9. Transakční zpracování

9.1. Transakce 3
9.1.1. Vlastnosti transakce 3
9.1.2. Stavy transakce 4
9.2. Transakce v SQL 6
9.3. Zotavení po chybách a poruchách 10
9.3.1. Zotavení využívající žurnálu 13
9.3.2. Stínové stránkování 20
9.3.3. Poruchy energeticky nezávislé paměti 22
9.4. Řízení souběžného přístupu 23
9.4.1. Sériové a uspořádatelné plány 23
9.4.2. Zajištění uspořádatelnosti 28
9.4.3. Uzamykací protokoly 28
9.4.4. Protokoly založené na časových razítkách 32
9.4.5. Další typy protokolů 35
9.4.6. Řešení problému zablokování 36
9.4.7. Zotavení souběžných transakcí 37

9.5. Zotavení a souběžný přístup v SQL 38
Literatura 41

9.1. Transakce

Transakce (databázová) **je jednotka provádění programu, která zpřístupňuje, případně i modifikuje data v databázi.**

9.1.1. Vlastnosti transakce

- ***ACID vlastnosti***

➢ ***Atomičnost (Atomicity)***

Atomičnost transakce **znamená, že buď je provedena celá transakce nebo žádná z databázových operací, které ji tvoří.**

➢ ***Konzistence (Consistency)***

Konzistence transakce **znamená, že izolovaná transakce zachovává konzistenci databáze.**

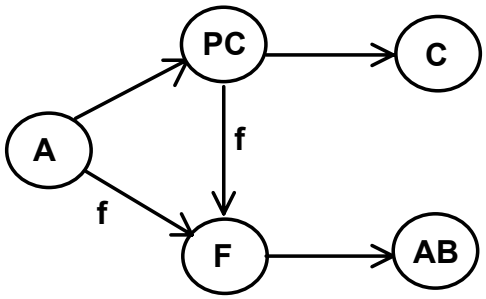
➢ ***Izolace (Isolation)***

Izolace transakce **znamená, že i při souběžném běhu transakcí SŘBD zajistí, že pro každou dvojici souběžných transakcí T_i a T_j se T_i jeví, že T_j skončila dříve, než T_i zahájila provádění nebo T_j zahájila provádění až poté, co T_i skončila.**

➢ ***Trvalost (Durability)***

Trvalost transakce **znamená, že poté, co transakce úspěšně skončí, budou mít všechny změny v databázi, které transakce provedla, trvalý charakter a to i při výpadku systému.**

9.1.2. Stavy transakce



Aktivní **(A) - počáteční stav, transakce v něm setrvává po dobu provádění.**

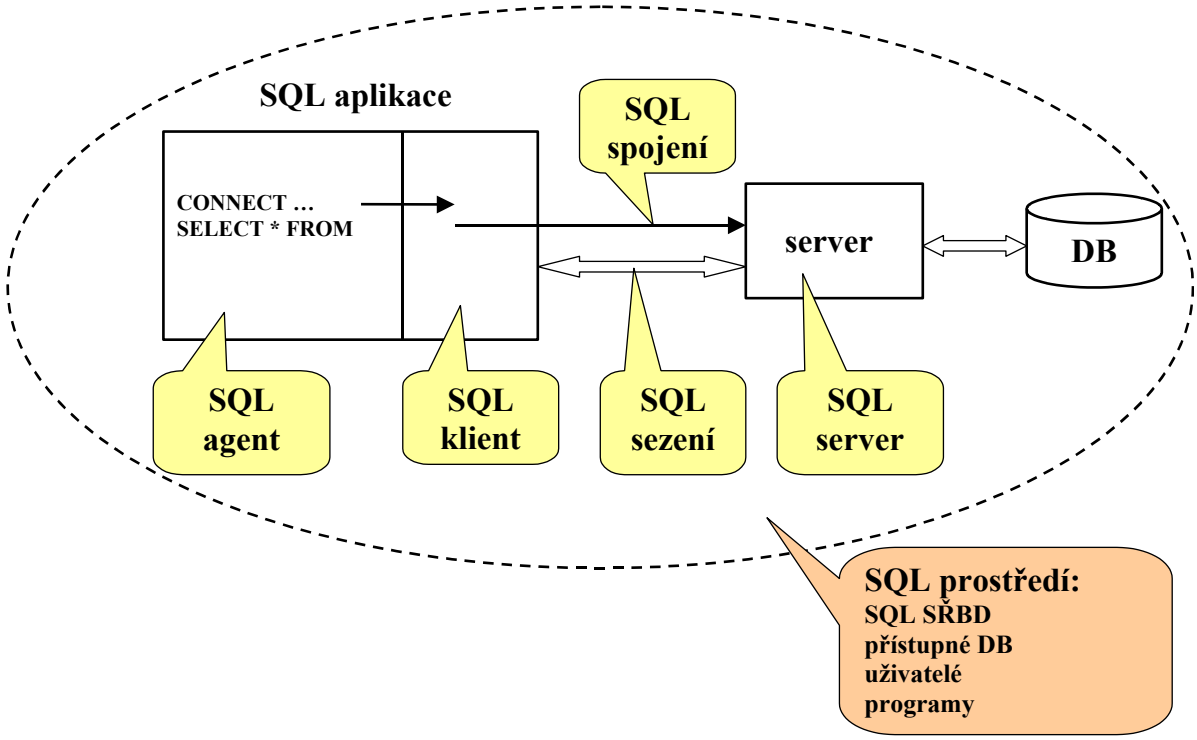
Částečně potvrzená **(PC) – po provedení posledního příkazu.**

