Fájlszervezés

Adatbázisok tervezése, megvalósítása és menedzselése

Kezdetek

- **Célok:**
  - gyors lekérdezés,
  - gyors adatmódosítás,
  - minél kisebb tárolási terület.

- Nincs általánosan legjobb optimalizáció. Az egyik cél a másik rovására javítható (például indexek használatával csökken a keresési idő, nő a tárméret, és nő a módosítási idő).

Költségek

- Hogyan mérjük a költségeket?
- Memória műveletek nagyságrenddel gyorsabbak, mint a háttértárolóról beolvasás, kiírás.
- Az író-olvasó fej nagyobb adategységeket (blokkokat) olvas be.
- A blokkméret függhet az operációs rendszertől, hardvertől, adatbázis-kezelőtől.
- A blokkméretet fixnek tekintjük. Oracle esetén 8K az alapértelmezés.
- Feltételezzük, hogy a beolvasás, kiírás költsége arányos a háttértároló és memória között mozgatott blokkok számával.

Blokkok

- A blokkok tartalmaznak:
  - leíró fejlécet (rekordok száma, struktúrája, fájlok leírása, belső/külső mutatók (hol kezdődik a rekord, hol vannak üres helyek, melyik a következő blokk, melyik az előző blokk, statisztikák (melyik fájlból hány rekord szerepel a blokkban)),
  - rekordokat (egy vagy több fájlból),
  - üres helyeket.
- Feltesszük, hogy B darab blokkunk van.

Rekordok

- A fájl rekordokból áll.
- A rekordok szerkezete eltérő is lehet.
- A rekord tartalmaz:
  - leíró fejlécet (rekordstruktúra leírása, belső/külső mutatók, (hol kezdődik egy mező, melyek a kitöltetlen mezők, melyik a következő rekord, melyik az előző rekord), törlési bit, statisztikák),
  - mezőket, melyek üresek, vagy adatot tartalmaznak.
- A rekordhossz lehet:
  - állandó,
  - változó (változó hosszú mezők, ismétlődő mezők miatt).
- Az egyszerűség kedvéért feltesszük, hogy állandó hosszú rekordokból áll a fájl, melyek hossza az átlagos rekordméretnek felel.

Milyen lekérdezések?

- Milyen lekérdezéseket vizsgáljunk?
- A relációs algebrai kiválasztás felbontható atomi kiválasztásokra, így elég ezek költségét vizsgálni.
- A legegyszerűbb kiválasztás:
  - A=a (A egy keresési mező, a egy konstans)
- Az esetek vizsgálatánál az is számít, hogy az A=a feltételnek megfelelő rekordokból lehet-e több, vagy biztos, hogy csak egy lehet.
- Fel szoktuk tenni, hogy az A=a feltételnek eleget tevő rekordokból nagyjából egyforma számú rekord szerepel. (Ez az egyenletességi feltétel.)

Mivel foglalkozunk majd?

- A következő fájlszervezési módszereket fogjuk megvizsgálni:
  - B^+-fa, B*-fa
  - bitmap index.
- Módosítási műveletek:
  - beszúrás (insert)
  - frissítés (update)
  - törlés (delete)
- Itt nem foglalkozunk azzal, hogy a beolvasott rekordokat bent lehet tartani a memóriában, későbbi keresések céljára.

Indexek

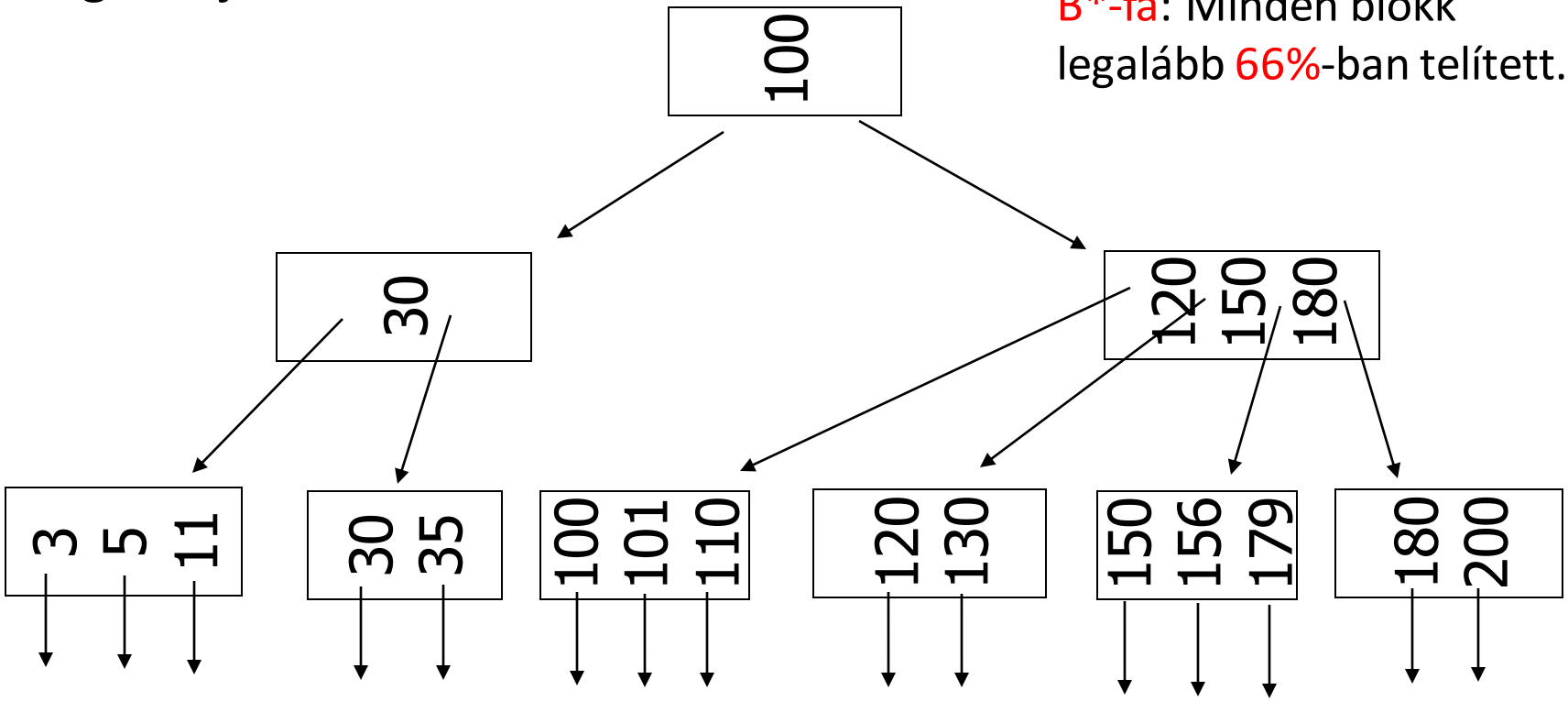
- Indexek használata:
  - keresést gyorsító segédstruktúra,
  - több mezőre is lehet indexet készíteni,
  - az index tárolása növeli a tárméretet,
  - nem csak a főfájlt, hanem az indexet is karban kell tartani, ami plusz költséget jelent.
- Az indexrekordok szerkezete:
  - **(a,p)**, ahol a egy érték az indexelt oszlopban, p egy blokkmutató, arra a blokkra mutat, amelyben az A=a értékű rekordot tároljuk (vagy többszintű index esetén esetleg egy másik indexblokkra).
  - az index mindig rendezett az indexértékek szerint.

