

### 3.3 Dynaamiset hakemistorakenteet

Käsittelyt hakemistot (hajautus, ISAM):  
hakemisto-osa on staattinen eli ei muutu muuten kuin uudelleenorganisoinnissa.

Ajan mittaan epätasapainoa:

- hajautuksessa solujen ylivuotoketjut
- ISAM: perusjaksojen ylivuotoketjut

ISAM: ylivuotoketjut 'tuhoavat' tasojen määrään perustuvan levyhakujen määrän laskennan.

Dynaamiset rakenteet:

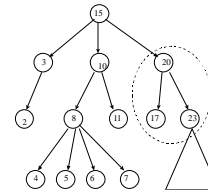
- rakennetta ylläpidetään vähittäisin muutoksin jatkuvasti tasapainossa
- lisäys- ja poisto-operaatiot tulevat vain jonkin verran monimutkaisemmiksi, laaja uudelleenorganisointi tarpeettomaksi
- jaksoja ei voida pitää aina täynnä → tilaa kuluu jonkin verran enemmän

35

### B-puu

Johdannoksi kerrataan puurakenne yleensä:

- juuri, lapset, vanhemmat
- lehtisolmut, sisäsolmut, alipuu
- solmun asteluku: binääripuu, yleinen puu
- solmun taso, puun korkeus



- solmun rakenne:
  - sisältö, esim. hakuavain
  - linkit lapsiin (ehkä myös isään)

Puurakenteen käyttö: esim. hakupuu

- solmut ohjaavat hakua seuraaville tasoille ja edelleen lehtisolmuihin asti
- vrt. ISAM-rakenne: hakemistojakso solmuna

36

Yleisen hakupuun solmu:

joukko osoittimia ja hakuavaimen arvoja

$\langle P_1, K_1, P_2, \dots, K(i-1), P_i, K_i, \dots, K(q-1), P_q \rangle$

järjestys:  $K_1 < K_2 < \dots < K(q-1)$

$P_i$  osoittaa alipuhun, jonka solmuille  $X$  pätee:

$$\begin{aligned} K(i-1) < X < K_i & \quad (i > 1, i < q) \\ X < K_1 & ; \quad X > K(q-1) \quad (\text{reunimmais}) \end{aligned}$$

(yhtäsuuruus ei tule kysymykseen, jos  $K_i$ :t todella avaimia eli yksikäsitteisiä)

Kun solmu on levyjakso,  $q$  voi olla kohtuullisen suuri  
=> puun korkeus pysyy "kurissa"  
(logaritmisuus, vrt. ISAM-laskelmat)

Solmussa voi olla

- koko tietueita (avaimet + vastaavat muut kentät), tai
- vain avaimia  $K$  => tarvitaan osoittimet vastaaviin tietueisiin (tietueiden dataan eli muihin attribuutteihin)

37

Hakupuu ei ole välttämättä tasapainossa; voi olla pieniä

alipuita ja pitkiä hakuketjuja

- puun muodostus lisäyksien sarjana?
- luonnin jälkeiset lisäys- ja poisto-operaatiot

Tasapainoinen hakupuu saadaan esim. ehdoilla, jotka määrittelevät B-puun (**kertaluokan p B-puu**):

1. B-puun sisäsolmu on muotoa

$\langle P_1, (K_1, D_1), P_2, \dots, (K(i-1), D(i-1)), P_i, (K_i, D_i), \dots, (K(q-1), D(q-1)), P_q \rangle$ ,

missä  $D_i$  on osoitin avaimen  $K_i$  tietueeseen ja  
 $P_i$  on osoitin puun seuraavan tason solmuun

$D_i$  voi olla jakso- tai tietueosoite.

2.  $K_1 < K_2 < \dots < K(q-1)$ .

3.  $P_i$ :n osoittamassa alipuussa olevat solmut ovat avainarvoltaan  $K(i-1)$ :n ja  $K_i$ :n välissä (vrt. hakupuu).

4. Jokaisessa solmussa on **enintään p osoitinta** (rakenneosoitinta puuhun).

5. Jokaisessa sisäsolmussa (paitsi juuressa) on **vähintään  $\lceil p/2 \rceil$  osoitinta**.

Juuressa on vähintään kaksi osoitinta, paitsi jos se on B-puun ainoa solmu.

38

6. Kohdan 1 mukaisessa solmussa on  $q$  puuosoitinta ja  $q-1$  avainta ja dataosoitinta ( $q \leq p$ ).

7. Kaikki lehtisolmut ovat samalla tasolla. Lehtisolmun rakenne on muuten sama kuin sisäsolmun, mutta puuosoitimet ovat tyhjiä.

