ELLENŐRZŐ KÉRDÉSEK

1. Mit hívunk statikus, és mit dinamikus adatbázisnak? (1 pont)
   - Az adatbázis-alkalmazások alapján az adatbázis lehet:
     - Statikus: ritkán módosul, a lekérdezések gyorsasága a fontosabb
     - Dinamikus: gyakran módosul, ritkán végzünk lekérdezést

2. Fogalmazzunk meg 3 célt, amire az indexelés kiválasztásánál figyelni kell! (3 pont)
   - Célok:
     - gyors lekérdezés
     - gyors adatmódosítás
     - minél kisebb tárolási terület
   - Nincs általánosan legjobb optimalizáció
     - az egyik cél a másik rovására javítható
     - pl.: indexek használtaával csökken a keresési idő, nő a tárméret és nő a módosítási idő

3. Mit tételezünk fel, mivel arányos a beolvasás, kiírás költsége? (1 pont)
   - Feltételezzük, hogy a beolvasás, kiírás költsége arányos a háttértároló és memória között mozgatott blokkok számával.
     - ezáltal mérjük a költségeket
     - az író-olvasó fej nagyobb adategységeket, blokkokat olvas be
     - a blokkméret függhet az operációs rendszertől, hardvertől, adabáziskezelőtől
     - Oracle esetén 8K az alapértelmezés

4. Adjuk meg az alábbi paraméterek jelentését! l, b, B, T, bf, M, I(A) (7 pont)
   - A költségek méréséhez bevezetett paraméterek
     - l (lenght): rekordméret bájtokban
     - b: blokkméret bájtokban
     - T (tuple): rekordok száma
     - B: a fájl mérete blokkokban
       - $B = \lceil T/bf \rceil$
     - bf: blokkolási faktor
       - mennyi rekord fér el egy blokkban
       - $bf = \lfloor b/l \rfloor$ (alsó egészrész)
     - M: memória mérete blokkokban
     - I(A): képméret
       - az A oszlopban szereplő különböző értékek száma képméret, melyet I(A)-val jelölünk
       - $I(A) = |\Pi_A(R)|$
     - Példa: RxS mérete mekkora?
       - l(RxS) = l(R) + l(S)
       - T(RxS) = T(R) * T(S)
       - bf(RxS) = b / (l(R) + l(S))
       - B(RxS) = (T(R) * T(S)) * (l(R) * l(S)) / b

5. Adjuk meg RxS méretét blokkokban kifejezve! (2 pont)
   - B(RxS) = (T(R) * T(S)) * (l(R) * l(S)) / b =
     = (T(R) * T(S) * l(R) / b) + (T(R) * T(S) * l(S) /b) =
     = T(S) * B(R) + T(R) * B(S)

6. Mit jelent az egyenletességi feltétel? (1 pont)

- Egyenletességi feltétel: feltesszük, hogy az A = a feltételnek eleget tevő rekordokból nagyjából egyforma számú rekord szerepel.
    - 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.

7. Mekkora adategységet olvas az író-olvasó fej? (1 pont)

- Az író-olvasó fej nagyobb adategységeket, blokkokat olvas be.

8. Mitől függhet a blokkméret? (1 pont)

- A blokkméret függhet
    - az operációs rendszertől
    - a hardvertől
    - az adatbáziskezelőtől

9. Egyenletességi feltétel esetén hány blokkból áll a $\sigma_{A=a}(R)$ lekérdezés eredménye? (1 pont)

- $B(\sigma_{A=a}(R)) = B(R)/I(A)$

10. Soroljunk fel legalább 7 különböző fájlszervezési módszert? (7 pont)

- Kupac (heap)
- Hasító index (hash)
- Rendezett állomány
- Elsődleges index (=ritka index)
- Másodlagos index (=sűrű index)
- Többszintű index
- B^+-fa, B^*-fa

11. Kupac szervezés esetén mennyi a keresés költsége legrosszabb esetben? (1 pont)

- A=a keresési idő a legrosszabb esetben: B (tárméret, avagy a fájl mérete blokkokban)
- Megjegyzések:
    - A=a keresési idő B/2 átlagos esetben
    - Kupac szervezés: a rekordokat a blokk első üres helyére tesszük a beérkezés sorrendjében

12. Kupac szervezés esetén mennyi a beszúrás költsége? (1 pont)

- Beszúrás:
    - utolsó blokkba tesszük a rekordot: 1 olvasás + 1 írás
    - módosítás: 1 keresés + 1 írás
    - törlés: 1 keresés + 1 írás (üres heléy marad, vagy a törlési bitet állítják át)

