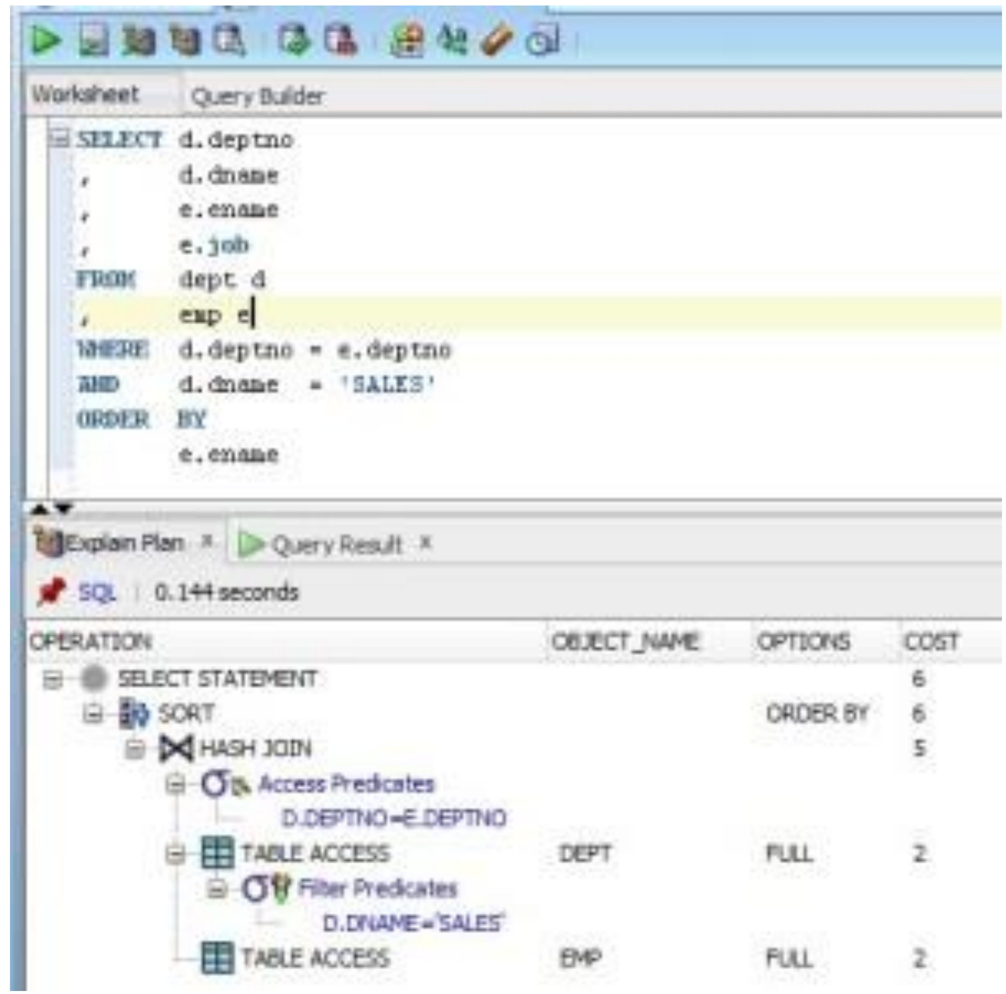
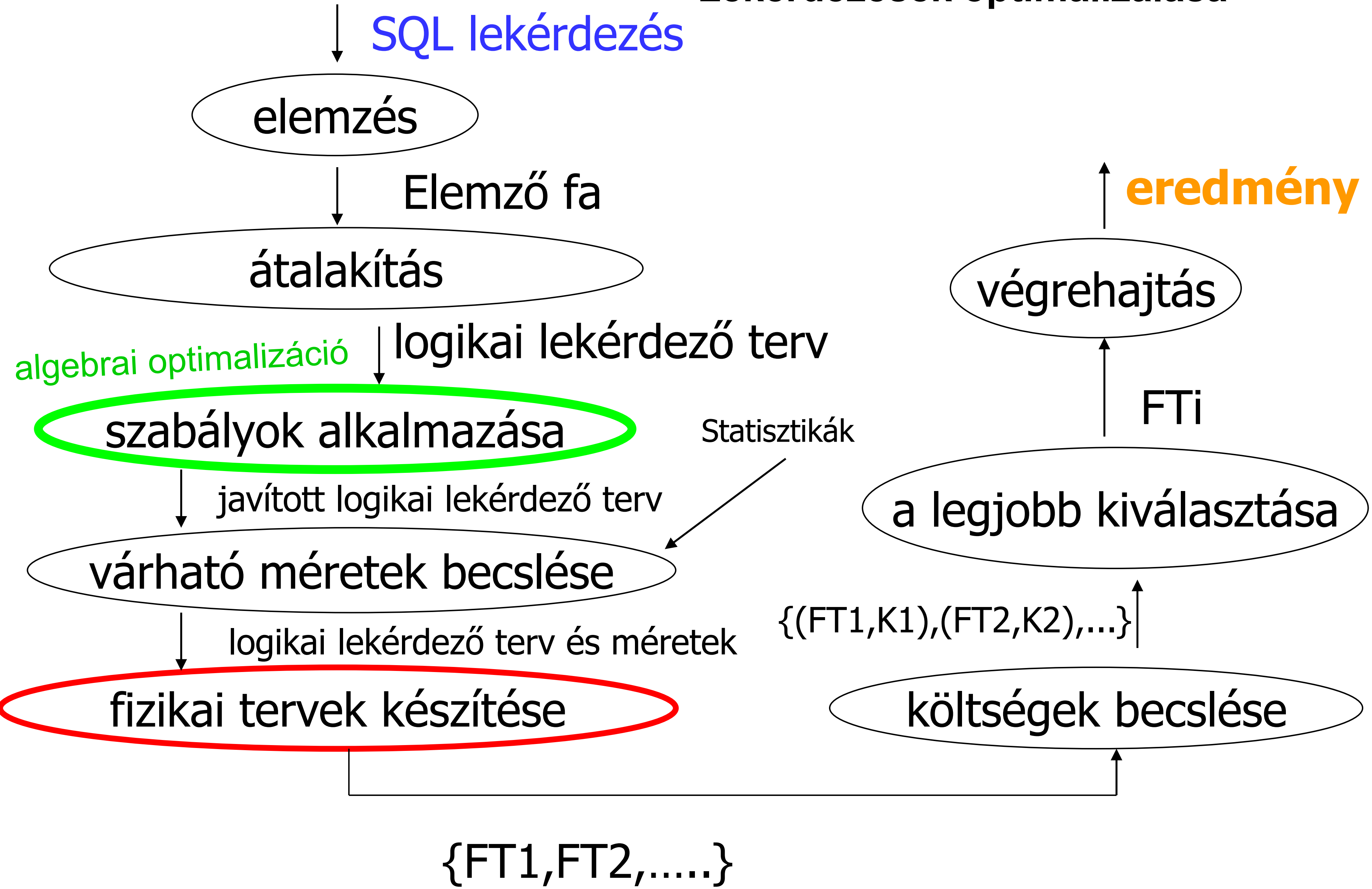
Fizikai tervek

