Dynamické hešovanie

Obmedzenie doteraz známeho (statického) hešovania:

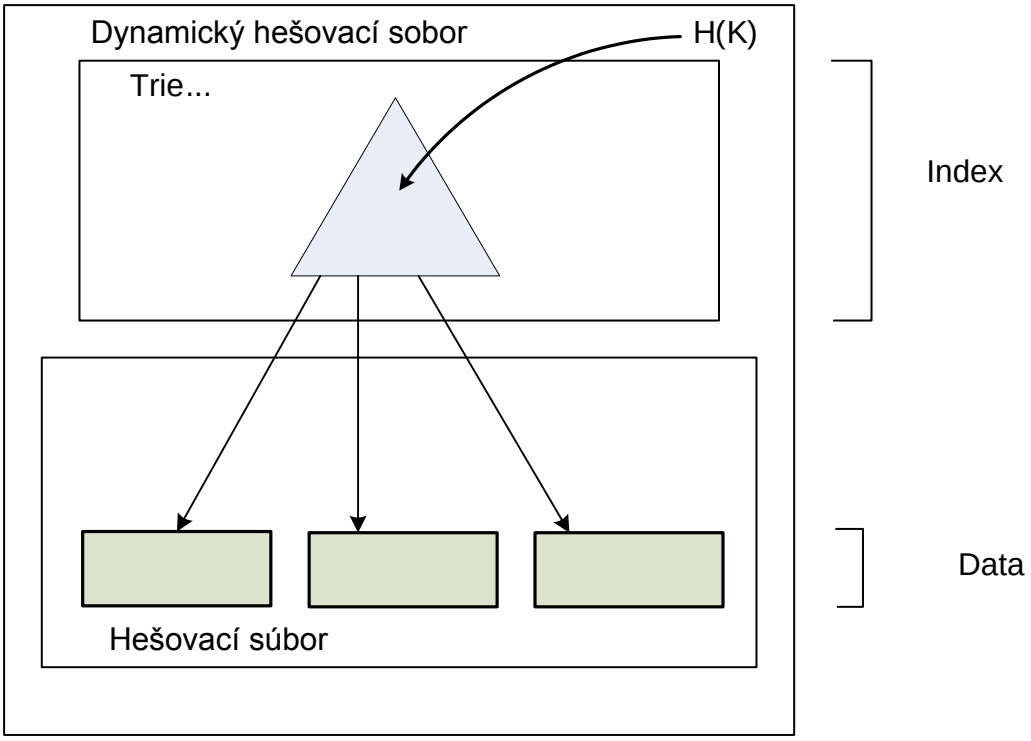
- **Vyhovuje iba pre súbory pevnej (vopred známej) veľkosti, lebo s touto veľkosťou je spätá hešovacia funkcia. Pri dynamickom raste efektívnosť degraduje, rekonštrukcia je ťažká až často aj nemožná.**
- **Pevná veľkosť súboru bez ohľadu na reálne vložené dáta.**

Myšlienka dynamického hešovania:

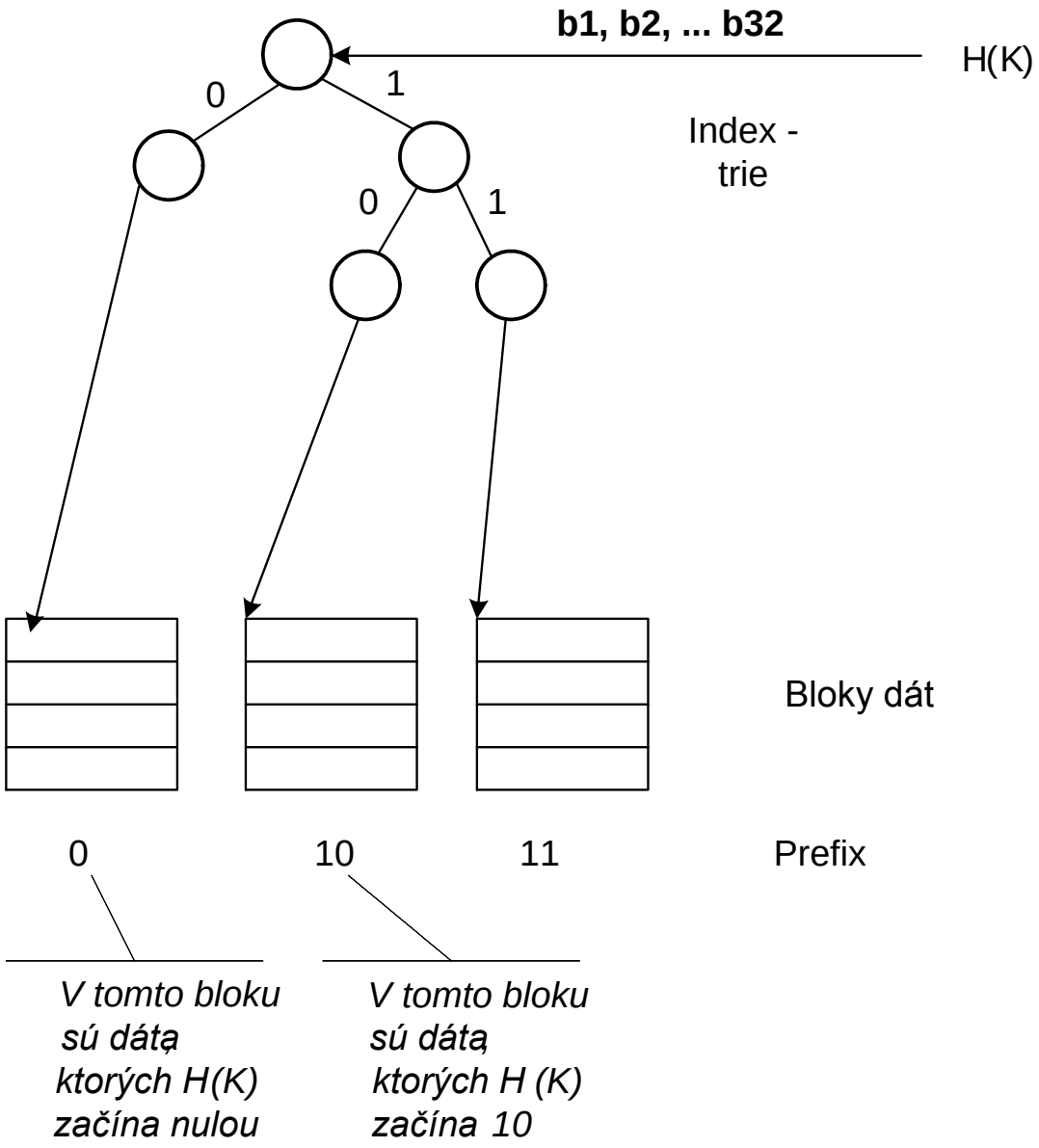
- pri raste a zmenšovaní súboru sa uloženie dát automaticky reorganizuje – veľkosť súboru sa mení podľa skutočne vložených dát
- zachováva okamžitý prístup k dátam v súbore (rovnako ako statické hešovanie)
- využitie iba istého počtu (postupne rastie) bitov z výsledku hešovacej funkcie
- použitie adresára blokov, ktorý sa nachádza v operačnej pamäti

<u>Organizácia</u>:

Kombinácia klasického hešovacieho súboru a indexu (v tvare binárneho znakového stromu - trie). Preto niekedy názov: indexovaný hešovací súbor. Index sa nachádza v operačnej pamäti, dáta na disku.
Tiež sa považuje za alternatívu B^+ stromu.
Princíp fungovania je zhodný s rozšíriteľným hešovaním, index je tu ale tvorený stromovou štruktúrou.
Na rozdiel od rozšíriteľného hešovacieho súboru index rastie plynule, nie skokom.
Voľné bloky uprostred súboru je nutné využívať prednostne (rovnaký postup ako prázdne bloky v prelňujúcom súbore – pozri príslušný text o statickom hešovacom súbore).

