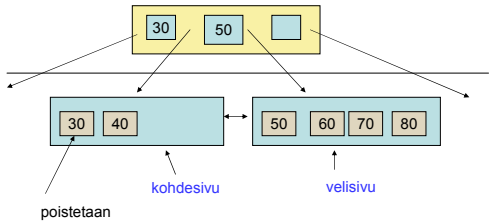
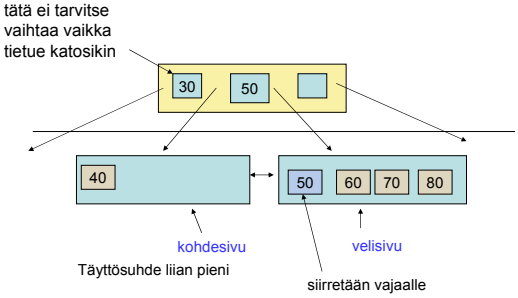
B+ -puun tasapainotus poistossa

- Poistot
  - Alkuperäisen B+ -puun idean mukaisesti tasapainotusta tehdään myös poistossa
  - Jos datasivun täyttösuhde laskee alle puoleen ja sivun ja sen velisivun (sibling, saman isäsivun alla oleva vierussivu) yhteenlaskettu tietomäärä ylittää ylittää sivukoon, järjestetään sivuparin tietueet uudelleen siirtämällä täydemmältä sivulta tietue vajaalle sivulle.

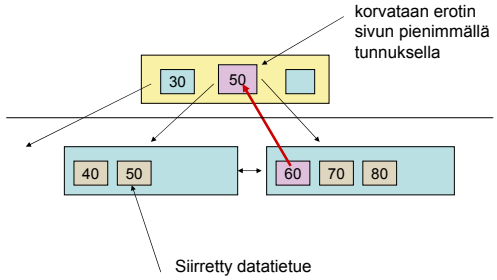
B+ -puun tasapainotus poistossa



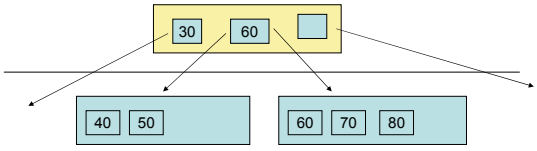
B+ -puun tasapainotus poistossa



B+ -puun tasapainotus poistossa



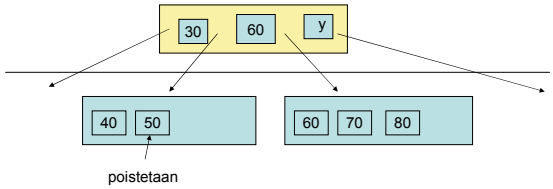
B+ -puun tasapainotus poistossa



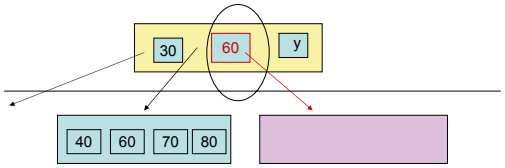
rakenne tasauksen jälkeen

B+ -puun tasapainotus poistossa

- Jos poiston kohteena olevan sivun ja sen velisivun yhteenlaskettu tietomäärä jää alle sivun kapasiteetin yhdistetään sivut ja poistetaan niiden välinen erotin hakemistosta.

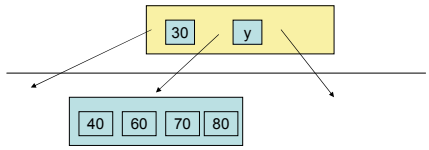


B+ -puun tasapainotus poistossa



siirretään kaikki velisivun tietueet kohdesivulle,
vapautetaan tyhjentynyt sivu,
poistetaan erotin hakemistosivulta

