Izolacje transakcji oraz anomalie

Robert A. Kłopotek

r.klopotek@uksw.edu.pl

Wydział Matematyczno-Przyrodniczy. Szkoła Nauk Ścisłych, UKSW

SZBD (DBMS) a transakcji

- Przetwarzanie transakcyjne wymaga znaczącego narzutu oraz wyszukanych rozwiązań od DBMS
- Większość komercyjnych SZBD, takich jak Oracle, MS SQL Server, IBM DB2 itp. są systemami transakcyjnymi
- Wiele bezpłatnych rozwiązań nie wspiera w całości transakcji, przez co oferują ograniczoną niezawodność dla krytycznych zastosowań włączając księgowość i systemy bankowe

Transakcje - problemy

Stan konta nr. 456		
1000 PLN	**BEGIN TRANSACTION** UPDATE Accounts set balance=balance+500 where account_id=456	
1500 PLN	. . .	**BEGIN TRANSACTION** UPDATE Accounts set balance=balance+500 where account_id=456
1400 PLN	**ROLLBACK**	UPDATE Accounts set balance=balance+100 where account_id=8899 **COMMIT**

Pierwsza transakcja wpłynęła na druga stwarzając nieprawidłowy stan konta (1400 PLN zamiast 900PLN), który został zapisany w bazie danych

Anomalie transakcji

- W praktyce wiele transakcji, zgłoszonych przez różne aplikacje z różnych stanowisk pracy, jest wykonywane równolegle przez SZBD (DBMS)
- Izolacje transakcji można łatwo zapewnić jeśli transakcje nie są wykonywane równolegle, czyli sekwencyjne. Jednak takie rozwiązanie w znaczącym stopniu by zredukowało wydajność SZBD (DBMS)

Anomalie transakcji

Kategoria	Opis
Brudny odczyt (ang. Dirty read / Uncommitted dependency	Występuje, gdy transakcja odczytuje dane, które nie zostały jeszcze zatwierdzone (COMMIT). Jeśli inne transakcja ostatecznie wycofa zmiany (ROLLBACK) wtedy pierwsza transakcja może odczytać zmiany, które nigdy nie istniały.
Utracona modyfikacja (ang. Lost update)	Występuje, gdy więcej niż jedna transakcja czyta te same dane. Modyfikacje , które zostaną zrobione później przez pierwszą transakcję zostaną nadpisane prze inną transakcję nieświadomą zmian, które nastąpiły.
Niepowtarzalny odczyt (ang. Non-repeatable read / Inconsistent Analysis)	Występuje, gdy transakcja odczytuje dwa razy ten sam wiersz, ale dostaje inne dane za każdym razem. Przykład: transakcja 1 czyta wiersz. Transakcja 2 modyfikuje lub usuwa ten wiersz oraz zostaje zakończona (COMMIT). Jeśli transakcja 1 odczyta teraz ten wiersz to dostanie inne dane lub odkryje, że wiersz już nie istnieje.
Fantomy (ang. Phantoms)	Fantom to wiersz, który spełnia kryteria wyszukiwania, ale nie jest widoczny na początku. Przykład: transakcja 1 czyta zbiór wierszy, który spełnia kryteria wyszukiwania. Transakcja 2 dodaje wiersz, który spełnia kryteria wyszukiwania transakcji 1. Jeśli transakcja 1 ponownie wykona to samo wyszukiwanie dostanie inny zbiór rekordów.

Poziomy izolacji – SQL-92

Pozim Izolacji\Anomalia	Brudny odczyt	Niepowtarzalny odczyt	Fantomy
READ UNCOMMITTED	możliwy	możliwy	możliwy
READ COMMITTED	brak	możliwy	możliwy
REPEATABLE READ	brak	brak	możliwy
SERIALIZABLE	brak	brak	brak

Standard SQL-92 definiuje cztery poziomu izolacji transakcji (kolejność od najniższego stopnia izolacji do najwyższego):

- **READ UNCOMMITTED** - niezatwierdzony odczyt (poziom izolacji 0),
- **READ COMMITTED** - odczyt zatwierdzonych danych (poziom izolacji 1),
- **REPEATABLE READ** - powtarzalny odczyt (poziom izolacji 2),
- **SERIALIZABLE** - uszeregowany (poziom izolacji 3).