B-fák

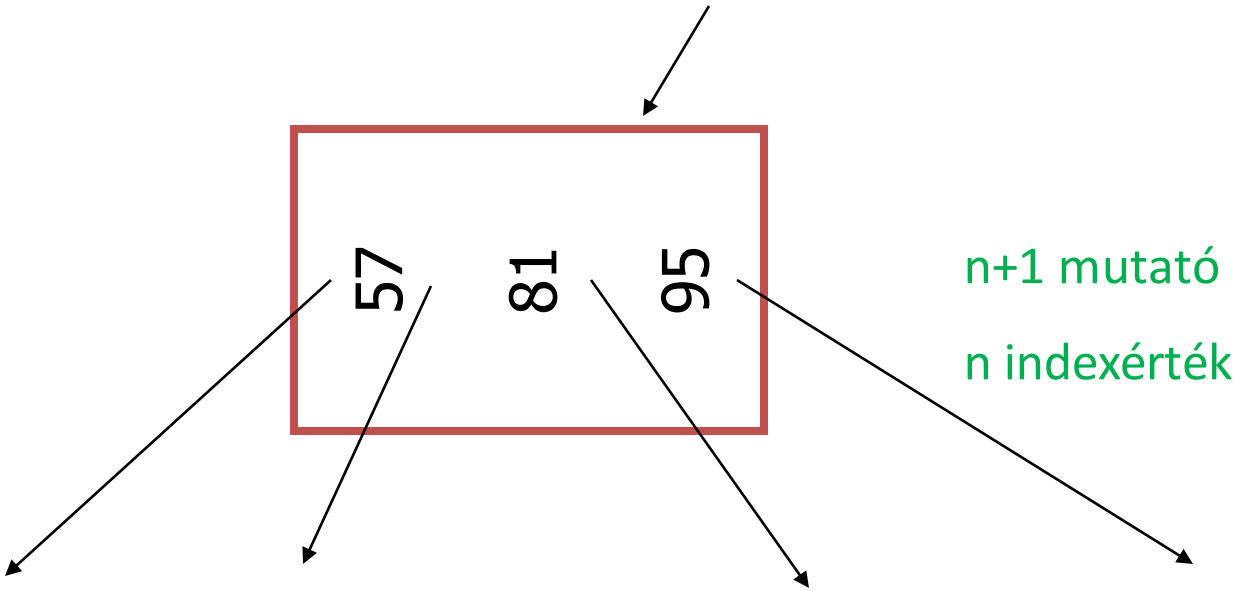
A többszintű indexek közül a **B⁺-fák, B*-fák** a legelterjedtebbek.

B⁺-fa: Minden blokk legalább 50%-ban telített.

B*-fa: Minden blokk legalább 66%-ban telített.

