ELTE | IK

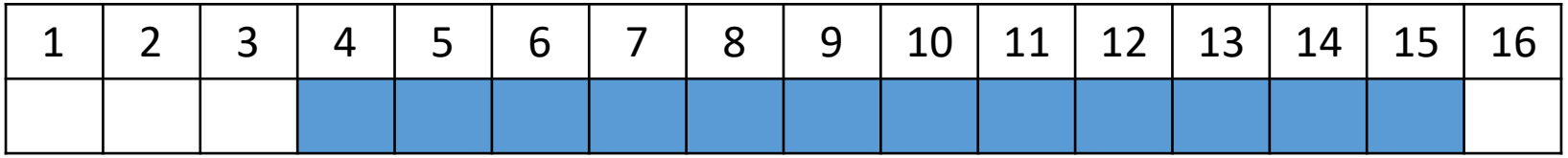
INFORMATIKAI KAR

Adatbázisok 2.

4. gyakorlat

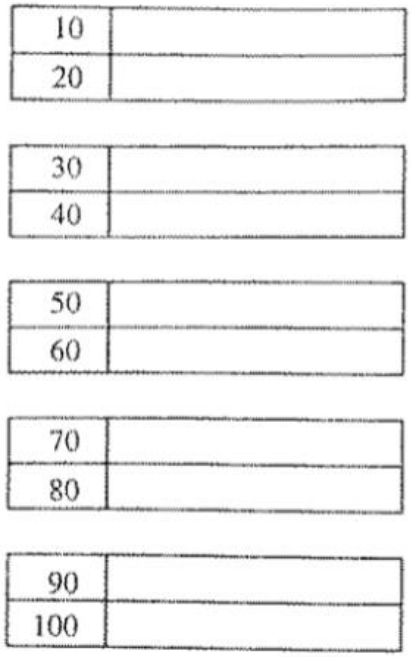
Blokk olvasások száma

- 16 blokkunk van összesen
- A kék blokkokban van tárolva a táblánk (12 blokk)
- Hány blokkot kell beolvasni, ha egy konkrét sort keresünk?

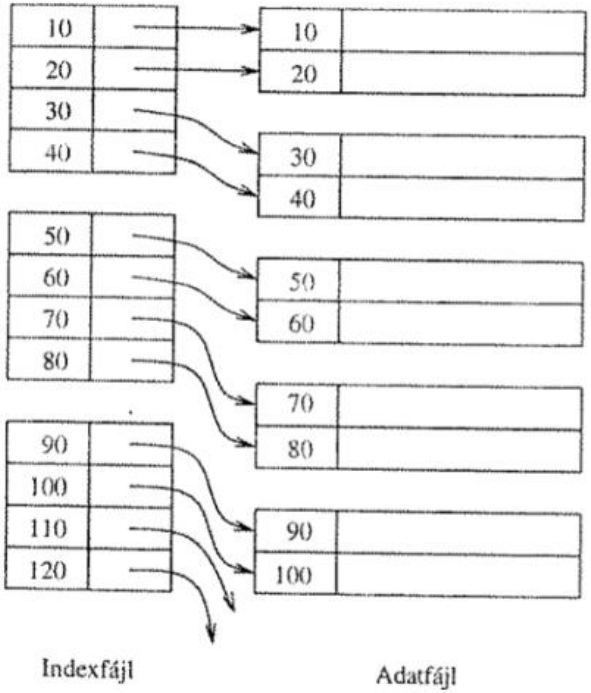


Indexek

- Elsődleges index: feltehetjük, hogy a fájlunk rendezett
- Sűrű index: minden rekordhoz tartozik egy mutató az index fájlban
- Az index kulcs-mutató párjai kevesebb helyet foglalnak, mint a rekordok