Anomalie - rozwiązania

- Współczesne DBMS pozwalają na określenie poziomu izolacji
- Różne poziomy izolacji wpływają na pojawienie się lub nie różnych anomalii
- Poziomy izolacji pozwalają na zachowanie balansu pomiędzy wydajnością a wymaganą izolacją
- Przykład:
  - Ms SQL Server 2008+ dostarcza 5 poziomów izolacji transakcji, instrukcja SET TRANSACTION ISOLATION LEVEL LevelName (Read uncommitted, Read committed, Repeatable read, Snapshot, Serializable)
  - Baza danych H2 ma 3 poziomy, instrukcja: SET LOCK_MODE (0 - Read Uncommitted, 1 - Serializable, 3 - Read Committed)
  - MySQL 5.7 ma 4 poziomy izolacji: Read uncommitted, Read committed, Repeatable read, Serializable
  - Oracle ma 3 poziomy izolacji: Read Committed, Serializable, Read Only

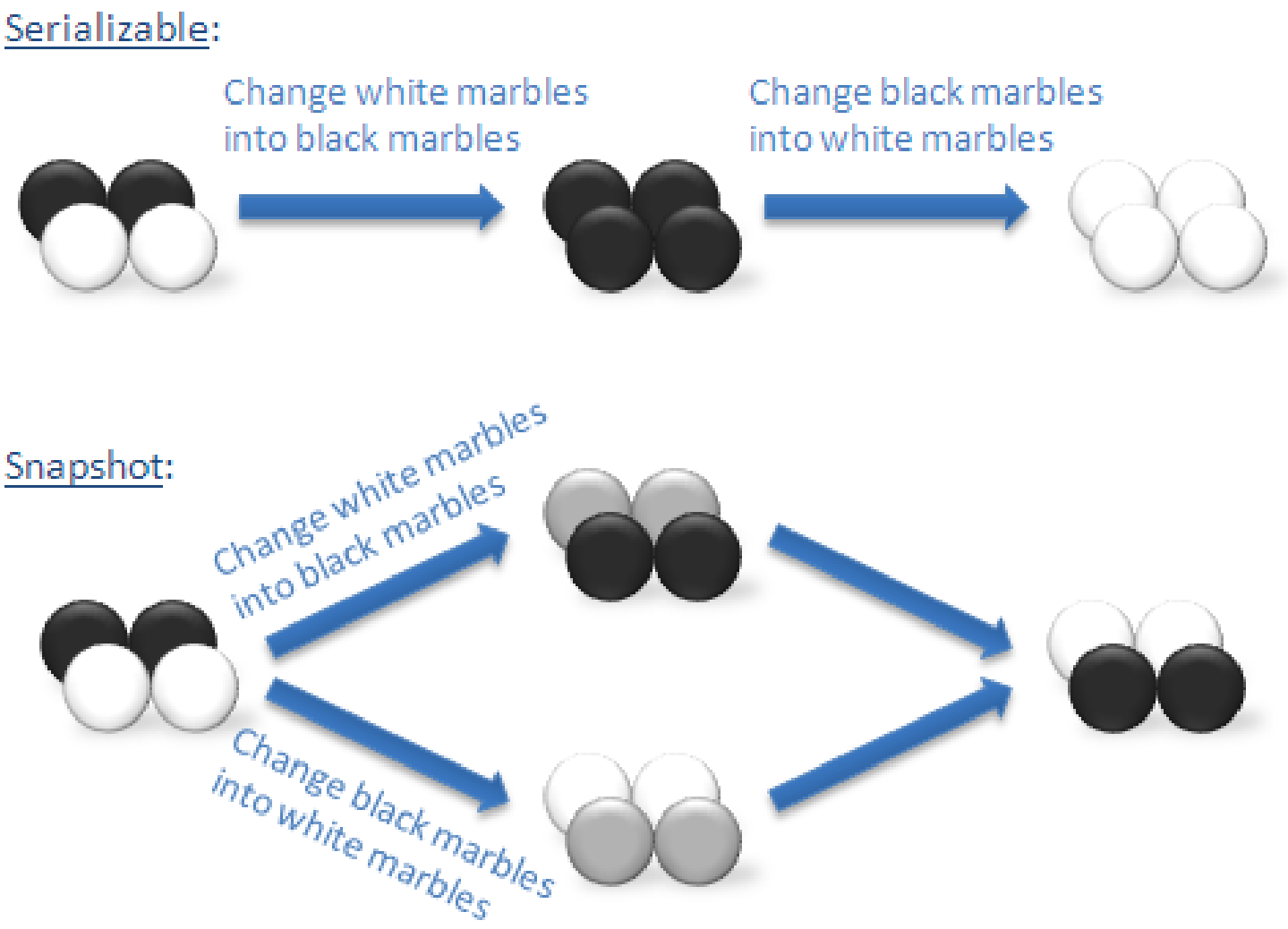
Poziomy izolacji MS SQL 2008+

Isolation level	Dirty read	Nonrepeatable read	Phantom
Read uncommitted	Yes	Yes	Yes
Read committed	No	Yes	Yes
Repeatable read	No	No	Yes
Snapshot	No	No	No
Serializable	No	No	No

