Integracja systemów transakcyjnych

Robert Wrembel
Politechnika Poznańska
Instytut Informatyki
Robert.Wrembel@cs.put.poznan.pl
www.cs.put.poznan.pl/rwrembel

Zarządzanie transakcjami

- Szeregowanie ⇨ zapewnienie spójności danych
  - metoda znaczników czasowych
  - metoda blokowania
- Wykrywanie zakleszczeń
- Protokół zatwierdzania (2PC/3PC)

Szeregowanie transakcji

- **Uszeregowanie**
  - **sekwencja operacji w ramach zbioru współbieżnych transakcji, która zachowuje oryginalny porządek operacji w każdej z transakcji**
  - **niech S będzie uszeregowaniem złożonym z operacji należących do zbioru transakcji $S=\{T_1, T_2, ..., T_n\}$ wówczas, dla każdej transakcji T_i w uszeregowaniu S, oryginalny porządek operacji transakcji T_i musi być zachowany w S**
- **Uszeregowanie sekwencyjne**
  - **wszystkie operacje należące do tej samej transakcji są wykonywane po sobie**

$$o_1^{T_1}, o_2^{T_1}, \ldots, o_n^{T_1}, o_1^{T_2}, o_2^{T_2}, \ldots, o_n^{T_2}, o_1^{T_3}, o_2^{T_3}, \ldots$$

Szeregowanie transakcji

- **Uszeregowanie niesekwencyjne**
  - **operacje należące do różnych transakcji się przeplatają**

$$o_1^{T_1}, o_1^{T_2}, o_1^{T_3}, \ldots, o_1^{T_n}, o_2^{T_1}, o_2^{T_2}, o_2^{T_3}, \ldots, o_2^{T_n}, o_3^{T_1}, o_3^{T_2}, \ldots$$

Szeregowanie transakcji

- Anomalie wynikające ze współbieżnego dostępu
  - dirty read
  - lost update
  - nonrepeatable reads / phantoms
  - inconsistent reads
- Uszeregowanie sekwencyjne rozwiązuje problem anomalii

Szeregowanie transakcji

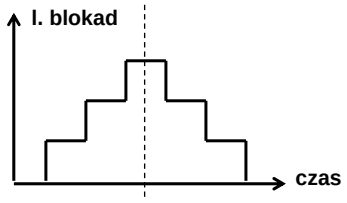
- Cel szeregowania transakcji
  - znalezienie uszeregowania niesekwencyjnego, którego wynik jest identyczny z wynikiem uszeregowania sekwencyjnego ⇨ transakcje mogą być wykonywane współbieżnie
  - powyższe definiuje kryterium poprawności uszeregowania
  - jeżeli kryterium poprawności jest spełnione ⇨ transakcje są uszeregowalne (serializable)

Szeregowanie transakcji

- Metody zapewnienia uszeregowalności
  - blokowanie
  - metody znaczników czasowych (timestamp ordering TSO)
  - metoda optymistyczna

Blokowanie

- Two-phase locking protocol (2PL)
  - faza zakładania blokad
  - faza zwalniania blokad



kompatybilność blokad

	shared	exclusive
shared	✓	x
exclusive	x	x

Wielowersyjność

- Dwie wersje danych
  - sprzed modyfikacji
  - modyfikowana

kompatybilność blokad

	S	X
S	✓	✓
X	✓	x

- Metody blokowania
  - konflikty eliminowane poprzez oczekiwanie na zwolnienie zasobów
  - prosta implementacja
  - zakleszczenia

Metody TSO

- Brak transakcji oczekujących
- Transakcje w konflikcie są wycofywane i restartowane
- Znacznik czasowy
  - znacznik czasowy transakcji: unikalny ID nadawany przez SZBD, odzwierciedla moment startu transakcji
  - znacznik czasowy danych
    - read_timestamp - TS transakcji, która jako ostatnia odczytała rekord
    - write_timestamp - TS transakcji, która jako ostatnia zmodyfikowała rekord

