11. Transakční zpracování

Ing. Vladimír Bartík, Ph.D.
RNDr. Marek Rychlý, Ph.D.

Osnova

11.1. Transakce

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

11.1.1. Vlastnosti transakce

- **ACID vlastnosti**
  - **Atomičnost (Atomicity)**

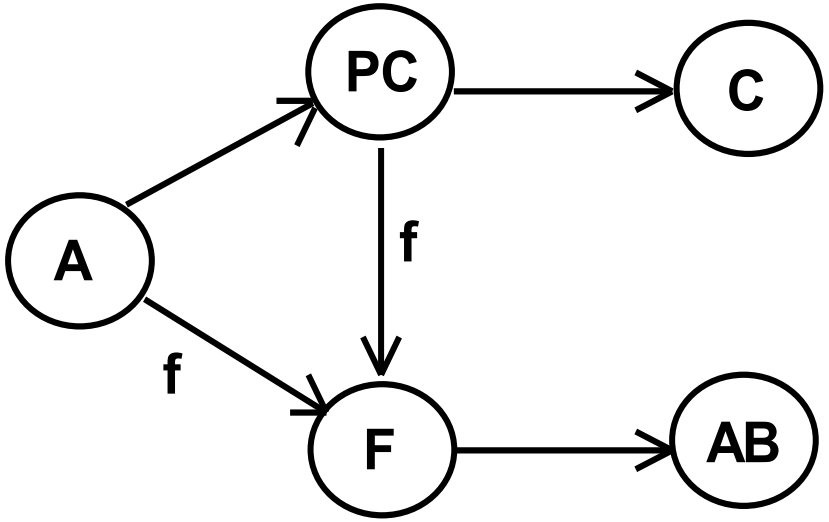
    Atomičnost transakce **znamená, že buď je provedena celá transakce nebo žádná z databázových operací, které ji tvoří.**
  - **Konzistence (Consistency)**

    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)**

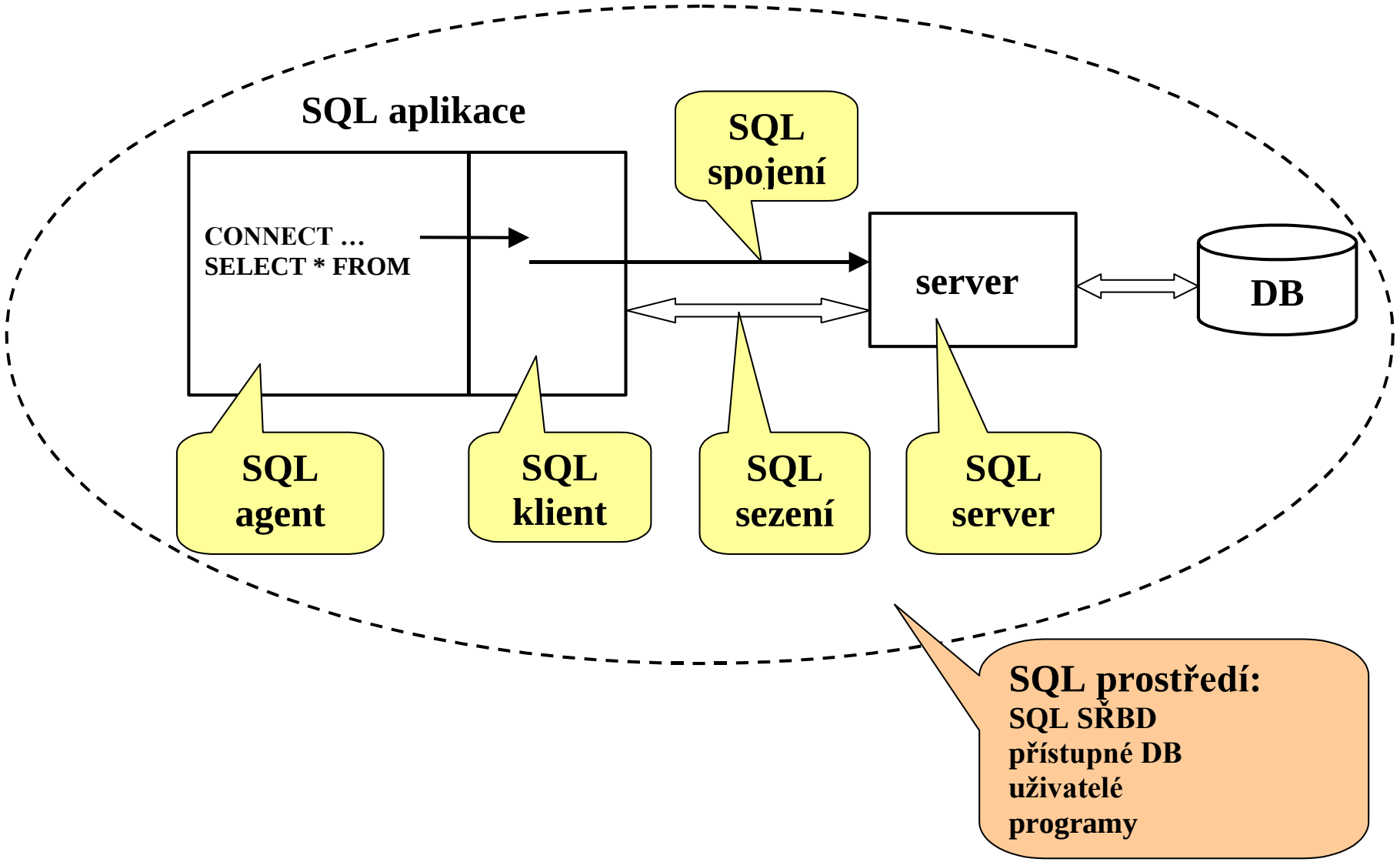
    Trvalost transakce **znamená, že poté, co transakce úspěšně skončí, budou mít všechny změny v databázi, které transakce provedla, trvalý charakter a to i při výpadku systému.**

