Databázové systémy

Tomáš Skopal

– fyzická implementace
relačních databází

Osnova

- správa disku, stránkování, buffer manager
- organizace databázových souborů
- indexování
  - jednoatributové indexy
    - B^+-strom, bitové mapy, hašování
  - víceatributové indexy

Úvod

- relace/tabulky uloženy v souboru(rech) na disku
- potřeba organizace záznamů uvnitř souboru pro jejich efektivní uložení, modifikaci a přístup k nim

Příklad:
Zaměstnanec(jmeno char(20), vek integer, mzda integer)

Stránkování

- záznamy fyzicky organizovány ve stránkách pevné velikosti (blocích o několika kB) na disku
- důvod je HW, disk obsahuje rotační plotny a čtecí hlavy, data je potřeba přizpůsobit tomuto mechanismu
- HW je schopen přistupovat k celým stránkám (I/O operace – čtení, zápis)
- čas pro I/O operaci =
  = seek time + rotational delay + data transfer time
- sekvenční přístup ke stránkám je mnohonásobně rychlejší než náhodný přístup, odpadá seek time a rotational delay

  Příklad: načtení 4 KB může trvat typicky 8 + 4 + 0,5 ms = 12,5 ms; tj. samotné čtení trvá pouhých 0,5 ms = 4% celkového času!!!

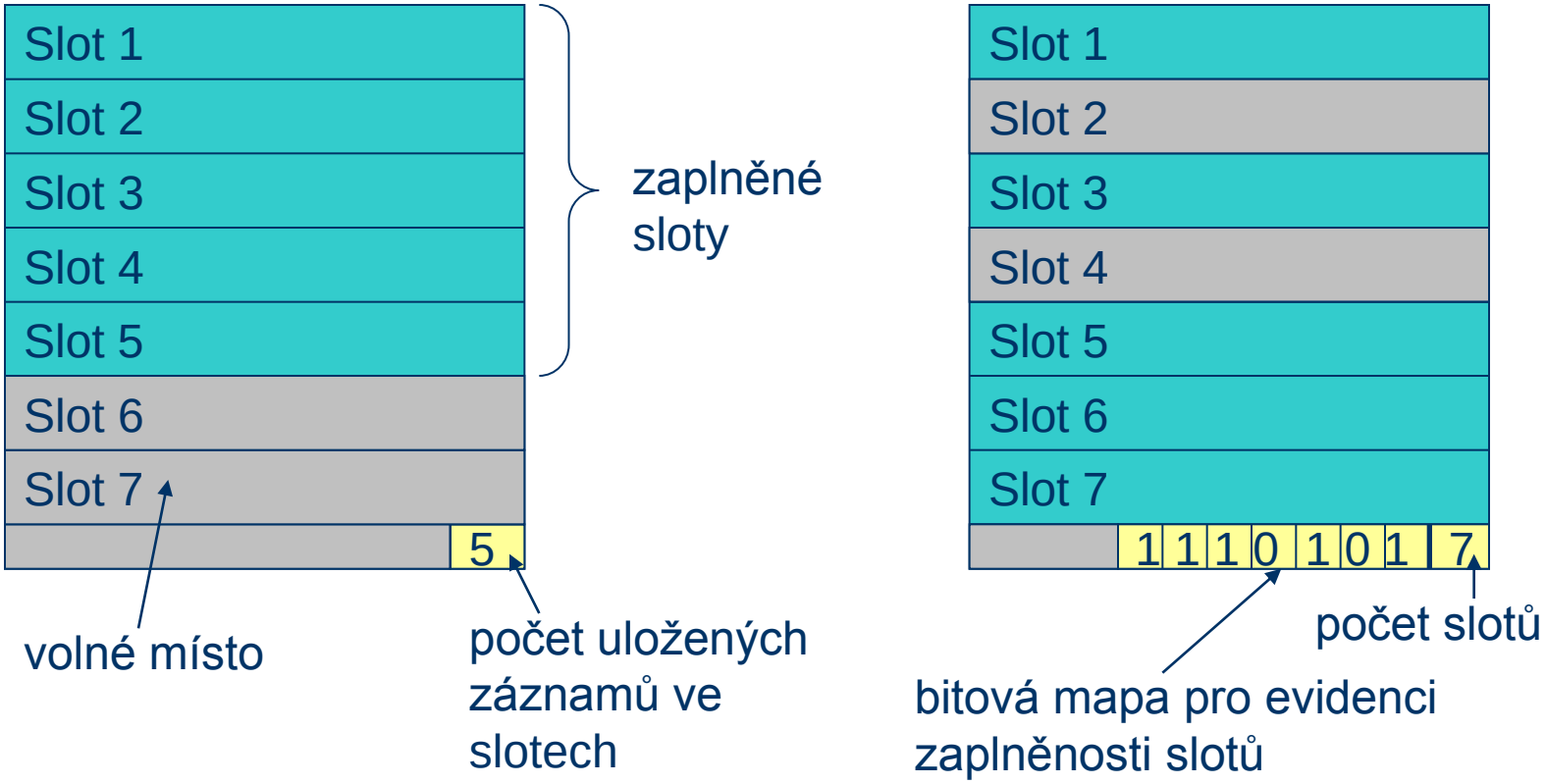
Stránkování, pokr.

- I/O jako jednotka časových nákladů
- stránka je rozdělena na sloty, do kterých se ukládají záznamy, identifikována před *page id*
- záznam může být uložen
  - přes více stránek = lepší využití místa, ale potřeba více I/O pro manipulaci se záznamem
  - nebo jen v jedné stránce (za předpokladu že se tam vejde) = příp. nevyužití celé stránky, méně I/O
  - v ideálním případě záznamy bezezbytku vyplňují stránku
- záznam identifikován pomocí **rid** (record id), což je dvojice *page id* a *slot it*

