Prednáška 2

Porovnáme

- Možnosti organizácii súborov:
  - Súbory na halde (náhodné poradie; vloženie na koniec súboru)
  - Utriedené súbory, (k*)
  - Klastrovaný B+ strom, (k*)
  - Súbor na halde s neklastrovaným B+ stromom, <k, rid>
  - Klastrovaný hash index, (k*)
- Operácie:
  - Table scan (TS): čítanie všetkých záznamov v tabuľke
  - Vyhľadávanie rovnosti (... where __ = __)
  - Hľadanie rozsahu (... where __ <> __)
  - Vloženie záznamu
  - Zmazanie záznamu

Predpoklady v našej analýze

- Súbory v halde:
  - Hľadanie rovnosti kľúča; presne jedna zhoda.
- Utriedené súbory:
  - Súbory defragmentované priebežne → žiadne voľné miesta v súbore.
- Neklastrované indexy:
  - Veľkosť dátovej položky = 10% veľkosti záznamu.
- Hash index
  - Cca 80% obsadenosť stránky, keďže sa nedefragmentuje priebežne

  → veľkosť súboru = 1.25 veľkosti dát.
- B+ strom
  - Cca 67% obsadenosť stránky, keďže sa nedefragmentuje priebežne.

  → veľkosť súboru = 1.5 veľkosti dát.

Sledované hodnoty

- B = počet zaplnených dátových stránok
- D = čas potrebný na prečítanie/zápis stránky
- R = maximálny počet záznamov v stránke
- C = čas potrebný na spracovanie záznamu
- A = počet stránok s rovnosťou

Halda

- **TS**: prečítame všetky záznamy každej stránky B*(D + RC) +
- **Rovnosť**:
  - unique kľúče: niekedy môžeme nájsť skôr, niekedy neskôr, v priemere ½ TS -
  - non-unique kľúče: TS -
- **Rozsahový dopyt**: TS -
- **Vloženie**: prečítame poslednú stránku, vložíme záznam a zapíšeme stránku 2D + C ++
- **Zmazanie**: ako v prípade hľadania rovnosti TS -

Utriedený súbor

- **TS**: prečítame všetky záznamy každej stránky $B*(D + RC)$ +
- **Rovnosť**: používa sa binárne vyhľadávanie $D*\log_2 B + C*\log_2 R$ +
- **Rozsahový dopyt**: nájde sa rovnosť + počet záznamov Rovnosť + AD ++
- **Vloženie**: bin. vyhľ. nájde miesto kam vložiť a od toho miesta posunie všetky záznamy vpravo. V priemere pol súboru prečíta aj zapíse, teda Rovnosť + $2*(\frac{1}{2}B*(D + RC))$ = Rovnosť + $B*(D + RC)$ --
- **Zmazanie**: Vloženie --

B+ strom

- **TS**: číta celý súbor a pozerá, či je listový uzol alebo nie (1/0,67)*B*(D + RC) -
  - 67%-ná priemerná zapĺnenosť stránok zhoršuje skenovanie
- **Rovnosť**: závisí od výšky stromu a arity kľúcov VS*(D + RC) +
  - výška stromu VS ~ $\log_R B$
- **Rozsahový dopyt**: zijde k listom a ide po šípkach Rovnosť + AD ++
- **Vloženie**: nájde miesto a vloží, no niekedy musí deliť a teda viac stránok prečítať aj zapísať Rovnosť*(1..VS) + (D + RC) +
- **Zmazanie**: Vloženie +

Neklastrovaný B+ strom

- **TS**: nepoužíva index ale rid, efektívny ak ide o viacstĺpcové kľúče ++
- **Rovnosť**: zijde k listom a jedno čítanie Rovnosť* + D + RC +
- **Rozsahový dopyt:** záleží od selektivity (lepší ako predošlý B+ strom, ak je do 5%, ideálne pod 1%)

- **Vloženie**: nemá zmysel, dáta sa vkladajú inde, strom sa len akutalizuje
- **Zmazanie**: Rozsahový dobyt alebo Rovnosť2 + reálne zmazanie +

Poznámka: Rovnosť* z predošlej snímky

Hash index (klastrovaný)

- **TS**: lepšia zapĺnenosť oproti B+ stromom (1/0,8)*B*(D + RC) -
- **Rovnosť**: vyráta hash, čo povie oblasť kam treba ísť 1,2*(D + RC) ++
- **Rozsahový dopyt:** interval cez hash sa nerobí TS -
- **Vloženie**: nájde oblasť a zapíše (niekedy vytvára stránku pretečenia) +
- **Zmazanie**: Vloženie ++

Poznámky:

- Štandardne je 5 : 1 – pomer prvých stránok a stránok pretečenia
- priemerné zaplnenie: 80%

Celkové porovnanie – počtu stránok

	Table scan	Rovnosť	Rozsah	Vloženie	Mazanie
Halda	+ (~B)	- (~0,5 B)	- (~B)	++ (2)	- (~0,5 B)
Utriedený	+ (~B)	+ (~$\log_2$B)	++ (~$\log_2$B+A)	-- (~B)	-- (~B)
B+strom	- (~1,49 B)	+ (~4)	++ (~4+A)	+ (~5 .. 12)	+ (~5 .. 12)
B+strom neklastr.	++ (~0,149*B)	+ (~4+1)	+ až -- (~4+A*R)		+ (~5 .. 12) + 2
Hash index	- (~1,25 B)	++ (~1,2)	- (~1,2 B)	+ (~2,2)	++ (~1,2)

