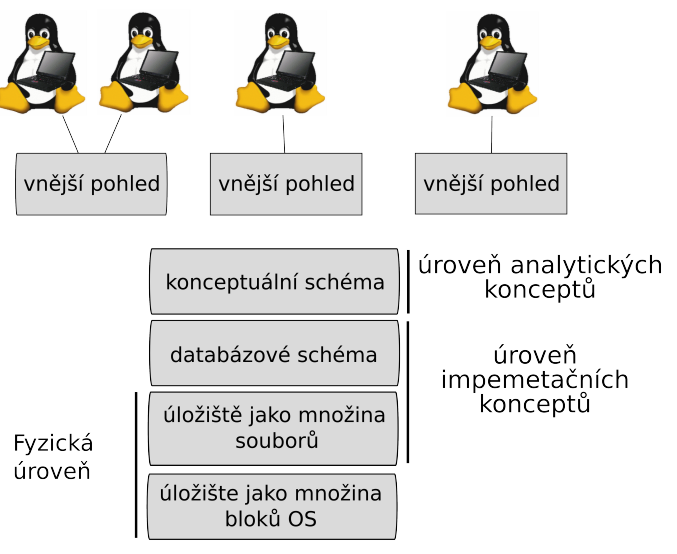
**BI-SPOL-11 3 úrovně pohledu na data (konceptuální, implementační, fyzická). Struktury pro ukládání dat v relačních databázích s ohledem na rychlý přístup k nim (speciální způsoby uložení, indexy apod.)**

BI-DBS



### 3 úrovně pohledu na data

**Konceptuální**

* zabývá se modelováním reality, komunikace se zákazníkem
* snaží se nebýt ovlivněna budoucími prostředky řešení (způsobem implementace)
* používá se grafická notace (např. UML class diagram, ER model)
* integrace různých uživatelských pohledů
* výsledek je vstupem pro realizaci databáze
* slouží jako dokumentace

**Implementační (databázový)**

* vztahuje se ke konkrétnímu databázového modelu a používá jeho konstrukční dotazovací a manipulační prostředky (relační, objektová, síťová, XML…)
* realizace datové struktury popsané v konceptuálním modelu, na základě konkrétní technologie (relační, objektová, …)
* model je transformován do modelu, který odpovídá konkrétní technologii
* popisuje, čím je datový obsah systému, popsaný konceptuálním a strukturálním modelem, realizován

**Fyzická**

* fyzické uložení dat (sekvenční soubor, indexy, clustery…)
* uživatelé (včetně vývojářů) jsou od ní odstíněni databázovou vrstvou (logickou vrstvou DBMS)
* pro přístup se využívá přístupový interface (dotazovací jazyk, API...)

### Struktury pro ukládání dat v relačních databázích s ohledem na rychlý přístup k nim

ROWID – unikátní ID řádku v rámci celé databáze, používají se pro implementaci indexů

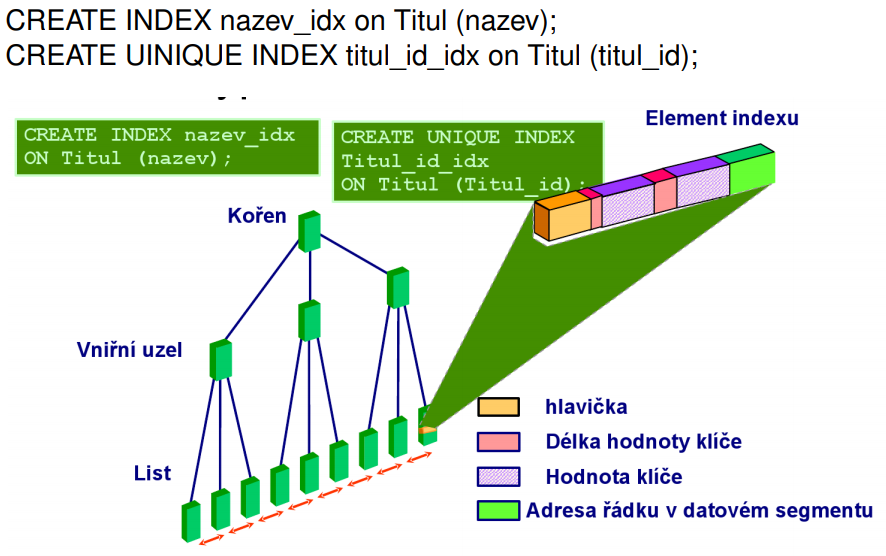
* ROWID je v Oracle, ale obdoba je v každé DB
* Implementační trik
* Struktura <object\_id, file\_id, block\_id, row\_id>
  + Block\_id – ID bloku v rámci souboru
  + Row\_id – ID řádky v rámci bloku
* Může se během života DB měnit