Poziom izolacji Serializable vs Snapshot

- SERIALIZABLE – blokuje zakres wierszy (LOCK) aż do momentu zakończenia transakcji

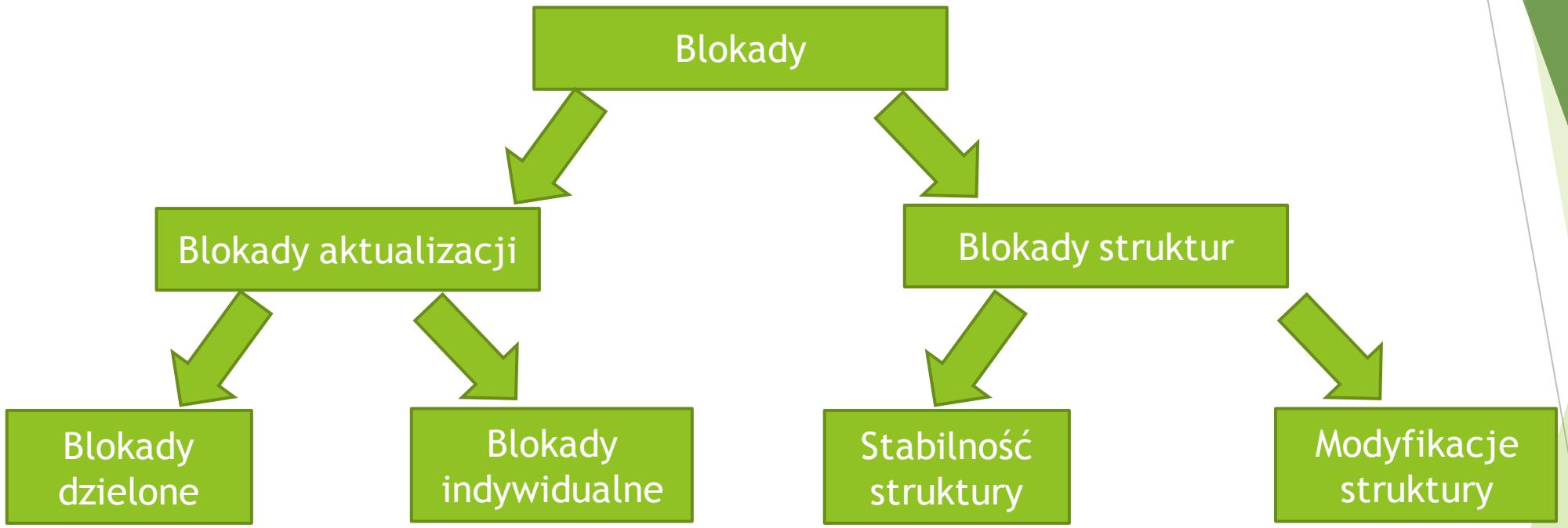
- SNAPSHOT - wykorzystuje technologię wersjonowania wierszy, transakcja widzi swoją lokalną kopię danych



Blokady aktualizacji

- Aby zapewnić odpowiedni poziom izolacji, SZBD (DBMS) może założyć różnego rodzaju blokady na:
  - Rekordy, które będą modyfikowane
  - Zakresy rekordów (strona w MS SQL Server)
  - Całe tabele
- Podczas wykonywania zapytania DBMS sprawdza czy rekordy objęte zapytaniem są zablokowane czy nie
- Blokady mogą eskalować – jeśli ponad 90% rekordów danej tabeli jest zablokowanych przez proces to wtedy cała tabela zostaje zablokowana. Baza danych Oracle nigdy nie eskaluje blokad.

Blokady



Blokady aktualizacji są często narzucane przez DBMS automatycznie. Odnoszą się bezpośrednio do poziomu izolacji i mogą powodować dostęp tylko do odczytu dla modyfikowanych obiektów (np. rekordu). Blokady indywidualne uniemożliwią dostęp dla innych transakcji.

