



Järjestetty peräkkäisrakenne

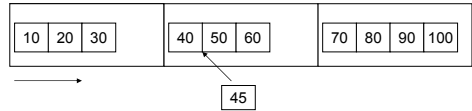
- Sequential file
- Tietueet sijaitsevat tiedostossa jonkin kentän (esimerkiksi työntekijänumeron) mukaan järjestettyinä.
 - Tietueiden järjestys on jokin tyypillisesti käsittelyssä usein tarvittava järjestys
 - Rakenne tukee tietueiden käsittelyä peräkkäin halutussa järjestyksessä

1



Järjestetty peräkkäisrakenne

- Lisäykset pitää järjestetyssä peräkkäisrakenteessa sijoittaa oikealle kohdalleen järjestyksessä.



- Pahimmassa tapauksessa pitäisi siirtää koko tiedoston loppuosa uusiin paikkoihin
- Yleensä jätetään lisäyksille tilaa sivuille – täytösuhde jää alhaiseksi
- Jos sivulle tulee lisäys, joka ei mahdu sinne, otetaan käyttöön ylivuotosivu, joka linkitetään sivuun. Lisäysten kasautuessa ylivuotoketjusta voi tulla pitkä
 - Rakenne degeneroituu nopeammin kuin kasa
 - Tarvitaan useammin uudelleenorganisointeja

2



Järjestetty peräkkäisrakenne

- Poistot ja muutokset tehdään kuten kasarakenteeseen.
- Tietueen haku tiedostosta järjestysavaimen perusteella vaatii tiedostoa läpikäymällä saman verran levyhakuja kuin kasarakenteessa, jos tietue löytyy eli $N/2$. Löytymättömyys voidaan todeta keskimäärin samalla levyhakujen määrällä. Tässä suhteessa rakenne on parempi kuin kasa.
- Järjestetty peräkkäisrakenne tukee myös järjestysavaimen perustuvia arvovälihakuja sekä erisuuruusvertailuja !=, <, <=

3



Järjestetty peräkkäisrakenne

- Järjestetty peräkkäisrakenne mahdollistaa myös binäärihaun käytön.
- Periaate:
 - Etsi tietuetta tiedoston keskimmaiselta ($k = \text{roof}(N/2)$) sivulta
 - Jos ei löytynyt ja avain < sivun pienin avain, jatka samalla periaatteella sivuilta 1 .. $k-1$.
 - Jos ei löytynyt ja avain > sivun suurin avain, jatka samalla periaatteella sivuilta $k+1$.. N .
 - Haku päättyy kun
 - tietue löytyy tai
 - tietuetta ei löydy sivulta mutta avain sijoittuu sivun pienimmän ja suurimman avaimen väliin (ei ole tiedostossa) tai
 - etsintää pitäisi jatkaa tyhjistä sivujoukosta

4



Järjestetty peräkkäisrakenne

- Binäärihaussa tarvitaan enintään $\text{roof}(\log_2 N)$ levyhakua.
- ($\text{roof}(x) = \text{pienin kokonaisluku, joka on suurempi kuin } x$)
- Aiemman esimerkin 8000 employee-tietueen tiedostossa oli 800 lohkoa, eli tietueen haku järjestysavaimen perusteella vaatisi enintään $\text{roof}(\log_2 8000) = 10$ levyhakua, joka satunnaishakuajalla laskettuna samalla n 10 ms satunnaishakuajan levyllä olisi samaa luokkaa kuin keskimääräinen saantiaika kasarakenteesta (100ms vs 85 ms)
 - Tosin tämä tiedosto voidaan sijoittaa kahdelle vierekkäiselle sylinterille, joten hakuvarren siirtoja ei juurikaan tule ja oikeasti päästään pahimmillaan vähän alle kasarakenteen keskimääräisen saantiajan (vain pyörähdysviive 5 ms lasketaan)
 - isommilla tiedostoilla binäärihaku parantaa asemaansa suhteessa koko tiedoston peräkkäiseen läpikäyntiin

5



Järjestetty peräkkäisrakenne

- Järjestetty peräkkäisrakenne ei ole tyypillinen tietokantojen yhteydessä käytetty tiedostorakenne
- Vanhanaikaisissa eräsovelluksissa se oli laajasti käytetty
 - soveltuu erityisen hyvin tilanteisiin, jossa pitää tahdistaa usean laajan aineiston käsittely

6



Hajautukseen perustuvat tiedostorakenteet

- Hajautukseen perustuvissa tiedostorakenteissa on tavoitteena yksittäisen tietueen nopea haku.
- Tähän pyritään siten, että tietueen sijoituspaikan eli solun (cell, bucket) osoite lasketaan jonkin tietueessa olevan tiedon eli hajautusavaimen (hash key) perusteella.
- Parhaassa tapauksessa solu olisi tietty tiedoston lohko. Yleensä solu kuitenkin muodostuu useasta lohkoista
 - enintään yhdestä kotilohkosta ja
 - joukosta ylivuotolohkoja (tai jatkolohkoja)
 - ylivuotolohkot voivat olla solukohtaisia jai jaettu