Ominaisuudet 4 ja 5 takaavat B-puun tasapainon. Lehtien samantasoisuus seuraa puun konstruktiosta.

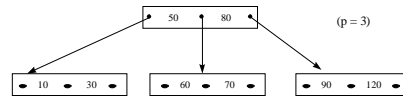
Avainarvon  $X$  haku B-puusta:

- haetaan avainta  $X$  juuresta
- jos ei löydy, seurataan sitä puuosoitinta, jolle pätee  $K(i-1) < X < K_i$  ja haetaan avainta  $X$  osoitetusta alipuusta (solmusta) (vrt. ISAM-hakemiston käyttö)

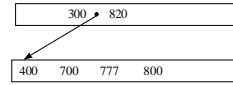
Lisäys ja poisto ovat monimutkaisempia, toteuttavat dynaamisuuden.

Toisin kuin ISAM-rakenne, B-puu muodostetaan vähitellen peräkkäisin lisäyksin; ei erillistä luontivaihtetta eikä uudelleenorganisointia.

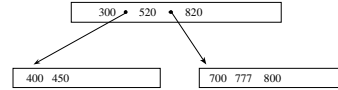
39



Lisäys B-puuhun: (p = 6)



- lisäään 520: mahtuu solmuun
- lisäään 450: ei mahdu



Lisäyksen jälkeen tilaa on monelle uudelle lisäykselle!

40

Lisäys B-puuhun:

- ensimmäinen lisäys tyhjäin juurisolmuun (muuta ei ole)
- $p-1$  lisäyksen jälkeen juureen ei voida enää lisätä (ehto 4: enintään  $p$  osoitinta  $\Rightarrow$   $p-1$  avainta)
- tällöin tarkastellaan  $p$  avaimen joukkoa (juuren solmut + lisättävä):
  - jätetään vain järjestyksessä keskimäinen avain juureen ja jaetaan muut kahteen osaan: kahdeksi tason 1 solmuksi (uusia solmuja, joille pätee ehto 5)
  - asetetaan osoittimet vastaamaan jakoa
- jaettaessa muu kuin juurisolmu menetellään kuten edellä sijoittamalla kuitenkin puolet solmuista uuteen solmuun samalla tasolla ja viemällä keskimäinen solmu jaettavan solmun isäsolmuun
- jos isäsolmussa ei ole tilaa, toistetaan menettely tälle jne (periaatteessa juureen asti)
- Jos jaot etenevät juureen asti ja jos juuri on täynnä, juuri jaetaan kahtia (ja perustetaan uusi juuri, joka jää tässä vaiheessa hyvin vajaaksi!). Tässä (ja vain tässä) tilanteessa puun korkeus kasvaa yhdellä.

41

Poisto B-puusta:

- poisto on yksinkertainen solmunsisäinen operaatio, mikäli poisto ei riko ehtoa 5 (solmu tulisi liian tyhjäksi).
- jos poiston jälkeen solmussa olisi alle  $p/2$  avainta, siihen siirretään vähintään yksi avain viereisestä (veljes)solmusta; jos avaimia on yhteensä liian vähän kahteen solmuun, ne yhdistetään yhdeksi solmuksi
- solmujen yhdistäminen poistaa yhden osoittimen ja avaimen solmujen isäsolmusta; jos siinä olisi nyt liian vähän osoittimia, jatketaan kuten edellä (yhdistäminen voi edetä pahimmillaan juureen asti).

Solmun avainlukumäärän ylä- ja alaraja takaavat sen, että jako ja yhdistäminen eivät toistu usein: yhden jaon tai yhdistämisen jälkeen ei tarvita uutta samaa operaatiota 'pitkään aikaan' (samassa kohdassa B-puuta).

Tasapainotilassa (satunnaiset lisäykset ja poistot) solmut ovat keskimäärin 69 %:sti täynnä.

42

Esimerkki. Olkoon  $K = 10$  tavua, levyjakso 4 KB, puuosoitin (jakso-osoite) 10 B ja dataosoitin (tietueosoite) 15 B.

B-puun solmu (enintään  $p$  osoitinta):

$$10p + (10+15) * (p-1) \leq 4096 \text{ eli } p \leq 117:$$

solmussa on avaimia 58 -116 kpl.

Em. 0.69-tuloksen mukaan osoittimia on noin 80 kpl.

=> taso	avaimia	osoittimia	solmuja
0 (juuri)	79	80	1
1	6320	6400	80
2	505600	512000	6400
yhteensä n. 512000			

Solmun sisällä avaimet ovat järjestyksessä, samoin solmut 'löydetään' järjestykseen perustuen.

