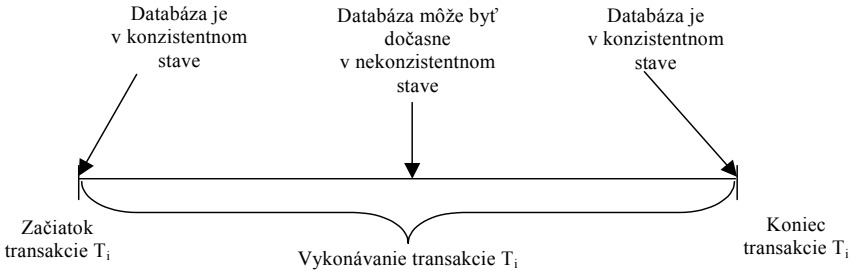
10. Transakčný manažment

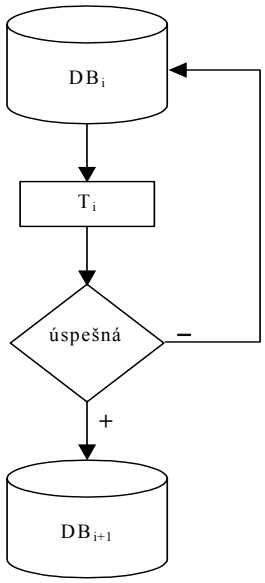
Transakcia je často považovaná aj za základnú jednotku práce databázového systému [Date 96].

Databáza je v konzistentnom stave vtedy a len vtedy, ak sú splnené všetky integritné obmedzenia nad ňou definované (viď kapitola 7). Stav databázy sa mení počas operácií update, insert, delete. Samozrejme, že našou snahou je zabezpečiť, aby databáza nebola v nekonzistentnom stave. Avšak databáza môže byť dočasne v nekonzistentnom tvare počas vykonávania transakcie. Dôležité je, aby bola v konzistentnom tvare po ukončení transakcie.

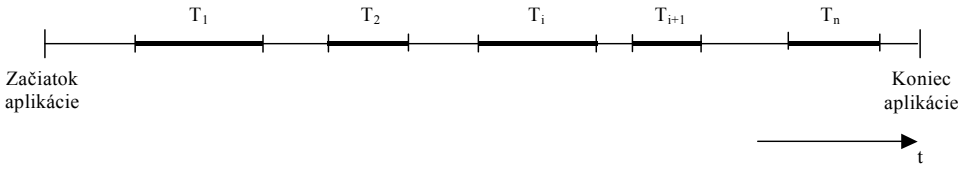


Obr. 10.1 Transakčný model

Transakcia umožní prevod databázy z jedného konzistentného stavu do iného konzistentného stavu – v prípade úspešného ukončenia transakcie, inak databáza zostane v pôvodnom konzistentom stave.

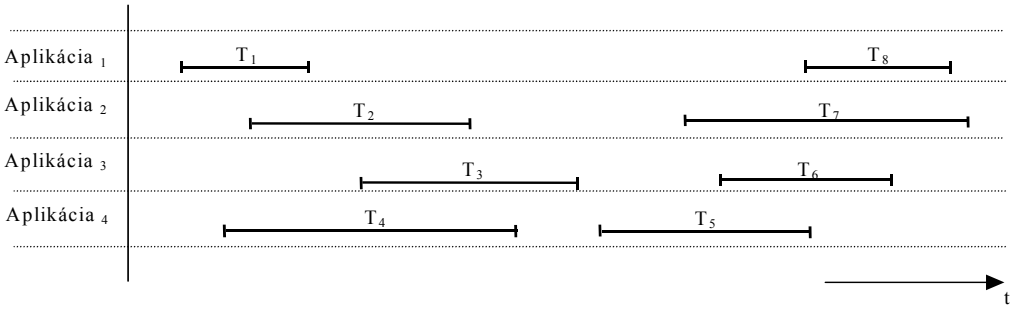


Obr. 10.2 Spracovanie transakcie



Obr. 10.3 Transakcie v aplikácii

V prípade paralelného spracovania viacerých aplikácií, ktoré používajú transakcie môže dôjsť taktiež k paralelnému prístupu k databázam, ktoré sú prehľadávané, alebo menené. Ešte väčší problém môže nastať pri distribuovanom spracovaní, ak používame repliky databáz, alebo ich časti. Databáza je vo vzájomnom konzistentnom stave, ak všetky kópie dátových objektov majú rovnaké hodnoty.



