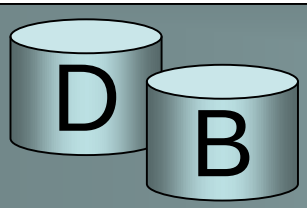
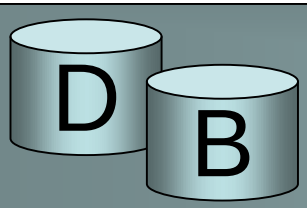


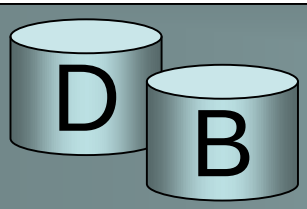
Yksitasoiset talletusrakenteet

- Yksitasoisia talletusrakenteita käytetään lähinnä datatietueiden talletukseen
  - järjestämätön peräkkäisrakenne (kasa, heap)
  - järjestetty peräkkäisrakenne (sequential file)
  - hajautusrakenne



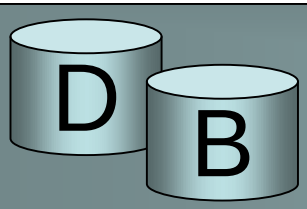
Järjestämätön peräkkäisrakenne (kasa)

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



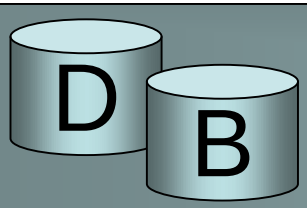
Järjestämätön peräkkäisrakenne (kasa)

- 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 (chain)



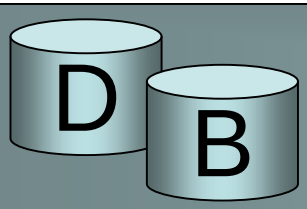
Järjestämätön peräkkäisrakenne (kasa)

- 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



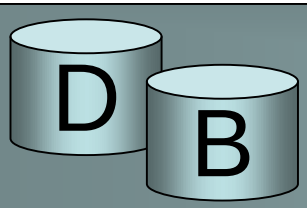
Järjestämätön peräkkäisrakenne (kasa)

- 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.
- Tietuejoukon haku edellyttää koko tiedoston läpikäyntiä



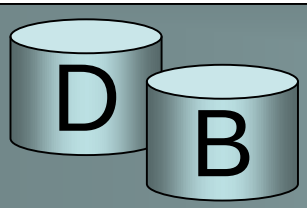
Järjestämätön peräkkäisrakenne (kasa)

- Tietueen poisto edellyttää
  - tietueen hakua
  - tietueen poistamista haetulta sivulta
    - yleensä merkitsemällä tietue poistetuksi
    - tietueelta vapautunut tila voidaan jättää käyttämättä tai vapauttaa muiden sivujen käyttöön joko
      - *yhtenäistämällä sivu tai*
      - *liittämällä tietueelle varattu alue vapaiden alueiden ketjuun.*
  - muuttuneen sivun vientiä takaisin levylle
- Poistojen seurauksena sivuille tulee tyhjää tilaa ja tiedoston täyttösuhde pienenee.



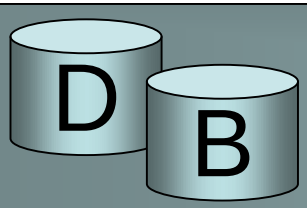
Järjestämätön peräkkäisrakenne (kasa)

- 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 täyteen 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)

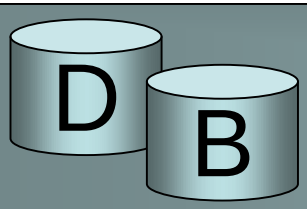
- Poistot ja muutokset degeneroivat rakennetta
  - rakenteesta tulee harva ja epäyhtenäinen
  - rakenne voidaan korjata ajoittaisin uudelleenorganisoinnein
    - uudelleenorganisoinnissa luodaan uusi kasarakenne ja korvataan sillä vanha
    - kaikki vanhan rakenteen varaan rakennetut indeksit on uudelleenorganisoinnissa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