Blokada stabilności struktury zapewnia, że obiekty (tabele, indeksy, itp.) nie zostaną usunięte, jeśli będą używane przez inną transakcję. Blokady modyfikacji struktury zapobiegają dostępowi do obiektów bazy danych, które ulegają modyfikacji (np. dodawanie kolumn).

Blokady w praktyce

Stan konta nr. 456	BEGIN TRANSACTION UPDATE Accounts set balance=balance+500 where account_id=456	
1000 PLN		
1500 PLN	. . . **ROLLBACK**	**BEGIN TRANSACTION**
1000 PLN		UPDATE Accounts set balance=balance+500 where account_id=456
900 PLN		UPDATE Accounts set balance=balance+100 where account_id=8899 **COMMIT**

Na poziomie izolacji READ COMMITED nastąpi blokada wiersza konta nr. 456. Stąd wykonanie drugiej transakcji zostanie wstrzymane dopóki blokada nie zostanie zdjęta przez zakończenie pierwszej transakcji.

Zakleszczenia (deadlocks)

- Zakleszczenie (deadlock) – to kombinacja blokad, która uniemożliwia na wykonanie jednej lub wielu transakcji oraz nie może być rozwiązana bez zakończenia z zewnątrz jednej z transakcji
- Zakończenie z zewnątrz transakcji oznacza użycie ROLLBACK dla tej transakcji przez co anuluje wszystkie aktualizacje – nie może być to zaakceptowane jako rozwiązanie

Zakleszczenie – prosty przykład

```
BEGIN TRANSACTION

UPDATE Accounts set
balance=balance+500
where account_id=456


UPDATE Accounts set
balance=balance+100
where account_id=8899

COMMIT
```

```
BEGIN TRANSACTION


UPDATE Accounts set
balance=balance+100
where account_id=8899



UPDATE Accounts set
balance=balance-100
where account_id=456
COMMIT
```

(1) Blokada konta 456
(3) Czekanie na zwolnienie blokady konta 8899

(2) Blokada konta 8899
(4) Czekanie na zwolnienie blokady konta 456

Zakleszczenia – jak ich unikać

- Zawsze blokuj zasoby w tej samej kolejności, dokładniej rób zapytania CRUD w takiej samej kolejności
- Przykład:
  - Zawsze blokuj numery konta począwszy od najniższego numeru
  - Zawsze blokuj tabelę Accounts przed datelą Customers

Jak monitorować blokady?

- MS SQL Server:
  - Enterprise Manager może być do tego użyteczny (poddrzewo Current activity)
  - Alternatywnie można korzystać z procedury składowanej sp_lock
- MySQL:
  - Polecenie „SHOW ENGINE INNODB STATUS\G”
  - Oracle Enterprise Manager for MySQL Database
- H2 server - pliki dbname.lock.db
- Pamiętaj:
  - Co do zasady – blokady są usuwane, kiedy transakcja się zakończy (nie zależnie czy poprzez COMMIT czy poprzez ROLLBACK)

Save points

- Przykład (MS SQL Server)

```
BEGIN TRANSACTION A

/* parę updatów */

SAVE TRAN B

/* parę updatów */

ROLLBACK TRAN B

COMMIT
```

Podajemy etykietę, aby móc później do niej wrócić jeśli będzie to konieczne

Nie zostanie wycofana cała transakcja – jedynie zmiany wykonane po SAVE TRAN B

Transakcje – czas wykonania

Kategoria	Opis
Standardowe krótkie transakcje (short-running)	• Nakładają blokady na zasoby, do których mają dostęp. Aby nie zmniejszać wydajności czas transakcji powinien być możliwie krótki. • Przeważająca większość transakcji to krótkie transakcje.
Długie transakcje (long-running)	• Wyobraźmy sobie modyfikacje projektu dla dużego przedsięwzięcia (np. sieć wodociągów dla nowej części miasta. Może zająć to wiele dni ay ukończyć nową wersję projektu. Nie chcielibyśmy, aby ktokolwiek widział zmiany, dopóki nie naniesiemy wszystkich zmian. • Długie transakcje nie mogą bazować na blokadach. To by zatrzymało jakiekolwiek prace nad projektem (np. dostęp tylko w trybie read-only). Powinna tu być stworzona kopia modyfikowanych danych. • Nie są tak popularne i częste w praktyce jak krótkie transakcje, natomiast często używane w ODBMS.

