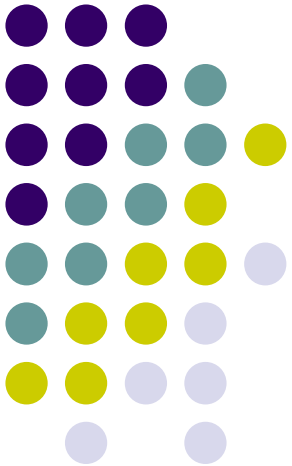
wykład

Zarządzanie współbieżnym wykonywaniem transakcji Cz.1



Plan i cel wykładu

- Celem wykładu jest przedstawienie i omówienie podstawowych algorytmów zarządzania współbieżnym wykonywaniem transakcji.

1/2

- Rozpoczniemy od przedstawienia algorytmów blokowania.
  - algorytm blokowania dwu-fazowego.
- Następnie przedstawimy zjawisko
  - zakleszczenia i omówimy podstawowe algorytmy rozwiązywania zakleszczenia.

2/2

- Na zakończenie wykładu, przedstawimy i omówimy problem duchów.

klasyfikacja algorytmów

Algorytmy zarządzania współbieżnym wykonywaniem transakcji możemy sklasyfikować następująco:

1. algorytmy blokowania - uszeregowanie transakcji wynika z kolejności uzyskiwanych blokad (algorytm blokowania dwufazowego – 2PL)

2. algorytmy znaczników czasowych – uszeregowanie transakcji wynika z wartości znaczników czasowych związanych z transakcjami

3. algorytmy optymistyczne - walidacja poprawności uszeregowania

Walidacja

algorytm blokowania (1)

• Blokada jest zmienną skojarzoną z każdą daną w bazie danych, określającą dostępność danej ze względu na możliwość wykonania na niej określonych operacji

• Ogólnie, z każdą daną mamy skojarzoną jedną blokadę Ze względu na proces blokowania, dane w bazie danych mogą występować w jednym z trzech stanów:

- dana nie zablokowana (*0*)
- dana zablokowana dla odczytu *R* (współdzielona *S*)
- dana zablokowana dla zapisu *W* (wyłączna *X*)

Algorytmy blokowania → zarządzanie współbieżnym wykonywaniem transakcji
Mechanizm blokad zakładanych przez transakcje.
Blokada – definicja
Trzy stany danych w BD

algorytm blokowania (2)

• System zarządzania bazą danych musi realizować trzy dodatkowe operacje na bazie danych:

- Blokowanie danej x do odczytu (LR(x))
- Blokowanie danej x do zapisu (LW(x))
- Odblokowanie danej x (UNL(x))

• Operacje blokowania muszą poprzedzać wykonanie operacji odczytu oraz zapisu danej

Podstawowy zbiór operacji transakcji (**odczyt, zapis, zatwierdzenie, wycofanie**) rozszerzony zostanie o 3 dodatkowe operacje, tj.:

- **blokowanie danej x do odczytu (LR(x));**
- **blokowanie danej x do zapisu (LW(x));**
- **odblokowanie danej x (UNL(x)).**

kompatybilność blokad

Dwie blokady są kompatybilne jeżeli mogą być założone na tej samej danej przez dwie różne transakcje

Blokada żądana \ Blokada uzyskana	R	W
R	✓	-
W	-	-

Pojęcie kompatybilności blokad - definicja.
Macierz kompatybilności blokad.

konwersja blokad

Transakcja posiadająca blokadę określonego typu na danej może dokonać jej konwersji w blokadę innego typu

Macierz konwersji blokad

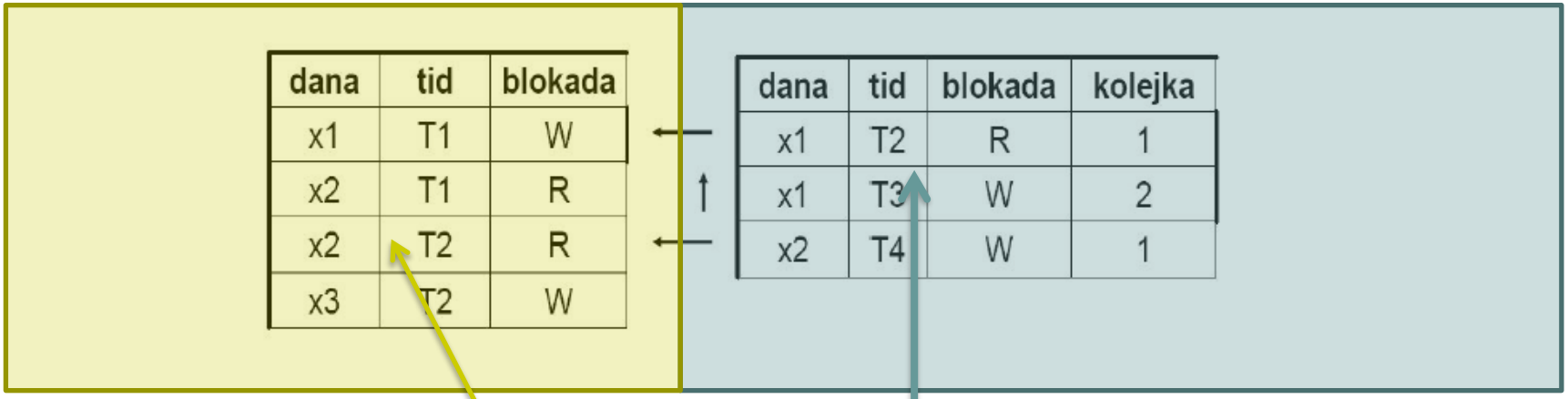
Blokada żądana \ Blokada uzyskana	R	W
R	✓	-
W	✓	✓

konwersja niedopuszczalna

Pojęcie konwersji blokad

implementacja algorytmów blokowania (1)

Struktury danych



Dwie kolejki transakcji, tj.:

- **kolejka transakcji, które uzyskały dostęp do danej;**
- **kolejka transakcji oczekujących na dostęp do danej.**

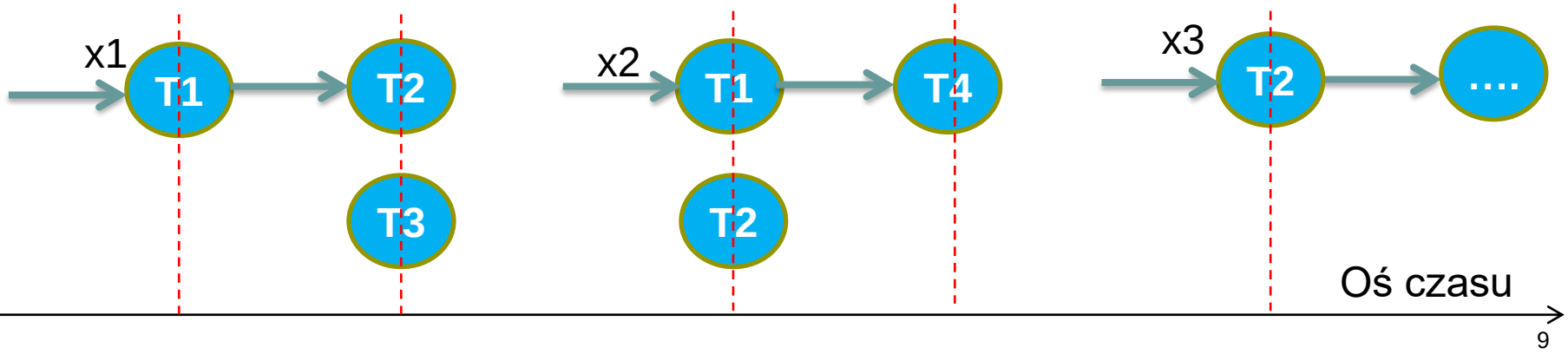
implementacja algorytmów blokowania (1)

