3. Tietokannan hakemistorakenteet

Tiedoston hakemisto (index) on tietorakenne, jonka tarkoituksena on nopeuttaa tiedon hakua tiedostosta.

- 'ylimääräinen' rakenne; perustiedostolla on oma organisointitapansa

Vrt. kirjan sivut, lopussa oleva asiasanahakemisto (t. henkilöhakemisto).

Esim. (binary search, 138, 591)
(buffering, 127-128)

Analogia vain osittainen: tiedostohakemisto perustuu usein (mutta ei aina) perustiedoston järjestykseen.

Hakemiston muodostus: indeksointi (indexing)

Tiedoston tietueiden haku voi perustua moneen saantipolkuun (access path):
- perustiedoston tiedostorakenteeseen
- hakemistoihin, jotka voivat olla perusavaimen tai muiden attribuuttien arvojen mukaisia
  --- erilaiset käsittelytarpeet ...

Hakemiston yleinen malli:

Olkoon F tiedosto, jonka tietueet ovat sivuilla P1, P2, … , PN.

Indeksointikenttä (indexing field) eli hakemistoavain (indexing key) K on yhden tai useamman kentän yhdistelmä, jonka arvoihin hakemisto perustuu.
- ei siis välttämättä relaation avain

Hakemisto muodostuu tietueista (v, p),

missä v on indeksointikentän arvo ja
p on osoitin tiedostoon F.

Osoitin p voi osoittaa (tarkasti) tietueen tai (karkeasti) sivun, jolta löytyy arvon v sisältäviä tietueita.

Hakemistoja on monenlaisia:
- yksitasoinen / monitasoinen
- tiheä / harva
- perushakemisto / järjestävä h. / oheishakemisto

Tiheä (dense) hakemisto: jokaiselle tietueelle on oma hakemistotietue eli parissa (v, p) p on tietueosoite. Tietueosoite tid ('tuple identifier') on muotoa (sivunumero, tietuenumero).

Harva (non-dense, sparse) hakemisto: hakemistotietue identifioi vain sivun eli p on sivunumero. Hakemistotietueita on siis vähemmän kuin perustietueita.

Hakemistotietue on yleensä perustietuetta lyhyempi; niitä mahtuu samankokoiseen levyjaksoon enemmän.

→ vaikka hakemisto on periaatteessa levytiedosto, se on käytännössä usein (ainakin osittain) puskurissa pitkäänkin

3.1 Yksitasoiset hakemistot

perustiedosto + (yksinkertainen) hakemistotiedosto

Esimerkki. Relaatio emp(ssn, name, salary)

harva hakemisto:

name	sivunro
Aalto	p1
Eerola	p2
Ketola	p3

tiheä hakemisto:

ssn	tid
1111	(p3,1)
1122	(p2,2)
3333	(p2,3)
4242	(p3,2)
5588	(p1,1)
7777	(p2,1)
9595	(p1,2)

perustiedosto:

	ssn	name	dno
p1:	5588	Aalto	2
	9595	Aaltonen	3
	. . . . .		
p2:	7777	Eerola	2
	1122	Frank	1
	3333	Hakala	4
	. . . . .		
p3:	1111	Ketola	1
	4242	Vainio	1
	. . . . .		

Harvan hakemiston tietue (v, p_i) osoittaa siis siihen sivuun, jonka ensimmäisen tietueen avainarvo on v. Tarkemmin:

1) v <= sivun p_i tietueiden avainarvot,
2) v > sivun p_{i-1} tietueiden avainarvot. (*)

Muut ominaisuudet:

- perustiedoston on oltava järjestetty hakemistoavaimen mukaan
- tiedostoon voi liittyä vain yksi harva hakemisto kerrallaan (yhden avaimen mukaan)
- periaatteessa harva hakemisto voisi olla vielä edellistä 'harvempi', jos perustiedoston sivut on ketjutettu
- hakemistotietueen avainarvo voi olla muukin ehdot (*) täyttävä avainarvo kuin tiedostossa esiintyvä (esimerkiksi ensimmäiselle sivulle "-∞")
- useimmat avainarvot eivät esiinny hakemistotietueissa

Tiheän hakemiston ominaisuudet:

- perustiedoston järjestys voi olla mikä tahansa

→ tiheä hakemisto tehdään tyypillisesti jonkin muun kuin relaation avainkentän mukaan (yksikäsitteisyys?)

