűsége, hogy egy sor minden feltételt kielégít: $(s_1/N_R) * (s_2/N_R) * \cdots * (s_n/N_R)$
  - Mivel $N_R$ sorunk van, így a sorok száma:

$$N_R * [(s_1/N_R) * (s_2/N_R) * \cdots * (s_n/N_R)]$$

  - Blokkok száma: sorok száma/$bf_R$

# Kiválasztás - méretbecslés

- Összetett diszjunkciós kiválasztás: $\sigma_{\theta_1 \vee \theta_2 \ldots \vee \theta_n}$
  - A feltevéseink és jelöléseink ugyanazok, mint az előző esetben.
  - Nézzük meg a valószínűségét annak, hogy egy sor NEM elégít ki egy feltételt:

$$1 - s_i / N_R$$

  - Ha a feltételek függetlenek, akkor annak a valószínűsége, hogy egy sor egyik feltételt sem elégíti ki:

$$(1 - s_1/N_R) * (1 - s_2/N_R) * \cdots * (1 - s_n/N_R)$$

  - Mivel $N_R$ sorunk van, így a sorok száma:

$$N_R * [(1 - s_1/N_R) * (1 - s_2/N_R) * \cdots * (1 - s_n/N_R)]$$

  - Blokkok száma: sorok száma/$bf_R$

# Műveletek visszavezetése rendezésre

- Vetítés: $\pi$
  - Önmagában nem túl érdekes, csak el kell hagyni a nem kért oszlopokat.
  - Viszont ezután előfordulhatnak ismétlődő sorok, amelyeket ki kell szűrni.
  - Ezt **rendezéssel** könnyen el tudjuk végezni.
- Halmazműveletek (mindegyikhez kell a **rendezés**)
  - $R \cap S$ – ki kell szűrni az ismétlődő értékeket.
  - $R \cup S$ – ki kell szűrni az ismétlődő értékeket.
  - $R - S$ – ha mindkét tábla rendezett, akkor elég egyszerre végigolvasni mindkét táblát.
- Rendezés művelet: $\tau$
  - SQL-ben ORDER BY záradék, természetesen ehhez is kell **rendezés**.

# Rendezés

- Két típusról beszélhetünk (külső és belső).
- **Belső rendezés** (internal sorting):
    - Ha a rendezendő adat befér teljesen a memóriába, akkor használhatunk ismert rendező algoritmusokat (quicksort).
    - Lemez IO szempontból ilyenkor egyszer be kell olvasni az összes blokkot és egyszer ki kell írni őket: $2 * B_R$

# Külső rendezés

- Leggyakrabban használt módszer: külső összefésülő rendezés (external sort-merge).
- Legyen M a pufferkészlet kapacitása blokkokban (hány blokk fér el benne) .
- **Két lépésből áll:**
  - Rendezési lépés (sort) – futamokat készítünk.
  - Összevonási lépés (merge) – a futamokat összefésüljük hosszabb futamokká.

# Külső összefésülő rendezés

- Rendezési lépés

```
i=0;
ismétlés
   az R reláció M blokkjának beolvasása a memóriába;
   az M blokk rendezése a memóriában;
   a rendezett M blokk kiírása az Ri fájlba (futamba);
   i=i+1;
amíg el nem fogynak a lapok
N = i;
```

# Külső összefésülő rendezés

- Összevonási lépés
- Ha $N < M$:

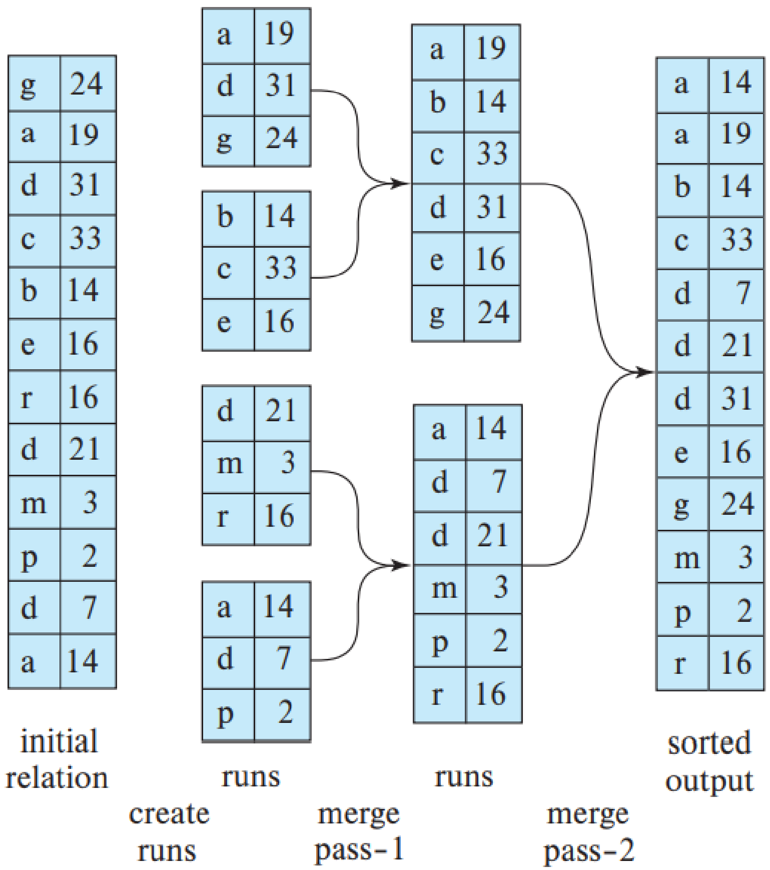
minden $R_i$ fájlhoz egy lap lefoglalása;
minden $R_i$-ből egy-egy blokk ($P_i$) beolvasása;
**ismétlés**
    a blokkok közül a rendezés szerinti első rekord
        kiválasztása;
    a rekord kiírása a kimenetre és törlése a blokkból;
    **ha** bármelyik blokk üres ($P_i$):
        **akkor** újabb blokk beolvasása az $R_i$-ből
**amíg** minden lap ki nem ürül

# Többmenetes külső összefésülő rendezés

- Hogyan alkalmazzuk, ha $N > M$?
  - Több menet van (ezért több menetes).
  - Minden menet M-1 futamot von össze, amíg nincs feldolgozva a reláció.
    - Kell hely M-1 blokknak és 1 kimeneti blokknak.
  - A következő menetben a futamok száma kisebb.
  - A végső menet megadja a végső kimenetet.

# Külső összefésülő rendezés példa

- Tegyük fel, hogy a memóriában 3 sor fér el (egyszerűsített eset).
- Rendezzünk az első oszlop alapján.



# Külső összefésülő rendezés költsége

- Rendezési lépés (futamok elkészítése): $2 * B_R$
  - A relációt teljesen be kellett olvasni és kiírni.
- Összevonási lépés:
  - Kezdetben $\lceil B_R/M \rceil$ összevonandó futam.
  - Minden menet $M - 1$ futamot rendez.
  - Tehát az összes menet száma: $\lceil log_{M-1}(B_R/M) \rceil$
  - Minden menetben minden blokkot olvasunk és írunk, kivéve az utolsó kiírást: $2 * B_R$
- Így a teljes költség: $2 * B_R + 2 * B_R * \lceil log_{M-1}(B_R/M) \rceil - B_R$

# Vetítés: $\pi$

- Vetítés művelet: $\pi_{A_1,A_2\dots}(R)$. Három lépése lehet:
- Felesleges mezők törlése.
  - Végig olvassuk a táblát és elhagyjuk a felesleges oszlopokat és kiírjuk.
  - Műveletigény: $B_R + B_{R_1}$, ahol $R_1$, az új reláció.
- Rendezés (ismétlődések megszüntetéséhez).
  - Az $R_1$-et rendezni kell az összes oszlop alapján.
  - Műveletigény: $2 * B_{R_1} + 2 * B_{R_1} * \lceil log_{M-1}(B_{R_1}/M) \rceil$
- Duplikált rekordok törlése.
  - A rendezett relációt végig olvassuk és töröljük az ismétlődő rekordokat.
  - Műveletigény: $B_R + B_{R_2}$, ahol $R_2$ az új reláció.
- Teljes költség: $B_R + B_{R_1} + 2 * B_{R_1} + 2 * B_{R_1} * \lceil log_{M-1}(B_{R_1}/M) \rceil + B_R + B_{R_2}$
- Felső becslés: $6 * B_R + 2 * B_R * \lceil log_{M-1}(B_R/M) \rceil$