#### Index

* pomocná (sekundární) struktura, která slouží pro rychlý přístup k datům (řádkům) podle klíče indexu (index key – **nemusí být unikátní**)
* klíč může být složený/jednoduchý, unikátní/neunikátní
* nad jednou tabulkou může existovat mnoho různých indexů (každý pro jiný index klíč)
* různé typy indexů:
  + indexy založené na struktuře B-stromu (B\*-Tree based indexes)
    - vysoká selektivita klíče indexu (blízko 1) – napč. PK, UK, username, (prijmeni, jmeno)
      * (skoro) unikátní data
  + bitmapové indexy
    - velmi nízká selektivita, např. pohlaví, okres
      * velmi neunikátní data
  + další
* váže se k jednomu nebo k více sloupcům tabulky, nad kterou se často dotazuje
* snižuje se rychlost zápisu (musíme updatovat indexy), ale zvyšuje rychlost čtení – když máme miliony záznamů, tak hledání je náročné – indexy velice zrychlují
* jsou nutné pro větší data, bez nich nefunguje DB rozumně
* index může bát složený
* Pro PK a UK se indexy vytváří automaticky – efektivní kontrola platnosti IO

#### B\*-Tree

* vyvážený *m*-ární strom
* Data jsou zpřístupněna pomocí klíčů a uložena vždy až v listech
* Data ve vnitřním uzlu jsou organizována (máme „intervaly“)
* faktor větvení (v praxi cca 100) + hloubka stromu (v praxi menší než 4)
  + je široky, ale mělký
* důležité je vyvážení stromu, na pozadí provádí DBMS
* když přistupuju k indexu, tak přistupuju od kořene – procházím ten strom, až se dostanu do listu – tam zjistím ROWID, kde ten řádek najdu
* průchod stromem:
  + v uzlu mám tabulku (dva sloupce – klíč a odkaz do nižšího patra stromu – když mám 3 větve, tak budu mít 3 řádky této tabulky)
  + porovnávám hodnotu, kterou hledám s hodnotami klíčů v tabulce a podle toho jak dopadne porovnání jdu doleva/doprava (když mám např. 3 řádky, tak porovnám s prvním, zjistím, že mám větší hodnotu – jdu na další řádek a zjistím, že mám větší – jdu na druhou větev)
  + a takto postupně snižuju rozsahy (intervaly) – dostanu se až do listu, kde už svoji hodnotu najdu a zjistím ROWID. Nebo nenajdu a zjistím, že taková hodnota klíče neexistuje

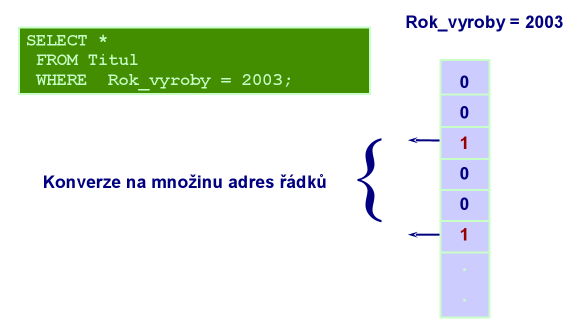


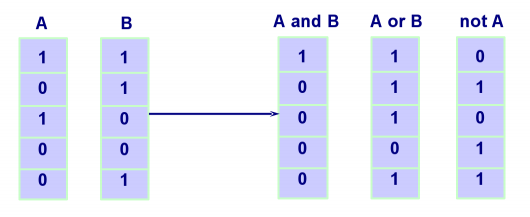
#### Bitmapový index

* narozdíl od B-stromu pro sloupce s nízkou selektivitou
* binární matice
* předvypočítané binární odpovědi pro každý záznam
* DML operace velmi drahé
* vhodné pro záznamy s neunikátními položkami
* Pokud u jednoho sloupce tabulka je nízká selektivita (počet možných hodnot), pak můžeme bitově označit každou možnou hodnotu a používat pro vyhledávání celé řádky
* Př.
  + Máme index nad rokem výroby
  + Při selectu (podle toho indexu) nám to vezme jen ten daný vector a podle hodnot (bereme 1) zjistíme, které tituly na to sedí
  + Větší síla se ukáže u komplikovanější WHERE klausule (mám AND, OR, NOT) – vezmu bitové vektory a udělám mezi nimi bitovou operaci – velmi rychlé

Obsah obrázku stůl

Popis byl vytvořen automaticky





Srovnání B-strom s bitová mapa

Obsah obrázku stůl

Popis byl vytvořen automaticky

**DSS** = decission support system – velká rozhodnutí, založena na historických datech

* Dnes termín OLAP
* Datové sklady, které sbírají data z provozu a nad tím se dělá analytika
* Málo současně pracujících uživatelů
* Často agregační dotazy