- tiheitä hakemistoja voi olla useita samanaikaisesti
- hakemiston ja perustiedoston sama järjestys vaikuttaa joidenkin hakujen tehokkuuteen (avainarvojen osaväli)
- jokaisella avainarvolla on oma hakemistotietue

Hakemiston käyttö operaatioissa:

- tiedoston tietueen haku kaksivaiheinen:
  1) etsi hakemistotietue ja siitä sivun osoite
  2) hae perustiedoston sivu

  - jos hakemistosivu on puskurissa, yksi levyhaku
  - hakemistojaksoja vähemmän kuin perustiedoston jaksoja → käsittely perustiedostoa nopeampaa
  - hakemisto järjestyksessä → binäärihaku käy

- tietueen lisäys tiedostoon:
  - lisätään tietue perustiedostoon
  - päivitetään hakemisto(t) uuden tilanteen mukaisiksi

  Esim. insert into emp values ('6666', 'Eskola', 2):
  - monikko sivulle p2
  - tietue ('6666', (p2,4)) tiheään hakemistoon
  (harvaa hakemistoa ei ole tarpeen päivittää)

- tietueen poisto: poisto perustiedostosta ja tiheästä hakemistosta ('oikeasti' tai poistomerkintä)

- tietueen päivitys: harj.teht.

Yleisesti: mitä enemmän hakemistoja on, sitä raskaampia ovat perustiedoston muutoksia sisältävät operaatiot.
(vrt. oheishakemistot)

Hakemistotyyppejä/ -nimityksiä:

1. Perushakemisto eli ensisijainen hakemisto (primary index)

- tiedoston perusavaimeen (primary key) perustuva (harva tai tiheä) hakemisto
- perustiedosto avainjärjestyksessä
  Esim. name-hakemisto edellä, jos name avain

- hakemisto on perustiedostoa pienempi, voi jopa mahtua keskusmuistiin

  - tietueita vähemmän (harva)
  - tietueet lyhyempiä (harva tai tiheä)

2. Ryvästävä eli järjestävä hakemisto (clustering index)

- lähekkäiset avainarvot ovat perustiedostossa lähekkäin, yleensä samassa jaksossa; muodostavat 'rypään' (cluster) (käytännössä järjestetty tdsto)
- perustana oleva kenttä = ryvästävä; arvojen ei tarvitse olla yksikäsitteisiä (voi toistua, jolloin kaikki tietyllä arvolla varustetut perustietueet ovat peräkkäin)
- voi olla harva tai tiheä

Esim. osastoittain perustiedossa olevat employee-monikot ; ryvästävä kenttä dno:

dno	sivunumero	perustiedosto:
1	p1	p1: ... , 1; ..., 1; ... , 1; ... , 2
2	p1	p2: ... , 2; ..., 2; ... , 2; ... , 2
3	p3	p3: ... , 2; ..., 2; ... , 3, ... , 5
5	p3	p4: ... , 5 ; . . . .
...	...	

- monia toteutustapoja (esim. jaksoon vain yhdellä avainarvolla varustettuja tietueita)
- erilaisia määritelmiä (onko ryvästävä kenttä (yli)avain vai ei)

3. Oheishakemisto

- secondary index, toisiohakemisto; muodostaa toissijaisen saantipolun

- ei ole perushakemisto eli ei voi nojautua perustiedoston järjestykseen
  →hakemisto on välttämättä tiheä

Jos oheishakemiston hakemistoavain ei ole yksikäsitteinen, on eri vaihtoehtoja:

1) useita hakemistotietueita samalle avainarvolle

Esim. perustiedosto, s. 4

oheishakemisto:

dno	tid
1	p2,2
1	p3,1
1	p3,2
2	p1,1
...	...