# Vetítés - méretbecslés

- Legyen S $:= \pi_{A_1,A_2 \dots A_k}(R)$
- Felső becslés: $B_R$
- Ha csak egy A oszlopra vetítünk, akkor V(A,R) sor van a vetületben:
  - Blokkok száma: $V(A,R)/bf_S$
- Több oszlop esetén az előforduló különböző értékek alapján megadhatjuk a sorok maximális számát:
  - $V(A_1,R) * V(A_2,R) * \cdots * V(A_k,R)$
- Viszont a vetületben nem lehet több sor, mint a táblában, így a sorok száma:
  - $MIN\ (V(A_1,R) * V(A_2,R) * \cdots * V(A_k,R), N_R)$
  - Blokkok száma: $MIN\ (V(A_1,R) * V(A_2,R) * \cdots * V(A_k,R), N_R)/bf_S$

ELTE | IK

# Unió

- $P \coloneqq R \cup S$
- P sorainak a száma (felső becslés): $N_R + N_s$
- P blokkjainak a száma (felső becslés): $B_R + B_s$
- Ismétlődődé értékek törlése.
  - P rendezése az összes mező alapján
  - Műveletigény: $2 * B_P + 2 * B_P * \lceil log_{M-1}(B_P/M) \rceil$
  - A rendezett relációt végig olvassuk és töröljük az ismétlődő rekordokat.
  - Műveletigény: $B_P + B_{P_1}$
- Teljes számítási költség:
  - $2 * B_P + 2 * B_P * \lceil log_{M-1}(B_P/M) \rceil + B_P + B_{P_1}$
- Felső becslés:
  - $4 * (B_R + B_S) + 2 * (B_R + B_S) * \lceil log_{M-1}((B_R + B_S)/M) \rceil$

# Különbség, metszet, szorzat

- Ezek a műveletek visszavezethetőek a természetes összekapcsolásra.
- Szorzat: $P \coloneqq R \times S$ ($R \bowtie S$ speciális esete, ahol a relációk sémáinak nincs közös attribútuma).
  - Sorok száma: $N_R \times N_S$
  - Blokkok száma: $B_R \times N_S + B_S \times N_R$
- Metszet: $P \coloneqq R \cap S$ ($R \bowtie S$ speciális esete, ahol a relációk sémái teljesen megegyeznek).
  - Sorok száma: $\min(N_R, N_S)$
  - Blokkok száma: $\min(B_R, B_S)$
- Különbség: $P \coloneqq R - S$
  - Sorok száma: $N_R$
  - Blokkok száma: $B_R$

# Összekapcsolások

- Skatulyázott ciklusos összekapcsolás (nested-loop)
  - Egyszerű (nested-loop join)
  - Blokk-skatulyázott ciklusos összekapcsolás (block nested-loop join)
  - Indexelt skatulyázott ciklusos összekapcoslás (index nested-loop join)
- Összefésüléses rendező összekapcsolás.
- Hasításos összekapcsolás (hash join).

# Nested-loop join

- Bármilyen összekapcsolási feltétel esetén működik (=, <, > stb.)
- Ha nincs feltétel, akkor a direkt szorzatot adja vissza.
- Általában a kisebb relációt választjuk **külső** relációnak, a nagyobbat pedig **belső** relációnak.

R **minden** $t_R$ rekordján
    S **minden** $t_S$ rekordján
        **ha** $t_R$ és $t_s$ rekordok kielégítik a feltételt:
            **akkor** a $(t_R, t_S)$ pár kiírása;
    **vége**
**vége**

# Nested-loop join költség

- **Legjobb** eset, ha a kisebb reláció elfér a memóriában (ez lesz a külső).
  - $B_R + B_S$
- **Legrosszabb** eset, ha mindkét relációból csak 1-1 blokk fér el a memóriában.
  - Ilyenkor S-t minden R-beli sornál végig kell olvasni.
  - $N_R * B_S + B_R$

# Nested-loop join költség példa

- $B_R = 500, T_R = 40\,000$
- $B_S = 1000, T_S = 100\,000$
- Ha a kisebb táblát választjuk külső táblának, akkor:
  - $N_R * B_S + B_R = (40\,000 \ * 1000) + 500 = 40\,000\,500$ I/O művelet
- Feladat: mennyi az eredmény ha a nagyobb táblát választjuk külső táblának?

