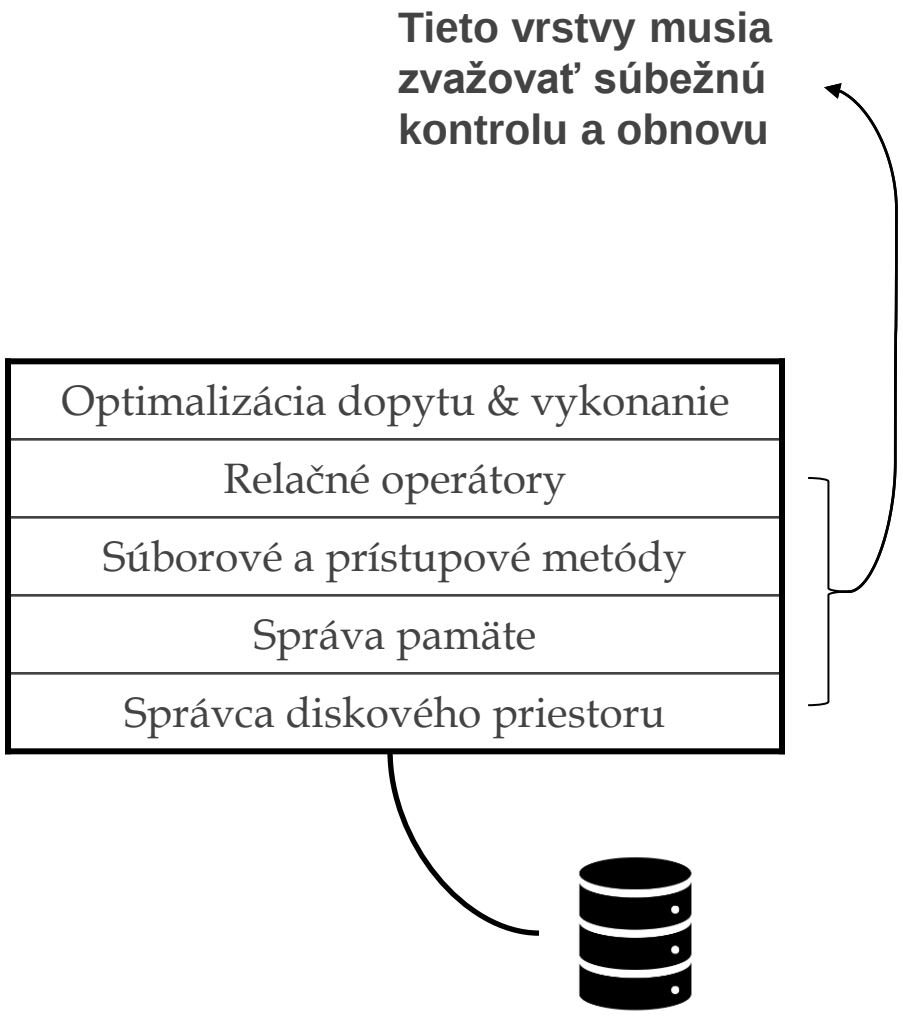
Štruktúra DBMS

Kapitola 1.8

Štruktúra DBMS

- Typický DBMS má vrstevnatú štruktúru.
- Jedna z niekoľkých možných architektúr.
- Každý systém má vlastné variácie.



Štruktúra DBMS

- Optimalizácia dopytu
  - Používa informácie o tom, ako sa údaje ukladajú, aby vytvorili efektívny plán vykonávania
- Plán vykonávania
  - Návrh ako sa bude vykonávať dopyt
  - Väčšinou je reprezentovaný ako strom relačných operátorov
- Súborové a prístupové metódy
  - Sleduje stránky v súboroch a organizuje informácie vrámci stránky
- Správca pamäte
  - Stará sa o prenos stránok z pamäte do hlavnej pamäte
- Správca diskového priestoru
  - Manažuje priestor na disku, kde sú uložené dáta

Ukladanie dát: Disky a súbory

Kapitola 7

Pamäť

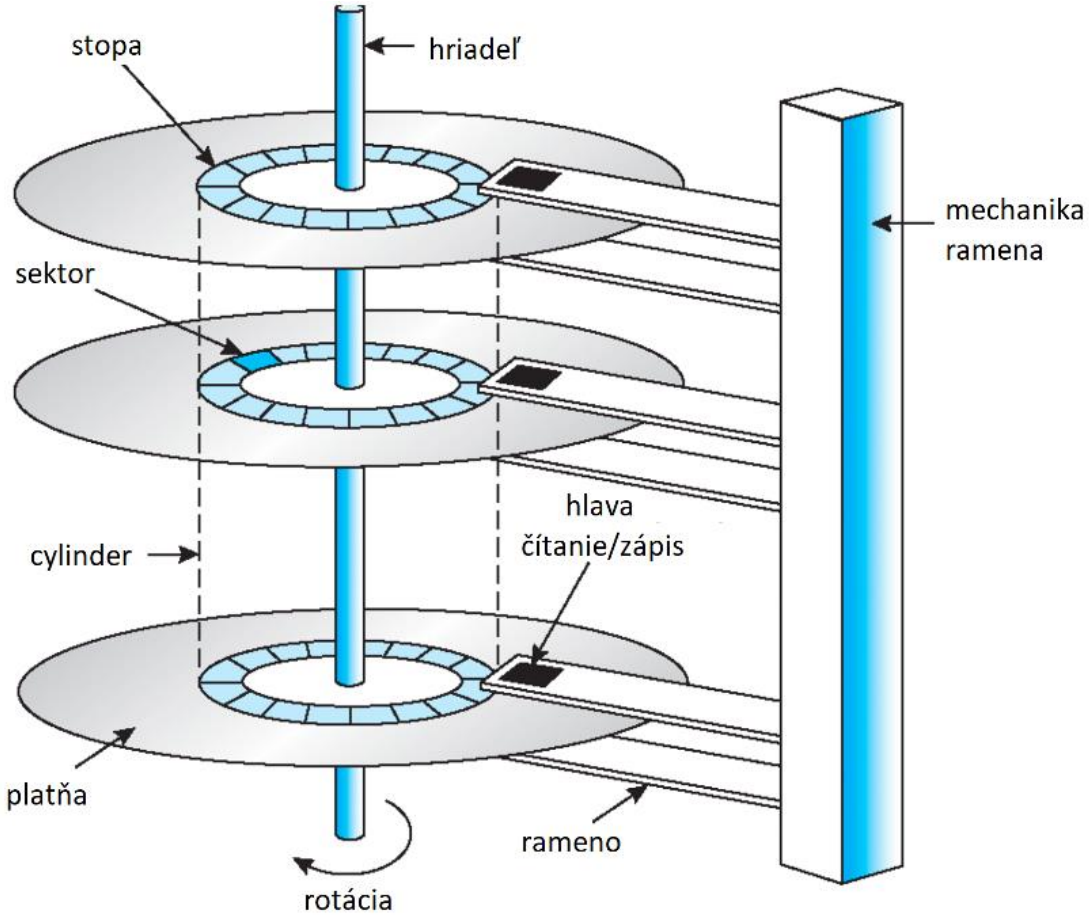
- Hierarchia:
  - Primárna pamäť - cache a hlavná pamäť (RAM), najrýchlejšia
  - Sekundárna pamäť - disky, pomalšia
  - Terciárna pamäť - pásky, najpomalšia

