Kapitola 10: Diskové a souborové struktury

- Přehled fyzických ukládacích médií
- Magnetické disky
- RAID (Redundant Array of Inexpensive Disks)
- Terciární úložiště
- Přístup k médiu
- Souborové organizace

Klasifikace fyzických médií

- Rychlost, se kterou jsou data přístupná
- Náklady na jednotku dat
- Spolehlivost
  - Ztráta dat při výpadku proudu nebo při pádu systému
  - Fyzická chyba na ukládacím médiu

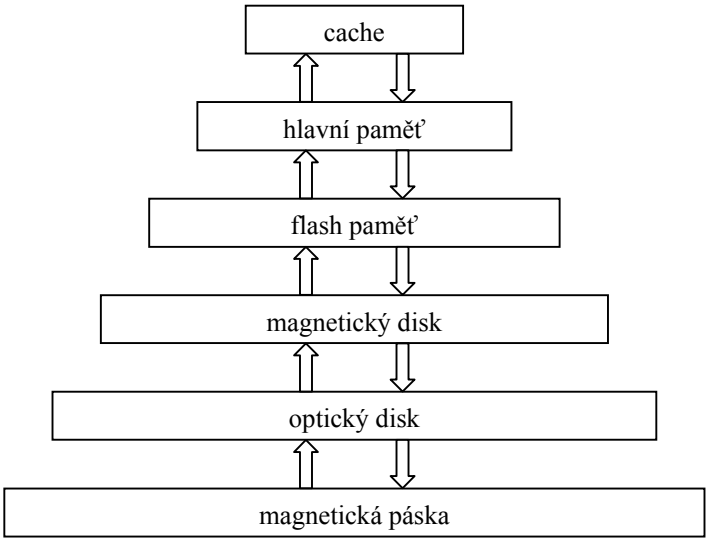
  Média lze rozdělit do skupin:
  - **Nestálá paměť** – obsah paměti se ztratí při přerušení proudu
  - **Stálá paměť** – obsah je uchován, když je napájení odpojeno. Zahrnuje sekundární a terciární úložiště, stejně jako hlavní paměť zálohovaná baterií.

Fyzická média

- Cache (keš) – nejrychlejší a nejdražší paměť, nestálá, obsluhována operačním systémem nebo hardwarem
- Hlavní paměť:
  - Strojové instrukce pracují s daty uloženými v hlavní paměti
  - Rychlý přístup, ale obecně stále příliš malá pro uložení celé databáze
  - Někdy též nazývaná jako *operační* paměť
  - Nestálá – obsah paměti je ztracený při pádu systému nebo výpadku napájení
- Flash paměti – čtení je téměř stejně rychlé jako hlavní paměť, obsah paměti je nezávislý na napájení, v některých případech může být omezený počet přepisovacích cyklů
- Magnetické disky – primární místo pro dlouhodobé ukládání dat, typicky lze uložit celou databázi.
  - Data musí být přesouvána z disku do hlavní paměti při zpracování a zapsána zpět pro uskladnění
  - **Přímý přístup** – obvykle lze data číst v libovolném pořadí (náhodný přístup)
  - Obsah většinou přežijí pád systému a výpadky napájení; chyba disku může způsobit ztrátu dat, ale tyto chyby jsou mnohem méně časté než pád systému.
- Optické disky – stálá paměť, nejznámější je CD-ROM, DVD. Většinou ve formě médií pro jeden zápis a vícenásobné čtení, které se většinou používají pro archivaci. Vyskytují se i média pro několikanásobné přepsání (počet přepisů se omezený).
- Pásková paměti – stálé, primární využití je archivace a zálohování