Obr. 10.4 Paralelné vykonávanie transakcií

- *Príklad 10.1 -.Oprava jedného riadku tabuľky*

```
BEGIN WORK
    UPDATE student
      SET st_skupina = "5Z011"
      WHERE os_cislo = 6304
COMMIT WORK
```

- *Príklad 10.2 - Oprava viacerých riadkov tabuľky*

```
BEGIN WORK
    UPDATE student
      SET st_skupina = "5Z011"
      WHERE os_cislo >= 1
        AND os_cislo <1000
COMMIT WORK
```

- *Príklad 10.3 - Viac operácií počas vykonávania transakcie*

```
BEGIN WORK
    INSERT INTO os_udaje(rod_cislo, meno, priezvisko)
      VALUES ("760812/7356", "Ján", "Nový")

    INSERT INTO student(os_cislo, rod_cislo, st_skupina)
      VALUES (125, "760812/7356", "5R012")
COMMIT WORK
```

10.1 Chyby pri spracovaní dotazov

Počas vykonávania transakcie je možný výskyt niekoľkých typov chýb. Základná klasifikácia možných chýb je nasledovná:

1) Porušenie integritných obmedzení
2) Chyby aplikačných programov
3) Zrušenie transakcie
4) Systémové chyby
5) Chyby média

10.2 Definícia transakcie

10.2.1 Úvod

- *Príklad 10.4 -.Jednoduchá transakcia*
  Uvažujme zmenu stavu konkrétneho študenta, ktorú môžme realizovať nasledovným príkazom:

```
UPDATE student
   SET stav = “S”
   WHERE os_cislo = 6304
```

V prípade, že túto operáciu chceme spracovať s transakčnou podporou, je potrebné zapúzdriť tento príkaz medzi príkazy, ohraničujúce začiatok a koniec transakcie.

```
BEGIN WORK
   UPDATE student
      SET stav = “S”
      WHERE os_cislo = 6304
COMMIT WORK
```

Príkazy BEGIN WORK a COMMIT WORK sú príkazy, ktoré ohraničujú začiatok a koniec transakcie v zmysle normy jazyka SQL.

- *Príklad 10.5 -.Viac operácií počas vykonávania transakcie*

```
BEGIN WORK
   INSERT INTO os_udaje(rod_cislo, meno, priezvisko)
      VALUES (“760812/7356”, “Ján”, “Nový”)

   INSERT INTO student(os_cislo, rod_cislo, st_skupina)
      VALUES (125, “760812/7356”, “5R012”)
COMMIT WORK
```

V tomto príklade, ktorý je prevzatý z úvodu kapitoly, je zrejmé ako je možné zapúzdriť niekoľko databázových operácií do jednej transakcie.

10.2.2 Ukončovacie podmienky transakcie

Ako z predchádzajúcich príkladov je zrejmé, uvažovali sme úspešné ukončenie každej z transakcií. Avšak transakcia môže skončiť správne, alebo môže končiť s nejakou chybou. Ak transakcia končí úspešne, hovoríme, že transakcia je potvrdená a toto potvrdenie je zabespečené príkazom **COMMIT**. V opačnom prípade transakcia môže skončiť bez toho, aby transakcia splnila svoje úlohy, v tom prípade hovoríme, že končí príkazom **ABORT** (abnormaly termination). Príkaz COMMIT signalizuje SRBD, že všetky zmeny vykonané transakciou sú zaznamenané v databáze a môžu sa stať viditeľnými, alebo prístupnými iným transakciám, ktoré chcú používať dáta doposiaľ používané touto transakciou. Bod, v ktorom začína potvrdzovanie ukončenia transakcie, je bodom, z ktorého niet návratu. Výsledky potvrdenej transakcie sú týmto perzistentne uložené v databáze a nemôžu byť vrátené späť.

