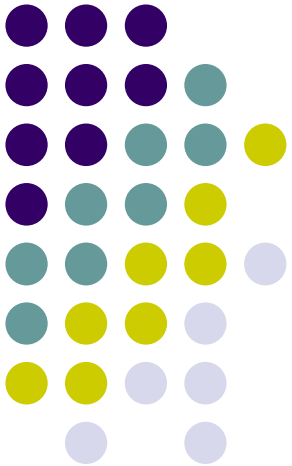
wykład

Zarządzanie współbieżnym wykonywaniem transakcji Cz.2



Plan i cel wykładu

- Celem wykładu jest przedstawienie i omówienie podstawowych algorytmów zarządzania współbieżnym wykonywaniem transakcji.
- Rozpoczniemy od przedstawienia algorytmów blokowania.
  - algorytm blokowania dwu-fazowego.
- Następnie przedstawimy zjawisko
  - zakleszczenia i omówimy podstawowe algorytmy rozwiązywania zakleszczenia.
- Na zakończenie wykładu, przedstawimy i omówimy problem duchów.

klasyfikacja algorytmów

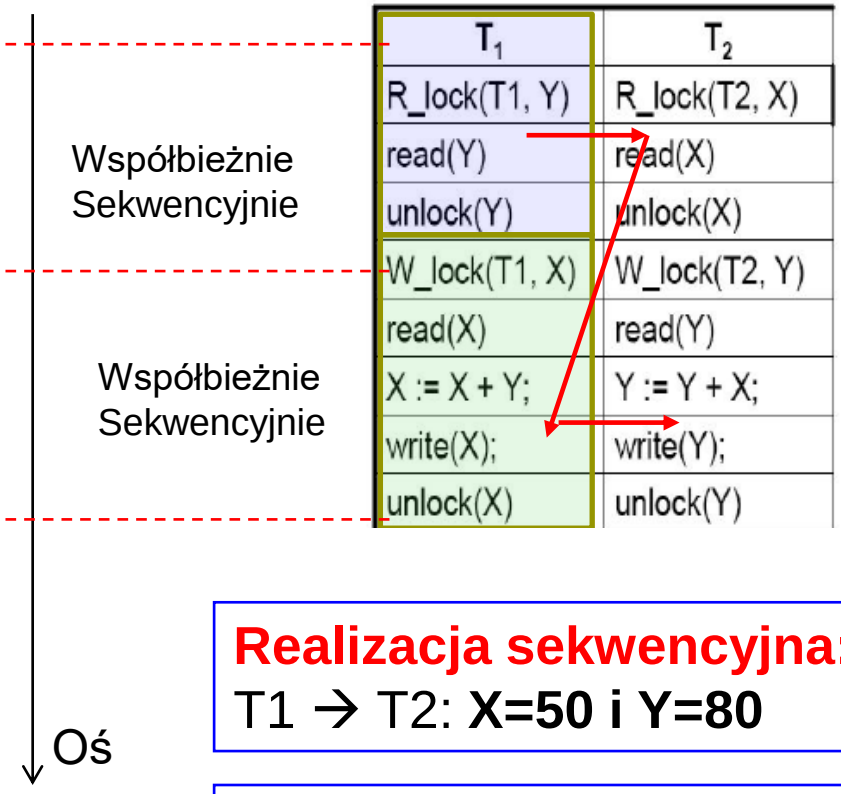
Algorytmy zarządzania współbieżnym wykonywaniem transakcji możemy sklasyfikować następująco:

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

algorytm blokowania



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

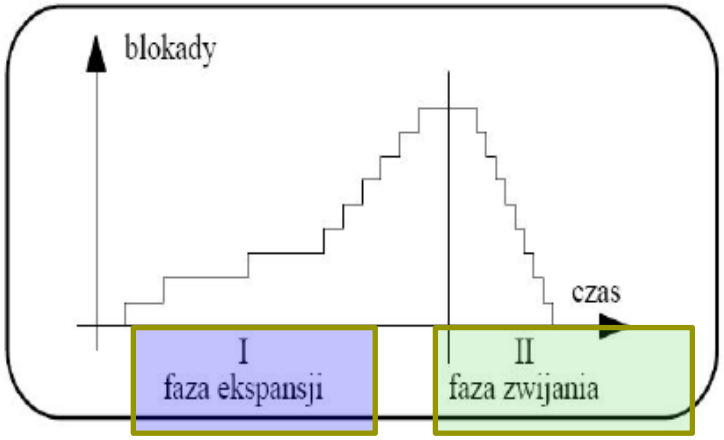
Realizacja sekwencyjna:
T2 → T1: **X=70 i Y=50**