2) hakemistotietue sisältää listan osoitteita (tietue vaihtuvanmittainen)

3) kiinteänmittaiset hakemistotietueet, vastaaville tietue-osoitteille oma hakemisto ('välitaso'):

hakemisto			välitaso		perustiedosto				
dno	bid	b1:	(p2,2)		p1:	...	...	2	(dno)
1	b1		(p3,1)			...	...	3	
2	b2		(p3,2)						
3	b3	b2:	(p1,1)		p2:	...	...	2	
4	b4		(p2,1)			...	...	1	
...	...					...	...	4	
		b3:	(p1,2)						
					p3:	...	...	1	
		b4:	(p2,3)			...	...	1	
						...	...		

Oheishakemisto muodostaa siis tavallaan vaihtoehtoisia loogisia järjestyksiä tietueille. Hakemiston järjestystä voidaan käyttää hyväksi (esim. binäärihaku); perustiedoston käsittely ei ole 'fyysisesti' peräkkäistä (eli vie aikaa).

3.2 Monitasoiset hakemistot

Hakemisto on tiedosto, jonka rakenne voi olla periaatteessa mikä tahansa. Käsittelyn kannalta sopiva rakenne on usein järjestetty peräkkäistiedosto.

Oikein pienelle hakemistolle riittää vaikka kasarakenne …

Kun hakemisto on suuri, haku peräkkäistiedostosta voi olla liian hidasta.

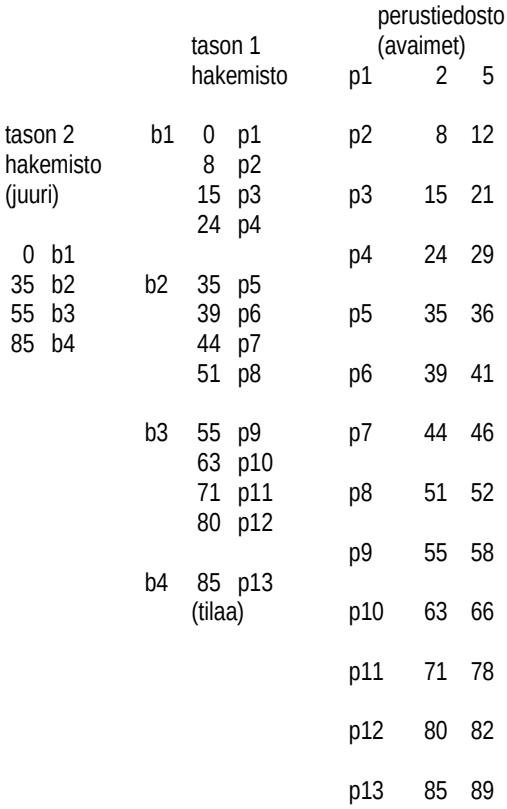
Järjestetylle hakemistotiedostolle voidaan muodostaa oma (järjestetty) hakemisto; jos tämäkin on liian 'pitkä', sille edelleen oma (järjestetty) hakemisto jne. Näin saadaan monitasoinen hakemistorakenne, jota sanotaan ISAM-rakenteeksi (Indexed Sequential Access Method).

Ominaisuuksia:

- Hakemisto on yleensä harva (joten seuraavan tason hakemisto on paljon edellistä pienempi).
- Uusia tasoja rakennetaan, kunnes koko uusi hakemisto mahtuu yhdelle sivulle.

  Käytännössä tasoja ei synny monta, ja ylimmän tason hakemisto(sivu) pysyy yleensä puskurissa.

