**BI-SPOL-10 Transakce a jejich vlastnosti - ACID**

BI-DBS

## Transakce

Máme dva základní požadavky na DBMS:

* Chránit data – ve smyslu odolnosti vůči různým haváriím serveru
* Poskytnout korektní, rychlý a asynchronní přístup většímu množství současně pracujících uživatelů

V architektuře DB stroje jsou moduly/komponenty:

**Modul řízení souběžného zpracování** (concurency control)

* zajišťuje uživatelům, že každý vidí konzistentní stav DB bez ohled na to, že více uživatelů přistupuje asynchronně ke stejným údajům

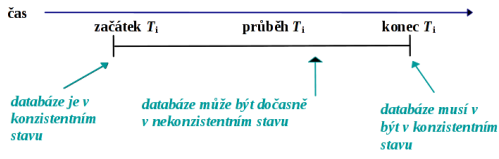
**Modul zotavení z chyb** (recovery)

* zajišťuje, že stav DB není narušen v případě chyby SW, systému nebo fyzického média v průběhu zpracování úlohy měnící data v databázi. Důsledkem takového incidentu nesmí být nekonzistence databáze.

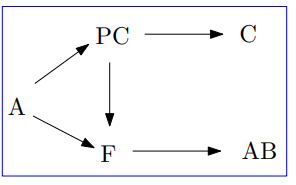
Prostředkem takového chování jsou **transakce**.

### Transakce

* sekvence akcí, které spolu „logicky souvisí“ (tj. je vhodné je vnímat jako jeden celek)
* sekvence akcí, které mění stav databáze
* skupina příkazů, která převádí jeden konzistentní stav databáze do jiného konzistentního stavu, přičemž mezistavy nemusí být konzistentní
* z pohledu DB je to atomický příkaz, provede se buď celý, nebo vůbec.
* **Transakční zpracování** (dodržení vlastností ACID) zajistí, že po (jakémkoliv) skončení transakce zůstane DB konzistentní (platí všechna IO definovaná ve schématu)



Příklad – převod peněz. Z účtu A odečteme a k účtu B přičteme – musí to být transakce, protože se to musí provést najednou nebo vůbec

**Stavový diagram transakce**

* *A (aktivní)* - od začátku (probíhají DML příkazy)
* *PC (částečně potvrzený)* - po provedení poslední operace transakce
* *C (potvrzený)* - po úspěšném zakončení (tzn. po potvrzení operace COMMIT)
* *F (chybný)* - nelze v normálním průběhu transakce pokračovat
* *AB (zrušený)* - po skončení operace ROLLBACK, uvedení do stavu před započetím transakce

#### Začátek a konec transakce

**Konec transakce**

* Explicitní
  + COMMIT (potvrzení)
  + ROLLBACK (zrušení)
* Implicitní
  + Ukončení session (záleží na klientovi, zda commit nebo rollback)

**Začátek transakce**

* Je obvykle vymezen skončením transakce předchozí nebo vznikem session

### Vlastnosti transakcí

**ACID** vlastnosti transakce:

* *atomicita (***Atomicity***)* - transakce musí buď proběhnout celá nebo vůbec
* *konzistence (***Consistency***)* - transakce transformuje databázi z konzist. stavu do jiného konzist. stavu
* *nezávislost (***Independence***)* - dílčí efekty jedné transakce nejsou viditelné jiným transakcím
* *trvanlivost (***Durability***)* - efekty úspěšně ukončené (potvrzené) transakce jsou trvale uloženy (persistence)

### Obnova databáze po pádu

* využívá se transakční žurnál (log soubory)
* v transakčním logu jsou „změnové vektory“
* operace použité při obnově: UNDO, REDO

#### Žurnál

* využívá se pro obnovení databáze po pádu
* jedná se o log soubory
* obsahuje sekvenci změnových vektorů pro operace commit, rollback a checkpoint  
  <transID, blockID, old data, new data>
* checkpoint (synchronizace database buffer cache a DB bloků na disku) má jednoznačné checkpoint number
* žurnál a přidružená infrastruktura umožňuje implementaci Atomicity a Durability u transakčního zpracování
* informace z transakčního žurnálu se používají **pouze** pro obnovu DB po chybě
* pro operaci ROLLBACK se používají jiné datové struktury
* (v ORACLE je žurnál implementován formou minimálně dvou souborů pevné velikosti, které se cyklicky přepisují)

#### Zotavení z chyby – třídy možných chyb

**Globální chyby (mají vliv na více transakcí)**

* Spadnutí systému serveru (např. výpadek proudu), důsledek je obecně ztráta obsahu vyrov. Pamětí
* Chyby systémové, které ovlivňují transakce, avšak nikoli celou databázi (např. uváznutí, odumření komunikace klienta se serverem)
* Chyby médií (např. incident na disku), které zapříčiní ohrožení databáze, nebo její části