2

Realizacja współbieżna:
X=50 i Y=50

3

algorytm blokowania dwufazowego

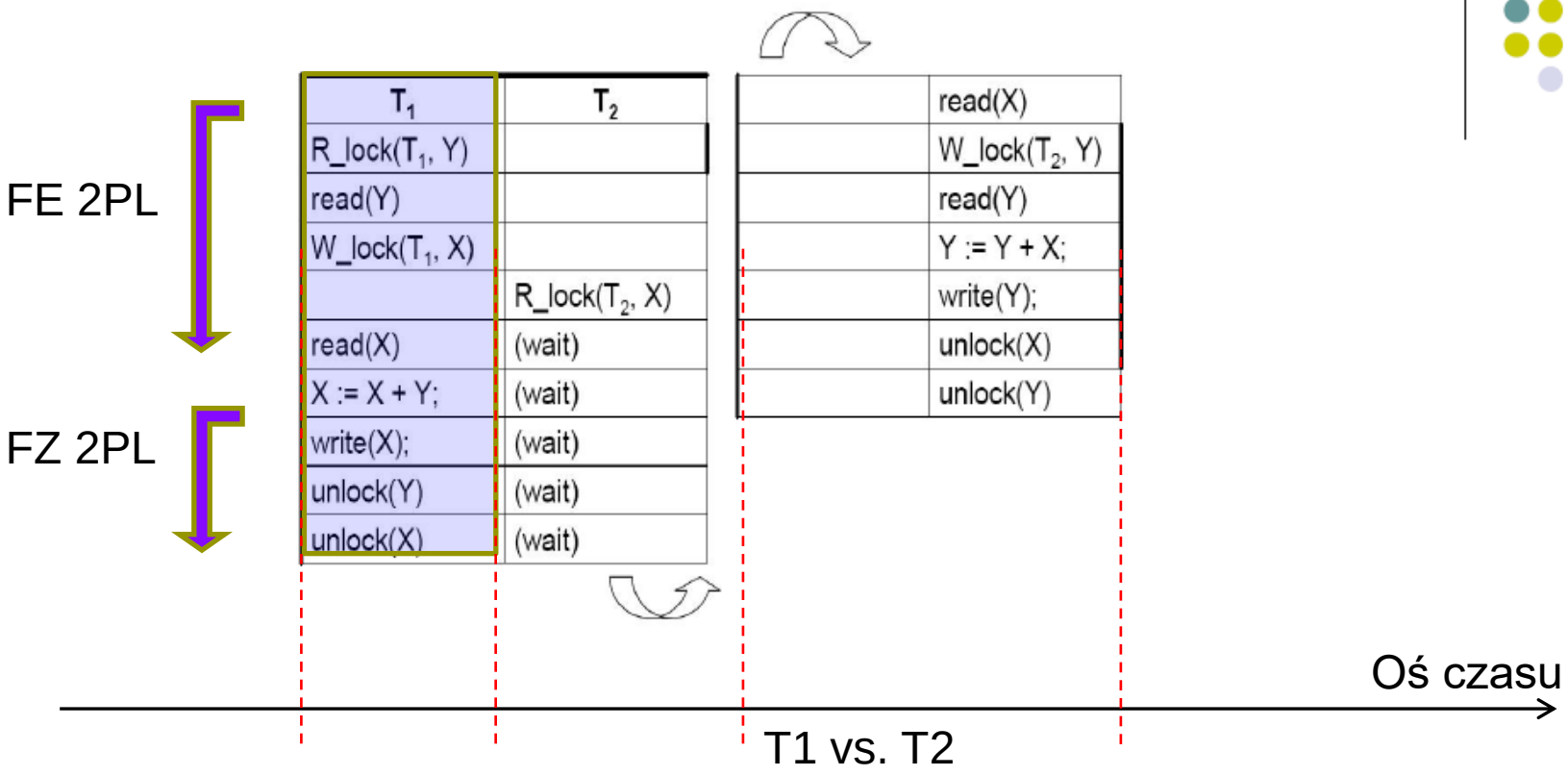


Algorytm podstawowy:

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*

Realizacja transakcji, zgodnie z algorytmem **2PL**, przebiega w dwóch fazach (stąd nazwa algorytmu): **w fazie ekspansji oraz w fazie zwijania**.