Struktury danych

dana	tid	blokada
x1	T1	W
x2	T1	R
x2	T2	R
x3	T2	W

dana	tid	blokada	kolejka
x1	T2	R	1
x1	T3	W	2
x2	T4	W	1

Analiza przykładu:



implementacja algorytmów blokowania (2)

Algorytmy zakładania i zdejmowania blokad 1

Algorytm zakładania blokady do zapisu 2

Algorytm odblokowania danej X przez transakcję tid 3

implementacja algorytmów blokowania (2)

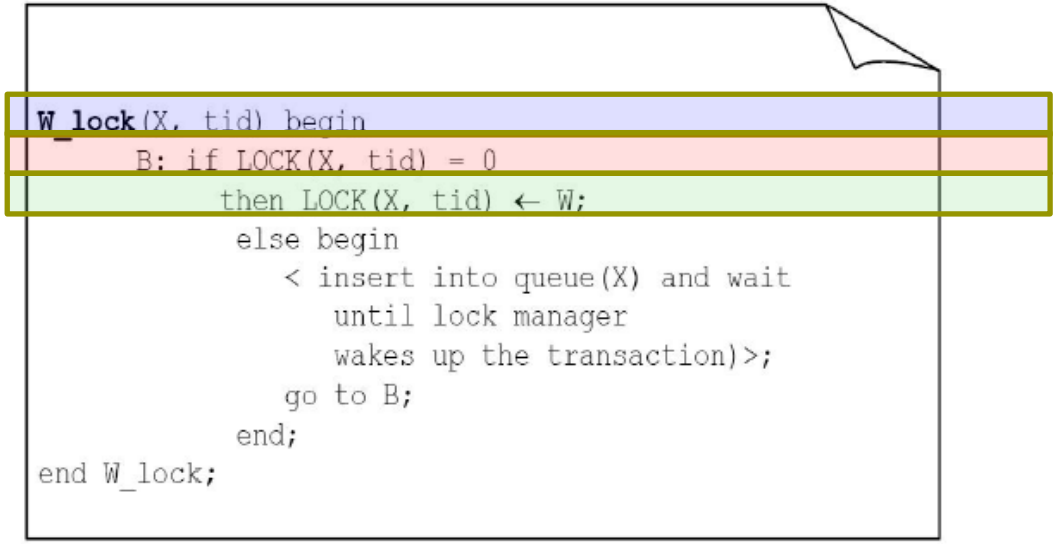
Operacje: ***LOCK, R_lock, W_lock, Unlock***

```
LOCK(X, tid) {0, R, W}
R_lock(X, tid) begin
      B: if (LOCK(X, tid)=0 or LOCK(X, tid)=R)
       then LOCK(X, tid) ← R;
       else begin
          < insert into queue(X) and wait
            until lock
            manager wakes up the transaction)>;
          go to B;
      end;
end R_lock;
```

Algorytmy zakładania i zdejmowania blokad

1

implementacja algorytmów blokowania (3)

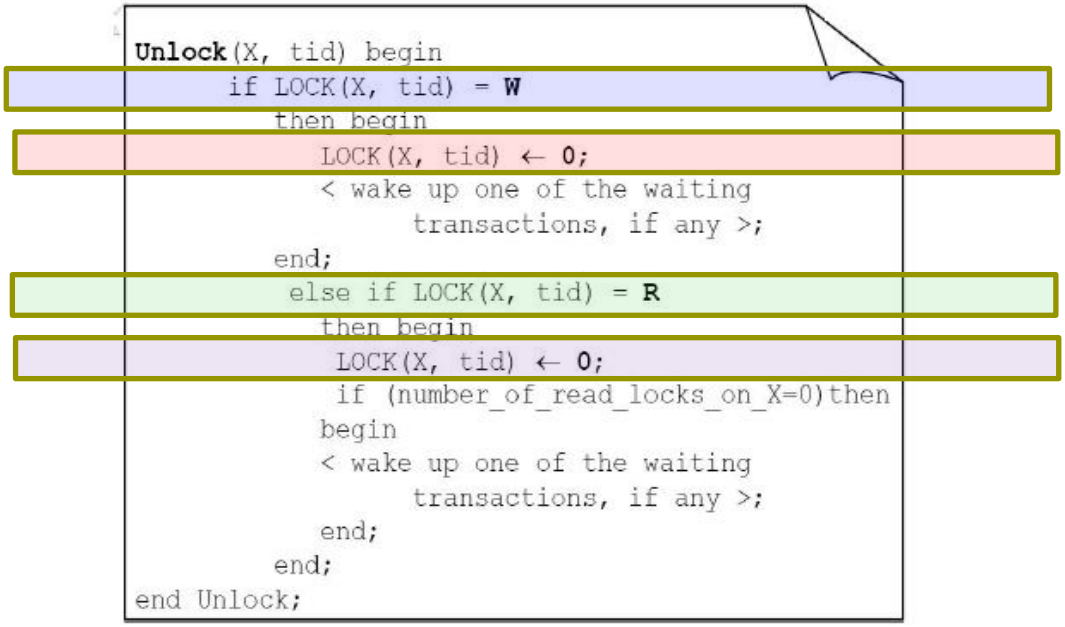


```
W_lock(X, tid) begin
      B: if LOCK(X, tid) = 0
           then LOCK(X, tid) ← W;
           else begin
              < insert into queue(X) and wait
                 until lock manager
                 wakes up the transaction)>;
              go to B;
           end;
end W_lock;
```

Algorytm zakładania blokady do zapisu

2

implementacja algorytmów blokowania (4)