Chybový stav **(F) – po zjištění, že normální provádění není dál možné.**

Zrušená (AB) – poté, co byly změny v databázi provedené transakcí anulovány (operace rollback), databáze bude ve stavu před zahájením transakce.

Potvrzená (C) – po úspěšném dokončení transakce.

9.2. Transakce v SQL



- *Zahájení sezení*

```
CONNECT TO {DEFAULT |
            string1 [AS string2] [USER string3]}
```

  - string1 je xxx, string 3 je yyy
  - může být iniciováno několik spojení, pouze jedno je aktivní

```
SET CONNECTION TO {DEFAULT | string}
```

- *Ukončení spojení*

  ➢ *explicitní*

```
 DISCONNECT {DEFAULT | CURRENT | ALL | string}
```

  ➢ *implicitní*
  - po posledním příkazu SQL v aplikaci

- *SQL transakce*
  - operace SQL jsou atomické

  ➢ *Zahájení transakce*

  - implicitní, SQL agent provádí příkaz SQL inicializující transakci (ne CONNECT, COMMIT, DECLARE CURSOR, …) a nemá transakci zahájenou
  - transakce nelze zanořovat (tzv. „plochý“ (flat) model), jedinou implicitní zanořenou úrovní jsou samotné příkazy SQL.

  ➢ *Ukončení transakce*

```
COMMIT [WORK]
ROLLBACK [WORK]
```

- Částečný rollback (není v SQL/92)
  - umožňuje vrátit část transakce
  - příkazy SAVEPOINT p, ROLLBACK p

Př)

```
příkaz1_transakce
SAVEPOINT p1
příkaz2_transakce
SAVEPOINT p2
příkaz3_transakce
ROLLBACK p2
příkaz4_transakce
```

ROLLBACK p1

- **Oracle: ano, SQLBase: ano**

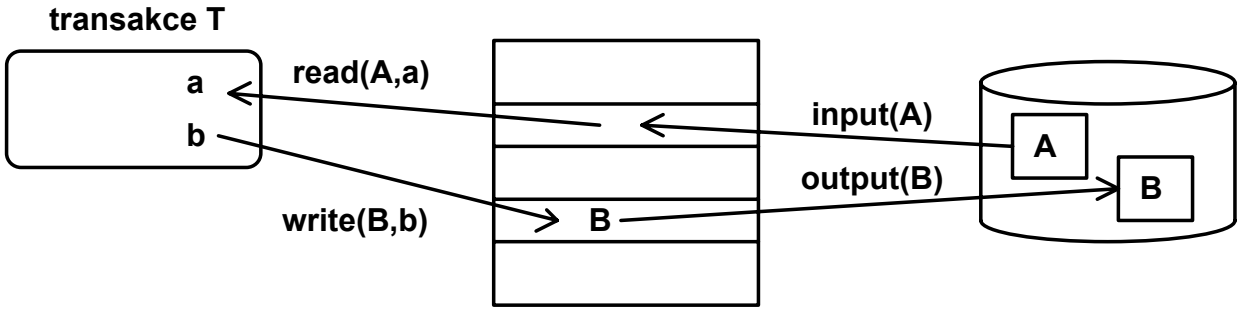
9.3. Zotavení po chybách a poruchách

Zotavení (recovery) **znamená obnovení konzistentního stavu databáze po výpadku systému.**

- ***Klasifikace pamětí***
  - ***energeticky závislá (volatile)***
  - ***nezávislá (nonvolatile)***
  - ***stabilní (stable)***
- ***Klasifikace výpadků***
  - ***výpadek transakce***
    - **logická chyba (např. data nenalezena)**
    - **systémová chyba (např. deadlock)**
  - ***zhroucení systému***
  - ***porucha disku***

V dalším budeme předpokládat pouze jednu transakci běžící v daném okamžiku.

- **Model přístupu transakce k datům**



a, b ... lokální proměnné transakce

A,B ... datové položky z databáze

input (B) – načte blok B z disku do vyrovnávací paměti

output (B) – zapíše blok B z vyrovnávací paměti na disk

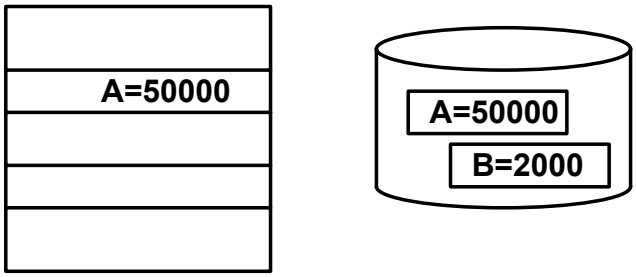
read (A, a) – přiřadí hodnotu *A* do lokální proměnné *a*

write (A, a) – přiřadí hodnotu *a* datové položce *A* ve vyrovnávací paměti

Poznámka: Databázi budeme v této části chápat jako tvořenou jednak bloky na disku, jednak bloky ve vyrovnávací paměti.