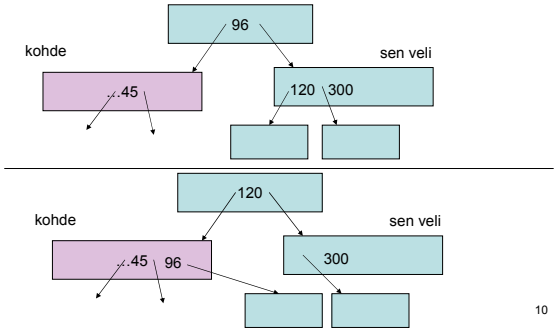
B+ -puun tasapainotus poistossa



B+ -puun tasapainotus poistossa

- Hakemistomerkinnän poiston takia hakemistosolmu voi jäädä vajaatäyttöiseksi
- Sivun täyttösuhdetta korjataan samalla periaatteella kuin datasivujenkin eli siirtämällä velisolmusta täydennystietue, siirto tehdään isäsolmun kautta vyöryttämällä
- Vyörytys: Olkoon kyseessä oikeanpuoleinen veli eli velisolmussa on isompia avaimia. Vyörytys tapahtuu tällöin seuraavasti
  - Lisätään isäsolmusta kopioitu kohdesolmun ja velisolmun välinen avain kohdesolmun avainlistan loppuun
  - Lisätään velisolmun osoitin p_0 kohdesolmun osoitinlistan loppuun
  - Korvataan isäsolmussa ollut kohdesolmun ja velisolmun välinen erotin velisolmun erotinlistan pienimmällä avaimella
  - Poistetaan velisolmun avain- ja osoitinlistojen alkupäästä ensimmäinen avain ja osoitin
  - Sivuja yhdistettäessä vyörytetään kaikki velisolmun tietueet kohdesolmuun ja poistetaan isäsolmusta erotin sekä osoitin velisolmuun
- Puun korkeus alenee, jos poistettavana on juurisolmun ainoa erotin.

Vyörytys



B+ -puun tasapainotus poistossa

- Kaikki toteutukset eivät käytä tasapainotusta poistojen yhteydessä, vaan antavat täyttöasteen laskea.
- B+ puun hakemistosivujen keskimääräiseksi täyttöasteeksi muodostuu lisäys- ja poistotasapainotusta käytettäessä noin 67%

B+ -puun ominaisuuksia

- Olkoon B+ -puun korkeus h
  - haettaessa indeksointiavaimen perusteella joudutaan tutkimaan h sivua (≤ h levyhakua)
  - lisättäessä joudutaan lukemaan h sivua ja kirjoittamaan vähintään 1 ja enintään h+2 sivua
  - poistettaessa joudutaan lukemaan vähintään h ja enintään 2h-1 ja kirjoittamaan vähintään 1 ja enintään 2h-1 sivua
  - Indeksointiavaimen arvon muuttaminen täytyy B+ -puun yhteydessä hoitaa poisto ja lisäys operaatioina.

B+ -puut tietokantatoteutuksessa

- Useimmat tietokannan hallintajärjestelmät mahdollistavat B+ -puun käytön joko
  - taulun toteutusrakenteena
    - datasivuilla on taulun rivejä vastaavia tietueita
    - ratkaisun huono puoli on se, että rivit voivat vaihtaa sivua, mikä tekee oheishakemistojen käytön työlääksi,
      - *kaikki osoitteet sivua vaihtaneeseen tietueeseen on vaihdon yhteydessä päivitettävä*
      - *Oraclessa tämä on ratkaistu siten, että B+ -puu rakenteisen tiedoston oheishakemisto ei käytäkään osoittamiseen tietueen osoitetta vaan tietueen pääavainta (haku hidastuu)*
  - tiheän hakemiston toteutusrakenteena
    - datasivuilla on tiheän hakemiston hakemistomerkintöjä
    - tämä on yleisin B+ -puun käyttötilanne.
    - taulun rivit ovat tällöin yleensä kasarakenteena

B+ -puu

- Edellä käsitellyt B+ -puun haku- ja lisäys-toiminnot perustuvat indeksointiavaimen yksikäsitteisyyteen
  - tietyllä avaimella varustettu tietue on tietyllä sivulla.
- Rakennetta voi kuitenkin käyttää myös toistuvien avainarvojen yhteydessä (indeksointi vaikkapa henkilön nimen perusteella)
  - vaihtoehto: datasivukohtaiset ylivuotolistat
  - yleisesti käytetty vaihtoehto (esim. Oracle, DB2): Otetaan rivitunnus (rid) mukaan indeksointiavaimeen tällöin avaimet ovat aina yksikäsitteiset