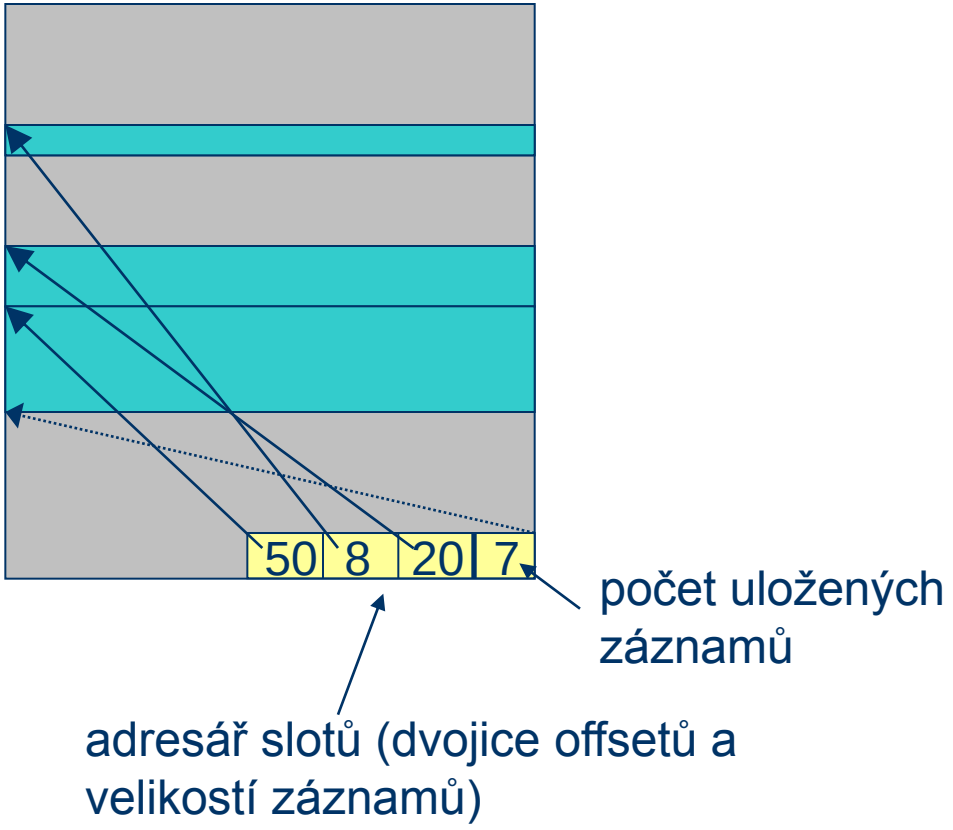
Stránkování, pokr.

- záznam je tvořen hodnotami datových typů pevné velikosti → pevná velikost záznamu
- přítomnost datových typů proměnlivé velikosti → proměnlivá velikost záznamu např. typy varchar(X), blob, ...
- záznamy pevné délky = sloty pevné délky
- záznamy proměnlivé délky = potřeba adresáře slotů v hlavičce každé stránky

Organizace stránky pro záznamy pevné velikosti, příklad



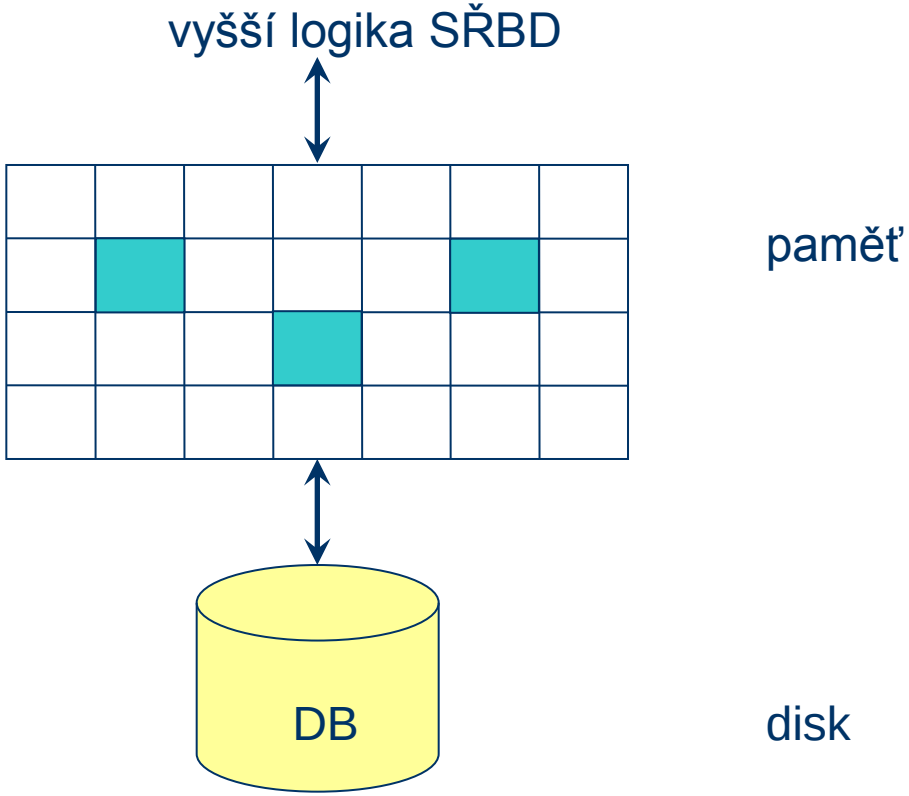
Organizace stránky pro záznamy proměnlivé velikosti, příklad



Buffer a jeho správa

- buffer = kus hlavní paměti pro dočasné uchování diskových stránek, diskové stránky se mapují do rámců v paměti 1:1
- každý rámec má 2 příznaky: **pin_count** (počet referencí na stránku v rámci) a **dirty** (příznaky modifikace záznamů)
- slouží k urychlení opakovaného přístupu ke stránkám - správce bufferu implementuje operace **read** a **write**
- odstínění vyšší logiky SŘBD od diskového managementu
- implementace **read** provede načtení stránky z bufferu, pokud tam není, provede se nejdříve načtení z disku (fetch), zvýšení **pin_count**
- implementace **write** zapíše stránku do bufferu, nastaví se **dirty**
- pokud v bufferu není místo (během read nebo write), uvolní se nějaká jiná stránka → různé politiky uvolňování, např. LRU (least-recently-used), pokud má uvolňovaná stránka nastaveno **dirty**, uloží se (store)

Buffer a jeho správa, schéma



Organizace databáze

- datové soubory
  (obsahující veškerá data tabulek)
- indexové soubory
- systémový katalog – obsahuje metadata
  - schémata tabulek
  - jména indexů
  - integritní omezení, klíče, atd.

