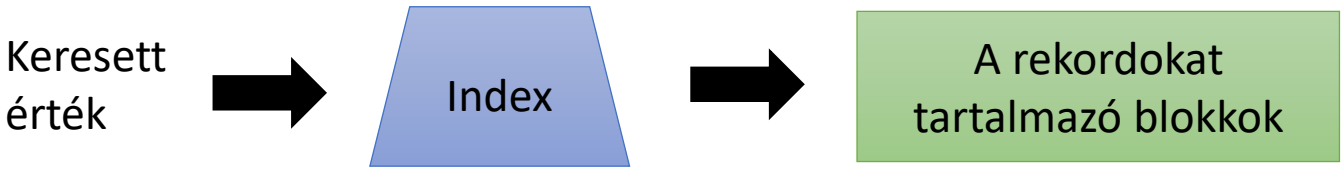
ELTE | IK
INFORMATIKAI KAR

Adatbázisok 2

Indexek

Mik azok az indexek?

- Az **indexstruktúrák** kiegészítő adatszerkezetek, amelyek célja a keresések felgyorsítása.
- Az indexet egy **keresés mezőre** (keresési kulcs) készítjük el, amely egy vagy több mezőből is állhat.
- Több mezőre is készíthetünk indexet.
- Hátrányai:
  - Az index tárolása növeli a tárméretet.
  - Nem csak a főfájlt, hanem az indexet is karban kell tartani, ami plusz költséget jelent.



Az indexek szerkezete

- Az index **indexbejegyzéseket** (kulcs-mutató párokat) tartalmaz:
  - **(a, p)** – ahol „a” a keresési kulcs, azaz egy érték az indexelt oszlopban, „p” pedig egy blokkmutató, amely arra a blokkra mutat, amelyben az „a” értékű rekordot tároljuk.
- Az index mindig rendezett az indexérték szerint (rendezett index).
- Új jelölések:
  - Jelölje bf a blokkolási faktort, amely azt mutatja, hogy az adott objektum rekordjaiból hány fér el egy blokkban.
  - Jelölje T egy objektum rekordjainak a számát.
- A indexek mérete általában sokkal kisebb, mint a reláció mérete, mivel:
  - $bf(I) \gg bf(R)$, ahol I egy index az R reláción.

Indexek kiértékelésének szempontjai

- Támogatott elérési módok:
  - Keresés egyenlőségi feltétel alapján.
  - Intervallum lekérdezés.
- Műveletek költségei:
  - Elérési idő (keresési idő)
  - Beszúrás
  - Módosítás
  - Törlés
- Tárhely igény

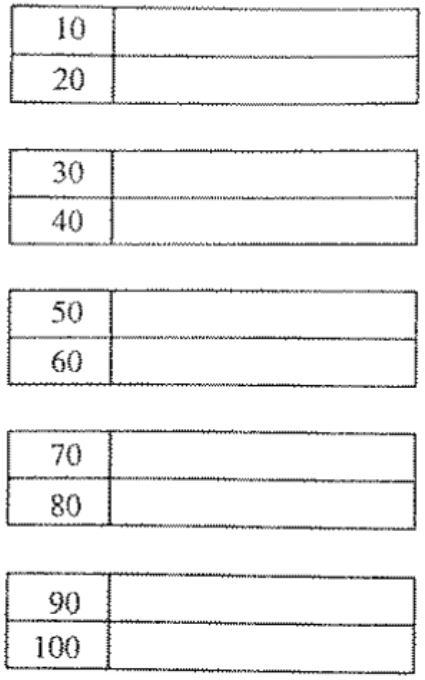
Index típusok a főfájl szervezése alapján

- **Elsődleges index** (primary index).
  - A fájl rendezett a keresési kulcs alapján.
  - Csak egy elsődleges indexünk lehet.
- **Másodlagos index** (secondary index).
  - A főfájl nem rendezett a keresési kulcs szerint.
  - Több másodlagos indexet is készíthetünk.