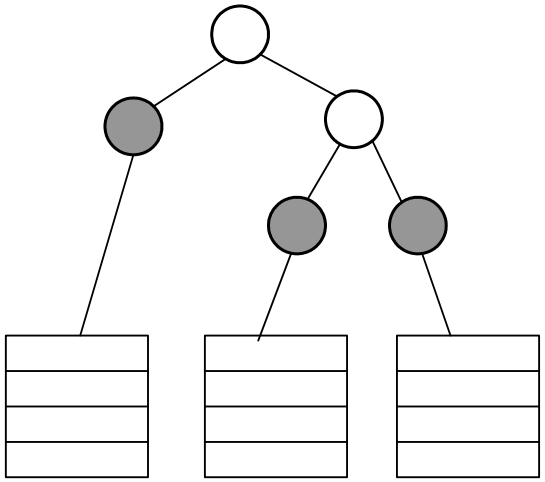


Princíp:
Hešovacia funkcia H(K) mapuje kľúče K do bitových reťazcov dĺžky L (napr. L = 32).
H_d(K) označuje prvých *d* bitov H(K) a nazýva sa prefix H(K) dĺžky *d*.
Pre prístup k adresám blokov slúžia prefixy premenlivej dĺžky.
Pri sprístupňovaní traverzovanie: 0 - doľava, 1 - doprava.



Dĺžka prefixu bloku sa nazýva <u>hĺbka bloku</u> *d* (prvý blok na obr. má hĺbku 1, ostatné dva majú hĺbku 2).
Maximálna hĺbka bloku udáva <u>hĺbku *D* hešovacieho súboru</u> (v príklade: *D* = 2).
Na dáta v bloku s hĺbkou menšou ako *D* odkazuje viac prvkov adresára (napr. na prvý blok na obr.).

Z implementačných dôvodov i z dôvodu zmenšenia pamäťovej náročnosti indexu sa používa nasledujúci variant štruktúry:



Prázdny vrchol ..... interný
Plný vrchol .....externý

Strom trie už nemusí byť vyvážený
Hĺbka rovná vzdialenosti externého vrchola od koreňa

Aby bola expanzia indexu flexibilná a plynulá, je index reprezentovaný explicitne (vo väčšine prípadov sa zmestí do internej pamäti). Obsah interných a externých vrcholov býva rôzny (z dôvodu šetrenia pamäťou):

Interný

Otec	
Ľavý	Pravý

Externý

Otec	
Poč. zázn	Blok (index/adresa)

Poč. zázn: aktuálny počet záznamov v bloku
- pri vyhľadávaní zamedzí prenos prázdneho bloku
- pri vkladaní informuje o možnosti preplnenia

<u>Operácia nájdi záznam s kľúčom K:</u>

1. Vypočítaj I = h_d (K) (prvých D bitov hodnoty hešovacej funkcie),
2. Pomocou adresára (trie) sprístupni blok P[i],
3. V bloku P[i] nájdi záznam s kľúčom K.

<u>Operácia vlož</u>:

1. Aplikuj H(K)…. výsledok bitový reťazec B;
2. Ak koreň interný, traverzuj podľa B k externému vrcholu E;

3. Ak externý vrchol E neukazuje na žiadny blok, alokuj blok, ulož doňho záznam a nastav naňho adresu "Blok"

inak

ak blok s adresou "Blok" nie je plný, vlož záznam,
inak opakuj až do vyriešenia situácie preplnenia:

A. externý vrchol E transformuj na interný a vytvor mu dvoch synov – externé vrcholy; každému z týchto externých vrcholov alokuj blok (zatiaľ iba v operačnej pamäti).

B. presuň záznamy z preplneného bloku a nový záznam do ľavého alebo pravého bloku použijúc ďalší bit bitového reťazca (zväčšenie hĺbky D súboru). Ak do jedného z blokov nepadnú žiadne záznamy, dealokuj ho – jeho dvojča ostáva preplnené (zapíše sa do súboru), situáciu rieš opakovaním od bodu A.

V prípade, že dôjde k využitiu všetkých bitov z výsledku hešovacej funkcie a záznam nebolo možné vložiť dôjde ku kolízii. Na jej riešenie je možné použiť oblasť preplňujúcich blokov, alebo preplňujúci súbor. V prípade, že sa nevyužívajú bity z výsledku hešovacej funkcie (tento výsledok nemá zaručenú unikátnosť), ale z unikátneho kľúča, nedochádza k vzniku kolízií.