13. Mit mond meg a h(x) hasító függvény értéke? (1 pont)

- Egy $h(x) \in \{1, .., K\}$ hasító függvény értéke mondja meg, hogy melyik kosárba tartozik a rekord, ha x volt az indexmező értéke a rekordban.
- Megjegyzés: Hasítóindex-szervezés (Hashelés)
    - a rekordokat blokkláncokba (bucket – kosár) soroljuk és a blokklánc utolsó blokkjának első üres helyére tesszük a rekordot a beérkezés sorrendjében.
    - a blokkláncok száma
        - előre adott: K (statikus hasítás)

        - a tárolt adatok alapján változhat (dinamikus hasítás)
    - a besorolás az indexmező értékei alapján történik
    - a hasító függvény általában maradékos osztáson alapul, pl. mod(K)

14. Mikor jó egy hasító függvény és ilyenkor milyen hosszúak a blokkláncok? (2 pont)
    - Akkor jó egy hasító függvény, ha nagyjából egyforma hosszú blokkláncok keletkeznek, azaz egyenletesen sorolja be a rekordokat.
    - Jó hasítófüggvény esetén a blokklánc B/K blokkból áll. (összes blokk/blokkláncok száma)

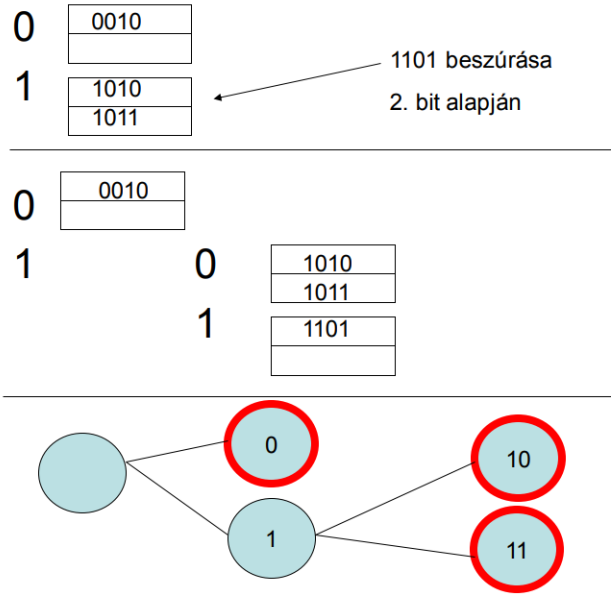
15. Mennyi a $\sigma_{A=a}(R)$ lekérdezés keresési költsége jó hasító index esetén? (1 pont)
    - Keresés (A=a)
        - ha az indexmező és a keresési mező eltér, akkor kupac szervezést jelent
        - ha az indexmező és a keresési mező megegyezik, akkor csak elég a h(a) sorszámú kosarat végignézni, amely B/K blokkból álló kupacnak felel meg (jó hasító index esetén), azaz B/K legrosszabb esetben → a keresés K-szorosára gyorsul

16. Ha túl nagynak választjuk a K-t hasításkor, akkor ez milyen problémát okozhat? (1 pont)
    - Nagy K esetén sok olyan blokklánc lehet, amely egy blokkból fog állni, és a blokkban is csak egy rekord lesz.
    - Ekkor a keresési idő: 1 blokkbeolvasás, de B helyett T számú blokkban tároljuk az adatokat.
        - B << T
            - T: rekordok száma egy fájlban
            - B: blokkok száma egy fájlban (egy blokk rekordokat tartalmaz)

17. Milyen keresésre nem jó a hasító indexelés? (1 pont)
    - Intervallumos (a < A < b) típusú keresésre nem jó.

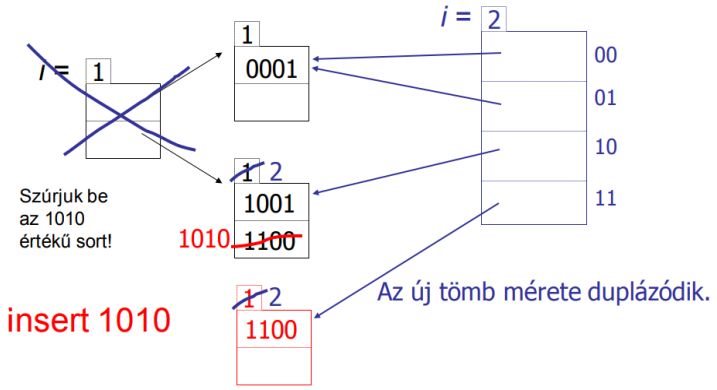
18. Mit jelent a dinamikus hasító indexelés és milyen két fajtáját ismerjük? (3 pont)
    - Előre nem rögzítjük a kosarak számát, a kosarak száma beszúráskor, törléskor változhat.
    - Dinamikus hasító indexek:
        - kiterjeszthető (expendable)
        - lineáris

