Databázové systémy 2

Michal Krátký, Radim Bača

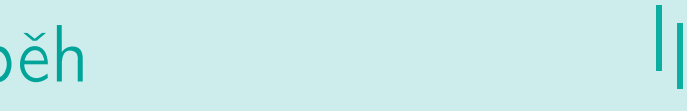
Katedra informatiky
Fakulta elektrotechniky a informatiky
VŠB – Technická univerzita Ostrava

2024/2025

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

Obsah přednášky

Paralelní vykonávání transakcí – souběh



- Pokud k DBS přistupuje více uživatelů současně, mluvíme o **paralelním vykonávání transakcí** neboli **souběhu** (angl. **concurency**).
- **Izolovanost v ACID** říká, že transakce se navzájem **neovlivňují**, tzn. transakce nesmí vidět **žádné aktualizace** ostatních transakcí, které nebyly potvrzeny před začátek této transakce.
  - Je možné to zajistit?
  - Pokud je to možné, jak drahé je to zajistit?
  - Nesníží taková izolovanost transakcí propustnost?

Plány vykonávání transakcí

- Posloupnost operací transakcí se nazývá **plán**, **plány vykonávání transakcí** budeme používat pro demonstraci problémů souběhu a jejich řešení.
- Pokud jsou plány provedeny paralelně, mluvíme o **plánu souběžném** nebo také o **plánu paralelním**.
- Pro zjednodušení budeme používat **operace**:
  - **READ** namísto SELECT,
  - **WRITE** namísto UPDATE, nicméně tato operace může zahrnovat i INSERT a DELETE.
- Pro zjednodušení budou tyto operace pracovat pouze **s jedním záznamem tabulky**.

Problémy souběhu

- Problém souběhu nastává, pokud dvě transakce A a B chtějí číst nebo zapisovat stejný záznam, existují **tři možnosti konfliktů**[1]: **RW** (**READ - WRITE**), **WR** a **WW**.
- Při těchto konfliktech mohou nastat, mimo jiné, tyto problémy souběhu:
  1. **ztráta aktualizace** (**lost update**),
  2. **nepotvrzená závislost** (**uncommitted dependency**),
  3. **nekonzistentní analýza** (**inconsistent analysis**).
  4. **neopakovatelné čtení** (**unrepeatable read**)
  5. **výskyt fantomů** (**phantom read**).

[1] Při RR žádný problém nevzniká.

1. Problém ztráty aktualizace

Transakce A	Čas	Transakce B
`READ` t	t_1	-
-	t_2	`READ` t
`WRITE` t	t_3	-
-	t_4	`WRITE` t

V čase t_4 dojde ke **ztrátě aktualizace** provedené transakcí A v čase t_3.

2. Problém nepotvrzené závislosti

Transakce A	Čas	Transakce B
-	t_1	`WRITE` t
`READ` t	t_2	-
-	t_3	`ROLLBACK`

- Transakce A se čtením záznamu v čase t_2 stala **závislou na nepotvrzeném zápisu** transakce B v čase t_1. A provedla **špinavé čtení** (**dirty read**).
- Transakce A pracuje po čase t_3 s **neplatnými hodnotami záznamu**, tedy s hodnotami získanými v čase t_2, ačkoli platné hodnoty jsou hodnoty z času před t_1.

3. Problém nekonzistentní analýzy 1/2

- Transakce A a B pracují s účty acc_1, acc_2 a acc_3. Před časem t_1 mají účty hodnotu 30, 20 a 50.
- Transakce A **počítá součet** konečných zůstatků na účtech, transakce B **převádí částku** 10 z účtu 1 na účet 3.