Př) Spořitelna - převod částky 10000Kč z účtu A na účet B

```
T:  read (A, a)
        a = a - 10000
        write (A, a)
        read (B, b)
        b = b + 10000
        write (B, b)
```



Předpokládejme, že při operaci *read (B, b)* se uloží na disk z vyrovnávací paměti modifikovaný blok s *B*.

V konzistentním stavu platí, že součet stavů na účtech A a je konstantní.

- ***Zotavení a atomičnost transakce***

Př) Výpadek systému mezi *output(A)* a *output(B)* v předchozím příkladě (na disku je nová hodnota *A*, ale původní hodnota *B*). Provést v rámci zotavení transakci T znovu nebo neprovádět nic?

K zajištění atomicity transakce a trvalosti změn je nutné před modifikací databáze uložit do stabilní paměti informace o modifikací.

Jinou možností je, že transakce pracuje se „svou kopií“ databáze.

9.3.1. Zotavení využívající žurnálu

Žurnál (log file) **- je posloupnost záznamů žurnálu (log record) zaznamenávající všechny modifikace databáze.**

- ***Typy záznamů žurnálu***

  < T_i , start > - transakce T_i zahájila provádění.

  < T_i , X_i , H_1 , H_2 > - transakce T_i provedla zápis datové položky X_i , H_1 značí původní a H_2 novou hodnotu položky X_i.

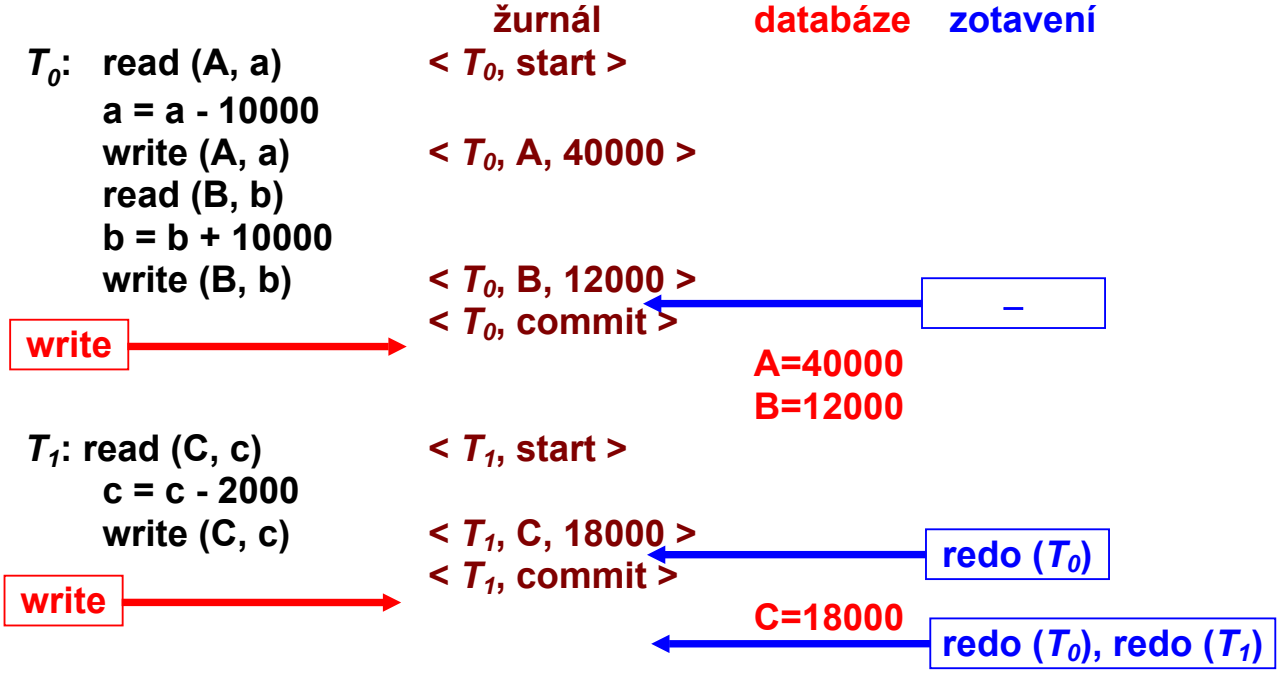
  < T_i , commit > - transakce T_i potvrdila změny (skončila úspěšně).

  < T_i , abort > - transakce T_i byla zrušena.