19. Kiterjeszthető hasítás esetén a h(K) érték alapján melyik kosárba kerül a rekord? (2 pont)
    - Megjegyzés: Kiterjeszthető hasító index
        - Minden kosár 1 blokkból áll. Keresési költség: 1.
        - Legyen k > log(a rekordok várható számának felső korlátja)
          → azaz k hosszú bináris sorozatból több van, mint ahány rekord
        - A h hasító függvény értéke egy k hosszú bináris sorozat (kódszó).
        - A kosarakhoz rendelt kód prefix kód. A maximális kód hossza legyen i.
    - A h(K) k hosszú kódnak vegyük az i hosszú elejét, és azt a kosarat, amelynek kódja a h(K) kezdő szelete.
        - Ha van hely a kosárban: tegyük bele a rekordot.
        - Ha nincs hely a kosárban: nyissunk meg egy új kosarat, és a következő bit alapján osszuk ketté a telített kosár rekordjait.
            - ha ez a bit mindegyike megegyezik, akkor a következő bitet vesszük a szétosztáshoz, és így tovább



20. Milyen probléma keletkezhet kiterjeszthető hasító index esetén és mi rá a megoldás? (2 pont)

- Megjegyzés:
  - A bináris fa levelei a kosárblokkok kódszavai. A hasító függvény értékéből annyi bitet használunk, ahányadik szinten szerepel a levél.
  - A gráfot a memóriában tároljuk.
- Probléma: Ha az új sorok hasító értékének eleje sok bitben megegyezik, akkor hosszú ágak keletkezhetnek.
  → nincs kiegyensúlyozva a fa
- Megoldás:
  - A bináris gráfot teljessé is tehetjük.
  - A gráfot egy tömbbel ábrázoljuk.
  - Ekkor minden kosár azonos szinten lesz, de közös blokkjai is lehetnek a kosaraknak.
  - Túlcsordulás esetén a kosarak száma duplázódik.
  - Legyen például h(k) 4 bites és 2 rekord férjen el egy blokkba. Az i jelzi, hogy hány bitet használunk fel.



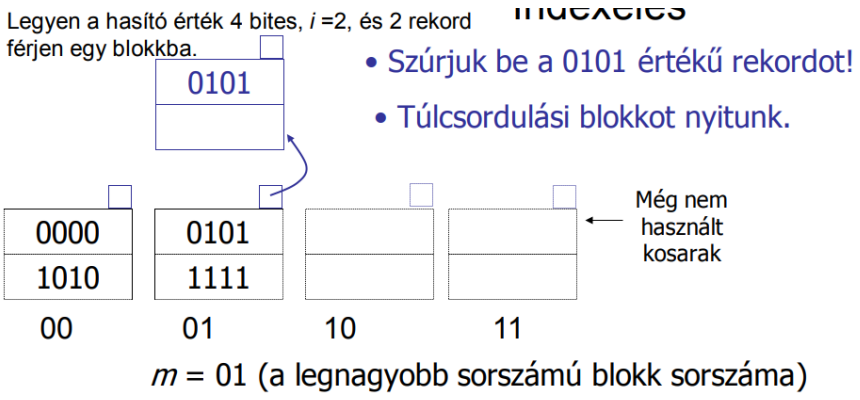
- Példa: 14-16. dia (fizika.ppt)

21. Lineáris hasító index esetén mikor nyitunk meg új kosarat? (1 pont)

- Lineáris hasító index:
  - A kosarak 1 vagy több blokkból is állhatnak.
  - Új kosarat akkor nyitunk meg, ha egy előre megadott értéket elér a kosarakra jutó átlagos rekordszám.
    → ($\frac{\text{rekordok száma}}{\text{kosarak száma}} >$ küszöb)
  - A kosarakat 0-tól kezdve sorszámozzuk, és a sorszámot binárisan ábrázoljuk.