11.1.2. 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.

11.2. Transakce v SQL



11.2. Transakce v SQL

- **Zahájení sezení**

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

  - ***string1*** **identifikuje SQL server, *string3* uživatele**
  - **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**

11.2. Transakce v SQL

- **SQL transakce**
  - **Příkazy 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.**
    - **standard SQL-99 zavádí příkaz START TRANSACTION pro explicitní zahájení transakce (v dialektech SQL někdy v podobě BEGIN /BEGIN TRANSACTION)**
  - **Ukončení transakce**

```
COMMIT
ROLLBACK
```

11.2. Transakce v SQL

- **Částečný rollback (není v SQL/92, až SQL:1999)**
  - **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**

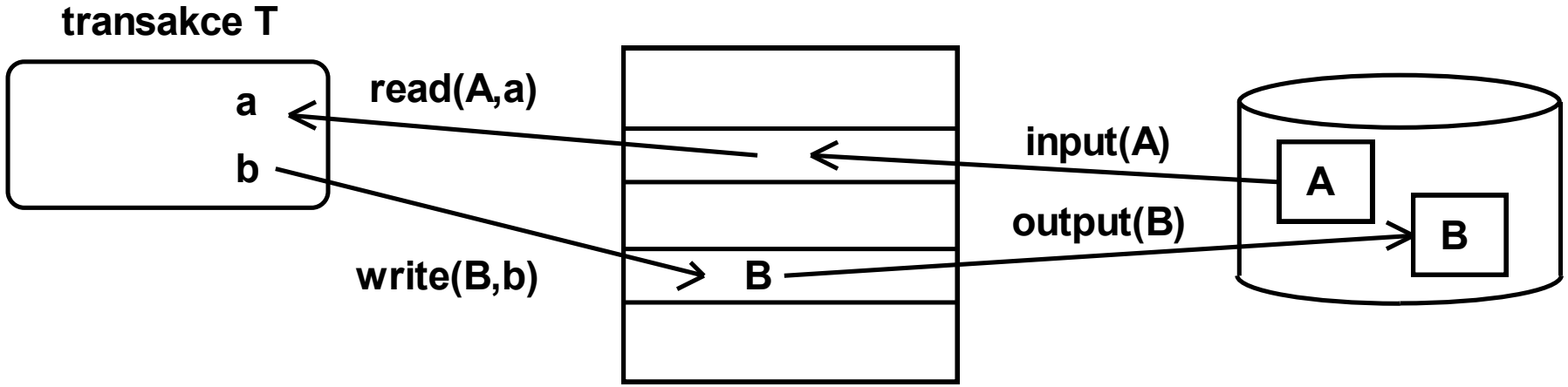
11.3. Zotavení po chybách a poruchách

- **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.