Metody TSO

- Szeregują transakcje w taki sposób, że transakcje starsze (mniejsze ID) mają wyższy priorytet niż młodsze
- Transakcja T może przeczytać lub zapisać rekord R jeżeli ostatnia modyfikacja R została wykonana przez transakcję starszą
  - w przeciwnym przypadku T jest restartowana z nowym znacznikiem czasowym

Metody TSO

- Transakcja T wykonuje READ(x)
  - rekord x został zmodyfikowany przez młodszą transakcję, tj.
    TS(T) < write_timestamp(x)
    - rollback T
    - nadaj T nowy TS
    - restart T
  - TS(T) >= write_timestamp(x)
    - READ(x)
    - przypisz read_timestamp(x):=
      max{TS(T), read_timestamp(x)}

Metody TSO

- Transakcja T wykonuje WRITE(x)
  - x został wcześniej odczytany przez młodszą transakcję, tj. TS(T) < read_timestamp(x)
    - rollback T
    - nadaj T nowy TS
    - restart T
  - x został wcześniej zmodyfikowany przez młodszą transakcję, tj. TS(T) < write_timestamp(x)
    - rollback T
    - nadaj T nowy TS
    - restart T
  - otherwise: execute T
    - przypisz write_timestamp(x) = TS(T)

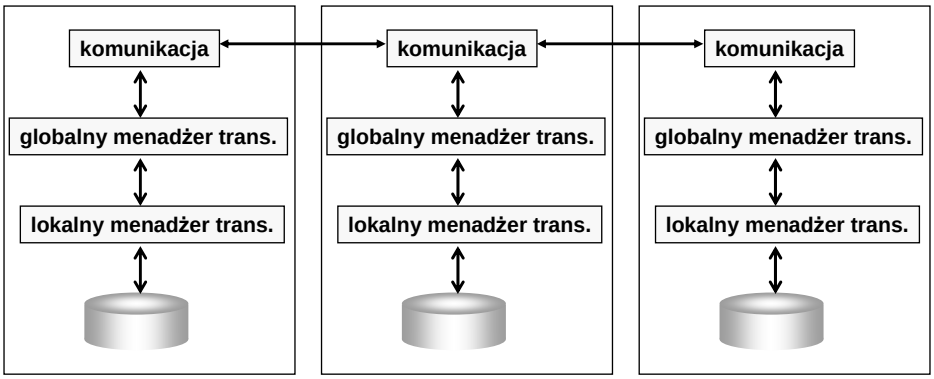
Metoda optymistyczna

- Założenie: konflikty są rzadkie
  - przed zatwierdzeniem transakcji T system sprawdza, czy wystąpił konflikt, jeśli tak, to T jest wycofywana
- Większa współbieżność - brak oczekiwania na zwolnienie zasobów
- Duże koszty restartu transakcji
- Fazy
  - read: odczyt i modyfikacja danych
    - modyfikacja na lokalnych kopiach danych
    - transakcja T otrzymuje znacznik czasowy START(T)
  - validation:
    - sprawdzenie, czy spełnione jest kryterium uszeregowalności
    - transakcja T otrzymuje znacznik czasowy VALIDATION(T)
  - write: zapis zmodyfikowanych danych na dysk
    - transakcja T otrzymuje znacznik FINISH(T)

Metoda optymistyczna

- **Przejście fazy walidacji możliwe gdy:**
  1. **wszystkie transakcje T_i z wcześniejszymi TS musiały się zakończyć przed rozpoczęciem transakcji T, tj. FINISH(T_i) < START(T)**
  2. **jeżeli T rozpoczyna się zanim wcześniejsza transakcja T_e się kończy, wówczas**
     a) zbiór rekordów zmodyfikowanych przez T_e jest rozłączny ze zbiorem rekordów odczytywanych przez T i
     b) T_e kończy zapis zanim T wchodzi w fazę walidacji, tj. START(T)<FINISH(T_e)<VALIDATION(T)
        - gwarantuje to sekwencyjny zapis na dysk (modyfikacje) i brak konfliktów