- *Príklad 10.6 -.Transakcia s možnosťou vrátenia databázy do pôvodného stavu.*

```
BEGIN WORK
    WHENEVER ERROR CONTINUE
      UPDATE student
        SET st_skupina = “5Z011”
        WHERE os_cislo >= 1
              AND os_cislo <1000
    WHENEVER ERROR STOP

    IF status=0 THEN
      COMMIT WORK
    ELSE
      ROLLBACK WORK
```

- *Príklad 10.7 -.Viac operácií s možnosťou prerušenia vykonávania transakcie, alebo vrátenia databázy do pôvodného stavu*

```
BEGIN WORK
    SELECT count(*) INTO psc_poc FROM mesto
      WHERE psc = “01069”

    IF psc_poc=0 THEN ABORT

    WHENEVER ERROR CONTINUE
      INSERT INTO os_udaje(rod_cislo, meno, priezvisko, psc)
        VALUES (“760812/7356”, “Ján”, “Nový”, “01069”)
    WHENEVER ERROR STOP

    IF status=0 THEN
      COMMIT WORK
    ELSE
      ROLLBACK WORK

    WHENEVER ERROR CONTINUE
      INSERT INTO student(os_cislo, rod_cislo, st_skupina)
        VALUES (125, “760812/7356”, “5R012”)
    WHENEVER ERROR STOP

    IF status=0 THEN
      COMMIT WORK
    ELSE
      ROLLBACK WORK
```

10.2.3 Charakteristiky transakcie

Je zrejmé, že databázové operácie vykonávajú počas transakcie operácie typu Read a Write nejakých dát. Dátové položky, ktoré sú počas transakcie načítané, tvoria tzv. množinu **ReadSet** (RS), podobne dátové položky, ktoré sú zapisované, množinu **WriteSet** (WS). Tieto dve množiny nemusia byť vzájomne exkluzívne, to znamená, že tie isté položky sa môžu vyskytovať v oboch množinách. Zjednotenie týchto dvoch množín tvorí tzv. základnú množinu položiek – **BaseSet** (BS). Teda platí:

$$\mathbf{BS} = \mathbf{RS} \cup \mathbf{WS}$$

Príklad 10.8 -.Charakteristiky transakcií

```
BEGIN WORK
    SELECT max(os_cislo) INTO max_oc FROM student
      WHERE os_cislo < 5000
        and rocnik = 1

    INSERT INTO student(os_cislo, rod_cislo,st_skupina, stav)
      VALUES (max_oc+1,"760812/7356", "5Z011", "S")

COMMIT WORK
```

V tomto príklade transakcie, je možné uvažovať s nasledovnými množinami charakteristík:

- RS = {os_cislo, rocnik}
- WS = {os_cislo, rod_cislo, st_skupina, stav}
- BS = {os_cislo, rod_cislo, st_skupina, stav}

10.2.4 Formulácia transakcie

Z predchádzajúceho textu intuitívne chápeme význam transakcie. Z hľadiska správnosti a použitia v riadiacich algoritmoch je nevyhnutné formálne definovať transakciu. Označme $O_{ij}(x)$ nejakú operáciu O_j vykonávanú transakciou T_i, ktorá spracováva databázový objekt x. Operácia O_j reprezentuje jednu z dvojice operácií Read, alebo Write. Formálne:

$$O_j \in \{\mathbf{Read, Write}\}$$

Každá z týchto operácií je chápaná ako atomická, to znamená, že jej vykonanie je neprerušiteľné. Označme OS_i ako množinu všetkých operácií vykonávaných v transakcii T_i, potom

$$OS_i = \cup_j O_{ij}$$

Označme N_i ako ukončovací príkaz transakcie T_i, kde:

$$N_i \in \{\mathbf{Abort, Commit}\}$$

Poznámka

V ďalšom texte pre operácie Read, Write, Abort, Commit budeme pre zjednodušenie používať skratky R, W, A, C

Definícia

Množina $\mathbf{OSN_i}$ obsahuje všetky operácie a ukončovacie príkazy transakcie $\mathbf{T_i}$, čiže

$$\mathbf{OSN_i = OS_i \cup \{N_i\}}$$

♦

Definícia - Precedencia operácií

Precedenciu operácií označme $\{_i$ a vyjadruje poradie vykonania operácií transakcie $\mathbf{T_i}$.

♦

Definícia

Čiastočné usporiadanie $\mathbf{P = \{OSN_i, \prec_i\}}$ je usporiadaním prvkov $\mathbf{OSN_i}$, podľa binárnej relácie $\prec_i$ usporiadania všetkých prvkov $\mathbf{OSN_i}$.

