RAID

- **Redundantní pole levných (nezávislých) disků** (Redundant Arrays of Inexpensive (Independent) 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 spíše chápáno jako **independent** (**nezávislý**) než levný.

Jazyk pro definici dat (Data definition language; DDL)

- Značení pro specifikaci definice schématu databáze
- DDL kompilátor generuje množinu tabulek uložených v *datovém slovníku* (*Data dictionary*)
- Datový slovník obsahuje *metadata* (data o datech)
- *Jazyk pro ukládání a definice dat* (*Data storage and definition language*) – speciální typ DDL, pomocí kterého se specifikuje struktura *uložení dat* a metody použité databázovým systémem k *přístupu k datům*

PRIKLAD: SELECT, INSERT, UPDATE

Jazyk manipulace s daty (Data manipulation language; DML)

- Jazyk pro zpřístupnění a manipulaci s daty organizovanými příslušným datovým modelem
- Dvě třídy jazyků
  - Procedurální – uživatel specifikuje, která data jsou vyžadována a jak je získat
  - Neprocedurální (deklarativní) – uživatel specifikuje, která data jsou vyžadována, ale ne, jak je získat

PRIKLAD: CREATE, ALTER, DROP

Pohledy

• Poskytují mechanismus, jak skrýt nějaká data před nějakými uživateli. Pro vytvoření pohledu použijeme příkaz:

create view *v* **as** <výraz dotazu>

kde:
– <výraz dotazu> je jakýkoliv dovolený výraz
– *v* reprezentuje jméno pohledu

Definice pohledu

• Pohled je definován příkazem **create view**, který má tvar

create view *v* **as** <výraz dotazu>

kde <výraz dotazu> je jakýkoliv správný výraz relační algebry. Jméno pohledu je reprezentováno proměnnou *v*.

• Jakmile je pohled definován, jeho jméno může být použito pro odkazování na virtuální relaci, která pohled generuje.
• Definice pohledu není to samé jako vytvoření nové relace vyhodnocením dotazovacího výrazu. Definice pohledu způsobuje uložení výrazu, aby mohl být substituován do dotazů, které používají tento pohled.

Příklady dotazů

• Pohled skládající se z poboček a jejich zákazníků

create view *všichni-zákazníci* **as**
(**select** *pobočka-jméno, zákazník-jméno*
from *vkladatel, účet*
where *vkladatel.číslo-účtu = účet.číslo-účtu*)
union
(**select** *pobočka-jméno, zákazník-jméno*
from *půjčovatel, účet*
where *půjčovatel.půjčka-číslo = půjčka.půjčka-číslo*)

Optimalizace – nalezení nejlevnějšího plánu pro vykonání dotazu
optimalizace dotazů využívá statistické informace z databázového katalogu aby určila cenu jednotlivých plánů a pak vybere ten s nejnižší cenou. Mezi statistické informate patří např. počet záznamů v každé relaci nebo velikost záznamů
Cena dotazu jsou seeky + zapsané a načtené bloky.

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 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

Plány

• **Plány** jsou posloupnosti, které určují chronologické pořadí provádění instrukcí souběžných transakcí
  – plán pro množinu transakcí musí obsahovat všechny operace prováděné těmito transakcemi
  – musí zachovávat pořadí instrukcí stejné jako v každé jednotlivé transakci

Serializovatelnost

• Základní předpoklady – každá transakce zachovává konzistenci databáze
• Tedy sériový plán zachovává konzistenci databáze
• Plán je serializovatelný, pokud je ekvivalentní sériovému plánu. Různé formy ekvivalence plánů vedou k následujícím pojmům:
  – Serializovatelnost podle konfliktu
  – Pohledová serializovatelnost
• Ignorujeme všechny instrukce kromě **čtení** a **zápisu** a předpokládáme, že transakce mohou provádět libovolné výpočty na datech v lokálních vyrovnávacích pamětech mezi čteními a zápisy. Naše zjednodušené plány se tedy skládají pouze z operací **čtení** a **zápisu.**
• Říkáme, že plán *S* je *serializovatelný podle konfliktu*, jestliže je ekvivalentní podle konfliktu se sériovým plánem.

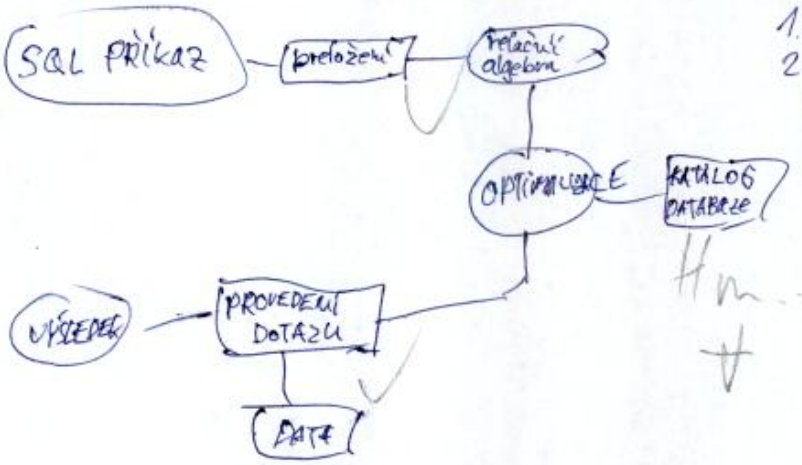
HIERARCHIA PAMATI: primarne – cache, operacna pamat
Sekundarne – rotacny pevny disk
Terciarne – magneticke pasky, CD, DVD
3 ZAKLADNE ZLOZKY CASOVYCH NAKLADOV: doba nastavenia ramena s hlavou na miesto pre citanie/zapis
Rotacne oneskorenie = doba otocenia disku
Samotna doba citania/zapisu, 3-6 Mb/s

CUDZI KLUC – integracne obmedzenie a sluzi na spojenie 2 entit medzi ktorymi je vztah. Tento vztah definuje prave relacia v ktorej sa nachadza cudzi kluc
PODMIENKA PRE CUDZI KLUC: π_{uco}(zapis) „je podmnozinou alebo sa rovna“ π_{uco}(student)

Nakreslete schéma procesu zpracování dotazu (co se děje v databázi mezi zadáním příkazu SELECT a vrácením odpovědi) a jednotlivé kroky stručně komentujte.