- Prečo sa používajú pomalšie pamäte?
  - Cena a množstvo dát na ukladanie (veľa dát → veľa RAMky → vyššia cena ako pri sekundárnej alebo terciárnej pamäti)
  - Problémy s 32-bitovými OS – len 2^{32} bajtmi (~4GB) vie priamo pracovať
  - RAM je nestála (volatile)

Disky a súbory

- DBMS ukladá informácie primárne na harddisky.
- Toto má významné dôsledky pre návrh DBMS
- **- Čítanie (READ)**: prenos dát z disku do hlavnej pamäte (RAM).
- **- Záspis (WRITE)**: prenos dát z RAM na disk.
- Obe operácie sú časovo náročné, a musíme s nimi pracovať opatrne

Súčasti disku



Disky

- Sekundárne úložné zariadenie.
- Dáta sú ukladané a sťahované v jednotkách nazývaných **diskové bloky**.
- Hlavné výhody oproti páskam: **priamy prístup (direct access)** vs. **sekvenčný (sequential access)**.
- Čas na načítanie bloku z disku sa líši v závislosti od umiestnenia na disku → stratégické umiestnenie dát na disku má veľký vplyv na výkon DBMS
- Komponenty, ktoré ovplyvňujú čas:
  - **prístup k cylindru** (čas, kým sa posunie hlava na cylinder, kde je požadovaný blok)
  - **otočné oneskorenie** (čas, kým sa blok dostane pod hlavu)
  - **doba prenosu** (čas na prenos dát na/z diskovej plochy)

Usporiadanie stránok na disku

- Ak nejaké dáta sú často používané spolu, najlepšie je ukladať ich (postupne):
  - na rovnaký blok
  - na bloky na rovnakej stope
  - na bloky na rovnakom cylindri
  - na bloky na nasledujúcom cylindri
- Súbor by mal byť usporiadaný v blokoch idúcich za sebou, aby sa minimalizovalo prístupové a rotačné omeškanie.
- Možnosť predbežného načítavania (**pre-fetching**) niekoľkých stránok v tom istom čase tiež má vplyv na výkon DBMS

RAID (Redundant Array of Independent Disks)

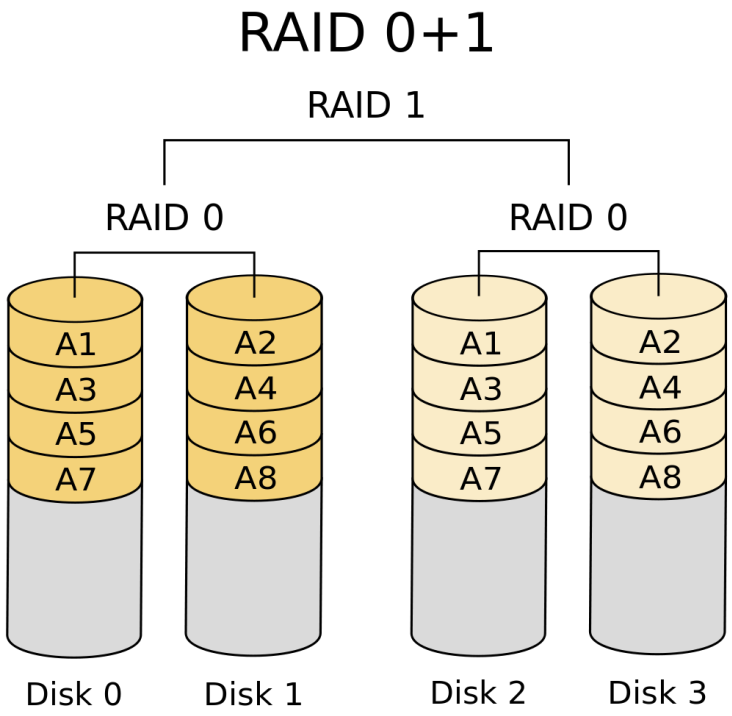
- Je to diskové pole, ktoré slúži na zvýšenie výkonu a spoľahlivosti systému
- Skladá sa z niekoľkých diskov, ktoré tvoria abstrakciu jedného veľkého disku.
- 2 hlavné techniky:
    - **Data striping** - dáta sú rozdelené, veľkosť oddielu sa nazýva sekvenčná jednotka (striping unit), partície sú distribuované cez niekoľko diskov.
    - **Redundancy** - dáta sú uložené na viacero diskov, čo umožňuje rekonštrukciu dát v prípade zlyhania disku
- Spoľahlivosť
    - Chybovosť 1 disku za 50 000 hodín = cca 5,7 roka
    - RAID o 100 diskoch za 500 hodín = cca 21 dní

RAID úrovne

- Úroveň 0: Žiadna redundancia (Stripping)
  - Paralelné čítanie (N x rýchlejšie oproti 1 disku – pre N diskov)
  - Paralelný zápis (N x rýchlejšie oproti 1 disku)
- Úroveň 1: Zrkadlenie (Mirrored)
  - Každý disk má rovnaký zrkadlový obraz
  - Paralelné čítanie (N x rýchlejšie oproti 1 disku)
  - Zápis – rýchlosť ako pre jeden disk (to isté ide na všetky disky)

RAID úrovne

- Úroveň 0+1: Rozdelenie a zrkadlenie (Striping and Mirroring)
  - Dvojička navzájom zrkadliacich diskov je považovaná za jeden disk pre RAID 0
  - Paralelné čítanie – rovnako rýchle ako pri RAID 0 a RAID 1
  - Zápis zahŕňa dva disky – polovičná rýchlosť oproti RAID 0



RAID úrovne

- Úroveň 3: Bit-Interleaved Parity (bitové delenie)
  - Sekvenčná jednotka: 1 bit. Jeden kontrolný disk.
  - Každé čítanie a zápis zahŕňa všetky disky; diskové pole dokáže spracovať len jednu požiadavku v danom čase.