22. Lineáris hasító index esetén a h(K) érték alapján melyik kosárba kerül a rekord? (2 pont)

- Ha n kosarunk van, akkor a hasító függvény értékének utolsó log(n) bitjével megegyező sorszámú kosárba tesszük, ha van benne hely.
  Ha nincs benne hely, akkor hozzáláncolunk egy új blokkot, és abba tesszük.
- Ha nincs megfelelő sorszámú kosár, akkor abba a sorszámú kosárba tesszük, amely csak az első bitjában különbözik a keresett sorszámtól.



23. Rendezett állomány esetén adjuk meg a bináris (logaritmikus) keresés lépéseit! (4 pont)

1. Beolvassuk a középső blokkot.
2. Ha nincs benne az A=a értékű rekord, akkor eldöntjük, hogy a blokklánc második felében, vagy az első felében szerepelhet-e egyáltalán
3. Beolvassuk a felezett blokklánc középső blokkját
4. Addig folytatjuk, amíg megtaláljuk a rekordot, vagy a vizsgálandó maradék blokklánc már csak 1 blokkból áll.

24. Mennyi a keresési költség rendezett mező esetében? (1 pont)

- Keresési idő: $log_2(B)$

25. Mennyi a keresési költség rendezett mező esetében, ha gyűjtő blokkokat is használunk? (1 pont)
- Megjegyzés:
  - Beszúrás: keresés + üres hely készítése miatt a rekordok eltolása az összes blokkban, az adott találati blokktól kezdve (B/2 blokkot be kell olvasni, majd az eltolások után visszaírni = B művelet)
  - Szokásos megoldások:
    - Gyűjtő (túlcsordulási) blokk
    - Üres helyeket hagyunk a blokkokban
- Gyűjtő (túlcsordulási blokk) használata:
  - az új rekordok számára nyitunk egy blokkot, ha betelik, hozzáláncolunk egy újabb blokkot
  - keresést 2 helyen végezzük:
    - $log_2(B-G)$ költséggel keresünk a rendezett részben
    - ha nem találjuk, akkor a gyűjtőben is megnézzük (G blokkművelet, ahol G a gyűjtő mérete)
    - azaz az ÖSSZKÖLTSÉG: $\boldsymbol{log_2(B-G)+G}$
  - ha a G túl nagy a $log_2(B)$-hez képest, akkor újrarendezzük a teljes fájlt (a rendezés költsége $B \cdot log_2(B)$)

26. Mennyi a keresési költség rendezett mező esetében, ha minden blokkot félig üresen hagyunk? (1 pont)
- Üres helyeket hagyunk a blokkokban:
  - például félig üresek a blokkok
  - a keresés után 1 blokkművelettel visszaírjuk a blokkot, amibe beírtuk az új rekordot
  - tárméret $2 \cdot B$ lesz
  - keresési idő: $log_2(2 \cdot B) = \boldsymbol{1 + log_2(B)}$
  - ha betelik egy blokk, vagy elér egy határt a telítettsége, akkor 2 blokkba osztjuk szét a rekordjait, a rendezettség fenntartásával

27. Milyen mindig az indexrekord szerkezete? (1 pont)
- 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
  - az index mindig rendezett az indexértékek szerint

28. Adjuk meg az elsődleges index 5 jellemzőjét! (5 pont)
- főfájl is rendezett
- csak 1 elsődleges indexet lehet megadni (mert csak egyik mező szerint lehet rendezett a főfájl)
- elég a főfájl minden blokkjának legkisebb rekordjához készíteni indexrekordot
- indexrekordok száma: T(I) = B (ritka index)
- indexrekordokból mindig sokkal több fér el egy blokkba, mint a főfájl rekordjaiból
  - bf(I) >> bf – azaz az indexfájl sokkal kisebb rendezett fájl, mint a főfájl
  - B(I) = B / bf(I) << B = T / bf

29. Mit hívunk fedőértéknek? (1 pont)
- Az indexfájlban nem szerepel minden érték, ezért csak fedő értékeket kereshetünk
- Fedő érték: a legnagyobb olyan indexérték, amely a keresett értéknél kisebb vagy egyenlő

30. Mennyi a keresési költség elsődleges index esetén? (1 pont)
- fedő érték keresése az index rendezettsége miatt bináris kereséssel történik: $log_2(B(I))$
- a fedő indexrekordban szereplő blokkmutatónak megfelelő blokkot még be kell olvasni: $\boldsymbol{1 + log_2(B(I))}$