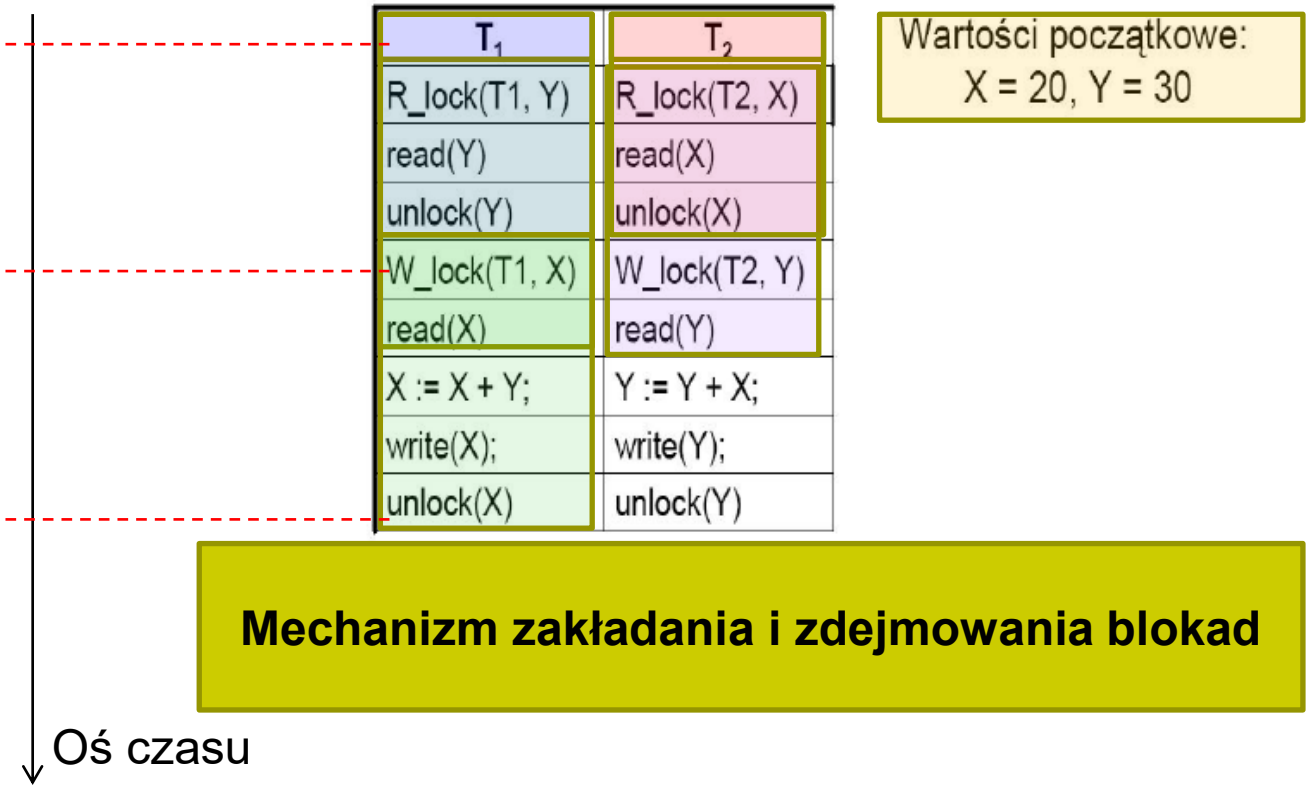


```
Unlock(X, tid) begin
      if LOCK(X, tid) = W
         then begin
            LOCK(X, tid) ← 0;
            < wake up one of the waiting
                  transactions, if any >;
         end;
          else if LOCK(X, tid) = R
            then begin
             LOCK(X, tid) ← 0;
             if (number_of_read_locks_on_X=0)then
            begin
            < wake up one of the waiting
                  transactions, if any >;
            end;
         end;
end Unlock;
```

Algorytm odblokowania danej X przez transakcję tid

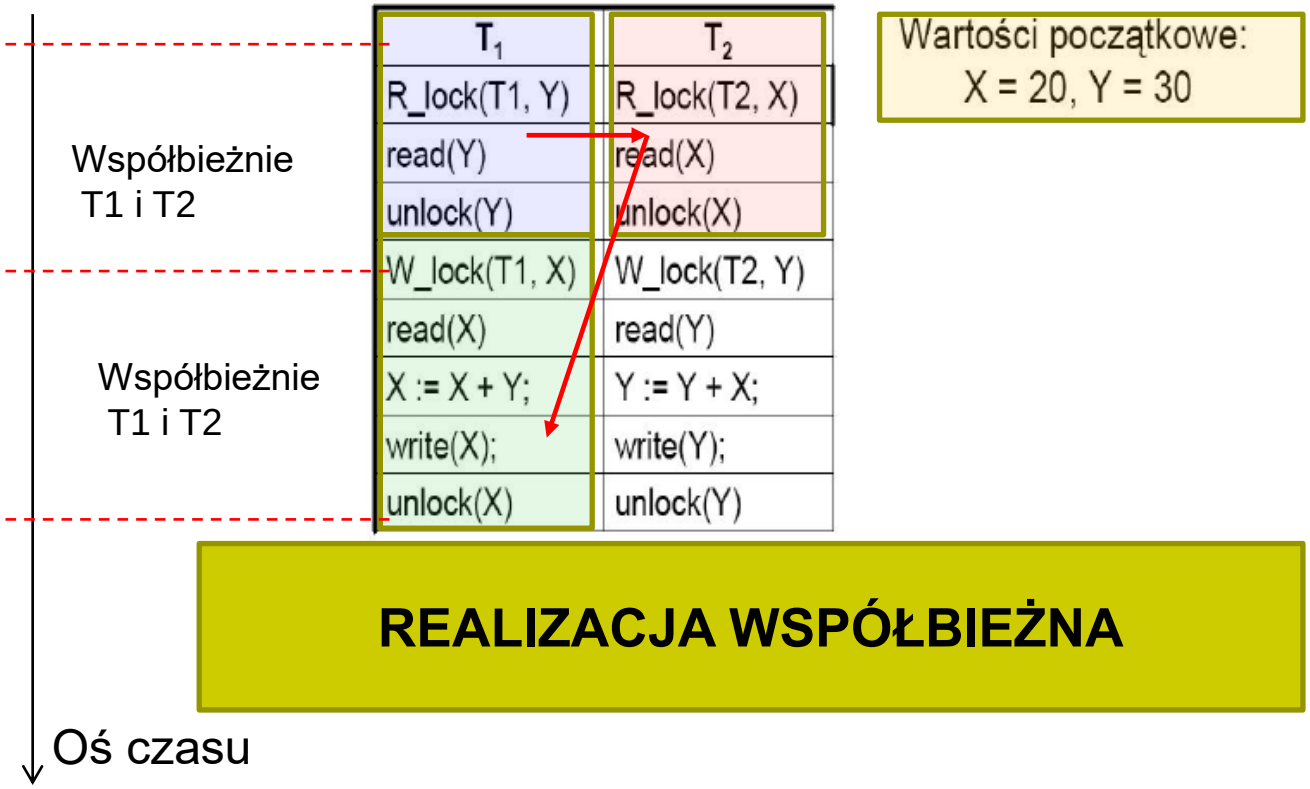
3

algorytm blokowania (1)



Przykład działania mechanizmu zakładania i zdejmowania blokad
Załóżmy, że wartości początkowe danych X i Y wynoszą, odpowiednio, 20 i 30.

algorytm blokowania (1)