♦

Definícia - Transakcia

Transakcia T_i je čiastočné usporiadanie všetkých jej operácií a ukončovacích príkazov, čiže

$\mathbf{T_i = \{OSN_i, \prec_i\}}$,

pričom platí

1. $\mathbf{OSN_i = OS_i \cup \{N_i\}}$
2. pre každú dvojicu operácií $\mathbf{O_{ij}}$, $\mathbf{O_{ik}}$, kde $\mathbf{O_{ij} \in OS_i}$, a $\mathbf{O_{ik} \in OS_i}$, a platí, že ak $\mathbf{O_{ij}=R(x)}$ a $\mathbf{O_{ik}=W(x)}$ pre ľubovoľný dátový objekt **x**, potom buď $\mathbf{O_{ij} \prec_I O_{ik}}$, alebo $\mathbf{O_{ik} \prec_I O_{ij}}$
3. pre $\forall\ \mathbf{O_{ij} \in OS_i}$, $\mathbf{O_{ij} \prec_I N_i}$

Prvá podmienka formálne definuje doménu ako množinu operácií Read a Write vykonávaných v transakcii vrátane ukončovacích príkazov, ktorými môžu byť operácie Commit, alebo Abort.

Druhá podmienka špecifikuje usporiadanie konfliktných dvojíc operácií typu Read a Write.

A tretia podmienka znamená, že každý ukončovací príkaz predchádzajú všetky ostatné operácie.

Je dôležité si uvedomiť, že usporiadanie operácií $\prec_i$ musí byť definované a je aplikačne závislé. Ďalej je veľmi dôležité si uvedomiť, že pre každú konfliktnú dvojicu operácií, je nutné definovať precedenciu $\prec_i$.

Definícia – Konfliktné operácie

Dve operácie $\boldsymbol{O_i(x)}$ a $\boldsymbol{O_j(x)}$ sú konfliktné, ak $\boldsymbol{O_i(x)=Write}$ alebo $\boldsymbol{O_j(x) = Write}$.

♦

Z predchádzajúcej definície vyplýva, že konflikt medzi dvomi operáciami môže nastať vtedy, ak aspoň jedna z operácií je operácia Write a obe operácie pristupujú k tomu istému dátovému objektu.

□ *Príklad 10.9*

Uvažujme jednoduchú transakciu *T*, ktorá obsahuje v poradí nasledovné operácie:

```
Read(x)
Read(y)
x:=x+y
Write(x)
COMMIT
```

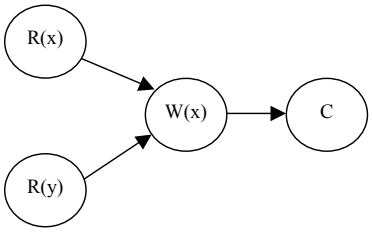
Potom zápis transakcie, podľa formálneho zápisu je nasledovný:

OSN = {R(x), R(y), W(x),C}
≺ = {(R(x), W(x)),(R(y),W(x)),(W(x),C),(R(x),C),(R(y),C)}

kde pre dvojicu operácií $O_i \prec O_j$ používame zápis (O_i,O_j).

Výhodné pri definícií transakcie je aj používanie grafickej reprezentácie, pomocou acyklických grafov. Potom transakcia môže byť špecifikovaná ako graf, ktorého vrcholy sú operácie transakcie a jeho hrany reprezentujú poradie vykonania pre každú dvojicu operácií.

Príklad 10.11
Transakciu z príkladu 10.10 môžeme reprezentovať nasledovným grafom.



Obr. 10.5 Orientovaný acyklický graf reprezentujúci transakciu

V grafe nie je nutné kresliť hrany, pre ktoré existuje tranzitivita medzi vrcholmi.

10.3 Vlastnosti transakcie

Pretože transakcia musí zabezpečovať celistvosť databázy a spoľahlivosť spracovania, sú definované štyri vlastnosti transakcie:

1. Atomicita
2. Konzistencia
3. Izolovanosť
4. Perzistencia

V anglosaskej literatúre je pre vlastnosti transakcie používaná skratka **ACID** (Atomicity, Consistency, Isolation, Durability).