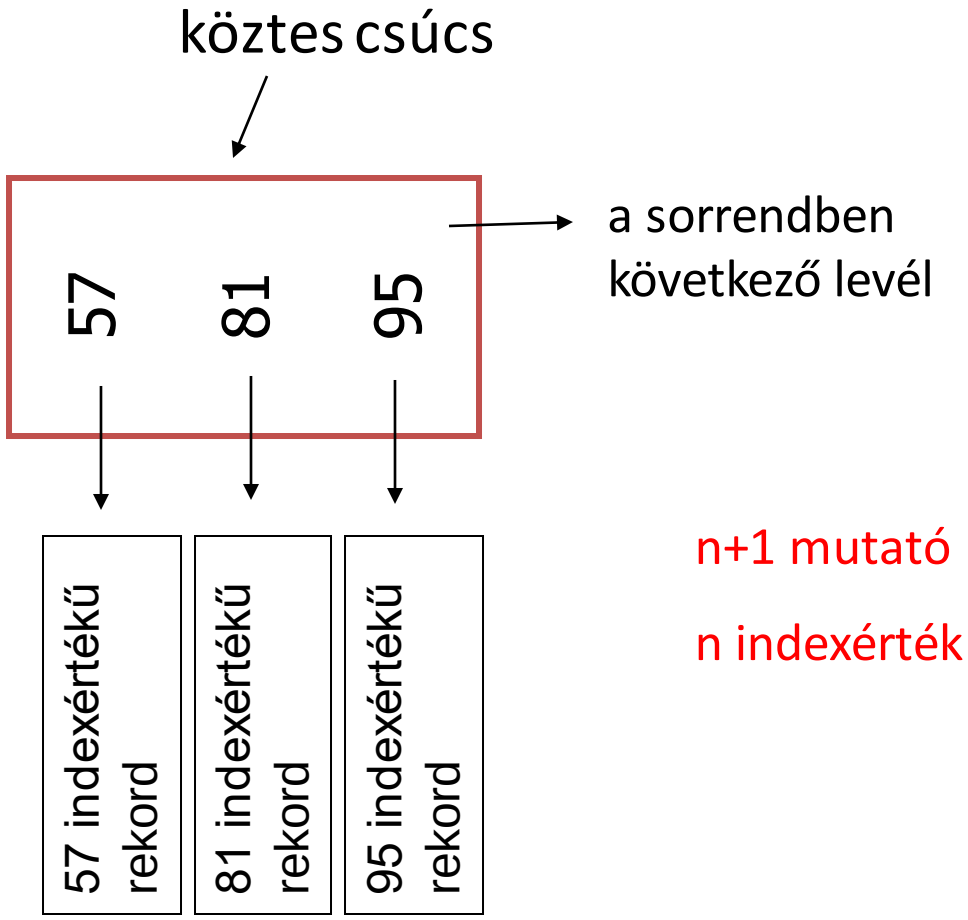


Köztes (nem-levél) csúcs szerkezete



Ahol k a mutató által meghatározott részgráfban szereplő tetszőleges indexérték.

Levél csúcs szerkezete



Felépítésre vonatkozó szabályok

- Szabályok (feltesszük, hogy egy csúcs n értéket és n+1 mutatót tartalmazhat):
  - a gyökérben legalább két mutatónak kell lennie (kivéve, ha a B-fa (B+-fa) egyetlen bejegyzést tartalmaz),
  - a levelekben az utolsó mutató a következő (jobboldali) levélre mutat, legalább $\lfloor (n+1)/2 \rfloor$ mutatónak használatban kell lennie,
  - köztes pontokban minden mutató a B-fa következő szintjére mutat, közülük legalább $\lceil (n+1)/2 \rceil$ darabnak használatban kell lennie,
  - ha egy mutató nincs használatban úgy vesszük, mintha NILL mutató lenne.

Beszúrás I.

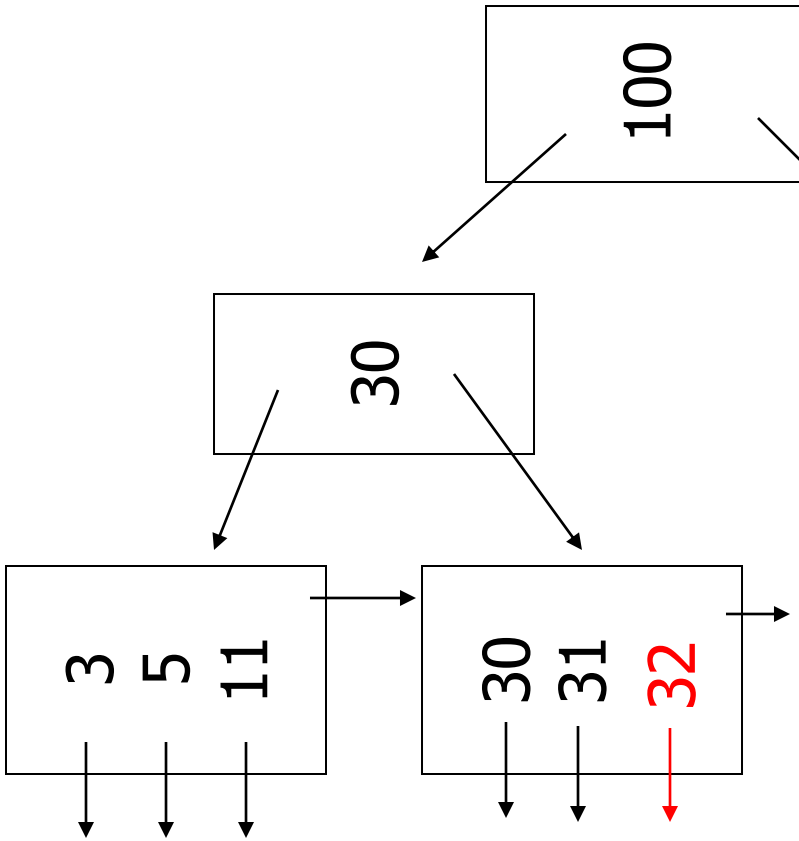
- Ha van szabad hely az új kulcs számára a megfelelő levélben, beszúrjuk.
- Ha nincs, kettévágjuk a levelet, és szétosztjuk a kulcsokat a két új levél között, így mindkettő félig lesz telítve, vagy éppen csak egy kicsit jobban.
- Az eggyel feljebb lévő szint megfelelő csúcsát ennek megfelelően kell kiigazítani.

Beszúrás II.

- Egy csúcs szétvágása tehát hatással lehet a fölötte lévő szintre is. Itt az előbbi két pontban megadott stratégiát alkalmazzuk rekurzívan.
- Itt viszont: ha N olyan belső csúcs, aminek kapacitása n kulcs, n+1 mutató és most az (n+2). mutatót illesztenénk be, akkor szintén létrehozunk egy új M pontot. N-ben most marad $\lceil n/2 \rceil$ kulcs, M-ben lesz $\lfloor n/2 \rfloor$ kulcs, a "középen lévő" kulcs pedig az eggyel fentebbi szintre kerül, hogy elválassza N-t és M-t egymástól.
- Ha a gyökérbe nem tudunk beszúrni, mert nincs hely, akkor szétvágjuk a gyökeret két új csúcsra, és „fölöttük” létrehozunk egy új gyökeret, aminek két bejegyzése lesz.

Példa beszúrásra I.