Ne használjuk az egyszerű nested-loop-ot!

# Block nested-loop

- Az előző esetben az R reláció minden sorához végig olvastuk az S reláció blokkjait.
- Valójában elég csak az R reláció blokkjaihoz végigolvasni S blokkjait.

R **minden** $X_R$ blokkján<br>
    S **minden** $X_S$ blokkján<br>
        $X_R$ **minden** $t_R$ rekordján<br>
            $X_X$ **minden** $t_S$ rekordján<br>
                **ha** $t_R$ és $t_s$ rekordok kielégítik a feltételt:<br>
                    **akkor** a $(t_R, t_S)$ pár kiírása;<br>
            **vége**<br>
        **vége**<br>
    **vége**<br>
**vége**

# Block nested-loop költség

- **Legjobb** eset, ha a kisebb reláció elfér a memóriában (ez lesz a külső).
  - $B_R + B_S$
- **Legrosszabb** eset, ha mindkét relációból csak 1-1 blokk fér el a memóriában.
  - Ilyenkor S-t minden R-beli **blokknál** végig kell olvasni.
  - $B_R * B_S + B_R$
- Vegyük észre, hogy az előző esetben a költség jóval nagyobb volt.

ELTE | IK

# Block nested-loop példa

- $B_R = 500, T_R = 40\,000$
- $B_S = 1000, T_S = 100\,000$
- Ha a kisebb reláció (R) befér a memóriába és feltehetjük, hogy $M < B_R - 2$:
  - $1000 + 500 = 1500$ I/O
- Ha $M = 102$ és a lehető legtöbb blokkot töltjük be a kisebb relációból:
  - $\lceil B_R/(B-2) \rceil * B_S + B_R = 5 * 1000 + 500 = 5500$ I/O

# Index nested-loop join

- Ha a belső reláción van indexünk, akkor használhatjuk a sorok elérésére.
- Költség: $B_R + N_R * c$
- Ahol c a belső relációból index szerinti kiválasztás költsége.

$$c = HT_I + \left\lceil \frac{SC(A,R)}{bf_S} \right\rceil = HT_I + \left\lceil \frac{\frac{N_S}{V(A,S)}}{bf_S} \right\rceil =$$

$$HT_I + \left\lceil \frac{B_S}{V(A,S)} \right\rceil \approx \left\lceil \frac{B_S}{V(A,S)} \right\rceil$$

- Mivel a $HT_I$ általában jóval kisebb.
- Azaz a teljes költség elsődleges index és egyenletességi feltétel esetén:

$$B_R + N_R * B_S / V(A,S)$$

# Nested-loop join összefoglalás

- A **kisebb táblát** válasszuk külső táblának.
- A külső tábla **lehető legtöbb** blokkját töltsük be a memóriába.
- Olvassuk végig a belső tábla blokkjait (használjunk **indexet**, ha van).
- A három nested-loop-ból kettő van a gyakorlatban is használatban:
  - Block nested-loop
  - Index nested-loop

# Sort-merge join

- Először a relációkat rendezni kell az összekapcsolási mezők szerint.
- Ezután összefésüljük a rendezett relációkat.

```
R rendezése a megfelelő mezők alapján;
S rendezes a megfelelő mezők alapján;
cursor_r <- R
cursor_s <- S;
amíg cursor_r és cursor_s
   ha cursor_r > cursor_s:
      cursor_s léptetés;
   ha cursor_r < cursor_s:
      cursor_r léptetés;
      cursor_s visszaállítása (ha szükséges);
   ha cursor_r és cursor_s-hez tartozó pár illeszkedik:
      a pár kiírása;
      cursor_s léptetés;
```