algorytm blokowania dwufazowego

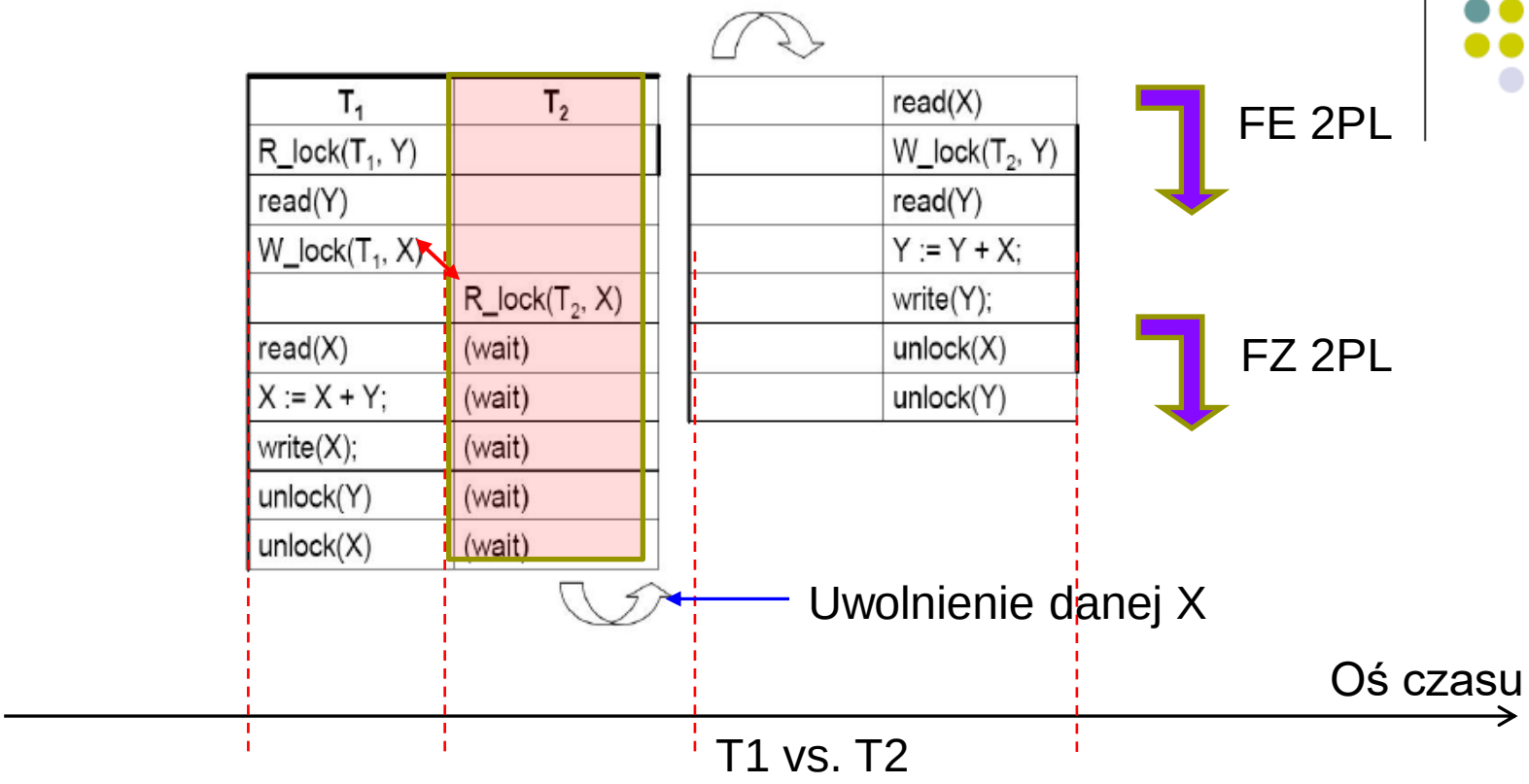


T_1	T_2
R_lock(T_1, Y)	
read(Y)	
W_lock(T_1, X)	
	R_lock(T_2, X)
read(X)	(wait)
X := X + Y;	(wait)
write(X);	(wait)
unlock(Y)	(wait)
unlock(X)	(wait)