Datové soubory

- halda
- uspořádaný soubor
- hašovaný soubor

Sledujeme průměrné I/O náklady jednoduchých operací:

1) sekvenční načtení záznamů
2) vyhledání záznamů na rovnost (podle vyhledávacího klíče)
3) vyhledání záznamů na rozsah (podle vyhledávacího klíče)
4) vložení záznamu
5) vymazání záznamu

Cost model:

N = počet stránek, R = počet záznamů na stránku

Jednoduché operace, SQL příklady

- sekvenční načtení
  SELECT * FROM Zaměstnanci
- vyhledání na rovnost
  SELECT * FROM Zaměstnanci WHERE věk = 40
- vyhledání na rozsah
  SELECT * FROM Zaměstnanci
  WHERE mzda > 10000 AND mzda < 20000
- vložení záznamu
  INSERT INTO Zaměstnanci VALUES (...)
- vymazání záznamu podle **rid**
  DELETE FROM Zaměstnanci WHERE rid = 1234
- DELETE FROM Zaměstnanci WHERE mzda < 5000

Halda (heap file)

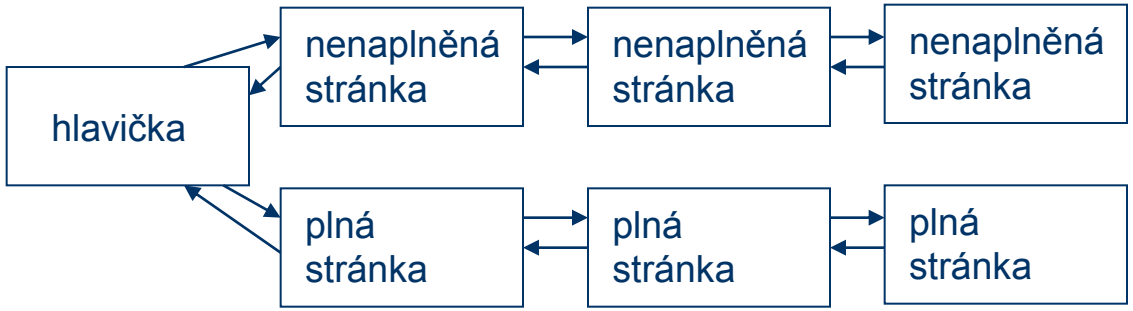
- záznamy ve stránkách uloženy neuspořádaně sekvenčně za sebou, resp. jsou ukládány tak, jak přicházejí požadavky *insert*
- vyhledání stránky možné pouze sekvenčním průchodem (a operace GetNext)
- rychlé vkládání záznamů na konec souboru
- problémy s mazáním → „díry“ (kusy prázdného prostoru)

Údržba prázdných stránek haldy

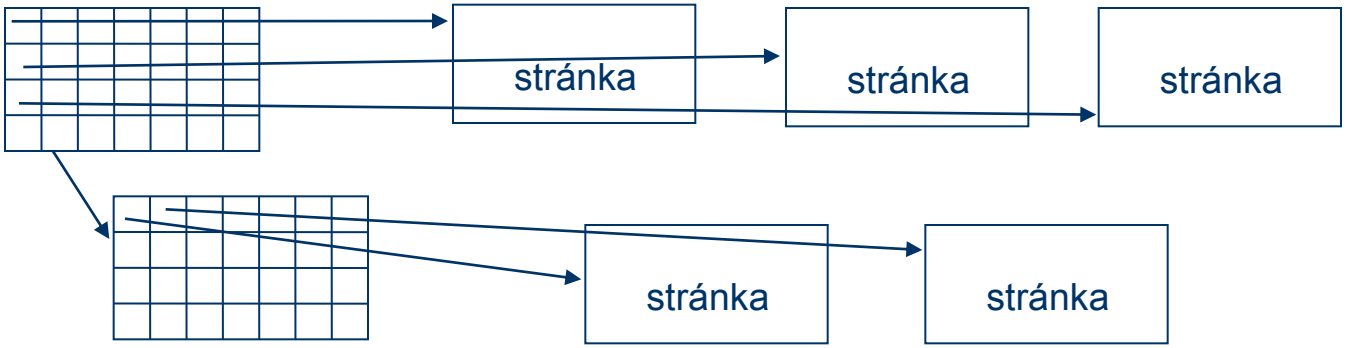
- dvojitě spojový seznam
  - hlavička + seznamy zaplněných a nezaplněných stránek
- adresář stránek
  - spojový seznam adresářových stránek
  - každá položka v adresáři ukazuje na datovou stránku
  - příznakový bit zaplněnosti stránky pro každou položku

Údržba prázdných stránek haldy, pokr.

dvojitě spojový seznam



adresář stránek



Halda, náklady jednoduchých operací

- sekvenční načtení = N
- vyhledávání na rovnost = 0,5*N nebo N
- vyhledávání na rozsah = N
- vložení záznamu = 1
- vymazání záznamu = 2 za předpokladu, že vyhledávání podle **rid** stojí 1 I/O,
  pokud se maže na shodu nebo na rozsah, náklady jsou N nebo 2*N

Setříděný soubor (sorted file)

- záznamy ve stránkách uloženy uspořádaně podle vyhledávacího klíče (jeden nebo více atributů)
- stránky souboru jsou udržovány spojitě,
  tj. neexistují „díry“ prázdného prostoru
- umožňuje rychlé vyhledávání podle klíče a to jak na rovnost, tak na rozsah
- pomalé vkládání a mazání, „hýbání“ se zbytkem stránek
- v praxi se používá kompromis – za začátku je setříděný soubor, každá stránka má „volnou rezervu“, kam se vkládá; pokud je rezerva zaplněna, využívají se aktualizační stránky (spojový seznam). Jednou za čas je třeba provést reorganizaci, tj. setřídění