$acc_1 = 30$	$acc_2 = 20$	$acc_3 = 50$
Transakce A	**Čas**	**Transakce B**
READ acc_1	t_1	–
$suma = 30$		
READ acc_2	t_2	–
$suma = 50$		
–	t_3	READ acc_3
–	t_4	WRITE $acc_3 = 60$
–	t_5	READ acc_1
–	t_6	WRITE $acc_1 = 20$
–	t_7	COMMIT
READ acc_3	t_8	–
$suma = 110$ ne 100		

3. Problém nekonzistentní analýzy 2/2

$acc_1 = 30$	$acc_2 = 20$	$acc_3 = 50$
Transakce A	**Čas**	**Transakce B**
READ acc_1	t_1	-
$suma = 30$		
READ acc_2	t_2	-
$suma = 50$		
-	t_3	READ acc_3
-	t_4	WRITE $acc_3 = 60$
-	t_5	READ acc_1
-	t_6	WRITE $acc_1 = 20$
-	t_7	COMMIT
READ acc_3	t_8	-
$suma = 110$ ne 100		

Transakce A má k dispozici nekonzistentní databázi a proto vykoná **nekonzistentní analýzu** (spočítá nekorektní součet 110 namísto hodnoty 100).

4. Neopakovatelné čtení

Transakce A	Čas	Transakce B
READ t	t_1	
	t_2	WRITE t
READ t	t_3	

V čase t_1 a t_3 získá transakce A odlišnou hodnotu záznamu t, mluvíme o **neopakovatelné čtení**.

5. Výskyt fantomů

- Mějme následující tabulku `Student`:

login	jmeno	rocnik
jan001	Jan	1
mil002	Milan	3

- Mějme plány dvou paralelních transakcí[2]:

Transakce A	Čas	Transakce B
`SELECT * from student`	t_1	-
`  WHERE rocnik BETWEEN 1 AND 2`		
-	t_2	`INSERT INTO student`
		`VALUES('mar006',`
		`'Marek',2)`
	t_3	`COMMIT`
`SELECT * from student`	t_4	-
`  WHERE rocnik BETWEEN 1 AND 2`		
`COMMIT`	t_5	-

[2] Dostaneme různé výsledky v čase t_1 a t_4

Techniky řízení souběhu

- V minulosti byla vyvinuta celá řada **technik pro řízení souběhu**, které řeší výše zmíněné problémy souběhu:
  - **zamykání** (**locking**),
  - **správa verzí** (**multiversioning**),
  - časová razítka (timestamps),
  - validace.
- Jelikož současné DBS používají často **kombinaci zamykání a správy verzí**, budeme se zabývat pouze těmito dvěma technikami.

Techniky řízení souběhu

- **Zamykání**, **pesimistický přístup** k souběžnému zpracování: předpokládáme, že paralelní transakce se budou pravděpodobně navzájem ovlivňovat.
  - Při **zamykání** spravuje systém jednu kopii dat a jednotlivým transakcím přiděluje zámky.
- **Správa verzí**, **optimistický přístup**: předpokládáme, že paralelní transakce se ovlivňovat nebudou.
  - Při **správě verzí** systém vytváří kopie dat a sleduje, která z verzí má být viditelná pro ostatní transakce (v závislosti na úrovni izolace).

Zamykání

- **Zamykání:** Pokud transakce A chce provést čtení nebo zápis objektu v databázi (nejčastěji záznamu), pak požádá o zámek na tento objekt.
- **Typy zámků:**
  1. **Sdílený zámek (S)** je požadován **před čtením záznamu** (select). Další označení: zámek pro čtení, angl. **shared lock** nebo **read lock**.
  2. **Výlučný zámek (X)** je požadován **před aktualizací záznamu** (insert, update, delete[3]). Další označení: zámek pro zápis, angl. **exclusive lock** nebo **write lock**.
- **Uzamykání se provádí bez požadavku uživatele DBS**: pokud transakce chce získat nebo aktualizovat záznam, je **automaticky požadováno** přidělení zámku.

[3] S určitými odlišnostmi.

Požadavek na zamčení záznamu

- Při **požadavku na zamčení záznamu** mohou nastat tyto možnosti:
  1. Pokud transakce A **drží výlučný zámek (X)** na záznam t, pak požadavek paralelní transakce B na zámek libovolného typu na záznam t **není proveden**.
  2. Pokud transakce A **drží sdílený zámek (S)** na záznam t, pak:
     1. požadavek paralelní transakce B na zámek **X** na záznam t **není proveden**,
     2. požadavek paralelní transakce B na zámek **S** na záznam t **je proveden**. Více transakcí tedy může držet zámek **S** pro t.
- Pokud požadovaný zámek nemůže být přidělen, pak transakce přejde do **stavu čekání** (**wait state**) na uvolnění zámku. DBS řadí čekající transakce do fronty, aby transakce nezůstala v tomto stavu navždy (nazýváme **livelock** nebo **starvation**).