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

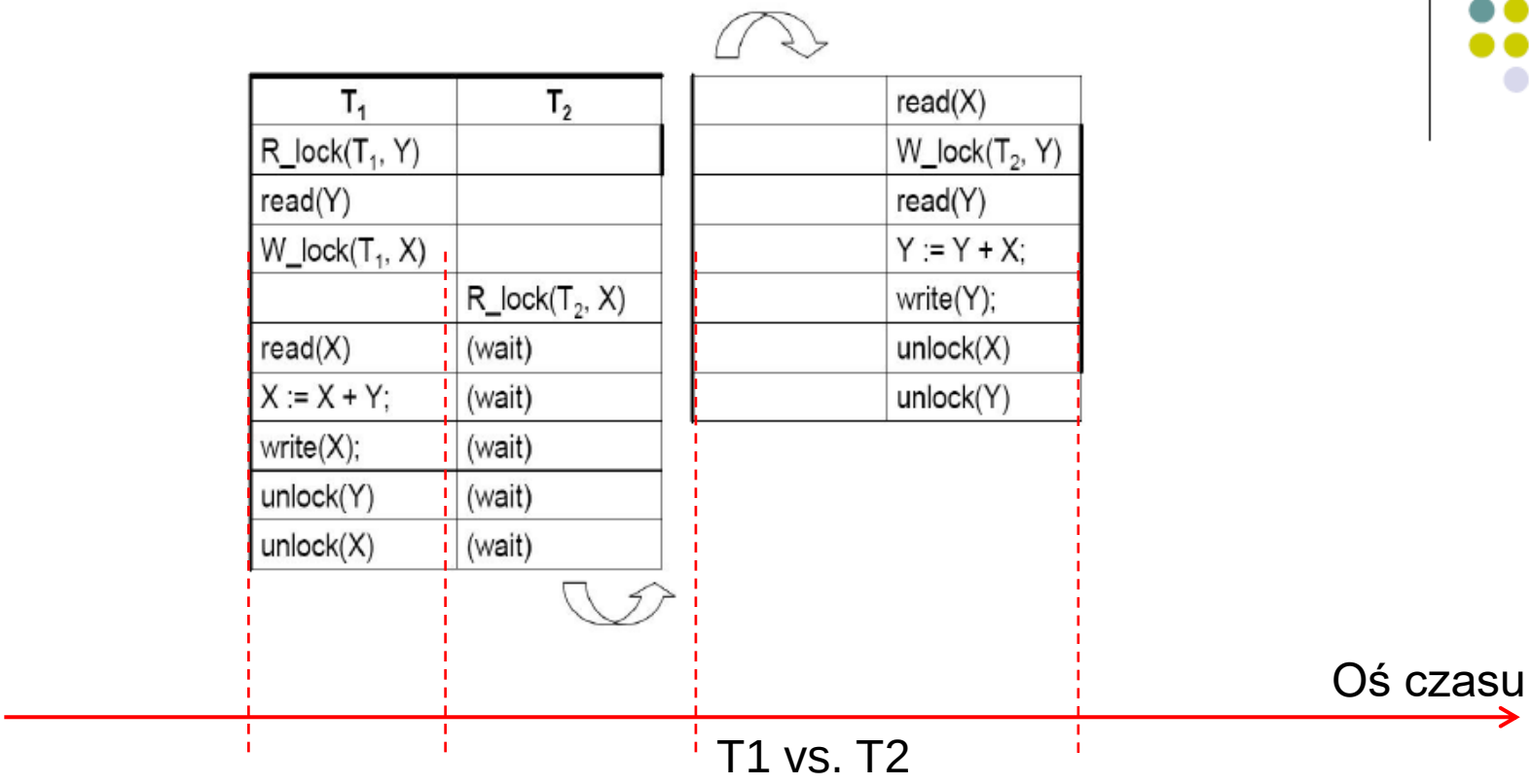
Działanie algorytmu blokowania dwufazowego: przykładowa realizacja transakcji T1 i T2

algorytm blokowania dwufazowego



Działanie algorytmu blokowania dwufazowego: przykładowa realizacja transakcji T1 i T2

algorytm blokowania dwufazowego (3)



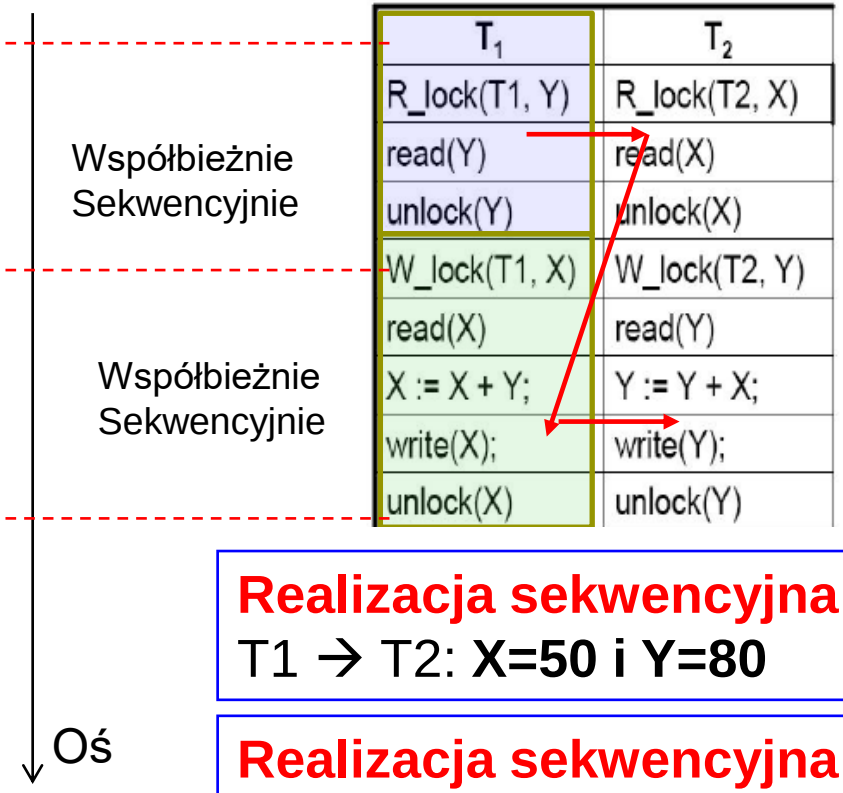
T_1	T_2
$R\_lock(T_1, Y)$	
read(Y)	
$W\_lock(T_1, X)$	
	$R\_lock(T_2, X)$
read(X)	(wait)
X := X + Y;	(wait)
write(X);	(wait)
unlock(Y)	(wait)
unlock(X)	(wait)

