

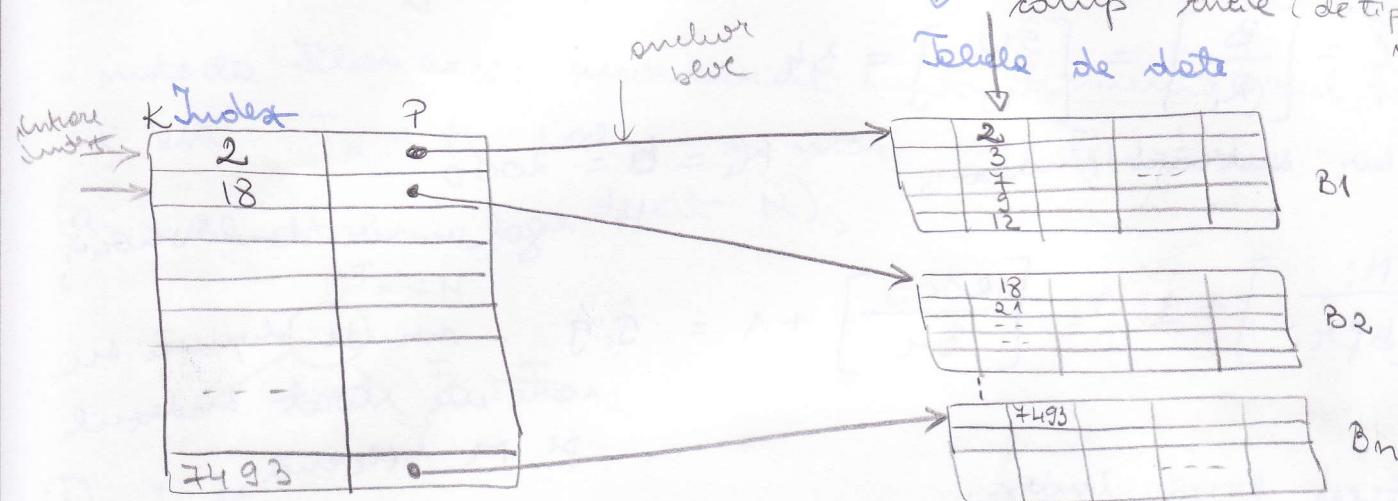
Index

- modalitate de acces la date după cheie și criterii

- ① Index primar: se construiește după câmpul cheie al tabelui de date, iar tabelul de date este ordonat după câmpul cheie.
 Indexul primar este fără sup. la construcția indexelor multimedii, d.p.d. că accesului la date sunt indexuri care crează o eficiență bună fără de secundă fără index.

stocată în blocurile de extensie
câmp cheie (de tip numeric natural)

Tabel de date



SO = DOS / WINDOWS

B = 512 octeti

- pp că dim. unei su registrări este de 100 octeti $\Rightarrow R = 100$ octeti;
 considerăm su registrare de dim. fixă;

bif $R =$ bloc factor $R =$ nr. de su registrări pe bloc

$$bif R = \left[\frac{B}{R} \right] = \left[\frac{512}{100} \right] = 5 \Rightarrow$$

dintre-un bloc pot fi stocate 5 recorduri;

OBS. tabelul este sortat fizic după câmpul cheie, valoarea 1 nu există.

K = cheie de indexare

P = pointer

- linile din Index s.u. intrările index;

- în Index se generează o intrare pt. fiecare bloc cu pt. frecare su registrare dintr-un bloc;

- metoda care slocoi pt. frecare bloc și intrare din index și anume bloc (indexul legătura, nu recorduri)

- pp că totale, are $N = w^4$ su registrări

$$bif = \left[\frac{N}{bif R} \right] + 1$$

(OBS: d.c. N este multiplu de $bif R$ nu mai este număr de $+1$);

$$b = \left[\frac{N}{bfri} \right] + 1 = \left[\frac{10^4}{5} \right] = 2000 \text{ blocuri (unirea se face tot la fel.)}$$

- pp că valoarea lui P e stocată pe 6 octeți, iar K e stocat pe 9 octeți; $\Rightarrow \langle P \rangle = 6$; $\langle K \rangle = 9$

$$R_i = \langle K \rangle + \langle P \rangle = 15 \text{ octeți}$$

\hookrightarrow spațialul necesar pt a stoca o intrare din index;

- uniform, din index se găsește tot pe M extenuă stocată;

$bfri =$ bloc factor index index

$$bfri = \left[\frac{B}{R_i} \right] = \left[\frac{512}{15} \right] = 34$$

$$N_i = \text{nr. intrări index}; \quad N_i = b = 2000$$

\hookrightarrow egal cu nr. de blocuri;

$$b_i = \left[\frac{N_i}{bfri} \right] + 1 = \left[\frac{2000}{34} \right] + 1 = 59 - \text{nr de blocuri de care este stocat indexul pe } M \text{ extenuă},$$

② Acces la index:

- pp că indexul nu este mult și doarces acces direct la tabelă

- 2 posibilități pt. acces:

1) acces secvențial: se aduce din M bloc cu bloc și se caută sau se dorește acces direct la tabelă

\hookrightarrow nr. de blocuri necesare să fie transferate

2) prin metode de hibriditate - căutare binară (bazată pe proprietăți de ordonare a tabeliei)

$$mb = \lceil \log_2 b \rceil + 1 =$$

$$= \lceil \log_2 2000 \rceil + 1 = 11 \text{ blocuri}$$

③ Acces cu index:

1) căutare binară $\rightarrow mb = \lceil \log_2 b_i \rceil + 1 + 1 =$

$\underbrace{\qquad}_{\text{uniform, din bloc de}} \uparrow \text{index} \text{ de date}$

$$= \lceil \log_2 59 \rceil + 1 + 1 = 7 \text{ blocuri}$$

OBS: 7 blocuri compozitul u 11 blocuri de la secundul index

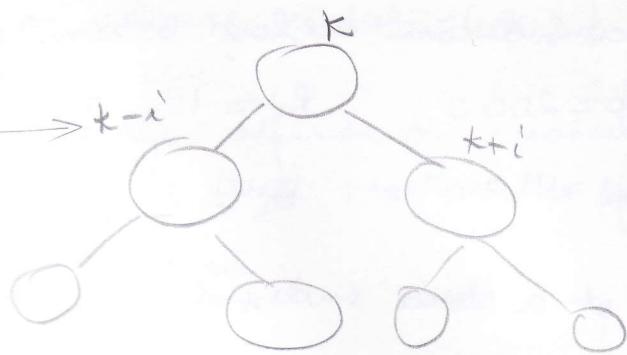
code de căutare binomială:

- căutare binară obisnuită

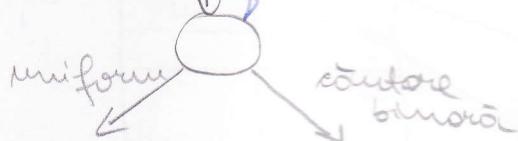
- căutare binară uniformă

- căutare Shor:

OBS: se desvăluă în timpul căutării un orice binar.

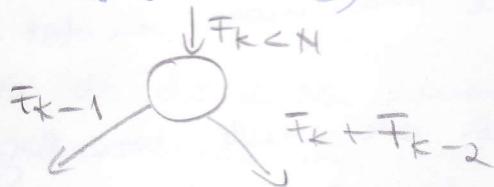


STAR: $k = \lceil \frac{\log n}{2} \rceil$; $k = \text{rădăcina sechecelui}$



OBS: ordinea pt STAR nu are sens; dimensiune

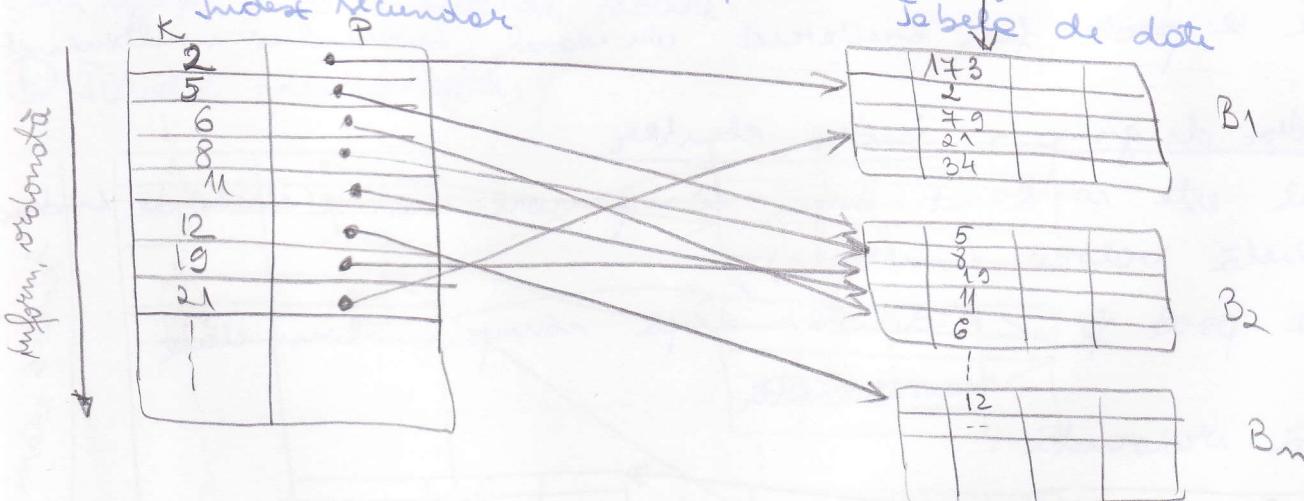
- metode Fibonacci: recurențele sunt termenii și-așezării Fibonacci. k este $F_k < n$ (căcă mai mare nr. Fibonacci mai mare decât n);



② Index secundar:

- se aplică tabelelor neveredante după compuile cheie, iar indexul K conținește după valoarea compului cheie;

Index secundar



OBS: construisește indexul tot secundar;

K = cheie secundară

- def. fără de indexul primar: în rezultat sunt, pt fiecare sugețitor J o altăre, deci unul sauva o singură informație pt un bloc;

- considerăm următorul volum de date și anivelul):

$$b = 200 \text{ } \Omega \quad R_i = 15 \Rightarrow b + R_i = 34$$

⇒ pt a stocă în desul sună mările de 25 g de blenuri;

② Trees farō under

- canticile secentistice - unele metode foarte bune sunt ca;

$$Nb \approx \frac{b}{2} = 1000 = \underline{\underline{10^3}}$$

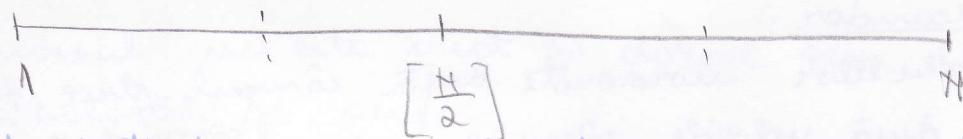
Sur de "blanc" véhicule.