- ***Odložená modifikace databáze***

  Atomičnosti je dosaženo zaznamenáváním modifikací do žurnálu, ale provedení zápisů je potlačeno, než se transakce dostane do stavu částečného potvrzení. Zotavení používá proceduru *redo (T_i)*.

Př) T_0 viz T, T_1 výběr 2000 Kč z účtu C (20000)



redo (T_i) je zotavovací procedura, která na základě informací v v žurnále nastaví všechny datové položky aktualizované transakcí T_i na novou hodnotu.

- *Okamžitá modifikace databáze*

  Umožňuje provádět modifikace databáze, když je transakce v aktivním stavu (tzv. nepotvrzené modifikace). V případě výpadku je potřeba u nedokončených transakcí vrátit původní hodnoty a u dokončených znovu zapsat nové hodnoty. Schéma zotavení používá procedury *undo(T_i)* a *redo(T_i)*.

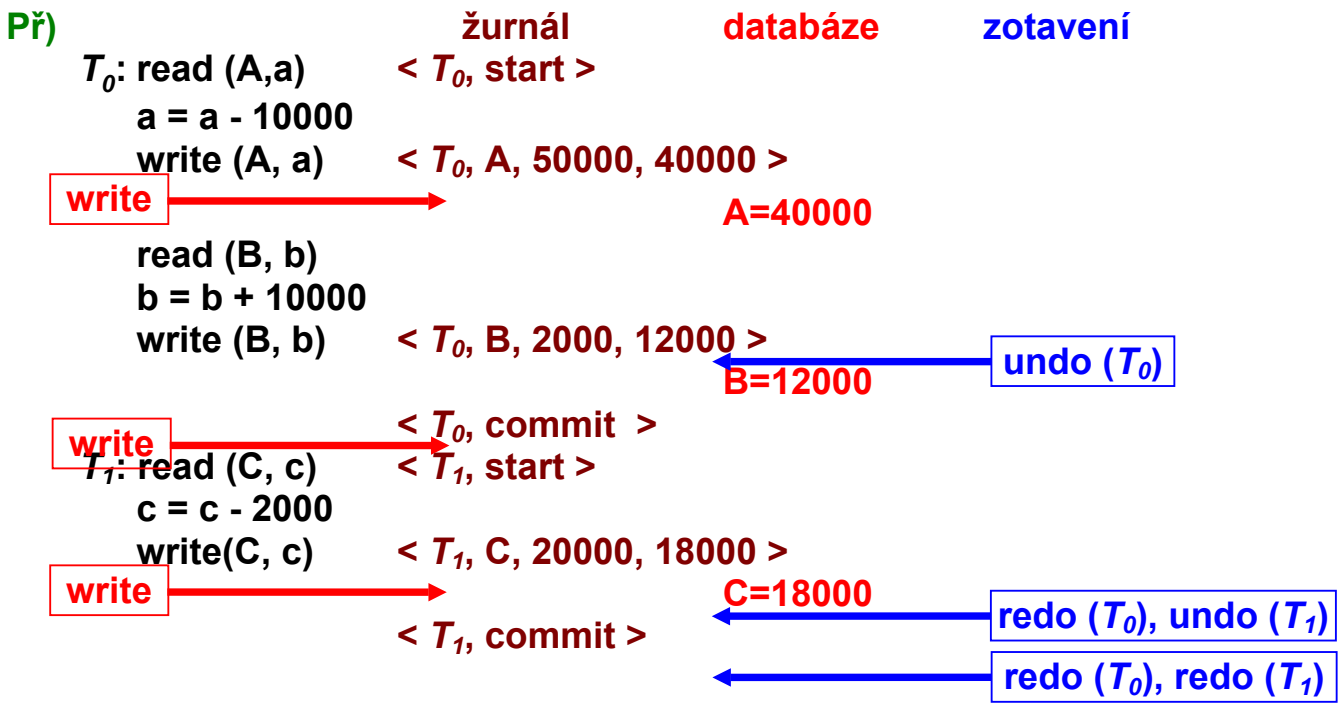


Schéma zotavení: Na transakci T_i se aplikuje zotavovací procedura:

- *undo (T_i)*, jestliže žurnál obsahuje < T_i , start>, ale ne < T_i , commit>
- *redo (T_i)*, jestliže žurnál obsahuje < T_i , start> i < T_i , commit>

- *Kontrolní body*

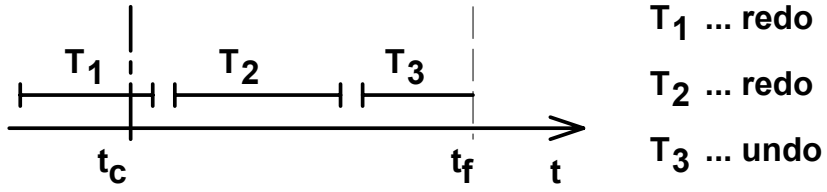
  Kontrolní bod (checkpoint) je periodické ukládání vyrovnávacích pamětí žurnálu a databáze na disk z důvodu snížení režie související se zotavením po výpadku.