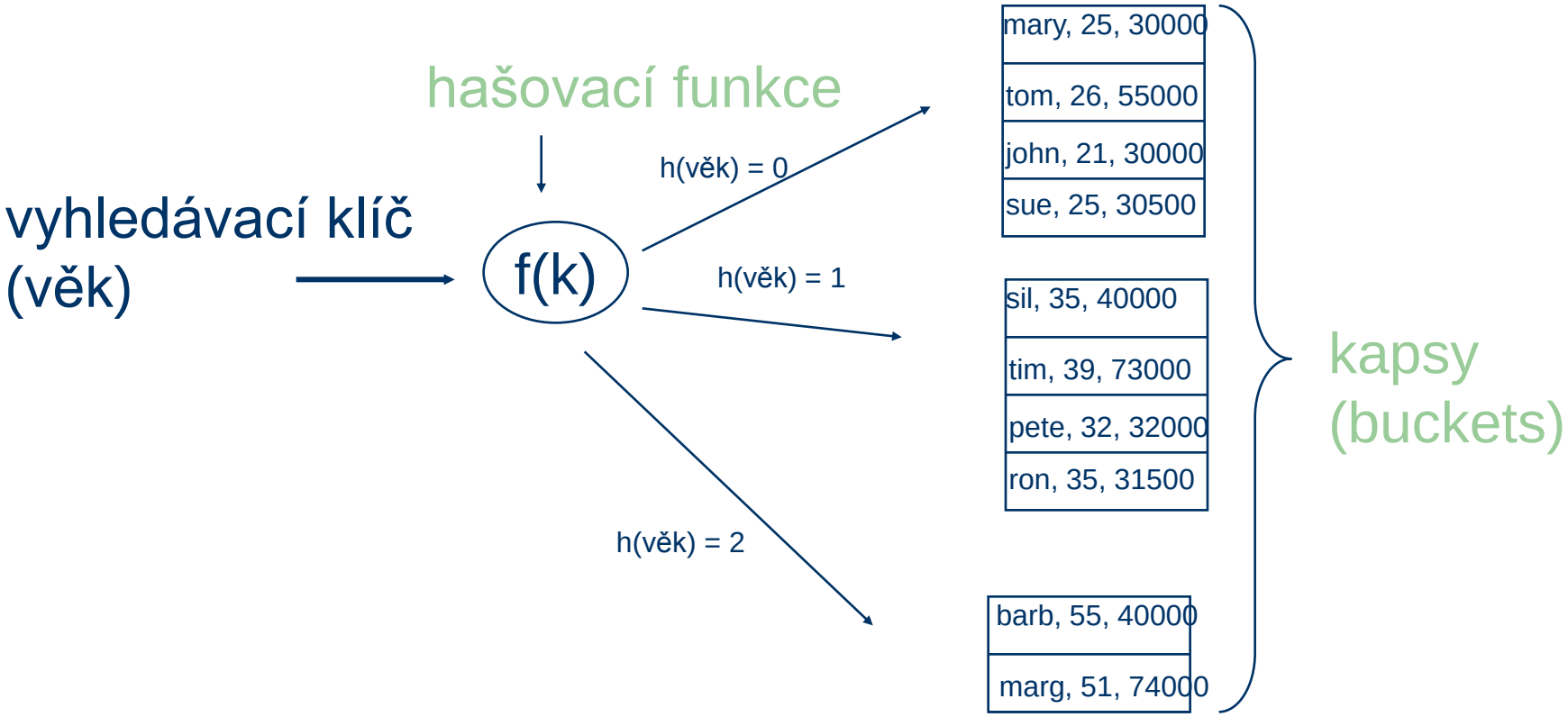
Setříděný soubor, náklady jednoduchých operací

- sekvenční načtení = N
- vyhledávání na rovnost = $\log_2 N$ nebo N
- vyhledávání na rozsah = $\log_2 N + M$
  (kde M je počet relevantních stránek)
- vložení záznamu = N
- vymazání záznamu = $\log_2 N + N$ podle klíče, jinak 1,5*N

Hašovaný soubor (hashed file)

- organizován do skupiny K kapes (buckets), kapsa může sestávat z několika stránek
- záznam je vložen/čten do/z kapsy, která je určena hašovací funkcí a klíčem pro vyhledání; id kapsy = f(klíč)
- pokud není v kapse místo, vytvoří se nové stránky, které se na kapsu napojí (spojový seznam)
- rychlé dotazy na shodu a mazání na shodu
- vyšší prostorová režie, komplikace se zřetězenými stránkami (řeší dynamické hašovací techniky)

Hašovaný soubor



Hašovaný soubor, náklady jednoduchých operací

- sekvenční načtení = N
- vyhledávání na rovnost = N/K (v ideálním případě)
- vyhledávání na rozsah = N
- vložení záznamu = N/K (v ideálním případě)
- vymazání záznamu na shodu = N/K + 1 (v ideálním případě), jinak N