Príklad:

1 0 1 0 1 1 0 1

1.	Disk	1 1
2.	Disk	0 1
3.	Disk	1 0
4.	Disk	0 1
Check disk		0 1 (XOR funkcia)

2-násobné
zrýchlenie

RAID úrovne

- Úroveň 4: Block-Interleaved Parity (blokové delenie)
  - Sekvenčná jednotka : 1 diskový blok. Jeden kontrolný disk.
  - Paralelné čítanie je možné pre malé požiadavky, veľké požiadavky môžu využiť plnú šírku pásma
  - Zápisy zahŕňajú upravený blok a kontrolu disku

Príklad:

Paralelné čítanie =>
4-násobné zrýchlenie

1.	Disk	1. blok
2.	Disk	2. blok
3.	Disk	3. blok
4.	Disk	4. blok
Check disk		(XOR funkcia)

RAID úrovne

- Úroveň 5: Block-Interleaved Distributed Parity (blokové delenie s distribúciou parity)
  - Rovnaká ako RAID úroveň 4, ale paritné bloky sú rozdelené na všetky disky.

Príklad:

Check disk (XOR funkcia) – stále iný disk (označenie X)

1.	Disk	D D D
2.	Disk	D D D
3.	Disk	D D X
4.	Disk	D X D
5.	Disk	*X* D D

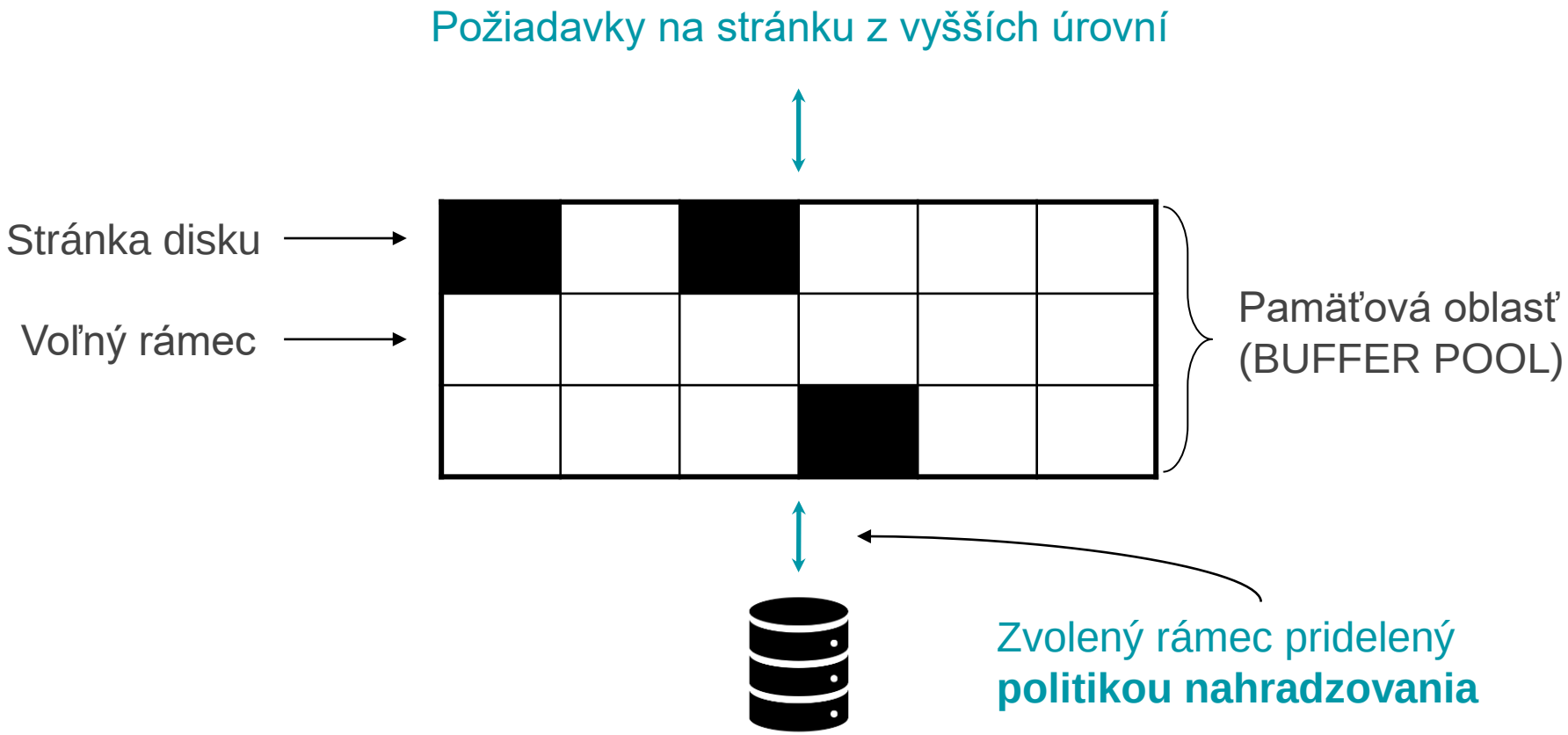
RAID úrovne

- Úroveň 6:
  - P + Q redundantná schéma
  - Reed – Solomonové kódovanie
  - 2 kontrolné bity pre každé 4 bity dát
  - Zotavenie z chyby 2 diskov

Správa miesta na disku

- Najnižšia vrstva softvéru DBMS.
- Vyššie úrovne vyžiadajú túto vrstvu k:
  - prideľovaniu/uvoľneniu stránky
  - čítaniu/zápisu stránky
- Žiadosť o pridelenie stránok musí byť splnená prideľovaním stránok postupne na disku!
- Vyššie úrovne nemusia vedieť ako sa to robí, alebo ako je voľné miesto riadené.

Manažment pamäte v DBMS



Dáta musia byť v RAM aby DBMS mohol s nimi pracovať!

Žiadosť o stránku

- Dve dôležité premenné pre každý rámec v BP
  - **pin_count** - koľkokrát bola stránka požadovaná a neuvoľnená v danom rámci
  - **dirty_flag** – boolean hodnota indikujúca, či bola stránka zmenená
- Ak požadovaná stránka je v buffer pool, zvýši pin_count rámca o 1
- Ak požadovaná stránka nie je v buffer poole:
  - Vyberie rámec na **výmenu**, na základe politiky nahradzovania a zvýši pin_count o 1
  - Ak je rámec zmenený (dirty flag zapnutý), zapíše ho na disk
  - Načíta požadovanú stránku do zvoleného rámca
- Vráti adresu rámca, ktorá obsahuje požadovanú stránku

* Ak žiadosti možno predvídať (napr. sekvenčné prehľadávania) stránky môžu byť **načítané** spolu s niekoľkými ďalšími stránkami naraz

Politiky nahradzovania v pamäti

- Typy:
  - LRU (Least recently used) - vyberie sa rámec s pin_count 0
  - Clock – pridá sa premenná **current**, ktorá môže nadobudnút hodnoty od 1 po N (počet rámcov v BP) a boolean premenná **refferenced**, ktorá indikuje, či je pin_count 0
  - FIFO (First in first out)
  - MRU (Most recently used)
- **Sekvenčné zaplavenie (Sequential flooding)**
  - situácia spôsobená LRU a opakovaným sekvenčným prehľadávaním.
  - počet buffer rámcov < počet stránok v súbore znamená, že každá požiadavka na stránku spôsobí I/O. V tomto prípade by bolo lepšie použiť MRU.