<u>Operácia vymaž:</u>
Ak po zrušení zostane v bloku a jeho "susednom" bloku iba toľko záznamov, že sa zmestia do jediného bloku, presunú sa tam a voľný blok sa dealokuje. Keďže každý vrchol pozná svojho otca nie je problém zistiť brata príslušného externého vrcholu a podľa počtu záznamov v príslušnom bloku rozhodnúť o možnosti zlúčenia. Operácia prebieha cyklicky podobne ako vkladanie.

Príklad:

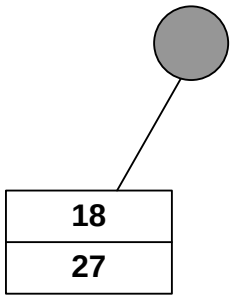
Zapíšme do prázdneho súboru nasledujúce 4 záznamy (kľúče) pričom budeme predpokladať v tabuľke uvedené hodnoty hešovacej funkcie.

K	H(K)
27	0101
18	1110
39	0001
36	0110

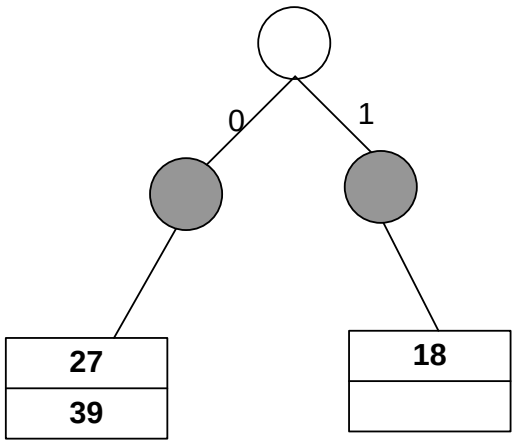
Na začiatku štruktúra pozostáva z koreňa – externého vrchola.

Nech blokovací faktor = 2.

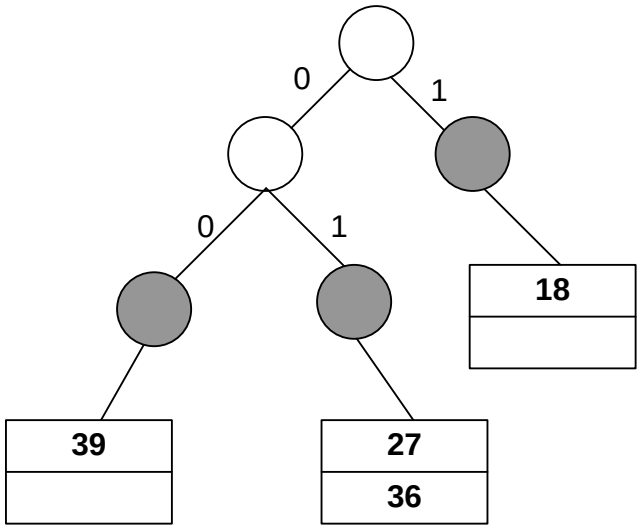
Záznamy v bloku budeme udržiavať utriedené: zefektívni to vyhľadávanie aj keď spomalí vkladanie.



Na začiatku štruktúra obsahuje iba 1 externý vrchol. Uložíme prvé 2 kľúče do jedného bloku bez uvažovania bitov reťazca.



Pri vložení 39 nastáva preplnenie, teda externý vrchol sa stáva interným a vložia sa dva nové externé ako jeho synovia. Pôvodné záznamy a vkladaný záznam sa rozdelia podľa prvého bitu ich reťazcov: 27, 39 vľavo (bit 0), 18 vpravo (bit 1).



Vloženie 36 preplňuje ľavý blok. Znovu sa nahradí externý vrchol interným s dvomi externými synmi a záznamy sa redidtribuujú podľa postupnosti prvých dvoch bitov príslušného bitového reťazca. Hĺbka ľavého podstromu vzrastie na D=2.

Ako rozšíriteľné hešovanie ale úspornejšia forma adresára: nezdvojnásobuje sa, ale rastie (klesá) plynule.