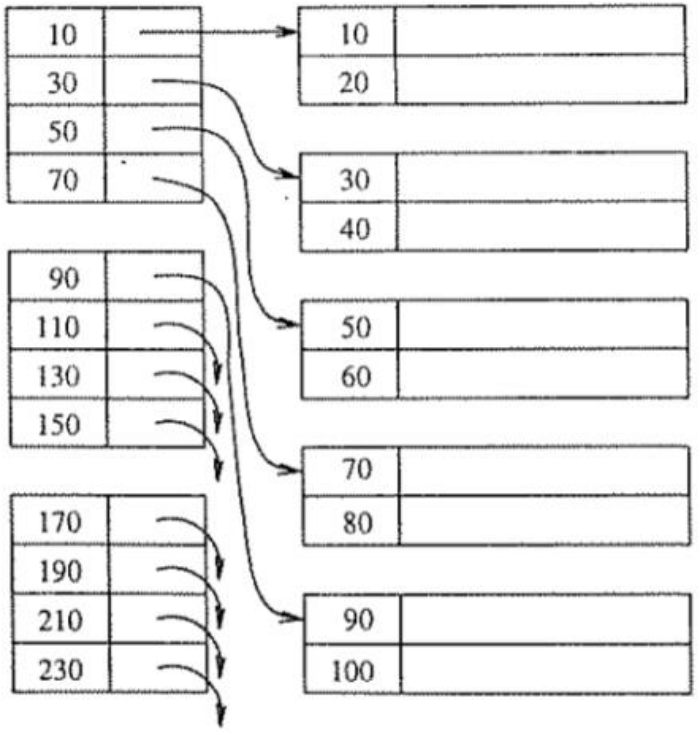
4.2. ábra. *Egy szekvenciális fájl*



4.3. ábra. *Sűrű index (balra) szekvenciális adatfájlon (jobbra)*

Indexek

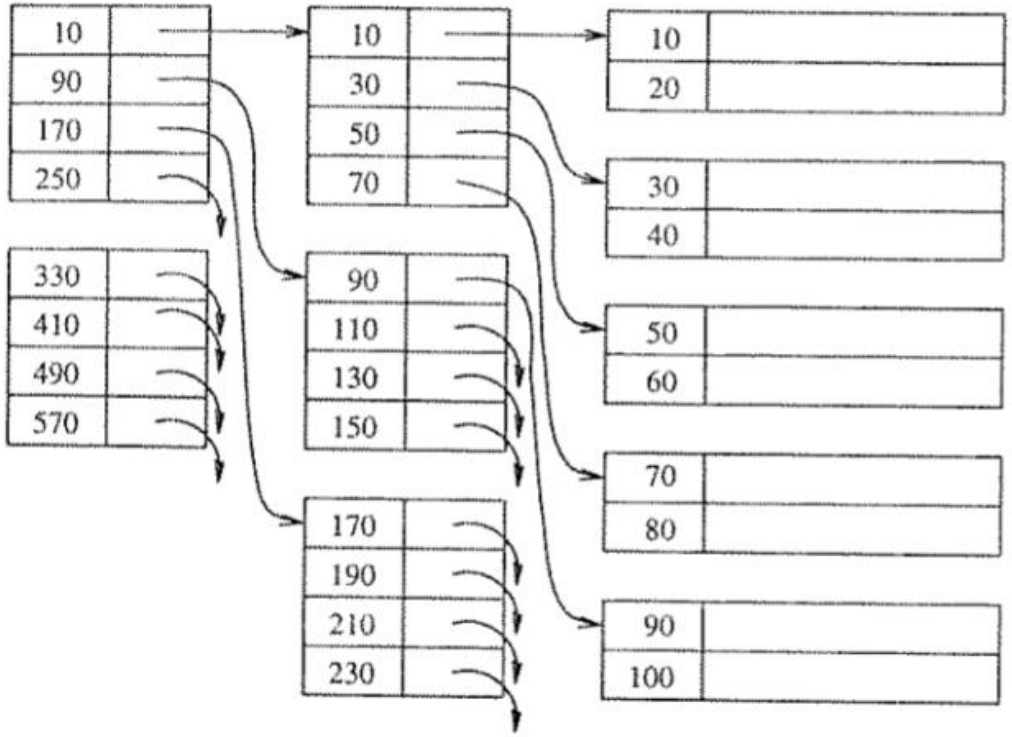
- Ritka index: minden blokkhoz egy mutató tartozik az indexfájlban
- A kulcs az adatblokk első rekordjának a kulcsa