Zarządzanie transakcjami rozproszonymi



Rozproszone zarządzanie współbieżnością

- **Rozproszona uszeregowalność**
  - **jeżeli porządek transakcji w każdym z węzłów jest uszeregowalny, wówczas globalny porządek jest również uszeregowalny jeżeli jest spełnione:**

$$T_m^{S_1} \prec T_n^{S_1}, T_m^{S_2} \prec T_n^{S_2}, \ldots, T_m^{S_i} \prec T_n^{S_i}$$

    - m, n – numer kolejnej transakcji
    - S_i – węzeł
- **Metody**
  - **blokowania**
    - **centralized 2PL**
    - **primary copy 2PL**
    - **distributed 2PL**
    - **majority locking**
  - **znaczników czasowych**

Centralized 2PL (C2PL)

- **Jeden węzeł zarządza wszystkimi blokadami**
  - **zawiera pełne dane nt. zakładanych blokad**
- **Tylko jeden zarządca blokad w systemie RBD może zakładać i zwalniać blokady**
- **Przebieg 2PL dla transakcji globalnej zainicjowanej z węzła S1**
  1. **koordynator trans. w S1 dzieli T na podtransakcje**
     - **jeśli T modyfikuje replikę, wówczas koordynator musi zagwarantować uaktualnienie wszystkich kopii - blokowanie kopii w trybie wyłącznym**
     - **wybór repliki do odczytu (jest nią replika lokalna, jeśli istnieje)**
  2. **lokalny koordynator trans. zakłada i zwalnia blokady zgodnie z poleceniami centralnego zarządcy blokad (standardowy 2PL wykorzystywany lokalnie)**
  3. **Centralny zarządca blokad kontroluje kompatybilność blokad**
     - **blokady niekompatybilne trafiają do kolejki oczekiwań**

Centralized 2PL

- Cechy
  - prosta implementacja
  - łatwość wykrycia zakleszczenia ⇨ informacja o wszystkich blokadach znajduje się w jednym węźle
  - niskie koszty komunikacji
  - centralny zarządca blokad
    - wąskie gardło
    - większa zawodność

Primary Copy 2PL (PC2PL)

- W przypadku istnienia replik jedna jest wybierana jako główna
- Zarządcy blokad istnieją w wielu węzłach (nie wszystkich)
- Modyfikacja wykonywana najpierw na głównej replice
  - koordynator trans. musi określić miejsce składowania głównej repliki i wysłać żądanie jej zablokowania do lokalnego zarządcy blokad
  - tylko główna replika jest blokowana na wyłączność
- PC2PL gwarantuje, że tylko główna replika jest aktualna
  - synchronizacja głównej repliki z pozostałymi może być realizowana jako odrębna transakcja
- PC2PL stosowane gdy
  - modyfikacje występują rzadko
  - nie jest wymagana aktualna wersja danych
- Trudniejsze wykrywanie zakleszczeń globalnych

Distributed 2PL (D2PL)

- Zarządca blokad jest w każdym węźle
- Lokalny zarządca blokad jest odpowiedzialny za swój węzeł
- Dla niereplikowanych danych D2PL jest równoważny 2PL
- Dla replikowanych danych
  - każda replika może być wykorzystana do odczytu
  - wszystkie repliki muszą zostać zablokowane przed modyfikacją
- Cechy
  - trudne wykrywanie i rozwiązywanie zakleszczeń
  - brak wad C2PL

Majority Locking (ML)

- Zarządca blokad jest w każdym węźle
- Lokalny zarządca blokad jest odpowiedzialny za swój węzeł
- Dla replikowanych danych
  - przed modyfikacją transakcja musi założyć blokady na więcej niż połowie replik
  - synchronizacja pozostałych replik wykonywana później (w odrębnej transakcji)
- Cechy
  - mniejsza liczba blokowanych węzłów
  - trudne wykrywanie i usuwanie globalnych zakleszczeń

Znaczniki czasowe

- **Cel**
  - **porządkowanie transakcji globalnie tak, aby starsze transakcje (z mniejszymi TS) miały wyższy priorytet w przypadku konfliktów**
- **Generowanie TS**
  - **zegar systemowy ⇨ brak synchronizacji zegarów pomiędzy węzłami**
  - **zegar logiczny ⇨ duplikaty TS w różnych węzłach**
- **Możliwe rozwiązanie**
  - **localTS + siteID**
  - **synchronizacja TS między węzłami**