Järjestämätön peräkkäisrakenne (kasa)

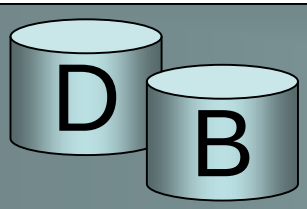
- Kyselyn
- Tarkastellaan taulua employee. Oletetaan että taulussa on 8000 riviä (ei ihan pikkufirma).
- Tehdään kysely
- select * from employee where ssn=’1234567’

- 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



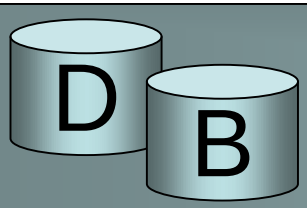
Järjestämätön peräkkäisrakenne (kasa)

- Olkoon levyn pyörimisnopeus on 100 kierrosta sekunnissa, lohkokoko 4K tavua, uralla 50 lohkoa, levypintoja 12 sekä täyttösuhde 80%.Kohdistusaika keskimäärin 5ms.
- Yhden Employee-tietueen keskipituus voisi olla 300 tavua.
- Lohkossa olisi tällöin keskimäärin 10 tietuetta, uralla 500 ja sylinterillä 6000, eli tarvitaan 16 uraa (alle 2 sylinteriä) ja yhteensä 16 kierrosta koko tiedoston siirtämiseen.
- Oletetaan että käytössä on ns. kaksoispuskurointi, eli puolet puskureista on käsittelyssä samalla kun toiseen puoleen luetaan. Tällöin luku voi jatkua keskeytyksettä jos ohjelma ehtii käsitellä puskurit nopeammin kuin luku etenee. Aikaa kuluisi tällöin kohdistusaika+pyörähdysviive+16*siirtoaika (+yksi lyhyt sylinterisiirtymä)=
- 5ms+5ms+16*10 ms(+10ms)=180 ms ja avainpohjaisessa haussa keskimäärin puolet tästä eli n 90 ms (olettaen, että tietue löytyy)



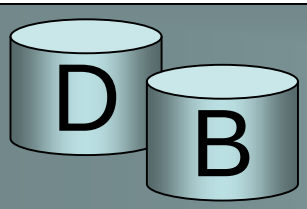
Järjestämätön peräkkäisrakenne (kasa)

- Rakenne voitaisiin toteuttaa myös sivujen ketjuna, mutta tällöin menetettäisiin peräkkäishausta saavutettava etu tietueiden haussa
- Esimerkin tapauksessa koko tiedoston hakuun kuluisi 800 *(10ms+0.2ms) = 8160 ms ja vaikka kohdistusaika jatetään huomiotta niin 800*(5ms+0.2ms)=4160ms (>4s) ja avainhakuun puolet tästä.