**Lokální chyby (v jedné transakci)**

* Logické chyby, které by měly být „odchyceny“ a ošetřeny na úrovni transakce explicitním vyvoláním operace ROLLBACK (pokus o porušení IO při zápisu do DB, dělení nulou při výpočtu)

#### Zotavení z chyby – po restartu systému

* Časové známky (v žurnálu a v hlavičkách db souborů); slouží k nalezení místa (v žurnálu) odkud je třeba začít s rekonstrukcí databáze
* Požadavky
  + Na transakce, jejichž stav bude v důsledku incidentu nedefinovaný je nutné použít ROLLBACK
  + Transakce, které byly potvrzené před tím, než nastala chyba systému, avšak které nebyly dokončeny fyzicky (např. odeslání vyrovnávacích pamětí na disk), je nutné zopakovat a uložit do datových souborů
* Technicky má obnova systému po pádu dvě fáze:
  + Roll Forward – přehrání transakčního žurnálu (a obnova vyrovnávací paměti)
  + Roll Back – odvolání transakcí, které nebyly v době pádu dokončeny

#### Rozvrhy

* Transakce se skládají z dílčích operací T1 = {T11, ... ,T1n } a T2 = {T21, ..., T2m }. Provádět je můžeme celé sériově (např. nejdřív celá T1 , pak celá T2 ), nebo můžeme míchat dílčí operace (paralelně).
* ***rozvrh***: stanovení provádění dílčích akcí více transakcí v čase
* ***rozvrhovač****:* se snaží o maximalizaci paralelismu zpracování
* nebezpečí **ztráty aktualizace**: hodnota X v DB měla být 79, ale je 84  
  Obsah obrázku stůl

  Popis byl vytvořen automaticky
* nebezpečí **dočasné aktualizace** (při chybě systému). Po provedení operace ROLLBACK u transakce T1 bude aktualizace provedená transakcí T2 založena na posléze odvolaných změnách, takže výsledek bude chybný

Obsah obrázku stůl

Popis byl vytvořen automaticky

* je uspořádatelný, pokud precedenční graf neobsahuje kružnici (tzn. tento obrázek má 2 neuspořádatelné rozvrhy)

**Problémy s paralelním zpracováním**

* *neopakovatelné čtení* (někdo transakci pod rukama změní hodnoty)
  + transakce T1 provede SELECT a použije načtené hodnoty
  + transakce T2 poté změní hodnoty některých řádků
  + když T1 později provede tentýž select, dostane jiné hodnoty
    - z pohledu transakce T1 je to špatně, že v rámci jedné transakce dostane ze stejného dotazu jiná data
* *fantóm* (někdo transakci udělá insert/delete a commitne změny)
  + T1 provede select a použije načtené hodnoty
  + T2 smaže nebo přidá řádky
  + T1 v dalším dotazu obdrží jinou sadu dat
    - Má řádky navíc/chybí – z pohledu transakce T1 je to divný

#### Stupně izolace

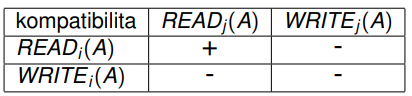
Obsah obrázku stůl

Popis byl vytvořen automaticky

* Všechny DB stroje pracují v úrovni 1 (read commited) – s neopakovatelným čtením a fantómem dokážeme žít – tamto je moc restriktivní (ztěžuje práci více uživatelů)
* Pomocí úrovně izolace lze nastavit požadované chování prokládaných transakcí. Čím vyšší úroveň, tím blíže ACID a tím horší paralelizace.

#### Uspořádatelnost rozvrhu

* Potřebujeme řídit souběh transakcí, tak aby si vzájemně nepřepisovali práci, ale nechceme to dělat sériově (celá T1, celá T2), ale **chceme to dělat paralelně**
* Chceme, aby ten rozvrh byl **korektní** (korektní = ekvivalentní sériovému rozvrhu)
* Chceme, aby náš databázový stroj byl navržen tak, aby nás nechal vytvářet paralelní rozvrhy mnoha souběžných transakcí – ale musí to zaručovat uspořádatelnost
* Budeme stavět na kompatibilitě operací READ a WRITE
* Dvě operace jsou konfliktní, jestliže výsledky jejich různého sériového volání nejsou ekvivalentní
* Operace, které nejsou konfliktní nazýváme kompatibilní (+ OK, - konflikt)