31. Adjuk meg a másodlagos index 5 jellemzőjét! (5 pont)
    - főfájl rendezetlen (az indexfájl mindig rendezett)
    - több másodlagos indexet is meg lehet adni
    - a főfájl minden rekordjához kell készíteni indexrekordot
    - indexrekordok száma: T(I) = T (sűrű index)
    - indexrekordból sokkal több fér el egy blokkba, mint a főfájl rekordjaiból
        - bf(I) >> bf – azaz az indexfájl sokkal kisebb rendezett fájl, mint a főfájl
        - B(I) = T / bf(I) << B = T / bf

32. Hogyan keresünk a másodlagos indexben és mennyi a keresés költsége? (5 pont)
    - Az indexben keresés az index rendezettsége miatt bináris kereséssel történik: $log_2(B(I))$
    - A talált indexrekordban szereplő blokkmutatónak megfelelő blokkot még be kell olvasni, így : $\mathbf{1 + log_2(B(I))}$
    - $1 + log_2\big(B(I)\big) \ll log_2(B)$ (rendezett eset)
    - Az elsődleges indexnél rosszabb a keresési idő, mert több az indexrekord

33. Mit hívunk klaszterszervezésű táblának? (1 pont)
    - Megjegyzés: Klaszter (nyaláb, fürt) → azonos indexű értékek (blokkok) fizikailag egymás után helyezkednek el
    - Klaszterszervezés egy tábla egy A oszlopra:
        - az azonos A értékű sorok fizikailag egymás után blokkokban helyezkednek el
        - Cél: az eslő találat után az összes találatot megkapjuk soros beolvasással

34. Mit hívunk klaszterindexnek? (1 pont)
    - Klaszterindex:
        - klaszterszervezésű fájl esetén index az A oszlopra

35. Mikor mondjuk, hogy 2 tábla klaszterszervezésű? (1 pont)
    - Klaszterszervezés két tábla esetén az összes közös oszlopra:
        - a közös oszlopokon egyező sorok egy blokkban, vagy fizikailag egymás utáni blokkokban helyezkednek el
        - Cél: összekapcsolás esetén az összetartozó sorokat soros beolvasással megkaphatjuk

36. Ha t szintű indexet használunk, mennyi a keresési költség blokkműveletek számában mérve? (1 pont)
    - Megjegyzés: többindexű index
        - az indexfájl (1. indexszint) is fájl, ráadásul rendezett, így ezt is meg lehet indexelni, elsődleges indexszel
        - a főfájl lehet rendezett vagy rendezetlen (az indexfájl mindig rendezett)
        - t-szintű index: az indexszinteket is indexeljük, összesen t szintig
    - Keresési idő:
        - a t-edik szinten ($I^{(t)}$) bináris kereséssel keressük meg a fedő indexrekordot
        - követjük a mutatót, minden szinten, és végül a főfájlban: $\boldsymbol{log_2\left(B\left(I^{(t)}\right)\right) + t}$ **blokkolvasás**
        - ha a megelelő szint 1 blokkból áll, akkor t+1 blokkolvasást jelent (t=?)
        - minden szint blokkolási faktora megegyezik, mert egyforma hosszúak az indexrekordok

37. Ha t szintű indexet használunk, a legfelső szinten milyen keresést használunk? (1 pont)
    - A t-edik szinten ($I^{(t)}$) bináris kereséssel keressük meg a fedő indexrekordot.

38. Ha t szintű indexet használunk és a legfelső szint 1 blokkból áll, akkor mennyi a keresési költség? (1 pont)
    - ha a megelelő szint 1 blokkból áll, akkor **t+1** blokkolvasást jelent (t=?)

39. Ha t szintű indexet használunk, mennyi az indexszintek blokkolási faktora és miért? (2 pont)
    - minden szint blokkolási faktora megegyezik, mert egyforma hosszúak az indexrekordok

40. Ha t szintű indexet használunk, vezessük le, hogy hány blokkból áll a legfelső szint! (12 pont)

	FŐFÁJL	1. szint	2. szint	...	t. szint
blokkok száma	B	B/bf(I)	$B/bf(I)^2$	...	$B/bf(I)^t$
rekordok száma	T	B	B/bf(I)	...	$B/bf(I)^{(t-1)}$
blokkolási faktor	bf	bf(I)	bf(I)	...	bf(I)