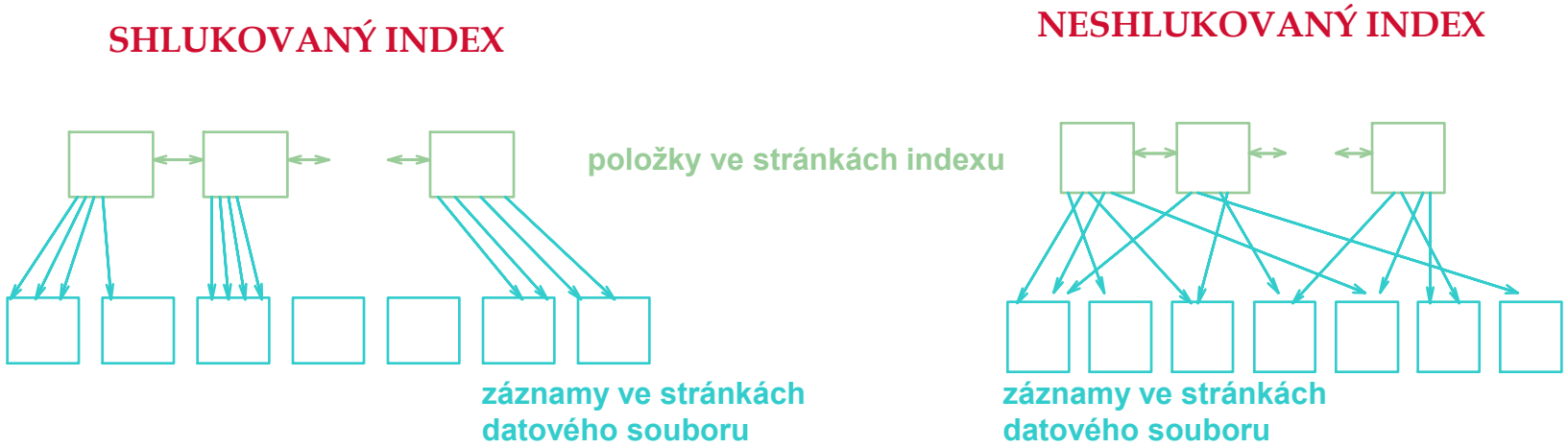
Indexování

- index je pomocná struktura umožňující rychle vyhledávat podle vyhledávacího klíče (klíčů)
- organizována do stránek podobně jako datové soubory
- zpravidla v jiném souboru
- obsahuje pouze (některé) hodnoty klíčů a odkazy k příslušným záznamům (tj. rid)
- spotřebují daleko menší velikost prostoru (např. 100x méně) než datové soubory

Indexování, principy

- položka indexu může obsahovat
  - celý záznam (index a datový soubor splývají)
  - dvojici <klíč, rid>
  - dvojici <klíč, rid-list>, kde rid-list obsahuje seznam odkazů na záznamy se stejným klíčem
- shlukované vs. neshlukované indexy
  - **shlukované:** uspořádání položek ve stránkách indexu je (téměř) stejné jako uspořádání záznamů ve stránkách datového souboru, tuto vlastnost mají pouze stromové indexy + indexy obsahující celé záznamy (i hašované)
  - **neshlukované:** pořadí klíčů v obou strukturách není dodrženo

Indexování, principy



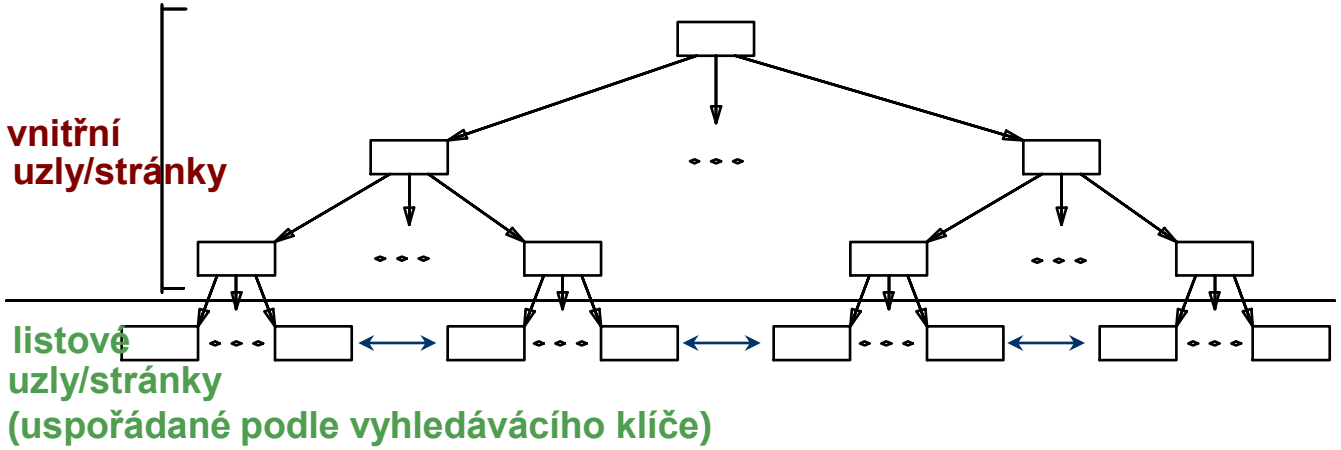
Výhodou shlukovaného indexu je velké zrychlení při vyhledávání na rozsah (rozsahový/intervalový dotaz), neboť stránky se záznamy jsou čteny sekvenčně. U neshlukovaného (a navíc stromového) indexu se sekvenčně čtou pouze stránky indexu.

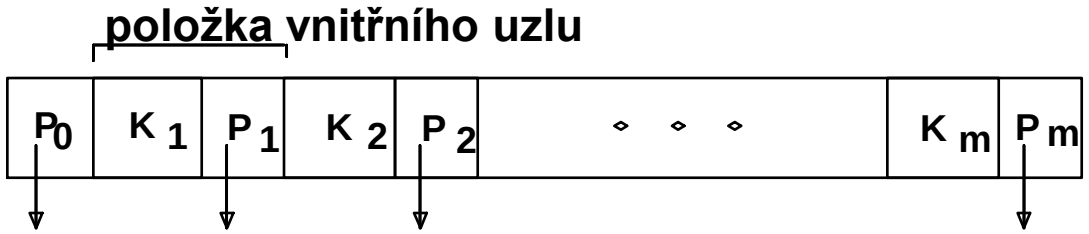
Nevýhody: velká režie při udržování uspořádání datového souboru, zvlášť pokud existují další indexy

B+-strom

- vychází z B-stromu, což je stránkovaný, vyvážený stromový index (Rudolf Bayer, 1972).
- poskytuje logaritmické složitosti pro vkládání, dotaz na shodu, mazání na shodu
- zaručuje 50% zaplněnost uzlů (stránek)
- B+-strom rozšiřuje B-strom o
  - provázání listových stránek pro efektivní rozsahové dotazy
  - vnitřní uzly obsahují indexované intervaly, tj. všechny klíče jsou v listech

B+-strom, schéma





Demo: http://slady.net/java/bt/

Hašovaný index

- podobně jako hašovaný soubor využívá kapsy a hašovací funkci
- v kapsách jsou pouze hodnoty klíčů spolu s odkazy na záznamy **rid**
- stejné výhody/nevýhody