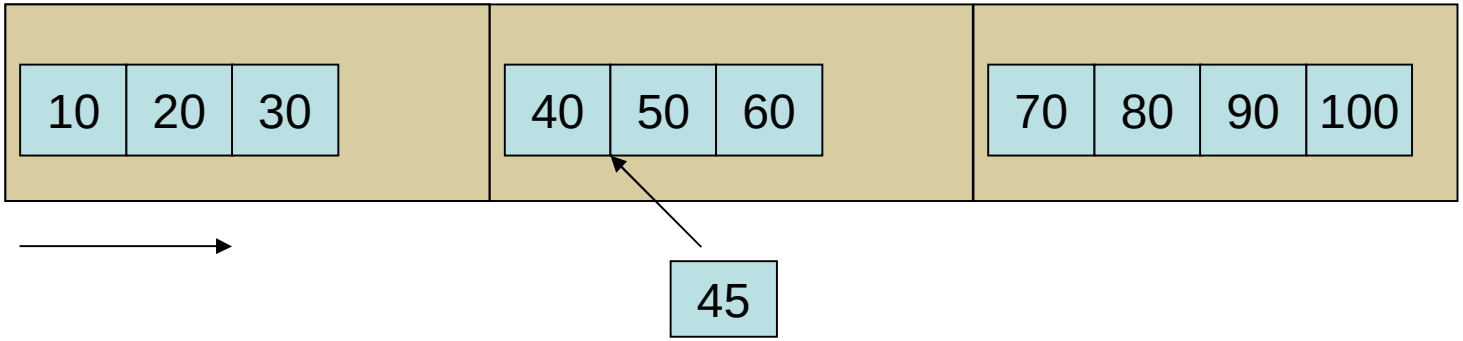
Järjestetty peräkkäisrakenne

- 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

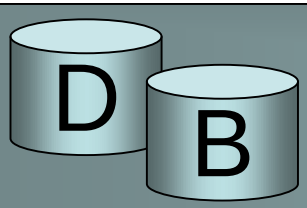


Järjestetty peräkkäisrakenne

- Lisäykset pitää järjestetyssä peräkkäisrakenteessa sijoittaa oikealle kohdalleen järjestyksessä.

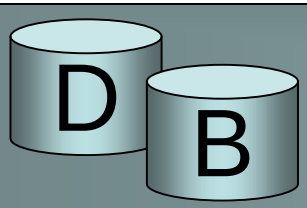


- Pahimmassa tapauksessa pitäisi siirtää koko tiedoston loppuosa uusiin paikkoihin
- Yleensä jätetään lisäyksille tilaa sivuille – täyttösuhde jää alhaiseksi
- Jos sivulle tulee lisäys, joka ei mahdu sinne, otetaan käyttöön ylivuotosivu, joka linkitetään sivuun. Lisäysten kasautuessa ylivuotoketjusta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



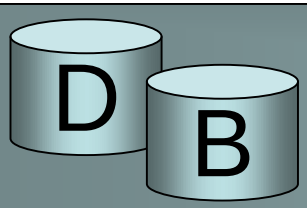
Järjestetty peräkkäisrakenne

- Poistot ja muutokset tehdään kuten kasarakenteeseen.
- Tietueen haku tiedostosta järjestysavaimen perusteella vaatii tiedostoa läpilukemalla 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 järjestysavaimeen perustuvia arvovälihakuja sekä erisuuruusvertailuja !=,<,<=



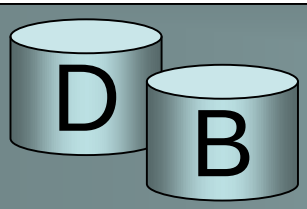
Järjestetty peräkkäisrakenne

- Järjestetty peräkkäisrakenne mahdollistaa myös binäärihaun käytön.
- Periaate:
  - Etsi tietuetta tiedoston keskimmäiseltä (k=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 tyhjästä sivujoukosta



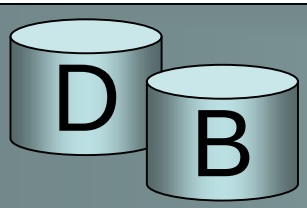
Järjestetty peräkkäisrakenne

- Binäärihaussa tarvitaan enintään roof($\log_2 N$) levyhakua.
- *(roof(x) = pienin kokonaisluku, joka on suurempi kuin x)*
- Aiemman esimerkin 8000 employee-tietueen tiedostossa oli 800 lohkoa, eli tietueen haku järjestysavaimen perusteella vaatisi enintään roof($\log_2 800$) = 10 levyhakua, joka satunnaishakuajalla laskettuna samalla n 10 ms satunnaishakuajan levyllä olisi hieman vähemmän kuin keskimääräinen saantiaika kasarakenteesta
- 10*(10ms+10/50*10ms)=102ms vs. 160 ms
  - Tosin tiedosto voidaan sijoittaa kahdelle vierekkäiselle sylinterille, joten hakuvarren siirtoja ei juurikaan tule ja jos lasketaan vain pyörähdysviive 5 ms) niin saataisiin 10*5ms+2ms= 52 ms
- Isommilla tiedostoilla binäärihaku parantaa asemaansa suhteessa koko tiedoston peräkkäiseen läpikäyntiin



