Fyzická organizácia dát

Ján Mazák

FMFI UK Bratislava

Disky

Klasický pevný disk (HDD)

- dáta sa spracúvajú po stránkach / blokoch (0.5–4kB)
  (pojmy *page* a *block* budeme voľne zamieňať,
  v literatúre však môžu označovať rôzne veci)
- seek time (pohyb hlavy) 2-3 ms
- rotational delay (otáčanie platní) 0-4 ms
- prenos dát 0.25 ms / 64 kB page

Niektoré kroky zahŕňajú fyzický pohyb súčiastok, nevieme zrýchliť.

Disky

Flash storage / Solid state drive (SSD)

- rýchle pri náhodnom prístupe, netreba fyzický pohyb hardvéru
- granulárne čítanie (napr. 8 kB)
- zápis väčších blokov (napr. 512 kB)
- wear leveling — jedno miesto vydrží len niekoľko tisíc zápisov

Výroba SSD 2-3x drahšia ako HDD (z obchodného hľadiska obrovský rozdiel). Ale spotrebujú menej elektriny.

Disky

Triky na urýchlenie:

- niektoré operácie možno urýchliť použitím dočasnej vyrovnávacej pamäte (buffering, caching).
- uložiť často čítané bloky do cache
- načítať vopred bloky, ktoré pravdepodobne bude treba v blízkej budúcnosti
- zbierať dáta do vyrovnávacej pamäte (buffer) a zapísať naraz väčšie množstvo

Toto funguje aj mimo DBMS, ale pri operáciách veľkého rozsahu sa oplatí špecificky optimalizovať
(napr. AWS Caching Solutions).

Disky

Magnetické pásky

- využívajú sa najmä na lacné zálohovanie
- len pomalý sekvenčný prístup
- 15 TB / ks, vyvíjajú sa 1000 TB
- `https://spectrum.ieee.org/why-the-future-of-data-storage-is-still-magnetic-tape`

Logická organizácia dát

- hodnoty jednotlivých atribútov sú zoskupené do záznamov
- záznamy tvoria reláciu
- záznamy môžu mať fixnú alebo variabilnú dĺžku

Ako toto fyzicky reprezentovať na disku?

Fyzická organizácia dát

Neusporiadané záznamy v spájanom zozname:

- v každom diskovom bloku niekoľko záznamov a pointer na ďalší blok
- sekvenčné vyhľadávanie, trvá $O(n)$, čiže dlho
- vkladanie sa dá urýchliť, ak osobitne evidujeme plné bloky a bloky s práznym miestom, v ideálnom prípade je to $O(1)$
- mazanie záznamov môže spôsobiť fragmentáciu a predĺžiť vyhľadávanie neúmerne množstvu uložených dát

Fyzická organizácia dát

Usporiadané záznamy v spájanom zozname:

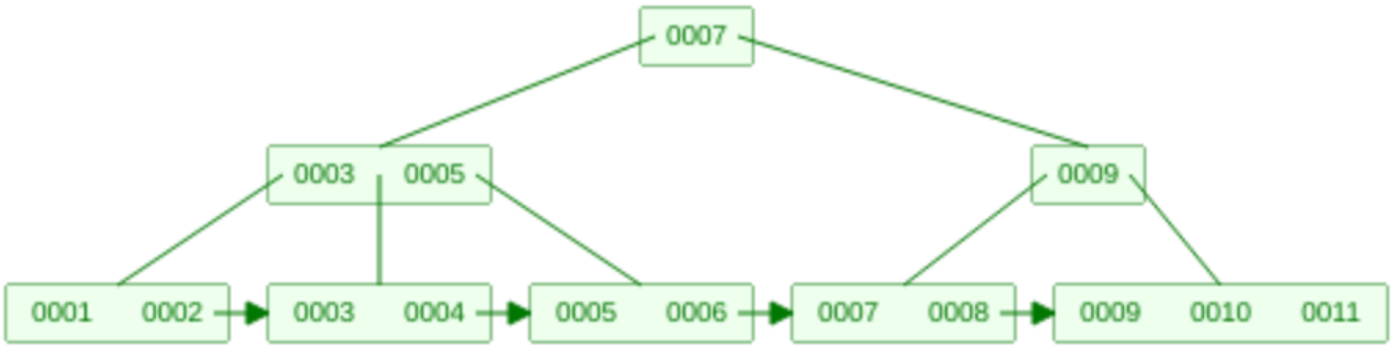
- záznamy usporiadané podľa nejakého kľúča, v každom diskovom bloku niekoľko záznamov a pointer na ďalší blok
- binárne vyhľadávanie $O(\log n)$
- vkladanie v najhoršom prípade $O(n)$, lebo treba spraviť miesto na nový záznam odsunutím existujúcich
- ak evidujeme osobitne usporiadané kľúče s odkazmi na blok, ktorý záznamy s danou hodnotou kľúča obsahuje, zrýchlime vyhľadávanie — indexovanie záznamov

B+ strom

B+ strom: vyvážený strom vhodný pre uloženie na disku.