Pamäťový manažment v DBMS vs. OS

- OS sa stará o miesto na disku a správu zásobníka. Prečo nenechať OS manažovať tieto úlohy?
- Oboch cieľom je poskytnúť prístup k viac dátam ako sa zmestí do RAM
- Výhody DBMS:
  - Vie predikovať poradie v akom budú stránky požadované → prefetching
  - Väčšia kontrola, keď je stránka zapísaná na disk

Súbory a indexy

- Ako sú použité stránky na ukladanie záznamov? Ako môžu byť organizované do súborov alebo logických kolekcií?
- Vyššie úrovne DBMS vidia stránky ako množinu záznamov, nezaujíma ich reprezentácia a spôsob ukladania.
- Súbor záznamov je množina záznamov, ktoré sa môžu nachádzať aj na rôznych stránkach.
- Záznam je identifikovaný pomocou dvojice tzv. **record id** (rid): <id stránky, čislo slotu>
- Možnosti pre štruktúru súborov:
  - Halda
  - Indexy

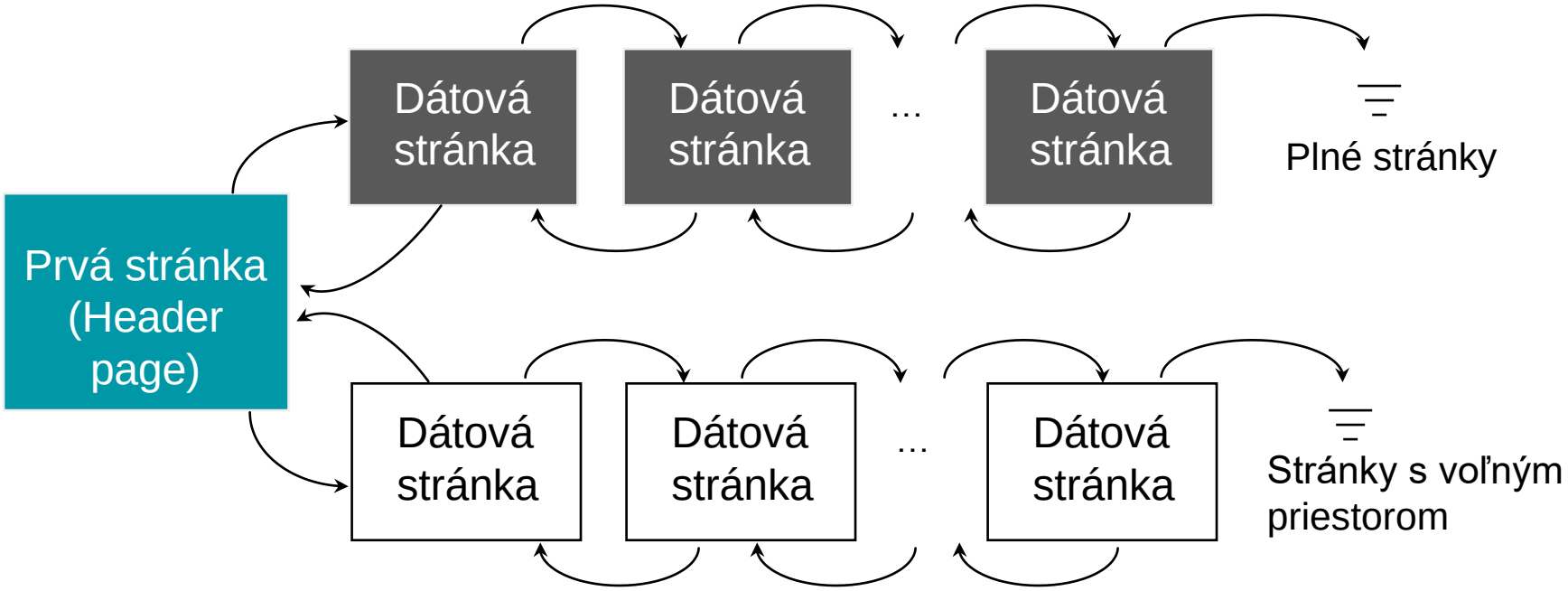
Haldové súbory

- Najjednoduchšia súborová štruktúra, ktorá obsahuje záznamy bez konkrétneho poradia.
- Každá stránka je rovnakej veľkosti, kde dáta nie sú zoradené
- Podporované operácie:
  - Vytvorenie a odstránenie súboru
  - Vloženie a zmazanie záznamu s daným rid → musíme sledovať všetky stránky, ktoré majú voľné miesto
  - Prehľadanie všetkých záznamov → musíme sledovať všetky stránky v súbore

- Dve možnosti ako môžeme tieto informácie sledovať:
  - Spájaný zoznam stránok
  - Adresár stránok

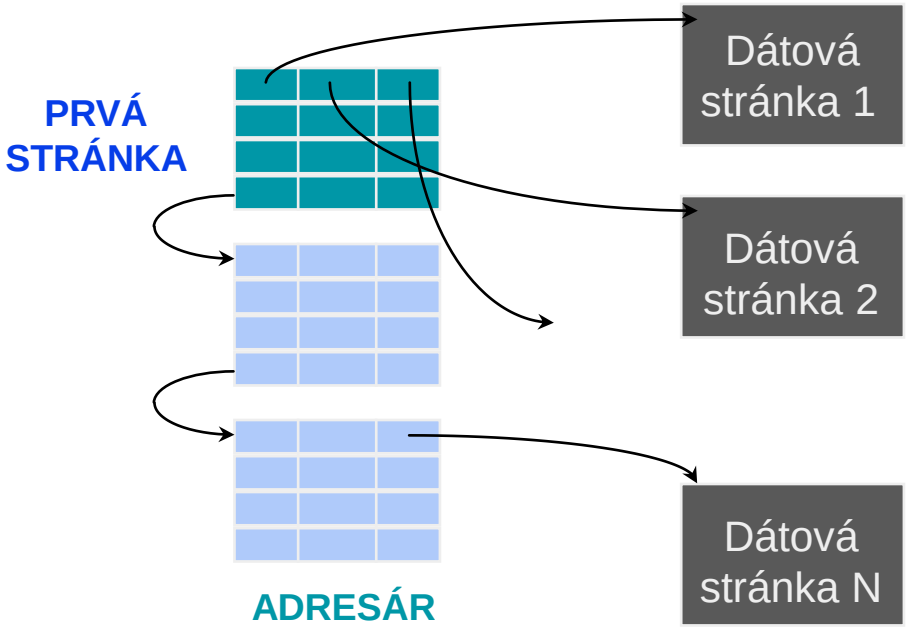
- V oboch prípadoch sú potrebné dva ukazovatele