Indeksin luonti ja hävitys

- TKHJ:ssä on yleensä komento create index, jolla taululle voidaan luoda hakemisto
  - Komentoa ei ole standardoitu ja niinpä sen muoto vaihtelee järjestelmäkohtaisesti
- Indeksi voidaan luoda milloin tahansa taulun luonnin jälkeen
  - siis tyhjälle taululle tai
  - olemassaolevalle taululle
    - jos tauluun ladataan suuri määrä dataa voi olla kannattavaa luoda indeksit vasta latauksen jälkeen – saadaan aikaan parempi rakenne eikä jouduta jatkuviin rakenteen tasapainotuksiin
    - Tietueet järjestetään ensin ja lisätään järjestyksessä

Indeksin luonti ja hävitys

- Indeksit voidaan myös ’tiputtaa’ pois drop index komennolla
  - halutaan uudelleenorganisoida indeksi
    - tiputetaan se pois ja luodaan uudelleen
  - suoritetaan iso päivityserä (lisäyksiä/poistoja/indeksointiavaimen muutoksia)
    - on yleensä nopeampaa ja indeksien rakenteen kannalta parempi tiputtaa indeksit ja luoda ne uudelleen erän jälkeen

Dynaamiset hajautusratkaisut

- Aiemmin käsitelty hajautusrakenne perustui siihen, että tietueelle laskettiin soluosoite hajautusavaimen perusteella.
- Hajautusfunktio oli kiinteä ja sen arvoalue (osoiteavaruus) piti kiinnittää funktiota määriteltäessä.
  - Tällöin ei välttämättä ole riittävää tietoa siitä, miten hajautusfunktio jakaa tietueet todellisessa käyttötilanteessa eikä välttämättä edes tietoa siitä, miten nopeasti tietueiden määrä kasvaa. Solua laajentavan ylivuotoketjun pituutta ei voida hallita ja ainakin alussa täytyy varata tilaa paljon yli todellisen tarpeen.

Dynaamiset hajautusratkaisut

- B+ -puussa hakupolun pituus pidetään hallinnassa puolittamalla sivu ja pitämällä kummatkin puolikkaat samanmittaisen hakupolun päässä.
- Sivun puolituksen ideaa sovelletaan myös dynaamisissa hajautusrakenteissa. Tunnetuimpia näistä ovat
  - laajeneva hajautus (extendible hashing) ja
  - lineaarinen hajautus (linear hashing)

Laajeneva hajautus

- Laajenevassa hajautuksessa on käytössä kiinteä hajautusfunktio h ja sillä kiinteä osoiteavaruus
  - osoiteavaruutta ei kuitenkaan suoraan vastaa mikään tilavaraus, joten se voi olla hyvinkin suuri
  - tilavarauksessa lähdetään liikkeelle pienestä varauksesta ja sitä kasvatetaan tarpeen mukaan
- Rakenteessa käytetään soluhakemistoa, jonka koko on 2^d soluosoitetta. Eksponentti d (≥1) kasvaa tiedoston kasvaessa. Sitä kutsutaan rakenteen globaaliksi syvyydeksi (global depth).

Laajeneva hajautus

- Soluhakemiston koko on siis 2, 4, 8, …. solua
- Olkoon rakenteen globaali syvyys d. Tällöin avaimen k solu (siis soluhakemiston indeksi) saadaan eristämällä hajautusfunktion antaman osoitteen lopusta d bitin pituinen osa tail_bits(h(k),d).
  - Jos d=1 otetaan viimeinen bitti => {0,1}
  - Jos d=2 otetaan 2 viimeistä bittiä => {00,01,10,11}