  Postup:

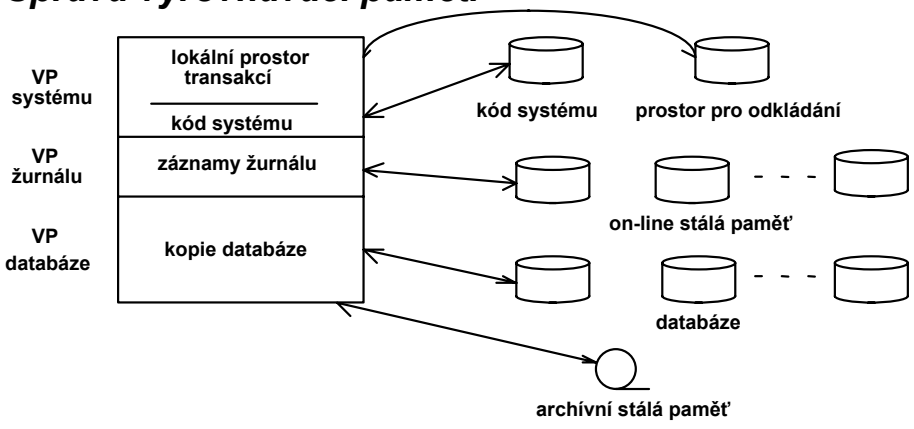
  1. uložení všech záznamů žurnálu z hlavní paměti
  2. uložení všech modifikovaných bloků DB z vyrovnávací paměti na disk
  3. uložení záznamu < checkpoint, T_1, T_2, ... > do stabilní paměti

  Schéma zotavení:

  1. nalezení množiny transakcí *T*, které probíhaly nebo byly zahájeny po posledním kontrolním bodu
  2. aplikace zotavovacích procedur *redo*(T_i) a *undo*(T_i) na každou transakci $T_i \in T$ podle použité techniky



- *Správa vyrovnávací paměti*



  - datové položky se nezapisují přímo na disk (viz operace *write*)
  - záznamy žurnálu se nezapisují okamžitě do stabilní paměti

  Zásady:

  - transakce T_i se dostává do stavu potvrzení (C) až po uložení záznamu < T_i, commit > do stabilní paměti
  - před záznamem < T_i, commit > musí být do stabilní paměti uloženy všechny záznamy žurnálu týkající se transakce T_i

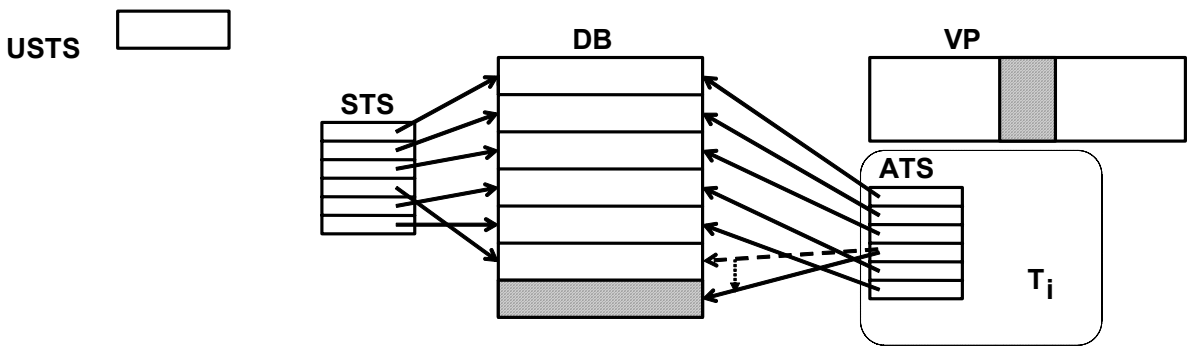
- **před uložením bloku dat do databáze musí být uloženy všechny záznamy žurnálu, týkající se daného bloku (tzv. pravidlo WAL (write-ahead logging))**

9.3.2. Stínové stránkování

- ***Podstata***

  Existence dvou tabulek stránek:

  - ***stínová tabulka stránek (STS)*** **- platná pro databázi, nemění se při provádění transakce,**
  - ***aktuální takulka stránek (ATS)*** **- platná pro danou transakci, mění se při provádění transakce**



- *Použití ATS pro zachování vlastností transakce*
  - změna ATS při prvé operaci *write* transakce do stránky:

1. *input* (není-li stránka ve VP)
2. najdi volnou stránku v DB
3. modifikuj ATS pro novou stránku
4. zapiš hodnotu do stránky ve VP

  - ukončení transakce:

1. uložení modifikovaných stránek VP na disk
2. uložení ATS na disk
3. změna hodnoty USTS (uložená ATS se stane STS)

- *Zotavení*
  - implicitní (nic se neprovádí) - návrat ke stavu před zahájením transakce

Nevýhody: fragmentace dat, sběr nepoužívaných stránek, komplikace při souběžném přístupu.

9.3.3. Poruchy energeticky nezávislé paměti

Archivace (backup) je ukládání obsahu databáze do stabilní paměti, typicky v pravidelných intervalech.