- t-ik szinten 1 blokk: $1=B/bf(I)^t$

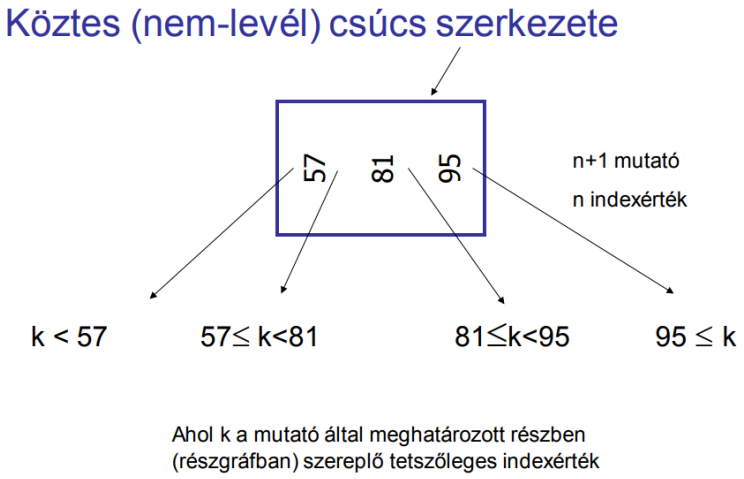
41. Ha t szintű indexet használunk, és a legfelső szint 1 blokkból áll, abból milyen egyenlet következik és mi a megoldása t-re? (2 pont)

- t-edik szinten 1 blokk: $\mathbf{1 = B/bf(I)^t}$

42. Mi a két legfontosabb jellemzője a $\mathbf{B^+}$-faindexnek? (2 pont)

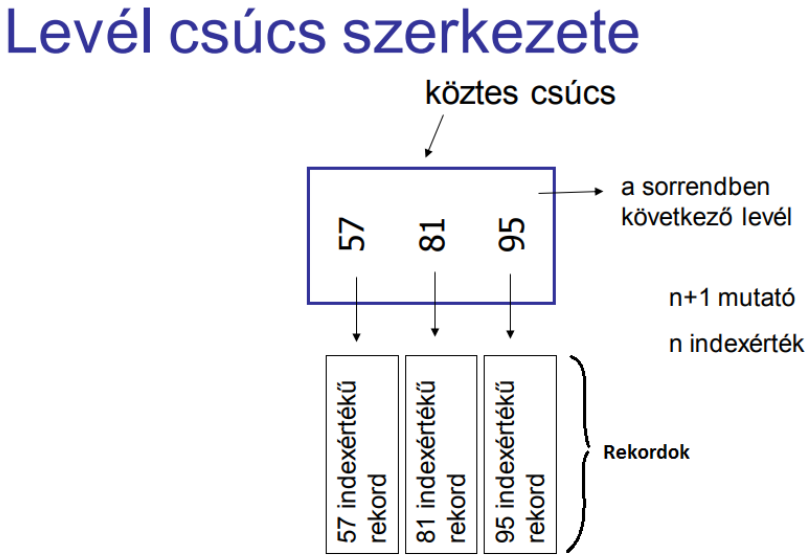
- Megjegyzés: a többszintű indexek közül az egyik legelterjedtebb
- Minden blokk legalább 50%-ban telített
- Telítettséget biztosító karbantartó algoritmusok

43. Egy példa alapján szemléltessük a köztes csúcs jellemzőit $\mathbf{B^+}$-fa index esetén! (8 pont)



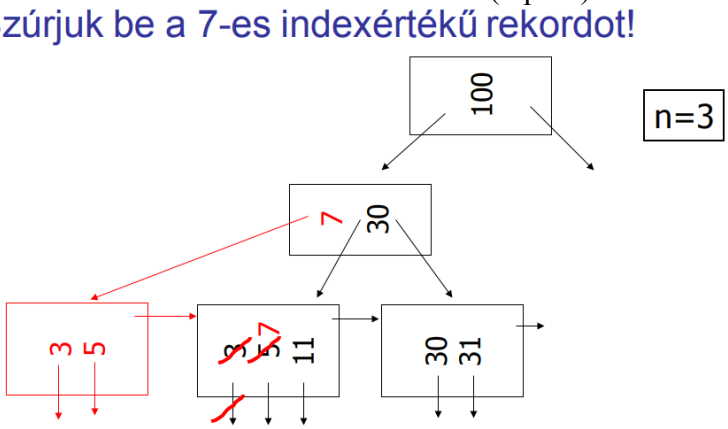
Ahol k a mutató által meghatározott részben
(részgráfban) szereplő tetszőleges indexérték