11.3. Zotavení po chybách a poruchách

- **Model přístupu transakce k datům**



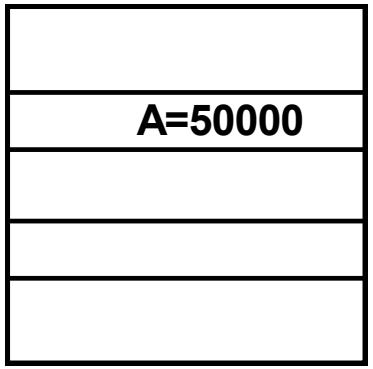
  - a, b ... lokální proměnné transakce
  - A,B ... datové položky z databáze
  - **input (B)** – načte blok s položkou B z disku do vyrovnávací paměti
  - **output (B)** – zapíše blok s položkou B z vyrovnávací paměti na disk
  - **read (A, a)** – přiřadí hodnotu položky A do lokální proměnné a
  - **write (A, a)** – přiřadí hodnotu a položce A ve vyrovnávací paměti

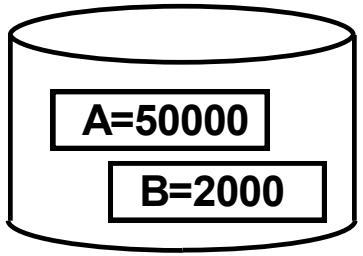
Poznámka: Databázi budeme v této části chápat jako tvořenou jednak bloky na disku, jednak bloky ve vyrovnávací paměti.

11.3. Zotavení po chybách a poruchách

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 pří operaci *read (B, b)* se uloží na disk z vyrovnávací paměti modifikovaný blok s *A*.

V konzistentním stavu platí, že součet stavů na účtech *A* a *B* 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 atomičnosti transakce a trvalosti změn je nutné před modifikací databáze uložit do stabilní paměti informace o modifikací.

11.3.1 Zotavení využívající žurnálu

- ***Žurnál (deník transakcí, 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.