* Definice:
  + Rozvrh je uspořádatelný (ekvivalentní s nějakým sériovým) pokud relativní pořadí konfliktních operací nad stejnými objekty je ve všech transakcích stejné
  + Máme rozvrhy S1 a S2 pro množinu transakcí T1.  
    S1 je s S2 ekvivalentní, pokud jsou splněny dvě podmínky:
    - Pokud se v prvním rozvrhu vyskytuje READ(A) v Ti a tato hodnota vznikla z WRITE(A) v transakci Tj, potom totéž musí být zachováno v druhém rozvrhu
    - Pokud se v prvním rozvrhu vyvolá poslední operace WRITE(A) v Ti, pak totéž musí být i v druhém rozvrhu
* Př:S3 je OK, S4 není OK (není ekvivalentní se sériovým)

Obsah obrázku stůl

Popis byl vytvořen automaticky

#### Precedenční graf

* orientovaný graf
* *vrchol*: jednotlivá transakce
* *hrana*: konfliktní operace nad stejnými daty, orientace je dána pořadím přístupu k datům v jednotlivých transakcích
* Rozvrh je uspořádaný (ekvivalentní sériovému), pokud precedenční graf nemá cyklus. Rozvrhy jsou ekvivalentní, pokud precedenční graf je stejný.
* Př.

Obsah obrázku stůl

Popis byl vytvořen automaticky

#### Uzamykací protokoly

* **Slouží pro synchronizaci transakcí**
* Toto se používá v praxi v modulu synchronizace transakcí
* testování uspořádatelnosti jakéhokoliv rozvrhu není to nejlepší pro praxi. Časové náročné.
* Přístup: Konstruovat transakce podle předem daných pravidel tak, že za určitých předpokladů bude každý jejich rozvrh uspořádatelný
* Princip: když pracuji s daty, tak si je zamknu, když už s nimi nebudu pracovat, tak je odemknu
* Do transakčního zpracování přidáme Lock(A) a Unlock(A)
* Implicitně to zamyká na úrovni řádků – kdyby to zamykalo celé tabulky, tak přijdeme o paralelní zpracování
* Existují různé druhy zámků (pro čtení, zápis, …)

Obsah obrázku text

Popis byl vytvořen automaticky

Obsah obrázku text

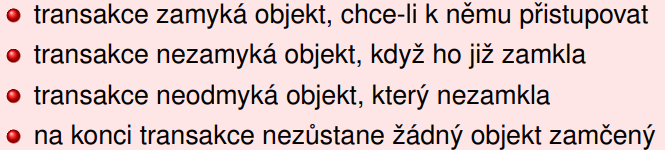
Popis byl vytvořen automaticky

Obsah obrázku text

Popis byl vytvořen automaticky

* Omezíme se dále na tzv. dobře formované transakce, které podporují přirozené požadavky na transakce:

**Dobře formované transakce**

****

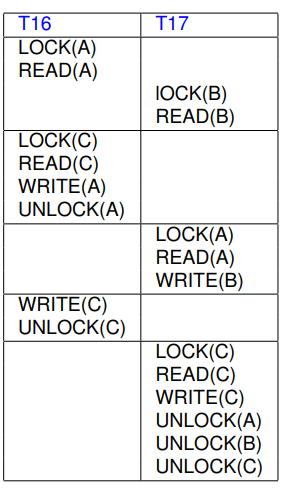
**Dvoufázový uzamykací protokol**

* Uzamykací protokol je množina pravidel, které udávají, kdy transakce bude zamykat a odemykat objekty databáze
* Transakce se skládá ze dvou fází
  + 1. fáze: uzamyká se, nic se neodemyká
  + 2. fáze: od prvního odemknutí se do konce už nic nezamyká

Jestliže všechny transakce v dané množině transakcí T jsou **dobře formované a dvoufázové**, pak každý jejich **legální rozvrh** je uspořádatelný.

* Funguje to i na situace, kdy nevíme, jak se ta transakce bude vyvíjet
* **Opět může nastat uváznutí**
  + Řeší se to tak, že nad DB ještě běží „logika“, která sleduje která transakce čeká na kterou
  + Buduje se graf – která transakce čeká na kterou
  + Když se v grafu detekuje cyklus, tak jsme v deadlocku – vybere se zůčastněný objekt a vyvolá nad vybranou transakcí ROLLBACK – zbytek se rozběhne a deadlock zmizí
  + Druhá možnost je, že se nastaví časový limit pro získání zámku – když čeká moc dlouho, tak na sebe aplikuje ROLLBACK
* př

Obsah obrázku text

Popis byl vytvořen automaticky

Obsah obrázku stůl

Popis byl vytvořen automaticky

Obsah obrázku stůl

Popis byl vytvořen automaticky

Shrnutí

Obsah obrázku text

Popis byl vytvořen automaticky