4.4. ábra. *Ritka index szekvenciális fájlon*

Indexek

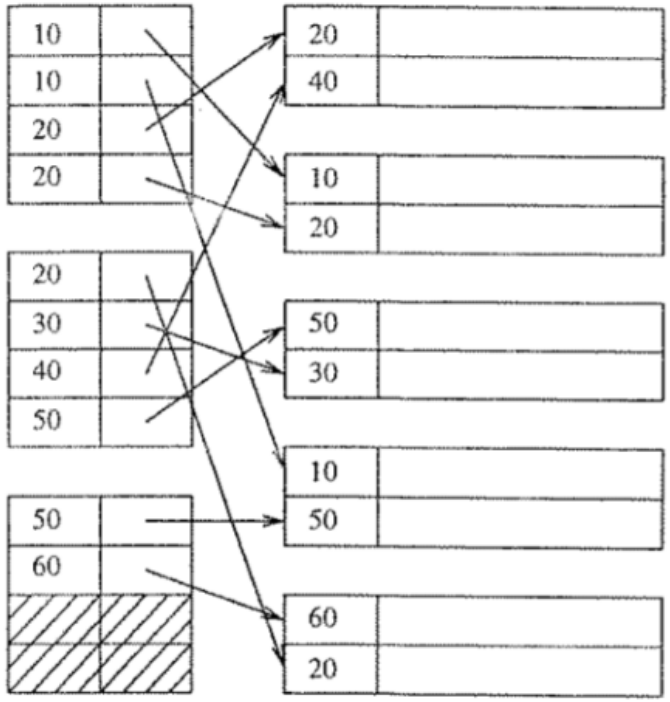
- Több szintű index: ha az indexre indexet készítünk, akkor a rekordok elérését még hatékonyabbá tehetjük.
- A második szintű indextől kezdve az indexek kötelezően ritkák.



4.5. ábra. *Második szintű ritka index készítése*

Indexek

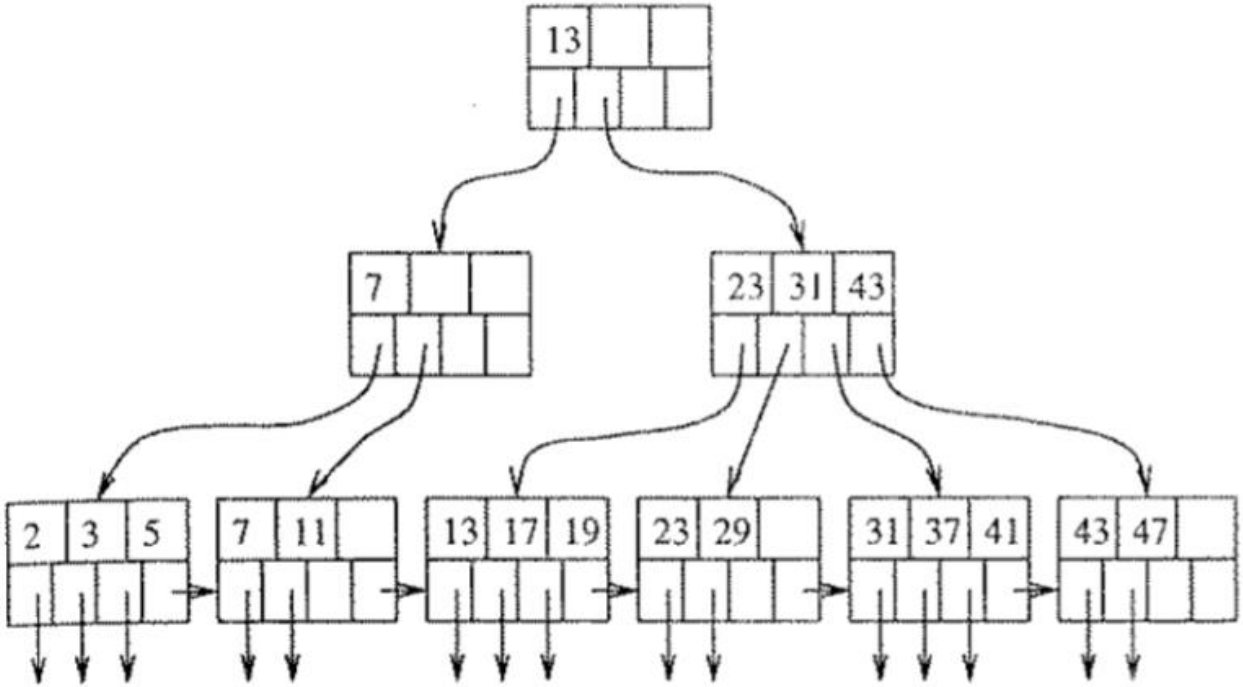
- Másodlagos index: a fájl nem rendezett, ezért sűrűnek kell lennie.
- Mivel nem csak egy mezőre szeretnénk indexeket létrehozni.