Haku solmun sisältä ei vaadi (uusia) levyhakuja.

- erilaisia toteutustapoja (solmurakenteita)
- millaisia? –

(esim.  $q = 200-400$  aika suuri peräkkäishakuun ...)

Kaikkien avainten sujuva läpikäynti järjestyksessä vaatii useiden jaksoiden pitämisen puskurissa (yksi jakso / B-puun taso). (B+ -puu on tässä suhteessa parempi ...)

43

### B+ -puu

= B-puun variantti, joka sopii paremmin hakemistoksi:

- kaikki avainarvot ja dataosoittimet sijoitetaan B+ -puun lehtitasolle
- sisäsolmuissa säilytetään avainarvoja, jotka ohjaavat hakua (lehtitasolle etenemistä) kuten B-puussa
- lehtisolmut muodostavat tiheän hakemiston perustiedostoon
- lehtisolmut voidaan lisäksi ketjuttaa => koko tiedoston läpikäynti järjestyksessä on tehokasta

Solmurakenne:

- sisäsolmu:

< P1, K1, P2, ... , K(i-1), Pi, Ki, ... , K(q-1), Pq >

- lehtisolmu:

<K1, D1, K2, D2, ... , K(q-1), D(q-1), P<sub>next</sub>>

B+ -puun rakenteen ehdot sekä lisäys- ja poisto-operaatiot ovat käytännössä hyvin samanlaiset kuin B-puun (1-7 edellä).

Ehdossa 3 alipuun avaimet voivat olla yhtäsuuria kuin (isä)solmun avainarvo:

$K(i-1) < X \leq K_i$  isäsolmussa avainarvo toistuu, "viittana" (= suurin avainarvo osoitetussa alipuussa)

44

Sisäsolmujen avainten ei hakuperiaatteen kannalta tarvitse olla lehtisolmuissa esiintyviä ('sopivat' avainarvot ISAM-rakenteen tapaan). Muodostettaessa B+ -puu peräkkäisillä lisäysoperaatioilla kopioidaan tason 1 solmujen rajakohtia osoittaviksi avaimiksi kuitenkin oikeita avainarvoja, jotka myös säilyvät poistoissa.

Esimerkki solmujen koosta:

Olkoon  $K = 10$  tavua, levyjakso 4 KB, puuosoitin (jakso-osoite) 10 B ja dataosoitin (tietueosoite) 15 B (kuten edellisessä esimerkissä).

Sisäsolmu:  $10p + 10(p-1) \leq 4096 \Rightarrow p \leq 205$

Lehtisolmu:  $(15+10)p + 10 \leq 4096 \Rightarrow p \leq 163$

Täyttyästeen 0.69 mukaan sisäsolmussa on keskimäärin 141 ja lehtisolmussa 112 osoitinta.

B+ -puun rakenne muodostuu seuraavasti:

taso	avaimia	osoittimia	solmuja
0 (juuri)	140	141	1
1	19740	19881	141
2	2783340	2803221	19881

lehtitaso  $2803221 * 112 = 3.1 * 10^8$  tietueosoitinta

(2-tasoisessa hakemistossa  $19881 * 112 = 2226672$  tietueosoitinta)

Tasojen lukumäärä pysyy siis hyvin pienenä!

45

Kun juuri on vasta muodostunut, puu on hieman toisenlainen:

taso	avaimia	osoittimia	solmuja
0	1	2	1
1	2 x 102	2 x 103	2
2 (min:)	21012	21218	206

lehtitaso  $21218 \times 82 = 1.7 \times 10^6$  avainta

(B\* -puu:  $2/3 * p$  ..  $p$  osoitinta / solmu; jako ja yhdistäminen: kaksi solmua kolmeksi tai päinvastoin))

B+ -puu muodostaa luontevan **tiheän** hakemiston perustiedostolle, joka siis voi olla järjestämätön (kasa).

46

**Esim. Oraclen hakemistot:**

- perustiedosto on rakenteeltaan kasa: tietueet voidaan sijoittaa jaksoihin tiiviisti (100 % täyttöaste), niitä ei tarvitse siirtää (voi olla myös ryväs, jossa tietueita monesta relaatiosta)
- hakemistot B+ -puun muotoisia oheishakemistojä (optimoituja)
- hakemisto luodaan create index -lauseella:
  - create unique index nimihak on employee (column lname asc, fname asc);
- unique valinnainen, asc vastaa oletusarvoa (desc: laskeva järjestys, ei merkitystä):
  - unique takaa rivien yksikäsitteisyyden
  - tehokkuussyistä perusavaimelle suositellaan eheysrajoitteen (primary key ... tai unique) käyttöä, jolloin järjestelmä tekee hakemiston
- create index -lauseen tarkennuksia (mm.):
  - muistialue, varausyksiköt
  - pctfree=30 (jakson varatila),
  - nosort: perustiedosto järjestyksessä (ei tarvitse järjestää)

(Hakemiston luonti create index -lauseella muistuttaa tavallaan ISAM-hakemiston luontivaihetta - eikä B+ -puun rakentamista vähitellen.)