⑥ Access via nodes:

$$mb = \left\lceil \log_2 b_i \right\rceil + l + 1 = \left\lceil \log_2 259 \right\rceil + l + 1 = 8 + 2 = 10$$

OBS: ca tot urmă de sursele în tabelă e mai mare, ceea ce rezultă că
aducerea subiectului este mai bună.

DBS: Mă în rezul ⑤ se scadă deci se crește baza logaritmului \Rightarrow se folosește metoda de rădare binară.



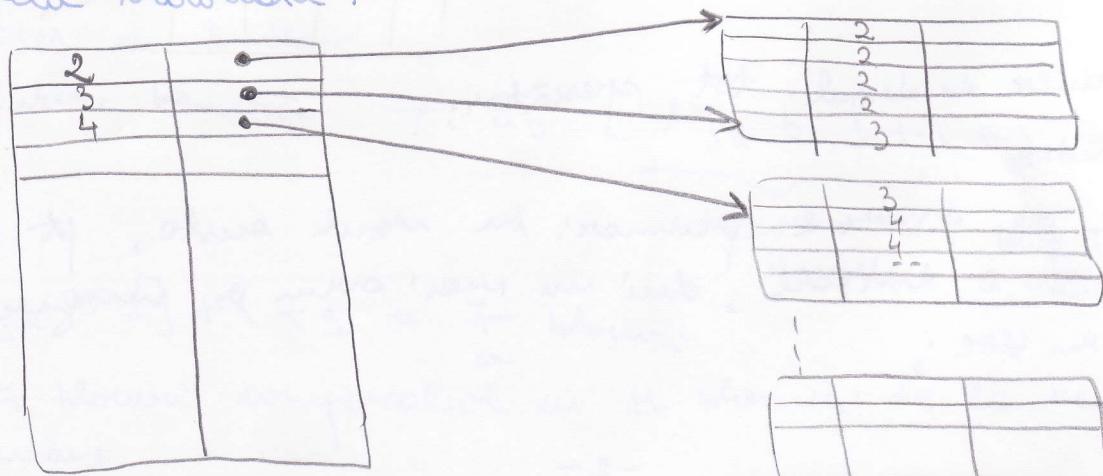
- deoarece importanță din 3 \Rightarrow să se pună să gresesc, dacă este metodă
prin care să poată face creșterea unui rezervor hidroenergetic multitudine).

③ Indeks de groep - indeks cluster

- indexul este constituit după comp. non-diag. (clase de similaritate sunt în tablă velor multiple);

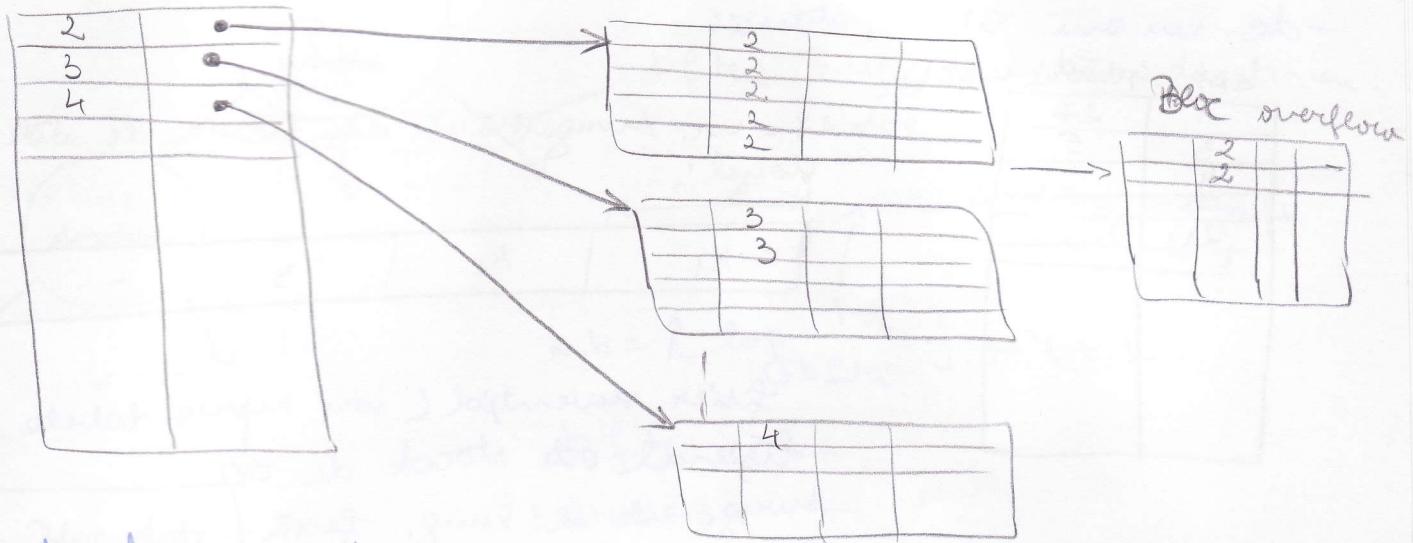
- tabella poche f.
 \ edonata - super comples non-simile
 \ neodonata

a) Jelvē ūdrotnē.