4.15. ábra. *Másodlagos index*

B-fák

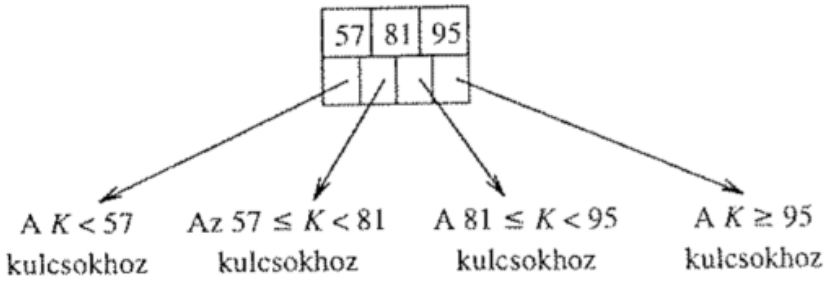
- Adatbázisokban leggyakrabban használt index.
- Fontos paraméter: egy csúcsban lévő kulcsok száma.
- Általában három szintesek.



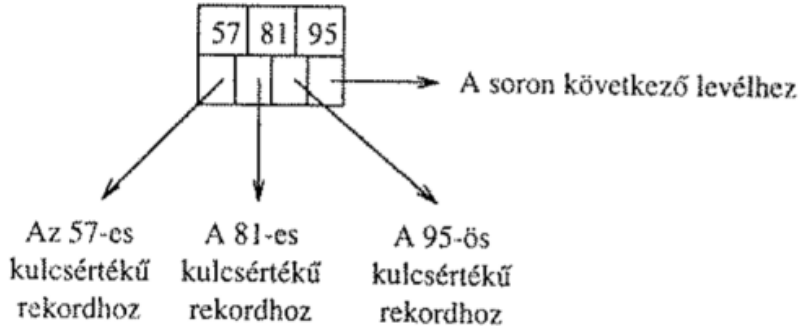
4.23. ábra. *B+-fa*

B-fák

- Egy csúcsban N kulcs és N+1 mutató van.
- Egy levélcsúcs utolsó mutatója a rákövetkező levélcsúcsra mutat.