47

**Vielä yleistä hakemistoista:****Harva vai tiheä hakemisto?**

- perustiedosto järjestyksessä vai kasa?
- harvassa yksi hakemistotaso vähemmän kuin tiheässä
- tiheän hakemiston tietueet yleensä selvästi perustietueita lyhyempiä  
(=> voi kompensoida korkeuseron)

**Oheishakemistojen merkitys levyhakujen kannalta:**

Oletetaan, että kasararakenteeseen on luotu 10 ISAM-rakenteista oheishakemistoa, ja jokaisen korkeus on 3.

Operaatio insert into R values (...)

vaatii kaikkiaan

$2 + 10 * (2 + 1 + 1) = 42$  levyhakua, jos mikään hakemistojaksoista ei ole puskurissa.

Saantiajalla 10 ms aikaa kuluu siis noin 0.4 s.

B+ -puuhakemistoissa ylempien hakemistotasojen päivitys lisää joissakin tapauksissa (harvoin) muutamia levyhakuja. Tasojen määrään vaikuttaa mm. tietuerakenne.

Muita insert-operaatioon liittyviä toimintoja:

- muodostetaan lokitietueita elvytystä varten, niiden kirjoitus levyille
- viite-eheyksien tarkistukset (mahdollisesti paljon levyhakuja)

48

**Hakemistojen sijoittelu levyille on yleisesti tärkeä asia:**

lähelle perustiedostoa (esim. samalle sylinterille?) vai ei.

Esim. 1+1 levyhaun tilanteet (jos hakemisto kaukana, kumpikin levyhaku hidas).

Hakemisto luodaan yleensä vasta tiedoston luonnin (taulun sisällön lataamisen) jälkeen.

- jos luodaan peräkkäisillä lisäyksillä, kaikki lisäykset vaativat hakemiston päivittämisen (ei harva ISAM)
- optimointimahdollisuudet epätasapainon estämiseksi

Taulun koko: aivan pienille tauluille ei kannata tehdä hakemistoa - käytöstä riippumatta.

raja ?? (noin 8 jaksoa ? 200-300 tietuetta ?)

Esimerkiksi liitosoperaatio tehostuu hakemistoa käytettäessä: tiheä hakemisto (ilman perustietueita) riittää jopa liitosehdon testaamiseen (perustietueita hakematta!). – palataan ...

49

**Eri relaatioiden yhdistäminen samaan tiedostoon?**

E&N: 'file of mixed records'

**Tarve?**

Eri relaatioiden rivien käyttö yhdessä, esimerkiksi osasto ja osaston työntekijät

Tekniikka: fyysinen ryvästäminen ('klusterointi'): osaston rivi ja vastaavan osaston tietueet sijoitetaan lähemmäksi (samaa jaksoon, ainakin samalle sylinterille)

- monimutkaistaa tiedoston rakennetta: tarvitaan tietuetyyppi; tilan allokointi (eripituisia)

(Ryvästämisen ei ole oikein relaatiomallin henkeen kuuluvaa; paremmin hierarkkiseen tai verkkomalliin sekä oliotietokantoihin.)

**Oracllessa:**

```
create cluster personnel
(dept_number number(2))
create table emp
( ...
deptno number(2) not null)
cluster personnel(deptno)
```

```
create table dept
(deptno number(2), ...)
cluster personnel(deptno)
```

50

- ryväs voi olla indeksoitu tai hajautettu; indeksoidussa rypäässä kaikki samalla ryväsavaimella varustetut rivit ovat peräkkäin:

```
(d) 11 Administration ...
(e) 12345 Clark ... 11
(e) 23456 Jones ... 11
(e) . . . 11
(d) 13 Management ...
(e) 88888 Hill ... 13
(e) 99999 Adams ... 13
. . .
```

Osastonumero voidaan tallentaa vain jaksokohtaisesti (eli jättää pois tietueista) ym. optimointia.