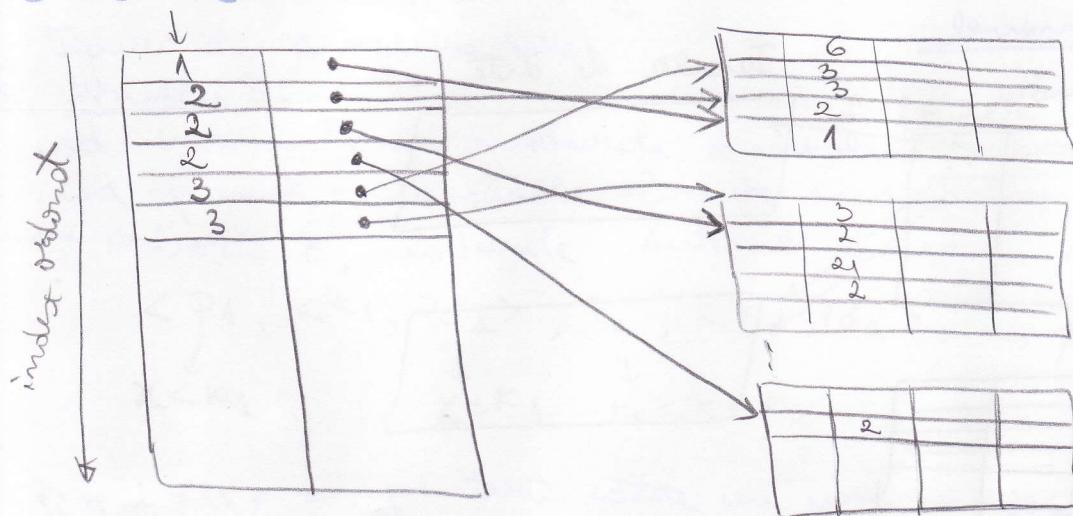
- de un bloc se termină cu o valoare căutată (ex. 3) sunt obligat să caute și în următorul bloc
- modul de suplementare și tabela depende de modulul de căutare; suplementarea poate fi: densă (forță rezervor de spațiu) sau non-densă;

blocul non-dens:



- OBS: de dorești să sturezi un 2 de astăzi: îl pot stura din primul bloc; de unde să mai sturezi un record \Rightarrow sporește un bloc overflow unde sturezi 2;
- le stergere - nu se stergă separat din valoarea inserată; când se redchiară din bloc overflow se scrie în blocul initial, se adaugă altă și se sterge blocul overflow;
- acest metoda e fără eficiență datorită întreținerii, ea are nevoie de un spațiu mai mare de stocare;

⑥ Tabelă neveredonată:



- compul după care s-a făcut adăugarea nu mai este comp cheie, deoarece avem valori multiple.

Caracteristica index:

- a) indexul este ordonat
- b) cu o singură excepție (casul 3.b), în index nu există cheie (nu se repetă valoare).

DBS:

1) -toti indexii se referă la DBMSC (SDL = Storage Definition Language);

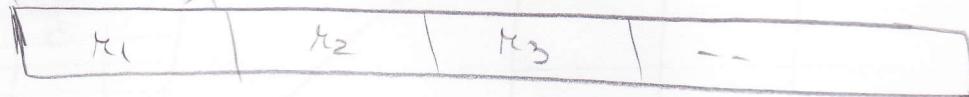
-de la un SDL, stau:

CK> record number

1	27
3	5
4	
7	
21	

27 = nr. suriectiv din tabel de date, nu blocul;

$\leftarrow B$



fisier sequential (nu repres. tabele de date)

- fisierul este stocat de SO

- lung. este de lung. fixă

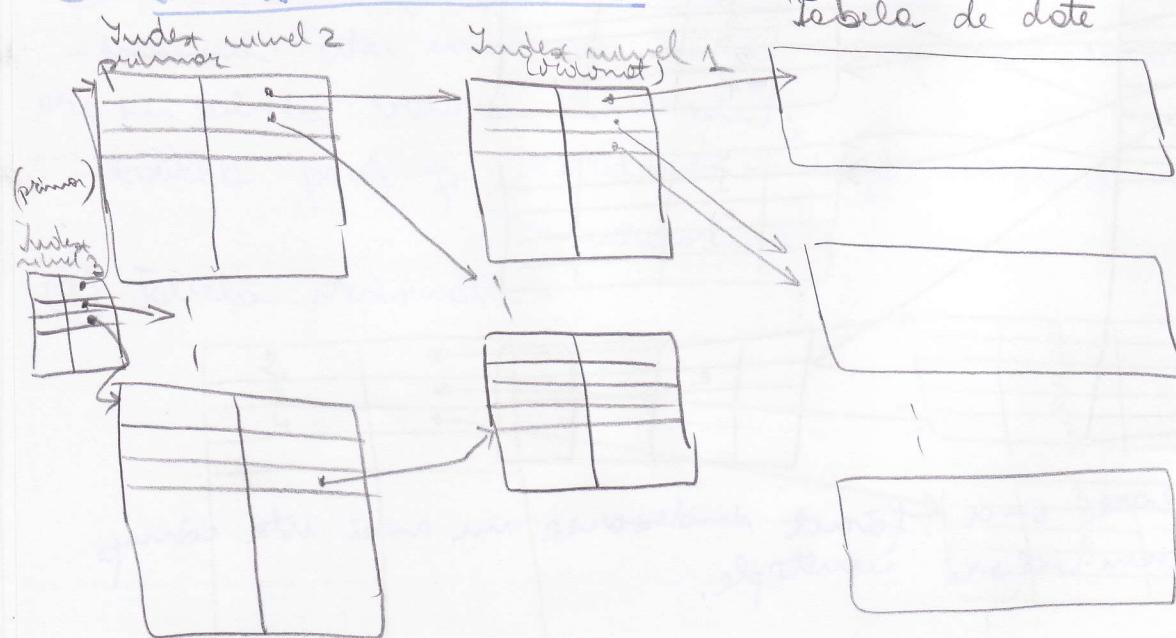
2) suriectivile de lungime variabilă:

VARCHAR (n) → tip de date; spațiul este compus din un număr n de caractere, unde n poate să depășească.

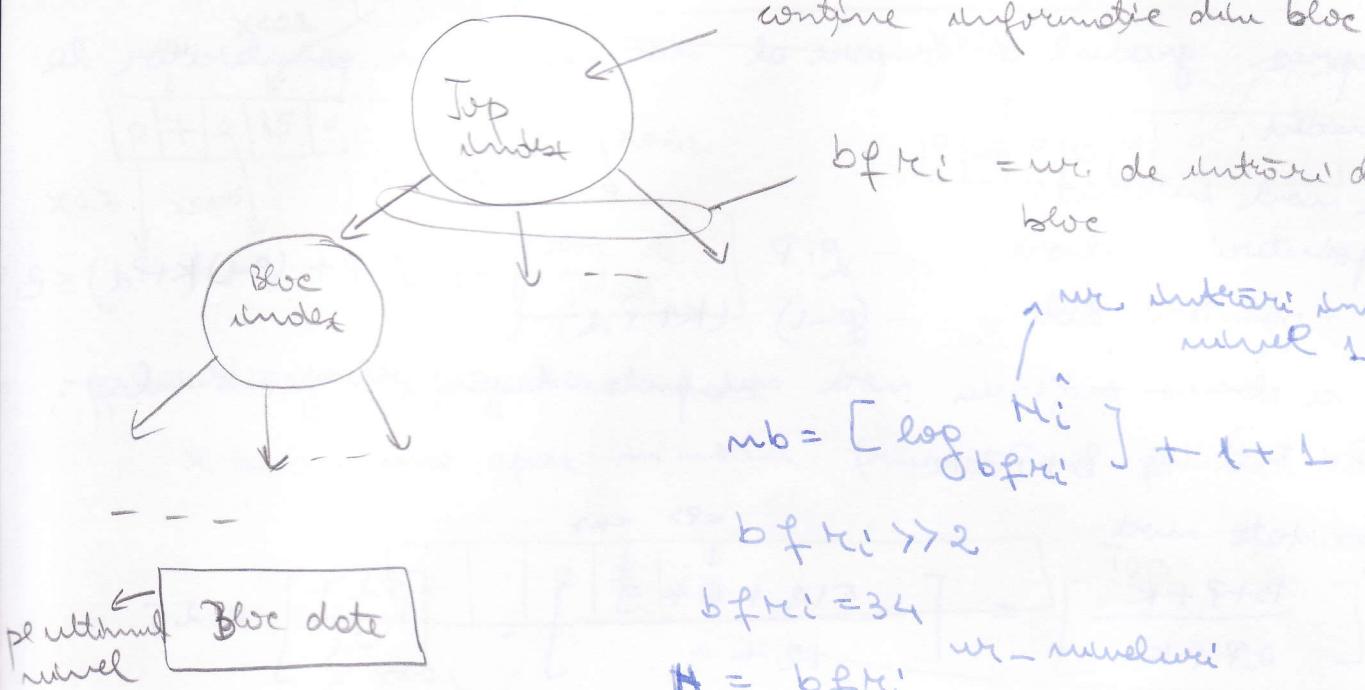
- de se face index pe VARCHAR, nu se face comparație a sețim din numărul de VARCHAR-ului, și a avea tot indexul de dim. fixă;

- un număr de lung. fixă și a face ușor manipularea.

④ Index multirevel



indexul de nivel 1 are o structură care depinde de date;
 indexul de nivel 1 este ordonat și este vizibil tot în tabel
 și stocat pe în exterior;
 - indexarea se face prin poziție indexare după numărul bloc;



contine informație dintr-un bloc

bfr_1 = nr. de interne dintr-un bloc

nr. interne index la nivel 1)

$$mb = \lceil \log_{bfr_1}^{N_1} \rceil + 1 + L$$

$$bfr_1 \gg 2$$

$$bfr_1 = 34$$

$$H = bfr_1$$

nr. de interne

reunite în un nod multimediu

- generarea indexului multimediu și generare de un obiect multimediu în care fiecare nod reprez. inform. reprezentată într-un bloc.
 OBS: - de cele mai multe ori sunt modificate în tabel. De aceea reflect modificările în index \Rightarrow reconstruire index (prin urmări proceduri de salvare/stocare într-un bloc).