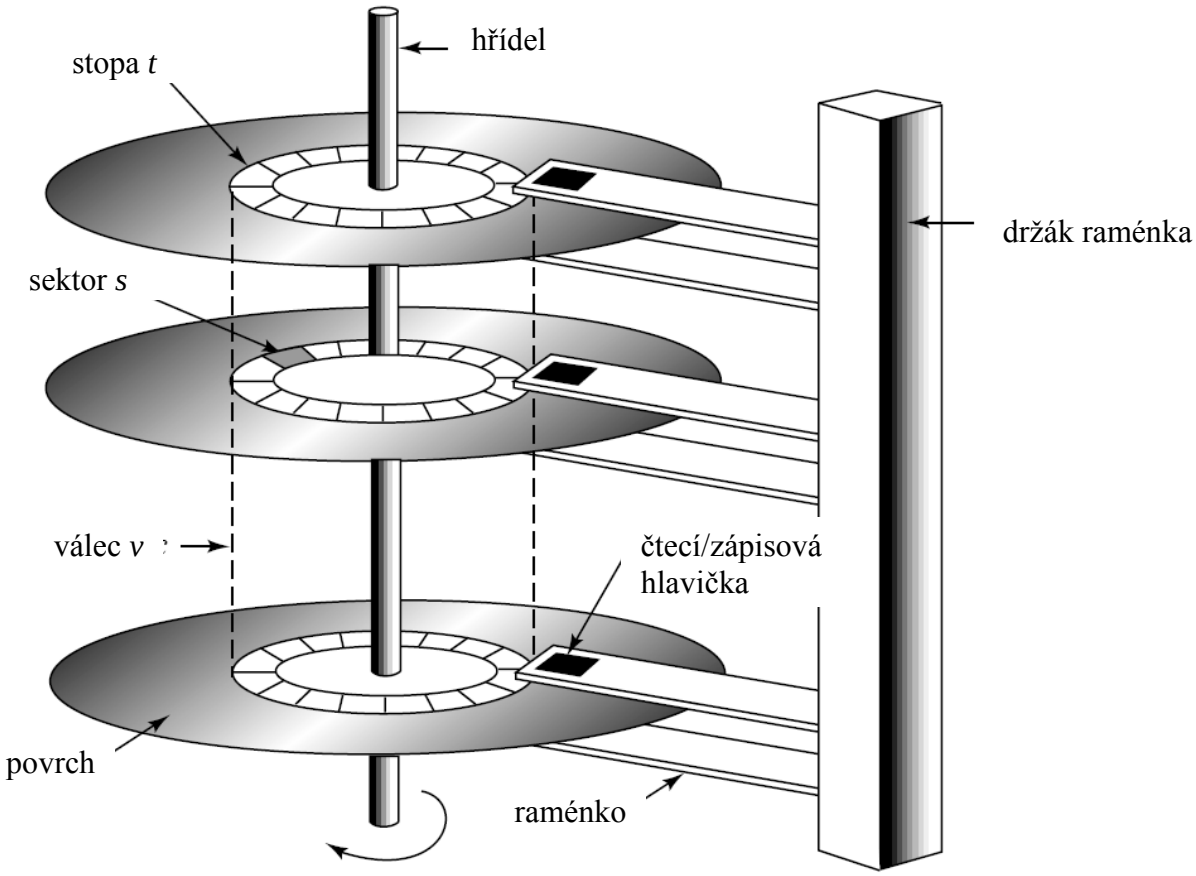
- Sekvenční přístup – mnohonásobně pomalejší než disk
- Velmi vysoká kapacita – i terabytové pásky
- Pásky lze v mechanice měnit, levnější médium než disk.

Hierarchie pamětí



- **Primární paměti:** nejrychlejší ale nestálé (cache, hlavní paměť)
- **Sekundární paměti:** další úroveň v hierarchii, stálé, poměrně rychlý přístup k datům, též nazývané jako *on-line paměti* (flash paměť, magnetický disk)
- **Terciární paměti:** nejnižší úroveň v hierarchii, stálé, pomalý přístup, též nazývané jako *off-line paměti* (magnetické pásky, optické disky)