Problemy z równoległością – przykład 1

BD sesja 1

```
SET TRANSACTION ISOLATION LEVEL
READ COMMITTED

begin transaction
update customers
set ordercount=2
where customerId='ALFKI'
```

ROLLBACK

BD sesja 2

```
SET TRANSACTION ISOLATION LEVEL
READ UNCOMMITTED

begin transaction
Select * from customers
where customerId='ALFKI'
```

COMMIT

BD sesja 2 nie jest wstrzymana. Możliwe są brudne odczyty, ponieważ sesja 1 może być anulowana

Problemy z równoległością – przykład 2

BD sesja 1

```
SET TRANSACTION ISOLATION LEVEL
READ COMMITTED

begin transaction
update customers
set ordercount=2
where customerId='ALFKI'
```

ROLLBACK

BD sesja 2

```
SET TRANSACTION ISOLATION LEVEL
READ COMMITTED

begin transaction
Select * from customers
where customerId='ALFKI'
```

COMMIT

BD sesja 2 jest wstrzymana dopóki sesja 1 nie zostanie zatwierdzona lub anulowana.

Problemy z równoległością – przykład 3

BD sesja 1

```
SET TRANSACTION ISOLATION LEVEL
READ COMMITTED

begin transaction
select * from customers where
customerId='ALFKI'
```

BD sesja 2

```
SET TRANSACTION ISOLATION LEVEL
READ COMMITTED

begin transaction
update customers set ordercount=3
where customerId='ALFKI'
commit
```

```
select * from customers where
customerId='ALFKI'
```

BD sesja 2 nie jest wstrzymana. Możliwe są niepowtarzalne odczyty w sesji 1.

Problemy z równoległością – przykład 4

BD sesja 1

```
SET TRANSACTION ISOLATION LEVEL
REPEATABLE READ

begin transaction
select * from customers where
customerId='ALFKI'
```

BD sesja 2

```
SET TRANSACTION ISOLATION LEVEL
READ COMMITTED

begin transaction
update customers set ordercount=3
where customerId='ALFKI'
commit
```

```
select * from customers where
customerId='ALFKI'
```

BD sesja 2 jest wstrzymana. Nie są możliwe nie-powtarzalne odczyty w sesji 1, ponieważ wszystkie zmiany są wstrzymywane.

Anomalie – rozwiązanie multi-versioning

- Dostarczają wysoce równoległy dostęp do danych
- Wiele wersji danych jest może być utrzymywanych:
  - Użytkownik A rozpoczyna transakcję i modyfikuje dane w tabeli klientów
  - Użytkownik B czyta oryginalne dane, bez blokowania – dzięki wersjonowaniu
- Jako konsekwencja:
  - Dane czytelnika nigdy nie będą blokowane przez pisarza
  - Zapytania wyłącznie odczytujące dane nigdy nie spowodują zakleszczenia oraz nigdy nie dostaną nieistniejących danych
- Wersjonowanie może być na poziomie pojedynczego zapytania lub transakcji w zależności od poziomu izolacji

The multiversioning approach

Zwartość tabeli widziana przez czytelników nie blokowana przez transakcje modyfikujące

id_klienta	Id_zamówienia	kraj
1990	254	PL
1991	458	UK
1992	112265	PL
1993	85	DE

Zwartość tabeli widziana przez transakcję 1, która zmodyfikowała zawartość i dodała wiersz, ale nie zatwierdziła zmian

id_klienta	Id_zamówienia	kraj
1990	254	PL
1991	772	UK
1992	112265	PL
1993	85	DE
1994	4596	DE

Każde zapytanie czy transakcja przechowuje swoją własną logicznie wersję danych. Ta funkcjonalność jest możliwa dzięki analizie online segmentów, które można wycofać (ROLLBACK). Używając tego DBMS może praktycznie odtworzyć zawartość bazy danych dla dowolnego momentu w przeszłości w zależności od długości logów transakcji.

Poziomy izolacji – baza danych Oracle

Poziom izolacji	DR	NRR	PH	Uwagi
READ COMMITED	NIE	NIE	NIE	Zjawisko „brudnych odczytów” nie występuje dzięki wersjonowaniu. Nigdy nie ma blokowania wierszy do odczytu, ponieważ Oracle DBMS może przywrócić wersję bazy danych sprzed rozpoczęcia transakcji i ją przez cały czas utrzymywać dla aktualnej transakcji.
SERIALIZABLE	NIE	NIE	NIE	Spójność odczytu, która jest dla pojedynczego zapytania jest zachowana dla całej transakcji. Każde zapytanie da powtarzalne wyniki podczas całego życia transakcji. Transakcja będzie cały czas widziała takie dane jakie były, kiedy się rozpoczęła.
READ ONLY	NIE	NIE	NIE	Ma wszystkie cechy SERIALIZABLE, ale nie pozwala na modyfikowanie.