Haldový súbor ako spájaný zoznam



Hlavnou nevýhodou je, že veľa stránok môže mať niekoľko voľných bajtov

Haldavý súbor ako adresár stránok



- Pri odkaze na stránku môže byť informácia o počte voľných bytov na stránke, alebo len 1 bit o voľnom mieste.
- Adresár je súhrn stránok, jeho implementácia ako spojený zoznam (ako je aj na obrázku) je len jedna z možností.

Indexy

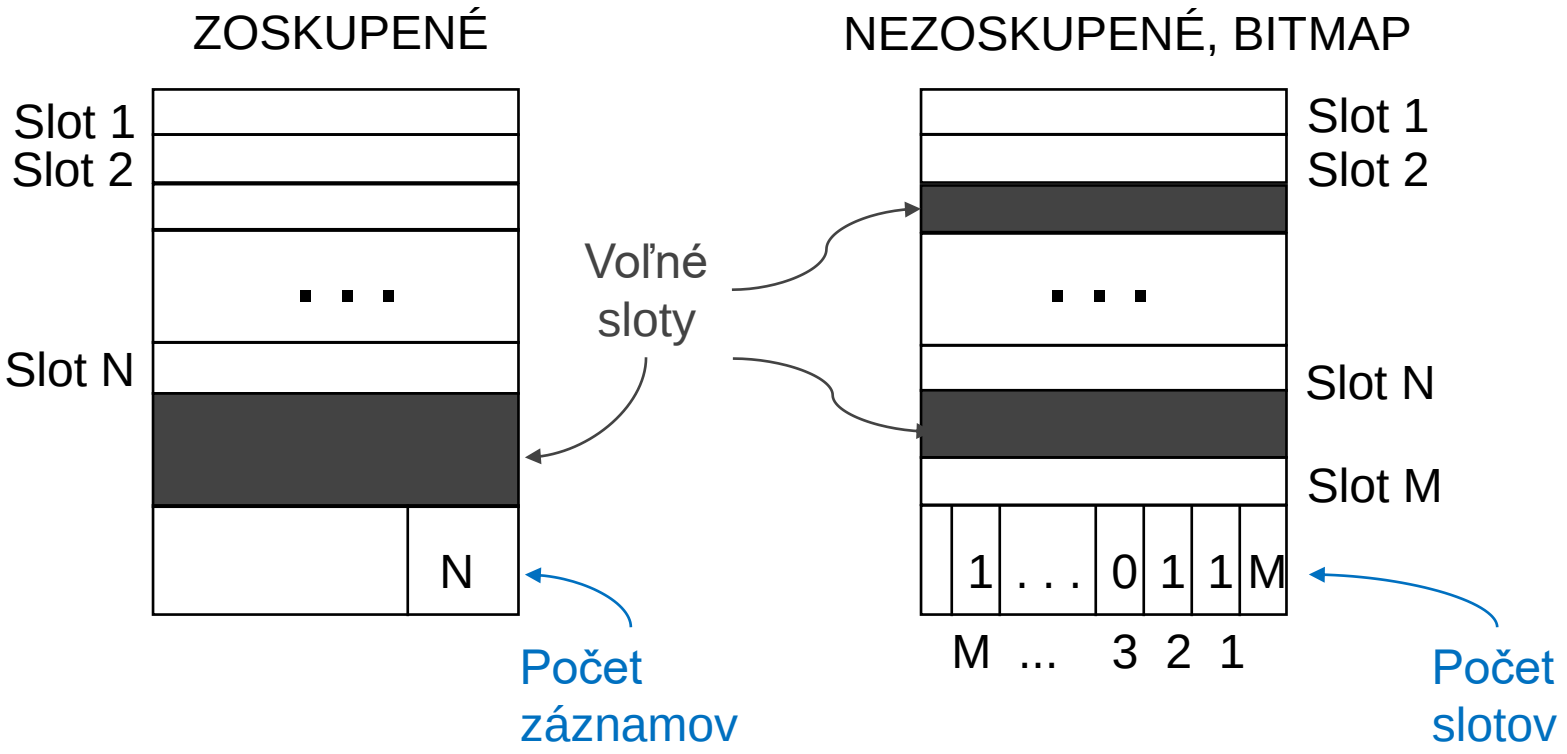
- Hlavnou výhodou je možnosť vyhľadávania rid záznamov podľa nejakej podmienky.
- Každý index obsahuje vyhľadávací kľúč
- Vyhľadávací kľúč - ľubovoľná množina polí zo súboru záznamov, pre ktoré vytvárame indexy
- Ak sa často vyhľadáva podľa veku, tak môžeme vytvorit index nad atribútom vek: <vek, rid>
- Index je množina *dátových položiek* (**k***), pomocou ktorej môžeme efektívne nájsť dátové záznamy s vyhľadávacím kľúča **k**.
    - **Daný záznam k*, vieme nájsť skoro na jeden diskový prístup. (detaily neskôr …)**
- Existuje mnoho spôsobov ako môžu byť indexy organizované (B+ strom, hash,...)