- záznamy (resp. odkazy na ne) sú uložené v listoch
- parametrizovaný hodnotou m tak, aby stupeň vetvenia bol medzi $m/2$ a m
- voľba m podľa veľkosti bloku (rádovo desiatky)
- malá výška stromu, zhruba $\log_m(n)$
- minimalizujeme počet blokov, čo treba načítať pri prístupe k listu
- vyhľadávanie, vkladanie, mazanie $O(\log_m(n))$

B+ strom



Vyskúšajte si:

`https://www.cs.usfca.edu/~galles/visualization/BPlusTree.html`

Index

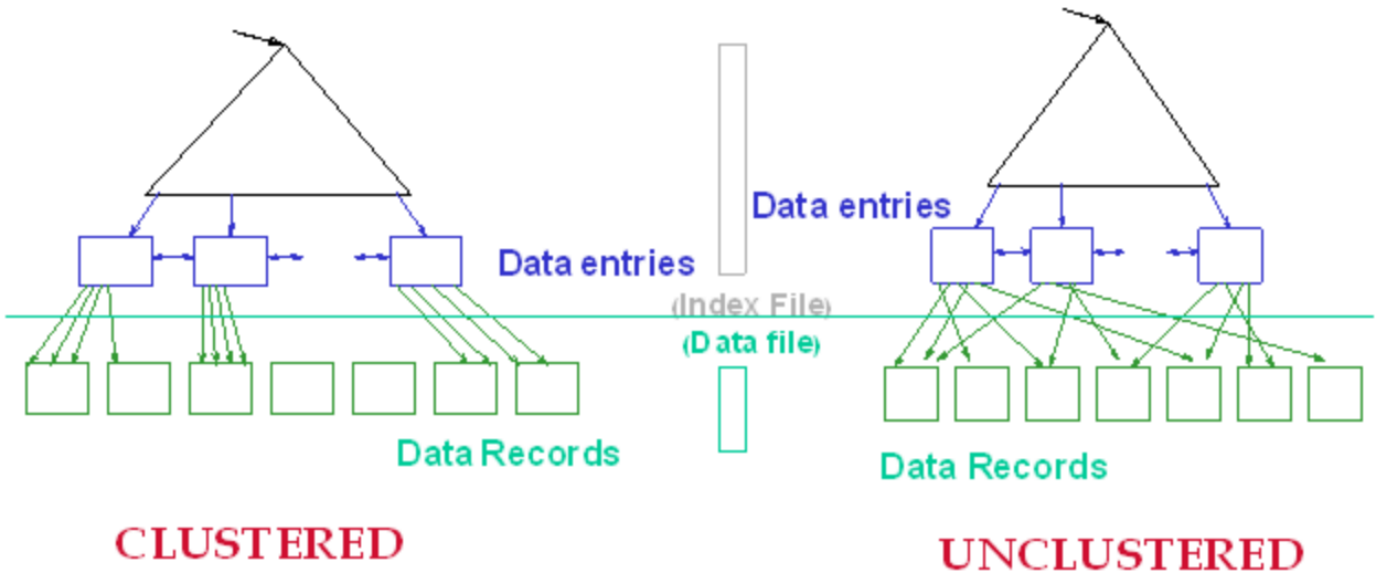
B+ strom by mohol obsahovať hodnoty priamo v nelistových uzloch (B strom). Neefektívne: pri traverzovaní stromu zbytočne čítame dáta, ktoré nás nezaujímajú.

Ak sú v listoch len referencie, akú štruktúru má *heap file*, v ktorom sú uložené samotné záznamy?

- usporiadané záznamy: clustered index
- neusporiadané záznamy: unclustered index