ELTE | IK

# Sort-merge join költsége

- A rendezett relációkat **csak egyszer** kell végigolvasni.
- Költség: rendezés költsége + $B_S + B_R$
- **Példa**:
- $B_R = 500, T_R = 40\ 000$
- $B_S = 1000, T_S = 100\ 000$
- M=100
- Költségek:
  - R rendezése: $2000 + 2000 * (log_{99}(1000/100)) = 4000$ I/O
  - $S$ rendezése: $1000 + 1000 * (log_{99}\ (500/100)) = 2000$ I/O
- Összesen:
  - rendezés költsége + $B_S + B_R = 4000 + 2000 + 1000 + 500 = 7500$ I/O

# Sort-merge join használata

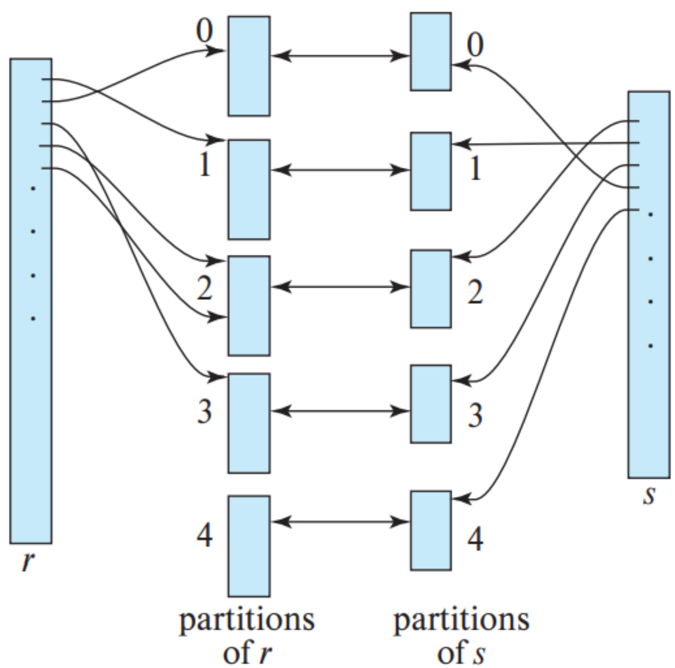
- Mikor érdemes ezt az algoritmust választani?
  - Ha egyik vagy mindkét tábla már egyébként is rendezve van az összekapcsoláshoz szükséges mezőkön.
  - Ha a kimenetet egyébként is rendezni kell.
  - Ha egyenlőség alapú vagy természetes összekapcsolást hajtunk végre.
- A rendezés történhet explicit, rendező operátorral vagy index-alapú beolvasással.

# Hash join

- A R és S reláció soraira ugyanazt a $\boldsymbol{h}$ **hasító függvényt** alkalmazzuk az összekapcsolási oszlopra. Olyan hasítófüggvényt válasszunk, hogy egy-egy kosár mindenképpen beférjen a memóriába.
- R rekordjai kerüljenek az $\boldsymbol{R_0} \dots \boldsymbol{R_{n-1}}$ kosarakba.
- S rekordjai kerüljenek az $\boldsymbol{S_0} \dots \boldsymbol{S_{n-1}}$ kosarakba.
- Ha egy rekord az R-ből egy rekord az S-ből kielégítik az összekapcsolási feltételt (azaz megegyezik a megfelelő oszlop értéke), akkor **ugyanabba** az indexű kosárba kell esniük.
- Ezért a hasítótáblák felépítése után **elég csak a kosárpárokat** beolvasni és megtaláljuk az összes illeszkedő rekordot.

# Hash join költsége

- Költség: $2 * (B_R + B_S) + (B_R + B_S) = 3 * (B_R + B_S)$
- Ha a kosárpárok túl nagyok (nem férnek be a memóriába), akkor:
  - Használhatunk a kosárpárok összekapcsolására nested-loopot.
  - **Rekurzívan** tovább particionálhatjuk a kosarakat (egy másik hasító függvénnyel).



# Hash join példa

- $B_R = 500, T_R = 40\,000$
- $B_S = 1000, T_S = 100\,000$
- $3 * (B_R + B_S) = 3 * (1000 + 500) = 4500$ I/O

# Hash join megjegyzések

- **Nagyon nagy** tábláknál is használható.
- Ha ismerjük a külső tábla méretét, akkor használhatunk **statikus** hasítást.