DR – dirty reads, NRR – non-repeatable reads, PH – phantoms

Oracle - transakcje

- Transakcje są rozpoczynane domyślnie (nie ma potrzeby używać BEGIN TRANSACTION)
- SET TRANSACTION ISOLATION LEVEL może być użyte tylko raz jako pierwsza instrukcja transakcji
- Tak jak w większości systemów bazodanowych READ COMMITTED jest najczęściej używanym poziomem izolacji (TIL)

Problemy z równoległością – przykład 1

BD sesja 1

```
SET TRANSACTION ISOLATION LEVEL
READ COMMITTED

update customers
set ordercount=2
where customerId='ALFKI'
```

BD sesja 2

```
SET TRANSACTION ISOLATION LEVEL
READ COMMITTED

Select * from customers
where customerId='ALFKI'
```

ROLLBACK

COMMIT

BD sesja 2 nie jest wstrzymana, jednak nie zobaczy żadnych zmian naniesionych przez sesję 1.

Problemy z równoległością – przykład 3

BD sesja 1

```
SET TRANSACTION ISOLATION LEVEL
READ COMMITTED

select * from customers
where customerId='ALFKI'
```

BD sesja 2

```
SET TRANSACTION ISOLATION LEVEL
READ COMMITTED

update customers set ordercount=3
where customerId='ALFKI'
commit
```

```
select * from customers where
customerId='ALFKI'
```

BD sesja 2 nie jest wstrzymana. Niemożliwe są niepowtarzalne odczyty w sesji 1.

Problemy z równoległością – przykład 4

BD sesja 1	BD sesja 2
SET TRANSACTION ISOLATION LEVEL READ COMMITED insert into customers (customerId,ename,...) values ('ALFKI2', …)	
	SET TRANSACTION ISOLATION LEVEL READ SERIALIZABLE
COMMIT	
	select * from customers where customerId='ALFKI'

BD sesja 2 nie zauważy dodanego rekordu o nowym kliencie. Ponieważ mamy poziom izolacji SERIALIZABLE, to wszystkie zapytania zwrócą zawartość zatwierdzoną w momencie rozpoczęcia transakcji.