Znaczniki czasowe

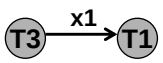
- **Synchronizacja TS między węzłami**
  - **każdy węzeł wysyła aktualną wartość TS w komunikatach do innych węzłów**
  - **węzeł odbierając komunikat porównuje swój TS (localTS) z otrzymanym (remoteTS)**
    - **jeśli localTS<remoteTS, wówczas localTS:=remoteTS + increment_by**
    - **w przeciwnym przypadku nie podejmuj żadnej akcji**

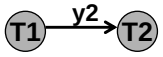
Rozproszone zakleszczenia

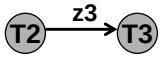
- Dane są 3 transakcje
  - T1 zainicjowana w S1 żąda operacji w S2
  - T2 zainicjowana w S2 żąda operacji w S3
  - T3 zainicjowana w S3 żąda operacji w S1
- Blokady

Time	S1	S2	S3
t1	read_lock(T1, x1)	write_lock(T2, y2)	read_lock(T3, z3)
t2	write_lock(T1, y1)	write_lock(T2, z2)	
t3	write_lock(T3, x1)	write_lock(T1, y2)	write_lock(T2, z3)

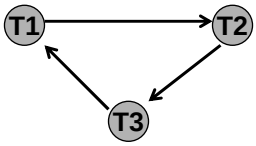
lokalny WFG

S1 (x1, y1) 

S2 (y2, z2) 

S3 (z3) 

globalny WFG



Wykrywanie zakleszczeń

- W SRBD zbudowanie lokalnego WFG nie wystarcza do wykrycia zakleszczenia
- Wymagane jest skonstruowanie globalnego WFG jako sumy lokalnych WFG
- Metody
  - scentralizowana (centralized)
  - hierarchiczna (hierarchical)
  - rozproszona (distributed)

Przykład 1

➲ **Dane jest 5 transakcji T1, T2, T3, T4, T5**

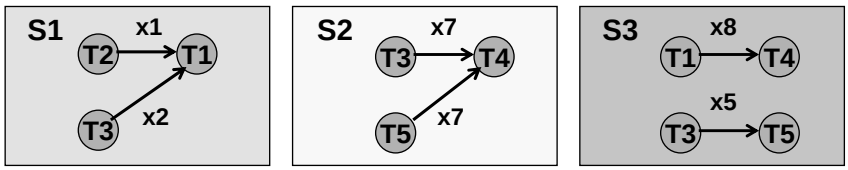
- **T1 zainicjowana w S1 żąda operacji w S3**
- **T2 zainicjowana w S3 żąda operacji w S1**
- **T3 zainicjowana w S1 żąda operacji w S2 i S3**
- **T4 zainicjowana w S2 żąda operacji w S3**
- **T5 zainicjowana w S3 żąda operacji w S2**

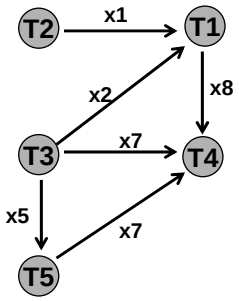
	S1	S2	S3
1.	read_lock(**T1**, x1)	read_lock(T4, x7)	read_lock(T2, x4)
2.	read_lock(**T1**, x2)		write_lock(T4, x8)
3.	write_lock(T2, x1)		
4.	write_lock(**T3**, x2)		write_lock(**T1**, x8)
5.		write_lock(**T3**, x7)	
6.			write_lock(**T5**, x5)
7.		write_lock(**T5**, x7)	
8.			read_lock(**T3**, x5)

Przykład 1

A. Dla każdego węzła utworzyć lokalne WFG
B. Utworzyć globalny WFG
C. Sprawdzić, czy wystąpiło zakleszczenie

Rozwiązanie





	S1	S2	S3
1.	read_lock(**T1**, x1)	read_lock(T4, x7)	read_lock(T2, x4)
2.	read_lock(**T1**, x2)		write_lock(T4, x8)
3.	write_lock(T2, x1)		
4.	write_lock(**T3**, x2)		write_lock(**T1**, x8)
5.		write_lock(**T3**, x7)	
6.			write_lock(**T5**, x5)
7.		write_lock(**T5**, x7)	
8.			read_lock(**T3**, x5)