	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

algorytm blokowania



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

Realizacja współbieżna:
X=50 i Y=50

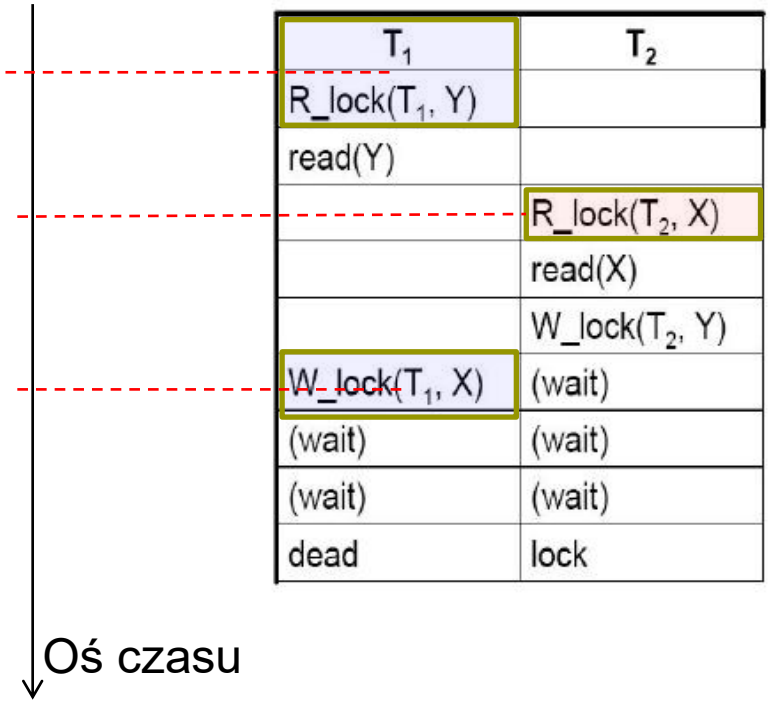
3

Realizacja 2PL:
X=50 i Y=80 (T1 → T2)

4

zakleszczenie transakcji

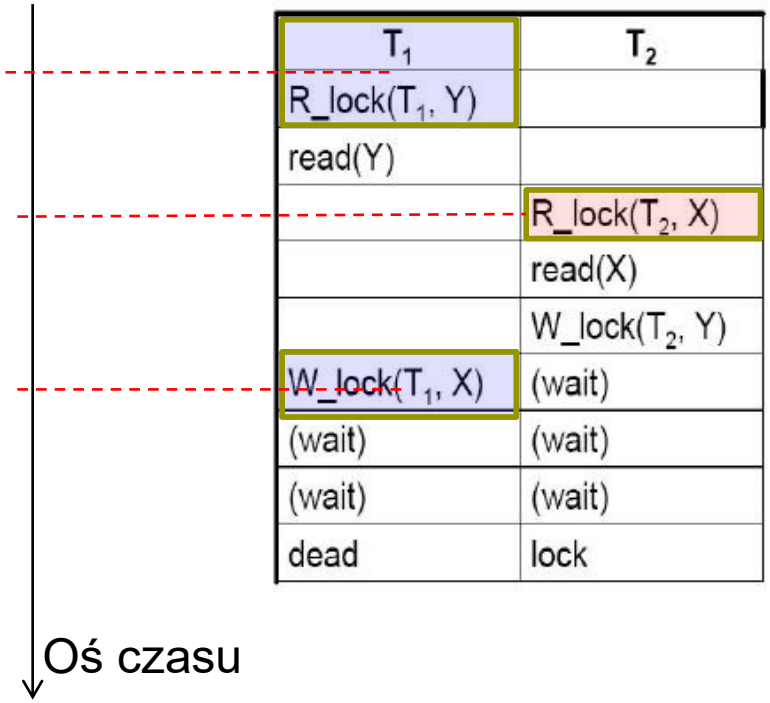
Realizacja współbieżna
T1 i T2



Przykład współbieżnej realizacji transakcji T1 i T2, zgodnie z algorytmem 2PL, przedstawiony na slajdzie