Přísné dvou-fázové zamykání

- **Přísné dvou-fázové zamykání** (nebo jen dvou-fázové zamykání, **strict two-phase locking**):
  - **Fáze 1:** požadování zámků záznamů.
  - **Fáze 2:** uvolňování zámků:
    - Pokud je nějaký zámek uvolněn, **není možné** požadovat další zámek.
    - Zámky jsou uvolněny **automaticky** na konci transakce (při COMMIT nebo ROLLBACK).

1. Řešení problému ztráty aktualizace 1/2

Transakce A	Čas	Transakce B
READ t	t_1	-
(získán zámek S na t)		
-	t_2	READ t
		(získán zámek S na t)
WRITE t	t_3	-
(požadavek na zámek X na t)		
wait	t_4	WRITE t
		(požadavek na zámek X na t)
wait		wait
wait		wait

- V čase t_3 **není transakci** A **přidělen** požadovaný zámek **X**, jelikož B drží pro záznam zámek **S**.
- Transakce A přejde do **stavu čekání na uvolnění** zámku **S** transakcí B.

1. Řešení problému ztráty aktualizace 2/2

Transakce A	Čas	Transakce B
READ t (získán zámek S na t)	t_1	-
-	t_2	READ t (získán zámek S na t)
WRITE t (požadavek na zámek X na t)	t_3	-
wait	t_4	WRITE t (požadavek na zámek X na t)
wait		wait
wait		wait

- Ze stejného důvodu přejde transakce B v čase t_4 do **stavu čekání** na uvolnění zámku **S** transakcí B.
- Obě transakce navzájem čekají na uvolnění zámku **S**, došlo k **uváznutí** (**deadlock**).

2. Řešení problému nepotvrzené závislosti

Transakce A	Čas	Transakce B
-	t_1	`WRITE` t (získán zámek X na t)
`READ` t (požadavek na zámek S na t)	t_2	-
wait	t_3	`COMMIT/ROLLBACK` (uvolnění zámku X na t)
opakuj: `READ` t (získán zámek S na t)	t_4	

- V čase t_2 není transakci A přidělen zámek **S**, A přejde do stavu čekání na uvolnění zámku **X** transakcí B.
- V čase t_3, při ukončení transakce B, je automaticky uvolněn zámek **X** pro t.
- V čase t_4 pokračuje A v činnosti, hodnota záznamu t je hodnota z času t_1 (při COMMIT transakce B) nebo před t_1 (při ROLLBACK). **Problém souběhu je vyřešen.**

3. Řešení problému nekonzistentní analýzy 1/2

$acc_1 = 30$	$acc_2 = 20$	$acc_3 = 50$
Transakce A	**Čas**	**Transakce B**
`READ` acc_1 (získán zámek S na acc_1) $suma = 30$	t_1	-
`READ` acc_2 (získán zámek S na acc_2) $suma = 50$	t_2	-
-	t_3	`READ` acc_3 (získán zámek S na acc_3)
-	t_4	`WRITE` $acc_3 = 60$ (získán zámek X na acc_3)
-	t_5	`READ` acc_1 (získán zámek S na acc_1)
...	...	...

3. Řešení problému nekonzistentní analýzy 2/2

$acc_1 = 30$		$acc_2 = 20$ $acc_3 = 50$
Transakce A	**Čas**	**Transakce B**
...	...	...
-	t_6	`WRITE` $acc_1 = 20$ (požadavek na zámek X na acc_1)
`READ` acc_3 (požadavek na zámek S na acc_3)	t_7	wait
wait		wait

- V čase t_6 není transakci B přidělen zámek **X** pro záznam acc_1, transakce B přejde **do stavu čekání na uvolnění** zámku **S** pro acc_1 transakcí A.
- Podobně, V čase t_7 není transakci A přidělen zámek **S** pro záznam acc_3, transakce A přejde **do stavu čekání na uvolnění** zámku **X** pro acc_3 transakcí B. **Došlo k uváznutí.**