Stránkové a záznamové formáty

- Stránka je ako množina "šuflíkov" (slot), kde každý obsahuje jeden záznam.
- Možnosti manažovania slotov v stránke závisí od toho, či ide o záznamy s:
  - Pevnou dĺžkou - stačí riešiť iba to, že ktoré sloty sú prázdne, aby sme vedeli kam vložiť záznam
  - Premenlivou dĺžkou - treba riešiť aj to, aby ten prázdny slot bol dostatočne veľký

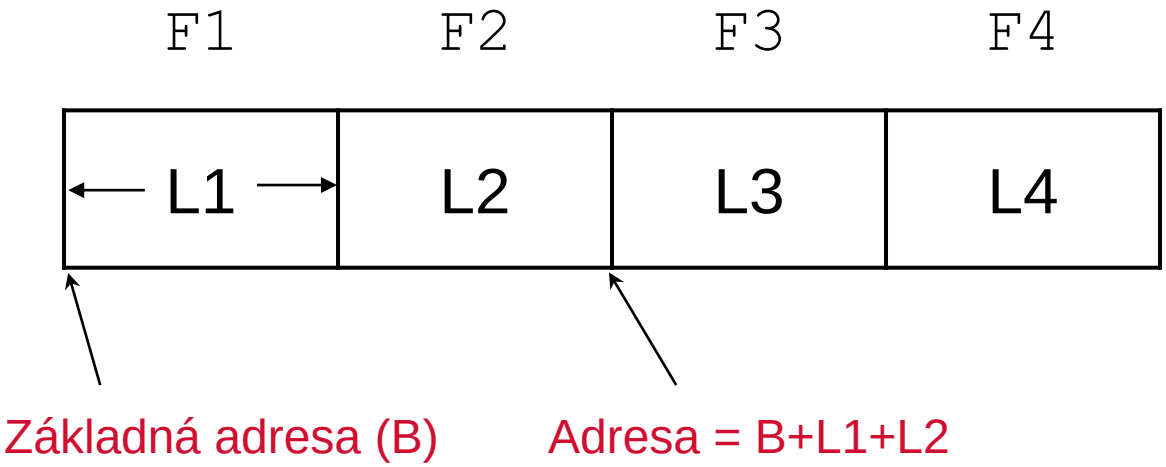
- Možnosti manažovania polí v zázname závisí od toho, či ide o záznamy s:
  - Pevnou dĺžkou → pevná dlžka polí a teda aj počet polí
  - Premenlivou dĺžkou → rovnaký počet polí, záznamy sú premenlivej dĺžky kvôli tomu, že niektoré z polí je premenlivej dĺžky