- Paraméterek, költségek
- Fizikai fájlszervezés, indexek
- Műveletek megvalósítása, kiszámítási költség, outputméret
- Optimális fizikai terv meghatározása



Lekérdezések optimalizálása



Indexelés

- **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).
- Az adatbázis-alkalmazások alapján az adatbázis lehet:
  - **statikus (**ritkán módosul, a lekérdezések gyorsasága a fontosabb),
  - **dinamikus** (gyakran módosul, ritkán végzünk lekérdezést).
- 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.**

Indexelés

- Célszerű a fájlokat blokkokba szervezni.
- 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.
- 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**.

Indexelés

- A költségek méréséhez paramétereket vezetünk be:
- **l** - (length) **rekordméret** (bájtokban)
- **b** - **blokkméret** (bájtokban)
- **T** - (tuple) **rekordok száma**
- **B** - a **fájl mérete blokkokban**
- **bf** – **blokkolási faktor** (mennyi rekord fér el egy blokkban: $bf = \lfloor b/l \rfloor$ - alsó egészrész)
- $\mathbf{B = \lceil T/bf \rceil}$
- **M** – **memória mérete blokkokban**
- Például **R×S mérete** mekkora:
  - **l(R×S) = l(R)+l(S)**
  - **T(R×S) = T(R)*T(S)**
  - **bf(R×S) = b / (l(R)+l(S))**
  - **B(R×S) = (T(R)*T(S)) * (l(R)+l(S)) / b**
    = (T(S)*T(R)*l(R)/b) + (T(R)*T(S)*l(S)/b)=
    =T(S)*B(R) + T(R)*B(S)

Indexelés

- **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)
- Kétféle **bonyolultság**ot szokás vizsgálni:
  - **átlagos**,
  - **legrosszabb** eset.
- 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.**)

Indexelés

- Az A oszlopban szereplő különböző értékek számát **képméret**nek hívjuk és **I(A)**-val jelöljük.
- $\mathbf{I(A)=|\Pi_A(R)|}$
- Egyenletességi feltétel esetén:
  - $\mathbf{T(\sigma_{A=a}(R)) = T(R) / I(A)}$
  - $\mathbf{B(\sigma_{A=a}(R)) = B(R) / I(A)}$
- A következő fájlszervezési módszereket fogjuk megvizsgálni:
  - **kupac (heap)**
  - **hasító index (hash)**
  - **rendezett állomány**
  - **elsődleges index (ritka index)**
  - **másodlagos index (sűrű index)**
  - **többszintű index**
  - **B^+-fa, B^*-fa**
- Azt az esetet vizsgáljuk, mikor az A=a feltételű rekordok közül elég az elsőt megkeresni.
- **Módosítási műveletek:**
  - **beszúrás (insert)**
  - **frissítés (update)**
  - **törlés (delete)**
- Az egyszerűsített esetben nem foglalkozunk azzal, hogy a beolvasott rekordokat bent lehet tartani a memóriában, későbbi keresések céljára.