Tipuri de suplementare:

② Arborei B:

- nod intern = toate nodurile indexului de nivel > 1
- nod frumos = blocurile diferențe indexului de nivel 1
- în arborei B, nodurile interne au num. structură:

$$\langle P_1, \langle k_1, P_{d1} \rangle, P_2, \langle k_2, P_{d2} \rangle, \dots, P_i, \langle k_i, P_{di} \rangle, \dots, P_g \rangle$$

$$x < k_1 \quad x = k_1 \quad k_1 < x < k_2 \quad x > k_{g-1}$$

$P_i, i=1..g =$ pointerii către un nod index (de nivel inferior sau niv. actual); și u. pointerii privind căre indexă în slt bloc index

$P_{di}, i=1..g-1 =$ pointerii către blocuri de date; și u. pointerii datei $x =$ valoarea compusă după care se face scasun;

- P_1 aduce un bloc index de, $x < k_1$

- P_{d+1} aduce bloc de date de, $x = k_1$

- P_2 aduce bloc index de $k_1 < x \leq k_2$

- P_L aduce un nod index de $x > k_{L-1}$;

OBS: L reprez gradul de varpere al modului = nr. nodurilor le bleuri interne.

• Intr-un nod intern:

- L - pozitie arbore

- L · P

$$L \cdot P + (L-1)(k+P_d) \leq B$$

- L-1 - pozitie date

- (L-1) (k+P_d)

OBS: cae ce stocă intr-un nod nu poate depășii valoarea blocului;

$$L(2P+k) \leq B+P+k$$

d = capacitate nod

$\leftarrow P \quad \leftarrow k$

$$d = \left[\frac{B+P+k}{2P+k} \right] = \left[\frac{512+8+9}{12+9} \right] = \left[\frac{527}{21} \right] = 25$$

\Rightarrow 25 de pozitii pot fi menținute stocate într-un nod;

L = gradul de varpere al unui nod este cel puțin 2 la top index (nodul de cel mai mare nivel) și cel puțin $\frac{d}{2}$ la celelalte noduri interne;

• Modul frunză are urmări structură

$(R_{d1}, k_1, P_{d2}, k_2, \dots, R_{di}, k_i, P_{di+1}, \dots, P_{dL})$

- din modul frunză \nexists pozitii arbore, pt că sunt în pozitii frunză și văd numai bleuri de date;

$$x < k_1 \quad k_1 \leq x < k_2$$

$$x \geq k_{L-1}$$

- L primează și $L-1$ din;

$$L \cdot P + (L-1)k \leq B \Rightarrow d = \left[\frac{B+k}{P+k} \right] = \left[\frac{512+9}{9+6} \right] = \left[\frac{521}{15} \right] = 34$$

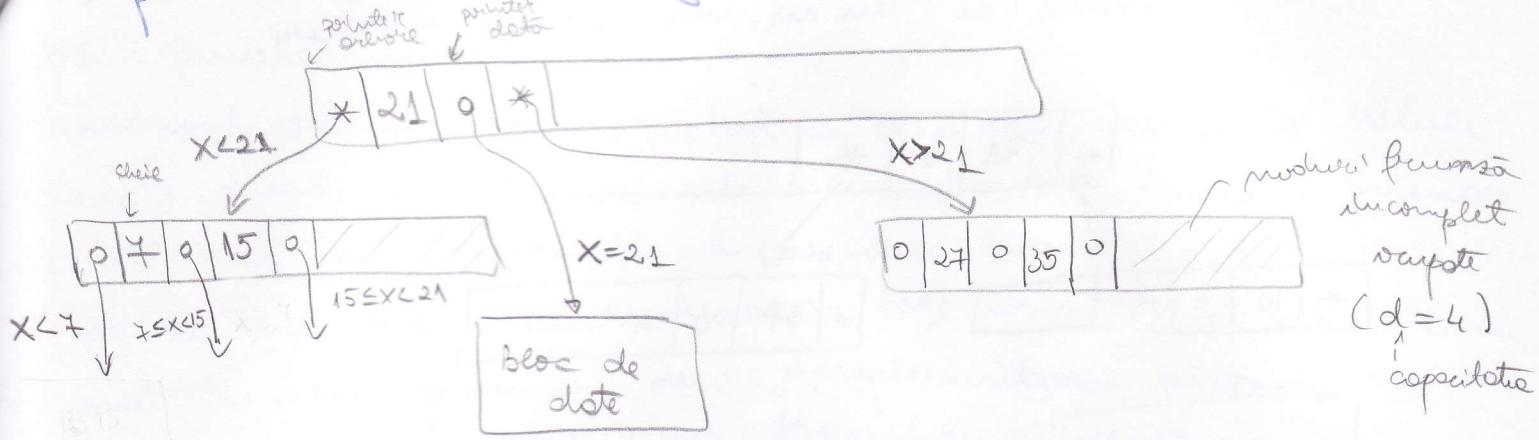
capacitate modului frunză

OBS: modul frunză are capacitate mai mare decât modurile interne;

L este cel puțin $\frac{d}{2}$; L = grad de varpere.

