



Levytiedoston käsittely

- Olkoon
 - S= kohdistusaika
 - R= pyörähdysviive
 - T= jakson siirtoaika= kierrosaika/uran lohkojen lukumäärä.
- Lohkon saantiaika = $S+R+T$
 - kohdistusaika ja pyörähdysviive vaihtelevat, arvioissa käytetään usein keskimääräistä hakuaikaa
- **Sivupyynnön toteutuksen palveluaikaan on vielä lisättävä jonotusaika sillä levyohjain voi suorittaa vain yhtä pyyntöä kerrallaan – jonotusaikaan vaikuttaa kaikki levyille kohdistuva kuorma**

1



Levytiedoston käsittely

- Haettaessa k lohkoa, lohkojen yhteenlaskettu saantiaika on
 - enintään $k*(S+R+T)$, jos lohkot sijaitsevat satunnaisesti levyllä
 - $S+k*(R+T)$, jos lohkot sijaitsevat satunnaisesti samalla sylinterillä ja
 - minimissään $S+R+k*T$, jos lohkot ovat peräkkäin samalla sylinterillä
- Kaavojen tekijöistä S ja R ovat samaa millisekunneissa mitattavaa kokoluokkaa ja T noin tuhannesosa niistä (mikrosekunneja)
- Jos $S=5\text{ms}$, $R=5\text{ms}$ ja $T=0.02\text{ms}$ ja $k=1000$, niin kokonaissaantiajan vaihteluväli olisi **0.030s - 10.02s**

2



Levytiedoston käsittely

- Tiedostot sijoitetaan levyille
 - yhdelle tai useammalle peräkkäisistä lohkoista koostuvalle alueelle
 - voidaan hyödyntää peräkkäiskäsittelyn nopeutta koko tiedoston lukemisessa
 - puskurien hallinnassa voidaan käyttää ns. **ennakoivaa haku (pre-fetch)** – haetaan valmiiksi puskureisin järjestyksessä seuraavia sivuja vaikka niitä ei ole vielä pyydetty

3



Levytiedoston käsittely

- Vaikka tiedoston tietueet alunperin sijoitettaisiin tiiviisti lohkoihin voivat poistot ja muutokset huonontaa rakennetta
 - peräkkäisjärjestykseen jää väliin tyhjiä lohkoja poistojen takia
 - jatkotietueet yliviuotolohkoissa vaativat poikkeamista optimaalisesta lukemisjärjestyksestä – mahdollisesti jopa siirtymistä pois sylinteriltä

4



RAID-levyt ja rinnakkaisuus

- RAID = **redundant array of independent disks**
- ryhmä toisistaan riippumattomia levyjä, jotka toimivat rinnakkain yhtenä **loogisena levyinä**
- tavoitteena luotettavuuden ja tehokkuuden parantaminen
 - tehokkuutta parannetaan **viipaloinnilla (striping)**
 - luotettavuutta **redundanssilla** (ylimäärällä)

5



RAID-levyt ja rinnakkaisuus

- Viipaloinnin periaatteena on jakaa tiedosto usealle levyille. Tiedonsiirtoa levyjen ja puskureiden välillä voidaan suorittaa rinnakkain koska jokainen levyohjain toimii itsenäisesti. Näin tiedonsiirtonopeus kasvaa.
- Käytössä on eri **viipalekokoihin (striping unit)** perustuvia viipalointiperiaatteita. Jos käytettävissä on D levyä, sijoitetaan i :s viipale levyille $i \bmod D$.