Bitové mapy

- jsou vhodné pro indexování atributů s malou doménou (jednotky až desítky hodnot)
  - vhodné např. pro atribut **RODINNÝ_STAV** = {svobodný, ženatý, rozvedený, vdovec}
  - nevhodné např. pro atribut **CENA_VÝROBKU** (mnoho hodnot), tam bude lepší B-strom
- pro každou HODNOTU ***h*** indexovaného atributů ***a*** se zkonstruuje bitová mapa (binární vektor), kde jednička na pozici i znamená, že hodnota ***h*** se vyskytuje v i-tém záznamu tabulky (jako hodnota atributu ***a***) a platí
  - bitový součet (OR) všech map pro atribut vytvoří samé jedničky (každý záznam nabývá v daném atributu nějaké hodnoty)
  - bitový součin (AND) libovolných dvou map atributu je nula (každý záznam nabývá v atributu nejvýše jedné hodnoty)

Jméno	Adresa	Rodinný stav
František Novák	Liberec	svobodný
Rostislav Drobil	Praha	ženatý
René Vychodil	Ostrava	ženatý
Kamil Svoboda	Beroun	svobodný
Pavel Horák	Cheb	rozvedený

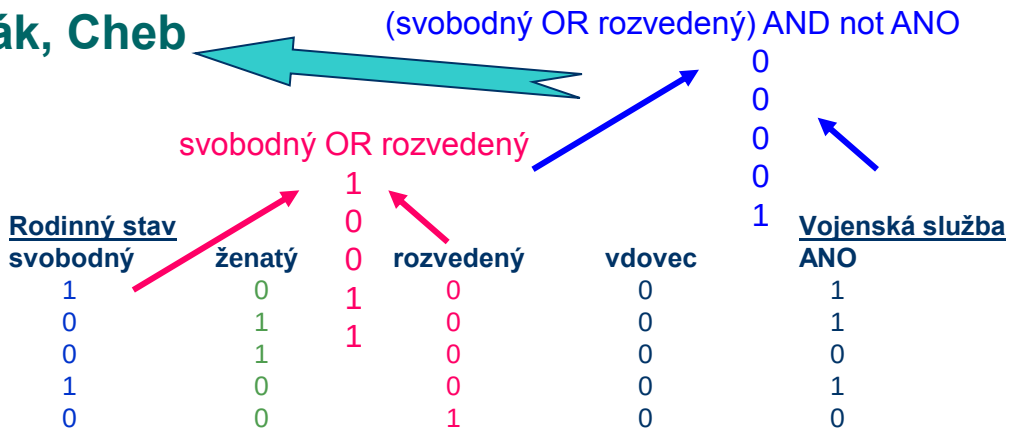
svobodný	ženatý	rozvedený	vdovec
1	0	0	0
0	1	0	0
0	1	0	0
1	0	0	0
0	0	1	0

Bitové mapy

- vyhodnocení dotazu
  - bitové operace s mapami jednotlivých hodnot atributů
  - výsledná bitová mapa označuje záznamy vyhovující dotazu
- příklad
  - **Kteří svobodní nebo rozvedení neabsolvovali vojenskou službu?**
    (bitmap(**svobodný**) **OR** bitmap(**rozvedený**)) **AND not** bitmap(**ANO**)

odpověď: Pavel Horák, Cheb

Jméno	Adresa	Vojenská služba	Rodinný stav
František Novák	Liberec	ANO	svobodný
Rostislav Drobil	Praha	ANO	ženatý
René Vychodil	Ostrava	NE	ženatý
Kamil Svoboda	Beroun	ANO	svobodný
Pavel Horák	Cheb	NE	rozvedený



Bitové mapy

- výhody
  - úspora místa, navíc lze efektivně (de)komprimovat podle potřeby
  - úspora místa souvisí i s rychlostí vyhodnocování dotazu, bitové operace jsou navíc rychlé
  - dotazy nad mapami lze jednoduše paralelizovat
- nevýhody
  - omezeno pouze na atributy s malou doménou
  - intervalové dotazy se zpomalují přímoúměrně s počtem hodnot v intervalu (je potřeba procházet bitové mapy všech hodnot v intervalu, neexistuje uspořádání)

Víceatributové indexování

- uvažujme konjunktivní rozsahový dotaz
  SELECT * FROM Zaměstnanci WHERE
  mzda BETWEEN 10000 AND 30000 AND
  věk < 40 AND
  name BETWEEN ‘Dvořák’ AND ‘Procházka’
- jednoduchá řešení řešitelná pomocí B+-stromu (počet indexovaných atributů M = 3):
  1) tři nezávislé indexy
  2) jeden index M zřetězených atributů

  - obě řešení jsou špatná, druhá varianta stačí pouze pro dotazy na shodu (tedy ne na rozsah), první ani na to

Víceatributové indexování, příklady

Tři samostatné indexy:

...WHERE **mzda BETWEEN 10000 AND 30000** AND **věk < 40** AND **name BETWEEN 'Dvořák' AND 'Procházka'**

r1	Čech Jaroslav	17000	27
r2	Dostál Jan	21000	33
r3	Malý Zdeněk	15000	45
r4	Mrázek František	22000	37
r5	Novák Josef	32000	25
r6	Novák Karel	13000	19
r7	Oplustil Arnošt	9000	36
r8	Papoušek Jindřich	19000	50
r9	Richter Tomáš	26000	41
r10	Zlámal Alois	13000	52

{r3, r4, r5, r6, r7, r8}

r7	Oplustil Arnošt	9000	36
r6	Novák Karel	13000	19
r10	Zlámal Alois	13000	52
r3	Malý Zdeněk	15000	45
r1	Čech Jaroslav	17000	27
r8	Papoušek Jindřich	19000	50
r2	Dostál Jan	21000	33
r4	Mrázek František	22000	37
r9	Richter Tomáš	26000	41
r5	Novák Josef	32000	25

{r6, r10, r3, r1, r8, r2, r4, r9}

r6	Novák Karel	13000	19
r5	Novák Josef	32000	25
r1	Čech Jaroslav	17000	27
r2	Dostál Jan	21000	33
r7	Oplustil Arnošt	9000	36
r4	Mrázek František	22000	37
r9	Richter Tomáš	26000	41
r3	Malý Zdeněk	15000	45
r8	Papoušek Jindřich	19000	50
r10	Zlámal Alois	13000	52

{r6, r5, r1, r2, r7, r4}

průnik = {r4, r6}

Víceatributové indexování, příklady

Index zřetězených atributů:

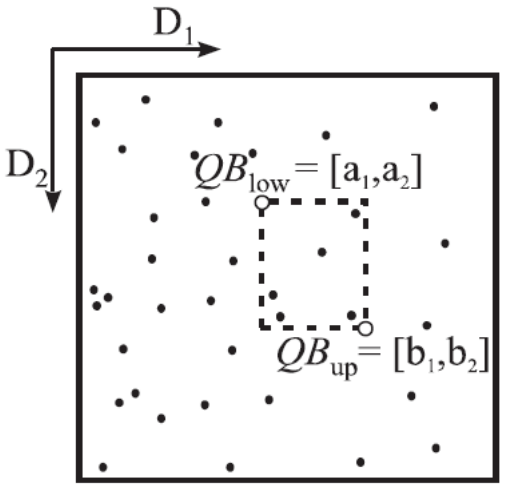
...WHERE **mzda BETWEEN 10000 AND 30000** AND **věk < 40** AND **name BETWEEN 'Dvořák' AND 'Procházka'**

r1	Čech Jaroslav	17000	27
r2	Dostál Jan	21000	33
r3	Malý Zdeněk	15000	45
r4	Mrázek František	22000	37
r5	Novák Josef	32000	25
r6	Novák Karel	13000	19
r7	Oplustil Arnošt	9000	36
r8	Papoušek Jindřich	19000	50
r9	Richter Tomáš	26000	41
r10	Zlámal Alois	13000	52

{r4, r6}

Prostorové indexování

- abstrakce M-tice klíčů jako M-rozměrných vektorů
  <Novák Josef, 32000, 25> ⇒ [sig('Novák Josef'), 32000, 25]
  - musí se zachovat uspořádání klíčů, např. pro sig(*)
- transformace na problém vyhledávání v M-rozměrném prostoru R^M
- konjunktivní rozsahový dotaz = (hyper)-kvádr QB v prostoru R^M vymezen dvěma body, dolní a horní meze rozsahů



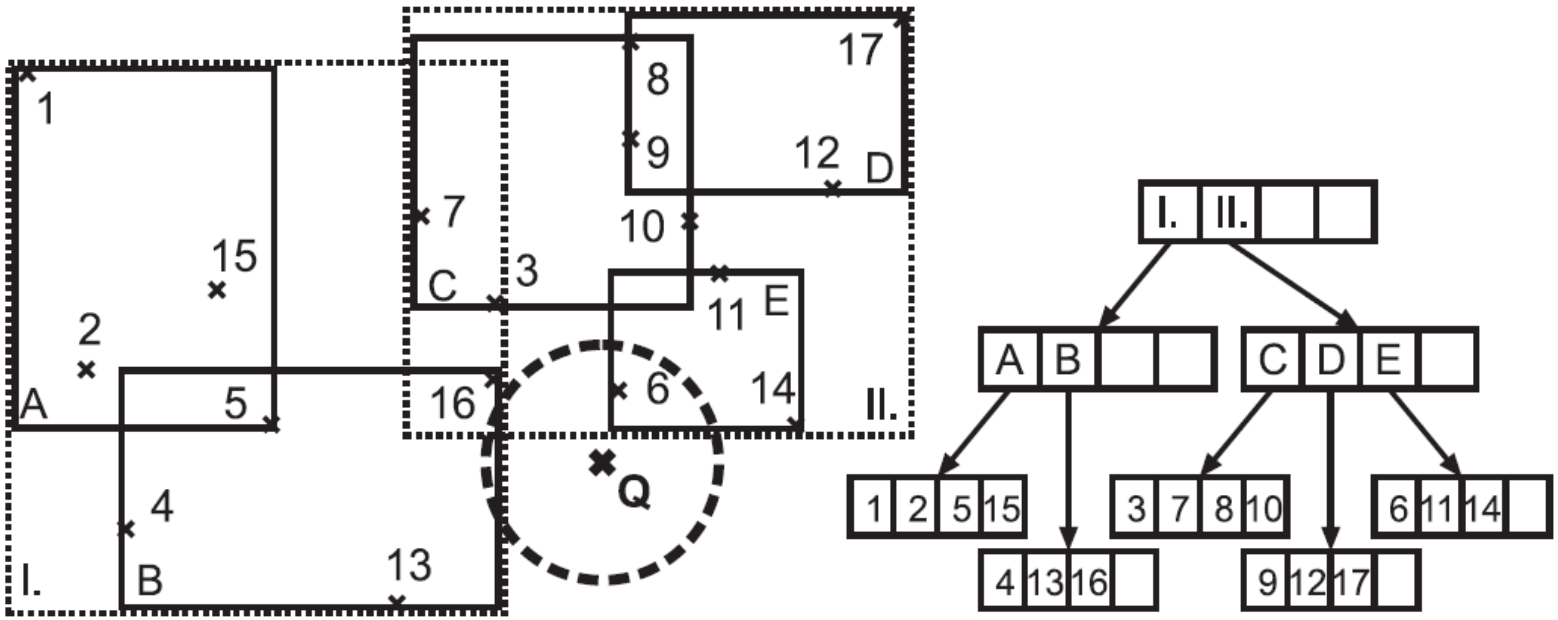
Prostorové indexování

- různé indexy (spatial access methods), založené na stromové struktuře, hašování i sekvenčním průchodu
- společným rysem nesekvenčních indexů je snaha o shlukování těch vektorů blízko ve stejné části indexu, které jsou shlukovány i v prostoru
- během rozsahového dotazu je potom (v ideálním případě) přistupováno jen k těm stránkám, které obsahují klíče uvnitř dotazovacího kvádru
- velmi dobře fungují do dimenze 10, potom přestávají být účinné a lepší jsou sekvenční indexy, ve kterých jsou klíče reprezentovány malým počtem bitů