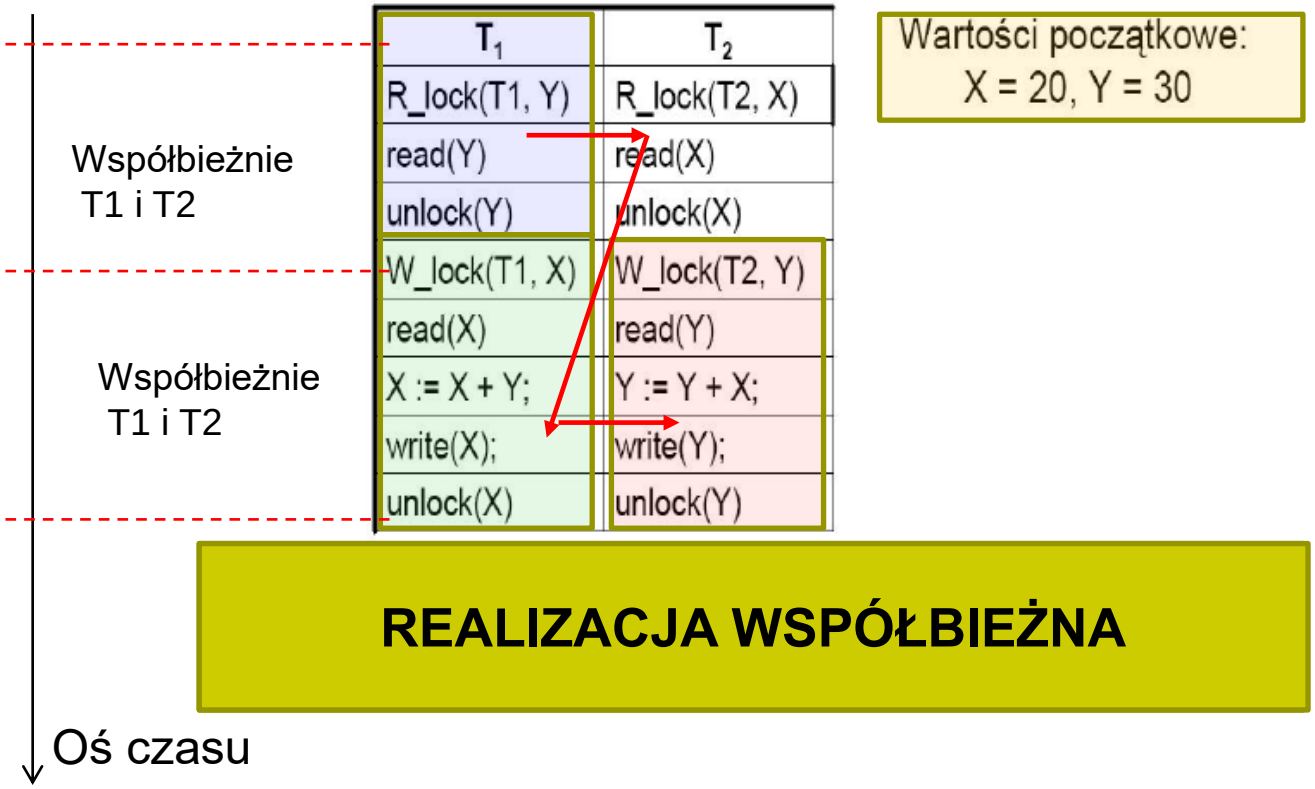
T_1	T_2
R_lock(T1, Y)	R_lock(T2, X)
read(Y)	read(X)
unlock(Y)	unlock(X)
W_lock(T1, X)	W_lock(T2, Y)
read(X)	read(Y)
X := X + Y;	Y := Y + X;
write(X);	write(Y);
unlock(X)	unlock(Y)

Wartości początkowe:
X = 20, Y = 30

REALIZACJA WSPÓŁBIEŻNA

Przykład działania mechanizmu zakładania i zdejmowania blokad

algorytm blokowania (1)

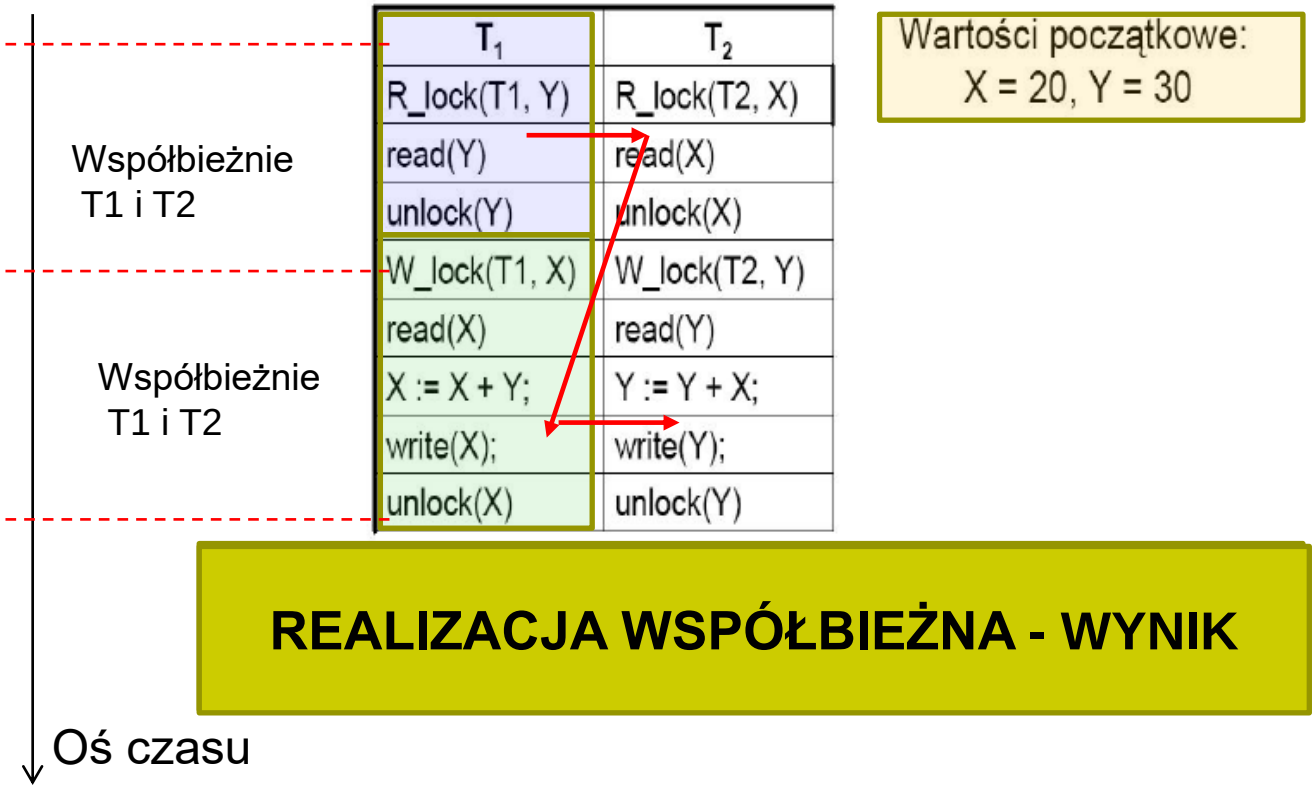


Wartości początkowe:
X = 20, Y = 30

REALIZACJA WSPÓŁBIEŻNA

Przykład działania mechanizmu zakładania i zdejmowania blokad

algorytm blokowania (1)