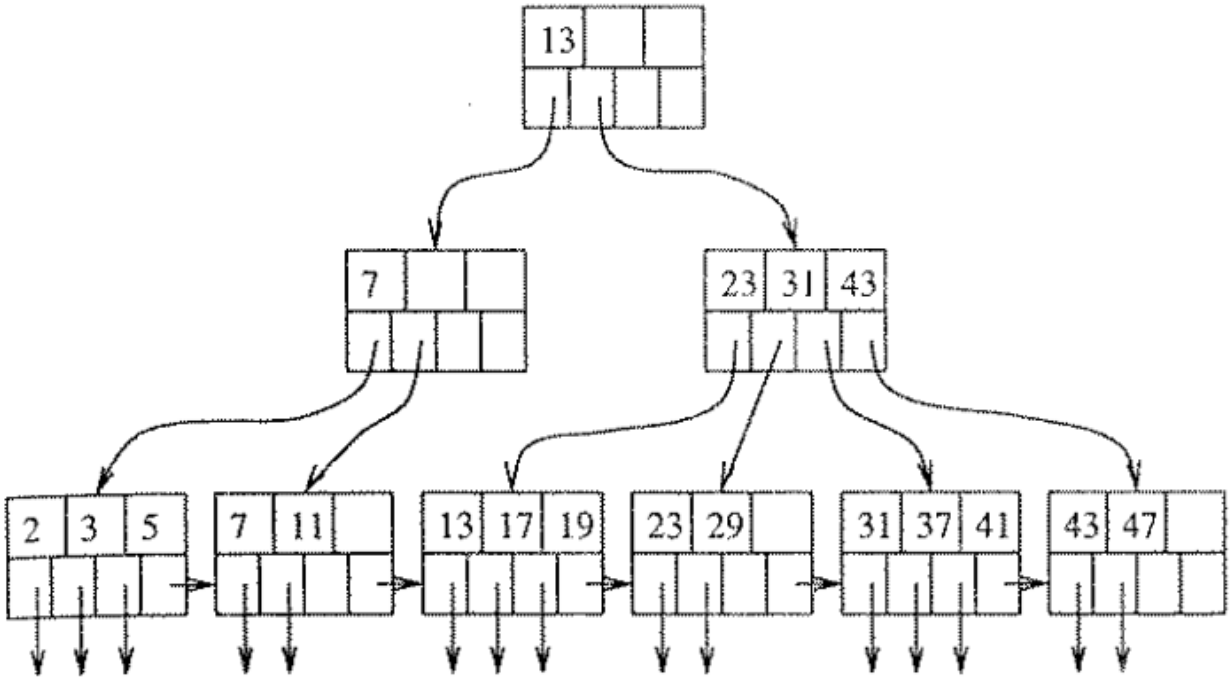
4.22. ábra. *Egy B+-fa jellegzetes belső csúcsa*



