Poglavje 5

Organizacije datotek in indeksiranje

Povzeto po [1]

Uvod..

- Potrebno je določiti **stroškovni model**, ki opredeljuje stroške dostopanja do podatkov
- Analizirali bomo tri osnovne datotečne organizacije:
  - neurejeno datoteko,
  - urejeno datoteko,
  - razpršeno datoteko.

Uvod

- Namen analize je pravilna izbira datotečne organizacije, glede na potrebe.
- Rešitev za hitrejši dostop do zapisov je indeksiranje

Stroškovni model..

- Stroškovni model omogoča oceniti stroške (v smislu izvajalnega časa) I/O operacij nad stranmi v PB
- Za analizo bomo uporabljali naslednje parametre:
  - **B**: število podatkovnih strani,
  - **R**: število zapisov na stran,
  - **D**: (povprečen) čas, potreben za branje ali pisanje strani,
  - **C**: čas za procesiranje zapisa,
  - **H**: čas, ki je potreben za uporabo hash funkcije nad zapisom (pri razpršenih datotekah).
- Tipične vrednosti znašajo: D=15 ms, C in H=100 ns

Stroškovni model..

- Poenostavitve:
  - Ker so I/O operacije časovno obsežnejše, stroške procesiranja lahko zanemarimo (C = H = 0),
  - natančen model bi bil preveč zapleten, zato se zaradi lažjega razumevanja osredotočimo le na število strani, ki se jih prebere ali zapiše na disk in nam služijo kot število I/O operacij.
- Kot stroškovno metriko torej uporabljamo število I/O operacij, ki prenašajo (branje in pisanje) strani

Stroškovni model

- Poenostavitve lahko uporabimo, saj se izkaže, da predstavljeni model kljub poenostavitvi odraža dejanske trende pri uporabi različnih datotečnih organizacij

Primerjava datotečnih organizacij..

- Primerjamo stroške preprostih operacij za tri vrste datotek:
  - neurejeno datoteko (**heap**): zapisi si vrstijo naključno,
  - urejeno datoteko (**sorted**): vrstni red zapisov je določen glede na *iskalni ključ*),
  - razpršeno datoteko (**hashed**): zapisi so grupirani v skupine na podlagi razpršilne funkcije (hash function).
- Cilj je poudariti, kako pomembna je pravilna izbira datotečne organizacije

Primerjava datotečnih organizacij..

- Operacije:
  - **skeniranje**: preberi vse zapise v datoteki. Vse strani iz datoteke je potrebno prenesti v Buffer pool
  - **iskanje s pogojem ekvivalence**: poišči vse zapise, ki ustrezajo določenemu pogoju. Strani, ki vsebujejo ustrezne zapise je potrebno prenesti z diska in na teh straneh je potrebno locirati želene zapise

Primerjava datotečnih organizacij..

- Operacije:
  - **iskanje z definiranjem območja**: poišče vse zapise iz določenega območja. Ustrezne strani je potrebno presesti z diska
  - **vstavljanje**: Vstavi dan zapis v datoteko. Potrebno je identificirati stran datoteke, v katero se bo zapis vstavil, prebrati to stran z diska, jo ažurirati in zapisati nazaj na disk. Od datotečne organizacije je odvisno ali bo potrebno poleg želene strani, z diska brati in nanj pisati še dodatne strani
  - **brisanje**: Brisanje zapisa s podanim rid (record id). Potrebno je identificirati stran z danim zapisom, jo prebrati z diska, jo ažurirati in zapisati nazaj na disk

Primerjava datotečnih organizacij..

- Ko zanemarimo v izračunih vrednosti za C in H, potem dobimo naslednjo primerjalno tabelo z ocenami za posamezno datotečno organizacijo

Primerjava datotečnih organizacij..