Realizacja obu transakcji ulega zawieszeniu

zakleszczenie transakcji



T_1	T_2
R_lock(T_1, Y)	
read(Y)	
	R_lock(T_2, X)
	read(X)
	W_lock(T_2, Y)
W_lock(T_1, X)	(wait)
(wait)	(wait)
(wait)	(wait)
dead	lock

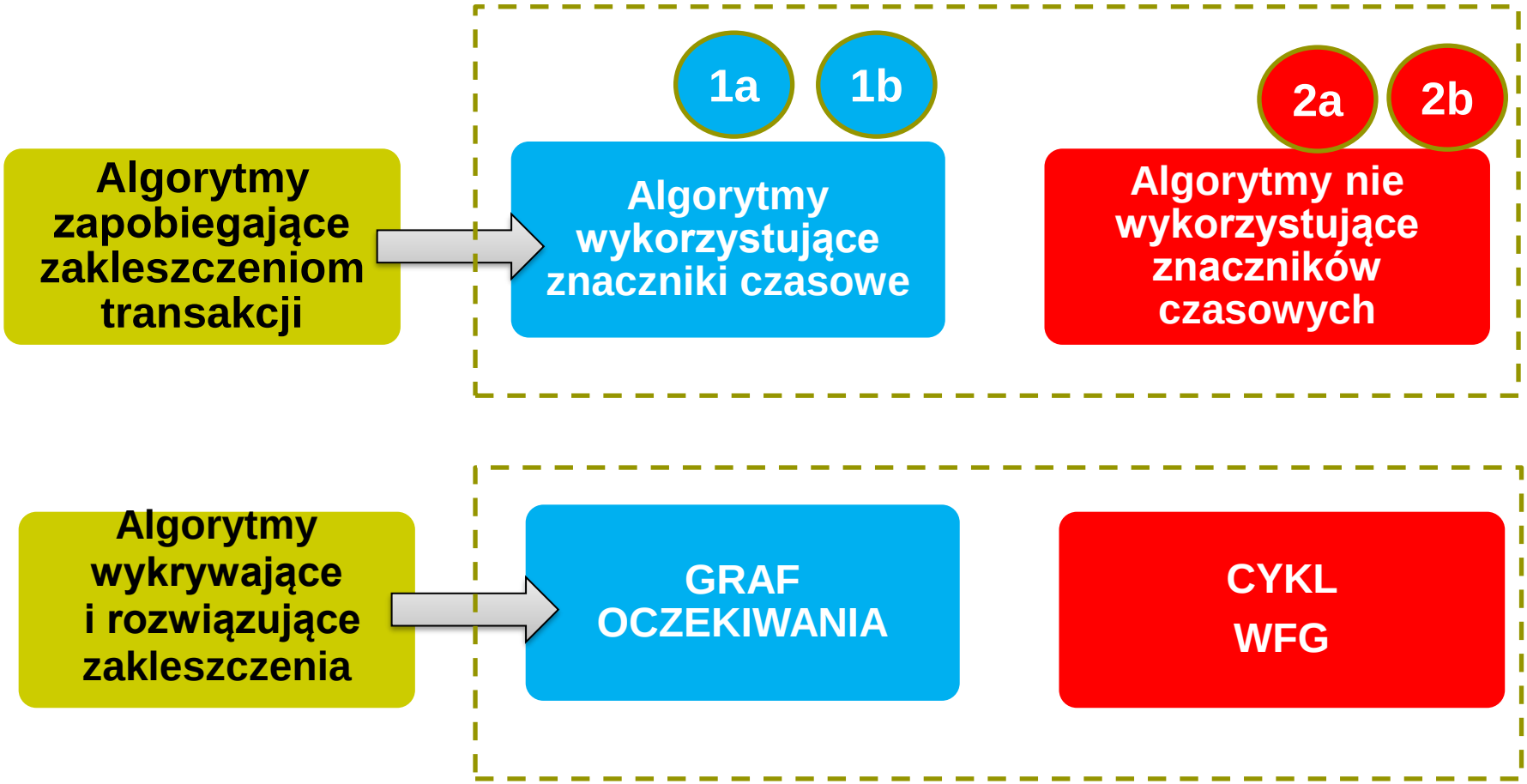
Dwa podejścia do problemu zakleszczenia transakcji:

- **Wykrywanie i rozwiązywanie** zakleszczenia
- **Zapobieganie wystąpieniu** zakleszczenia

Przykład współbieżnej realizacji transakcji T1 i T2, zgodnie z algorytmem 2PL, przedstawiony na slajdzie

Stan wzajemnego oczekiwania transakcji na uwolnienie zasobów nazywamy zakleszczeniem transakcji.