Indexy so stromovou štruktúrou

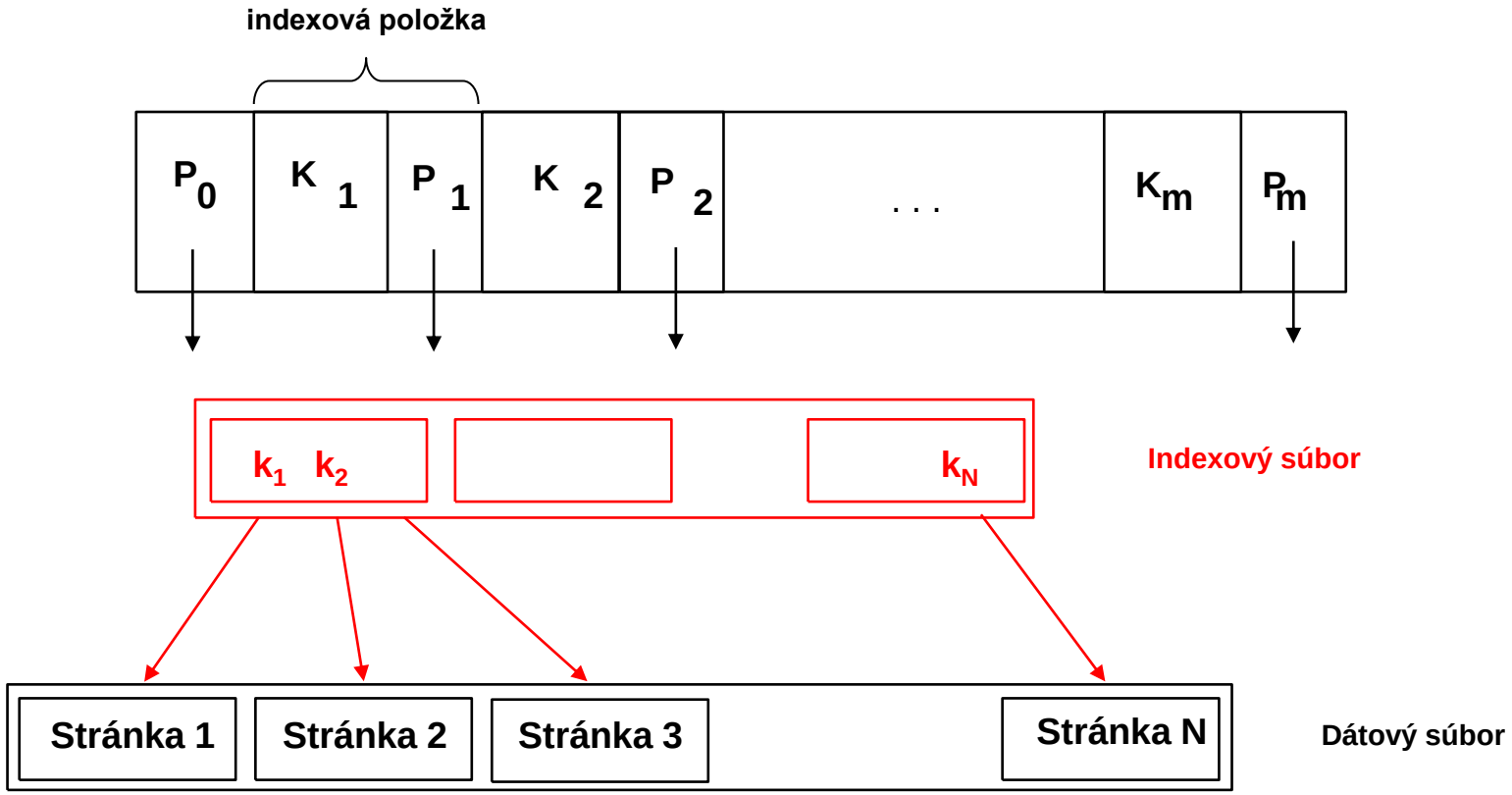
Kapitola 10

Úvod

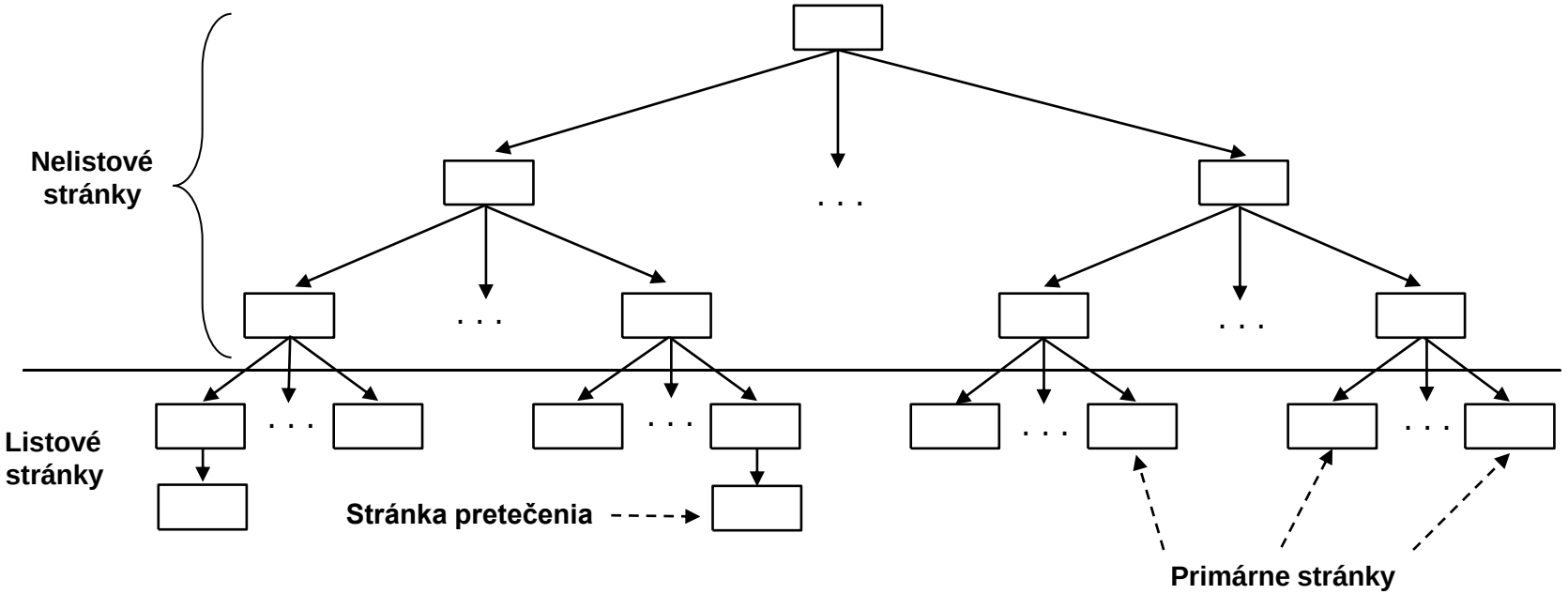
- Pomocou dátovej položky k* môžeme dostať jednu alebo viac záznamov
- Dátová položka môže byť:
  - dátový záznam s vyhľadávacím kľúčom k,
  - <k, rid>, kde rid je id dátového záznamu s vyhľadávacím kľúčom k
  - <k, rid-list>, kde rid-list je zoznam id dátových záznamov s vyhľadávacím kľúčom k
- Výber je nezávislý od indexovej techniky použitej na vyhľadanie dátových položiek k*.
- B+ strom je univerzálny (rôzne kľúče), jeho hlavnou výhodou je rozsahové vyhľadávanie, má dynamickú štruktúru
- Isam má statickú štruktúru → dáta sú finálne (málokedy sa niečo pridá/zmaže), môžeme robiť pre-fetching, vieme zabezpečiť 100%-nú zaplnenosť skoro všade

Vyhľadávania

- Vyhľadávanie môže prebiehať:
  - binárnym vyhľadávaním
  - zapamätaním si kľúča pre prvý záznam každej stránky.



ISAM



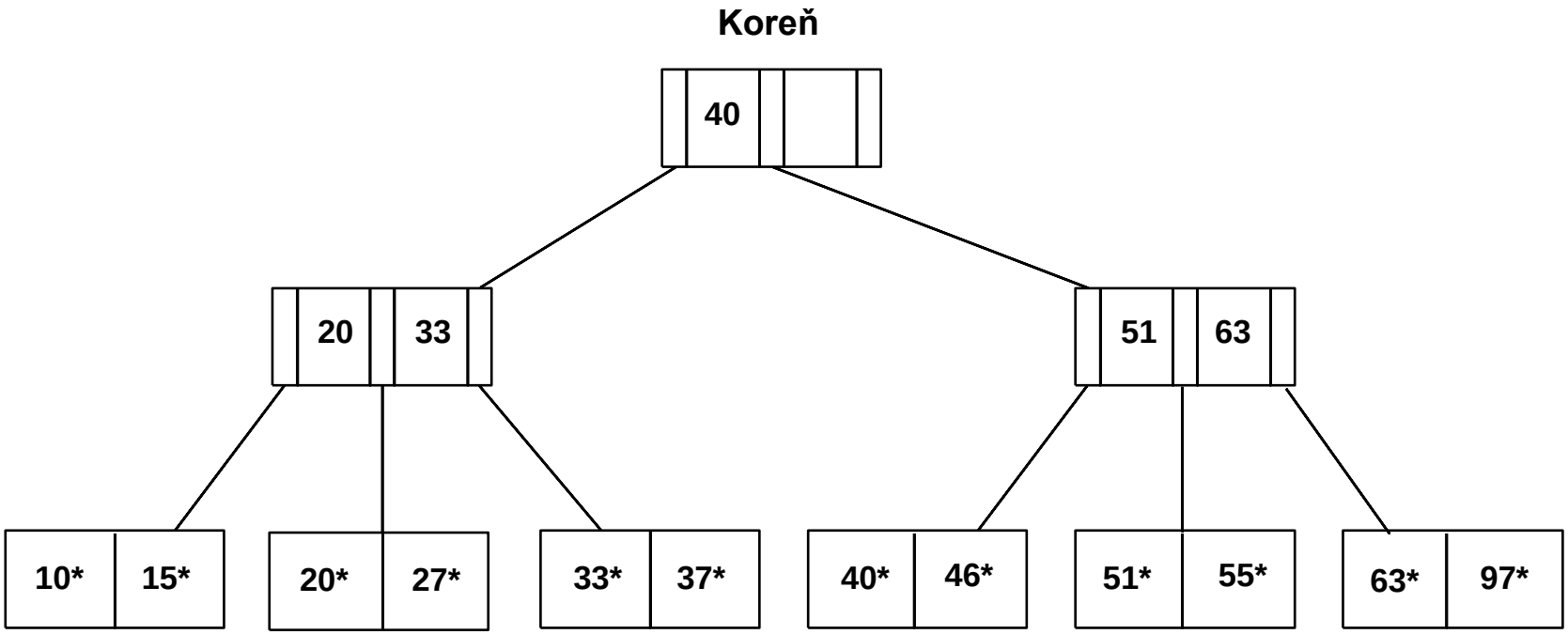
- Kľúč hovorí, že všetky záznamy menšie ako ten kľúč sú naľavo a väčšie rovné napravo
- Ak predsa pridávame niečo, treba vytvoriť stránku pretečenia
- Na disku sú najprv listové, potom vnútorné stránky a potom sú stránky pretečenia
- Keďže listové sú zaradom, netreba vedieť adresu predchodcu a nasledovníka → pre-fetching
- Listové stránky obsahujú iba dátové položky.