4.21. ábra. *Egy B+-fa jellegzetes levele*

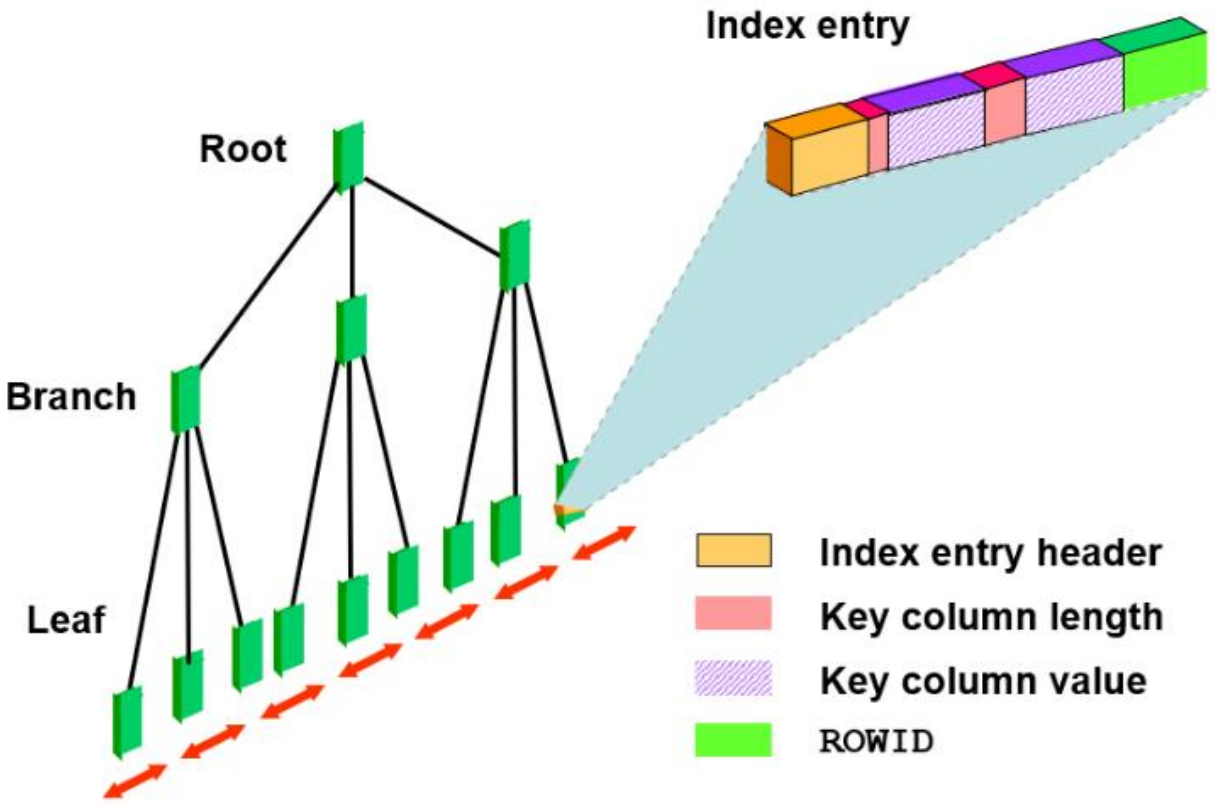
B-fák

- Három szintű fa: a legalsó szint sűrű, a többi ritka.



4.23. ábra. *B+-fa*

B-fák


Index entry
Root
Branch
Leaf
Index entry header
Key column length
Key column value
ROWID

Feladatok

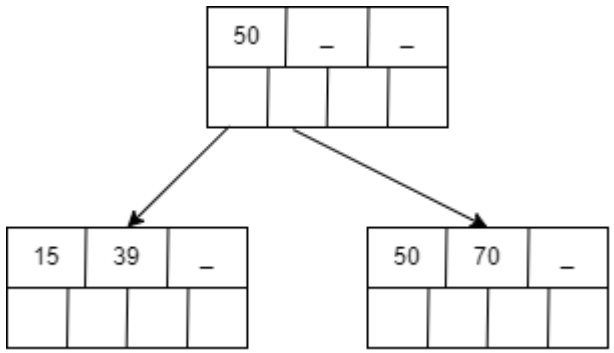
- Jelölések
  - T(R) – az R reláció sorainak a száma.
  - bf(R) – blokkolási faktor, azaz R reláció hány sora fér el egy blokkban. Blokkolási faktorról indexek esetében is beszélhetünk.
  - B(R) – az R reláció sorainak tárolásához szükséges blokkok száma
- 1. feladat
  - Van egy R táblánk, egy I1 sűrű és egy I2 ritka indexünk az alábbi paraméterekkel: T(R) = 10 000, bf(R) = 20, bf(I1) = 100, bf(I2) = 100
  - Számoljuk ki a következőket: B(R), B(I1), B(I2)
- 2. feladat
  - Végezzük el az előző feladatot úgy, hogy a blokkok csak 80%-ban lehetnek tele.

Feladatok

- 3. feladat
  - T(R) = 1 000 000, bf(R) = 20
  - Egy „A” oszlopra készítünk B+ fa indexet, amelyre bf(I) = 50
- Kérdések
  - (1) B(I) = ?
  - (2) Hány blokkot kell beolvasnunk egy konkrét sor megtalálásához a legrosszabb esetben, ha:
    - a) a tábla sorai rendezetlenül vannak tárolva és nem használunk indexet
    - b) a tábla sorai rendezetten vannak tárolva és nem használunk indexet
    - c) a fenti B+ fa indexet használjuk