6



RAID-levyt ja rinnakkaisuus

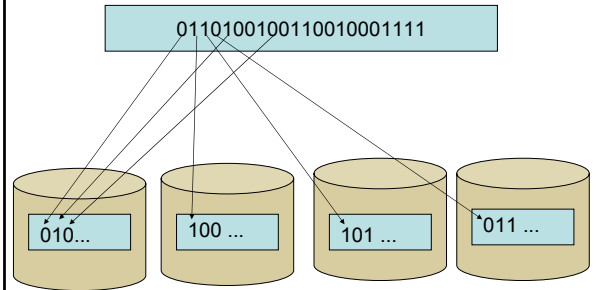
- Viipalekoko voisi olla **yksi bitti**.
 - Tällöin D peräkkäistä bittiä ovat kukin eri levyillä ja kaikki D levyä osallistuvat jokaiseen hakuoperaatioon.
 - Koska **pienin siirtoyksikkö on lohko** joudutaan **sivupyynnön yhteydessä siirtämään D lohkoa**.
 - Jos käsittely on peräkkäiskäsittelyä, on siirron yhteydessä tehty ennakkoiva haku D-kertaisella siirtonopeudella
 - Jos käsittely on hajakäsittelyä, on siirretty turhaa dataa, mutta siihen ei ole mennyt enempää aikaa kuin yhtä levyä käytettäessä

7



RAID-levyt ja rinnakkaisuus

yhden bitin viipalekoko – 4 levyä



8



RAID-levyt ja rinnakkaisuus

- Lohkotason viipaloinnissa viipaleen koko on yksi lohko.
 - Jos sivupyynnot kohdistuvat eri levyille voidaan pyynnot suorittaa rinnakkain. Jonotusaika pienenee joten myös hajakäsittely nopeutuu.
 - Peräkkäiskäsittelyssä peräkkäiset lohkot sijaitsevat eri levyillä, joten tiedonsiirtonopeus kasvaa
 - Suurella levymäärällä saadaan suuremmat tehot

9



RAID-levyt ja rinnakkaisuus

- Tietojen jakaminen usealle levyille aiheuttaa luotettavuusongelman.
- **N levyn järjestelmässä häiriön todennäköisyys on N-kertainen yhteen levyyn verrattuna**
- Jos yhden levyn kohdalla häiriövälin odotusarvo on 50000 tuntia (5.7 vuotta), niin 100 levyn järjestelmässä se onkin enää $50000/100 = 500$ tuntia eli noin 21 päivää
- Luotettavuutta lisätään **ylimäärällä**, jolloin häiriöitä voi sietää.

10



RAID-levyt ja rinnakkaisuus

- Eräs ylimäärän muoto on **peilaus (mirroring)**
 - Tieto kirjoitetaan kahdelle identtiselle levyille.
 - Jos toinen levy vikaantuu, voidaan jatkaa toisella kunnes vika saadaan korjattua.
 - Kumpakin levy voi käyttää sivupyynnöiden toteutukseen, pyyntö ohjataan levyille, jolla on lyhyempi saantiaika.
 - Levyjen tehollinen kapasiteetti puolittuu
 - Kallis ratkaisu

11



RAID-levyt ja rinnakkaisuus

- Peilauksen vaihtoehtona on erilaisia virheenkorjauksen mahdollistavia tekniikoita, Hamming koodi, bittitason tai lohkotason viipalointiin perustuvat pariteettibitit omalla levyllään tai hajautettuna yksikön levyille.

12



Tietojen tallennusrakenteet

- Jokaisella tiedostolla on **otsake (header)**, joka sisältää tiedostoon liittyvää hallintatietoa
 - tiedot tiedostoon kuuluvista lohkoista esim. taulukkona, joka voi muodostua ketjutetuista osista
 - tietoja tietueiden rakenteesta
 - muuta hallintatietoa

13



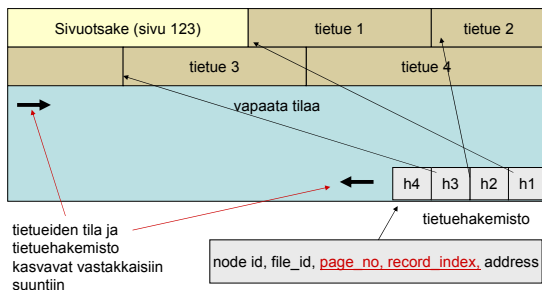
Tietojen tallennusrakenteet

- Jokaiseen sivuun sisältyy **sivuotsake (page header)**, johon sisältyy
 - sivutunniste** (page identifier)
 - yleensä sivun järjestysnumero tiedostossa, on tiedostokuvaajan perusteella muunnettavissa fyysiseksi levyosoitteeksi
 - ensimmäisen ja viimeisen tietueen osoite**
 - käyttötietoa**
- Sivuun sisältyy myös **tietuehakemisto**, josta saadaan kunkin tietueen osoite
- Lisäksi sivulla ovat **tietueet**