Stránkové formáty: Pevná dĺžka

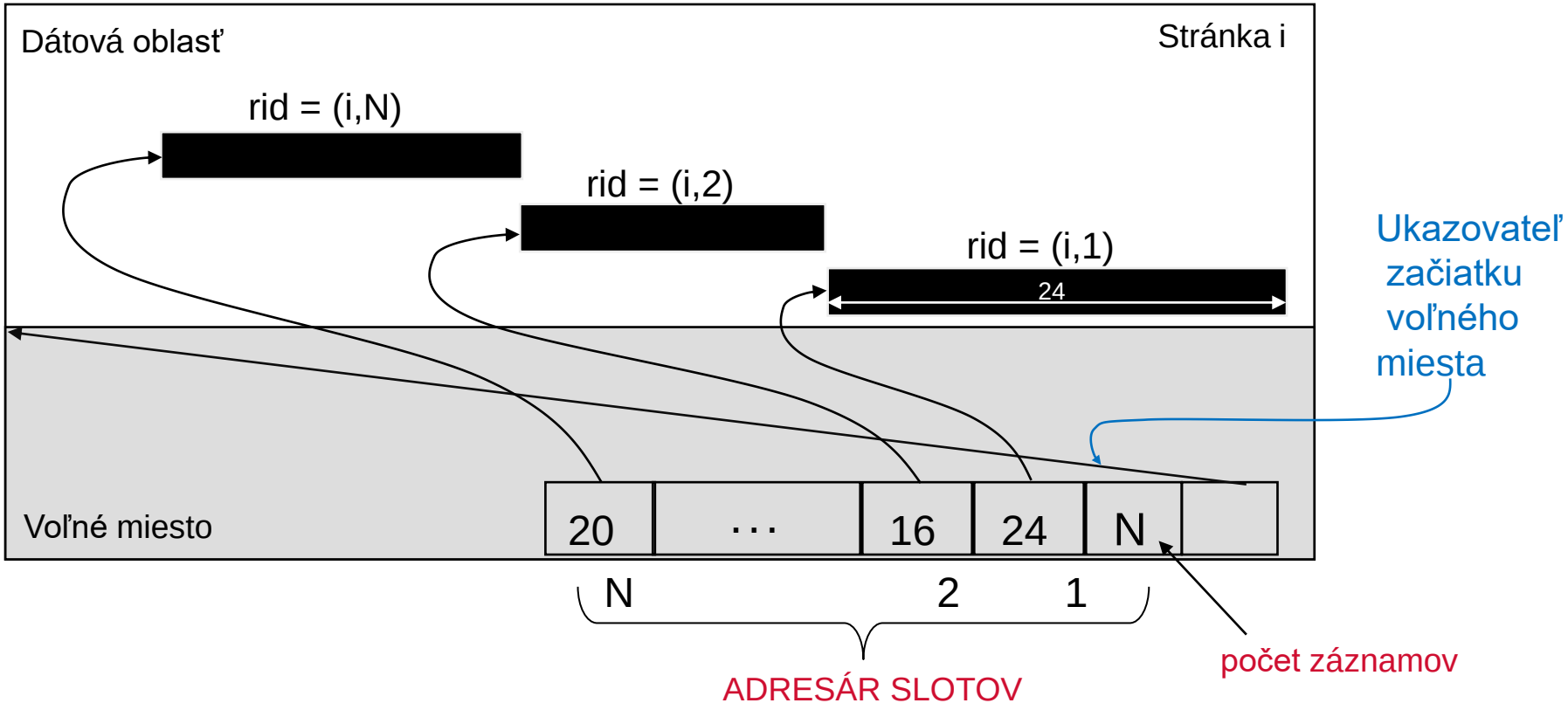


Záznamové formáty: pevná dĺžka

F_i = i-té pole
L_i = dĺžka i-tého poľa



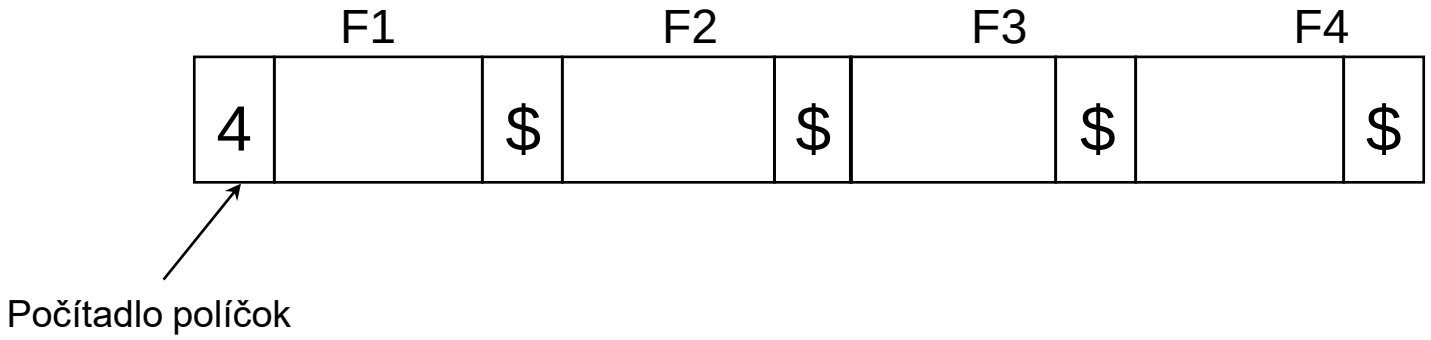
Stránkové formáty: Premenlivá dĺžka



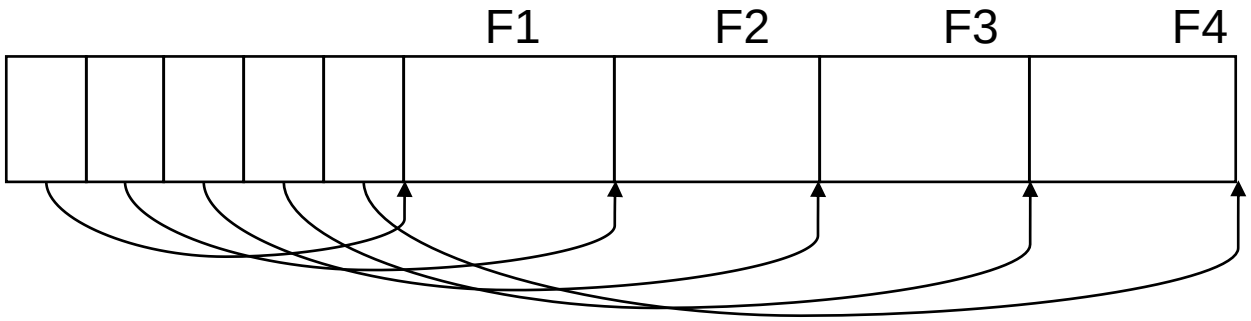
- Pre každú stránku sa vytvorí adresár slotov s informáciami <offset záznamu, dĺžka záznamu>
- Môže pohybovať záznamami na stránke bez zmeny rid, iba offset sa zmení
- Použitelné aj pre fixnú dĺžku záznamov, keď ich často potrebujeme presúvať