Oracle Multi-versioning - dyskusja

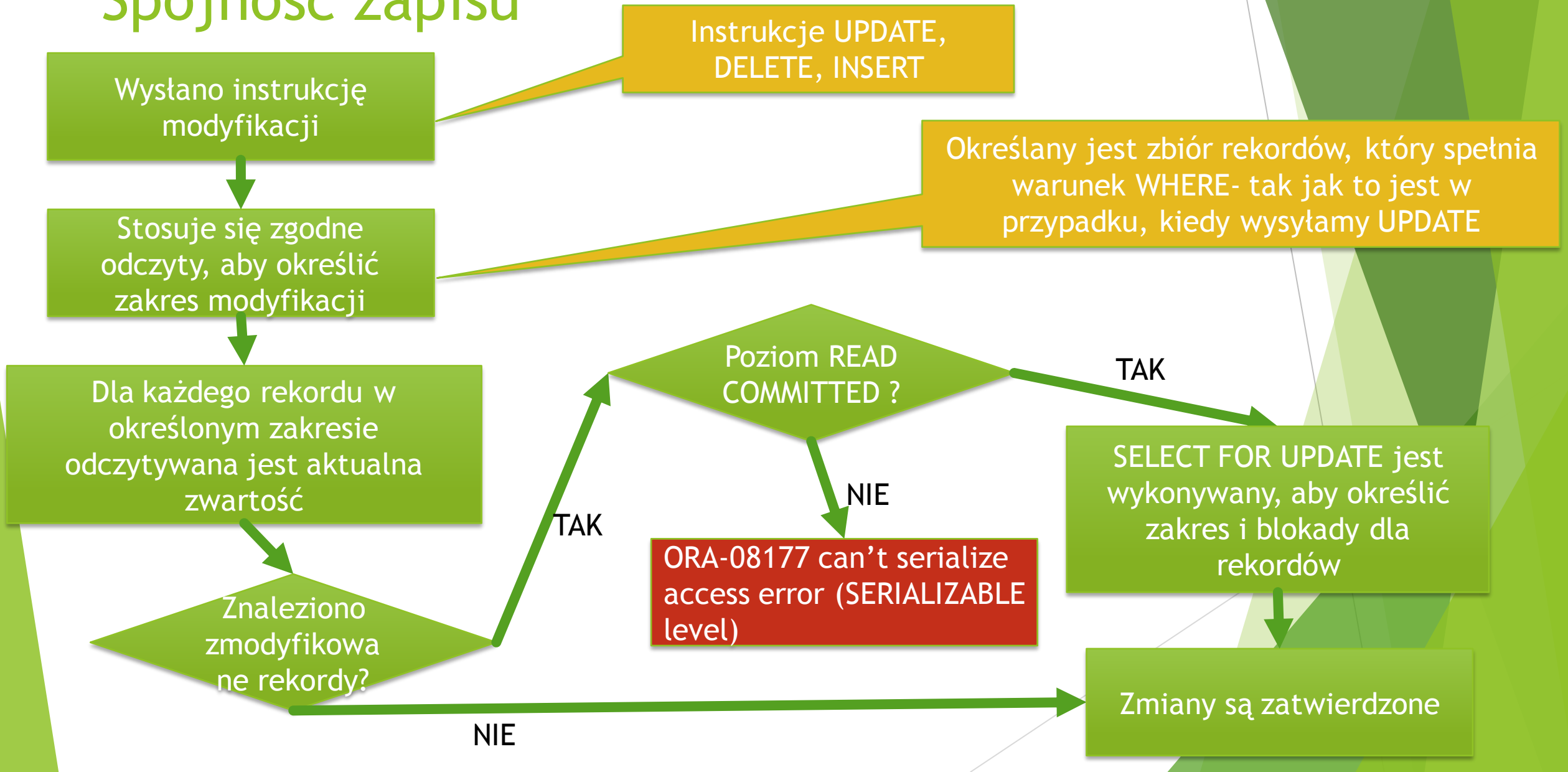
- SERIALIZABLE TIL
  - Zakłada, że żadne zmiany nie będą robione w transakcji równolegle
  - „Can't serialize access for this transaction" – błąd może wystąpić, gdy próbujemy modyfikować dane zmienione przez inne transakcje od momentu rozpoczęcia transakcji
  - Nadal SELECT … FOR UPDATE może być użyty, aby serializować (narzucić sekwencyjność) dostęp do wybranych rekordów
- READ ONLY TIL
  - Możemy natrafić na błąd „Snapshot too old error" – to się zdarza, gdy segmenty wycofania (ROLLBACK), zawierające wymagany stan danych w przeszłości, zostały już nadpisane z powodu nieodpowiedniego rozmiaru segmentów wycofania. Nie powinno to mieć miejsca w systemie produkcyjnym z odpowiednimi rozmiarami segmentów wycofywania.