- ryväs on SQL-kyselyjen kannalta 'läpinäkyvä', järjestelmä hoitaa sisäisesti.
- säästää tilaa
- hidastaa peräkkäiskäsittelyä (osastot tai työntekijät), lisäyksiä, ryväsavaimen päivityksiä

51

### Dynaamiset hajautusmenetelmät (oik. jatkaa lukua 2)

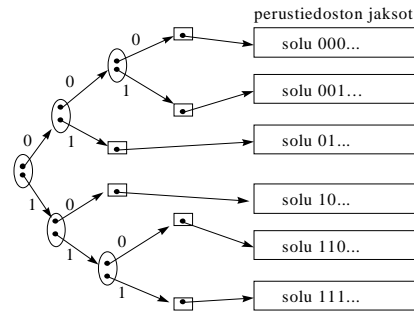
Tavoite: välttää soluun (kotiosoitteeseen) liittyvä tietueiden kasautuminen (ylivuotojaksojen ketjuuntuminen).

Erlaisia menetelmiä: dynaaminen hajautus, laajentuva hajautus, lineaarinen hajautus (ei käsitellä, E&N 5.9.3).

Dynaaminen hajautus: binääripuun muotoinen hakemisto, jonka jokaisessa lehtisolmussa on osoitin yhteen soluun.

Hajautusfunktion arvoa  $h(v)$  tarkastellaan binääriesityksensä mukaan (bittijonona) ensimmäisestä bitistä alkaen:

0-bitti osoittaa vasempaan alipuuhan, 1-bitti oikeaan



52

### Hakemiston dynaamisuus:

Kun solun jakso tulee täyteen, se jaetaan kahdeksi jaksoksi, solun tietueet jaetaan hajautusarvon seuraavan bitin mukaan kahteen jaksoon ja lisätään hakemistotietue (osoittimet vanhaan ja uuteen jaksoon).

Kun solu (esim. xyz0...) tyhjenee, se poistetaan, sisarussolun (solun xyz1...) hakemistopolulta jätetään yksi hakemistotietue pois ja liitetään sisarussolu edelliseen hakemistotietueeseen (xyz).

Hienojakoisempikin dynaamisuus on mahdollista: puolityhjä solu voidaan yhdistää sisarussoluunsa, jos tila riittää (täyttösuhteelle alaraja) jne.

Hakemistotietueet ovat tässä varsin pieniä, bittitason käsittelyn takia niitä (ja vastaavia tasoja) on tietysti enemmän kuin kokonaisia avainarvoja käsiteltäessä. Hakemisto voidaan kuitenkin usein pitää keskusmuistissa tai vain muutamassa levyjaksossa.

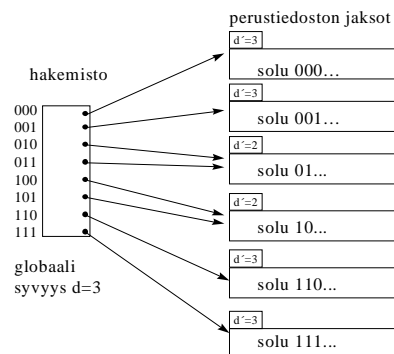
(Puurakenteen pakkaaminen levyjaksoihin on epätriviaali asia: tasoittain vai alipuittain?)

53

### Laajentuva hajautus (extendible hashing; E&N 5.9.3)

Ylläpidetään hakemistoa, jossa on  $2^d$  alkioita (hakemistotietuetta). Parametri  $d$  on hakemiston globaali syvyys.

Jokaisen solun (perustietueiden jakson) lokaali syvyys  $d'$  kertoo, moneenko hajautusavaimen bittiin solun osoitus perustuu.



Normaalitapauksessa 2 levyhakua: yksi hakemistoon ja yksi soluun (tai hakemisto keskusmuistissa).

54

### Dynaamisuus:

- ylivuoto solussa, jolle  $d' < d$  : otetaan käyttöön uusi perustiedoston jakso, jaetaan perustietueet uuteen ja vanhaan jaksoon ja päivitetään uuden jakson osoite hakemistossa (osoitteen paikka on olemassa)
- ylivuoto solussa, jonka  $d' = d$  : uuden perusjakson lisäksi kaksinkertaistetaan hakemisto eli asetetaan  $d \leftarrow d+1$  (esimerkissä esim. solu 110, hakemisto-osoitteet 1110 ja 1111 käyttöön)
- poisto analogisesti: yhdistetään tyhjät solut ja päivitetään osoitteet; hakemisto voidaan poiston yhteydessä puolittaa, jos kaikkien solujen  $d' < d$  (missään ei tarvita hajautusarvon viimeistä bittiä)