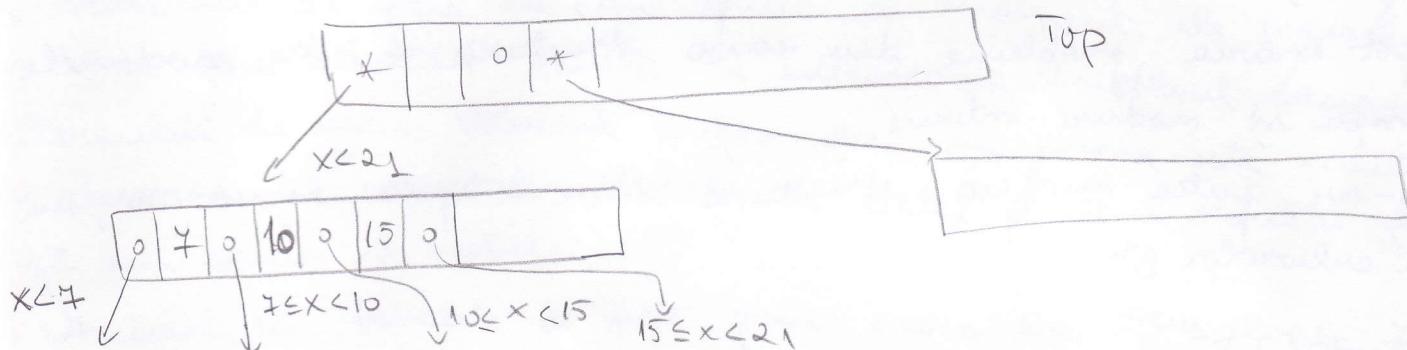
Să se sătuaște în care subintervale să fie format dintr-un singur interval de informație. Începe.

Computerul sănătățe le potrivire / stergere date din tabele;



- pp că adaug o nouă înregistrare - Add

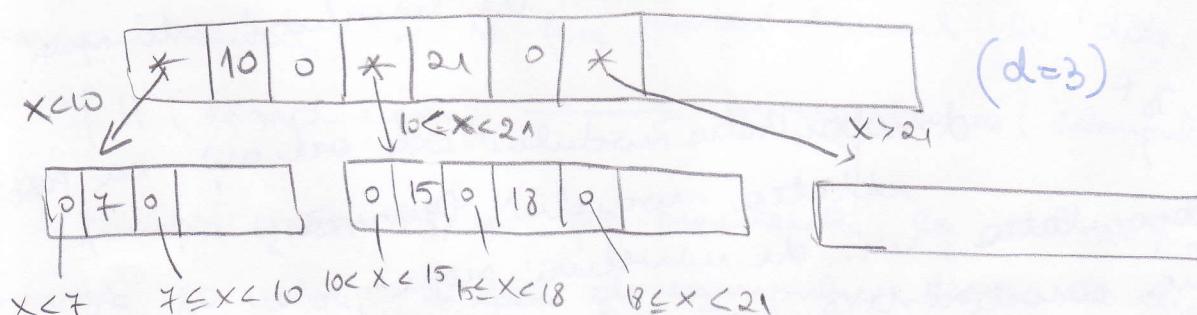
$x = 10$; mai apoi în modul frunză un poziție date;



- vreau să adaug o nouă înregistrare \rightarrow Add 18 dar d este deși 4 (în modul frunză complet ocupat);

$7, 10, 15, 8$ \rightarrow Separare mod în 2 module

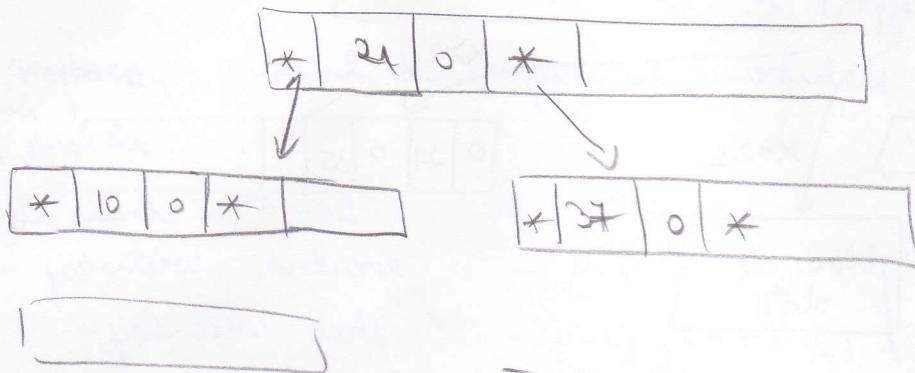
- se consideră noulul sprijinitor (se consideră pe 10); 10 se ve de la nivelul superior (nuță în modul de deasupra);



- pp că suntem transfer de la un modul de sus, dată nu mai sun loc;

10, 21, 37

↳ tragește un număr superior și se generează un
ordine cu un nr. mai mare de număruri.



- la stergerea de surgență: de. este loc, se aduce, după
stergere, nu x din numărul superior;

- pot apărea probleme din cauza structurii \Rightarrow bătăi sau lucru
făcându-se moduri interne.

- S-ar putea să fie structura acela și tipuri de moduri, în
rezultatul cărora B⁺.