Problemy z równoległością – przykład 3

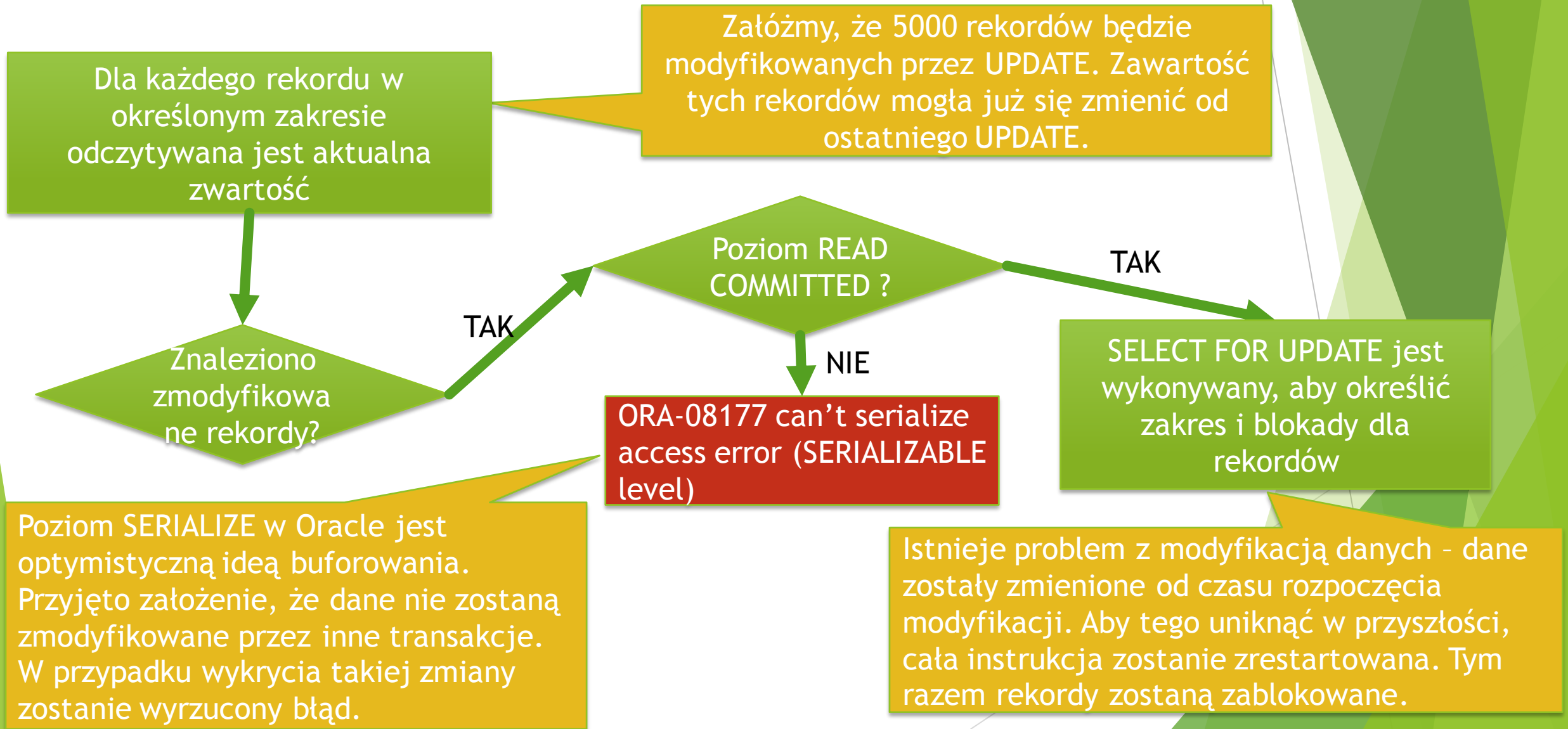
BD sesja 1	BD sesja 2
SET TRANSACTION ISOLATION LEVEL … update customers set ordercount=3 where customerId='ALFKI'	
	SET TRANSACTION ISOLATION LEVEL … update customers set ordercount=2 where customerId='ALFKI' commit
COMMIT	
	select * from customers where customerId='ALFKI'

BD sesja 2 jest wstrzymana na blokadzie zapisu nałożonej przez transakcję BD sesja 1, tak jak to ma miejsce w innych SZBD

Spójność zapisu



Spójność zapisu - dyskusja



Spójność zapisu - konsekwencje

- Triggery mogą zostać uruchomione DWA RAZY dla tego samego rekordu poprzez pojedynczą instrukcję UPDATE
  - Kiedy po raz pierwszy zatwierdza zmiany – przed wykryciem „brudnego” rekordu
  - Kiedy po raz drugi zatwierdza zmiany – po ponownym odczycie danych w trybie SELECT FOR UPDATE
- Triggery nie powinny być uruchamiana w autonomicznych transakcjach
- Jeśli gorące dane są często modyfikowane przez różne transakcje, algorytm aktualizacji dwufazowej może generować znaczne obciążenie

Podsumowanie

- Multi-versioning może mieć negatywny wpływ na wydajność w niektórych przypadkach:
  - Wyobraźmy sobie długą kwerendę, która czyta dużą ilość danych
  - Niektóre dane mogły być zmodyfikowane chwilę po starcie kwerendy, stąd logi wycofania (rollback logs) muszą być sprawdzone aby otrzymać oryginalne dane – tak aby zapewnić spójność wersji
  - Jako konsekwencja: „the longer query runs, the longer query runs”
- Mechanizmy blokowania wbudowane w rekord – SZBD nie wyrwie się z blokad
- Nie można zakładać, że znając jedne SZBD możemy zgadnąć jak dział inny SZBD
- Należy być świadomym szczegółów implementacji różnych SZBD , aby uniknąć błędów i/lub słabej wydajności
- Nie należy próbować unikać przetwarzania transakcyjnego (zwłaszcza na DBMS Oracle…)

Pytania?