Indexelés

- **Kupac szervezés:**
  - a rekordokat a blokk első üres helyre tesszük a beérkezés sorrendjében.
- **Tárméret**: **B**
- A=a **keresési idő**:
  - B (a legrosszabb esetben),
  - B/2 (átlagos esetben egyenletességi feltétel esetén).
- **Beszúrás**:
  - utolsó blokkba tesszük a rekordot, 1 olvasás + 1 írás
  - módosítás: 1 keresés + 1 írás
  - törlés: 1 keresés + 1 írás (üres hely marad, vagy a törlési bitet állítják át)

Indexelés

- **Hasítóindex-szervezés** (Hashelés):
  - a rekordokat **blokklánc**okba (**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.
- Egy **$h(x)\in\{1,\dots,K\}$** **hasító függvény** értéke mondja meg, hogy melyik kosárba tartozik a rekord, ha x volt az indexmező értéke a rekordban.
- A hasító függvény általában maradékos osztáson alapul. Például **mod(K).**
- Akkor **jó egy hasító függvény**, ha nagyjából egyforma hosszú blokkláncok keletkeznek, azaz egyenletesen sorolja be a rekordokat.
- Jó hasító függvény esetén **a blokklánc B/K blokkból áll**.

Indexelés

- **Keresés** (**A=a**)
  - – ha az indexmező és keresési mező eltér, akkor kupac szervezést jelent,
  - – ha az indexmező és keresési mező megegyezik, akkor csak elég a **h(a)** sorszámú kosarat végignézni, amely **B/K blokkból** álló kupacnak felel meg, azaz **B/K** legrosszabb esetben. **A keresés K-szorosára gyorsul.**
- Miért nem érdemes nagyon nagy K-t választani?
- **Tárméret**: **B**, ha minden blokk nagyjából tele.
- Nagy K esetén sok olyan blokklánc lehet, amely egy blokkból fog állni, és a blokkban is csak 1 rekord lesz. Ekkor a keresési idő: 1 blokkbeolvasás, de B helyett T számú blokkban tároljuk az adatokat.
- **Módosítás**: B/K blokkból álló kupac szervezésű kosarat kell módosítani.
- **Intervallumos (a < A < b) típusú keresésre nem jó.**

Indexelés

Tegyük fel, hogy 1 blokkba 2 rekord fér el.

Szúrjuk be a következő hasító értékkel rendelkező rekordokat!

INSERT:

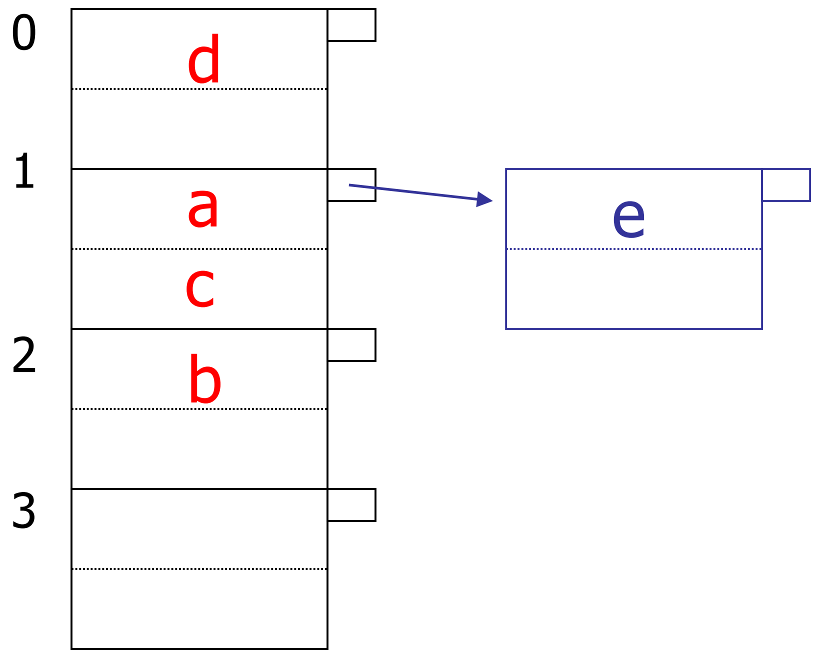
h(a) = 1

h(b) = 2

h(c) = 1

h(d) = 0

h(e) = 1



Indexelés

Töröljük a következő hasító értékkel rendelkező rekordokat!

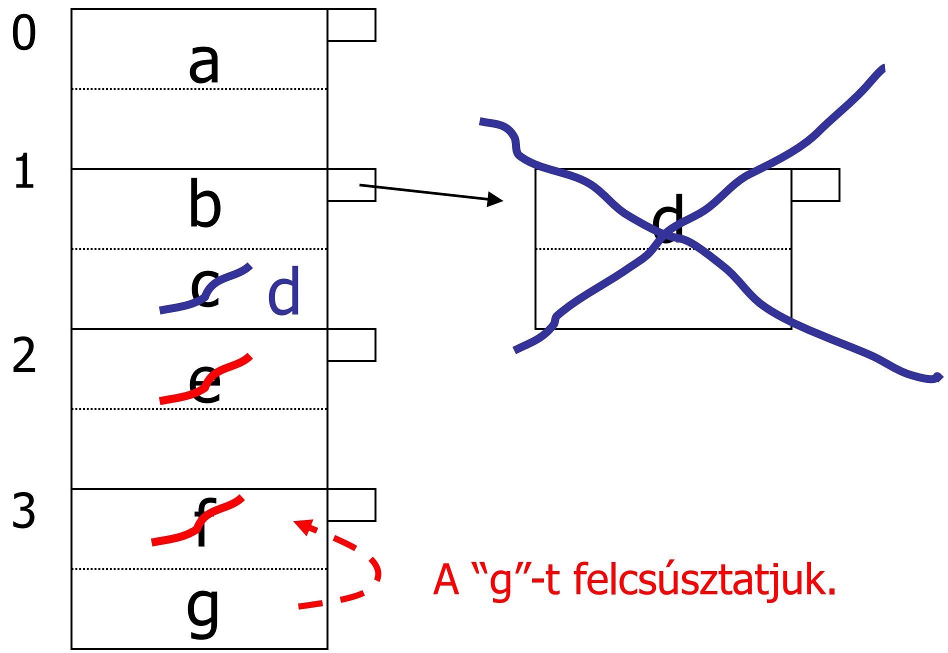
(A megüresedett túlcsordulási blokkokat megszüntetjük.)