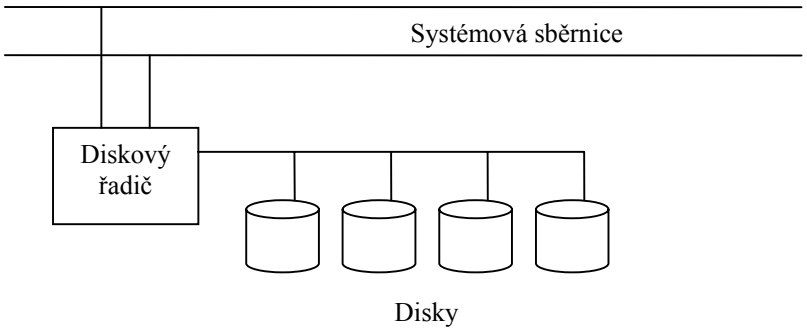
Mechanizmus magnetických disků



Magnetické disky

- Čtecí/zápisová hlavička – zařízení pohybující se v těsné blízkosti nad povrchem plotny disku, slouží k magnetickému čtení a zápisu informací na disk.
- Povrch plotny je rozdělený do kruhových *stop* (tracks), každá stopa je dělena do *sektorů.* Sektor je nejmenší datová jednotka, kterou lze číst a zapisovat.
- Pro čtení/zápis sektoru:
  - Raménko je přesune hlavičku na správnou stopu
  - Plotna se nepřetržitě otáčí, data jsou čtena/zapisována, když se sektor dostane pod hlavičku.
- Způsob montáže: více diskových ploten na jedné hřídeli s více hlavičkami (jedna pro plotnu) na společném raménku.
- **Válec** *i* se skládá z *i-té* stopy na každé plotně

Diskový podsystém



- Diskový řadič (controller) – rozhraní mezi počítačovým systémem a hardware disku.
  - Přijímá vyšší příkazy (high-level commands) pro čtení/zápis sektoru
  - Spouští akce typu vystavení raménka na správnou stopu a vlastní čtení a zápis dat.

Výkonnostní míry disků

- **Přístupová doba** – doba potřebná ke čtení/zápisu (od vyslání požadavku na čtení/zápis po zahájení přenosu dat). Skládá se z:
  - **Čas vystavení** – doba potřebná pro vystavení raménka na požadovanou stopu. Průměrná doba vystavení je třetina nejhorší hodnoty.
  - **Rotační zpoždění** – doba potřebná na otočení plotny, aby se požadovaný sektor dostal pod hlavičku. Průměrná hodnota je polovina nejhoršího času (tj. polovina doby jedné otáčky)
- **Rychlost přenosu dat** – rychlost s jakou jsou data přenášena z disku nebo na disk.
- **Střední doba poruchy** (MTTF) – průměrná doba mezi dvěma výpadky disku.

Optimalizace blokového přístupu k disku

- **Blok** – souvislá posloupnost sektorů na jedné stopě
  - Data jsou mezi diskem a hlavní pamětí přenášena v blocích
  - Velikost se pohybuje od 512 bytů do několika kilobitů
- Algoritmy pro vystavování raménka na stopy řadí požadavky tak, aby byl pohyb raménka minimalizovaný (algoritmus *výtah* je často používaný)
- Organizace souborů – bloky jsou organizovány tak, aby pořadí odpovídalo pořadí v jakém budou data čtena. Blízké informace jsou uložené na stejném válci nebo sousedním válci.
- **Stálé vyrovnávací paměti** – urychlují zápis dat na disk tím, že jsou data uložena do vyrovnávací paměti (stálá paměť); řadič zapíše data na disk ve chvíli, když disk nemá práci.

RAID

- **Nezávislá pole levných disků** (Redundant Arrays of Inexpensive Disks) – technika organizace disků, která spojuje více běžně dostupných disků do jednoho systému.
- Původně poměrně levná alternativa k velkým a velmi drahým diskům
- Dnes jsou RAID používány pro jejich vysokou spolehlivost a rychlost spíše než z ekonomických důvodů. Proto je „I“ v názvu chápáno jako **independent** (nezávislý) než levný.