**OLTP** = online transaction processing – aktuální data, každodenní transakce

### Způsoby uložení dat (uložení relační tabulky)

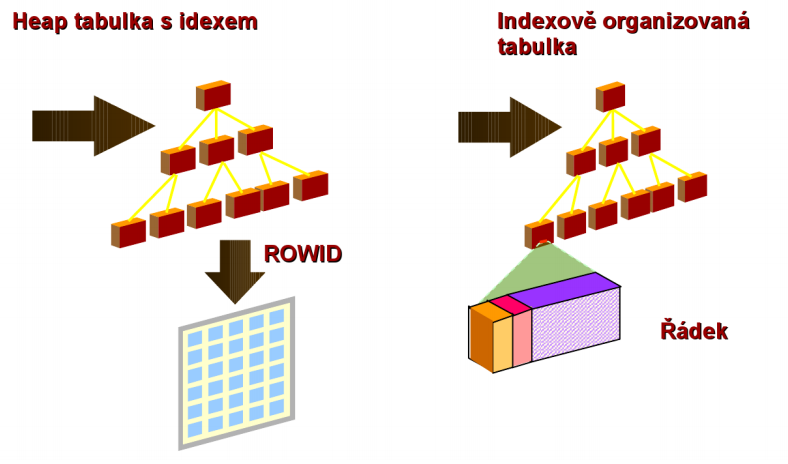
#### Heap tabulka

* Neorganizovaná hromada
* Informace se vkládají na první volné místo (v paměti)
* Vyhledávání se provádí per položka

#### Heap tabulka s indexy

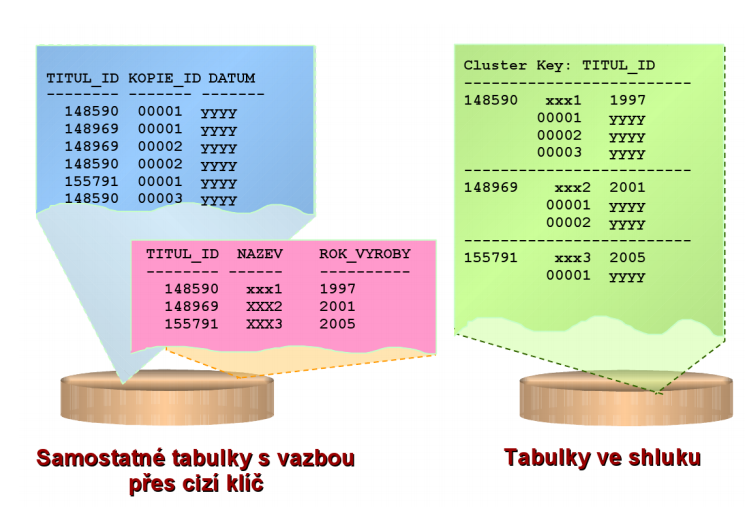
* Nejběžnější – k heapu mám nějaké **indexy** (viz. výše) pro rychlé hledání

#### Indexově organizovaná tabulka



* Na levo – máme strom s IDčky, dostaneme se do listu, kde máme UID – přes to se dostaneme do tabulky
  + Indexů můžu mít kolik chci
* Na pravo – v listu mám ten samotný záznam – rychlejší
  + Klíč, pomocí kterého můžu mít organizovanou tabulku je jen jeden

#### Cluster



* Možnost, jak fyzicky uložit data ze dvou tabulek společně
  + JOIN tabulek je hotový i ve fyzickém uložení
* Bucket je jedna skupina (ty části oddělené čárou vpravo)
  + Je vhodné navrhovat cluster tak, aby velikost bucketů byla přibližně stejná
* Pokud tabulky často joinujeme, může se nám vyplatit to ukládat takto

**Další techniky zrychlení přístupu k datům:**

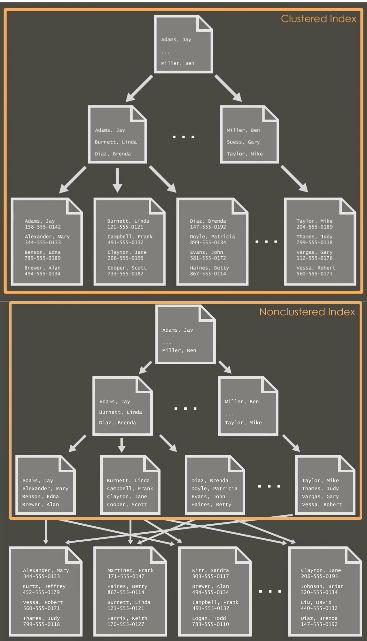
**Obsah obrázku text

Popis byl vytvořen automaticky**

**Pohled** – pojmenovaný select (můžeme s ním pracovat jako s tabulkou)

**Materializovaný pohledy** – select se uloží a nad tím se můžu znovu dotazovat – jako cache – je třeba se o to starat (např. aktualizace každých 10 minut)

Distribuované databáze – nemáme jeden DB server, ale je jich víc – rozdělujeme zátěž



**Toto není v přednášce**

**(Non-)Clustered index**