Index

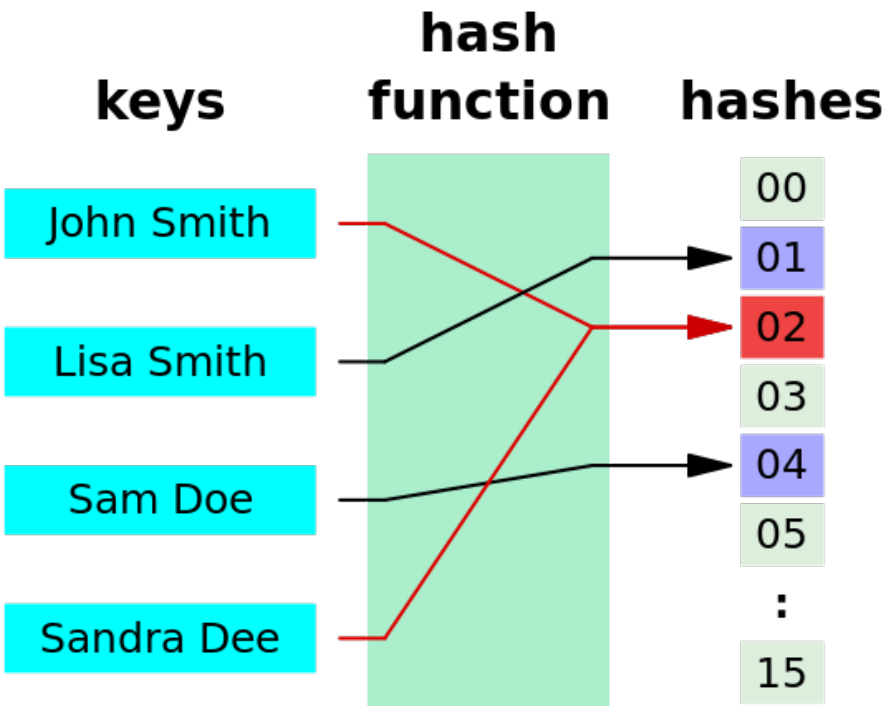
Výhody clustered indexu: rýchlejší prístup k okolitým záznamom, napr. ak hľadáme dáta z daného intervalu. Údržba indexu (odstraňovanie „unclusterizácie“) však niečo stojí.



https://courses.cs.washington.edu/courses/cse544/99sp/lectures/storage/sld025.htm

Hashovanie

Urýchlenie vyhľadávania: rozdelíme záznamy do skupín podľa hodnoty vypočítanej hashovacou funkciou.



Červeným je označená *kolízia*.

Hashovanie

„Dobrá“ hashovacia funkcia:

- rozdeľuje rovnomerne
- využíva celý obor hodnôt
- dá sa počítať veľmi rýchlo

Kolízie nám veľmi neprekážajú, použijeme spájaný zoznam.

(Rozdiel oproti tzv. kryptografickým hashovacím funkciám, kde obtiažnosť algoritmického hľadania kolízií je kľúčová.)

Hashovanie

Hashovanie možno použiť na vytvorenie indexu.

Pri dobrej hashovacej funkcii možno očakávať vyhľadávanie v konštantnom čase, ak index nie je *preplnený* — ak je záznamov priveľa, v každom hashovacom buckete je pridlhý spájaný zoznam. Vtedy možno buckety rozdeliť použitím dodatočnej hashovacej funkcie. Existujú tiež spôsoby hashovania, ktoré priebežne pridávajú buckety.

Nevýhoda: len podmienka s rovnosťou, nie nerovnosť či intervaly.

Hashovanie

Hashovanie možno použiť na horizontálne škálovanie.

Jednotlivé hash buckety môžeme skladovať na osobitných serveroch. Vyhľadávanie podľa kľúča aj join s podmienkou na rovnosť tak možno naďalej počítať efektívne.

Pri odobraní alebo pridaní servera však máme problém. Riešenie: *consistent hashing*. (Po kliknutí na link sa dozviete viac, neočakáva sa však, že to budete vedieť v rámci tohto predmetu.)

Vytvorenie indexu

Hash index na vyhľadávanie podľa empno:

```
CREATE INDEX ON employee USING HASH (empno);
```

Index na zabezpečenie unikátnosti mena (B+ strom):

```
CREATE UNIQUE INDEX i1 ON employee (name);
```

Index pre vypočítanú hodnotu (nie priamo atribút relácie):

```
CREATE INDEX ON employee (lower(name));
```

Index obsahujúci dodatočné dáta (podľa salary nemožno vyhľadávať, iba podľa empno a name):

```
CREATE INDEX ON employee (empno, name) INCLUDE (salary);
```

Druhy indexov

Druhy indexov:

- btree (default)
- hash
- gist (geografické dáta o polohe)
- bloom filter („user-installed“ in PostgreSQL)
- k-dimensional tree, ...

Indexy zrýchľujú vyhľadávanie, ale spomaľujú vkladanie.
Pri vkladaní veľkého množstva dát sa oplatí index zrušiť a potom nanovo vybudovať.

Literatúra

Disky

- https://cs186berkeley.net/notes/note3/
- https://www.db-book.com/slides-dir/PDF-dir/ch12.pdf
- https://www.db-book.com/slides-dir/PDF-dir/ch13.pdf

B+ stromy a indexy

- https://cs186berkeley.net/notes/note4/
- https://www.db-book.com/slides-dir/PDF-dir/ch14.pdf