Feladat: B+ fa építés

- Tegyük fel, hogy egy B+ fa blokkjában 3 kulcs fér el és 4 mutató. A kulcsok különbözőek.
- Szúrjuk be a B+ fába az alábbi kulcsértékeket a megadott sorrendben: 39, 15, 50, 70, 79, 83, 72, 43, 75, 45
- Adjuk meg a B+ fa minden olyan állapotát, amikor egy csomópont kettéosztására volt szükség. Például, az első kettéosztás utáni állapot:



Feladat: B+ fa építés tippek

- Levél csúcs kettéosztásakor minden kulcsot megőrzünk a régi és az új csúcsban. Egy új kulcs-mutató párt küldünk felfelé a szülő csúcsba, amit ott kell elhelyezni.
- Belső csúcs kettéosztásakor (N és M csúcsra) a mutatók első fele az N-be, a második fele az M-be kerül, de középen kimarad egy kulcs, ami az M-en keresztül elérhető legkisebb kulcsot tartalmazza. Ez nem kerül sem N-be, sem M-be, hanem felmegy M és N közös szülőjébe az M-re mutató mutatóval együtt.

Bitvektor indexek

- Egy oszlopra N hosszú bitvektorokat készítünk, ahol N a sorok száma és annyi bitvektorunk lesz, ahány különböző érték előfordul az oszlopban.

Neptun	Kar	Felvétel éve
ABC123	IK	2018
XYZ789	TTK	2019
ASD135	IK	2020
GOT999	IK	2019