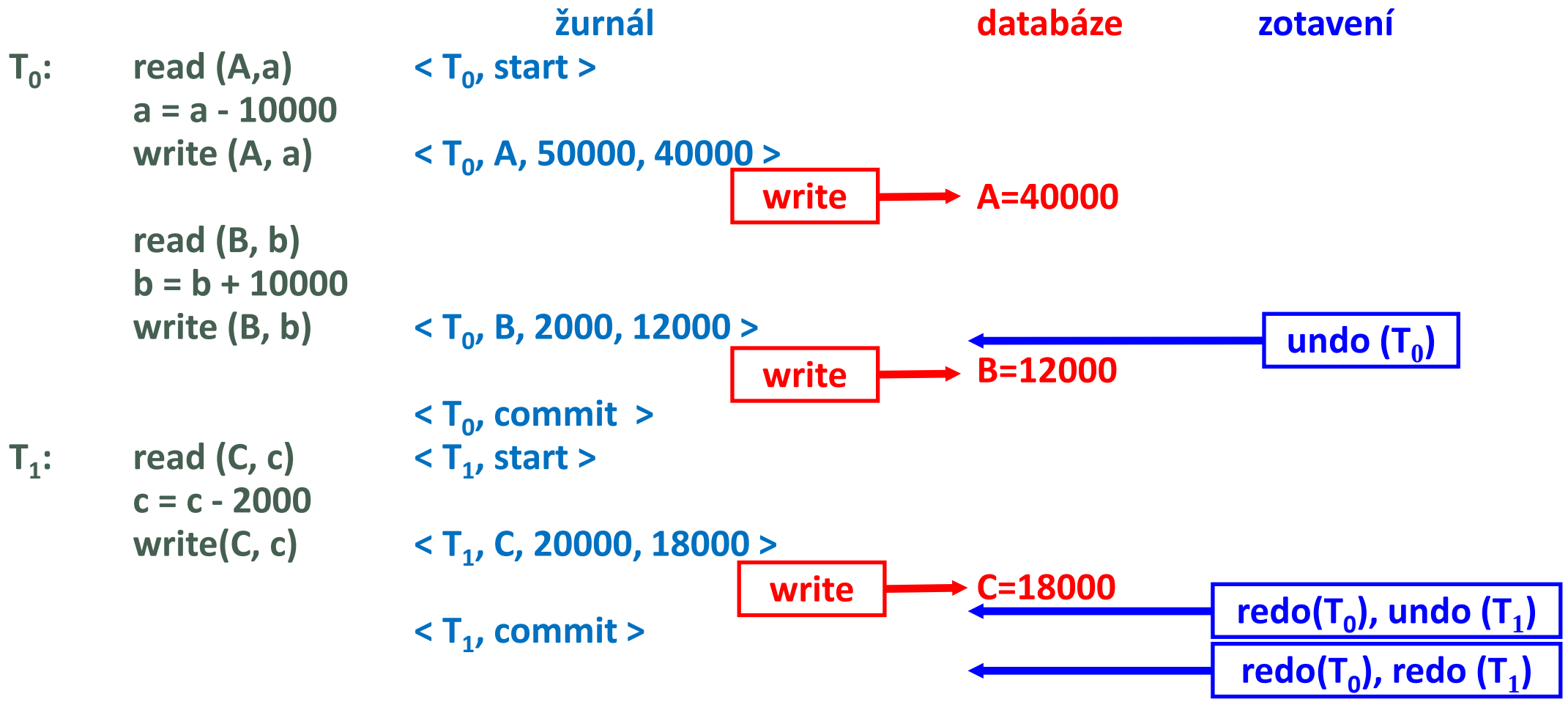
  < T_i , commit > - transakce T_i potvrdila změny (skončila úspěšně).

  < T_i , abort > - transakce T_i byla zrušena.

11.3.1 Zotavení využívající žurnálu

- **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).**

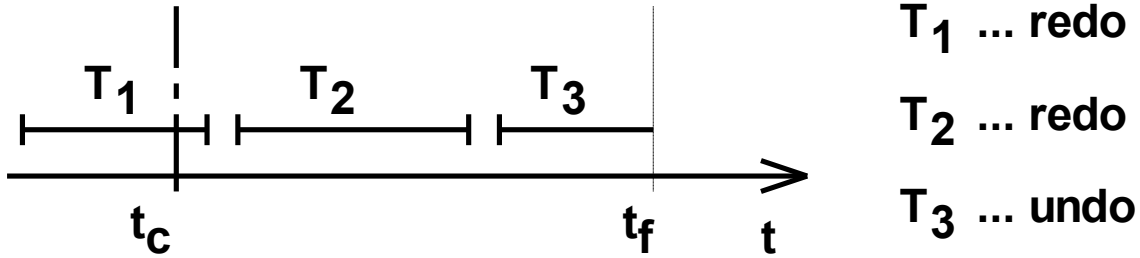
11.3.1 Zotavení využívající žurnálu



- Schéma zotavení: Na transakci Ti 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>

11.3.1 Zotavení využívající žurnálu

- ***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, a jejich roztřídění pro zotavení**
  2. **aplikace zotavovacích procedur *redo*(T_i) a *undo*(T_i) na každou transakci $T_i \in T$ podle výsledku roztřídění**

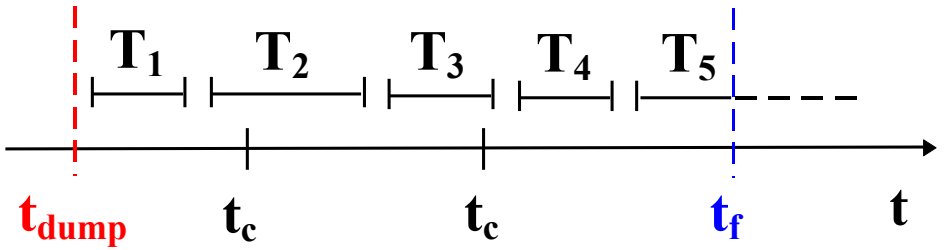


11.3.1 Zotavení využívající žurnálu

- **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 na disk musí být uloženy všechny záznamy žurnálu týkající se daného bloku do stabilní paměti (tzv. pravidlo WAL (write-ahead logging))**