Záznamové formáty: premenlivá dĺžka

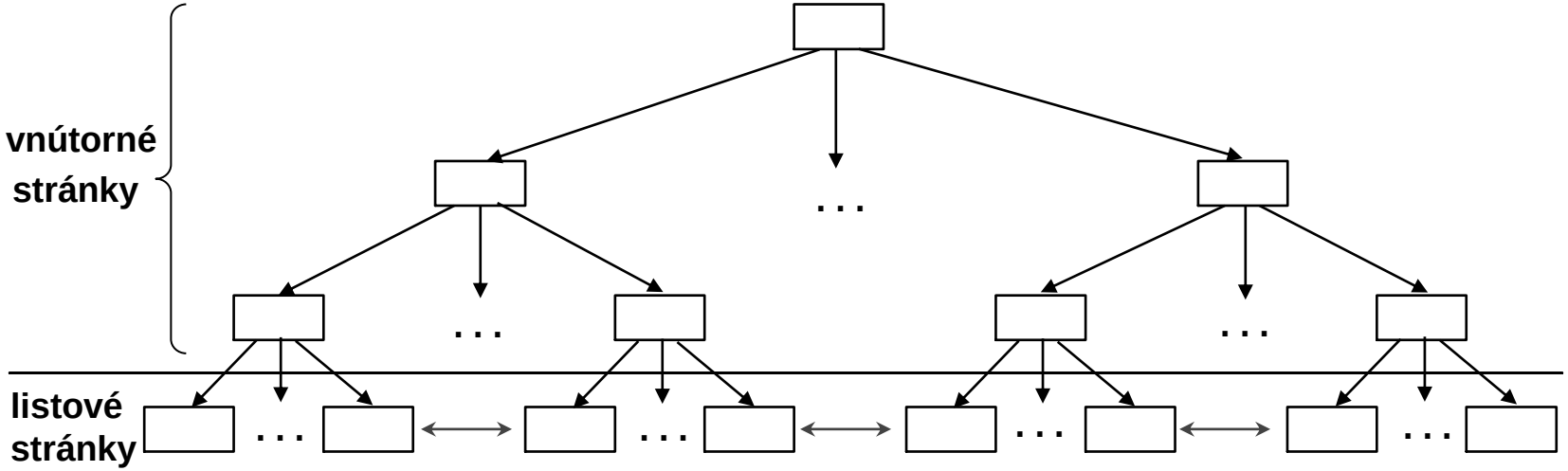
- Políčka oddelené špeciálnymi symbolmi



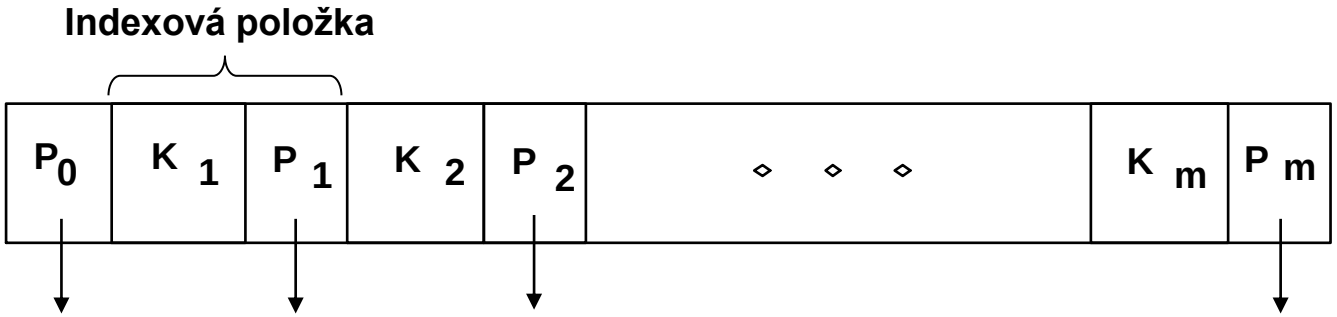
- Offsety začiatkov políčok



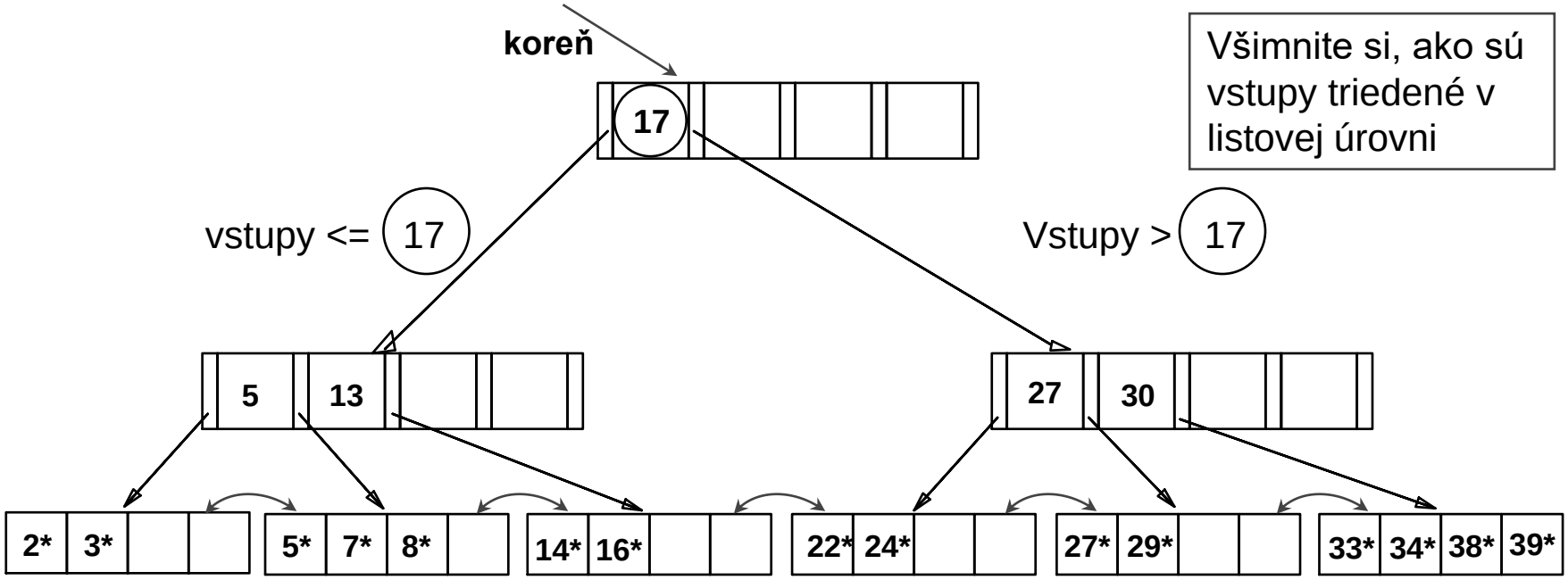
Index B+ strom



- Listové stránky obsahujú **dátové položky** a sú zreťazené
- Vnútorné stránky obsahujú **indexové položky** používané len na priamy prístup:



Príklad B+ Strom



- najdi 28*? 29*? všetky > 15* a < 30*
- vlož/zmaž: nájdi vstup v liste, potom ho zmeň. Niekedy treba nastaviť rodiča.
    - A zmena občas zmení počet stránok v strome či výšku stromu.

Indexy na základe hashu

- Dobré na vyhľadávanie pre rovnosť s kľúčom.
- Index je množina **oblastí (buckets)**.
  - oblasť = **primárna stránka** plus žiadna alebo viac **stránok pretečenia**.
  - oblasti obsahujú dátové položky alebo rovno dátové záznamy (**k***).
- *Hash funkcia* **h**: **h(*r*)** = číslo oblasti v ktorej je dátová položka alebo dátový záznam s hodnotou kľúča *r.*
  - *Žiadna potreba "indexových záznamov".*

Alternatívy pre vstup dát k* v indexe

- V dátovom vstupe k* môžeme uložiť:
  - Dátový záznam s kľúčovou hodnotou **k,** alebo
  - <**k**, rid dátových záznamov s hodnotou vyhľadávacieho kľúča **k**>, alebo
  - <**k**, zoznam rid dátových záznamov s vyhľadávacím kľúčom **k**>

- Výber alternatívy pre dátové vstupy je nezávislé od indexovacej techniky používanú k vyhľadávaniu dátových vstupov s danou hodnotou kľúča **k**.
  - Príklady techník indexovania: B + stromy, hashové štruktúry.
  - Typicky index obsahuje doplnkové informácie, ktoré ovplyvňujú vyhľadávanie dátových položiek.

Alternatívy pre vstup dát

- Alternatíva 1:
  - Záznamy v tabuľke sú uložené iba v indexe (namiesto haldy alebo utriedeného súboru).
  - Maximálne 1 index pre 1 tabuľku môže použiť alternatívu 1. (inak sú dátové záznamy duplikované a čo vedie k prebytočnému ukladaniu a potenciálnemu rozsynchronizovaniu.)
  - Ak sú dátové záznamy veľmi veľké, počet stránok obsahujúcich dátový záznam je veľký. Stránky na organizáciu indexu a dodatočné informácie o indexe budú tiež väčšie.

Alternatívy pre vstup dát

- Alternatívy 2 a 3:
  - Dátové položky sú typicky oveľa menšie než celé záznamy. Takže tieto alternatívy sú lepšie ako alternatíva 1 s veľkými dátovými záznamami, a to najmä v prípade, ak hľadané kľúče sú malé. (Podiel indexovacej štrukúry použitej k priamemu vyhľadávaniu, ktorá je závislá na veľkosti dátových vstupov, je omnoho menšia než s alternatívou 1).
  - Alternatíva 3 je kompaktnejšia než alternatíva 2, ale vedie k premenlivej veľkosti dátovej položky, aj keď hľadané kľúče sú pevnej dĺžky.

Klasifikácia indexu

- **Primárny** vs. **sekundárny**: Ak hľadaný kľúč obsahuje primárny kľúč, potom ho nazývame primárny index.
  - **Unikátny (Unique)** index: každá dátová položka má iné hodnoty kľúčových stĺpcov.

- **Klastrovaný** vs. **neklastrovaný**: Ak poradie dátových záznamov je rovnaké alebo „blízke“ poradiu dátových položiek, potom ho voláme klastrovaný index.
  - Alternatíva 1 implikuje klastrovaný; v praxi tiež klastrovaný implikuje alternatívu 1 (keďže zoradené súbory sú zriedkavé).
  - Záznamy môžu byť utriedené iba podľa jedného kľúča.
  - Cena vyhľadania záznamov výrazne závisí od toho či je index klastrovaný!

Klastrovaný vs. Neklastrovaný Index

- Klastrovaný index môže potrebovať aj stránky pretečenia.
- Niektoré implementácie klastrovaného indexu používajú oklieštené dátové položky, pričom dátové záznamy sú v utriedenom zozname
  - Bežne sú inde stĺpce typu BLOB alebo TEXT