Algorytmy vs. zakleszczenia transakcji



zapobieganie zakleszczeniom transakcji (1)

- Algorytmy wykorzystujące *znaczniki czasowe* transakcji - *TS(T)*, nadawane w momencie inicjacji transakcji:

- **wait-die**: Transakcja T_i próbuje uzyskać blokadę na danej X, tymczasem dana ta jest już zablokowana przez transakcję T_j. Jeżeli $TS(T_i) < TS(T_j)$ (T_i jest starsza T_j) wtedy transakcja T_i będzie czekać na zwolnienie blokady. W przeciwnym wypadku T_i będzie wycofana i restartowana z tym samym znacznikiem czasowym

1a

- **wound-wait**: Transakcja T_i próbuje uzyskać blokadę na danej X, tymczasem dana ta jest już zablokowana przez transakcję T_j. Jeżeli $TS(T_i) < TS(T_j)$ (T_i jest starsza T_j) wtedy transakcja T_j będzie wycofana i restartowana z tym samym znacznikiem czasowym. W przeciwnym wypadku T_i będzie czekać na zwolnienie blokady

1b

zapobieganie zakleszczeniom transakcji (1)

- Algorytmy wykorzystujące *znaczniki czasowe* transakcji - *TS(T)*, nadawane w momencie inicjacji transakcji:

- **wait-die**: Transakcja T_i próbuje uzyskać blokadę na danej X, tymczasem dana ta jest już zablokowana przez transakcję T_j. Jeżeli $TS(T_i) < TS(T_j)$ (T_i jest starsza T_j) wtedy transakcja T_i będzie czekać na zwolnienie blokady. W przeciwnym wypadku T_i będzie wycofana i restartowana z tym samym znacznikiem czasowym

1a

- **wound-wait**: Transakcja T_i próbuje uzyskać blokadę na danej X, tymczasem dana ta jest już zablokowana przez transakcję T_j. Jeżeli $TS(T_i) < TS(T_j)$ (T_i jest starsza T_j) wtedy transakcja T_j będzie wycofana i restartowana z tym samym znacznikiem czasowym. W przeciwnym wypadku T_i będzie czekać na zwolnienie blokady

1b

zapobieganie zakleszczeniom transakcji (2)

- Algorytmy nie korzystające ze znaczników czasowych.
- **no waiting**: Transakcja T_i próbuje uzyskać blokadę na danej X, tymczasem dana ta jest już zablokowana przez transakcję T_j. Transakcja T_i będzie wycofana i restartowana z pewnym opóźnieniem czasowym.

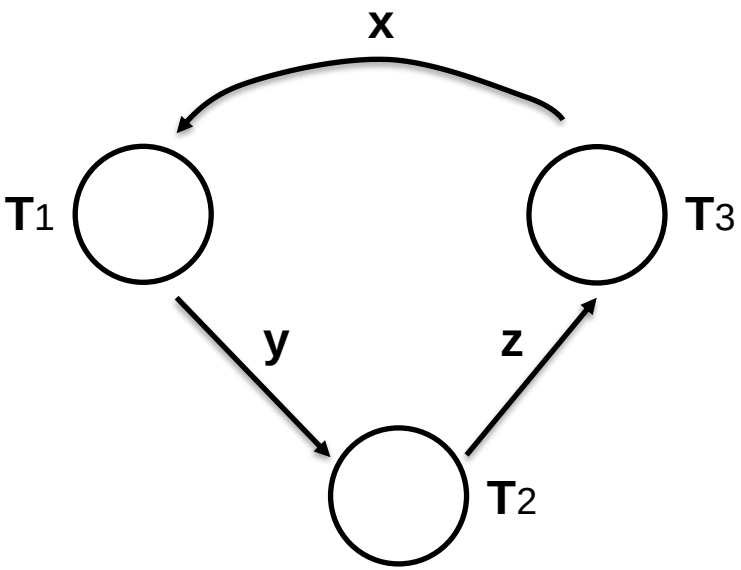
2a

- **cautious waiting**: Transakcja T_i próbuje uzyskać blokadę na danej X, tymczasem dana ta jest już zablokowana przez transakcję T_j. Jeżeli transakcja T_j nie czeka na uzyskanie innej blokady, T_i będzie czekać na zwolnienie blokady przez T_j. W przeciwnym wypadku T_i będzie wycofana i restartowana