11.3.2 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
  - uložení záznamů žurnálu do stabilní paměti,
  - uložení modifikovaných bloků DB z paměti na disk
  - uložení DB z disku do stabilní paměti
  - vytvoření záznamu < *dump* > v žurnálu ve stabilní paměti
- Zotavení
  - obnovení DB
  - zotavení od okamžiku archivace



11.4. Řízení souběžného přístupu

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

11.4.1 Sériové a uspořádatelné plány

- ***Plán (rozvrh)*** **udává chronologické pořadí provádění instrukcí souběžných transakcí.**
- ***Sériový plán*** **- instrukce jedné transakce bezprostředně za sebou.**
  - **pro *n* transakcí *n!* sériových plánů**
  - **sériový plán zachovává konzistenci**
  - **plány, které nejsou sériové, mohou porušit konzistenci**

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)

11.4.1 Sériové a uspořádatelné plány

	T_0	T_1
50000 A ←	read (A, a_0)	
	$a_0=a_0-10000$	
40000 A →	write(A,a_0)	
40000 A ←		read (A, a_1)
		pom=0.1*a_1
		$a_1=a_1-pom$
36000 A →		write(A,a_1)
2000 B ←	read(B,b_0)	
	$b_0=b_0+10000$	
12000 B →	write(B,b_0)	
12000 B ←		read(B, b_1)
		$b_1=b_1+pom$
16000 B →		write(B,b_1)

	T_0	T_1
50000 A ←	read (A, a_0)	
	$a_0=a_0-10000$	
50000 A ←		read (A, a_1)
		pom=0.1*a_1
45000 A →		$a_1=a_1-pom$
2000 B ←		write(A,a_1)
		read(B, b_1)
40000 A →	write(A,a_0)	
2000 B ←	read(B,b_0)	
	$b_0=b_0+10000$	
12000 B →	write(B,b_0)	
7000 B →		$b_1=b_1+pom$
		write(B,b_1)

- kritické jsou operace **read** a **write**

11.4.1 Sériové a uspořádatelné plány

- **Typické problémy, které je potřeba 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

11.4.1 Sériové a uspořádatelné plány

- **Typické problémy, které je potřeba řešit při řízení souběžného přístupu**
  - **Přepis nepotvrzené hodnoty**

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

  - **Nekonzistentní analýza**

Př) T1 zobrazí součet několika účtů, T2 mezitím provede převod.

11.4.1 Sériové a uspořádatelné plány

- **Uspořádatelné plány**
  - Binární relace **„je v konfliktu“** z množiny instrukcí transakce T_i do množiny instrukcí souběžné transakce T_j

T_i	T_j
:	:
I_{ix}	:
:	I_{jy}
:	:

I_x je_v_konfliktu_s I_y

I_x a I_y přistupují ke stejnému objektu

I_{ix} \ I_{jy}	read	write
read	N	A
write	A	A

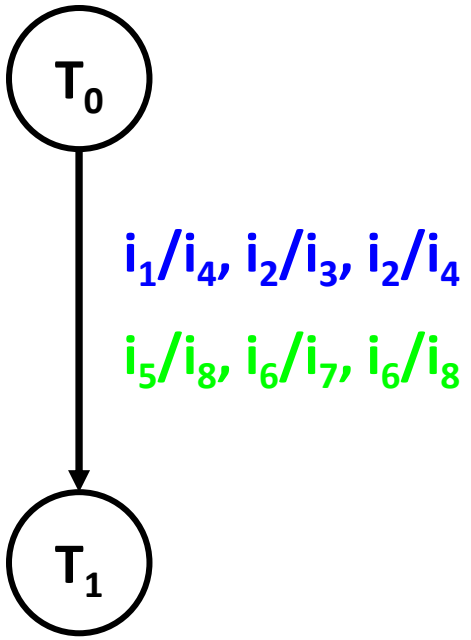
  - Instrukce I_{ix} a I_{jy} souběžných transakcí T_i a T_j **jsou v konfliktu**, právě když přistupují ke stejnému databázovému objektu (záznamu) a alespoň jednou z nich je write.
  - Plány S a S' se nazývají **ekvivalentní vzhledem ke konfliktům**, lze-li plán S transformovat na plán S' přehozením nekonfliktních instrukcí.
  - Plán S je **uspořádatelný vzhledem ke konfliktům**, existuje-li sériový plán, který je ekvivalentní s S vzhledem ke konfliktům.