Delete:

e

f

c



Indexelés

- **Dinamikus hasító indexek:**
  - **kiterjeszthető (expandable)**
  - **lineáris**
- Előre nem rögzítjük a kosarak számát, a kosarak száma beszúráskor, törléskor változhat.
- **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.**
- Minden kosárhoz tartozik egy legfeljebb k hosszú bináris sorozat (kódszó).
- A kosarakhoz rendelt kód **prefix kód**. A maximális kód hossza legyen i.
- A h(x) k hosszú kódnak vegyük az **i hosszú elejét**, és **azt kosarat, amelynek kódja a h(x) kezdő szelete**. Ha van hely a kosárban, tegyük bele a rekordot, ha nincs, akkor nyissunk egy új kosarat, és a következő bit alapján osszuk ketté a telített kosár rekordjait. Ha ez a bit mindegyikre megegyezik, akkor a következő bitet vesszük a szétosztáshoz, és így tovább.

Indexelés

0 | 0010

1 | 1010, 1011

1101 beszúrása

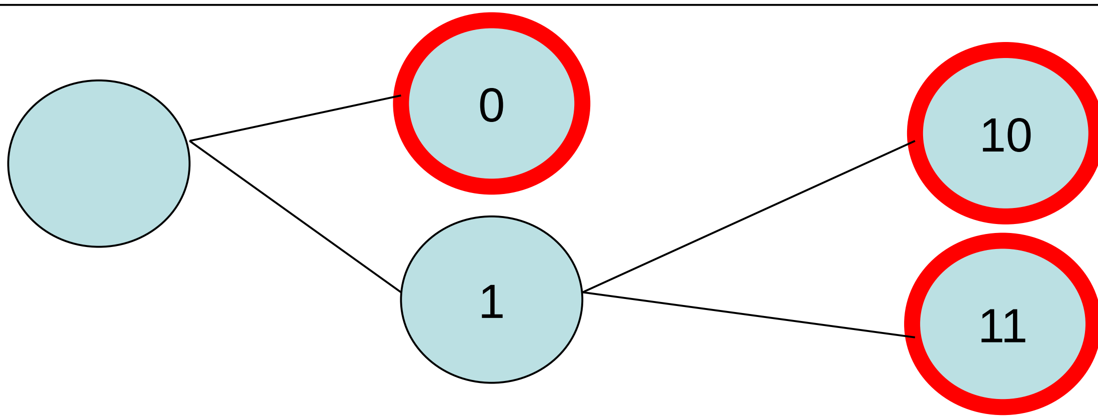
2. bit alapján

---

0 | 0010

1 | 0 | 1010, 1011

1 | 1101



Indexelés

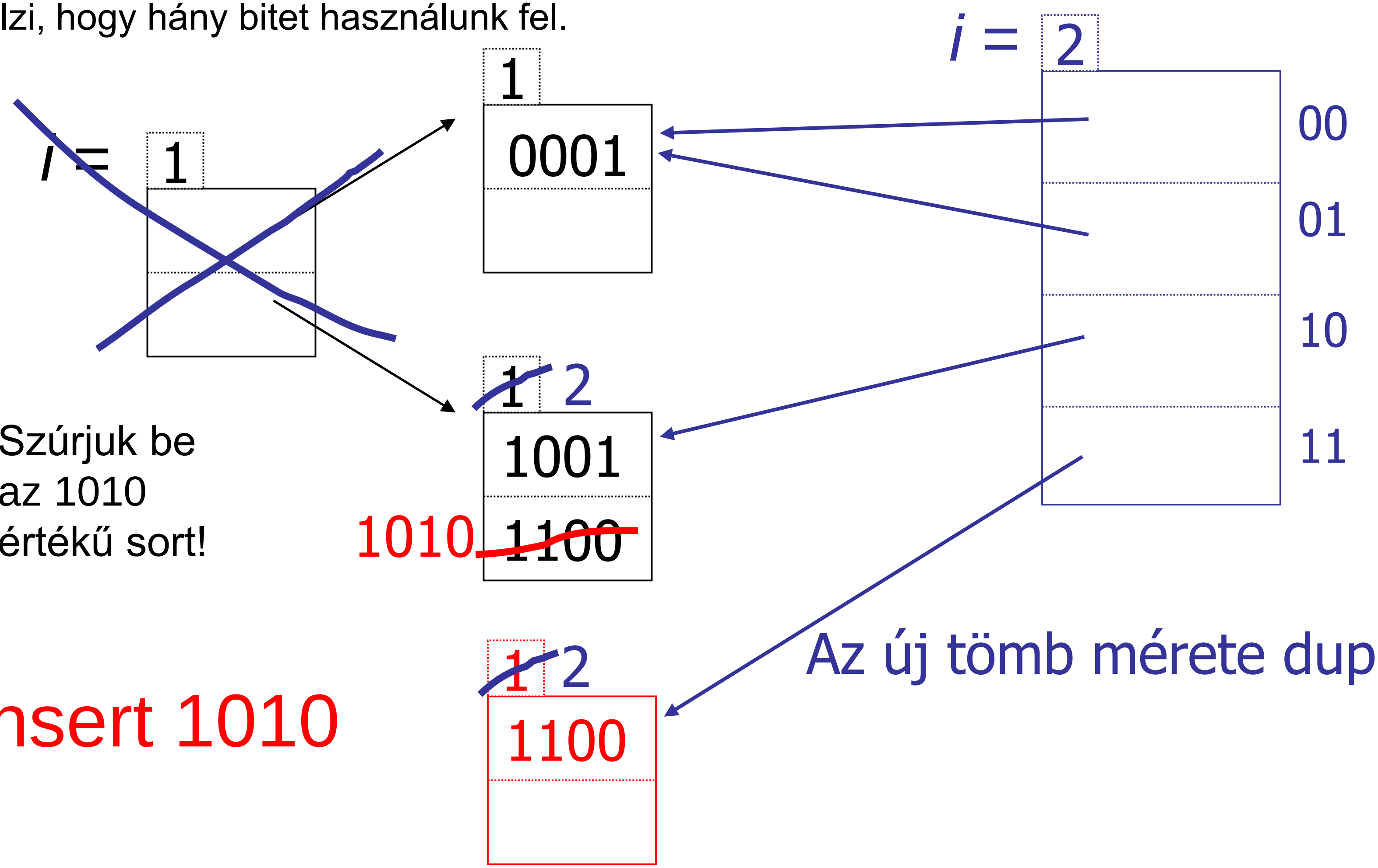
- A bináris fa levelei a kosárblokkok kódszavai. A hasító függvény értékéből annyi bitet használunk, ahányadik szinten szerepel a levél.
- A gráfot a memóriában tárolhatjuk.

Probléma: Ha az új sorok hasító értékének eleje sok bitben megegyezik, akkor **hosszú ágak** keletkezhetnek.

(Nincs kiegyensúlyozva a fa.)

Indexelés

A bináris gráfot teljessé is tehetjük. A gráfot egy tömbbel ábrázolhatjuk. Ekkor minden kosár azonos szinten lesz, de közös blokkjai is lehetnek a kosaraknak. Túlcsordulás esetén **a kosarak száma duplázódik.** Legyen például h(x) 4 bites és 2 rekord férjen egy blokkba. Az i jelzi, hogy hány bitet használunk fel.



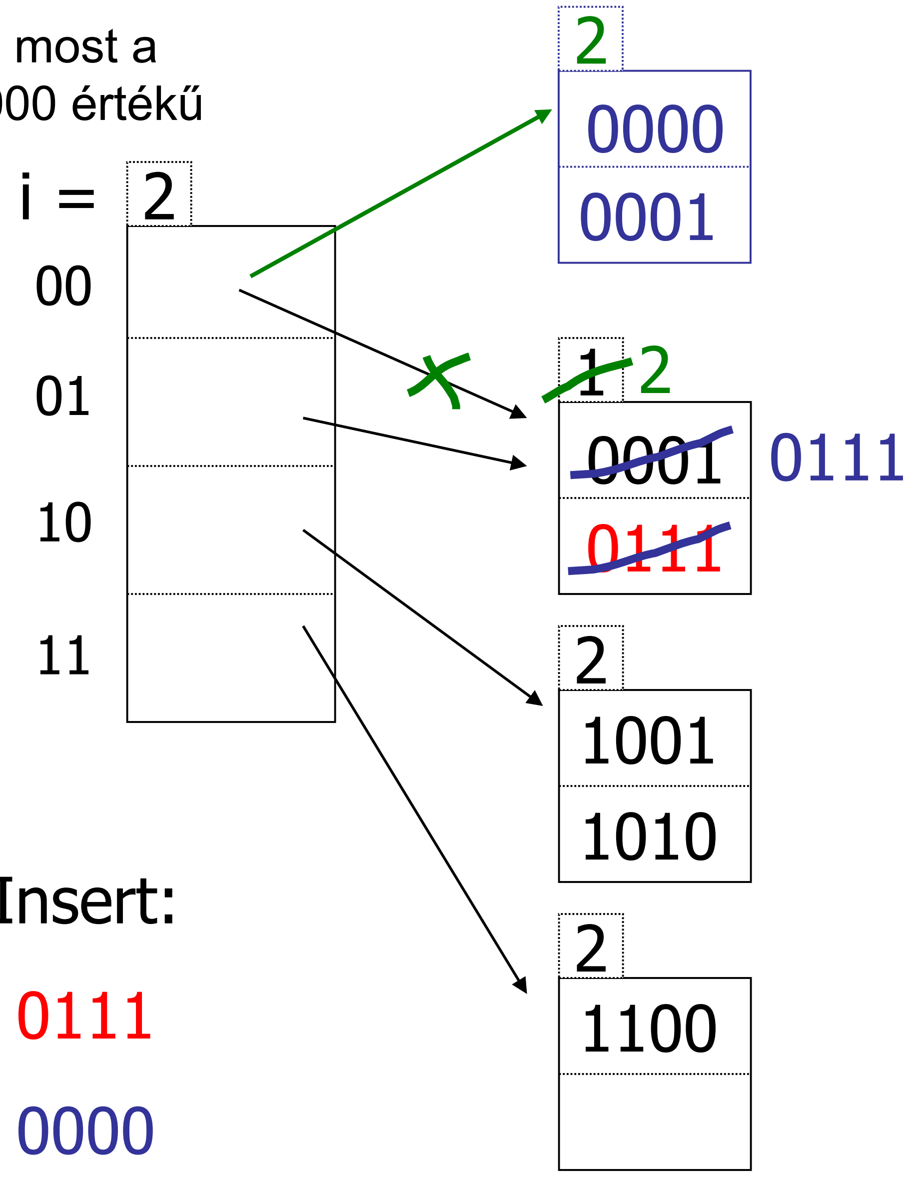
Szúrjuk be az 1010 értékű sort!