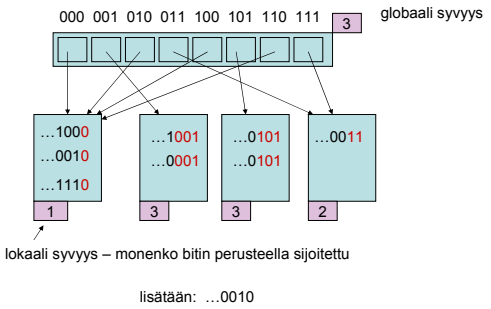
Laajeneva hajautus

- Lisäysten käsittely
  - Jos soluhakemistosta löytyvässä kotilohkossa on tilaa tietue lisätään sinne kuten aiemmin käsitellyssä hajautuksessa
  - Jos kotilohko (olkoon sen solutunnus binäärisenä {b}) on täynnä, otetaan käyttöön uusi lohko ja jaetaan ylivuotavan kotilohkon tietueet kotilohkon ja uuden lohkon välillä
    - kotilohkolla on paikallinen syvyys (local depth) p (monenko bitin perusteella tietueet on sijoitettu lohkoon)

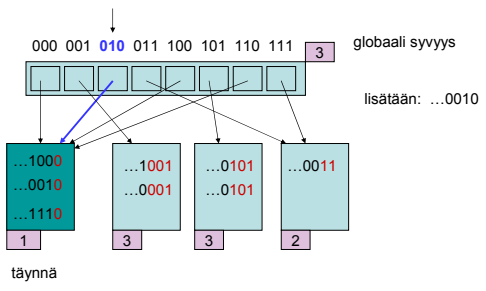
Laajeneva hajautus

- Lisäyksen käsittely jatkuu
  - tietueille tehdään uusi sijoittelu ottamalla käyttöön hajauttimen tuottaman osoitteen lopusta päin p+1:s bitti
  - ne tietueet, joilla tämä bitti on 0, jäävät kotilohkoon ja muut siirtyvät (oletetaan, että kaikki eivät menneet samaan lohkoon, näinkin voisi käydä ☹)
  - kotilohkon ja uuden lohkon paikalliseksi syvyydeksi asetetaan p+1
  - soluhakemiston alkiot, joiden indeksin p+1 viimeistä bittiä ovat 1{b} asetetaan osoittamaan uuteen lohkoon

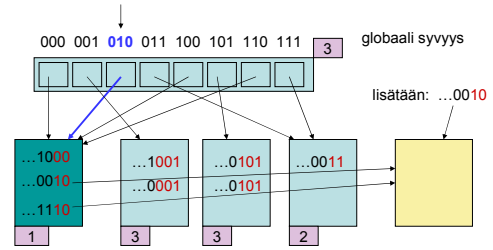
Laajeneva hajautus



Laajeneva hajautus

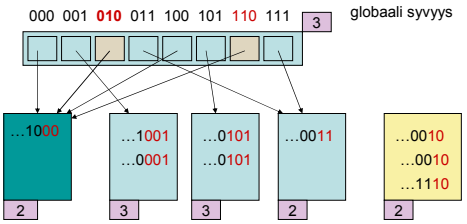


Laajeneva hajautus



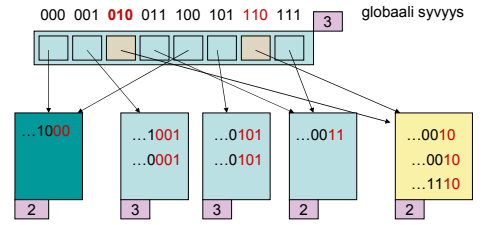
siirretään ne, joilla 2. viimeinen bitti on 1

Laajeneva hajautus



siirretään ne, joilla 2. viimeinen bitti on 1
vanhan ja uuden paikallinen syvyys 2
vaihdetaan ne soluosoittimet, joiden indeksi loppuu ..10 osittamaan uuteen

Laajeneva hajautus



siirretään ne, joilla 2. viimeinen bitti on 1
vaihdetaan ne soluosoittimet, joiden indeksi loppuu ..10 osittamaan uuteen

Laajeneva hajautus

- Jos jaettavan kotilohkon paikallinen syvyys on sama kuin rakenteen globaali syvyys, joudutaan tuplaamaan soluhakemiston koko, (jotta voitaisiin kirjata osoite soluun 1{b})
- Tuplaus tapahtuu helposti kopioimalla nykyinen soluhakemisto
- Tuplaus nostaa rakenteen globaalia syvyyttä yhdellä, kotilohkojen syvyydet säilyvät ennallaan