- Hakemistoja ei päivitetä luonnin jälkeen; hakemistosivuja ei siis tarvitse kirjoittaa takaisin levylle.



· Kun hakemistosivuja ei päivitetä, hakemiston avaimet voivat (tietuepoistojen jälkeen) kokonaan puuttua perustiedostosta. Hakemisto toimii silti; määritellään:

n-alkioisen hakemistosivun P tietue (vi,pi) peittää avainarvon v, jos

(1) i < n ja vi <= v < v(i+1), tai
(2) i = n ja vi <= v.

Ts. hakemistossa olevat 'avainarvot' voivat olla vain 'viittoja', jotka ratkaisevat, mikä hakemistoalkio kuuluu millekin seuraavan tason jaksolle.

Avainarvolla v varustetun tietueen haku l-tasoisesta ISAM-rakenteesta:

```
P ← rakenteen juurisivun osoite;
for j ← l downto 1 do
      B:=bufferfix(P);
      etsi puskurisivulta B avainarvon v peittävä
            hakemistotietue (vi, pi);
      P←pi;
      bufferunfix(B);
end;
B:=bufferfix(P);
hae avainarvolla v varustettua tietuetta puskurisivulta B;
bufferunfix(B);
end.
```


Algoritmi vaatii enintään l+1 levyhakua (vähemmän, jos ainakin juuri on jo puskurissa).

Tietueen lisäys: haetaan tietuetta edellisen algoritmin mukaan; lisätään sitten siihen jaksoon, johon avaimensa perusteella kuuluisi. (Jos avainrajoite ja löytyy, ei lisätä …)

Tasojen lukumäärä?
· määräytyy hakemistotason tiedoston jaksotuskertoimen (eli hakemistojakson koon ja hakemistotietueen pituuden) mukaan

Esim. tiedostossa N = 50000 sivua;
avainkentän pituus 12 tavua, sivunumero 4 tavua
ja sivun koko 4 KB.

- hakemistotietue vie enintään 12 + 4 + ? = 20 tavua
- hakemistotietueita mahtuu sivulle 4000/20 = 200 kpl
- tason 1 hakemistosivuja tarvitaan 50000/200 = 250 kpl
- tason 2 hakemistotietueet (250 kpl) mahtuvat kahdelle sivulle
- tasolla 3 kaksi sivuosoitetta mahtuu (oikein hyvin …) yhdelle sivulle (joka on hakemiston juuri)

Yleisesti: tasojen lkm = $\lceil \log_f N \rceil$, missä N = tason 1 tietueiden määrä ja f hakemistosivujen jaksotuskerroin. Kun yleensä f >>2, hierarkiasta tulee aina hyvin matala.

Huom.
Tiedoston kasvuun voidaan varautua täyttämällä perustiedoston jaksot alussa väljästi, jopa vain puoliksi (täyttöaste 50 %).

Hakemistoon voi helposti tulla aika paljon loogisesti tyhjää (kun perustason osoitteiden määrä pakottaa 'juuri ja juuri' ottamaan uuden tason käyttöön'). Tyhjille jaksoille ei kuitenkaan varata tilaa. Toisaalta tiedoston tuleva kasvu on usein tuntematon.

Tilan loppuessa perustiedoston jaksosta siihen voidaan ketjuttaa ylivuotojaksoja; tämä ei näy hakemiston puolella, mutta kasvattaa tietysti käsittelyaikaa yhdellä tai useammalla levyhaulla perustiedostossa.
Koska hakemistojaksoja ei muuteta, ne täytetään kokonaan.

Hakemistotasojen määrää voidaan kontrolloida luontivaiheessa mahdollisesti valitsemalla jakson koko sopivasti.