Insert 1010

Az új tömb mérete duplázódik.

Indexelés

Szúrjuk be most a 0111 és 0000 értékű sorokat!



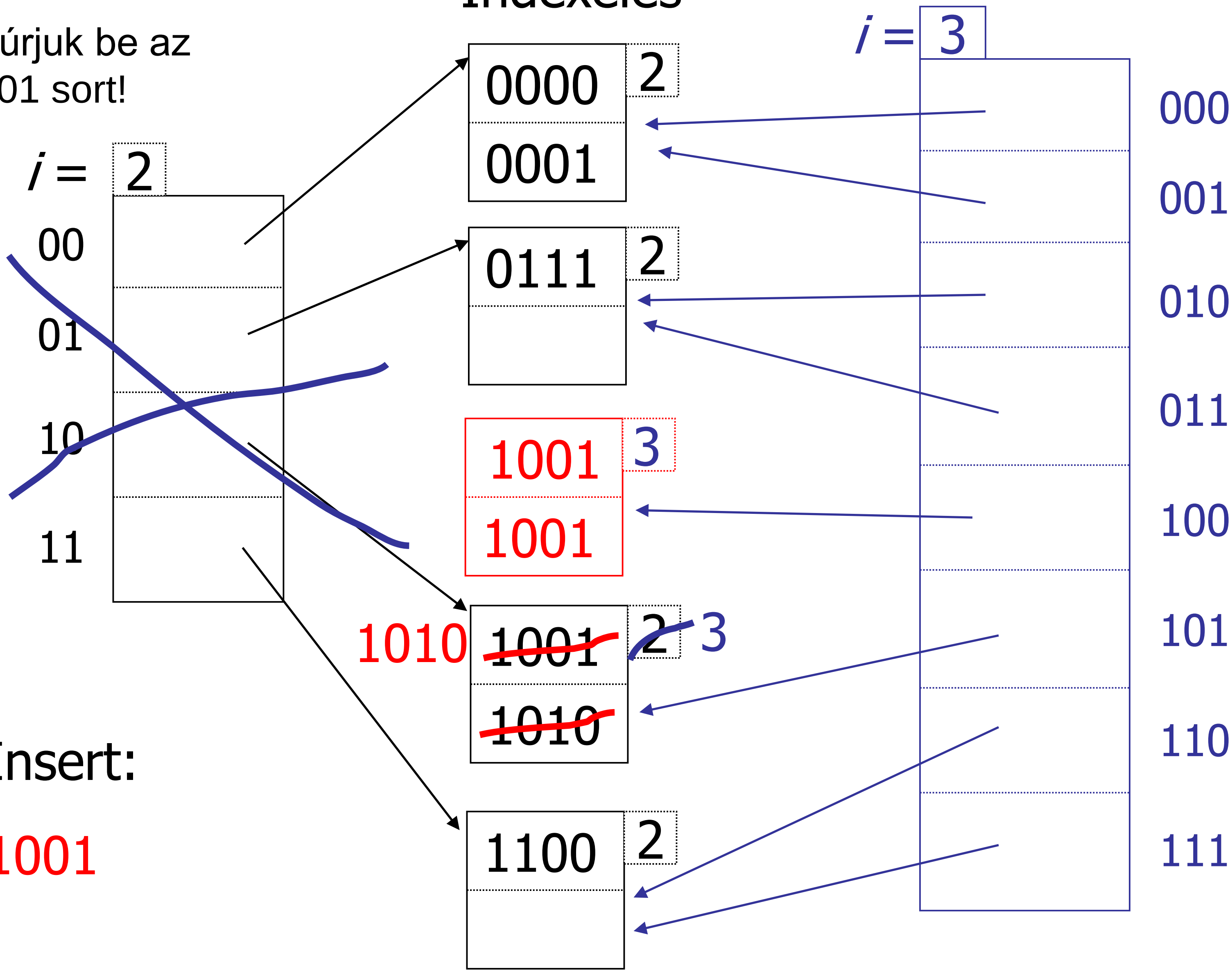
Insert:

0111

0000

Indexelés

Szúrjuk be az
1001 sort!