- Značilnosti:
  - **neurejena datoteka**: hitro skeniranje, vstavljanje in brisanje; počasno iskanje
  - **urejena datoteka**: omogoča hitro iskanje območja; srednja hitrost pri iskanju; počasno vstavljanje in brisanje
  - **razpršena datoteka**: hitro vstavljanje in brisanje zapisov; zelo hitro je iskanje na osnovi enakosti; nobene podpore izboru območja; skeniranje celotne datoteke je nekoliko počasnejše

File Type	*Scan*	*Equality Search*	*Range Search*	*Insert*	*Delete*
Heap	BD	$0.5BD$	BD	$2D$	$Search + D$
Sorted	BD	$Dlog_2B$	$Dlog_2B$+# *matches*	$Search + BD$	$Search + BD$
Hashed	$1.25BD$	D	$1.25BD$	$2D$	$Search + D$

Figure 8.1 A Comparison of I/O Costs

Primerjava datotečnih organizacij

- Zaključek:
  - Nobena organizacija ni dobra za vse operacije.
  - Neurejena datoteka je zelo primerna za skeniranje celotne datoteke
  - Razpršena datoteka je najprimernejša pri uporabi iskanja s pogojem ekvivalence
  - Urejena datoteka je najprimernejša pri iskanju intervalov

Indeksi – uvod..

- Včasih hočemo najti vse zapise, ki imajo določeno vrednost v določenem polju
- Organizacija neurejene datoteke nam ne pomaga pri iskanju zapisov
- INDEKS je zunanja podatkovna struktura, ki pomaga najti rid-je zapisov, ki ustrezajo določenemu pogoju
- INDEKS pospeši iskanje

Indeksi – uvod..

- Vsak indeks vsebuje iskalni ključ, ki je sestavljen iz enega ali več polj zapisov, ki se nahajajo v datoteki. Iskalni ključ je lahko katerikoli podmnožica polj zapisov v datoteki
- Datoteki, ki jo indeksiramo, pravimo pogosto indeksirana datoteka

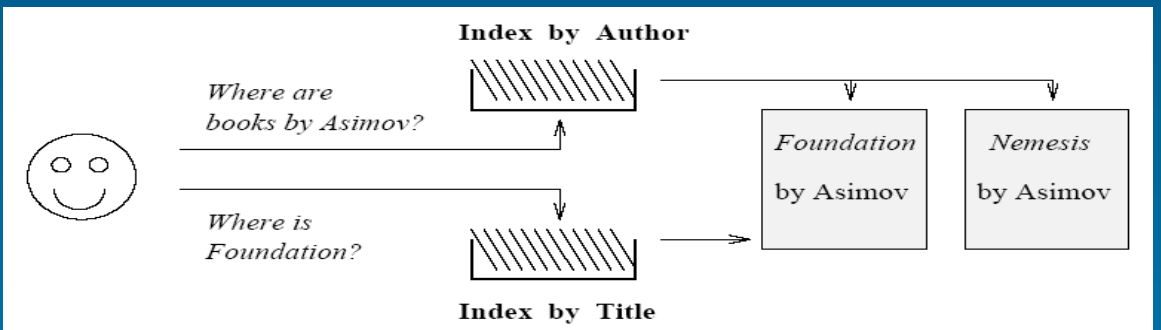

Figure 7.6 Indexes in a Library

Indexi – uvod..

- Indeks je datoteka, katere vsebina omogoča hiter dostop do zapisov v osnovni datoteki, ki je z indeksom indeksirana
- Pod hitrim dostopom pojmujemo dostop, ki je hitrejši, kot pa ga sicer omogoča organizacija osnovne datoteke
- Indeks si lahko predstavljamo kot zbirko vpisov *k**, kjer je *k* ključ indeksa, * pa kazalec(ci) na zapise, ki imajo indeksna polja enake ključu

Indexi – uvod

- Pojavita se 2 vprašanji:
  - Kako so podatkovni vpisi v indeksu organizirani, da bodo podpirali hitro in učinkovito iskanje zapisov?
    - Ena rešitev za organizacijo indeksa je uporaba razpršene datoteke (prejšnja slika), ki omogoča hitro iskanje zapisov.
    - Obstaja tudi več drugih indeksnih struktur, ki omogočajo hitro iskanje in bodo predstavljene v nadaljevanju.
  - Kaj natančno je shranjeno v podatkovnem vpisu (Podatkovni vpis *k** nam omogoča dobiti en ali več zapisov z vrednostjo ključa *k*)
    - k* je dejanski zapis (z iskalnim ključem k),
    - podatkovni vpis je par <k, rid>, kjer rid predstavlja *record id* zapisa z vrednostjo indeksnega ključa,
    - podatkovni vpis je par <k, rid-list>, kjer je rid-list seznam *record id-jev* tistih zapisov, ki imajo v poljih vrednost enako indeksnemu ključu.