Poznámky k ISAM

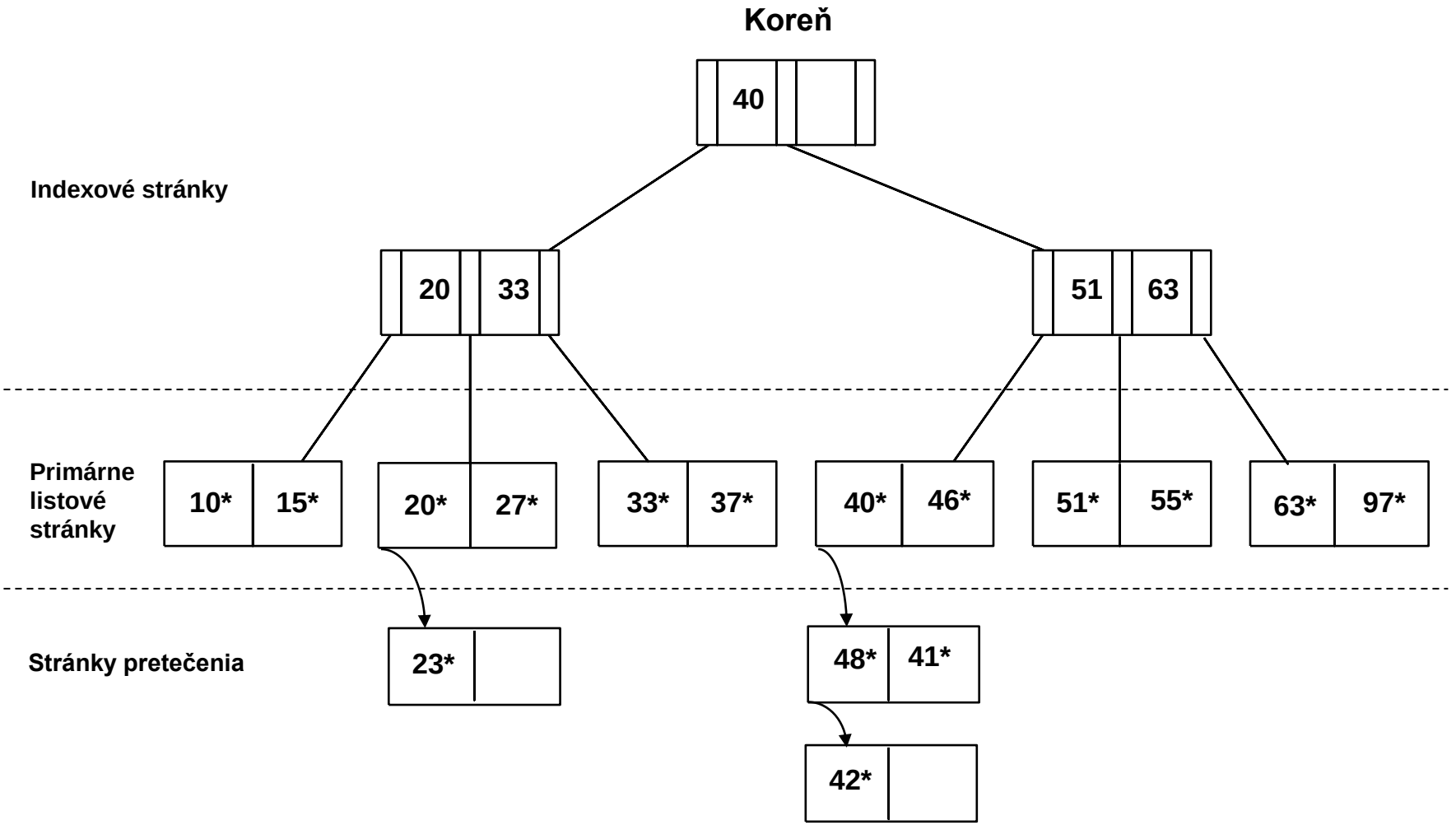
- Vytvorenie súboru:
  - Vytvorí sa utriedený zoznam, začnú sa vyrábať vnútorné uzly, indexové stránky a potom stránky pretečenia (tam dáta nemusia byť usporiadané).
- Hľadanie:
  - Začína v koreni a používa porovnanie kľúčov na nájdenie listu.
  - Cena $\log_R B$; R = #kľúčov v stránke, B = #listových stránok
- Vloženie:
  - Nájde zodpovedajúcu listovú dátovú položku a vloží daný údaj (vytvorí stránku pretečenia).
- Zmazanie:
  - Nájde a odstráni z listu alebo stránky pretečenia.

Statická stromová štruktúra: *vloženia/mazania ovplyvnia len listové stránky*.

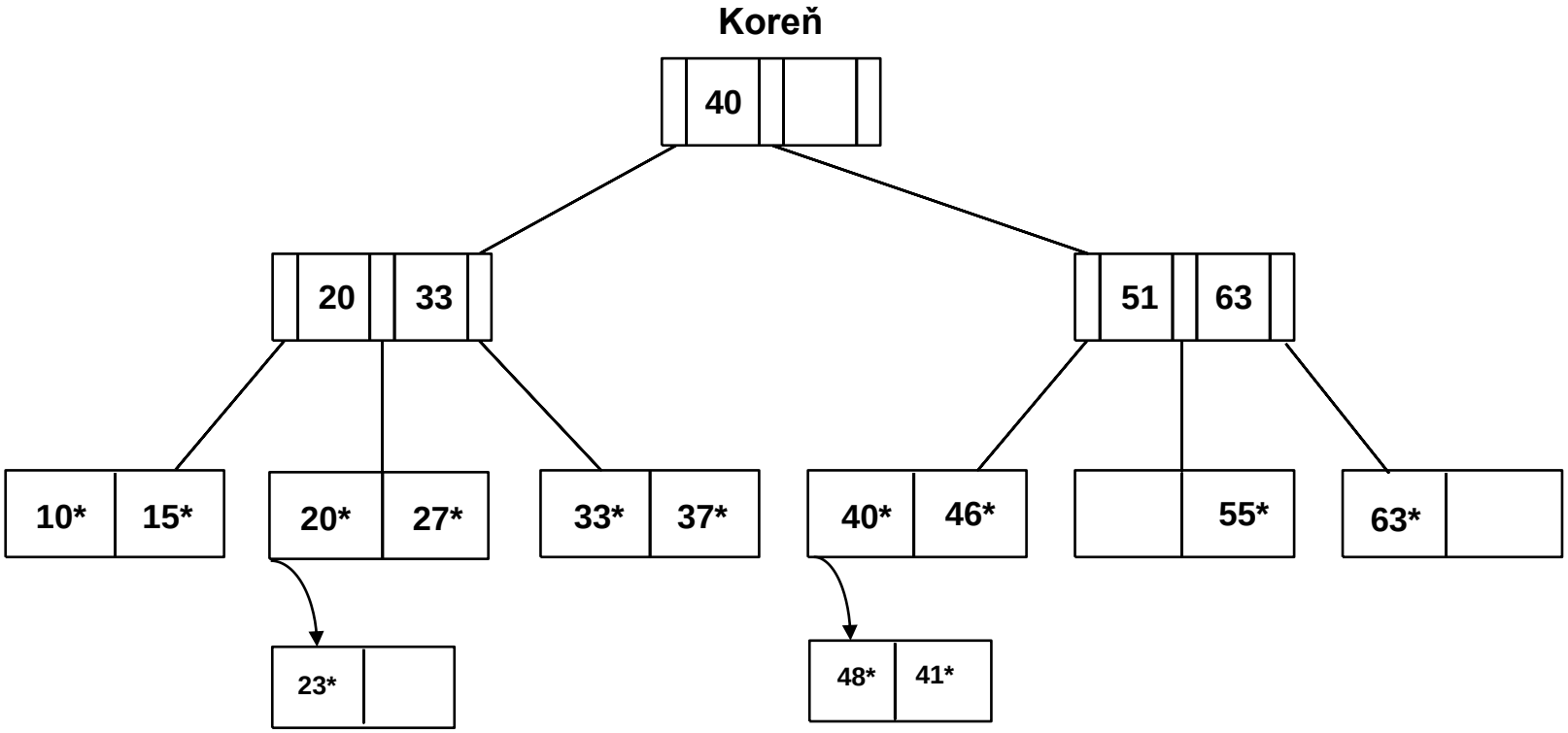
Príklad ISAM stromu



Po vložení 23*, 48*, 41*, 42* ...



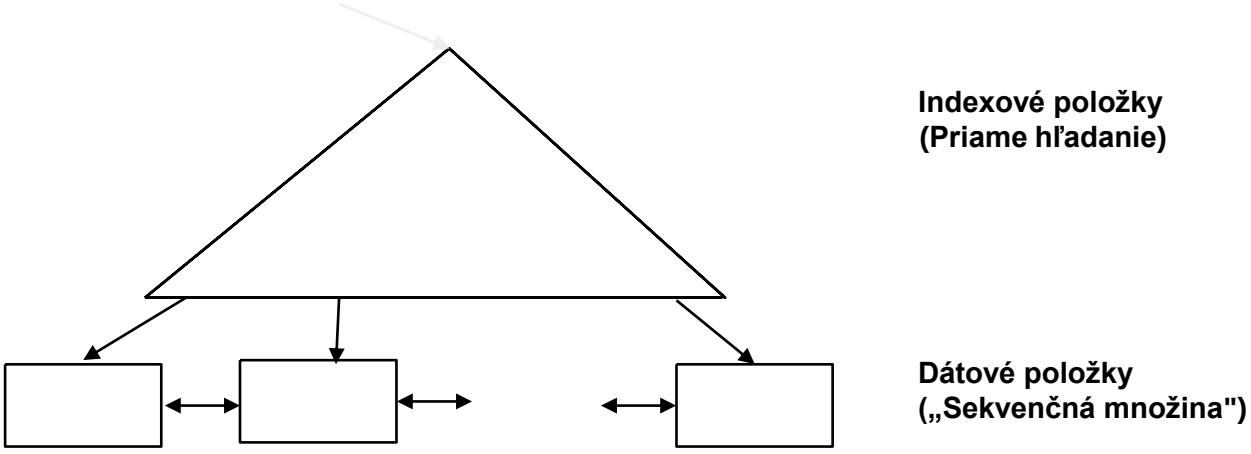
Po odstránení 42*, 51*, 97*



Pozor: 51 sa vyskytuje v indexových úrovniach, ale nie v liste!