11.4.1 Sériové a uspořádatelné plány

- **Uspořádatelné plány (pokračování)**
  - **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.**

Př.)

	T_0	T_1
1	read (A, a_0)	
2	write(A,a_0)	
3		read (A, a_1)
4		write(A,a_1)
5	read(B,b_0)	
6	write(B,b_0)	
7		read(B, b_1)
8		write(B,b_1)

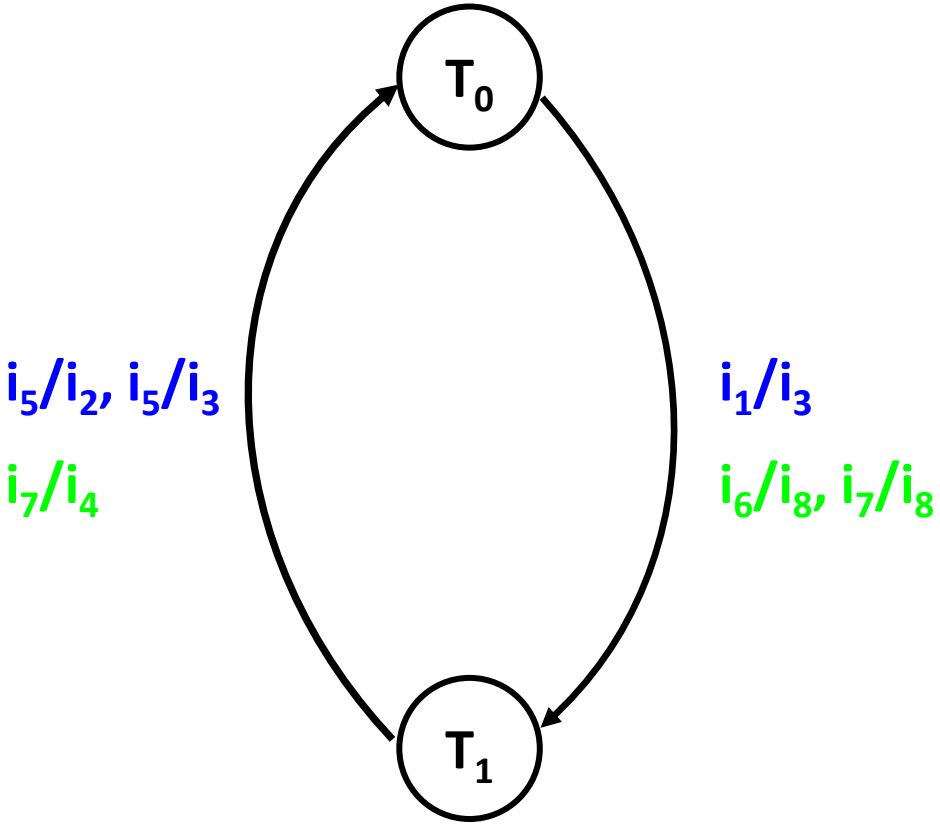


11.4.1 Sériové a uspořádatelné plány

- **Uspořádatelné plány (pokračování)**

  Př.)

