Tiedostorakenteet

- Tiedostojen tehokkuutta yhtä kyselyä kohti arvioidaan usein tarvittavien levyhakujen määrällä
  - kuten levykäsittelyn yhteydessä todettiin levyhakuja on kahden tyyppisiä
    - satunnaishakuja ja
    - peräkkäishakuja
- Hajautetun tietokannan tapauksessa vaikuttaa tehokkuuteen myös verkon yli siirrettävän tiedon määrä

Tiedostorakenteet

- Kokonaistehokkuuteen vaikuttaa myös puskureiden määrä – 'oikean levyhaun' tarpeen todennäköisyys vähenee puskureiden määrän kasvaessa
- Myös levyjen ominaisuudet ja verkkoyhteyksien nopeus vaikuttavat kokonaistehokkuuteen.

Tiedostorakenteet

- Levyhakujen määrä on yksinkertainen ja selkeä tehokkuusmitta:
  - ei riipu laitteistosta
  - riippuu tiedostorakenteesta
  - arvioinnissa voi käyttää keskiarvoja, maksimia, jne.
    vrt. tietorakenteiden analyysi

Tiedostorakenteet

- Tiedostorakenteen tasolla vaikuttavia tekijöitä ovat mm.
  - tietueiden järjestys
  - tietueiden sijainti suhteessa toisiinsa
    - Tyypillisesti kunkin relaatiotietokannan taulun tiedot muodostavat oman tiedostonsa. Joissakin järjestelmissä, esimerkiksi Oraclessa, tietueita voidaan haluttaessa ryvästää (cluster) siten, että usein yhdessä tarvittavat tietueet (vaikkapa opiskelijatietue ja opiskelijan suoritustietueet) sijoitetaan lähekkäin mieluiten samalle sivulle
  - miten tietueita kytketään
    - tiedoston sisällä
    - tiedostojen välillä
    - käytetäänkö suoria osoitepohjaisia kytkentöjä vaiko relaatiokantojen avainperustaisia epäsuoria kytkentöjä

Tiedostorakenteet

- Tiedostorakenteet tarjoavat tiedostojen käsittelyyn varsinaisten datatietueiden sijoittelun lisäksi erilaisia apurakenteita (indeksejä), joilla pyritään nopeuttamaan käsittelyä
- Relaatiotietokantojen avainperusteiset kytkennät vaativat apurakenteita toimiakseen tehokkaasti
- Apurakenteet vaativat levytilaa, puskuritilaa, omia ylläpitotoimia lisäysten ja muutosten yhteydessä

Tiedostorakenteet

- Tietokannan kokonaistehokkuuteen vaikuttaa myös erilaisten tietokantaoperaatioiden osuus ja jakautuma kokonaiskuormassa
  - paljonko on tietyn tyyppisiä kyselyjä, milloin
  - paljonko on lisäyksiä, miten ne jakautuvat
- Operaatioiden tarpeet voivat olla ristiriitaisia
- Kuorma ei välttämättä pysy samanlaisena vaan muuttuu ajan myötä

Järjestämätön peräkkäisrakenne (kasa)

- Järjestämättömässä peräkkäisrakenteessa (heap) tietueet sijoitetaan peräkkäin tiiviisti tiedoston sivuille lisäysjärjestyksessä - uusi tietue lisätään aina tiedoston loppuun.
- Rakenne on useissa tkhj:ssä taulujen toteutuksen perusrakenne
- Rakenne soveltuu hyvin peräkkäiskäsittelyyn, jossa tietueiden järjestyksellä ei ole merkitystä

Järjestämätön peräkkäisrakenne (kasa)

- Taulun perustaminen create table –lauseella luo tyhjän kasan
  - yleensä tämä tarkoittaa yhtenäisen sivualueen varaamista tiedostolle, alueen koko voi olla järjestelmän päättämä tai siihen voi vaikuttaa esim. järjestelmäparametreilla tai create table –lauseella
  - Oraclen käyttää tästä alkuvarauksesta nimitystä initial extent ja sen koon voi määritellä taulukohtaisesti
  - jos tiedosto kasvaa niin suureksi, ettei se enää mahdu alkuperäiselle sivualueelle otetaan käyttöön jatkoalueita. Tyypillisesti alueet kytketään ketjurakenteeksi

Järjestämätön peräkkäisrakenne (kasa)