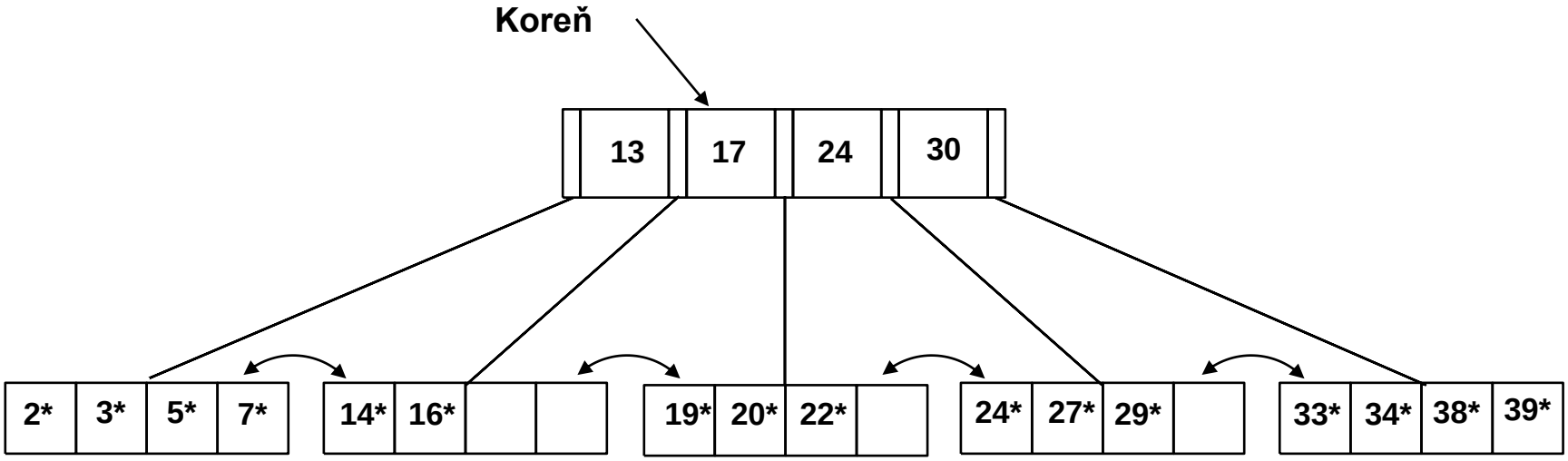
B+ strom: najpoužívanejší index

- Vloženie/zmazanie s cenou $\log_R B$ (R = zaplnenie, B = # listových stránok) → udržiava strom vyvážený.
- Minimum 50% zaplnenia (okrem koreňa).
- Každý uzol obsahuje d <= m <= 2d položiek. Parameter d sa nazýva rád stromu.
- Podporuje efektívne rozsahové hľadanie a hľadanie rovnosti.



Príklad B+ stromu

- Hľadanie začína v koreni a porovnania kľúčov ho nasmerujú k listu (ako v ISAM-e).
- Hľadanie 5*, 15*, všetky dátové položky >= 24* ...



Na základe hľadania 15 vieme, že nie je v strome!*

B+ strom prakticky

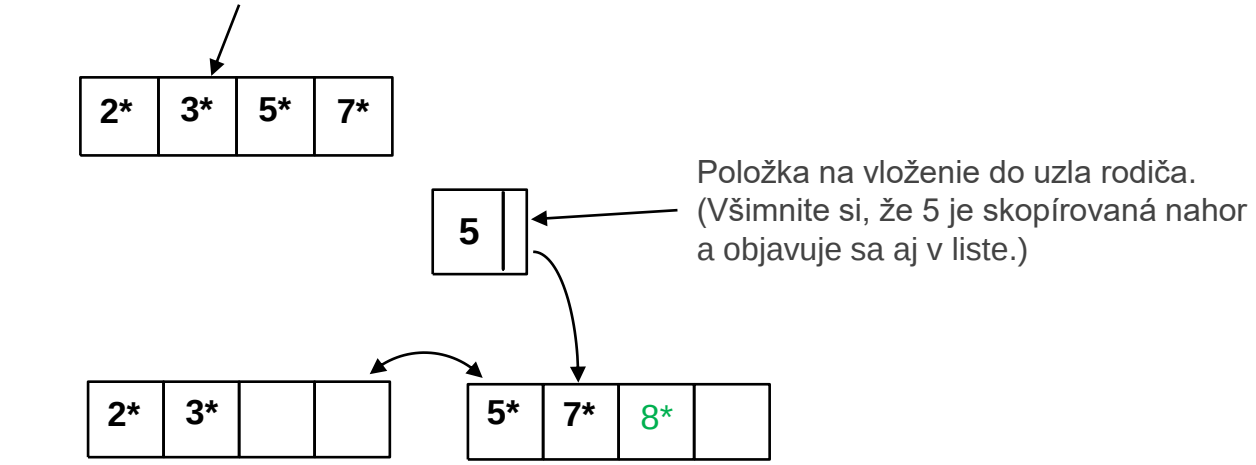
- Typický rád: 100. Typický faktor zaplnenia: 67%.
  - priemerné rozvetvenie = 133
- Typické kapacity:
  - Výška 4: 133^4 = 312,900,700 záznamov
  - Výška 3: 133^3 = 2,352,637 záznamov
- Často dokáže mať najvyššie úrovne v oblasti pamäti (Buffer pool):
  - Úroveň 1 = 1 stránka = 8 KB
  - Úroveň 2 = 133 stránok = 1 MB
  - Úroveň 3 = 17,689 stránok = 133 MB

Vloženie dátovej položky do B+ stromu

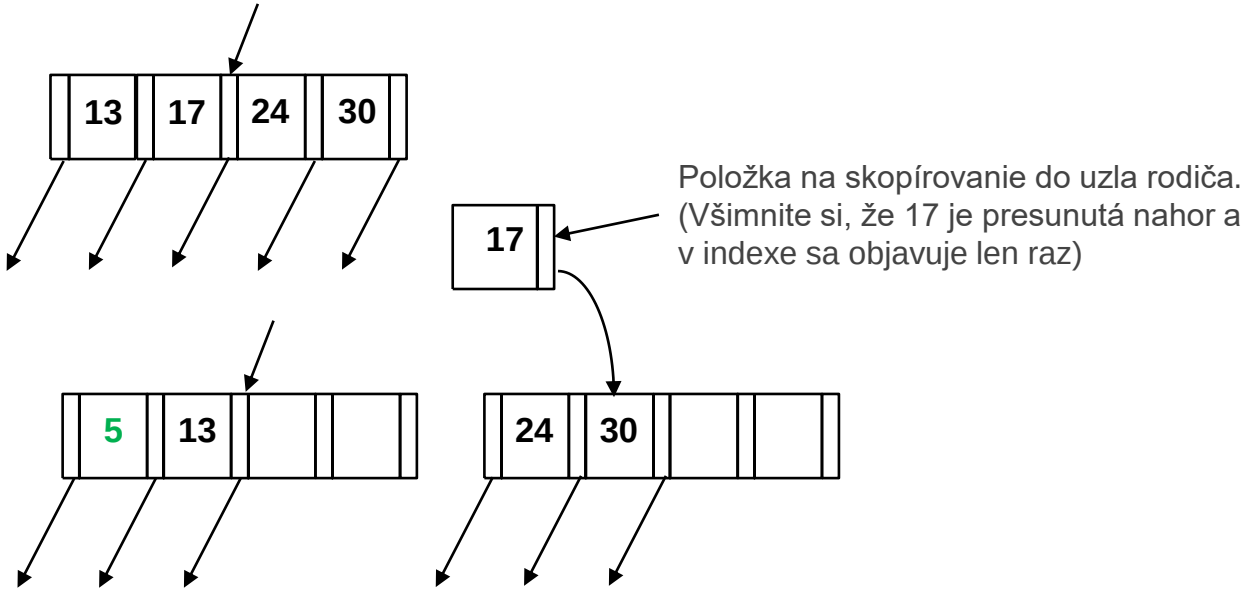
- Nájdi správny list *L.*
- Vlož dátovú položku do *L*.
  - Ak má *L* dosť miesta, *koniec*!
  - Ináč je potrebné *rozdeliť* *L (medzi L a nový uzol L2)*
    - Rovnomerne prerozdeľ položky, **skopíruj nahor** stredný kľúč.
    - Vlož indexovú položku ukazujúcu na *L2* do rodiča *L*.
- Toto sa môže diať rekurzívne
  - Na rozdelenie indexového uzla U, rovnomerne prerozdeľ položky medzi U a U2, ale **presuň nahor** stredný kľúč.
- Delenia "zväčšujú" strom; rozdelenie koreňa zväčšuje výšku.
  - Šírka stromu: je *širší* alebo *o jednu úroveň na vrchole vyšší.*