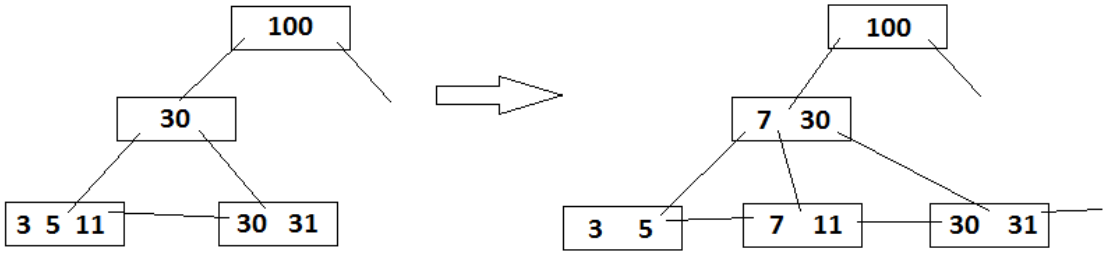
44. Egy példa alapján szemléltessük a levél csúcs jellemzőit $\mathbf{B^+}$-fa index esetén! (5 pont)



45. Mutassunk példát, mikor beszúráskor egy levélcsúcsot kettéosztunk **B**$^+$-fa index esetén! (5 pont)

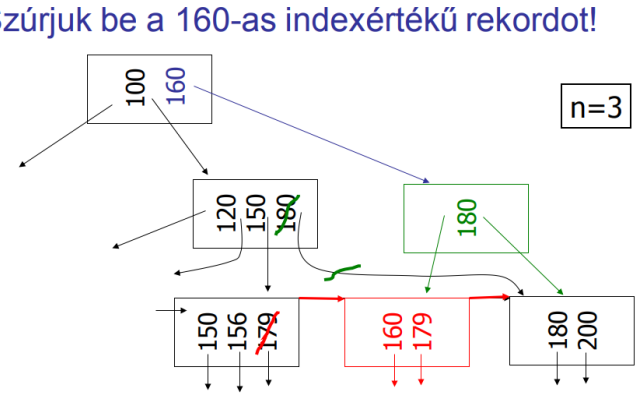
- A 7-es indexű rekord nem férne be alapból, mivel az adott kosár már tele van. ☹
- Tehát kb. fele-fele arányban kettéosztjuk azt a blokkot, ahová a 7-est kellene beszúrnunk, majd belerakjuk a 7-est
- De ezt még a változást a szerkezetbe is bele kell integrálni: legkisebb indexet be kell tenni az egy szinte felette lévő indexek közé. (fedő érték)
- Követelmény: mit szúrunk be és a beszúrás előtti, majd utáni két fát kell lerajzolni:

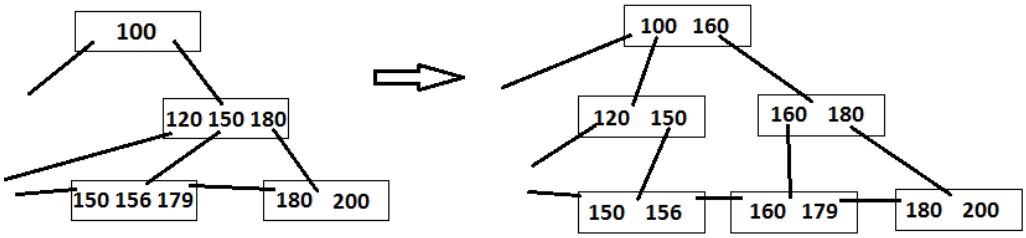




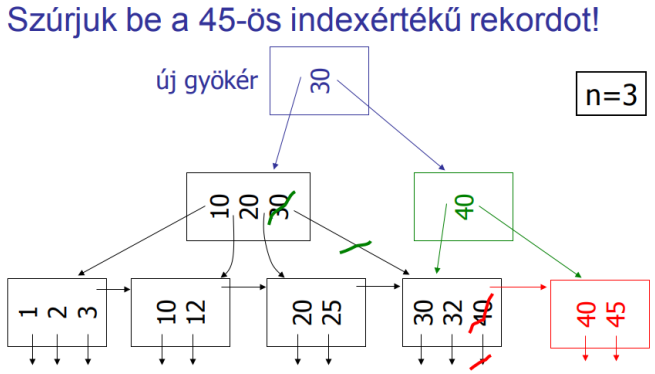
46. Mutassunk példát, mikor beszúráskor egy köztes csúcsot kettéosztunk **B**$^+$-fa index esetén! (5 pont)