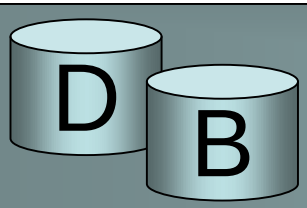
Järjestetty peräkkäisrakenne

- 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



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 lohkosta
  - enintään yhdestä kotilohkosta ja
  - joukosta ylivuotolohkoja (tai jatkolohkoja)
    - ylivuotolohkot voivat olla solukohtaisia tai jaettuja

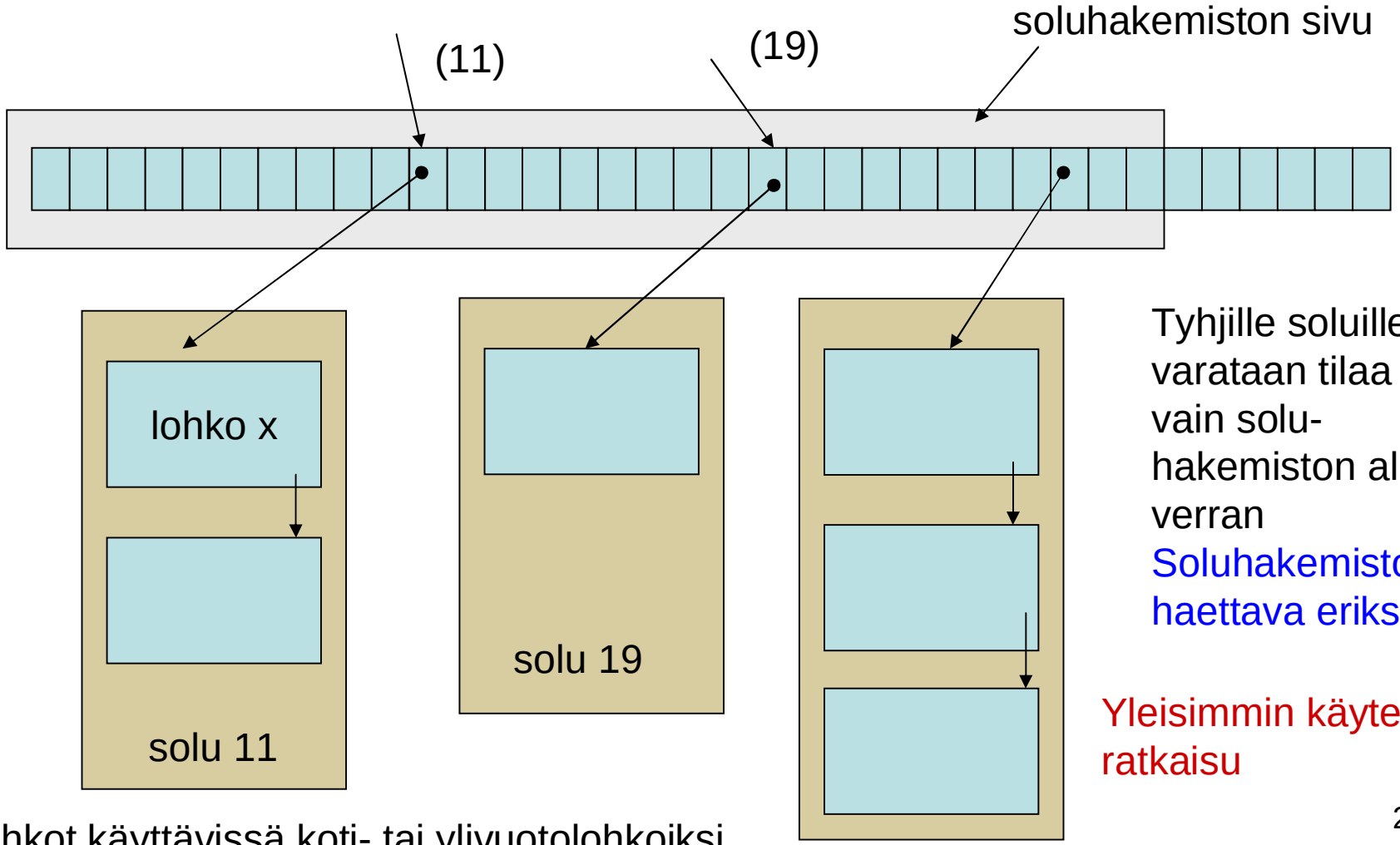


Hajautukseen perustuvat tiedostorakenteet

- 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

Hajautukseen perustuvat tiedostorakenteet

A) Solun osoite on soluhakemiston indeksi

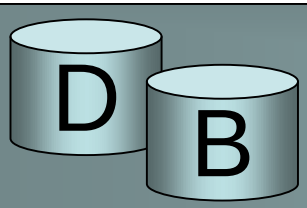


Tyhjille soluille varataan tilaa vain solu-hakemiston alkion verran

Soluhakemisto on haettava erikseen

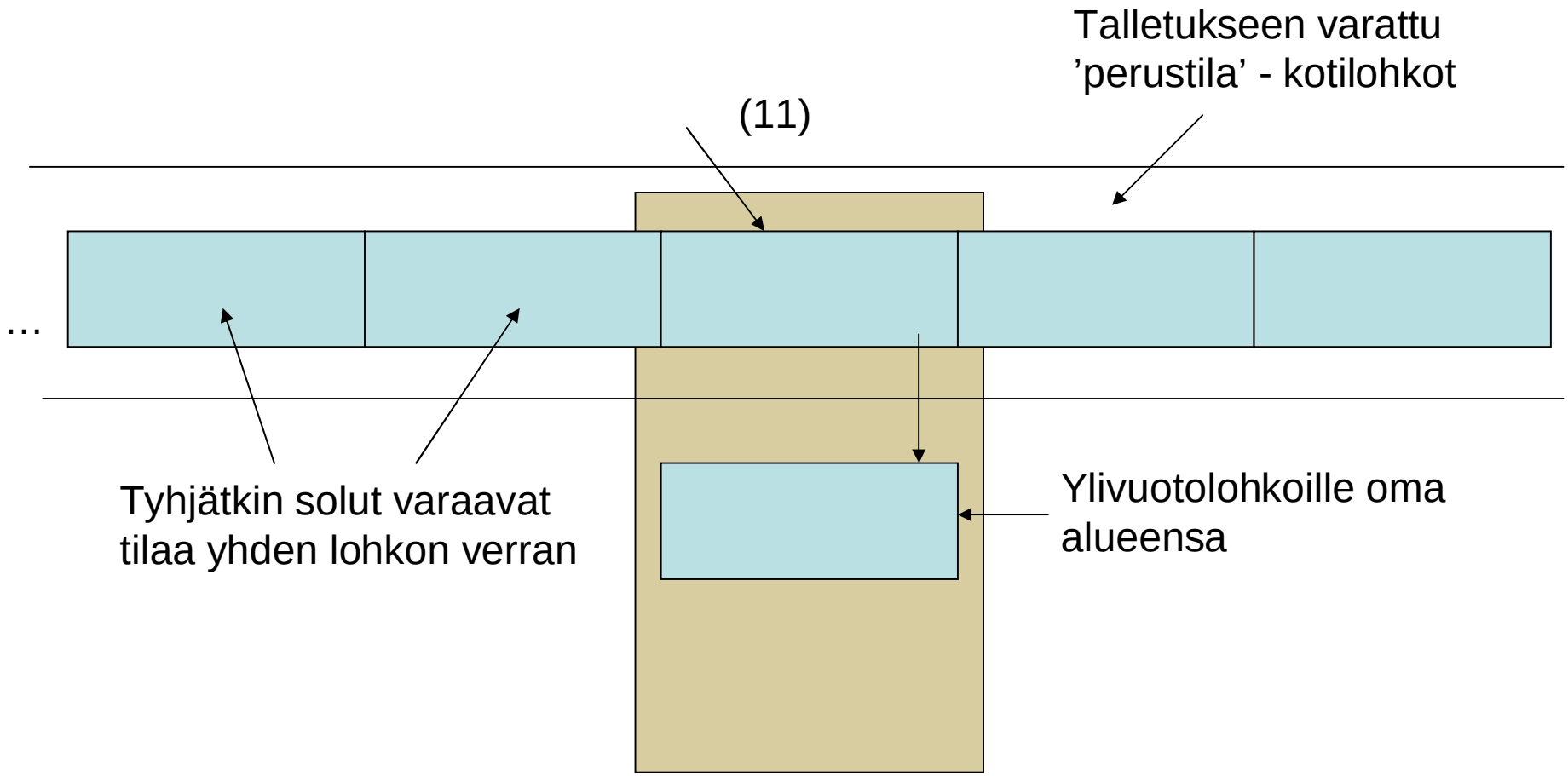
Yleisimmin käytetty ratkaisu