Insert:

1001

Indexelés

- **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.

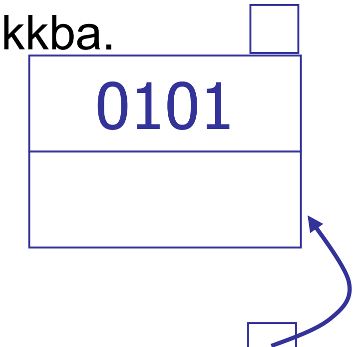
(rekordok száma/kosarak száma > küszöb)

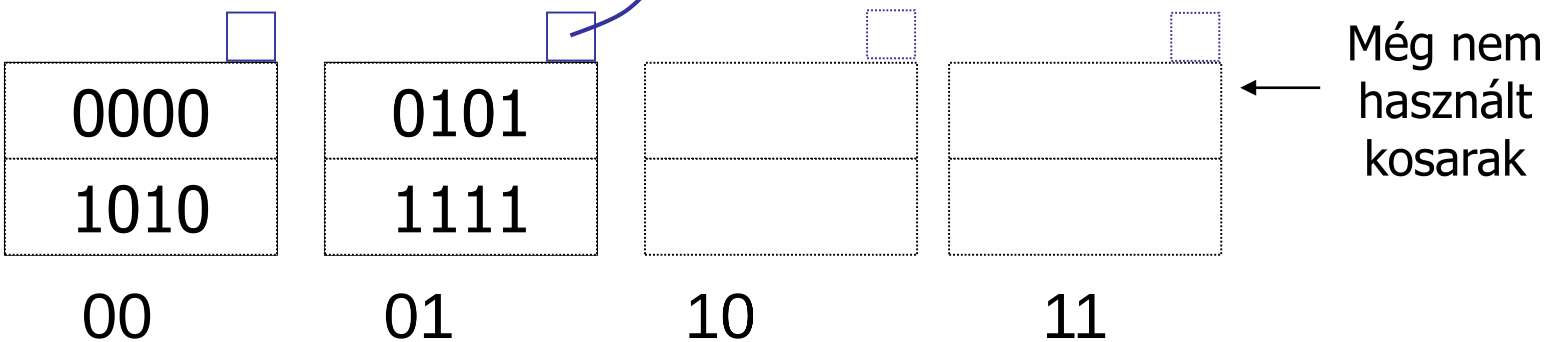
- **A kosarakat 0-tól kezdve sorszámozzuk, és a sorszámot binárisan ábrázoljuk.**
- Ha n kosarunk van, akkor a hasító függvény értékének **utolsó log(n) bit**jével megegyező sorszámú kosárba tesszük, ha van benne hely. Ha nincs, akkor hozzáláncolunk egy új blokkot és abba tesszük.
- Ha nincs megfelelő sorszámú kosár, akkor abba a sorszámú kosárba tesszük, amely csak az első bitjében különbözik a keresett sorszámtól.

Indexelés

Legyen a hasító érték 4 bites, *i* =2, és 2 rekord férjen egy blokkba.

- Szúrjuk be a 0101 értékű rekordot!
- Túlcsordulási blokkot nyitunk.





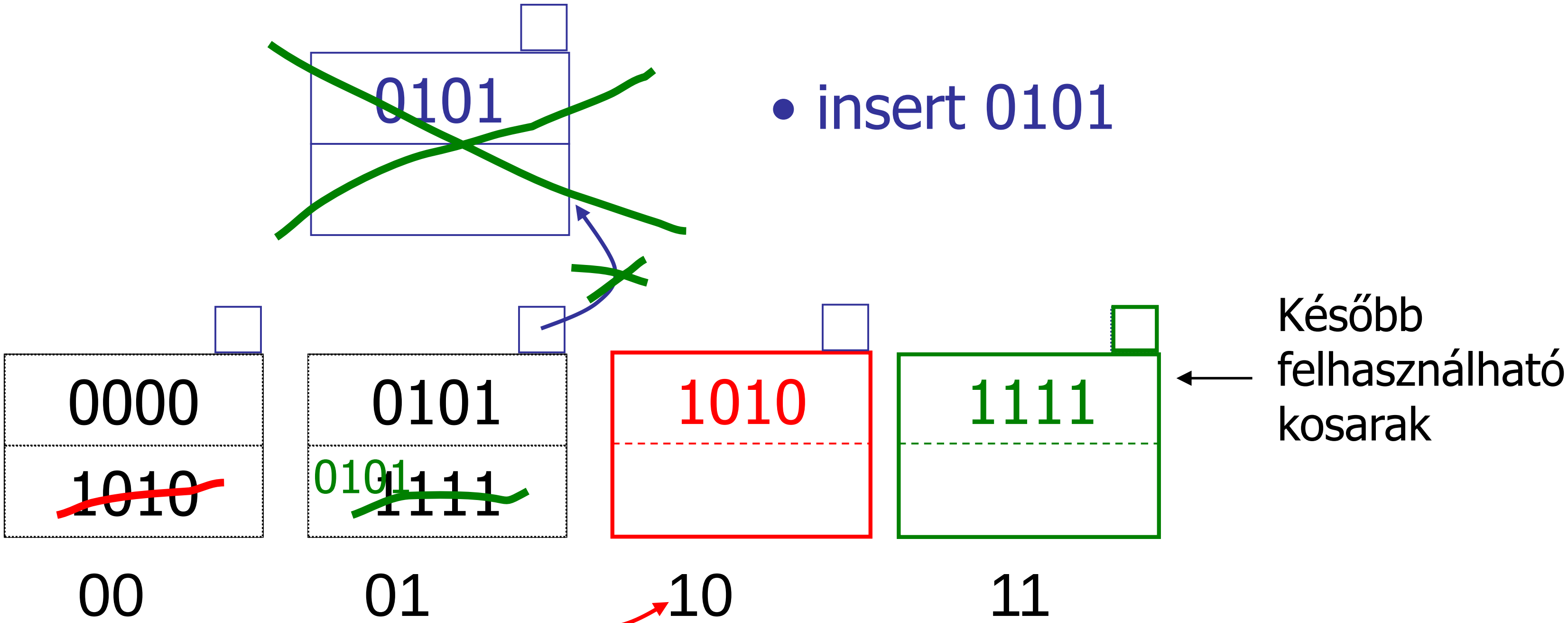
m = 01 (a legnagyobb sorszámú) blokk sorszáma