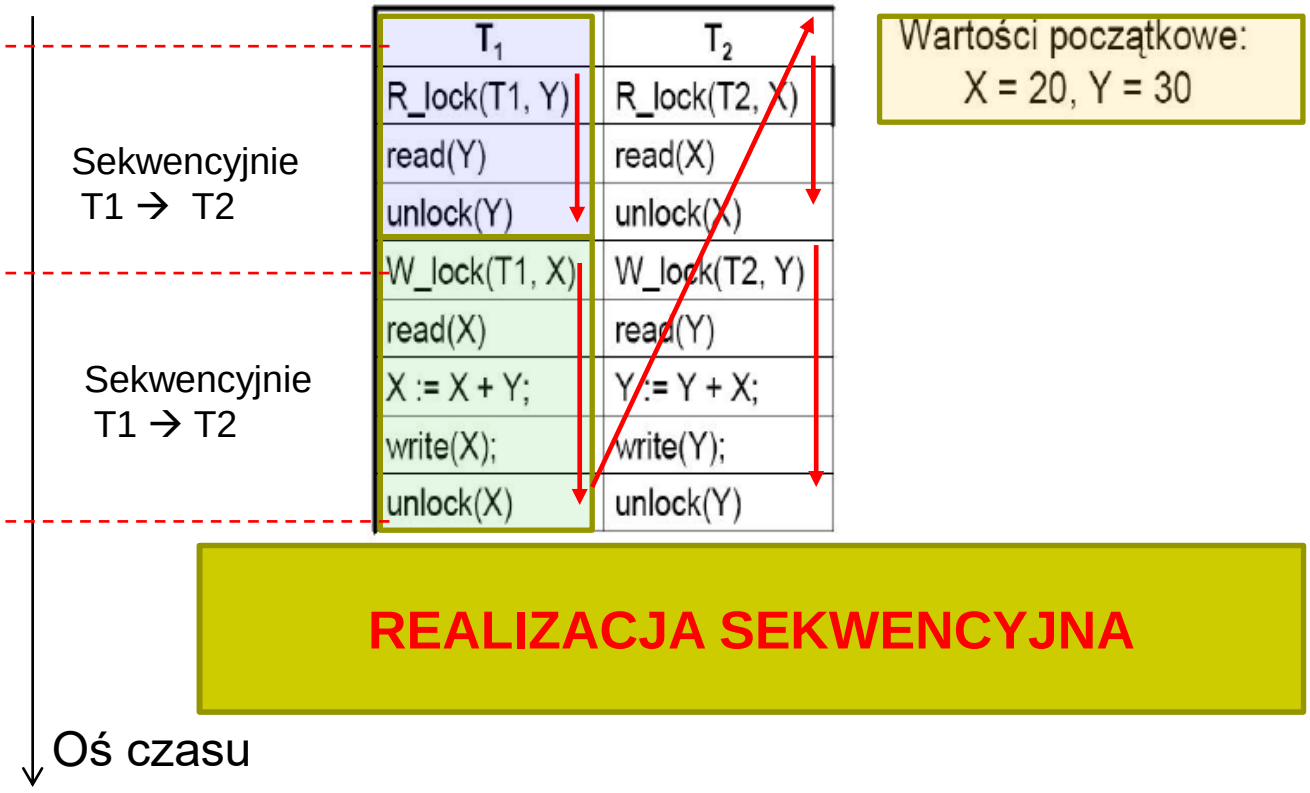
T_1	T_2
R_lock(T1, Y)	R_lock(T2, X)
read(Y)	read(X)
unlock(Y)	unlock(X)
W_lock(T1, X)	W_lock(T2, Y)
read(X)	read(Y)
X := X + Y;	Y := Y + X;
write(X);	write(Y);
unlock(X)	unlock(Y)

Wartości początkowe:
X = 20, Y = 30

REALIZACJA WSPÓŁBIEŻNA - WYNIK

Końcowy stan bazy danych uzyskany w wyniku przedstawionego współbieżnego wykonania transakcji T1 i T2 wynosi: **X=50 i Y=50.**

algorytm blokowania (1)



T_1	T_2
R_lock(T1, Y)	R_lock(T2, X)
read(Y)	read(X)
unlock(Y)	unlock(X)
W_lock(T1, X)	W_lock(T2, Y)
read(X)	read(Y)
X := X + Y;	Y := Y + X;
write(X);	write(Y);
unlock(X)	unlock(Y)