Zvýšení spolehlivosti pomocí redundance

- Pravděpodobnost, že některý disk z množiny *N* disků je chybný je mnohem vyšší než pravděpodobnost havárie jednoho určitého disku. Např. systém se 100 disků, každý s MTTF 100 tisíc hodin (cca 11 let) bude mít MTTF rovnu 1000 hodinám (cca 41 dnů)!!!
- **Redundance** (nadbytečnost) – ukládej zvláštní informaci, kterou lze využít pro opětovné vytvoření informace ztracené při výpadku disku
- Např. **zrcadlení**
  - Zdvojení každého disku, logický disk se skládá ze dvou fyzických disků
  - Každý zápis musí být proveden na obou discích
  - Pokud jeden z disků je vadný, data jsou stále k dispozici na druhém.

Zvýšení výkonnosti pomocí paralelizace

- Hlavní cíle paralelizace diskových systémů:
  - Vyvažování zátěže pro zvýšení propustnosti
  - Paralelizace požadavků na velké objemy pro snížení času odezvy
- Zlepšení přenosové rychlosti pomocí dělení dat na více disků
- **Dělení na bitové úrovni** (bit-level striping) – rozděl každý byte na bity, které se uloží na různé disky
  - v poli s osmi disky jde každý bit na zvláštní disk
  - každý přístup může číst data 8x rychleji než disk jeden
  - ale přístupová doba je stejná jako v případě jednoho disku (všechny disky jsou využity pro jeden přístup)
- **Dělení na blokové úrovni** (block-level striping) – rozděl data na bloky a každý blok jde na disk *(i mod n)+1*
  - Při čtení pouze jednoho bloku je zaměstnán pouze jeden disk

Úrovně RAIDu

- Schémata poskytující redundanci při nízkých nákladech pomocí dělení dat na disky spolu s paritními bity.
- Různé úrovně RAIDu mají různé náklady, výkon a spolehlivost
- **Úroveň 0:** dělení na úrovni bloků; žádná redundance.
  - Používané pro vysoké rychlosti čtení/zápis, ztráta dat není kritická
- **Úroveň 1:** zrcadlené disky, rychlé čtení, zápis rychlý jako při použití jednoho disku, redundance - vyšší spolehlivost, nízká kapacita
- **Úroveň 2:** použití paritní informace pro zvýšení spolehlivosti, která je schopna opravit chybu v jednom bitu => pro každý bit jeden disk, navíc paritní disky

- **Úroveň 3:** bitově prokládaná parita (bit-interleaved parity), dělení na úrovni bitů, ale jediný bit stačí pro detekci i opravu.
  - při zápisu je paritní bit vypočítán a uložen na zvláštní disk
  - rychlejší přenosová rychlost než má jeden disk, ale nelze vyřizovat několik požadavků současně (všechny disky se podílejí na čtení)
  - nahrazuje druhou úroveň, protože používá pouze jeden paritní disk
- **Úroveň 4:** blokově prokládaná parita (block-interleaved parity), dělení na úrovni bloků, paritní blok je opět na zvláštním disku.
  - Vyšší propustnost požadavků, jedno čtení nepoužívá všechny disk
  - rychlejší přenosová rychlost než má jeden disk
  - paritní disk je slabé místo, protože každý zápis znamená zápis na paritní disk
- **Úroveň 5:** blokově prokládaná parity rozložená po discích
  - v případě pěti disků, je paritní blok uložen na disk *(i mod n)+1* a data na ostatní disky
  - lze vyřizovat více požadavků současně, i zápisů
  - nahrazuje čtvrtou úroveň
- **Úroveň 6:** stejné jako pátá úroveň, ale ukládá další informaci pro možnost kompletní obnovy při výpadku více disků (ne jenom jednoho). Není moc rozšířené.