Rendezett fájl

Példa egy rendezett fájlra:

- 5 blokkból áll.
- Egy blokkba két rekord fér el.
- Egy mező alapján rendezett.



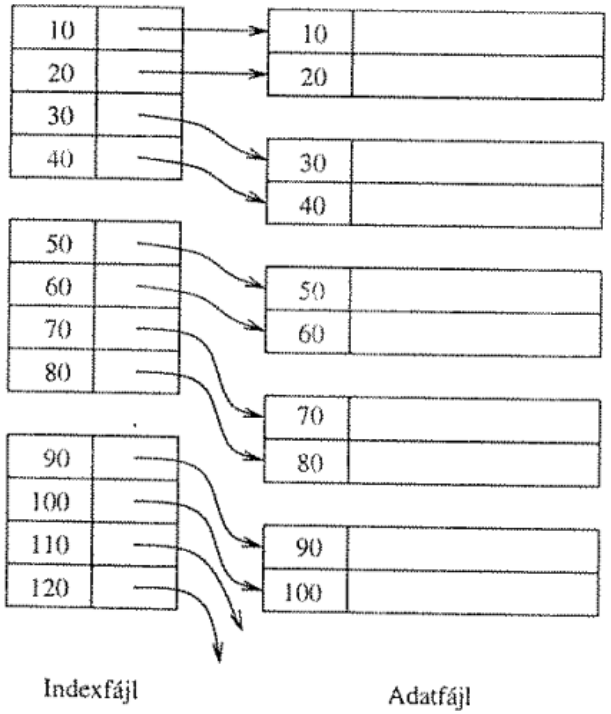
Egy rendezett fájl

Index típusok az index szerkezete szerint

- **Sűrű index**
  - Minden keresési kulcshoz tartozik egy index bejegyzés.
- **Ritka index**
  - Egy blokkhoz csak egy indexbejegyzés tartozik.
  - Kevesebb helyet foglal, mint a sűrű index, de a keresés lassabb lehet.

Sűrű index (elsődleges index)

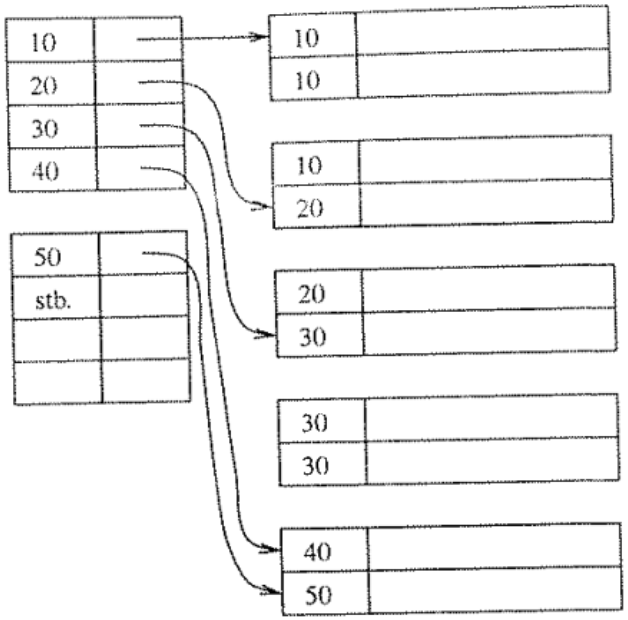
- $T(I) = T(R)$ és $B(I) = T(R)/bf(I)$
- $bf(I) \gg bf(R)$ ezért $B(I) \ll B(R)$
- Miért hasznos?
  - Az indexfájl kevesebb helyet foglal.
- Keresés ritka indexben:
  - **Bináris keresés** az indexfájlon: $log_2 B(I)$
  - Ezután még be kell olvasni az adatot.
  - **Összesen**: $log_2 B(I) + 1$



Sűrű index (rendezett főfájl)