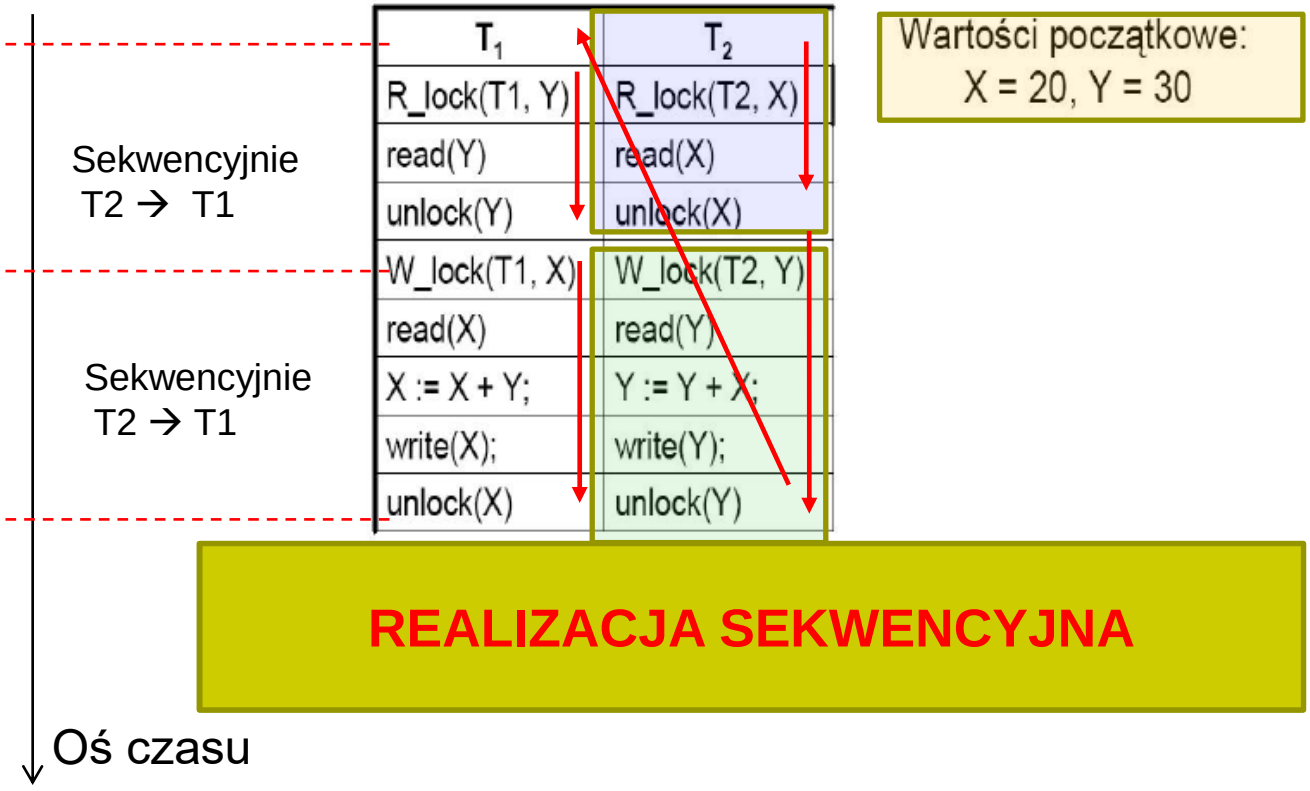
Wartości początkowe:
X = 20, Y = 30

REALIZACJA SEKWENCYJNA

Realizacja sekwencyjna:
T1 → T2: **X=50 i Y=80**

1

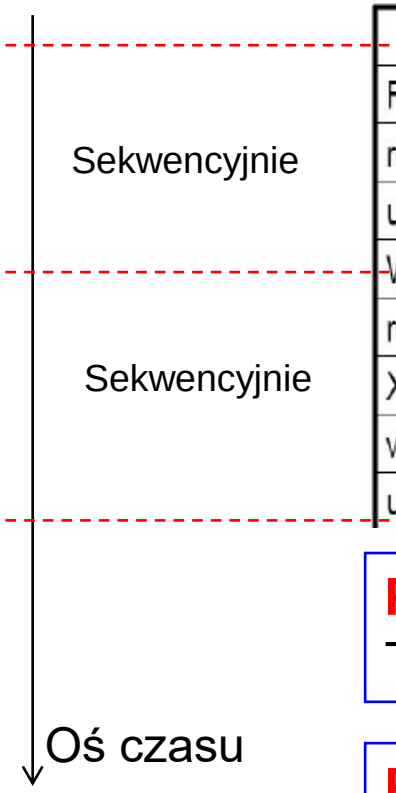
algorytm blokowania (1)



Realizacja sekwencyjna:
T2 → T1: **X=70 i Y=50**

2

algorytm blokowania (1)



T_1	T_2
R_lock(T1, Y)	R_lock(T2, X)
read(Y)	read(X)
unlock(Y)	unlock(X)
W_lock(T1, X)	W_lock(T2, Y)
read(X)	read(Y)
X := X + Y;	Y := Y + X;
write(X);	write(Y);
unlock(X)	unlock(Y)

Wartości początkowe:
X = 20, Y = 30

Realizacja sekwencyjna:
T1 → T2: **X=50 i Y=80**

1

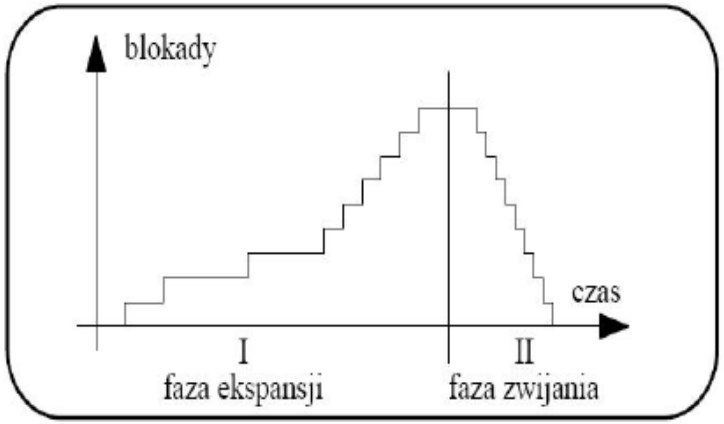
Realizacja sekwencyjna:
T2 → T1: **X=70 i Y=50**

2

WNIOSKI:

1. Przedstawiona realizacja współbieżna transakcji jest nieuszeregowalna.
2. Stosowanie blokad na danych nie gwarantuje automatycznie uszeregowalności realizacji zbioru transakcji.

algorytm blokowania dwufazowego (1)

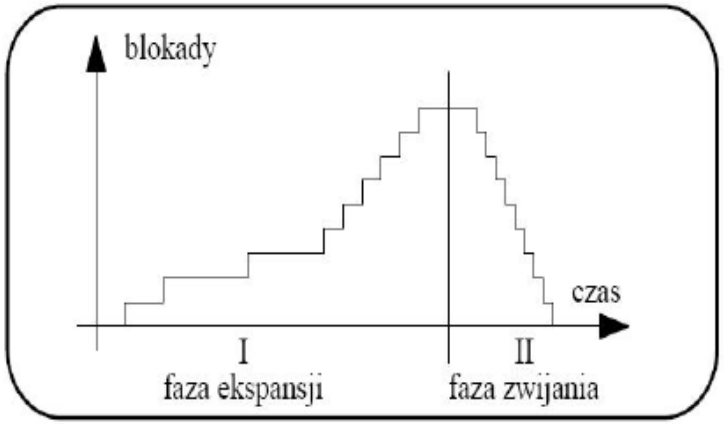


Algorytm podstawowy:

1. Każda operacja *read(X)* danej transakcji *T* musi być poprzedzona operacją *R_lock(X, T)* lub *W_lock(X, T)*
2. Każda operacja *write(X)* danej transakcji *T* musi być poprzedzona operacją *W_lock(X, T)*
3. Operacje *unlock(x, T)* dla danej transakcji *T* są wykonywane po zakończeniu wszystkich operacji *read* i *write*

Podstawowy algorytm blokowania:
algorytm blokowania dwufazowego (2PL: two-phase-locking).

algorytm blokowania dwufazowego (1)

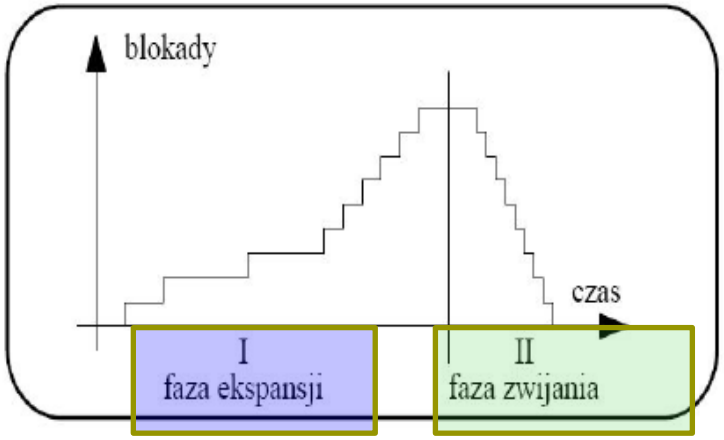


Algorytm podstawowy:

1. Każda operacja *read(X)* danej transakcji *T* musi być poprzedzona operacją *R_lock(X, T)* lub *W_lock(X, T)*
2. Każda operacja *write(X)* danej transakcji *T* musi być poprzedzona operacją *W_lock(X, T)*
3. Operacje *unlock(x, T)* dla danej transakcji *T* są wykonywane po zakończeniu wszystkich operacji *read* i *write*

Podstawowa wersja algorytmu 2PL

algorytm blokowania dwufazowego (1)



Algorytm podstawowy:

1. Każda operacja *read(X)* danej transakcji *T* musi być poprzedzona operacją *R_lock(X, T)* lub *W_lock(X, T)*
2. Każda operacja *write(X)* danej transakcji *T* musi być poprzedzona operacją *W_lock(X, T)*
3. Operacje *unlock(x, T)* dla danej transakcji *T* są wykonywane po zakończeniu wszystkich operacji *read* i *write*

algorytm blokowania dwufazowego (2)

- Algorytm statyczny: (1., 2., 3.)