Vloženie 8* do príklad B+ stromu

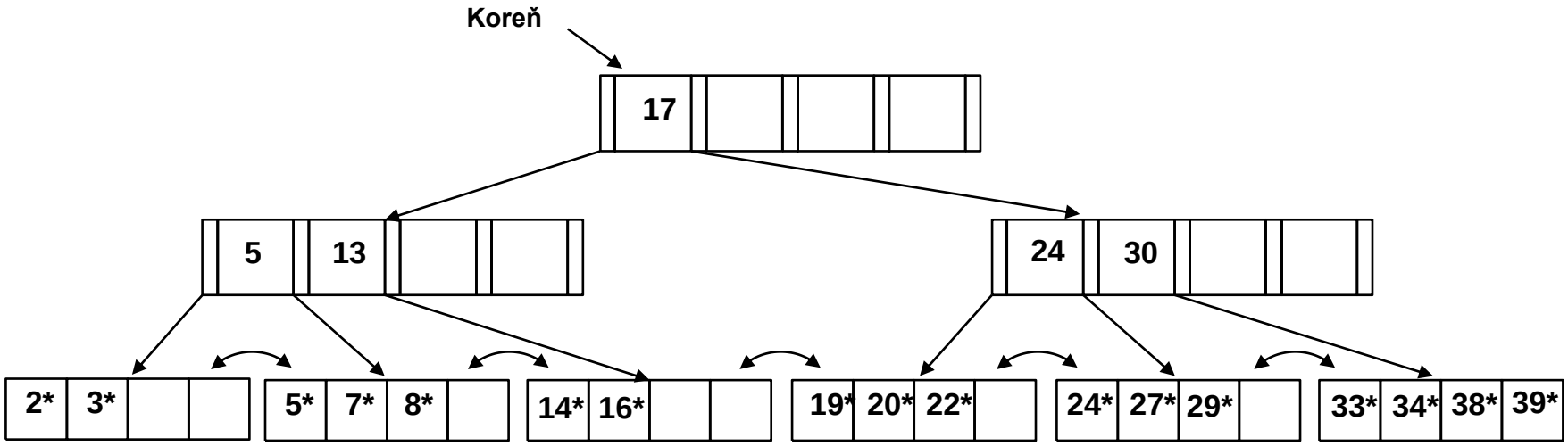
Všimnite si ako je garantovaná minimálna zaplnenosť pri delení listov a indexovej stránky.



Všimnite si rozdiel medzi **skopírovaním nahor** a **presunom nahor**; uistite sa, že rozumiete dôvodom pre tieto akcie.



Príklad B+ stromu po vložení 8*



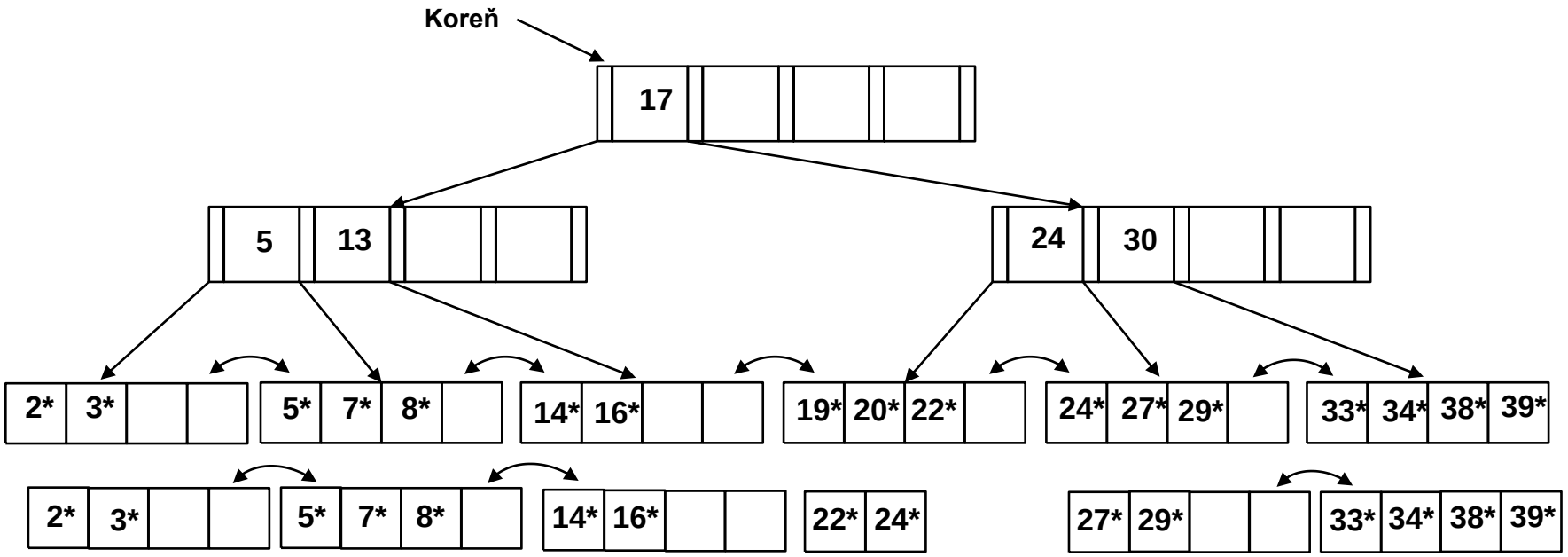
Udržiavanie poradia je kľúčové!

- Všimnite si, že koreň bol rozdelený, čo viedlo k nárastu výšky.
- V tomto príklade sa môžeme vyhnúť rozdeleniu pomocou prerozdelenia položiek; avšak v praxi sa to často nerobí.

Zmazanie dátovej položky z B+ stromu

- Začni v koreni, nájdi list *L*, do ktorého položka patrí.
- Odstráň položku.
  - Ak L je minimálne polovične zaplnený, koniec!
  - Ak má L iba **d-1** položiek,
    - Skús **prerozdeliť**, požičiavajúc od **súrodenca** *(*susedný uzol s rovnakým rodičom ako má uzol L*)*.
    - Ak prerozdelenie zlyhá, **spoj** *L* a súrodenca.
- Ak sa vyskytne spájanie, musí sa zmazať položka (ukazujúca na *L* alebo súrodenca) od rodiča uzla L.
- Spájanie sa môže šíriť ku koreňu, znižujúc výšku.

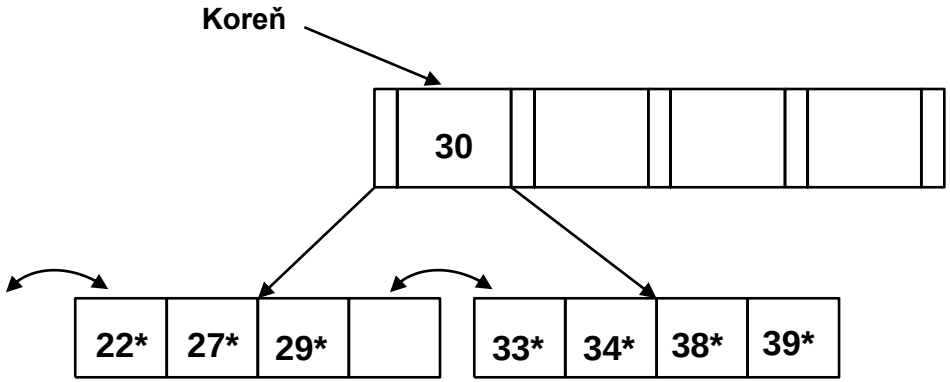
Príklad B+ stromu po vložení 8* a zmazaní 19* aj 20*

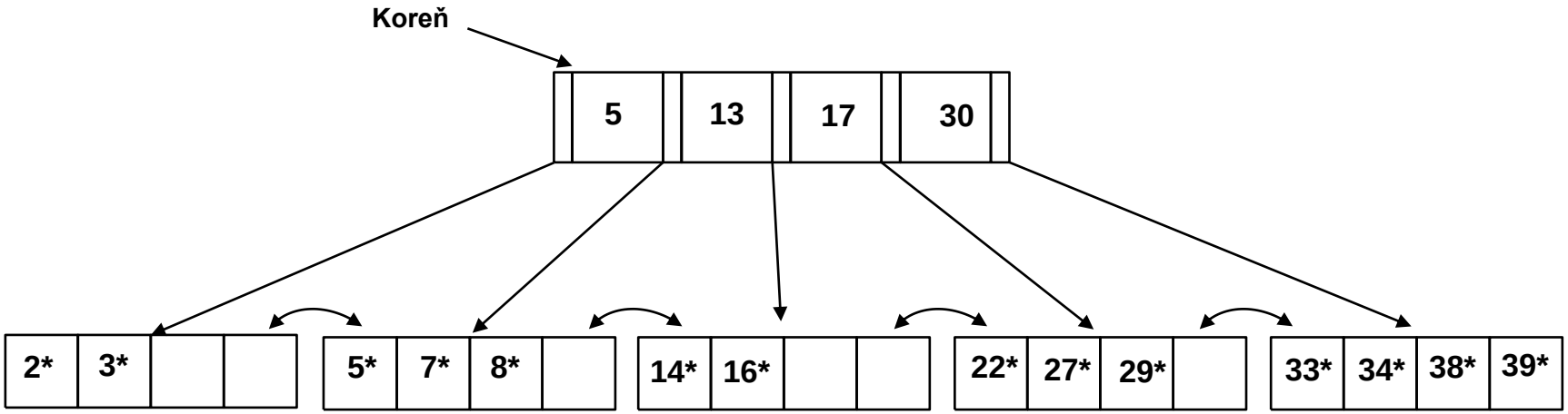


- Odstránenie 19* je ľahké.
- Odstránenie 20* je vykonané s prerozdelením. Všimnite si, ako je stredný kľúč skopírovaný nahor. Nižšie sú znázornené listy po rozdelení.