Prostorové indexování

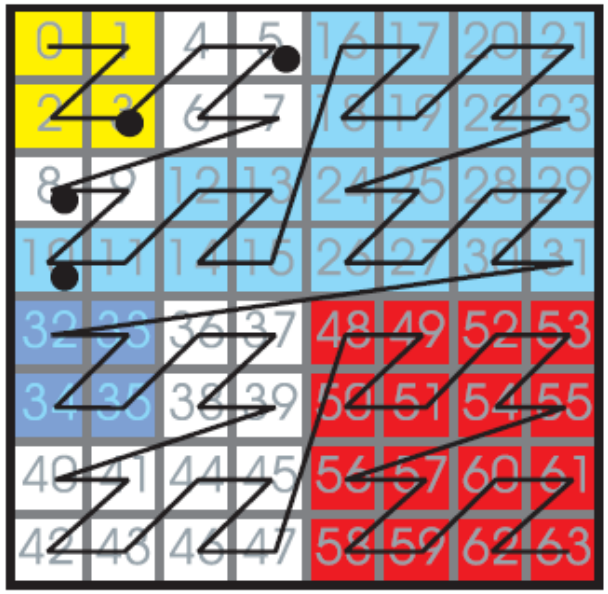
- stromové indexy
  - R-strom, UB-strom
- hašované indexy
  - Grid file
- sekvenční indexy
  - VA-file

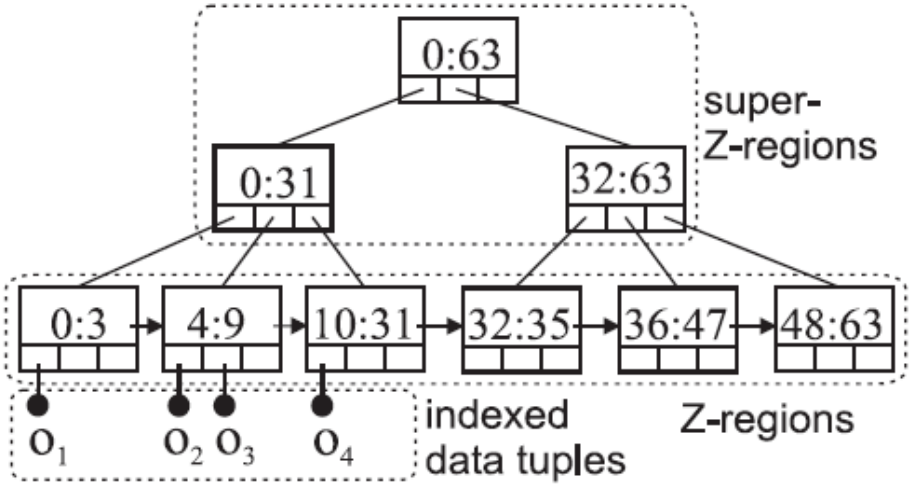
R-strom



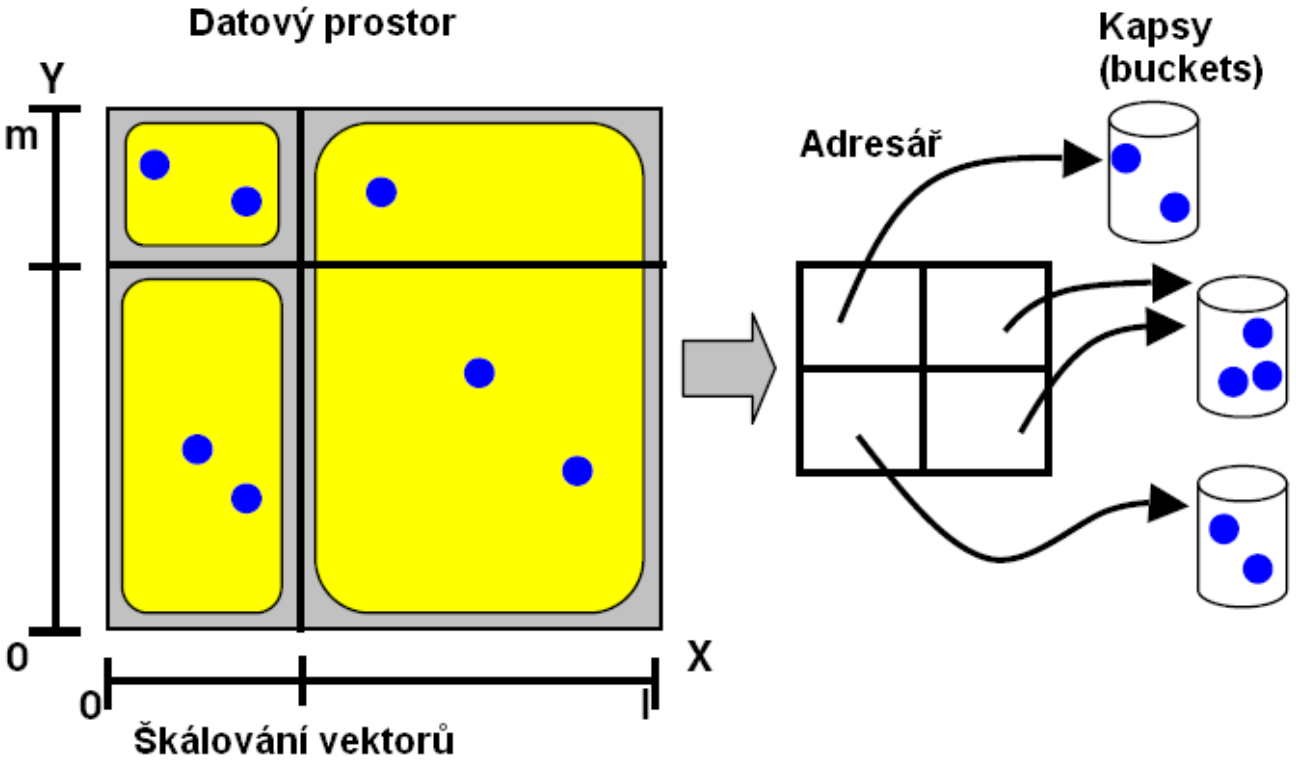
Demo: http://www.dbnet.ece.ntua.gr/~mario/rtree/

UB-strom




0:63
super-
Z-regions
0:31
32:63
0:3
4:9
10:31
32:35
36:47
48:63
o_1
o_2 o_3
o_4
indexed
data tuples
Z-regions

Grid file


Datový prostor
Y
m
0
0
X
l
Škálování vektorů
Kapsy
(buckets)
Adresář