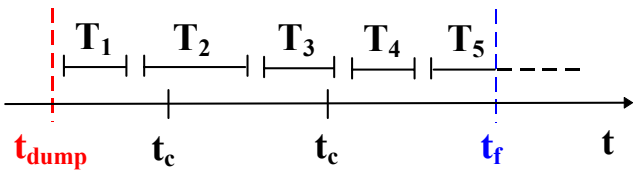
Obnova (restore) je obnovení databáze do stavu před poslední archivací.

- *Postup při archivaci*

1. uložení záznamů žurnálu do stabilní paměti,
2. uložení modifikovaných bloků DB z paměti na disk
3. uložení DB z disku do stabilní paměti
4. vytvoření záznamu < *dump* > v žurnálu ve stabilní paměti

- *Zotavení*

1. obnovení DB
2. zotavení od okamžiku archivace



9.4. Řízení souběžného přístupu

Schéma řízení **je souhrn pravidel použitých k zajištění souběžného přístupu.**

9.4.1. Sériové a uspořádatelné plány

Plán ***(rozvrh)*** **udává chronologické pořadí provádění instrukcí souběžných transakcí.**

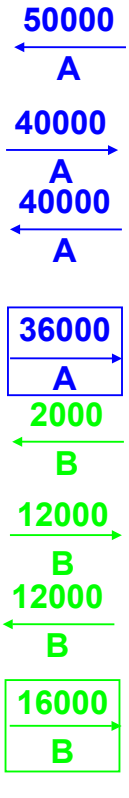
Sériový plán **- instrukce jedné transakce bezprostředně za sebou.**

Př) T_0 viz předchozí příklady, T_1 zvýší účet B o 10% A.

T_0
read (A, a_0)
$a_0=a_0-10000$
write(A,a_0)
read(B,b_0)
$b_0=b_0+10000$
write(B,b_0)

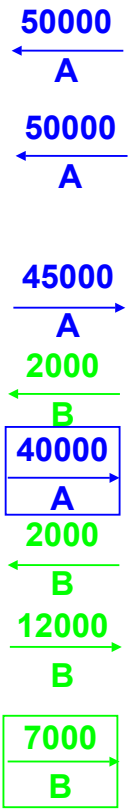
T_1
read (A, a_1)
pom=0.1*a_1
$a_1=a_1$-pom
write(A,a_1)
read(B, b_1)
$b_1=b_1$+pom
write(B,b_1)

- pro *n* transakcí *n!* sériových plánů
- sériový plán zachovává konzistenci
- plány, které nejsou sériové, mohou porušit konzistenci



T_0	T_1
read (A, a_0)	
$a_0=a_0-10000$	
write(A,a_0)	
	read (A, a_1)
	pom=0.1*a_1
	$a_1=a_1$-pom
	write(A,a_1)
read(B,b_0)	
$b_0=b_0+10000$	
write(B,b_0)	
	read(B, b_1)
	$b_1=b_1$+pom
	write(B,b_1)

- kritické jsou operace read a write



T0	T1
read (A, a_0)	
$a_0=a_0-10000$	
	read (A, a_1)
	pom=0.1*a_1
	$a_1=a_1$-pom
	write(A,a_1)
	read(B, b_1)
write(A,a_0)	
read(B,b_0)	
b0=b0+10000	
write(B,b_0)	
	$b_1=b_1$+pom
	write(B,b_1)

- *Typické problémy, které je třeba řešit při řízení souběžného přístupu*
  - *Ztráta aktualizace (přepis jinou transakcí)*

T_1	T_2
…	
read (Q,q)	…
…	read (Q,q)
write (Q,q)	…
…	write (Q,q)
…	…

  - *Závislost na potvrzení (jinou transakcí) načtené hodnoty*

T_1	T_2
	…
…	write (Q,q)
read (Q,q)	…
…	ROLLBACK

  - *Přepis nepotvrzené hodnoty*

T_1	T_2
	…
…	write (Q,q)
write (Q,q)	…
…	ROLLBACK

  - *Nekonzistentní analýza*

    Př) T_1 zobrazí součet několika účtů, T_2 mezitím provede převod.

- *Uspořádatelné plány*
  - *Binární relace na množině souběžných transakcí „je v konfliktu“*

T_i	T_j
:	:
I_x	:
:	I_y
:	:

I_x *je _v_konfliktu_s* I_y

I_x \ I_y	read	write
read	N	A
write	A	A

Instrukce I_x a I_y jsou ***konfliktní***, přistupují-li ke stejnému databázovému objektu a alespoň jednou z nich je *write*.

Plány *S a S'* se nazývají ***ekvivalentní vzhledem ke konfliktům,*** lze-li plán *S* transformovat na plán *S'* přehozením nekonfliktních instrukcí.

Plán *S* je ***uspořádatelný vzhledem ke konfliktům***, existuje-li sériový plán, který je ekvivalentní s *S* vzhledem ke konfliktům.