Metoda scentralizowana

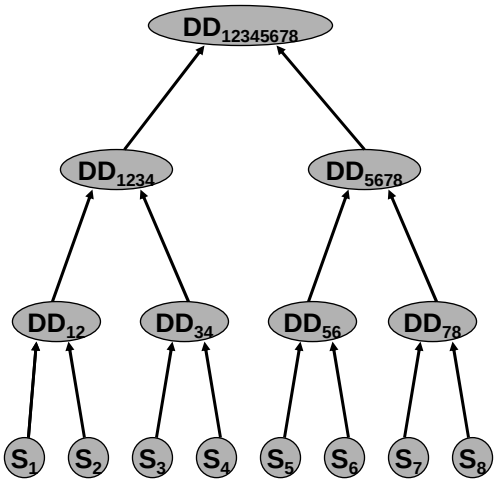
- **Jeden z węzłów zostaje wybrany jako tzw. Deadlock Detection Coordinator (DDC)**
- **DDC konstruuje i utrzymuje globalny WFG**
- **Każdy z lokalnych zarządców blokad okresowo wysyła swój WFG to DDC**
  - **optymalizacja ⇨ wysłanie tylko zmian w WFG**
- **DDC sprawdza istnienie cykli w globalnym WFG**
  - **wykrycie cyklu wymaga przerwania wybranej transakcji, wycofanie jej i restart**
  - **DDC informuje węzeł z wycofaną transakcją o konieczności jej restartu**
- **Awaria DDC uniemożliwia wykrycie zakleszczenia**

Metoda hierarchiczna

- Węzły tworzą hierarchię
- Każdy węzeł może wykryć zakleszczenie
  - węzeł - Deadlock Detector (DD)
- Każdy węzeł niżej w hierarchii wysyła WFG do węzła nadrzędnego
- Cechy
  - większa niezawodność
  - trudniejsza implementacja

Metoda hierarchiczna

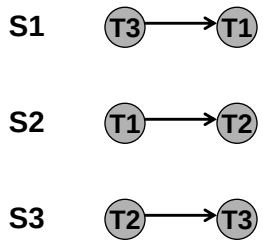
- Poziom 1: wykrywanie lokalnych zakleszczeń
- Poziom 2: węzły wykrywają zakleszczenie w 2 sąsiednich węzłach (1-2 i 3-4; 5-6 i 7-8)
- Poziom 3: węzły wykrywają zakleszczenie w 4 sąsiednich węzłach (1-2-3-4 i 5-6-7-8)
- Poziom 4: korzeń jest globalnym DD



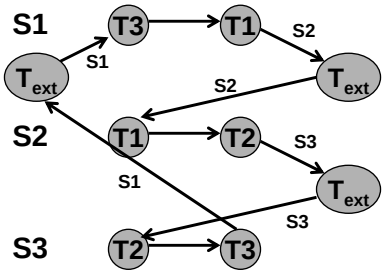
Metoda rozproszona

- Żądanie wychodzące do węzła zewnętrznego jest reprezentowane w lokalnym WFG za pomocą węzła T_{ext}
- Jeśli transakcja T1 w węźle S1 wysyła żądanie do S2, wówczas do lokalnego WFG jest dodawana krawędź T1 → T_{ext}
- W węźle S2 do lokalnego WFG jest dodawana krawędź T_{ext} → T1