	T0	T1
1	read (A, a_0)	
2		read (A, a_1)
3		write(A,a_1)
4		read(B, b_1)
5	write(A,a_0)	
6	read(B,b_0)	
7	write(B,b_0)	
8		write(B,b_1)



11.4.2 Zajištění uspořádatelnosti

- ***Techniky plánování (rozvrhování)***
  - ***pesimistické***
  - ***optimistické***
- ***Mechanismy***
  - **uzamykání**
  - **časová razítka**
  - **...**

11.4.3 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

11.4.3 Uzamykací protokoly

Př.) T_0 provede převod mezi účty A a B, T_1 pouze zobrazí A+B

T_0	T_1
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)	

Nekonzistentní analýza

T_0	T_1
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)

11.4.3 Uzamykací protokoly

- ***Uzamykací protokol*** **je soustava pravidel stanovující, kdy může transakce uzamknout, resp. odemknout databázový objekt (záznam).**
  - **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 uzamknout.**
  - **zajišťuje uspořádatelnost vzhledem ke konfliktům, ale nevylučuje možnost zablokování**

11.4.3 Uzamykací protokoly

- **Dvoufázový uzamykací protokol (pokračování)**
  - **Zjemnění: lock_S(Q), ... , upgrade(Q), ..., downgrade(Q), ... unlock(Q)**
  - **Jednoduché 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* udává, jak velká část databáze podléhá uzamykací operaci. Typické úrovně jsou řádek tabulky, blok, tabulka, databáze.**

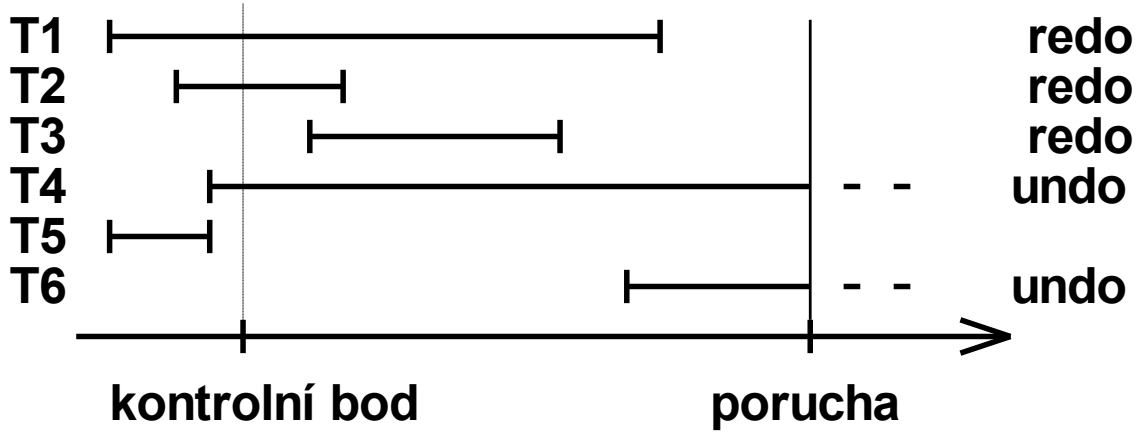
  Př) Oracle: řádek, tabulka (LOCK TABLE)

11.4.4 Řešení problému zablokování

- **Zablokování (deadlock)** je stav, kdy dvě nebo více transakcí, které jsou v aktivním stavu, nemohou pokračovat v provádění, protože si navzájem blokují požadované systémové prostředky (zde zámky).
  - k zablokování může dojít, když transakce čeká na 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)

11.4.5 Zotavení souběžných transakcí

- **Zotavení při několika souběžných transakcích**



11.5. Zotavení a souběžný přístup v SQL

- **Implicitně je požadováno zajištění uspořádatelnosti, neexistuje žádný příkaz pro uzamykání a odemykání**
- **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 – zaručuje uspořádatelnost

Literatura

1. Silberschatz, A., Korth H.F., Sudarshan, S.: Database System Concepts. Fifth Edition. McGRAW-HILL. 2006, str. 609-718.
2. Lemahieu, W., Broucke, S., Baesens, B.: Principles of Database Management. The Practical Guide to Storing, Managing and Analyzing Big and Small Data. Cambridge University Press 2018, str. 430-457.
3. Zendulka, J., Rudolfová, I.: Databázové systémy. IDS. Studijní opora. FIT VUT v Brně. 2006, str. 158-198.

SQL skripty

1. **K transakčnímu zpracování:**
   - ***transakce.sql* – ukázky vizualizace prováděcího plánu a pužití tipů optimalizátoru.**