Graf relace precedence transakcí je graf reprezentující binární relaci „T_i *předchází* T_j“ implikovanou konfliktními instrukcemi transakcí T_i a T_j. Plán je uspořádatelný vzhledem ke konfliktům právě když je odpovídající graf precedence acyklický.

9.4.2. Zajištění uspořádatelnosti

- *Techniky plánování (rozvrhování)*
  - *pesimistické*
  - *optimistické*
- *Mechanismy*

uzamykání, časová razítka, ...

9.4.3. Uzamykací protokoly

- *Podstata*
  - transakce před přístupem k objektu databáze požaduje přidělení zámku (uzamčení) tohoto objektu.
  - různé typy (režimy) uzamykání, typicky:
  - *sdílený zámek - lock_S(Q)*
  - *výlučný zámek - lock_X(Q)*

matice kompatibility

	S	X
S	A	N
X	N	N

Př) T_0, T_1 pouze zobrazí A+B

T_0	T1
lock_X(A)	
read (A, a_0)	
$a_0=a_0$-10000	
write(A,a_0)	
unlock(A)	
	lock_S(B)
	read(B,b_1)
	unlock(B)
	lock_S(A)
	read(A,a_1)
	unlock(A)
	display(a_1+b_1)
lock_x(B)	
read(B,b_0)	
$b_0=b_0$+10000	
write(B,b_0)	
unlock(B)	

T0	T1
lock_X(A)	
read (A, a_0)	
$a_0=a_0$-10000	
write(A,a_0)	
	lock_S(B)
	read(B,b_1)
	lock_S(A)
lock_X(B)	

Zablokování (deadlock)

Nekonzistentní analýza

Uzamykací protokol **je soustava pravidel stanovující, kdy může transakce uzamčít, resp. odemčít databázový objekt.**

- **existují protokoly zajišťující uspořádatelnost vzhledem ke konfliktům a případně i odstraňující nebezpečí zablokování**

- ***Dvoufázový uzamykací protokol (2PL)***
  1. ***Fáze růstu (growing)*** **- transakce uzamyká podle potřeby objekty, ale žádný neodemyká. Konec této fáze se nazývá *uzamykací bod* (lock point).**
  2. ***Fáze zmenšování (shrinking)*** **- transakce odemyká objekty, ale již nesmí žádný uzamčít.**
     - **zajišťuje uspořádatelnost vzhledem ke konfliktům, ale nevylučuje možnost zablokování**

  Modifikace:

  Striktní 2PL **- všechny výlučné zámky uvolňuje transakce až ve stavu potvrzení. Protokol zabraňuje kaskádnímu rušení transakcí (kaskádní rollback).**

  Rigorózní 2PL **- všechny zámky uvolňuje transakce až ve stavu potvrzení.**

Zjemnění: lock_S(Q), ... , upgrade(Q), ..., downgrade(Q), ... unlock(Q)

Jeddnoduché schéma uzamykání (často používané):

- ***Požaduje-li transakce operaci read(Q,q), systém nejprve provede uzamykací operaci lock_S(Q) a teprve pak read(Q,q).***
- ***Požaduje-li transakce operaci write(Q,q), systém provede uzamykací operaci upgrade(Q), resp.lock_X(Q) a teprve pak write(Q,q).***
- ***Všechny zámky držené transakcí jsou uvolněny teprve poté, co transakce potvrdí nebo je zrušena.***

- *Implementace uzamykání*
  Správce uzamykání (lock manager) používající tabulku zámků (hašovaná tabulka se seznamem uzamčených datových položek a čekajících transakcí + index identifikátorů transakcí).

- *Granularita uzamykání*

Granularita uzamykání udává, jak velká část databáze podléhá uzamykací operaci. Typické úrovně jsou řádek tabulky, blok, tabulka, databáze.

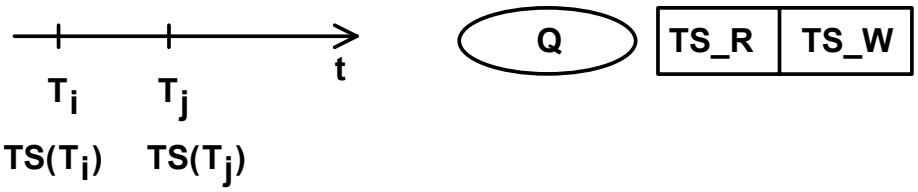
Př) Oracle: řádek, tabulka (LOCK TABLE), SQLBase: stránka, databáze

9.4.4. Protokoly založené na časových razítkách

Časovým razítkem rozumíme časový údaj vztažený k nějaké události.

- *Podstata*
  Transakcím a databázovým objektům jsou přiřazena časová razítka, která nesou informaci o čase určitých operací a potom se používají při zajištění uspořádatelnosti.

- Protokol s uspořádáním časových razítek (timestamp-ordering)



Platí-li pro T_i a T_j $TS(T_i) < TS(T_j)$, pak budou přípustné pouze plány ekvivalentní vzhledem ke konfliktům se sériovým plánem T_i, T_j.