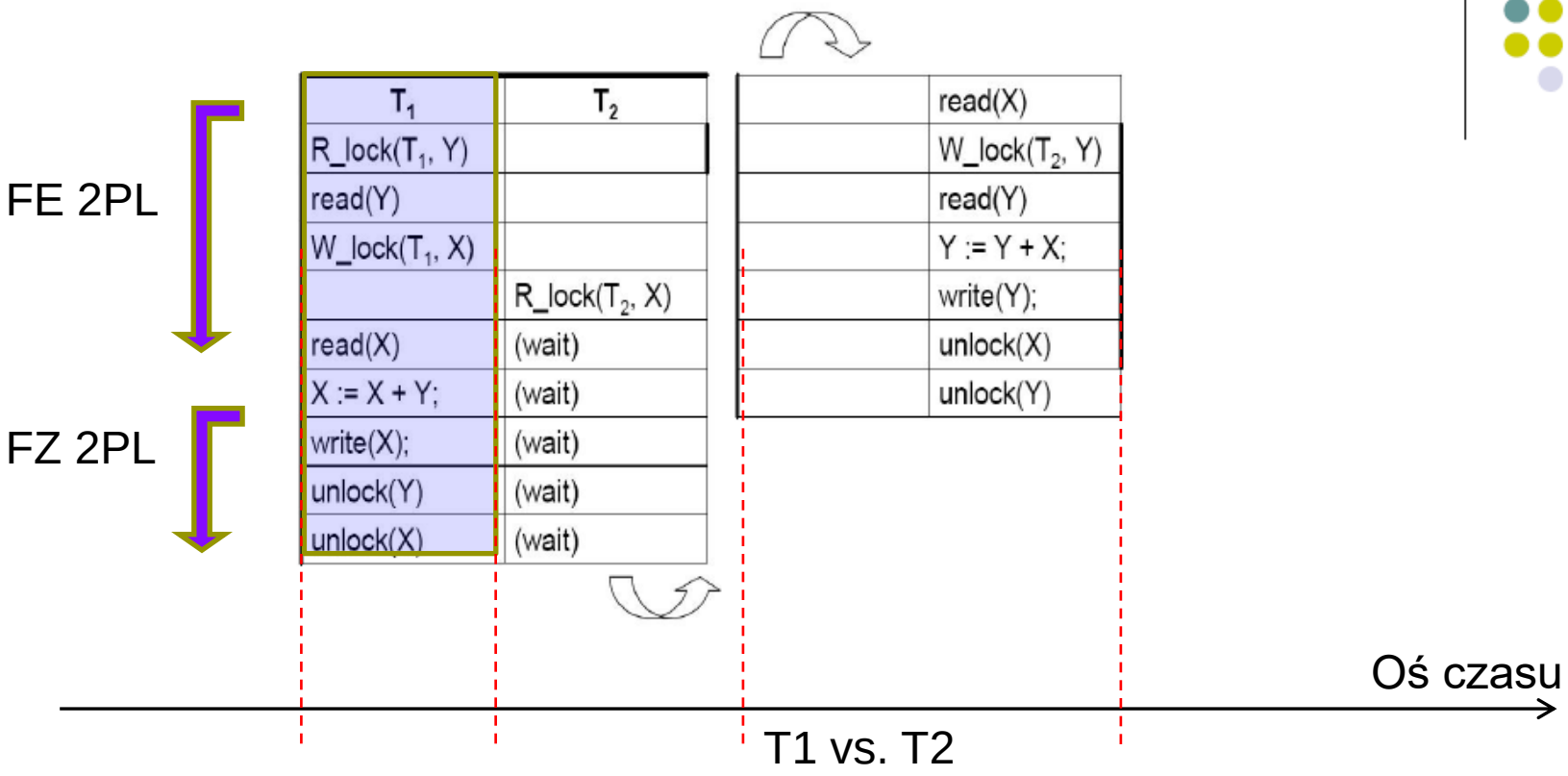
Wszystkie blokady muszą być uzyskane przed rozpoczęciem transakcji (przez predeklarowanie zbioru odczytywanych i modyfikowanych danych)

1

- Algorytm restryktywny: (1., 2.) Operacje *unlock(x, T)* dla danej transakcji *T* są wykonywane po operacji *commit* lub *rollback*

2

algorytm blokowania dwufazowego (3)