Erikoistapaus: Rakenne oli aikanaan tarkoitettu kaksitasoiseksi levymuistin rakenteen mukaisesti: ylin taso = sylinterihakemisto, alempi taso = urahakemisto (IBM ISAM …)

ISAM-rakenteen luonnin vaiheet:

1. Järjestetään alkuperäisen tiedoston perustietueet avaimen mukaan nousevaan järjestykseen.

2. Sijoitetaan perustietueet avaimen mukaiseen järjestykseen perustiedostolle varatuille sivuille P1, … , PN. Jätetään kasvuvaraa esim. 50 % jokaiselle sivulle.

3. Muodostetaan 1. tason hakemistotietueet (v1, P1, … , (vN, PN) ottamalla vi:n arvoksi pienin sivun Pi avainarvo. Poikkeus: ensimmäiseen tietueeseen kannattaa asettaa vi = pienin mahdollinen arvo (“-∞”), jotta tätäkään hakemistotietuetta ei tarvitse koskaan päivittää.

Sijoitetaan hakemistotietueet sivuille avaimen mukaisessa järjestyksessä (sivut täyteen).

4. Muodostetaan ylemmät tasot vastaavasti.

Huom.
Jos tiedosto luotaisiin suorittamalla yksittäisiä lisäyksiä (‘insert …’) tyhjään tiedostoon, rakenne joutuisi hyvin nopeasti epätasapainoon. (Ensimmäiset hakemistotietueet tulisi luoda aika keinotekoisesti – tai kaikki lisätyt tietueet joutuisivat aluksi ylivuotoketjuun.)

Hakemistoa jouduttaisiin myös ajoittain päivittämään (eli lisäysoperaatioiden nopeus vaihtelisi paljon).

Epätasapaino on joka tapauksessa ajan mittaan mahdollinen: rakenteen ‘korkeudeksi’ tulee tavallaan tasojen lukumäärä + pisimmän ylivuotoketjun pituus.
Kun näin syntyvä levyhakujen määrä tulee liian suureksi, rakenne on organisoitava uudelleen.

Uudelleenorganisointi (luonti uudelleen) on periaatteessa suoraviivainen, mutta työläs operaatio.

- luetaan vanhasta tiedostosta perustietueet hakemiston avulla järjestyksessä
- kirjoitetaan tietueet uuteen tiedostoon ja muodostetaan uusi hakemisto

Sopivan täyttöasteen valinta:
- tilansäästö
- seuraavan uudelleenorganisoinnin tarve

ISAM-rakenne kasan päälle rakennettuna hakemistona?

- Perustiedosto voi olla kasa, jos välittömästi sen ‘päälle’ rakennetaan järjestetty tiheä hakemisto (joka voi olla esimerkiksi ISAM-rakenteen alin taso).
Hakemiston tietue = (avainarvo, jakson tai tietueen osoite).

Tietueen hakua ei tarvitse aina ulottaa kasaan asti: tietueen olemassaolo näkyy jo tiheästä hakemistosta (esim. avainrajoitteen kontrolli).
Jos tämän hakemiston tietueet ovat merkittävästi perustietueita lyhyempiä, jo alimmalla tasolla tarvitaan paljon vähemmän jaksoja kuin kasassa.

Mihin ISAM-rakenne sopii?

- yleisesti kyselyihin, jotka perustuvat hakemistoavaimen tarkkaan arvoon tai sen alkuosaan:

```
select * from R where A = ‘abc’;
select * from R where A like ‘abc%’;
```

- hakemistoavain moniosainen, esimerkiksi suku- ja etunimi (lname, fname)

Tässä tehokkaita kyselyjä (ehtoja):

```
select * from employee
    where fname = ‘John’ and lname =’Smith’;
    where lname = ‘Smith’
    where lname like ‘Sm%’
    where fname < ‘J’ and lname = ‘Smith’
```

Tehottomia (hakemisto ei auta):

```
    where fname = ‘John’ and lname like ‘%son’
    where fname = ‘John’ and ssn = ‘123456789’
```