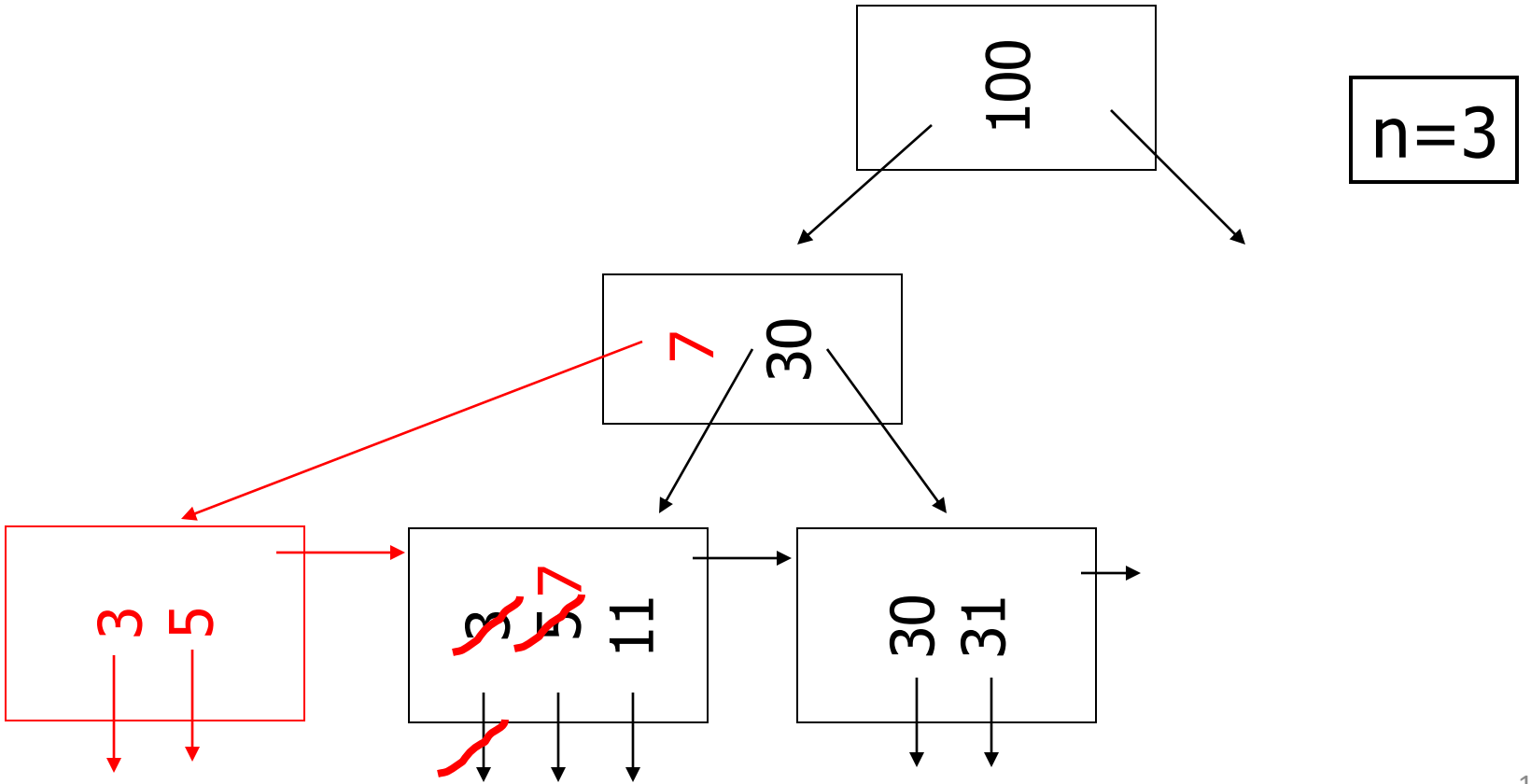
Szúrjuk be a 32-es indexértékű rekordot!



n=3

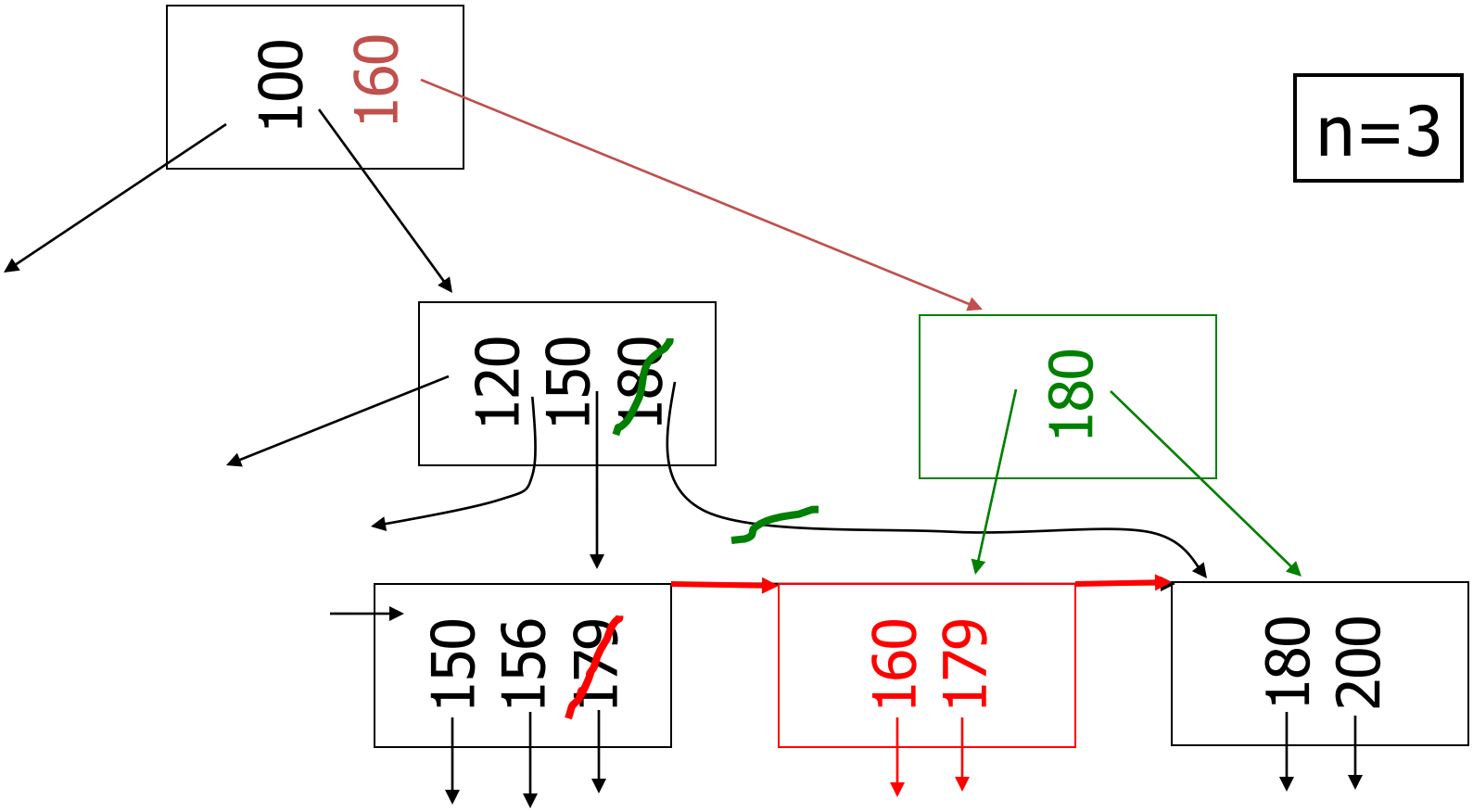
Példa beszúrásra II.