Přístup k úložnému prostoru

- Databázový soubor je rozdělený na bloky pevné délky, blok je základní jednotkou přenosu i alokace místa.
- Databázový systém se snaží minimalizovat počet přenášených bloků mezi diskem a hlavní pamětí. Toho lze dosáhnout pomocí uchovávání co možná největšího počtu bloků v hlavní paměti.
- **Vyrovnávací paměť (mezipaměť, buffer)** – část hlavní paměti použitelné pro ukládání kopií diskových bloků
- **Správce mezipaměti (buffer manager)** – podsystém odpovědný pro alokaci vyrovnávací paměti v hlavní paměti
- Program, který vyžaduje nějaký blok z disku, zavolá správce mezipaměti:
  - Pokud je blok v hlavní paměti, správce mu vrátí jeho adresu
  - Pokud blok není v paměti, správce alokuje volné místo, pokud není k dispozici nahradí některý již načtený blok
  - Nahrazovaný (vyhazovaný) blok je uložen na disk, pokud byl změněn.
  - Když je volné místo, správce přečte blok z disku a uloží ho do mezipaměti a aplikaci vrátí jeho adresu.

Nahrazování bufferů

- Většina operačních systému nahrazuje blok, který byl nejdéle nepoužitý (LRU – least recently used)
- LRU může být nevhodná strategie, pokud jsou data opakovaně procházena
- Optimalizátor dotazů používá různé kombinované strategie pro lepší správu mezipaměti.
- **Přišpendlený blok** (pinned block) – daný blok nelze vyhodit z mezipaměti
- **Ihned zahoď** (toss-immediately) – ihned po ukončení zpracování bloku je blok z mezipaměti uvolněn.

- MRU (most recently used) – blok naposledy použitý je uvolněn, během zpracování bloku je blok označen jako přišpendlený a nemůže být uvolněn, po ukončení zpracování se stává naposledy použitím blokem.
- Správce mezipaměti může používat různé statistické informace, např. nějaká relace je často používána => ponechej ji v paměti

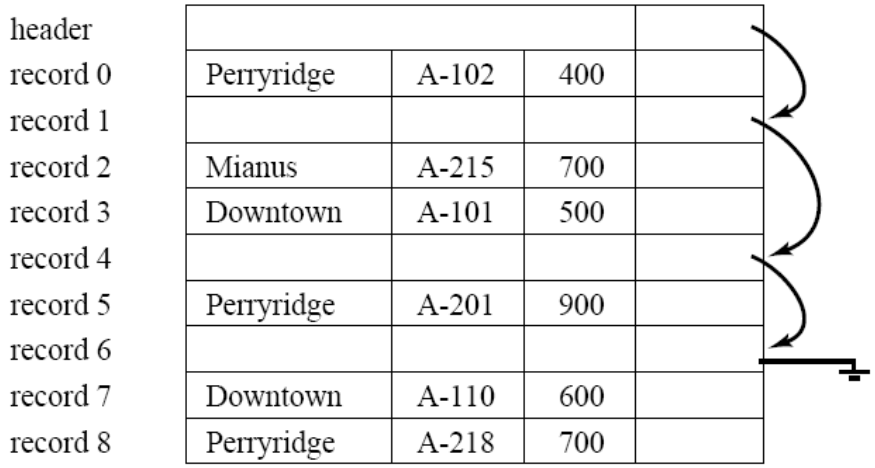
Souborové organizace

- Databáze je uložena v kolekci souborů. Každý soubor je tvořen posloupností záznamů (records). Záznam se skládá z jednotlivých atributů (polí).
- Jeden přístup:
  - Délka záznamu je pevná
  - Každý soubor má záznamy pouze stejného typu
  - Jeden soubor pro jednu relaci

  Tento případ je nejjednodušší na implementaci.