- Amikor valamit be kell szúrni, az egyre feljebbi szintek módosításához vezethet
  - ha ketté kell osztani a gyökeret, kell új gyökér
  - nő a fa magassága
- Keresés nagyon gyors
- DE: indexkarbantartás új rekord esetén nagyon költséges (akár nőhet is a fa magassága)
- Követelmény: mit szúrunk be és a beszúrás előtti, majd utáni két fát kell lerajzolni:
-



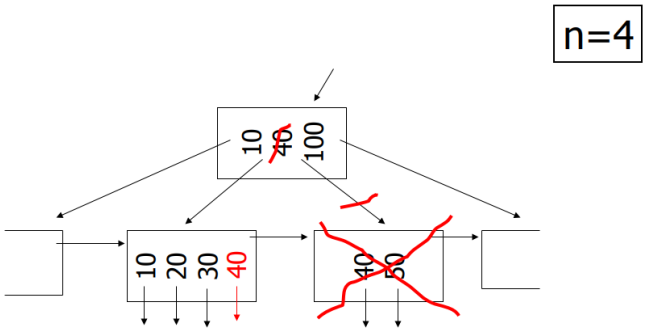


47. Mutassunk példát, mikor beszúráskor nő a **B**$^+$-fa index magassága! (5 pont)



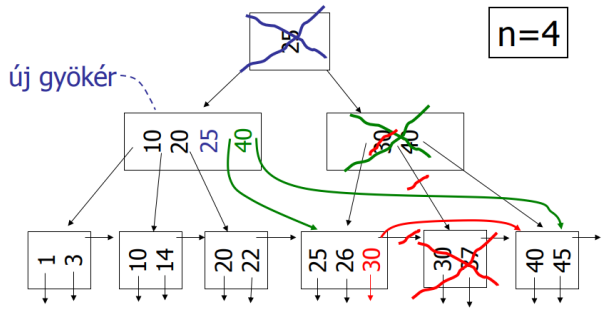
- Követelmény: mit szúrunk be és a beszúrás előtti, majd utáni két fát kell lerajzolni

48. Mutassunk példát, mikor törléskor megszüntetünk egy levélcsúcsot **B**$^+$-fa index esetén! (5 pont)



- Követelmény: mit szúrunk be és a beszúrás előtti, majd utáni két fát kell lerajzolni

49. Mutassunk példát, mikor törléskor csökken a **B**$^+$-fa index magassága! (5 pont)



- Követelmény: mit szúrunk be és a beszúrás előtti, majd utáni két fát kell lerajzolni

50. Mutassunk példát arra, mikor egy kevés elemszámú oszlopra bitmap indexet készítünk! (2 pont)

Bittérkép (bitmap) indexek

CUSTOMER #	MARITAL_ STATUS	REGION	GENDER	INCOME_ LEVEL
101	single	east	male	bracket_1
102	married	central	female	bracket_4
103	married	west	female	bracket_2
104	divorced	west	male	bracket_4
105	single	central	female	bracket_2
106	married	central	female	bracket_3

REGION='east'	REGION='central'	REGION='west'
1	0	0
0	1	0
0	0	1
0	0	1
0	1	0
0	1	0

- Bittérkép: 0-k és 1-ek reprezentálják, hogy a rekordban milyen érték van (true/false)

51. Mutassunk példát arra, mikor logikai feltételek kiértékelését bitmap vektorműveletekre vezetjük vissza! (7 pont)

```
SELECT COUNT(*)
FROM CUSTOMER
WHERE MARITAL_STATUS = 'married' AND REGION
   IN ('central','west');
```

status = 'married'		region = 'central'		region = 'west'						
0		0		0		0		0		0
1		1		0		1		1		1
1	AND	0	OR	1	=	1	AND	1	=	1
0		0		1		0		1		0
0		1		0		0		1		0
1		1		0		1		1		1

- olyan táblát képez, melyekben azoknál a rekordoknál szerepel 1 egy adott oszlopban, melyre a rá vonatkozó feltétel teljesül
- pl.: 0-kat és 1-eseket „éselünk össze”