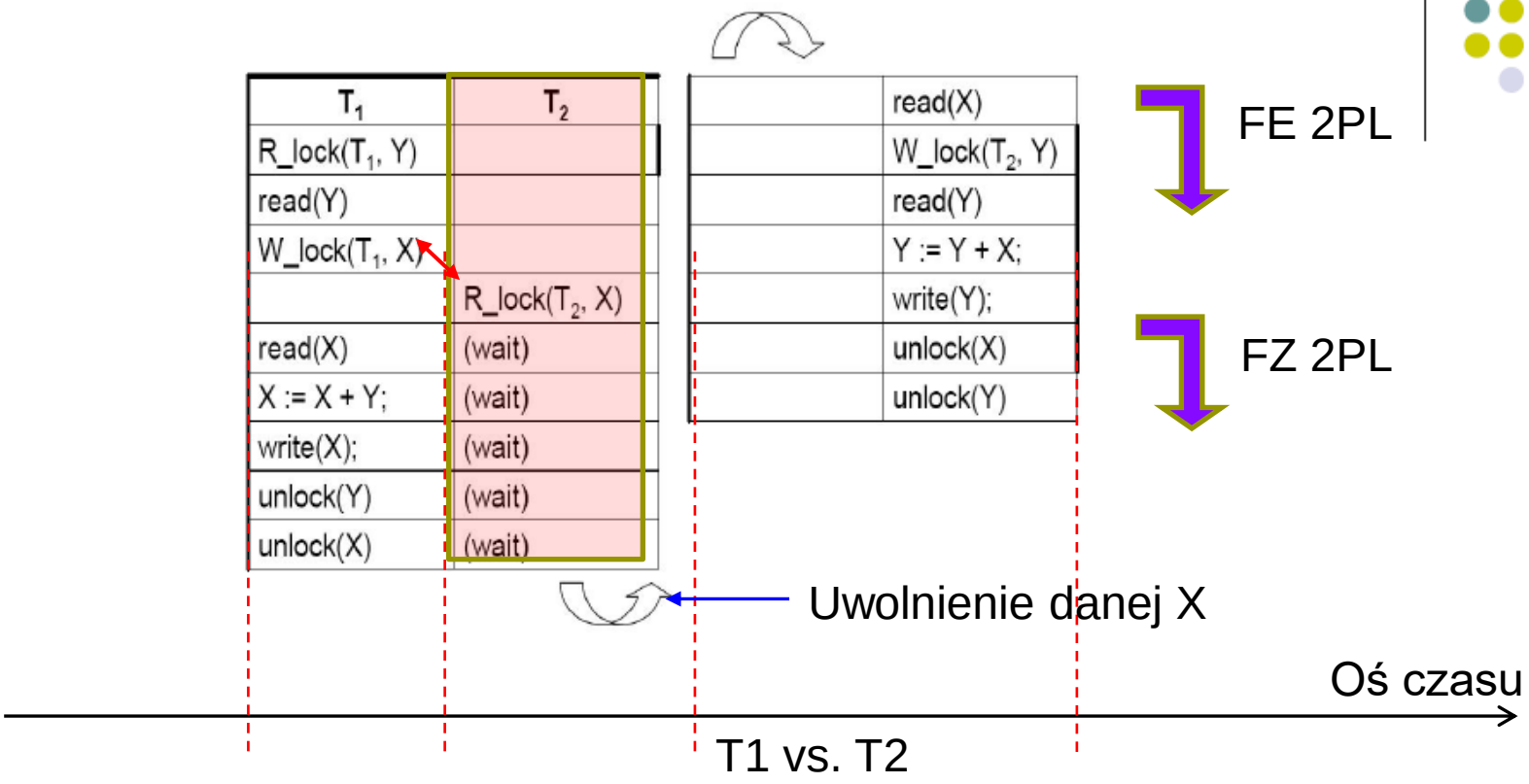


T_1	T_2
R_lock(T_1, Y)	
read(Y)	
W_lock(T_1, X)	
	R_lock(T_2, X)
read(X)	(wait)
X := X + Y;	(wait)
write(X);	(wait)
unlock(Y)	(wait)
unlock(X)	(wait)

	read(X)
	W_lock(T_2, Y)
	read(Y)
	Y := Y + X;
	write(Y);
	unlock(X)
	unlock(Y)

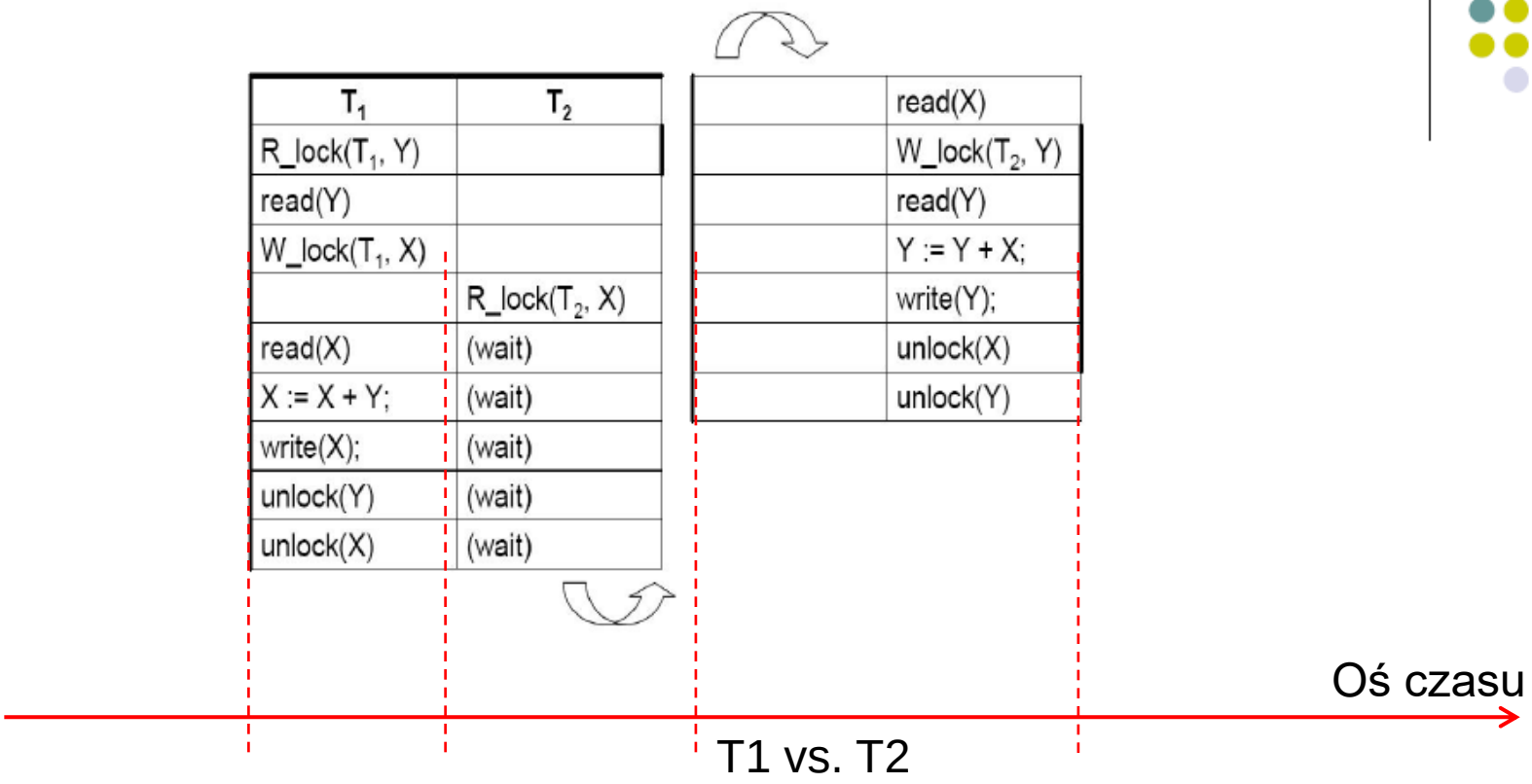
Działanie algorytmu blokowania dwufazowego: przykładowa realizacja transakcji T1 i T2

algorytm blokowania dwufazowego (3)



Działanie algorytmu blokowania dwufazowego: przykładowa realizacja transakcji T1 i T2

algorytm blokowania dwufazowego (3)



T_1	T_2
$R\_lock(T_1, Y)$	
read(Y)	
$W\_lock(T_1, X)$	
	$R\_lock(T_2, X)$
read(X)	(wait)
X := X + Y;	(wait)
write(X);	(wait)
unlock(Y)	(wait)
unlock(X)	(wait)

	read(X)
	$W\_lock(T_2, Y)$
	read(Y)
	Y := Y + X;
	write(Y);
	unlock(X)
	unlock(Y)

Końcowy stan bazy danych: współbieżna realizacja:**T1** i **T2**
X=50 i Y=80.

WNIOSEK:
Przedstawiona realizacja współbieżna transakcji jest uszeregowalna

KONIEC WYKŁADU
Cz. 1