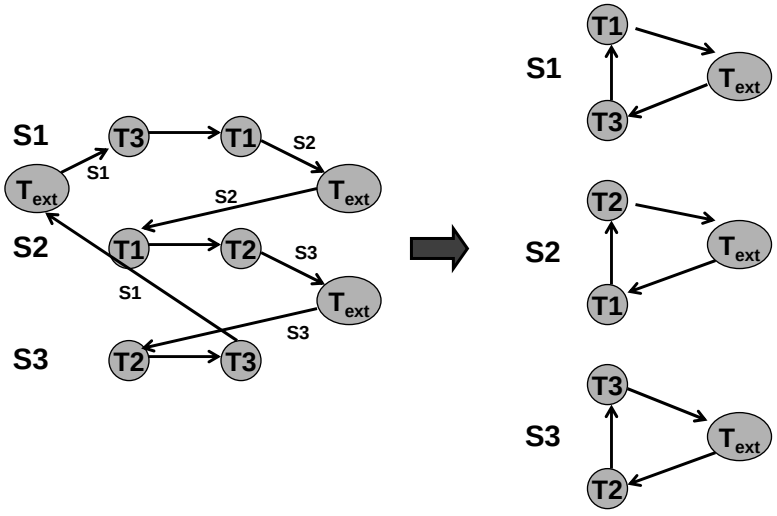
lokalny WFG



lokalny WFG dla metody rozproszonej



Metoda rozproszona



Metoda rozproszona

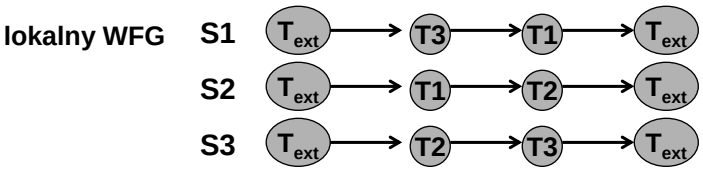
- Jeżeli lokalny WFG zawiera cykl bez węzła T_{ext}, wówczas w węźle występuje globalne zakleszczenie
- Globalne zakleszczenie może wystąpić jeśli lokalny WFG zawiera cykl z T_{ext}
  - T_{ext} może reprezentować różne transakcje w węzłach zdalnych ⇨ istnienie cyklu z T_{ext} nie musi oznaczać zakleszczenia
  - w celu potwierdzenia globalnego zakleszczenia lokalne WFG są scalane w globalny WFG

Metoda rozproszona

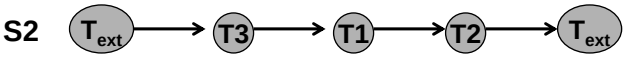
- Scalanie lokalnych WFG
  - węzeł S1 przesyła lokalny WFG tylko do tego węzła(ów) Si w którym transakcja T1 (w S1) oczekuje na zwolnienie blokady
  - Si odbiera WFG z S1 i scala go ze swoim WFG
  - po scaleniu WFG, Si sprawdza istnienie cyklu bez T_{ext}
  - jeżeli cykl nie wystąpił, wówczas scalony WFG jest przesyłany do kolejnego węzła

Metoda rozproszona

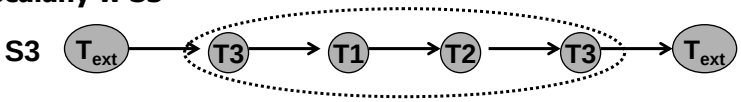
lokalny WFG



- T1 oczekuje na T2 ⇨ WFG jest przesyłany z S1 do S2 i scalany w S2



- T2 oczekuje na T3 ⇨ WFG jest przesyłany z S2 do S3 i scalany w S3



cykl bez T_{ext} ⇨ globalne zakleszczenie

Przykład 2

- Informacje o blokadach

Transaction	Data locked by transaction	Data transaction is waiting for	Site involved in operation
T1	x1	x8	S1
T1	x6	x2	S2
T2	x4	x1	S1
T2	x5		S3
T3	x2	x7	S1
T3		x3	S3
T4	x7		S2
T4	x8		S3
T5	x3	x7	S3

	S1	S2	S3
1.	write_lock(T3, x2)	read_lock(T4, x7)	
2.	read_lock(T1, x1)		write_lock(T4, x8)
3.	read_lock(T1, x2)		
4.	write_lock(T2, x1)	write_lock(T3, x7)	write_lock(T1, x8)
5.		write_lock(T5, x7)	
6.			write_lock(T5, x5)
7.			read_lock(T3, x5)

- Sprawdzić, czy występuje globalne zakleszczenie wykorzystując alg. rozproszonego wykrywania zakl.

Rozwiązanie