Sűrű index ismétlődő keresési kulcs esetén

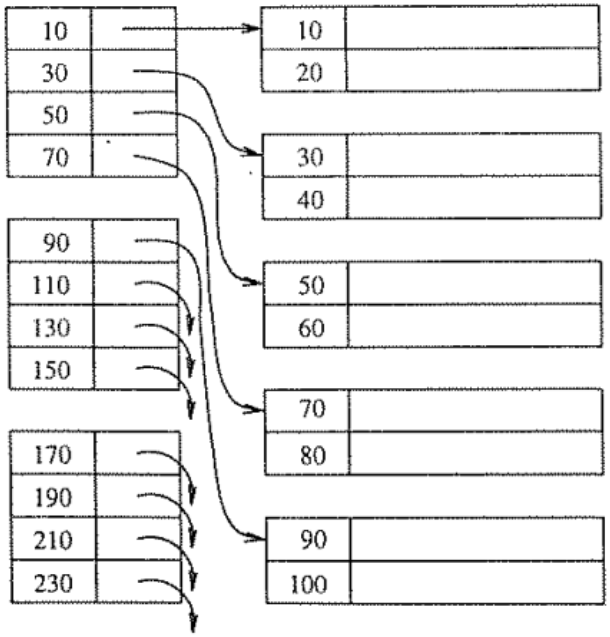
- Két megközelítés:
  - Minden rekordhoz tartozzon bejegyzés (akkor is ha ismétlődnek)
  - Hatékonyabb: Csak az első K értékkel rendelkező rekordhoz tartozzon indexbejegyzés.



Sűrű index ismétlődő kulcsokkal
(hatékonyabb megközelítéssel)

Ritka index (elsődleges index)

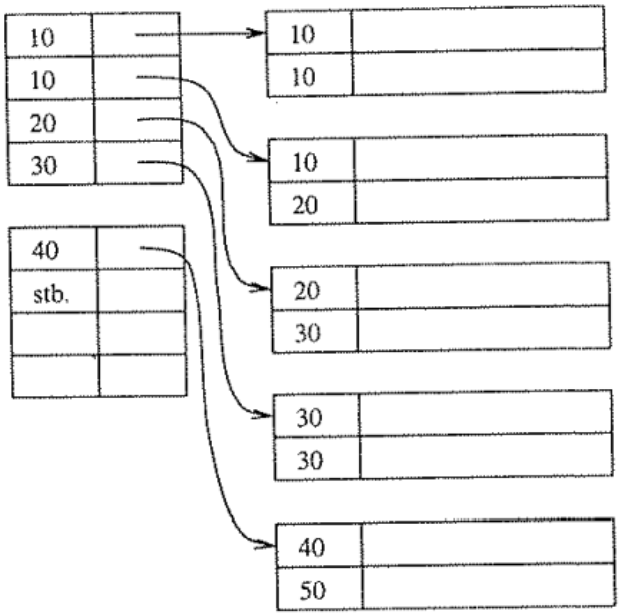
- Minden blokkhoz egy index bejegyzés tartozik:
- $bf(I) \gg bf(R)$ ezért $B(I) \ll B(R)$
- $T(I) = B(R)$ és $B(I) = B(R)/bf(I)$
- Keresés ritka indexben:
  - Az indexfájlban nem szerepel minden érték, ezért csak **fedő értéket** kereshetünk (a legnagyobb olyan indexértéket, amely a keresett értéknél kisebb vagy egyenlő).
  - Ugyanúgy **bináris keresés**: $log_2 B(I) + 1$
  - Viszont a sűrű indextől jobb a keresési idő, mivel kevesebb az index bejegyzés.



Ritka index (rendezett főfájl)