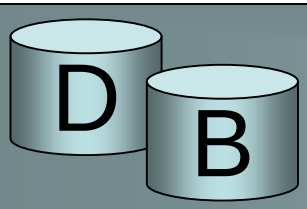
Lohkot käyttävissä koti- tai ylivuotolohkoiksi



Hajautukseen perustuvat tiedostorakenteet

B) Solun osoitteena on kotilohkon osoite

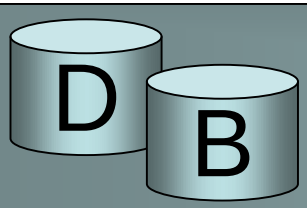




Hajautukseen perustuvat tiedostorakenteet

- 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).

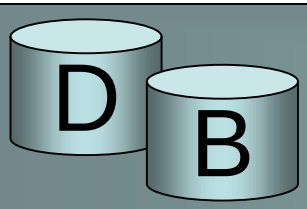
- Solujen sisällä tietueet sijoitetaan kasarakenteen tapaan eli lisäykset loppuun



Hajautukseen perustuvat tiedostorakenteet

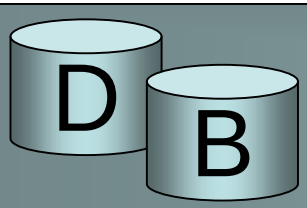
- Optimitapauksessa tietue löytyy yhdellä levyhaulla eli solun kotilohkosta*. Tämä edellyttää, että
  - hajautusfunktio jakaa tietueet tasaisesti osoiteavaruuden osoitteisiin
  - osoiteavaruus on määritelty riittävän isoksi niin, että kaikki tietueet voidaan sijoittaa solujen kotilohkoihin

* soluhakemistoa käytettäessä joudutaan satunnaisesti hakemaan myös soluhakemiston sivuja



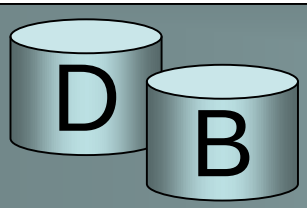
Hajautukseen perustuvat tiedostorakenteet

- Haku avaimella K:
  - lasketaan hajautusfunktiolla 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 lohkosta p
  - ellei tietuetta löydy käydään läpi solun muut lohkot
  - Jos hajautusavain ei ole tietueen avain on haussa käytävä läpi kaikki solun sivut.



Hajautukseen perustuvat tiedostorakenteet

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



Hajautukseen perustuvat tiedostorakenteet

- Hajautusfunktio on tyypillisesti muotoa

  $h(k) = int(k) \bmod B$

  – missä 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 hajautusavaimeen perustuvan yhtäsuuruusehdon avulla.