10.3.1 Atomicita

10.3.2 Konzistencia

Konzistencia transakcie jednoducho znamená jej správnosť. Čiže transakcia je program, ktorý transformuje databázu z jedného konzistentného stavu do nového konzistentného stavu. Existujú štyri úrovne konzistencie [Gray76]:

- Transakcia *T* má tretiu úroveň konzistencie, ak:
  1) *T* neprepisuje dáta inej transakcie
  2) *T* nepotvrdzuje žiadnu zmenu v databáze pokiaľ nie sú ukončené všetky zmeny transakcie (kým nenarazí na príkaz Commit)
  3) *T* nečíta dáta, ktoré používa iná transakcia
  4) Iné transakcie nečítajú dáta používané *T* pred jej ukončením
- Transakcia *T* má druhú úroveň konzistencie, ak:
  1) *T* neprepisuje dáta inej transakcie
  2) *T* nepotvrdzuje žiadnu zmenu v databáze pokiaľ nie sú ukončené všetky zmeny transakcie (kým nenarazí na príkaz Commit)
  3) *T* nečíta dáta, ktoré používa iná transakcia
- Transakcia *T* má prvú úroveň konzistencie, ak:
  1) *T* neprepisuje dáta inej transakcie
  2) *T* nepotvrdzuje žiadnu zmenu v databáze pokiaľ nie sú ukončené všetky zmeny transakcie (kým nenarazí na príkaz Commit)
- Transakcia *T* má nultú úroveň konzistencie, ak

  T neprepisuje dáta inej transakcie

Z uvedeného je zrejmé, že vyššie úrovne konzistencie dedia podmienky nižších úrovní.

10.3.3 Izolovanosť

Príklad 10.12

Uvažujme jednoduchú transakciu T_1, ktorá obsahuje nasledovné kroky:

```
Read(x)
x:=x+1
Write(x)
COMMIT
```

a jednoduchú transakciu T_2, ktorá obsahuje nasledovné kroky:

```
Read(x)
x:=x+1
Write(x)
COMMIT
```

Exituje niekoľko možností spracovania operácií v týchto transakciách. Jedna z nich môže byť:

```
T1: Read(x)
T1: x:=x+1
T1: Write(x)
T1: COMMIT
T2: Read(x)
T2: x:=x+1
T2: Write(x)
T2: COMMIT
```

Z uvedeného je zrejmé, že transakcie sú spracované jedna po druhej. V prípade, že na začiatku spracovania bola hodnota objektu x=100, po vykonaní prvej transakcie je hodnota x=101 a po vykonaní druhej transakcie x=102.

Ak prebieha spracovanie transakcií paralelne, napr. nasledovne:

```
T1: Read(x)
T2: Read(x)
T1: x:=x+1
T1: Write(x)
T2: x:=x+1
T2: Write(x)
T1: COMMIT
T2: COMMIT
```

v tomto prípade transakcia T_2 číta hodnotu x=100, je to nesprávna hodnota, pretože prvá transakcia inkrementuje túto hodnotu.

10.3.4 Perzistencia

Perzistencia znamená, že všetky potvrdené zmeny musia byť zaznamenané v databáze a prístupné aj po systémových haváriách.

10.4 Typy transakcie

Transakcie môžeme klasifikovať do rôznych tried, podľa toho aké techniky sa pri ich spracovaní používajú. Nižšie uvedený prehľad samozrejme nie je úplný a v literatúre sa môžeme stretnúť s inými, resp. ďalšími klasifikáciami. Základný pohľad na transakcie je podľa typu aplikácie, časového hľadiska, štruktúry, atď.

1. Podľa aplikácie
   - **centralizované** – používajú sa pri centralizovaných DBS a reprezentujú základný typ transakcie
   - **distribuované** –
2. Podľa doby trvania
   - **on-line** (krátke) –
   - **batch** (dlhé) –
3. Podľa štruktúry
   - **ploché** –
   - **hniezdené** –
4. Podľa usporiadania operácií Read a Write – tento typ klasifikácie transakcií nám umožní nazerať na transakciu z hľadiska poradia vykonávania operácií Read a Write a poskytuje nasledovné možnosti, podľa ktorých je možné pripraviť špecializované algoritmy ma zabezpečenie paralelizmu, resp. obnovy databázy.
   - **všeobecné** (general) – operácia Read predchádza operáciu Write nad tým istým objektom, bez ohľadu na špeciálne poradie ostatných operácií
     T= {R(x), R(y), W(y), R(z), W(z), W(w), C}
   - **dvojkrokové** – všetky operácie Read predchádzajú všetky operácie Write
     T= {R(x), R(y), R(z), W(y), W(z), W(w), C}
   - **obmedzené** (Read before Write) – Každý objekt musí byť načítaný predtým, ako bol zapísaný
     T= {R(x), R(y), W(y), W(x), R(w), R(z), W(z), W(w), C}
   - **dvojkrokové obmedzené** – všetky operácie Read predchádzajú všetky operácie Write a každý objekt musí byť načítaný pred tým ako bude zapísaný.
     T= {R(x), R(y), R(w), R(z), W(y), W(x), W(z), W(w), C}
   - **akčné** – každá dvojica operácií Read a Write nad tým istým objektom sa vykonáva atomicky
     T= {[R(x), W(x)], [R(y), W(y)], [R(w), W(w)], [R(z), W(z)], C}