Záznamy s pevnou délkou

- Jednoduchý přístup:
  - Záznam *i* je uložen na pozici *l*(i-1)* v souboru, kde *l* je délka záznamu
  - Přístup k záznamu je jednoduchý, ale záznam může přesahovat bloky
- Mazání záznamu *i* – možnosti:
  - Všechny následující záznamy (*i+1,…,n*) o jeden záznam níže
  - Přesuň poslední záznam *n* na místo *i-tého* záznamu
  - Udržuj si seznam prázdných záznamů
    * V hlavičce souboru si ulož číslo prvního smazaného záznamu
    * V prvním smazaném záznamu si ulož číslo dalšího smazaného záznamu
    * Tato čísla mohou být chápána jako ukazatele na další volnou paměť

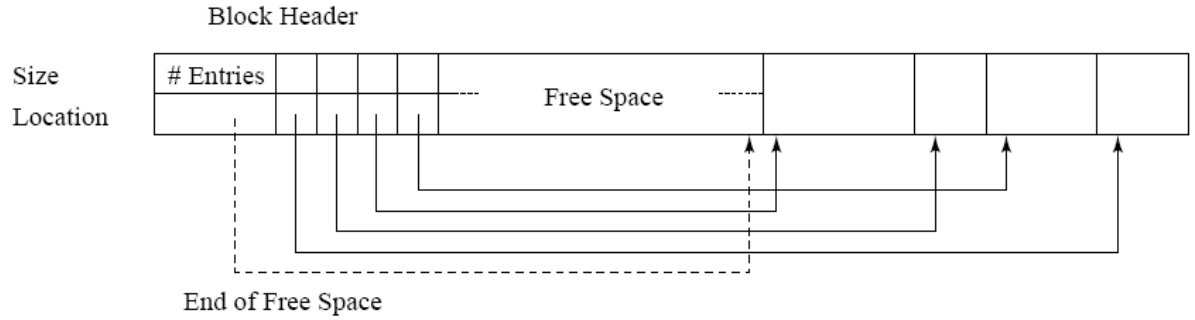


    * Prostorově nenáročné řešení, prázdné atributy lze využít pro uložení ukazatele

Záznamy s proměnnou délkou

- Záznamy s proměnnou délkou vznikají v DB systémech několika způsoby:
  - Ukládání více různých typů záznamů v jednom souboru
  - Záznamy obsahující atributy s proměnnou délkou

- Řešení pomocí řetězcové reprezentace:
  - Na konec záznamu je připojen zvláštní znak pro ukončení záznamu ⊥
  - Problémy s mazáním záznamů a jejich zvětšováním
- Dělený soubor (slotted page structure):



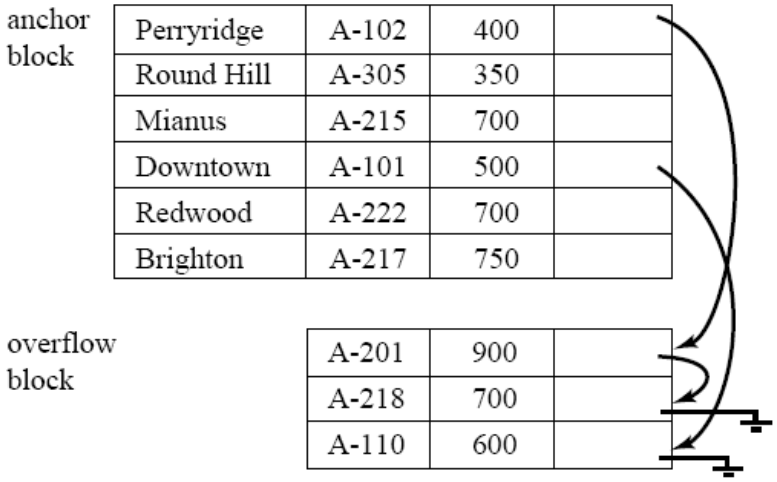
  - Hlavička obsahuje počet záznamů, ukazatel na konec volného místa a dvojici (ukazatel, délka) pro každý záznam
  - Při mazání jsou ostatní záznamy přesunuty tak, aby vzniklo souvislé volné místo, a hlavička je aktualizována
- Řešení s pevnou délkou pomocí vyhrazeného (rezervovaného) místa:
  - Lze použít pokud známe maximální délku záznamu
  - Každý záznam je pevně dlouhý (stejně jako maximálně dlouhý záznam)
  - Nevyužité místo je vyplněno NULL hodnotou nebo koncem záznamu ⊥