# Összekapcsolások méretbecslése: $R \bowtie S$

- Ha $R \cap S = \emptyset$, akkor $R \bowtie S = R \times S$
  - Sorok száma: $N_R \times N_S$
  - Blokkok száma: $B_R * N_S + B_S * N_R$
- Ha $R \cap S = \{A\}$, és A kulcs R-ben (S-nek idegen kulcsa)
  - Sorok száma: $N_S$
  - Blokkok száma: $(B_R * N_S + B_S * N_R)/N_R$

# Összekapcsolások méretbecslése: $R \bowtie S$

- Ha $R \cap S = \{A\}$, és A nem kulcs sem R-ben, sem S-ben.
  - Egyik irányból, minden R-beli sorhoz $N_S/V(A,S)$ sor illeszkedhet, ezért:
    $N_R * N_S/V(A,S)$
  - Másik irányból, minden S-beli sorhoz $N_R/V(A,R)$ sor illeszkedhet, ezért:
    $N_R * N_S/V(A,R)$
- Ezért a sorok száma: $N_S * N_R/\max(V(A,R),V(A,S))$
- Blokkok száma:$(B_R * N_S + B_S * N_R)/\max(V(A,R),V(A,S))$
- Speciális eset, ha $R.A \subseteq S.A$, akkor:
  - Sorok száma: $N_R * N_S/V(A,S)$
  - Blokkok száma: $(B_R * N_S + B_S * N_R)/V(A,S)$

ELTE | IK

# Összekapcsolás költségek összefoglalás

| **Algoritmus** | **Költség** | **I/O (példa alapján)** |
|---|---|---|
| Nested-loop | $N_R * B_S + B_R$ | 40 000 500 |
| Block nested-loop | $\lceil B_R/(B-2) \rceil * B_S + B_R$ | 5500 |
| Index nested-loop | $B_R + N_R * c$ | Index méretétől, különböző értékek számától is függ |
| Sort-merge join | rendezés költsége + $B_S + B_R$ | 7500 |
| Hash join | $3 * (B_R + B_S)$ | 4500 |

# Összekapcsolás költségek következtetések

- Általában a hash join és sort-merge join jobb választás, mint a nested-loop.
- Ezek közül is főleg a hash join.
- Megjegyzések:
    - Ha az adat eloszlása nem egyenletes, akkor a sort-merge jobb lehet.
    - Ha egyébként is kell rendezés, akkor szintén jó választás a sort-merge.
- A gyakorlatban mindhárom algoritmus előfordulhat.

# Haladó témák

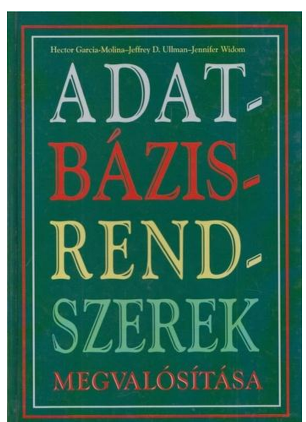
- Összetett lekérdezések feldolgozása:
  - Iterátor model (pipelineing) – leggyakrabban használt
  - Materializálásos model – ritkán használt
  - Vektorizálásos model – előfordul (speciális rendszerekben gyakrabban)
- Párhuzamos lekérdezés végrehajtás.

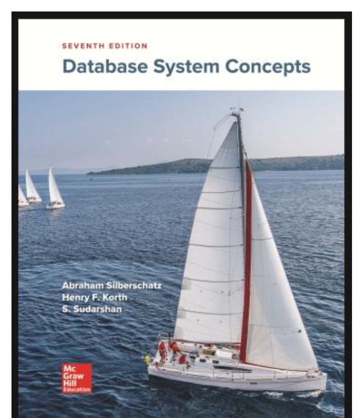
# Tankönyv fejezetek

- Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom:

  **Adatbázisrendszerek megvalósítása**

  - 6. fejezet: Lekérdezések végrehajtása
- Silberschatz, Korth, & Sudarshan: **Database System Concepts**
  - Chapter 15. Query Processing




ELTE | IK