10.5 Logický žurnál

Logický žurnál obsahuje informácie, ktoré umožnia obnoviť konzistenciu databázy, v prípade neukončenia transakcie. Z toho vyplýva, že logický žurnál je obyčajne sekvenčný súbor s premenlivou dĺžkou záznamu umiestnený v príslušnom databázovom priestore. Samozrejme, že môže byť implementovaný ako špecializovaná relácia v databázovom priestore.

Záznam typického logického žurnálu by mal obsahovať:

- ID transakcie
- ID činnosti – CHECK point
  Begin Work
  Abort
  Commit
  operácia
- ID operácie – Insert
  Delete
  Update
  Select
- čas udalosti
- záznam PRED operáciou
- záznam PO operácii
- záznam z CHECK Pointu

ID transakcie umožní identifikovať všetky záznamy patriace k danej transakcii, ktoré sú rozptýlené v logickom žurnáli, v poradí v akom sa vykonávali. ID činnosti nám umožní rozpoznať, či záznam sa týka kontrolného bodu, alebo samotnej transakcie a v prípade transakcie určuje počiatočný, prípadne ukončovací bod transakcie, alebo nás informuje o tom, že je monitorovaná databázová operácia. ID operácie nám v prípade databázovej operácie umožní analyzovať, aký typ operácie je monitorovaný v príslušnom zázname log. žurnálu. Záznam pred operáciou (BEFORE IMAGE) obsahuje záznam, resp. stránku, s hodnotami pred deštruktívnou operáciou a záznam po operácii (AFTER IMAGE) obsahuje stav záznamu, resp. stránky, po vykonaní deštruktívnej operácie.

V prípade zaznamenávania týchto informácií dokážeme zistiť poradie vykonávania transakcií a aj obnoviť databázu v prípade neúspešného ukončenia transakcie.

10.6 Metódy spracovania transakcií

Pravidlo LAR (Log Ahead Rule):

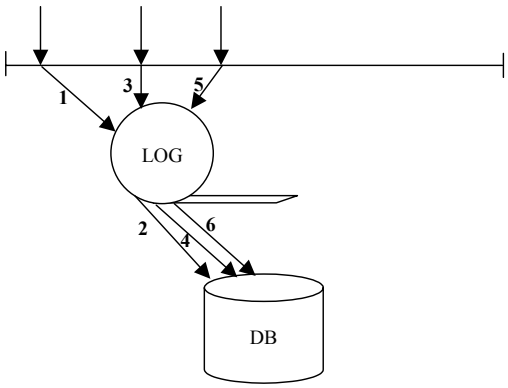
Databázový objekt môže byť modifikovaný až keď zmeny sú zaznamenané v logickom žurnále.

Toto pravidlo dovolí aby transakcia mohla vykonať operáciu UNDO v prípade prerušenia vykonávania transakcie. Keďže všetky zmeny sú najskôr zaznamenané v logickom žurnále, nemôže sa stať, že by sme neboli schopní vrátiť databázu do pôvodného stavu.

Vo všetkých systémoch programové vybavenie zabezpečujúce modifikáciu databázy uchováva zmeny v buffroch hlavnej pamäte a z týchto buffrov sú dáta presúvané do databázy, kde sa stávajú perzistentnými. Pokiaľ nie sú data prenesené z buffrov do databázy, nie je zaistená možnosť obnovy databázy. Obsah buffrov sa do databázy presúva podľa používaných metód spracovávania transakcií.

10.6.1 Metóda priameho zápisu

Metóda priameho zápisu je charakteristická tým, že pri každej operácii updatujeme logický žurnál a po modifikácii log. žurnálu aj samotnú databázu.