(b) Schimbul B⁺:

→ structura modului intern se va modifica:

$$\langle P_1, k_1, P_2, k_2, \dots, P_l, k_l, P_{l+1}, \dots, P_r \rangle \\ x \in k_1 \quad \begin{matrix} \downarrow \\ k_1 \leq k_2 \end{matrix} \quad \downarrow \quad x \in k_{l+1}$$

OBS: schimbul B ar putea fi mai eficient decât B⁺ dacă sănă
dintre peste un pointer date, dar nr. de număruri este mai
mare.

- pt schimbul B⁺: d-coordonata modului este același cu coor-

- de cruce c-oordonată modului furniză,

- B⁺ are puține caracteristici că este uniform + acces pp.

permutarea tuturor numărurilor.

$$d = \left[\frac{B+k}{P+k} \right]$$

- cea mai eficientă metodă de sușare este sușarea multîmuivel;

- sușul se autotimporează f. ex. (echivalând cu reinșearea);

OBS. finale Indes:

- 1) Indesul primar, prin îoptul să se aplică numai la tebeli ordonate după comp. care este nevoie. Iar, sușul primar este utilizat la construcția sușului multîmuivel.
- 2) de la nivelul 2 de sușare, toti sușii în orbezi și cu B^+ sunt sușuri primare, deci suplimentarea metodei de sușare este bine. Rezultatul \Rightarrow transformă structura de către în orbezi multi-roi.
- 3) Aceasi tabelă de date are asociată mai multă sușuri, funcție de cantitatea de secu. la un anumit moment. De fiecare dată când se face secu la tabelă se utilizează sușul corespunzător cantității de secu. Volumul informației pe suș este mult mai mare decât suprafața din tebel de date (pt că sușul se face după un comp. al tebelui).
- 4) Iatăcum se tabelile de date pe suș sunt logice. Fiecare suș reprezintă o metodă de secu logică. Iatăcum se și tabelă de date fără sușul D.N. secu fizic.

Tehnica HASH

- este o tehnică de secu direct; oferă cea mai rapidă metodă de secu dor cătreul de secu nu poate fi modificat.

- cheia de secu este secu și + moment;

- funcție HASH - faptul că secuul să se utilizeze în comp. HASH - compul după care se face un alt secu la date.

f_H (comp. hash) \longrightarrow surghetare (compul de surghetare)

OBS:

- funcție HASH trebuie să fie rezistentă la atacuri: de exemplu funcție de voluri ~~de~~ ale cămpului hash să nu văd secu și surghetare.

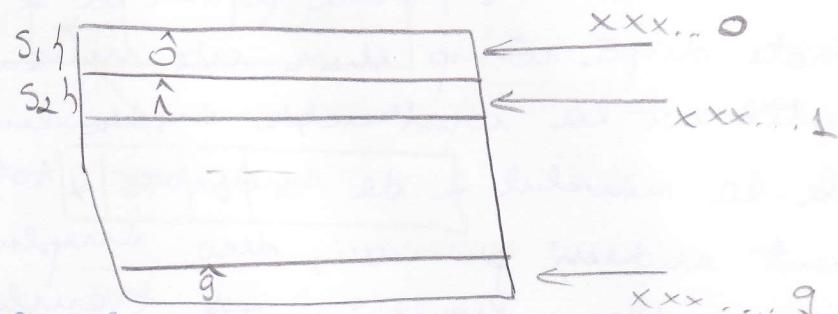
Exemplu:

adres - hash \rightarrow mărcitor

Z_{10} = adresa de rețea / modulu 10

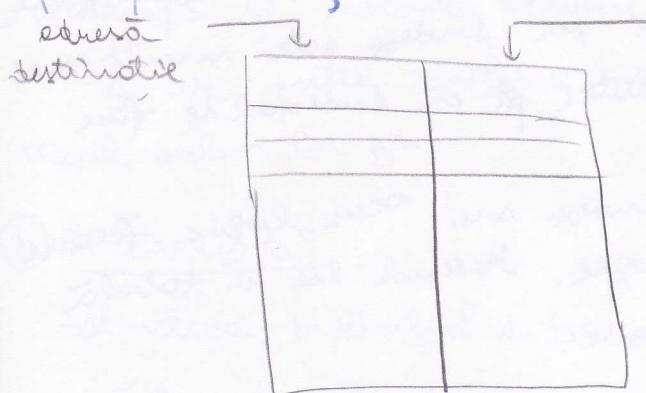
$f_H(Z_{10}) \rightarrow 0, 1, 2, \dots, 9$

- se definesc 10 spații:



- pentru funcția HASH, trebuie să fie segmentată în 10 grupuri; fiecare segmentare să reprezinte fișoarea corespunzătoare unei din "grupuri"; suntem definit o ordine fizică pt. aceste "grupuri"; ordinea este dată deoarece de funcția HASH.

OBS: ca funcție HASH să nu fie stocată, nu ecranizată funcție fizică interioară; este utilizată la routeruri sau rețele.



- înălțarea în next-hop se face cu funcție HASH în funcție de adresa destinație
- pt. det. next-hop se face către tot cu funcție HASH.

În funcție de locul unde se aplică HASH, și importă în 2 cazuri:

a) HASH intern: aplicat la memorarea / cunoașterea datelor în mem. internă (casul router)

b) HASH extern: utilizat la stocare / cercul la date pe mem. externă

- HASH-ul este f. mult utilizat în criptografie.