Vrste indeksov..

- **Gosti indeks**: na vsak zapis osnovne datoteke, kaže kazalec iz indeksa
- **Redki indeks**: kazalci iz indeksa kažejo na skupine zapisov osnovne datoteke
- **Primarni indeks**: indeksiranje osnovne datoteke je izvedeno po njenem ključu (ključ indeksa in ključ osnovne datoteke sta enaka)

Vrste indeksov..

- **Sekundarni indeks**: indeksiranje osnovne datoteke je izvedeno po podatkovnem elementu (polju), ki nastopa v zapisu osnoven datoteke, vendar pa ni ključ.
- **Enonivojsko indeksiranje**: indeksirana je osnovna datoteka – v indeksu poiščemo kazalec na polje ali skupino polj v osnovni datoteki, kjer nato poiščemo iskani zapis.
- **Večnivojsko indeksiranje**: indeksirana je osnovna datoteka, indeksiran je indeks na osnovno datoteko, indeksiran je indeks na indeks itd.

Vrste indeksov

- **Statično indeksiranje**: indeks ostaja nespremenjen, čeprav se osnovni datoteki zapisi dodajajo ali iz nje brišejo; ko postane iskanje neučinkovito, se izvede reorganizacija indeksa.
- **Dinamično indeksiranje**: indeks se ob dodajanju in brisanju zapisov osnovne datoteke prilagaja vsebini, tako da so iskalne poti do vseh zapisov v osnovni datoteki enako dolge; indeksa ni potrebno nikoli reorganizirati.

Gosti indeks vs. redki indeks..

- Osnovna datoteka ima strani s po tremi zapisi
- Zapis ima tri polja (*name, age, sal*)
- Redki indeks indeksira osnovno datoteko po polju *name*
- Gosti indeks pa indeksira osnovno datoteko po polju *age*

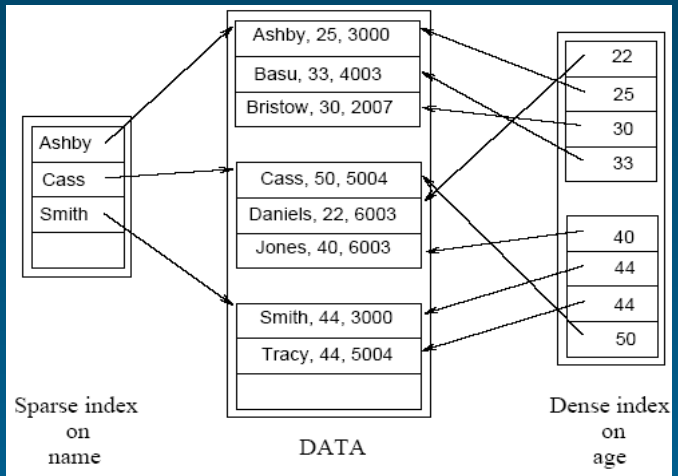

Figure 8.5 Sparse versus Dense Indexes

Gosti indeks vs. redki indeks

- Redki indeks kaže na skupine zapisov
- Gosti indeks indeksira vsak zapis v datoteki
- Redki indeks je običajno mnogo manjši kot gosti indeks
- Pravimo, da je datoteka **invertirana po polju**, če obstaja obstaja gosti sekundarni indeks nad tem poljem
- Pravimo, da je datoteka **popolnoma invertirana**, če obstaja gosti sekundarni indeks po vseh poljih, ki ne sestavljajo ključa