14



Sivun rakenne



15



Sivun rakenne

- Tietueen tunniste (record identifier, RID) muodostaa pari
 - sivutunniste
 - tietueen indeksi (tietuehakemistossa)
- Tuple identifier (TID) = record identifier

16



Tietueen rakenne

- Samassa tiedostossa olevat tietueet voivat olla keskenään
 - saman pituisia, eli kiinteäpituisia** (fixed length)
 - vaihtuvapituisia** (variable length)
 - erityyppisiä esim. kurssitietueita ja osallistujatietueita
 - saman tyyppisiä, mutta tietokenttien pituus tai määrä vaihtelee
- Tietueessa voi olla kiinteäpituisia tai vaihtuvapituisia kenttiä, yksikin vaihtuvapituinen kenttä tekee tietueesta vaihtuvapituisen

17



Tietueen rakenne

- Kiinteäpituisen kentän on jokaisessa tietueessa saman pituisen
 - smallint, integer, float, double, date ja timestamp tyyppiset arvot tallennetaan tyypillisesti (ei kuitenkaan välttämättä) kiinteäpituisiin kenttiin – kentän pituus riippuu tietotyyppistä. Esimerkiksi kokonaisluvut voitaisiin tallentaa binäärisinä 4 tavun pituisina kentteinä
 - SQL:n tietotyyppi char määrittelee kiinteäpituisen merkkijonon. Sillekin kiinteäpituisen kentän soveltuisi

18



Tietueen rakenne

- Vaihtuvapituisia kenttiä käytetään tyypillisesti tilanteissa, joissa tietoalkioiden arvot vaihtelevat merkittävästi pituudeltaan
 - Esimerkiksi vaihtuvapituiset merkkijonot
 - Yleensä vaihtuvapituisilla kentillä pyritään säästämään tilaa, mutta tilansäästöä voidaan saada muutenkin esimerkiksi tiivistämällä tietueet

19



Tietueen rakenne

- Relation employee(name, ssn, salary, address) monikon ('Smith, John', '010263-189F', 17000, 'Park Avenue ...') esittäminen:

- kiinteänmittaisilla kentillä:

20 tavua 11 tavua 4 tavua 30 tavua

Smith, John	010263-189F	17000	Park Avenue ...
-------------	-------------	-------	-----------------

20



Tietueen rakenne

- Vaihtuvanmittaiselle kentälle erilaisia esitystapoja:
- 2. **pituuskentän** avulla

010263-189	17000	11	Smith, John	26	Park Avenue ...
------------	-------	----	-------------	----	-----------------

- kiinteänmittaiset alussa kiinteissä paikoissa
- vaihtuvanmittaisen osan alkukohta samoin kenttien järjestys (vähintään) on kuvattava esim. sivun otsikkotietueessa

21



Tietueen rakenne

- 3. erotinmerkkien avulla

010263-189F # 17000 # Smith, John # Park Avenue ... #

Kenttien järjestys kiinnitetty

22



Tietueen rakenne

- Oraclen tietuerakenteessa kaikki kentät ovat periaatteessa vaihtuvapituisia/valinnaisia
 - Kentät esiintyvät tietueessa siinä järjestyksessä, missä ne on esitelty 'create table' -lauseessa (myöhemmät lisäykset tietueen loppuun).
 - Kunkin kentän alussa on **kentän järjestysnumero** ja **arvon pituus**
 - Jos kenttä on tyhjä (null), niin se puuttuu kokonaan tietueesta.

23