Szúrjuk be a 7-es indexértékű rekordot!



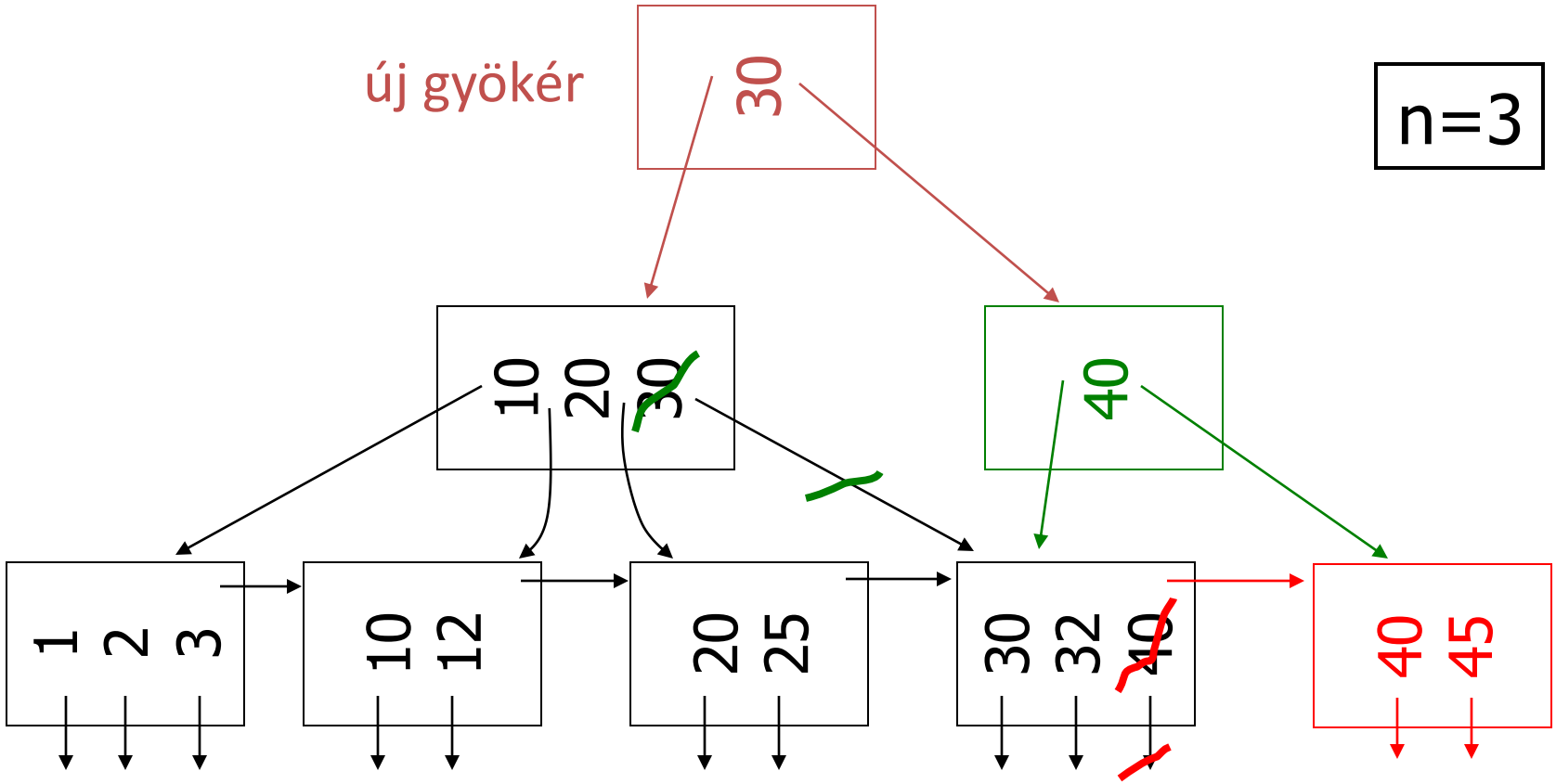
Példa beszúrásra III.

Szúrjuk be a 160-as indexértékű rekordot!



Példa beszúrásra IV.

Szúrjuk be a 45-ös indexértékű rekordot!