... a potom zmazanie 24*

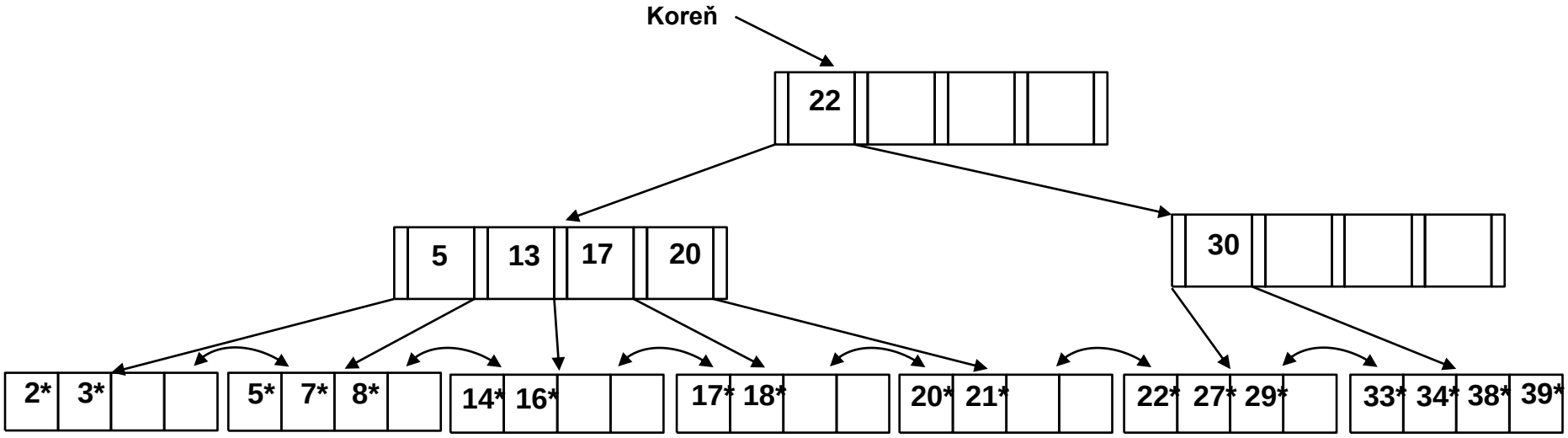
- Je potrebné spájanie.
- Pozoruj **vyhodenie** indexovej položky (vpravo) a **presunutie nadol** indexovej položky (dole).





Príklad nelistového prerozdelenia

- Strom je znázornený nižšie **počas mazania** 24*.
- V porovnaní s predchádzajúcim príkladom je možné prerozdeliť položku z ľavého dieťaťa koreňa do pravého dieťaťa.

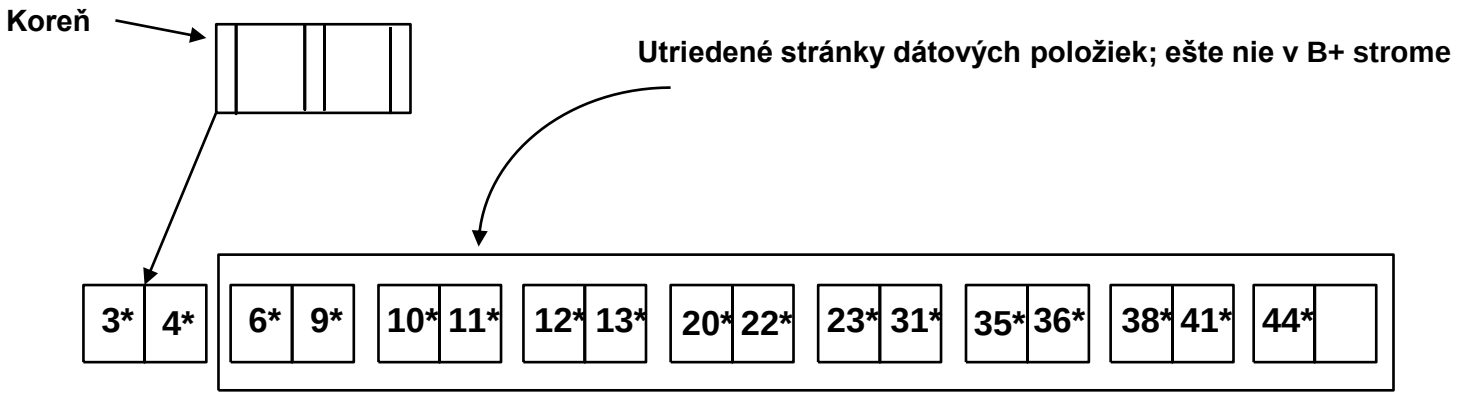


Prefixová kompresia kľúča

- Ak kľúčom je string a má veľa dlhých slov, tak nebude veľa smerníkov a môže narastať výška stromu → dôležitá pre zväčšenie rozvetvenia.
- Pozrie sa na poslednú hodnotu v ľavom súrodencovi a na prvú hodnotu v novom pravom súrodencovi (lebo sme delili uzol) a hore zapíše prefix
- Hodnoty kľúča v indexových položkách slúžia na určenie hraničného kľúča medzi podstromami; je možné ich často komprimovať.
- Vo všeobecnosti počas kompresie sa musí nechať každá indexová položka väčšia ako každá hodnota kľúča (v ľub. podstrome) naľavo.
- Vloženie/odstránenie sa musí vhodne modifikovať.

Naplnenie B+ stromu naraz (Bulk Loading)

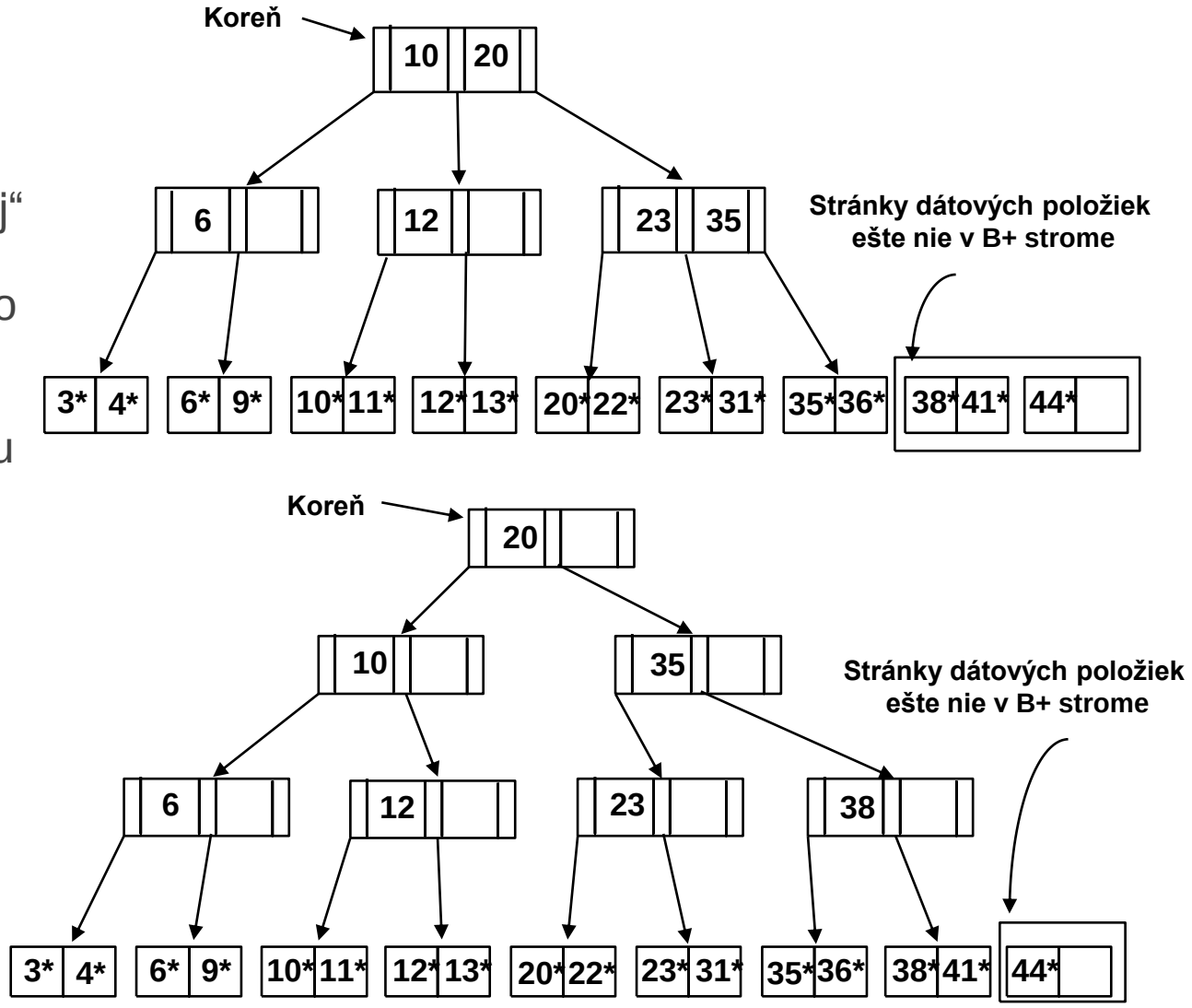
- B+ strom sa vytvorí až po nejakom čase.
- Ak máme veľkú kolekciu záznamov a chceme vytvoriť B+ strom v nejakom poli, tak vytváranie opakovaným vkladaním záznamov je veľmi pomalé.
- **Bulk Loading** môže byť vykonané efektívnejšie.
- Najprv sa utriedia všetky dátové položky priamo v tom dátovom súbore a až potom sa začne vyrábať B+ strom.
- Sekvenčne číta listy zľava doprava a jedným prechodom vybuduje strom vnútorných uzlov.