Ritka index ismétlődő keresési kulcs esetén

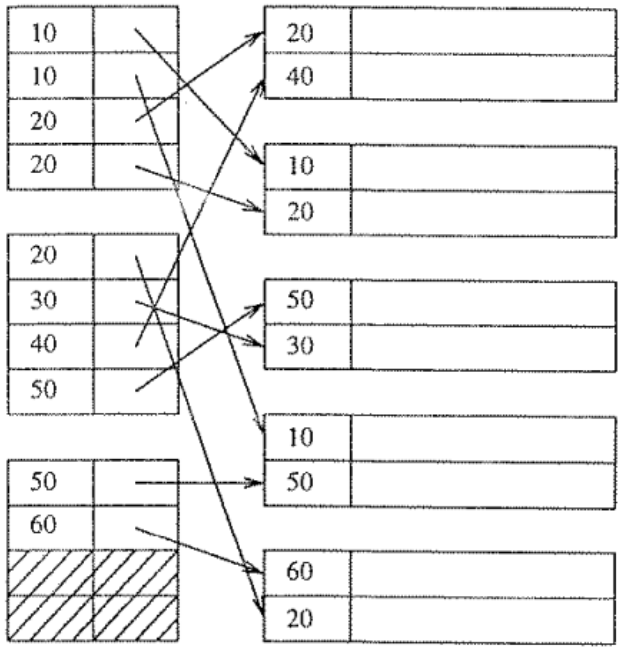
- Az indexfájlban ismétlődhetnek az értékek.
- Keresés során megkeressük az első előfordulást az index fájlban, a további értékeket pedig utána megtaláljuk.



Ritka index ismétlődő kulcsokkal

Másodlagos index

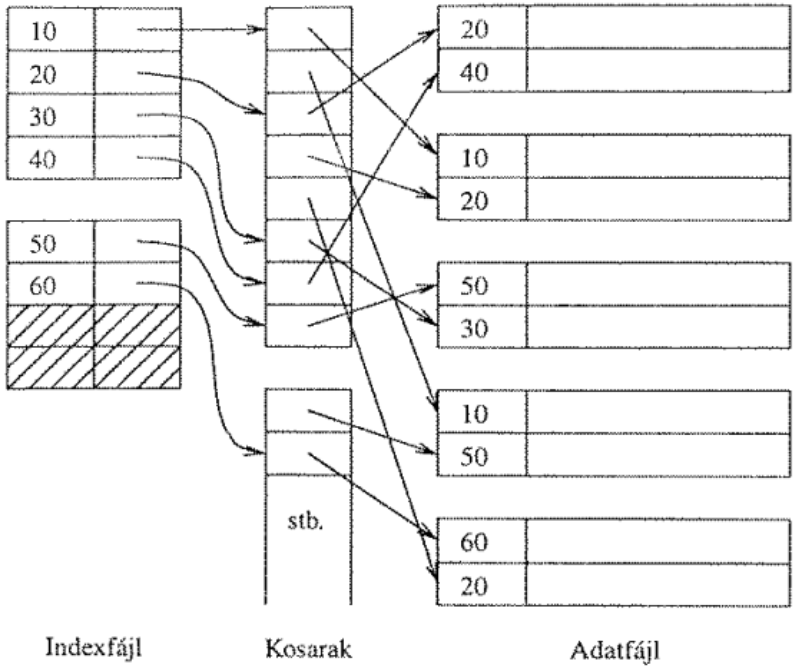
- A főfájl **nem rendezett** a keresési kulcs szerint.
- A másodlagos index szerkezetét tekintve **mindig sűrű** (miért?)
- Egy relációhoz **több** másodlagos indexet is készíthetünk.



Sűrű index ismétlődő kulcsokkal

Helymegtakarítás másodlagos indexben

- A főfájl és az indexfájl közé létrehozhatunk egy **kosártömböt**.
- **Előny**: ismétlődés esetén a keresési kulcs csak egyszer szerepel az indexben.



Nyalábolt fájl (klaszter szervezés, clustered file)

- **Klaszter szervezésű fájl** esetén két vagy több relációt úgy tárolunk, hogy fizikailag a rekordok össze vannak keveredve. Azaz egy blokkban különböző relációk rekordjai is szerepelhetnek (pl. Oracle).
- **Relációk**: Dolgozo(dnev, fizetes, oazon) és Osztaly(oazon, onev).
- Klaszter kulcs: **oazon**.
- **Klaszter index**: klaszterszervezésű fájl esetén index a klaszter kulcson.
- Ha a fizetés mezőre is készítünk indexet az másodlagos index lesz.