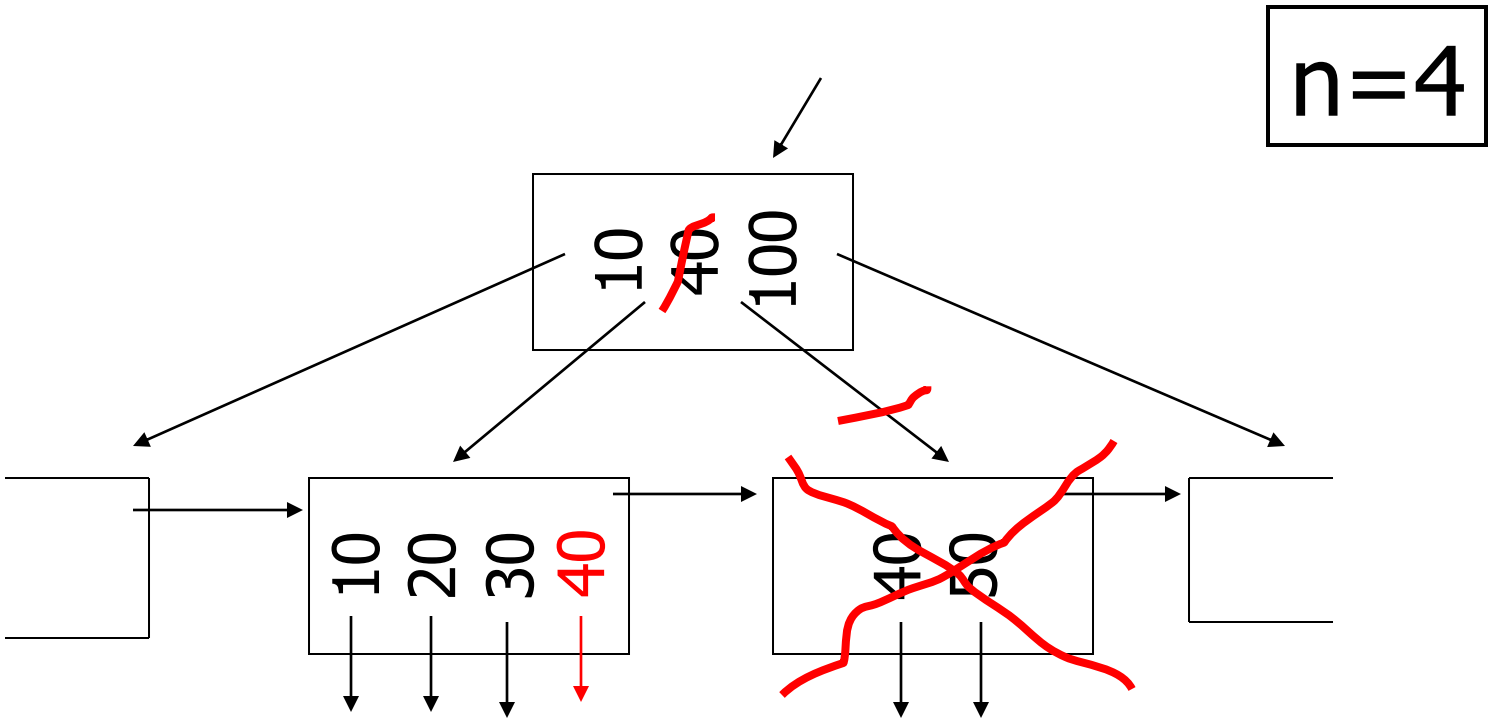


Törlés

- Ha a törlés megtörtént az N pontban, és N még mindig megfelelően telített, akkor készen vagyunk.
- Ha N már nem tartalmazza a szükséges minimum számú kulcsot és mutatót, akkor:
  - ha N valamelyik testvérével összevonható, akkor vonjuk össze. A szülőcsúcsban törlődik ekkor egy elem, s ez további változtatásokat eredményezhet.
  - Ha nem vonható össze, akkor N szomszédos testvérei több kulcsot és mutatót tartalmaznak, mint amennyi a minimumhoz szükséges, akkor egy kulcs-mutató párt áttehetünk N-be. (Baloldali testvér esetén a legutolsó, jobboldali testvér esetén a legelső kulcs-mutató párt.) A változást a szülő csúcson is „regisztrálni kell”.

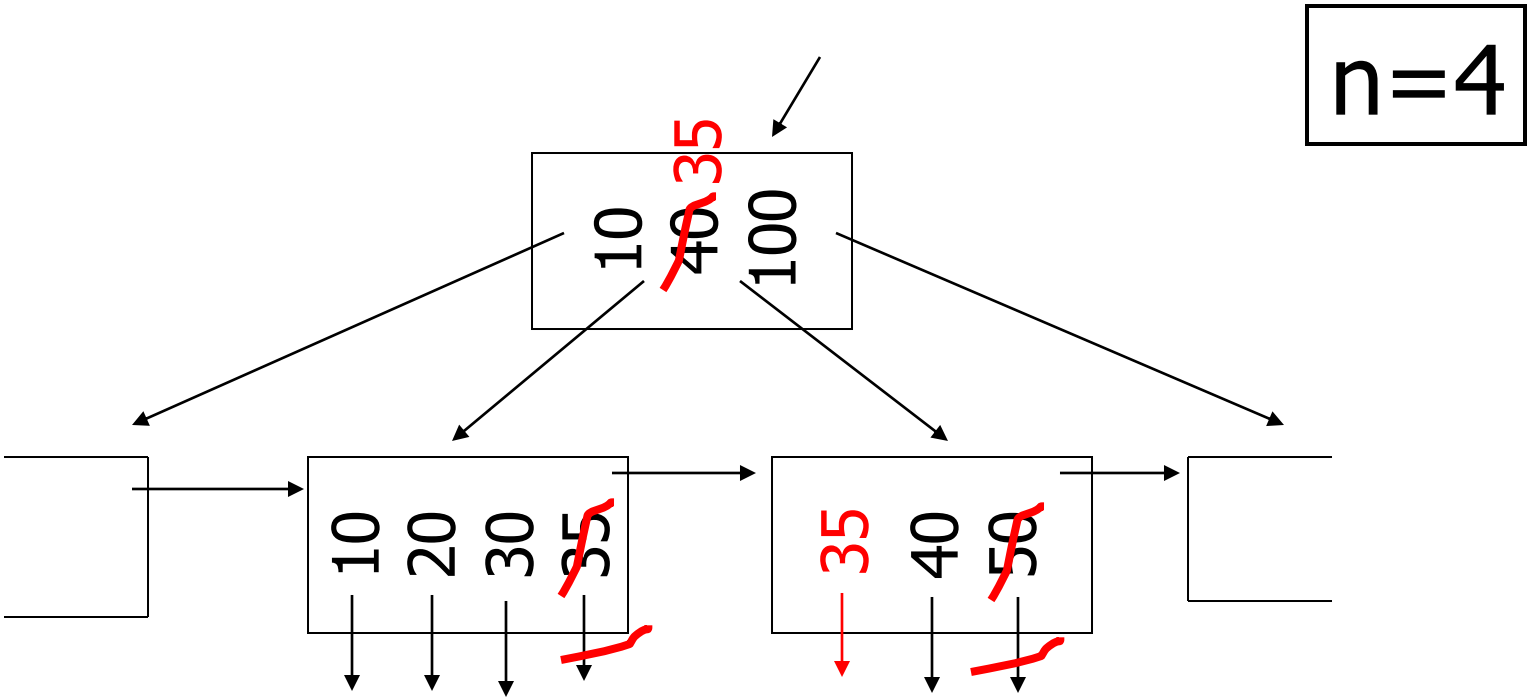
Példa törlésre I.