0	Perryridge	A-102	400	A-201	900	A-218	700
1	Round Hill	A-305	350	⊥	⊥	⊥	⊥
2	Mianus	A-215	700	⊥	⊥	⊥	⊥
3	Downtown	A-101	500	A-110	600	⊥	⊥
4	Redwood	A-222	700	⊥	⊥	⊥	⊥
5	Brighton	A-217	750	⊥	⊥	⊥	⊥

- Řešení s pevnou délkou pomocí ukazatelů:
  - Maximální délka záznamu není známa
  - Proměnná délka záznamu je vyjádřena pomocí zřetězeného seznamu záznamů pevné délky

0	Perryridge	A-102	400	
1	Round Hill	A-305	350	
2	Mianus	A-215	700	
3	Downtown	A-101	500	
4	Redwood	A-222	700	
5		A-201	900	
6	Brighton	A-217	750	
7		A-110	600	
8		A-218	700	

  - Nevýhodou je plýtvání místem ve všech záznamech seznamu kromě prvního
  - Lze řešit pomocí bloků dvou druhů:
    * Kotvící blok (anchor block) – obsahuje první záznam v řetězci
    * Přetokový blok (overflow block) – obsahuje záznam, které ukládají pouze proměnné atributy.

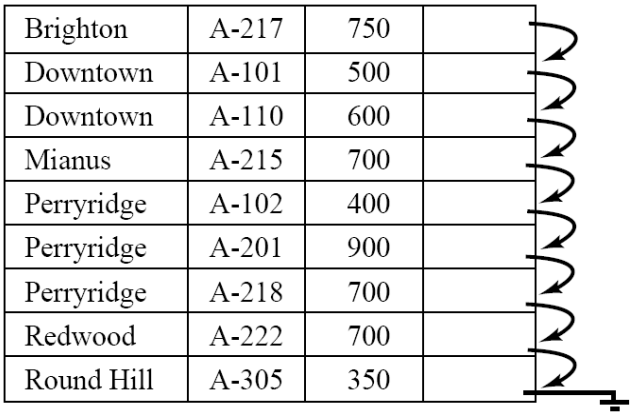


Organizace záznamů v souboru

- **Halda** (heap) – záznam je uložen kdekoli na volné místo v souboru
- **Sekvenční** – ukládáme záznamy za sebou uspořádané podle vyhledávacího atributu
- **Hešování** (hashing) – používá se hešovací funkce pro výpočet čísla bloku, kde má být záznam uložen. Toto číslo je vypočítáno na základě hodnot vybraných atributů.
- **Shlukování** – záznamy různých relací mohou být uloženy v jednom souboru a příbuzné záznamy jsou uloženy ve stejném bloku

Sekvenční soubor

- Vhodný pro aplikace, které postupně procházejí celý soubor
- Záznamy jsou obvykle uspořádány podle vyhledávacího klíče (atributu)



- Mazání pomocí řetězení volných záznamů
- Vkládání – nejprve nalézt místo pro místo pro vkládaný záznam
  - Pokud je tam volné místo, ulož
  - Pokud ne, vlož nový záznam do přetokového bloku
  - V obou případech se musí aktualizovat seznam volného místa
- Občas je nutná reorganizace souboru

Shlukování

- Vložení více relací v jednom souboru
- Příklad uložení relací *zákazník* a *účet* do jednoho souboru

Hayes	Main	Brooklyn
Hayes	A-102	
Hayes	A-220	
Hayes	A-503	
Turner	Putnam	Stamford
Turner	A-305	

  - Vhodná organizace pro dotazy zahrnující spojení relací a pro dotazy, které vypisují účty pro jednoho zákazníka
  - Nevhodné pro dotazy zpracovávající pouze relaci *zákazník*
  - Důsledkem této metody je soubor s proměnnou délkou záznamu