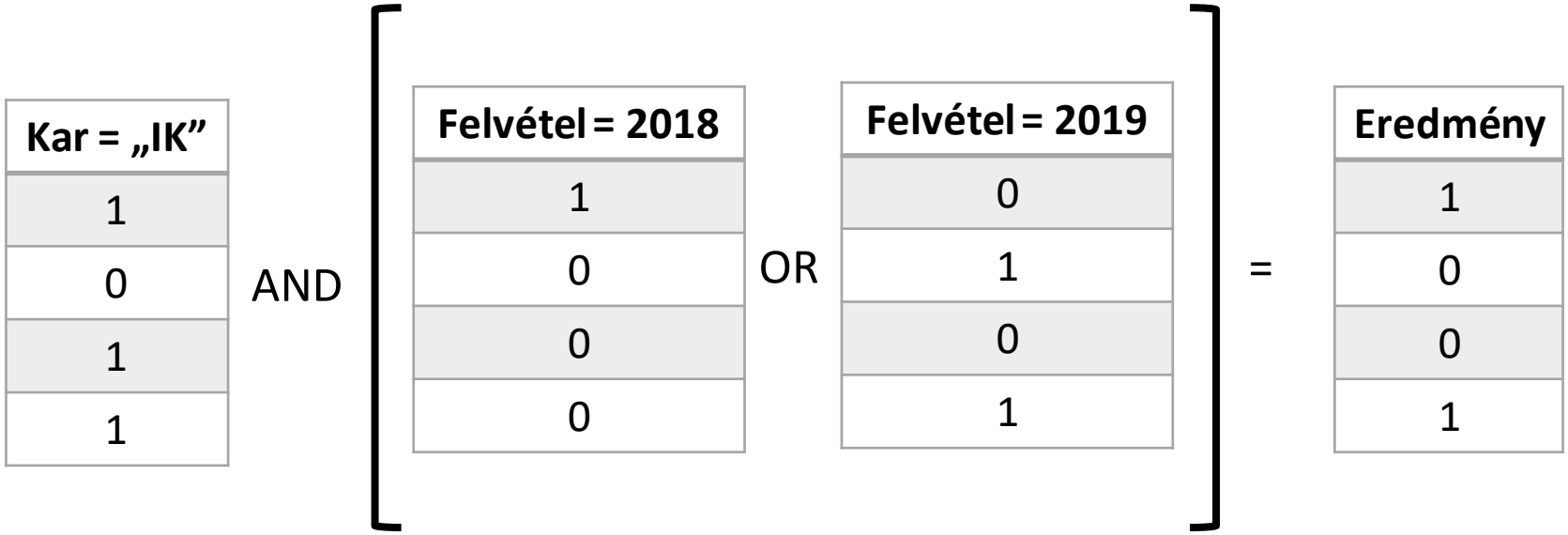
Kar = „IK”	Kar = „TTK”
1	0
0	1
1	0
1	0

Felvétel = 2018	Felvétel = 2019	Felvétel = 2020
1	0	0
0	1	0
0	0	1
0	1	0

Bitvektor indexek

- SELECT * FROM HALLGATOK

  WHERE KAR=„IK” AND FELVÉTEL_ÉVE IN (2018, 2019);



Bitvektor feladatok

- Tegyük fel, hogy a FOGLALKOZÁS, BELÉPÉS és OAZON oszlopokra létezik BITMAP index. Készítsük el a szükséges bitvektorokat és válaszoljuk meg a segítségükkel a lekérdezéseket.
- (1) Adjuk meg azoknak a dolgozóknak a nevét, akik 1981-ben léptek be és a foglalkozásuk hivatalnok (CLERK), vagy a 20-as osztályon dolgoznak és a foglalkozásuk MANAGER.

DNEV	FOGLALKOZAS	BELEPES	OAZON
SMITH	CLERK	1980	20
ALLEN	SALESMAN	1981	30
WARD	CLERK	1981	30
JONES	MANAGER	1983	20
MARTIN	SALESMAN	1981	30
BLAKE	MANAGER	1983	30
CLARK	MANAGER	1981	10
SCOTT	ANALYST	1982	20
KING	PRESIDENT	1981	10
TURNER	CLERK	1983	30

Bitvektor feladatok

- (2) Adjuk meg azoknak a dolgozóknak a nevét, akik nem 1981-ben léptek be és a 10-es vagy 30-as osztályon dolgoznak.

DNEV	FOGLALKOZAS	BELEPES	OAZON
SMITH	CLERK	1980	20
ALLEN	SALESMAN	1981	30
WARD	CLERK	1981	30
JONES	MANAGER	1983	20
MARTIN	SALESMAN	1981	30
BLAKE	MANAGER	1983	30
CLARK	MANAGER	1981	10
SCOTT	ANALYST	1982	20
KING	PRESIDENT	1981	10
TURNER	CLERK	1983	30

Szakaszhossz kódolás

- A BITMAP indexek sok nullásból és kevés egyesből állnak.
- Ha egy N rekordot tartalmazó tábla egy oszlopában M különböző érték van, akkor arra az oszlopra készített bitvektorok N*M bitet foglalnak.
- A 0-ásokból álló i hosszúságú szakaszokat kódoljuk:

0000000100000000000000010010000000100000

- Algoritmus:
  - Meghatározzuk, hogy az i binárisan kódolva hány bitből áll (ez lesz a j szám).
  - Ezt unárisan ábrázoljuk: j-1 darab 1-es aztán pedig egy 0-s.
  - Ezután hozzáfűzzük az i értékét binárisan.

Szakaszhossz kódolás példa

- Bitvektor: 00000000000001
- 13 db nullából áll
- Azaz i=13, ami binárisan 1101
- Ebből következik, hogy j=4, ami unárisan 1110
- Így a kódolt szakasz: 11101101

Megjegyzések:

- i=0 kódolva: 00
- i=1 kódolva: 01
- Feltételezzük, hogy egy kód mindig nullásokkal kezdődik, azaz ha a 100001 bitvektort kell kódolni, akkor először 0 db nullást kódolunk, ami 00 és csak utána a 4 db nullást.

Szakaszhossz kódolás feladat

- Tömörítsük a következő bitvektort szakaszhossz kódolással:
  00000000000000000000010000000101
  (23 db nulla, 7 darab nulla)
- Fejtsük vissza a következő, szakaszhossz kódolással tömörített bitvektort:
  1111010101001011