Obr. 10.6 Metóda priameho zápisu

Dôsledkom tejto metódy pri úspešnom ukončení transakcií je to, že po poslednej operácii je modifikovaná aj celá databáza, ale v prípade, že transakcia skončí neúspešne, pri procese obnovy z logického žurnálu sa musia zistiť všetky vykonané operácie a znovu zmeniť databázu tak, aby bola v stave ako na začiatku vykonania transakcie.

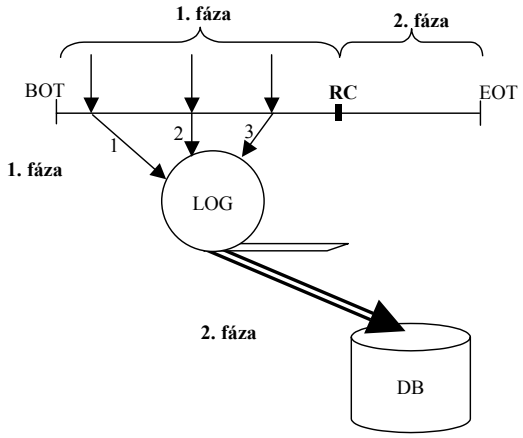
Algoritmus obnovy:

1. Nájdi v logickom žurnáli začiatok transakcie T a zisti čas začiatku.
2. Nájdi všetky transakcie T_i započaté alebo ukončené po začiatku transakcie T.
3. Vráť všetky, aj už ukončené transakcie T, T_i do pôvodného stavu v opačnom poradí ako boli vykonávané.
4. Zopakuj všetky transakcie T_i.

10.6.2 Metóda dvojfázového potvrdzovacieho protokolu (2PhC)

Táto metóda spočíva v tom, že transakcia sa rozdelí na 2 fázy. V prvej fáze sa všetky zmeny zapisujú do logického žurnálu. V druhej fáze sa prepisujú zmeny z logického žurnálu do databázy.

Transakcia je rozdelená signálom RC (Ready to Commit). Až keď transakcia vyšle signál RC, vykonávanie prejde do druhej fázy.



Obr. 10.7 Metóda dvojfázového potrdzovacieho protokolu

Pre túto metódu platia nasledovné **pravidlá**:

1. transakcia nesmie meniť hodnoty objektov DB, pokiaľ nie je potvrdená
2. transakcia nemôže byť potvrdená, pokiaľ nie sú vytvorené príslušné záznamy v logickom žurnále.

10.6.3 Metóda kontrolného bodu (CHECK point)

Samotná metóda spočíva v dvoch fázach, prvá fáza predstavuje periodické zaznamenávanie informácií o stave databázového systému, v tzv. kontrolných bodoch. Druhá fáza sa používa len pre obnovu databázy po systémovej chybe.

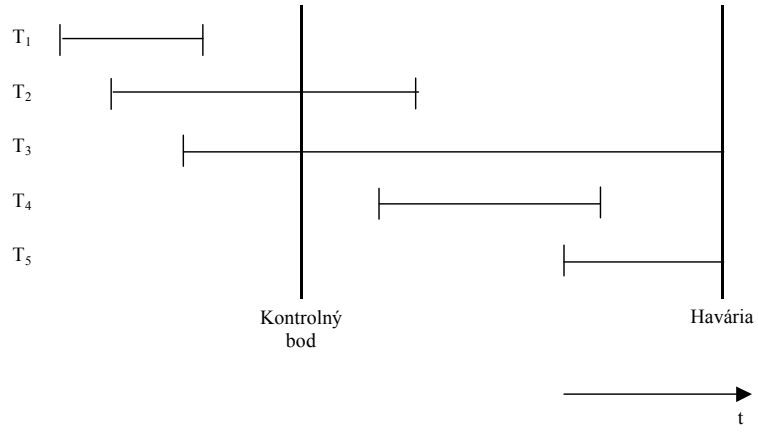
V prípade výskytu udalosti, ktorá vyvolá tzv. **kontrolný bod** sa vykonajú nasledovné činnosti:

- všetky zmeny v databáze sú prepísané z buffrov do logického žurnálu, resp. do databázy (databáza je v konzistentom stave)
- do logického žurnálu sa zaznamená tzv. záznam kontrolneho bodu, ktorý obsahuje zoznam všetkých bežiacich transakcií v danom okamihu

Pri tejto metóde sa môžeme stretnúť s piatimi **typmi transakcií**:

1. transakcia, ktorá začala aj skončila pred posledným kontrólným bodom
2. transakcia, ktorá začala pred posledným kontrolným bodom a skončila pred haváriou systémom
3. transakcia, ktorá začala pred posledným kontrolným bodom, ale pre haváriou ešte nebola ukončená
4. transakcia, ktorá začala po kontrolnom bode a skončila pred haváriou
5. transakcia, ktorá začala po kontrolnom bode, ale ešte neskončila

Príklad 10.13
Zobrazme transakcie, ktoré reprezentujú jednotlivé typy transakcií z predchádzajúceho textu.