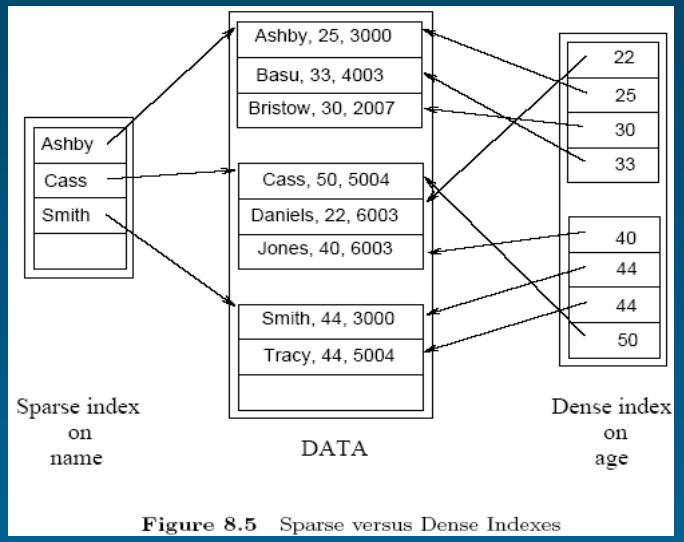

Figure 8.5 Sparse versus Dense Indexes

Primarni in sekundarni indeks..

- Indeks, ki je kreiran na množici polj, ki vsebuje tudi primarni ključ, se imenuje **primarni indeks**
- Indeks, ki ni primarni indeks je **sekundarni indeks**. Sekundarni indeks je definiran nad množico polj, ki niso sestavni del ključa
- Za dva podatkovna vpisa pravimo, da sta **duplikata**, če vsebujeta enaki vrednosti za iskalni ključ, ki je povezan z indeksom (kazalcem)

Primarni in sekundarni indeks

- Primarni indeks ne vsebuje duplikatov
- Sekundarni indeks lahko vsebuje duplikate
- Če vemo, da duplikati ne obstajajo (indeks vsebuje ključ), potem pravimo temu indeksu **unique indeks**

Indeksi s kompozitnimi iskalnimi ključi

- Indeksni ključ lahko vsebuje več polj. Takemu ključu pravimo **kompozitni iskalni ključ**
- Slika prikazuje razliko med kompozitnima iskalnima ključema <age,sal> in <sal,age>
- Če imamo indeks z indeksnim ključem, lahko izvajamo poizvedbe, pri katerih iščemo zapise z vrednostmi polj, ki so enake ključu (to imenujemo **equality query**)
- Druga vrsta poizvedbe pa nam omogoča, da je samo ena vrednost v ključu konstantna, druge pa ne. Tako lahko pri poizvedbi dobimo več zapisov (**range query**)

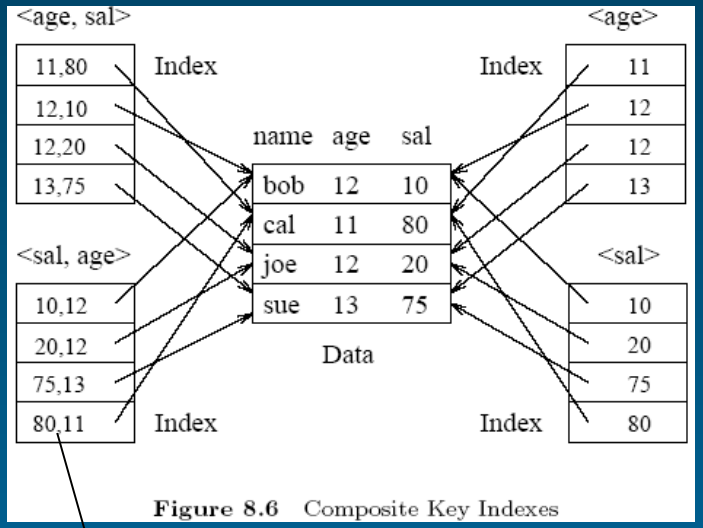

Figure 8.6 Composite Key Indexes

kompozitni indeksni ključ

Specifikacija indeksa v SQL 92

- Standard SQL 92 ne opredeljuje nobenega stavka za kreiranje ali brisanje indeksov. Še več, standard sploh ne zahteva implementacije indeksov
- V praksi je drugače; vsi komercialni SUPB podpirajo eno več vrst indeksiranja
- Primer SQL stavka za kreiranje B+ drevesnega indexa:

```
CREATE INDEX IndAgeRating ON Students
       WITH  STRUCTURE = BTREE,
             KEY = (age, gpa)
```