4. Řešení problému neopakovatelné čtení

Transakce A	Čas	Transakce B
`READ` t	t_1	
(požadavek na zámek S pro t)		
	t_2	`WRITE` t
		(požadavek na zámek X pro t)
`READ` t	t_3	wait

- V čase t_2 není transakci B přidělen zámek **X** pro záznam t, transakce B přejde **do stavu čekání na uvolnění** zámku **S** pro t transakcí A.
- V čase t_3 transakce A přečte stejné hodnoty záznamu t jako v t_1, problém souběhu **neopakovatelné čtení** je vyřešen.

5. Řešení problému výskyt fantomů

Transakce A	Čas	Transakce B
`SELECT * from student`	t_1	-
`  WHERE rocnik BETWEEN 1 AND 2`		
-	t_2	`INSERT INTO student`
		`VALUES('mar006',`
		`'Marek',2)`
	t_3	`COMMIT`
`SELECT * from student`	t_4	-
`  WHERE rocnik BETWEEN 1 AND 2`		
`COMMIT`	t_5	-

- Pokud je v čase t_1 zamčen jen záznam pro ročník 1 (záznam pro ročník 2 v tabulce neexistuje), pak problém **vyřešení není**.
- Musím dojít k **zamykání rozsahu dotazu**, nikoli zamykání existujících záznamů v tabulce.

Uváznutí

- **Dvou-fázové uzamykání může způsobit uváznutí**. Obecné schéma vzniku uváznutí:

Transakce A	**Čas**	**Transakce B**
získán zámek S na r_1	t_1	
-	t_2	získán zámek S na r_2
požadavek na zámek X na r_2	t_3	-
wait	t_4	požadavek na zámek X na r_1
wait		wait

- **Pro řešení uváznutí se používají tyto strategie:**
  1. **Detekce uváznutí**:
     1. nastavení časových limitů,
     2. detekce cyklu v grafu **Wait-For**.
  2. **Prevence uváznutí** pomocí časových razítek.

Detekce uváznutí

- **Nastavení časových limitů:**
  - Jakmile transakce trvá déle než **stanovený časový limit**, předpokládá se, že došlo k uváznutí, a transakce je **zrušena** pomocí ROLLBACK.
- **Detekce cyklu v grafu Wait-For**:
  - V grafu *Wait-For* zaznamenává systém transakce, které na sebe vzájemně čekají.
  - Jedna z uváznutých transakcí je vybrána a zrušena (pomocí ROLLBACK).
  - Ostatní uváznuté transakce mohou pokračovat v činnosti.

Prevence uváznutí 1/2

- Strategie se snaží **uváznutí předcházet** modifikací uzamykacího protokolu.
- Dvě varianty: **Wait-Die** a **Wound-Wait**:
- **Algoritmus**:
  1. Každé transakci je přiděleno **časové razítko**, čas začátku transakce, které je jedinečné.
  2. Pokud transakce A požaduje zámek na záznam, která je již uzamčena transakcí B, pak:
     1. Při variantě **Wait-Die**: pokud A je starší než B, pak A přejde do stavu čekání; pokud A je mladší než B, transakce A je zrušena a spuštěna znovu.
     2. Při variantě **Wound-Wait**: pokud A je starší než B, transakce B je zrušena a spuštěna znovu; pokud A je mladší než B, pak A přejde do stavu čekání.
  3. Pokud je transakce spuštěna znovu, ponechává si své původní časové razítko.

Prevence uváznutí 2/2

- První část jména popisuje situaci, kdy transakce A **je starší než** B.
  - V případě **Wait-Die**, do stavu čekání přejde transakce starší,
  - v případě **Wound-Wait** do stavu čekání přejde transakce mladší.
- Je dokázáno, že v případě těchto protokolů, nemůže dojít k uváznutí, k nekonečnému čekání transakce (**livelock**), ani k nekonečnému znovuspouštění transakce.
- Nevýhodou obou protokolů je relativně **vysoký počet operací** ROLLBACK.