Obr. 10.8 Typy transakcií pri metóde kontrolnéh bodu

Algoritmus obnovy databázy po havárii:

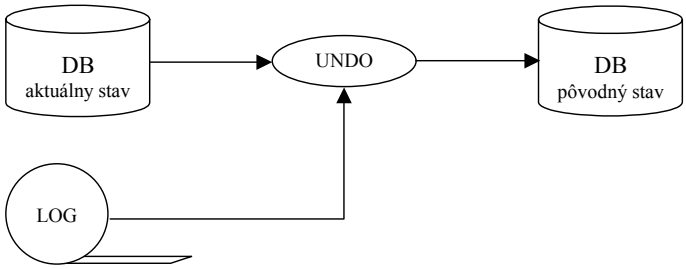
1. vyhľadaj posledný záznam o kontrolnom bode v logickom žurnáli
2. vytvor dva prázdne zoznamy – REDO - list a UNDO – list
3. zapíš všetky transakcie z posledného záznamu kontrolného bodu do UNDO–listu
4. čítaj záznamy z logického žurnálu:

   Ak je to BOT – potom pridaj transkaciu do UNDO – listu,

   ak EOT , potom presuň transakciu z UNDO do REDO – listu
5. opakuj krok 4, až kým nenarazíš na koniec logickéh žurnálu
6. spusti operáciu UNDO pre všetky transakcie, ktoré sa nachádzajú v UNDO-liste
7. spusti operáciu REDO pre všetky transakcie, ktoré sa nachádzajú v REDO-liste

Operácia UNDO:

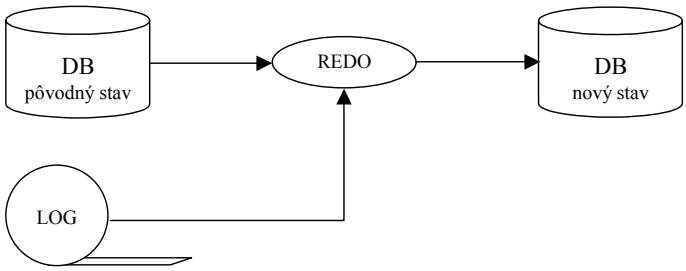
Operácia UNDO vráti všetky zmenené záznamy v DB do pôvodného stavu.



Obr. 10.9 Operácia UNDO

Operácia REDO:

Operácia REDO zopakuje všetky operácie vykonávané vrámci transakcie.



Obr. 10.10 Operácia REDO

Pri obnove databázy je možné s výhodou využiť pravidlo idenpotencie, ktoré hovorí o tom, že ak bol záznam niekoľko krát menený, tak pri obnove stačí vykonať obnovu do stavu, ktorý bol pred prvou operáciou nad týmto záznamom, alebo v prípade opakovania operácií stačí zopakovať len poslednú operáciu, ktorá prepisovala tento záznam.

Pravidlo idenpotencie:

UNDO (UNDO (UNDO (x)x)x) = UNDO(x)

REDO (REDO (REDO (x)x)x) = REDO (x)

10.7 Obnova databázy pri poškodení média

Databázu pri poškodení média je možné obnoviť len v prípade, že v systéme je zabezpečené vytváranie pravidelných kópií (Backup Copy), s pomocou ktorých je možné obnoviť databázu aj pri poškodení média. Samozrejme je potrebné, aby sme pri obnove mali k dispozícii celý log. žurnál, ktorý je vytváraný od vytvorenia poslednej kópie databázy. S pomocou archívnej kópie a logických žurnálov a použitím metódy kontrólneho bodu dokážeme obnoviť databázu do stavu, v akom bola v okamihu havárie. Samozrejme predpokladom je, že logikcké žurnály a kópia databázy sú umiestnené na inom médiu, ako je databáza.