Töröljük az 50-es indexértékű rekordot!



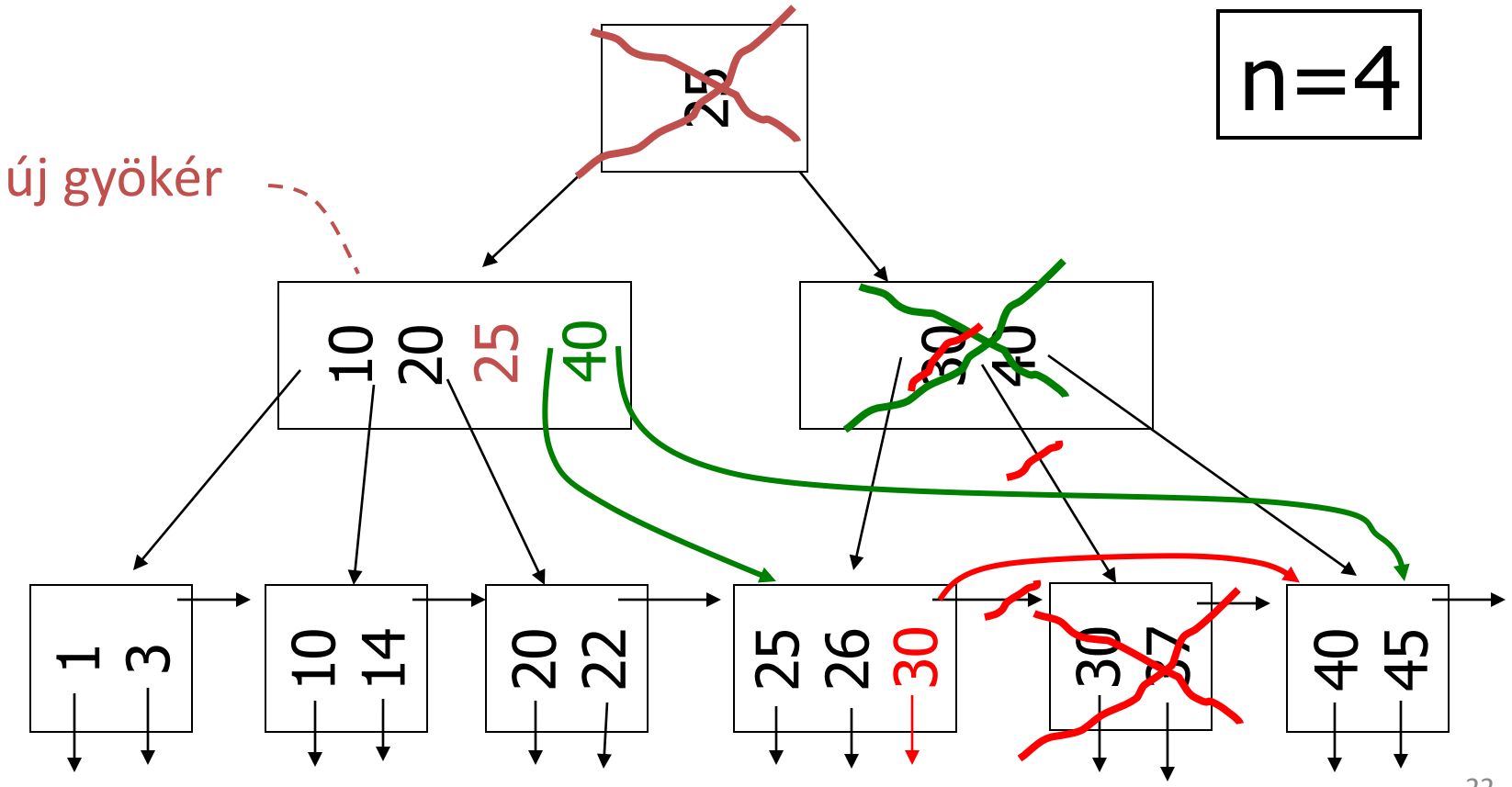
Példák törlésre II.

Töröljük az 50-es indexértékű rekordot!



Törlés III.

Töröljük a 37-es indexértékű rekordot!