Bulk Loading

Indexové položky pre listové stránky vždy vložené do „najpravejšej“ indexovej stránky nad úrovňou listu. Ak sa táto zaplní, rozdelí sa. (Delenie môže stúpať „najpravejšou“ cestou ku koreňu.)

Oveľa rýchlejšie než opakované vloženia



Vytváranie B+ stromu

- Možnosť 1: viacnásobné vloženia.
  - Pomalé.
  - Neposkytuje sekvenčné uloženie listov.

- Možnosť 2: **Bulk Loading**
  - Musí sa najskôr usporiadať (nevýhoda)
  - Menej čítaní a zápisov na disk
  - Listy budú uložené sekvenčne (a poprepájané).
  - Je možné kontrolovať faktor zaplnenia na stránkach.
  - Len na prvé vytvorenie stromu

Duplicity kľúčov

- Kľúče nie vždy musia byť unique → duplicita kľúčov → treba upraviť pôvodný algoritmus
- Možnosti:

1. Stránky pretečenia
   - Jedna hodnota nemôže pretiecť do viacerých listov
   - Výhodou je konzistencia vnútorných uzlov
2. „Najľavejšie“ vyhľadávanie
3. Zväčšenie kľúča (K → <K, rid>)
   - Pridáva na koniec súboru
   - Mazať sa nedá, lebo po presúvaní rid už nesedí
4. Zväčšenie kľúča (K → <K, rid-list>)
   - Podobná stránkam pretečenia

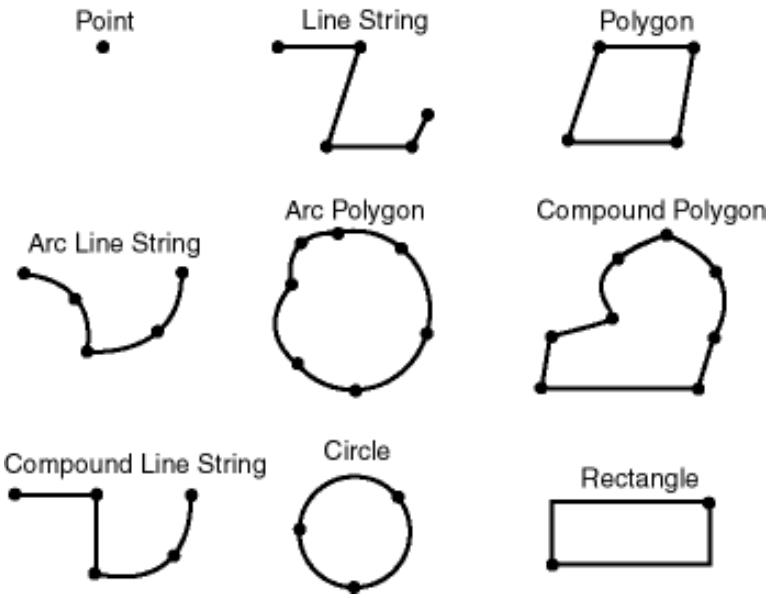
R-Tree Index

Basics, Variations, and Cost

By: Michael Lindemuth & Mark Turner

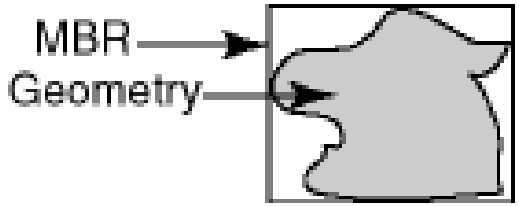
Priestorové (spatial) dáta

- Ľub. geometrické útvary
  - Bod: Mesto, keška
  - Čiara: Železnica, cesta, rieka
  - Polygón: hranica
  - Skupina útvarov: vleky
- Ľub. súradnicový systém
  - Metre
  - Pixely
  - WGS84 (GPS)



Geometic Types in Oracle 11g
Source: Oracle Spatial Developer's Guide 11g, Release 1

R-strom



- Primárne na priestorové dáta, ale aj pre viacstĺpcový index
- Vieme vkladať ľubovoľné geometrické útvary a použiť ľubovoľný súradnicový systém
- Vyrobí sa najmenší ohraničujúci obdĺžnik (Minimum Bounding Rectangle) = **MBR** okolo útvaru
- Súradnice sú kolmé na osy → pre každú os vieme odkiaľ pokiaľ siaha ten objekt
- Len MBR sa indexujú, samotný popis objektu sa nachádza inde
- Sú neklastrované (výnimkou sú niektoré imolementácie)
- Dynamický index

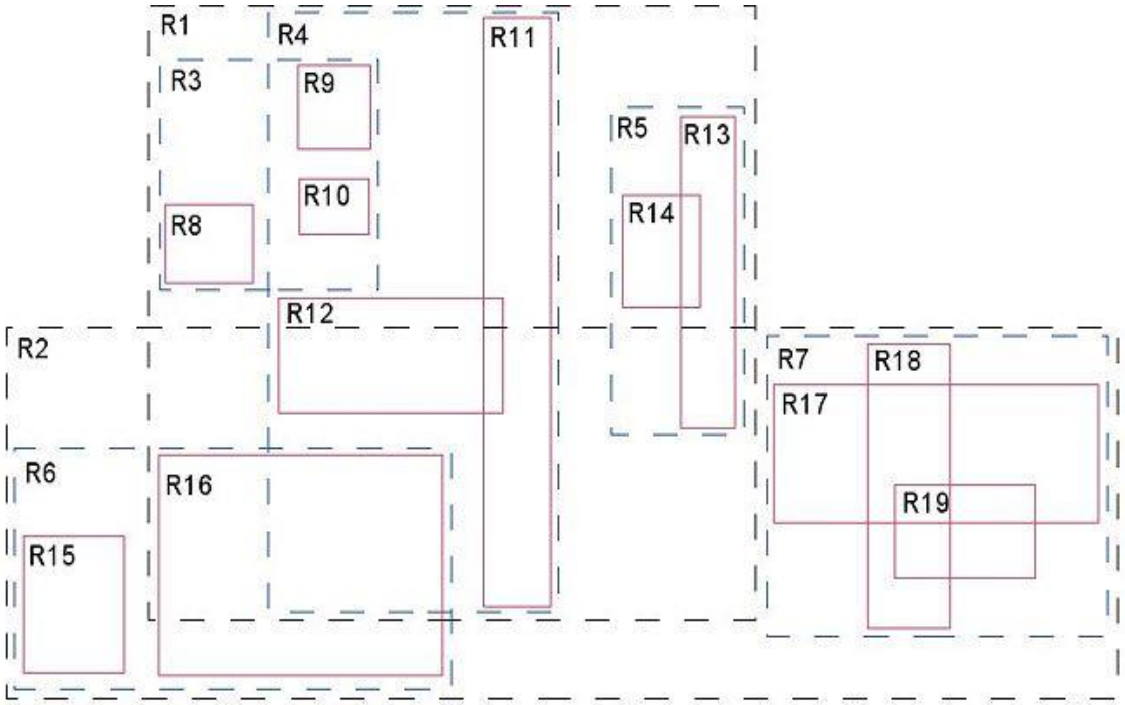
Priestorové dopyty

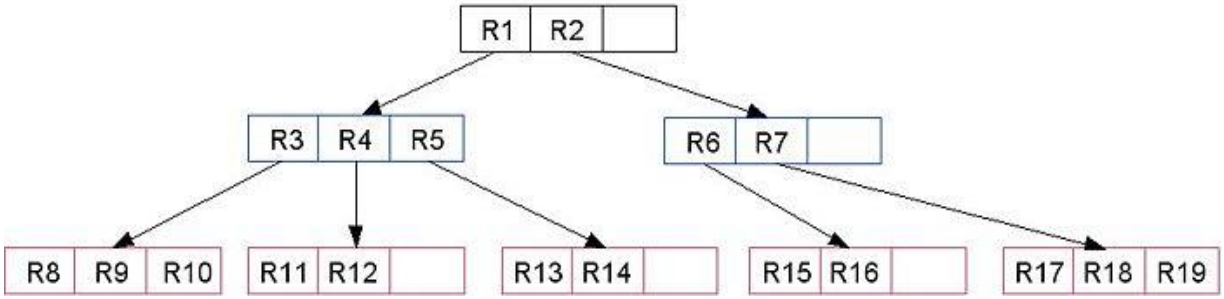
- štandardný insert, delete
- Rozsahový dopyt (Spatial Range)
  - *Nájdi všetky obce do 20km od Košíc*
- Hľadanie najbližších susedov (Nearest Neighbor)
  - *Nájdi najbližšiu reštauráciu od mojej pozície*
- Dopyt priestorového spájania (Spatial Join Queries)
  - *Nájdi hotely v ktoré sú do 1km od nejakého parku*
- Geometrické operácie
  - Equal(), Disjoint(), Intersect(), Touch(), Cross(), Within(), Contains(), Overlap(), Distance(), Buffer(), ConvexHull(), Intersection(), Union(), Difference(), SymmDiff(),…