Operace read(Q,q)

```
if TS(Ti) < TS_W(Q)then
   /* hodnota již přepsána pozdější transakcí */
   rollback(Ti)
else
begin
   read(Q,q);
   TS_R(Q) = max {TS_R(Q), TS(Ti)}
end
```


Operace write(Q,q)

```
if TS(Ti) < TS_R(Q) then
   /* hodnota již přečtena pozdější transakcí */
   rollback(Ti)
else
   if  TS(Ti) < TS_W(Q)then
      /* hodnota je zastaralá */
      rollback(Ti)
   else
   begin
      write(Q,q);
      TS_W(Q) = TS(Ti)
   end
```

- protokol zajišťuje uspořádatelnost a vyhýbá se zablokování

9.4.5. Další typy protokolů

- *Protokoly založené na validaci (validation-based)*
  - patří mezi optimistické techniky, vhodná pro prostředí, kde většina transakcí pouze čte

  Podstata: dvě nebo tři fáze transakce (čtení, validace, zápis). Ve fázi validace se ověřuje, zda došlo ke konfliktu s nějakou souběžnou transakcí (použití časových razítek pro začátek fází + informace o modifikovaných datech), uspořádání podle časového razítka pro validaci.

- Schémata s verzováním

  Podstata: Každá operace write(Q,q) vytváří novou verzi objektu Q. Schéma řízení přístupu musí zajistit, že při čtení transakce dostane správnou hodnotu (čtení je vždy úspěšné), zápis může vést na rolback transakce.

  Př) Oracle – kombinace s uzamykáním, použití tzv. rollback segmentů

9.4.6. Řešení problému zablokování

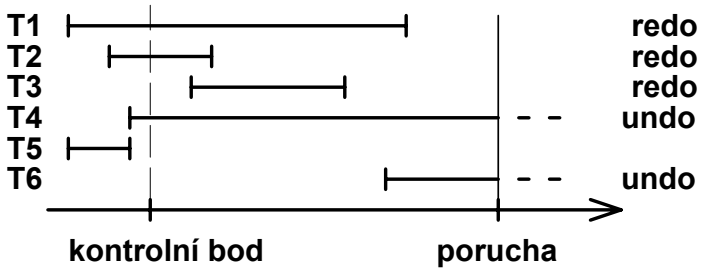
Zablokování (deadlock) je stav systému, kdy žádná z transakcí, které jsou v aktivním stavu, nemůže pokračovat v provádění, protože jí v tom brání některá jiná transakce.

- k zablokování může dojít, když transakce *čeká*, uvolnění systémových prostředků (typicky zámku) nějakou jinou transakcí, která je ale také nemůže uvolnit

- Varianty řešení
  - Použití protokolu zabraňujícího zablokování
  - *Maximální doba čekání (timeout)*
  - *Analýza grafu* binární *relace „čeká na“ (wait-for graph)*

9.4.7. Zotavení souběžných transakcí

- Zotavení při několika souběžných transakcích



- Kaskádní rollback

  Kaskádní rollback znamená zrušení transakce vyvolané zrušením jiné transakce.

  - možnost čtení nepotvrzené hodnoty může vést na kaskádní rollback.

9.5. Zotavení a souběžný přístup v SQL

- implicitně je požadováno zajištění uspořádatelnosti

- Nastavení vlastností příští transakce

```
SET TRANSACTION volby
```

režim přístupu
READ ONLY
READ WRITE

velikost
diagn. oblasti

izolační úroveň
ISOLATION LEVEL

- Izolační úroveň

  READ UNCOMMITTED
  READ COMMITTED
  REPEATABLE READ
  SERIALIZABLE

  SERIALIZABLE - zaručuje uspořádatelnost

SQL definuje tři *způsoby porušení uspořádatelnosti*:

Dirty read

T1	T2
read (Q,q)	
…	
write (Q,q)	
…	read (Q,q)
rollback	

Nonrepeatable read

T1	T2
read (Q,q)	
…	write (Q,q)
read (Q,q)	

Phantoms

T1	T2
read Q_1, Q_2, …Q_n	
…	insert Q_w
read Q_1, Q_2, …Q_n, Q_w	

Definice izolačních úrovní:

úroveň	Dirty read	Nonrep. read	Phantoms
READ UNCOMMITTED	A	A	A
READ COMMITTED	N	A	A
REPEATABLE READ	N	N	A
SERIALIZABLE	N	N	N

- při jiné úrovni než SERIALIZABLE by měl SŘBD poskytovat příkazy pro řízení souběžnosti
- standard SQL žádný explicitní mechanismus nezavádí

Př) Oracle: SERIALIZABLE, READ COMMITTED

SQLBase: Read Repeatability, Cursor Stability, Read Only, Release Lock

Literatura

1. Silberschatz, A., Korth H.F, Sudarshan, S.:Database System Concepts. Fourth Edition. McGRAW-HILL. 2001, str. 565 – 680.
2. Pokorný, J.: Databazová abeceda. Science, Veletiny, 1998, str. 57 – 60, 69 – 72, 187 – 190, 205 – 208, 217 – 220.