- Lisäys kasarakenteessa on nopeaa.
  - Haetaan tiedoston viimeinen sivu ja lisätään tietue sinne. Ellei tietue mahdu sivulle otetaan käyttöön seuraava sivu.
  - Korkeintaan 2 levyhakua (oletetaan kuvaajan olevan muistissa)
- Mutta, jos tiedostolle on määritelty avain, vaatii avaimen yksikäsitteisyyden tarkastus pahimmassa tapauksessa (silloin kun avain on yksikäsitteinen) koko tiedoston lukemisen

Järjestämätön peräkkäisrakenne (kasa)

- Jos tiedostossa on N sivua, vaatii tietueen haku avaimen perusteella keskimäärin N/2 levyhakua.
- Yleensä kasarakennetta ei käytetäkään yksinään, vaan sen päälle rakennetaan tiheitä indeksejä tehostamaan hakuja
  - esimerkiksi Oraclessa avaimen määrittely luo automaattisesti avainperustaisen indeksin.

Järjestämätön peräkkäisrakenne (kasa)

- Tietueen poisto edellyttää
  - tietueen hakua
  - tietueen poistamista haetulta sivulta
    - joko merkitsemällä tietue poistetuksi tai tiivistämällä sivu
  - muuttuneen sivun vientiä takaisin levylle
- Poistojen seurauksena sivuille tulee tyhjää tilaa ja tiedoston täyttösuhde pienenee
  - vapautunut tila saadaan helposti käyttöön vain, jos muutokset kasvattavat jotain sivulla olevaa tietuetta.

Järjestämätön peräkkäisrakenne (kasa)

- Jos poisto tyhjentää koko sivun, voidaan sivu liittää vapaiden sivujen ketjuun uudelleenkäytettäväksi.
- Vaihtuvapituisten tietueiden muutokset voivat kasvattaa tietueen pituutta.
  - Tietueiden pituuden kasvuun voidaan varautua jättämällä sivuille kasvuvaraa eli sivuja ei täytetä heti aluksi tiiviisti vaan vaikkapa vain 70 prosenttisesti.
  - Voi käydä myös niin, että kasvanut tietue ei enää mahdu sivulle. Tällöin käytetään tyypillisesti ylivuotosivuja tietueiden ylivuotaneiden osien tallennukseen. Tietueita ei siirretä kokonaan toiselle sivulle, koska silloin jouduttaisiin ylläpitämään indeksejä. Ylivuotosivut ovat huonoja peräkkäiskäsittelyn kannalta.

Järjestämätön peräkkäisrakenne (kasa)

- Poistot ja muutokset degeneroivat rakennetta
  - rakenteesta tulee harva ja epäyhtenäinen
  - rakenne voidaan korjata ajoittaisin uudelleenorganisoinnein
    - uudelleenorganisoinnissa luodaan uusi kasarakenne ja korvataan sillä vanha
    - kaikki vanhan rakenteen varaan rakennetut indeksit on rakennettava uudelleen.

Järjestämätön peräkkäisrakenne (kasa)

- Kyselyn
- Tarkastellaan taulua employee. Oletetaan että taulussa on 8000 riviä (ei ihan pikkufirma).
- Tehdään kysely
- select * from employee where ssn='1234567'

- Joudutaan läpikäymään koko tiedosto ellei tiedetä, että ssn on avain ja keskimäärin puolet jos se on määritelty avaimeksi ja löytyy (ensimmäinen osuma on ainoa) siis keskimäärin 4000 riviä

Järjestämätön peräkkäisrakenne (kasa)

- Olkoon levyn pyörimisnopeus on 100 kierrosta sekunnissa, lohkokoko 4K tavua, uralla 50 lohkoa, levypintoja 12 sekä täyttösuhde 80%.
- Yhden Employee-tietueen keskipituus voisi olla 300 tavua.
- Lohkossa olisi tällöin keskimäärin 10 tietuetta, uralla 500 ja sylinterillä 6000, eli tarvitaan 2 sylinteriä ja yhteensä 16 kierrosta koko tiedoston siirtämiseen eli maksimissaan 16*10 ms=160 ms +kohdistusaika +pyörähdysviive (yht. 10ms), eli noin 170 ms ja avainpohjaisessa haussa keskimäärin puolet tästä eli n 85 ms (olettaen, että tietue löytyy)