Laajeneva hajautus

- Rakenne laajenee sykäyksittäin
- Jos jakokohtaan osuvissa avaimissa on yhtenäinen bittisekvenssi, voidaan joutua tekemään monta jakoa ennen kuin tietueet saadaan jaettua
- Jos soluhakemisto kasvaa isoksi eikä sitä voida pitää keskusmuistissa voidaan hakuun tarvita 2 levyhakua
- jako ja tuplaus voivat edellyttää useita hakuja

Lineaarinen hajautus

- Lineaarisessa hajautuksessa (kotimaista alkuperää – Per Larsson) ei välttämättä tarvita soluhakemistoa
- Hajautusaluetta laajennetaan solu kerrallaan jakamalla jakovuorossa olevan solun sisältö solun itsensä ja uuden solun kesken
- Solut saavat jakovuoronsa järjestyksessä, eikä jaettava solu ole suinkaan välttämättä se jonka kohdalla ylivuoto tapahtuu – soluihin voidaan ylivuotavia tietueita varten liittää ylivuotolista

Lineaarinen hajautus

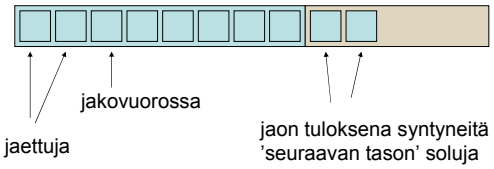
- Solun osoitteen määräämiseksi käytössä on sarja hajauttimia h_0, h_1, h_2, ...
- Nämä ovat muotoa $\mathbf{h_i} = h(k) \bmod (2^i N)$.
- N voidaan valita kakkosen potenssiksi 2^d, tällöin h_i eristäisi d+i bittiä perushajauttimen tuottaman arvon lopusta
  - jos d=5, niin h_0 eristää 5 bittiä, h_1 6 bittiä jne
  - tietueen haussa tarvitaan perushajauttimen lisäksi kahta hajautinta h_{taso} ja h_{taso+1}

Lineaarinen hajautus

- Haku
  - laske osoite $h_{taso}(h(k))$, eli ota d+taso bittiä lopusta
  - jos kyseessä on jakamaton solu, etsi tietuetta solusta.
  - jos kyseessä on jaettu solu muodosta uusi osoite $h_{taso+1}(h(k))$, eli ota d+1+taso bittiä lopusta
  - etsi tietuetta saadusta solusta
  - Solu on jakamaton, jos sen indeksi on suurempi tai yhtä suuri kuin jakovuorossa olevan solun indeksi

Lineaarinen hajautus

Olkoon d=1, taso=2, eli h_2 antaa osoitteet 0..7 ja h_3 osoitteet 0..15

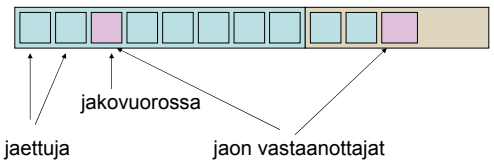


Lineaarinen hajautus

- Lisäyksessä, solmu lisätään haun määräämään lohkoon
  - jos solmu ei mahdu kotilohkoon, se lisätään ylivuotoketjuun.
  - Ylivuoto käynnistää jako-operaation
  - Olkoon jakovuorossa olevan solun taso+d bitistä muodostuva osoite {b}
  - Otetaan käyttöön uusi solu jonka osoite on 1{b}
  - jakovuorossa olevan solun tietueet jaetaan alkuperäisen soluun ja uuden solun kesken käyttäen hajautinta h_{taso+1}
  - Jos jakovuorossa oli tason viimeinen solu siirtyy jakovuoro soluun 0 ja tasoa kasvatetaan yhdellä, muuten jakovuoro siirtyy seuraavaan soluun.

Lineaarinen hajautus

Olkoon d=1, taso=2, eli h_2 antaa osoitteet 0..7 ja h_3 osoitteet 0..15



Lineaarinen hajautus

- Kullakin tasolla jaetaan vuorollaan jokainen tason solu. Kun kaikki on jaettu, on hajautusalue tuplautunut ja siirrytään seuraavalle tasolle.
- Rakenteessa voi olla pitkiäkin ylivuotoketjuja, mutta ne lyhenevät kun taso kasvaa.