Štruktúra R-stromu

- Dátová položka v listovom uzle: <I, smerník na objekt>
  - $I = (I_0, I_1, \ldots I_n)$
    - n = počet dimenzií priestoru
    - Každé I je dvojica [a,b] reprezentujúca rozsah MBR geometrického útvaru v danej dimenzii (a alebo b môžu byť nekonečno)
  - smerník na objekt ukazuje na záznam kde je uložený daný geometrický útvar ohraničený týmto MBR
- Indexové položka vo vnútornom uzle: <I, smerník na dieťa uzla>
  - I predstavuje MBR dieťaťa daného uzla
  - MBR koreňa je celý priestor
  - Môžu sa prekrývať

Štruktúra R-stromu





Šesť vlastností R-stromu

Dané:

- M je maximálny počet záznamov v uzle
- Parameter m ≤ M/2 predstavuje minimálny počet záznamov v uzle

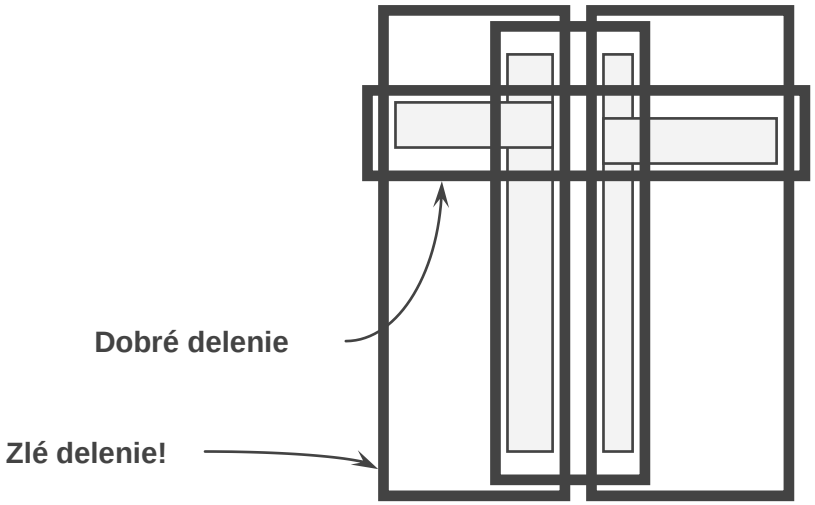
1. Každý listový uzol okrem koreňa musí obsahovať od m do M dátových položiek.
2. Pre každú dátovú položku platí, že <I, smerník na objekt> predstavuje najmenší obdĺžnik rovnobežný s osami priestoru obsahujúci indexovaný objekt
3. Každý vnútorný uzol okrem koreňa musí obsahovať od m do M indexových položiek.
4. Pre každú indexovú položku platí, že <I, smerník na dieťa uzla> vo vnútornom uzle predstavuje najmenší obdĺžnik rovnobežný s osami priestoru obsahujúci všetky MBR tohto dieťaťa
5. Koreň obsahuje aspoň dva záznamy pokiaľ nie je list
6. Všetky listy sú rovnako vzdialené od koreňa v strome.

Vkladanie

- Veľmi podobný vloženiu do B+stromu
- Začíname v koreni
- Opakuj pokiaľ nedôjdeš do listu:
    - Vyber dieťa, do ktorého úplne zapasuješ.
    - Ak také neexistuje, vyber také, ktoré potrebuje najmenšie zväčšenie objemu aby poňalo vkladaný objekt (jeho MBR) a zväčši MBR tohto dieťaťa.
- Ak dojde do listu a je voľné miesto vlož.
- Inak opakuje pokiaľ je potrebné:
    - Rozdeľ záznamy uzla do dvoch uzlov
    - Uprav MBR tohto uzla v rodičovi
    - Vlož indexový záznam do rodiča pre nový uzol

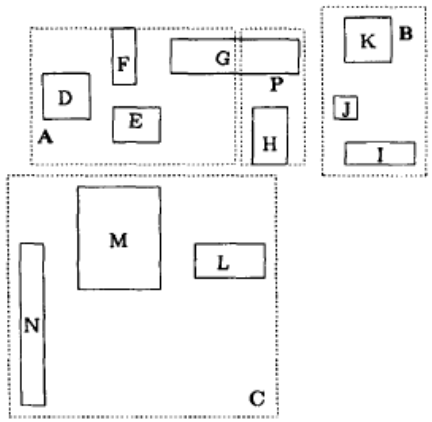
Delenie záznamov

- Dobré delenie:
  - Čo najmenšie prieniky
  - Čo najmenší objem
  - Čo najštvorcovejší obdĺžnik

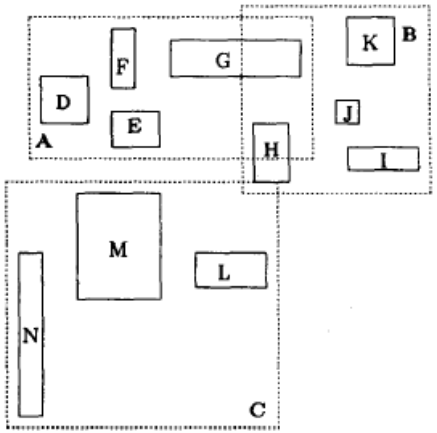


Varianty R-stromu

- R+strom
  - Delí záznamy tak, aby sa vnútorné uzly neprekrývali
    - Iba jedna cesta pri hľadaní
    - Smerníky na rovnaké deti môžu byť uložené vo viacerých rodičoch
    - Nie vždy sa dá zostrojiť (viď R++strom (Gurský, Šumák))



- R*strom
  - Hľadá čo najštvorcovejšie uzly
  - Namiesto delenia vyskúša záznamy z preplneného uzla znova vložiť
  - Insert trvá dlhšie
  - V praxi veľmi rýchly na vyhľadávanie



Mazanie

- Tiež podobné s B+stromami
  - Nájdi uzol z ktorého treba záznam zmazať
  - Ak uzol obsahuje málo záznamov realokuj ich
    - Vynáraj sa až ku koreňu a odstraňuj uzly v ktorých je málo záznamov, odstraňuj indexové záznamy ukazujúce na zmazaný uzol, a uprav ich MBR
    - Opätovne vlož dátové položky z odstráneného listu.

Hľadanie k najbližších susedov

- Dve možnosti
  - Vnáranie k najlepšiemu
  - S orezávaním
- Vnáranie k najlepšiemu (jeden prioritný rad)
  - Rátame najmenšiu vzdialenosť od bodu dopytu
  - Ak najbližší objekt je uzol, vezmeme všetky jeho deti a vložíme do prioritného radu
  - Ak najbližší objekt je dátová položka, tak je to ďalší najbližší sused
- S orezávaním (dva prioritné rady pre uzly a pre body (tých je len k))
  - Rátajú sa dve vzdialenosti pre každý uzol
    - Najmenšia vzdialenosť od bodu dopytu (utriedené podľa tejto vzdialenosti)
    - Minmax vzdialenosť (najmenšia z najvzdialenejších vzdialeností pre každú stenu)
  - Končíme, keď už nie sú uzly, ktoré sú bližšie, ako aktuálna hranica orezania

Rozsahové dopyty

- Ak aktuálny uzol nie je strom, zisti, ktoré deti majú prienik z obdĺžnikom dopytu
  - Rekurzívne opakuj pre všetky tieto deti
  - Ak uzol je list over prienik so všetkými MBR a v prípade úspechu aj prienik s geometrickým objektom

Implementácie

- PostgreSQL (PostGIS extensions), MySQL, Oracle používajú R-stromy na priestorové indexovanie
- Použitie: CAD/CAM software, design elektronických obvodov, geografické informačné systémy
- Ďalšie alternatívy
  - UB-stromy - prevod viacdimenzionálnej informácie do jedného rozmeru a indexovanie cez B+strom (Hilbertove krivky), hľadanie najbližších objektov je pomalé
  - K-d stromy - Nie je vyvážený
  - Grid files - väčší ako R-strom, nevyvážený
  - VA-súbor - aproximácie objektov cez bitové mapy, dopyty vždy prezrú všetky aproximácie a až potom idú pozerať k kandidátskym objektov, efektívne pre vysoké rozmery