Feladat

- Legyen a rekordok száma 1 000 000, és az indexelt oszlopban minden érték különböző. Minden rekordhoz tartozik kulcs a B-fában. Egy blokkba 10 rekord vagy (99 kulcs és 100 mutató) fér. Legyen a telítettség 70%, azaz legalább 69 kulcs és 70 mutató szerepel az indexblokkban. Mekkora az adatfájl és az index együttes mérete legrosszabb esetben? Mennyi a keresés blokkolvasási költsége legrosszabb esetben –adott keresési kulcs-értékkel rendelkező rekord megtalálásához szükséges átlagos lemez I/O műveletek száma?

Bitmap indexek (Bittérkép indexek)

Személy

név	nem	kor	kereset
Péter	férfi	57	350000
Dóra	nő	25	30000
Salamon	férfi	36	350000
Konrád	férfi	21	30000
Erzsébet	nő	20	30000
Zsófia	nő	35	160000
Zsuzsanna	nő	35	160000

érték	vektor
férfi	1011000
nő	0100111

érték	vektor
30000	0101100
160000	0000011
350000	1010000

Bitmap indexek haszna

SELECT COUNT(*)

FROM személy

WHERE nem=‘nő’ and kereset = 160000;

- 0100111 AND 0000011 = 0000011, az eredmény: 2.

SELECT név

FROM személy

WHERE kereset > 100000;

- 0000011 (160000) OR 1010000 (350000) = 1010011, azaz az 1., 3., 6. és 7. rekordokat tartalmazó blokko(ka)t kell beolvasni.

Tömörítés I.

- Ha a táblában n rekord van, a vizsgált attribútum pedig m különböző értéket vehet fel, ekkor, ha m nagy, a bitmap index túl naggyá is nőhet (n*m méret). Ebben az esetben viszont a bitmap indexben az egyes értékekhez tartozó rekordokban kevés az 1-es. A tömörítési technikák általában csak ezeknek az 1-eseknek a helyét határozzák meg.
- Tegyük fel, hogy i db 0-t követ egy 1-es. Legegyszerűbb megoldásnak tűnik, ha i-t binárisan kódoljuk. Ám ez a megoldás még nem jó: (a 000101 és 010001 vektorok kódolása is 111 lenne).
- Tegyük fel, hogy i binárisan ábrázolva j bitből áll. Ekkor először írjunk le j-1 db 1-est, majd egy 0-t, és csak ez után i bináris kódolását.
- Példa: a 000101 kódolása: 101101, a 010001 kódolása: 011011.

Tömörítés II.

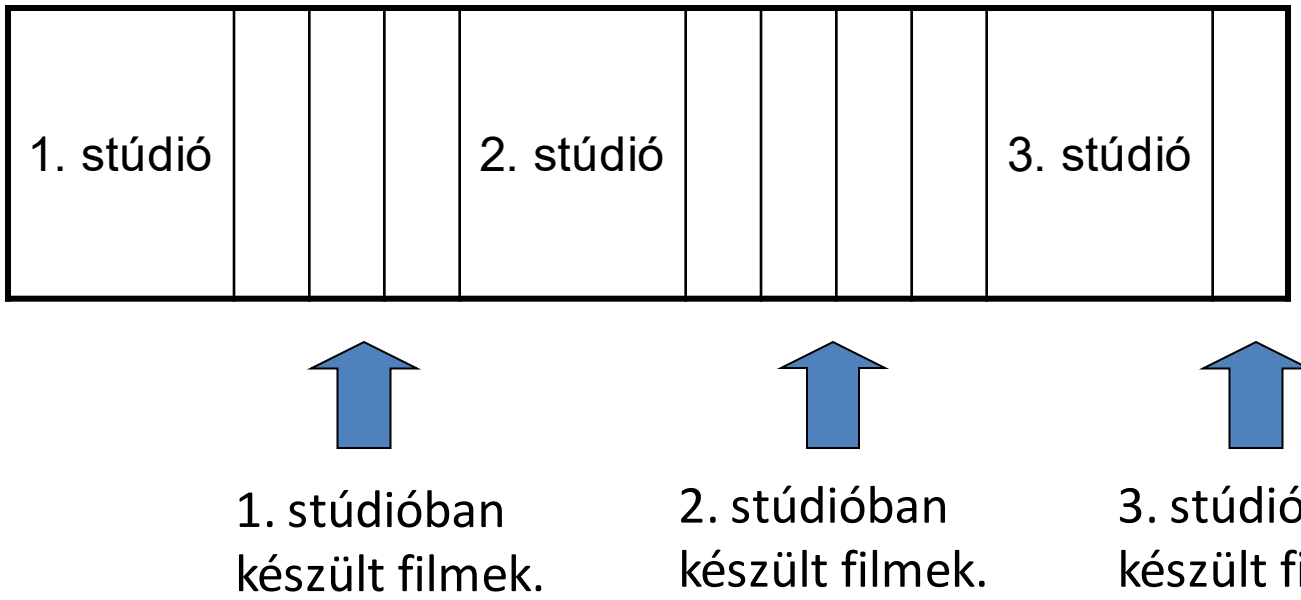
- Állítás: a kód így egyértelművé válik.
- Tegyük fel, hogy m=n, azaz minden rekordérték különböző a vizsgált attribútumban.
- Ekkor, mivel a bitmap indexekben n hosszú rekordokról van szó, egy rekord kódolása legfeljebb $2log_2n$. Az indexet alkotó teljes bitek száma pedig $2\underline{n}log_2n$, n^2 helyett.
- *(j ~ log_2i; length i_2= log_2i)*

Klaszter

- Klaszterszervezés esetén a két tábla közös oszlopain megegyező sorok egy blokkban, vagy fizikailag egymás utáni blokkokban helyezkednek el.
- CÉL: összekapcsolás esetén az összetartozó sorokat soros beolvasással megkaphatjuk.

Példa klaszterre

- Film (cím, év, hossz, stúdiónév)
- Stúdió (név, cím, elnök)



Pl. gyakori a: SELECT cím, év
FROM film, stúdió
WHERE cím LIKE '%Moszkva%' AND stúdiónév = név; jellegű lekérdezés.