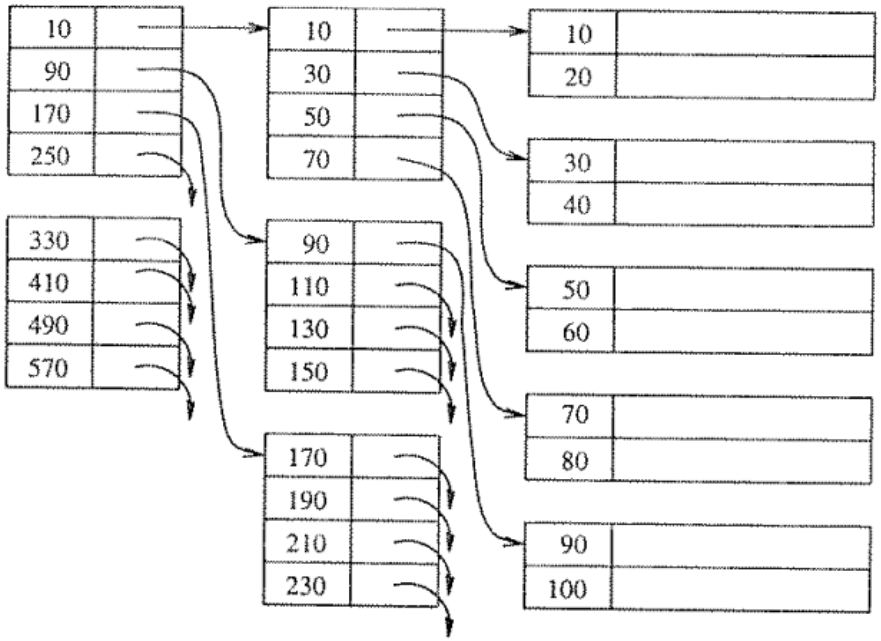
```
SELECT d.dnev
FROM dolgozo d JOIN osztaly d
ON d.oazon = o.oazon
WHERE o.onev=„SALES”;
```

(10, „SALES”)
(„BLAKE”, 1500)
(„KING”, 1900)

(20, „ACCOUNTING”)
(„ALLAN”, 1300)
(„FORD”, 1800)

Többszintű index

- Ha az index nem fér el a memóriában indexelhetjük az indexeket is.
- Az első szint lehet ritka vagy sűrű (a főfájl rendezésétől függően).
- A 2. szinttől kezdve az indexek **mindig ritkák** (miért?)



Többszintű index (t darab szinttel)

- Az indexeket is indexeljük összesen ***t* szintig**.
- A legfelső szinten bináris kereséssel megkereshetjük az indexbejegyzést (mivel rendezett). Ezután követjük a mutatókat teljesen a főfájlig.
- Ha a legfelső szint 1 blokkból áll, akkor $\boldsymbol{t+1}$ blokkolvasásra van szükség.
- Minden szint blokkolási faktora megegyezik, mert egyforma méretűek az index bejegyzések.

Többszintű index

- Mennyi a szintek száma?
  - $t = log_{bf(I)} B(R)$
- Mivel $bf(I) \gg 2$, ezért a többszintű indexben a keresés gyorsabb, mintha csak rendezett fájlszervezést használnánk.

	Főfájl	**1. szint**	**2. szint**		**t. szint**
Blokkok száma	$B(R)$	$B(R)/bf(I)$	$B(R)/bf(I)^2$	…	$B(R)/bf(I)^t$
Rekordok száma	$T(R)$	$B(R)$	$B(R)/bf(I)$	…	$B(R)/bf(I)^{t-1}$
Blokkolási faktor	$bf(R)$	$bf(I)$	$bf(I)$	…	$bf(I)$

Tankönyv fejezetek

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

  Adatbázisrendszerek megvalósítása

  - 4. fejezet: Indexstruktúrák (4.1, 4.2) (153-181. oldalak)
- Silberschatz, Korth, & Sudarshan: **Database System Concepts**
  - Chapter 14. Indexing (14.1, 14.2)