Příklad 9
6 bodů

4b



1. přeložení do relační algebry
2. optimalizace na základě katalogu
3. provedení dotazu na datech
4. podání výsledku

Hm...

+ co dělá!

Uvažujte běžný rotační pevný disk, na kterém je uložen soubor složený z 10 bloků.

Příklad 7
6 bodů

6

a. Uveďte a stručně charakterizujte tři základní složky časových nákladů potřebných ke čtení jednoho bloku souboru. *(4b)*
b. Představte pojem fragmentace souboru a jaké má důsledky. *(2b)*

a) posun čtecí hlavy + rotační spoždění + provedení operace čtení

hlava se posune do polohy, kde jsou očekávána data — rotuje se otočí do polohy, kde začíná sektor

b) Soubor je rozdělen na více částí (fragmentů) ty jsou různě po disku
- musí se často posouvat hlava a vše co je sprístupen spojeno
- pomalé, řeší se defragmentací disku

Spouštěč (Triggers)

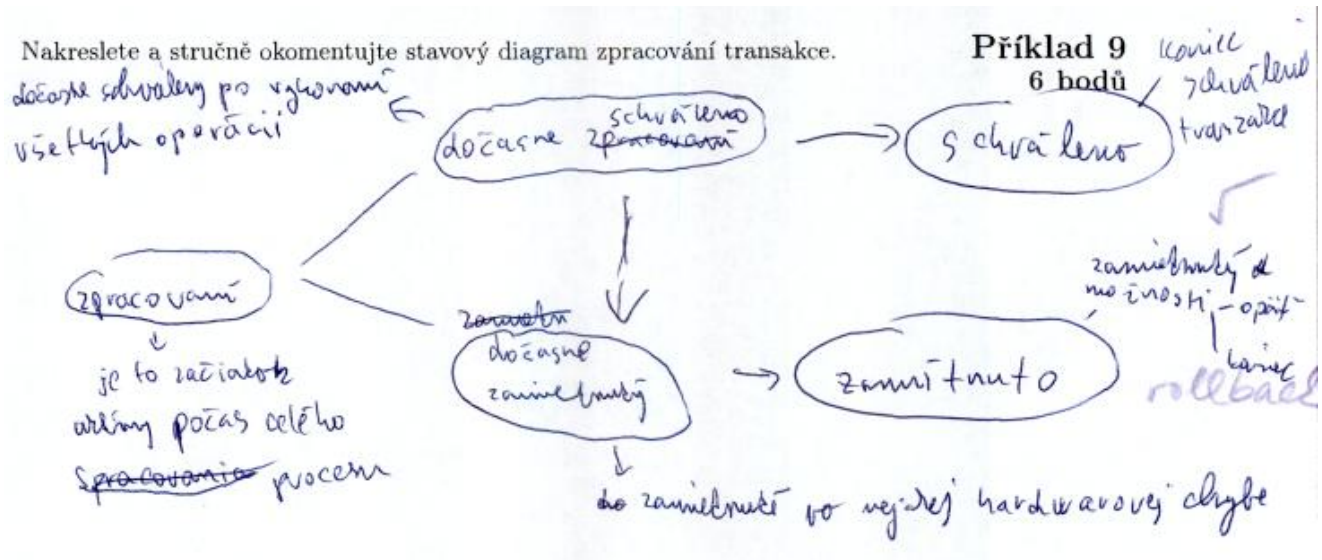
Spouštěč (Trigger, spoušť) je příkaz, který systém automaticky spouští jako vedlejší efekt modifikace databáze.

- Pro aktivaci tohoto mechanismu musíme:
  - Specifikovat podmínky, za jakých je spouštěč prováděn.
  - Specifikovat akce, které se budou dít při jeho spuštění.

Příklad triggeru

- Předpokládejme, že místo povolení záporných zůstatků účtů banka zachází se saldem (přečerpáním) takto:
  - nastaví zůstatek účtu na nulu
  - vytvoří půjčku s částkou salda
  - dá této půjčce číslo tohoto účtu
- Podmínka spuštění této spouště je

CO JE NAJLEPSIE (hashovanie, B+strom, B strom) pre:
HLADANIE PRVKU S ROVNAKOU HODNOTOU: hashovanie, B+ strom, B storm
HLADANIE VSETKYCH PRVKOV Z INTERVALU HODNOT – B+ strom, B strom, hashovanie
VKLADANIE JEDNEHO PRVKU – hashovanie, B+ strom, B strom
Hashovanie – rychle vdaka hashu, hladanie viacerych prvkov je pomale
Stromy – pomalsie kvoli listom ktore maju konecny pocet prvkov a moze ich byt mnoho, pre vkladanie je treba vela pamatoveho miesta, zlozitost pre viac prvkov je nizsia



RAID

- **Redundantní pole levných (nezávislých) disků** (Redundant Arrays of Inexpensive (Independent) 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 spíše chápáno jako **independent** (**nezávislý**) než levný.

-zakladne principy v nich pouzivane – zdrcadleni
– delenie dat na bitovej urovni alebo blokovej urovni