7



Hajautukseen perustuvat tiedostorakenteet

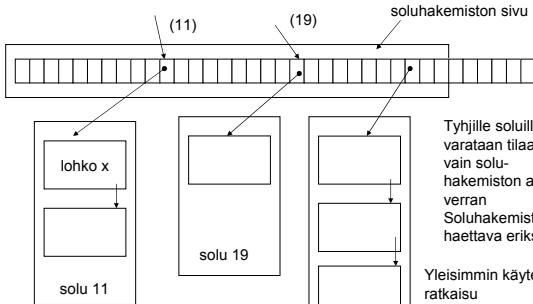
- Solun osoitteena on joko
 - kotilohkon osoite (tällöin ei käytetä soluhakemistoa) tai
 - soluhakemiston indeksi
 - soluhakemisto on taulukko, jonka alkioina on kotilohkojen osoitteita
- Mahdolliset soluosoitteet muodostavat osoiteavaruuden (address space)
 - soluhakemistoa käytettäessä saadaan pienellä lisätilalla kasvatettua osoiteavaruutta moninkertaiseksi

8



Hajautukseen perustuvat tiedostorakenteet

A) Solun osoite on soluhakemiston indeksi



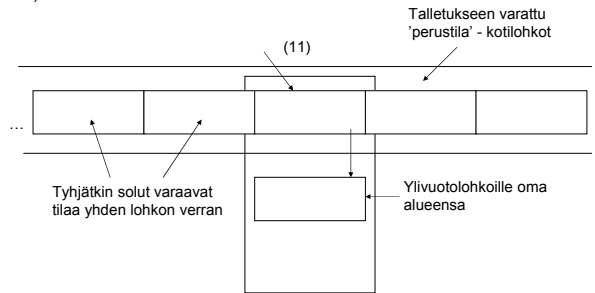
Lohkot käyttävissä koti- tai ylivuotolohkoiksi

9



Hajautukseen perustuvat tiedostorakenteet

B) Solun osoitteena on kotilohkon osoite



10



Hajautukseen perustuvat tiedostorakenteet

- Osoitteen laskentaan käytetään hajautusfunktiota (hajautinta) (hash function, randomizing function)
- Olkoon $h(X)$ hajautus funktio ja R_K tietue, jonka hajautusavain on K
- Tällöin R_K :n sijoitussolun osoite = $h(K)$.
- Solujen sisällä tietueet sijoitetaan kasarakenteen tapaan eli lisäykset loppuun

11



Hajautukseen perustuvat tiedostorakenteet

- Optimitapauksessa tietue löytyy yhdellä levyhaulla eli solun kotilohkosta*. Tämä edellyttää, että
 - hajautusfunktio jakaa tietueet tasaisesti osoiteavaruuden osoitteisiin
 - osoiteavaruus on määritelty riittävän isoksi niin, että kaikki tietueet voidaan sijoittaa solujen kotilohkoihin
- * soluhakemistoa käytettäessä joudutaan satunnaisesti hakemaan myös soluhakemiston sivuja

12



Hajautukseen perustuvat tiedostorakenteet

- Haku avaimella K:
 - lasketaan hajautusfuktiolla solun osoite $s=h(K)$
 - jos käytössä on soluhakemisto haetaan kotilohkon osoite sieltä $p=H[s]$ (jos hakemistoalkio on tyhjä, haku päättyy) muuten käytetään kotilohkon osoitteena solun osoitetta $p=s$.
 - haetaan sivu lohkoista s
 - ellei tietuetta löydy käydään läpi solun muut lohkot
 - Jos hajautusavain ei ole tietueen avain on haussa käytävä läpi kaikki solun sivut.

13



Hajautukseen perustuvat tiedostorakenteet

- Oletetaan, että tiedosto mahtuisi kasarakenneena N sivulle ja hajautusavaimella olisi k erilaista arvoa.
- Olkoon osoiteavaruus $0..B-1$
- Tällöin tulisi valita $B \leq k$ (muuten varataan tilaa soluille, jotka välttämättä jäävät tyhjiksi),
- Vain, jos $B \geq N$, voidaan päästä yhteen levyhakuun tietuetta haettaessa
- B:n kokoon vaikuttavat mm.
 - hajautusfunktio
 - hajautusavaimen rakenne
 - käytetäänkö soluhakemistoa, jolloin osoiteavaruuden koon kasvattaminen on kevyempää

14



Hajautukseen perustuvat tiedostorakenteet

- Hajautusfunktio on tyypillisesti muotoa $h(k) = \text{int}(k) \bmod B$
 - missä $\text{int}(k)$ muuntaa hajautusavaimen kokonaisluvuksi
- Hyvältä hajautusfunktiolta edellytetään, että se jakaa hajautusavaimen arvot tasaisesti osoiteavaruuteen
 - laskennan nopeus ei ole yhtä oleellista kuin keskusmuistihajautuksessa
- Hajautukseen perustuvat rakenteet nopeuttavat hakua vain kyselyissä, joissa tietueita haetaan hajautusavaimen perustuvan yhtäsuuruusehdon avulla.

15