Szabály: Ha **h(x)[*i*] ≤ *m***, akkor a rekordot tegyük a **h(x)[*i*]** kosárba, különben pedig a **h(x)[*i*] - 2^{i-1}** kosárba!

Megjegyzés: **h(x)[i]** és **h(x)[i] - 2^{i-1}** csak az első bitben különbözik!

Indexelés

- insert 0101



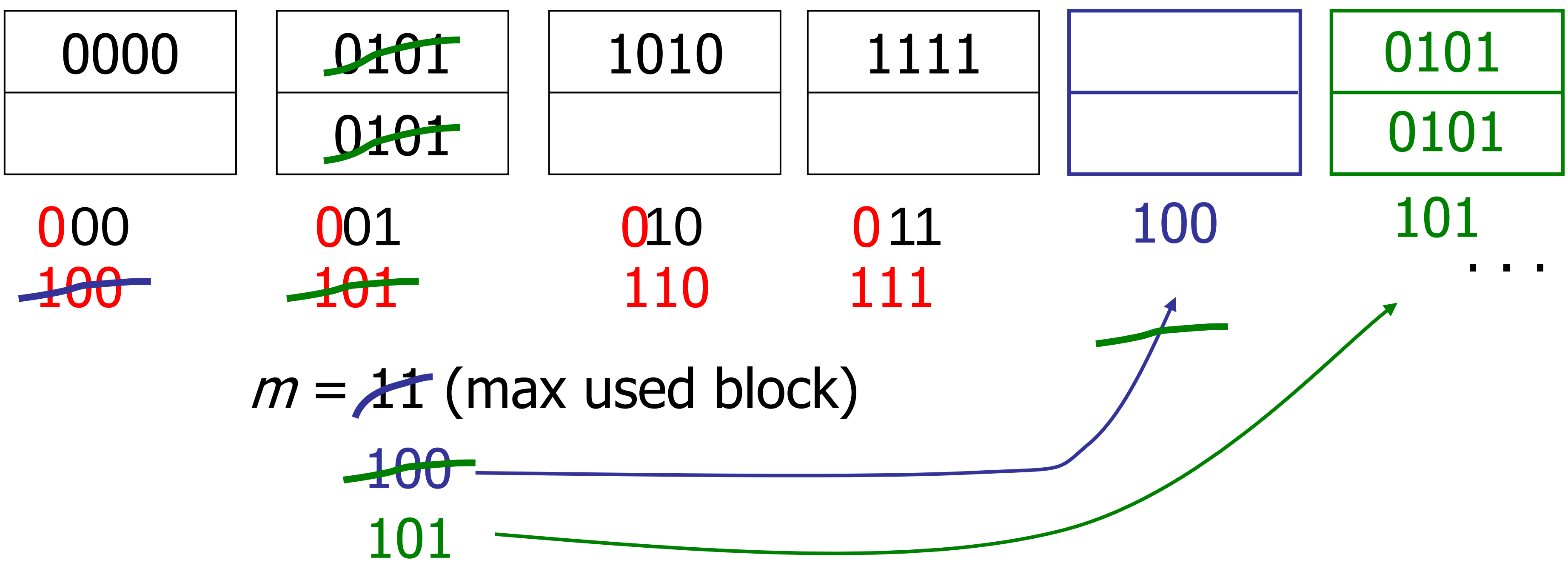
m = ~~01~~ (legnagyobb használt sorszám)
~~10~~
11

Tegyük fel, hogy átléptük a küszöbszámot, és ezért új kosarat kell nyitni, majd az első bitben különböző sorszámú kosárból át kell tenni ebbe az egyező végződésű rekordokat.

Indexelés

Ha i bitet használunk és 2^i kosarunk van, akkor a következő kosárnyitás előtt i-t megnöveljük 1-gyel, és az első bitben különböző sorszámú kosárból áttöltjük a szükséges rekordokat és így tovább.

$i = \cancel{2}\ 3$



<u>Összefoglalás:</u>
Fizikai tervezés, alapfogalmak, jelölések, költségek, kupac (szekvenciális) és hasításon alapuló szervezés

Köszönöm a figyelmet!