2b

metody wykrywania i rozwiązywania zakleszczeń

Graf oczekiwania *(waits-for graph – WFG))*

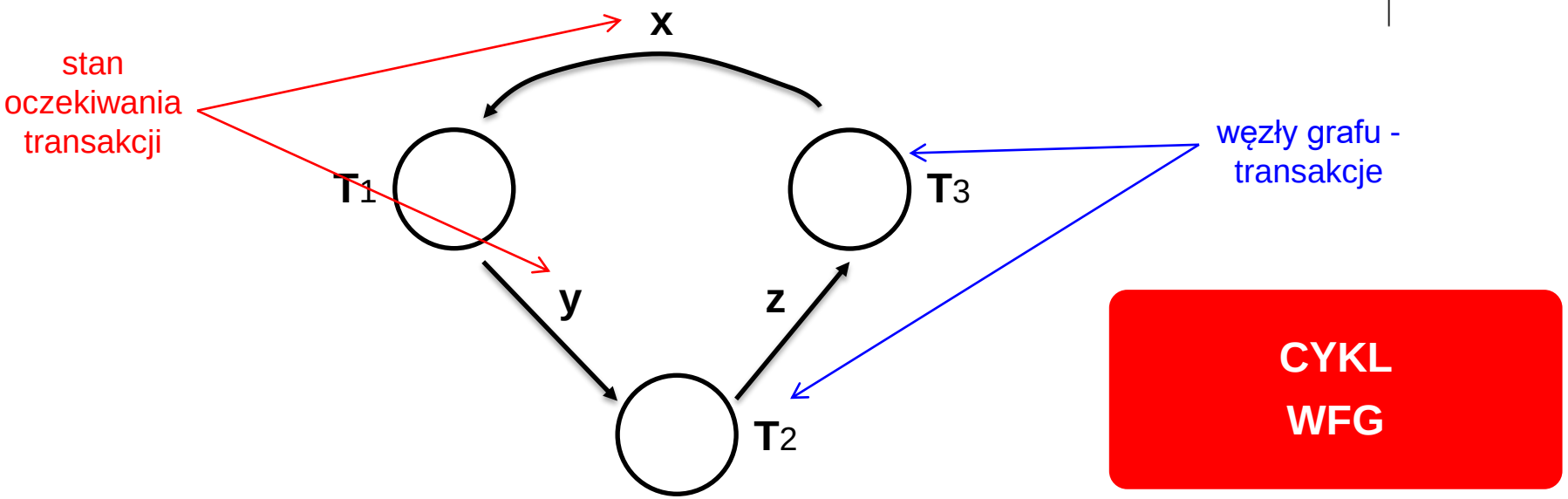


GRAF OCZEKIWANIA

Zakleszczenie jest zjawiskiem dość rzadkim i najczęściej obejmuje niewiele transakcji. Stąd taniej jest wykrywać zakleszczenia a następnie rozwiązywać je w momencie ich wystąpienia.

metody wykrywania i rozwiązywania zakleszczeń

Graf oczekiwania *(waits-for graph – WFG))*



- transakcja **T1** oczekuje na zwolnienie danej Y blokowanej przez transakcję **T2**.
- transakcja **T2** oczekuje na uwolnienie danej Z blokowanej przez transakcję **T3**.
- transakcja **T3**, z kolei, oczekuje na zwolnienie danej X - blokowanej przez transakcję **T1**.

Cykl w grafie WFG oznacza wystąpienie zakleszczenia w systemie

procedura wykrywania zakleszczenia (1)

Do budowy grafu WFG wykorzystuje się struktury opisujące blokady.
Z każdą blokadą związane są dwie listy:

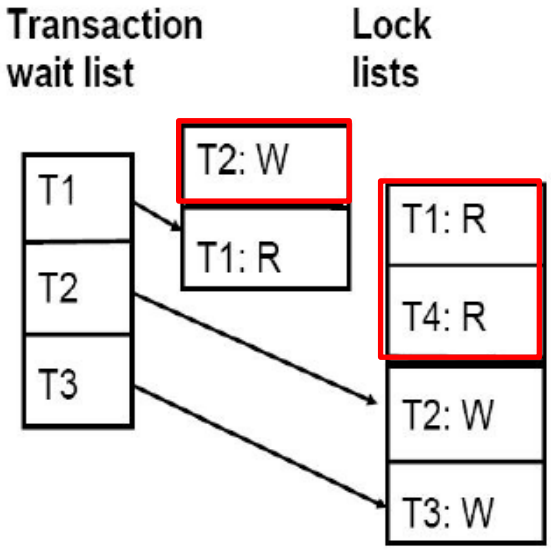
- lista transakcji, które uzyskały blokadę
- lista transakcji oczekujących na przydział blokady.

Obie listy mają postać ((Ti, mi),...), gdzie Ti – oznacza transakcję, natomiast mi – oznacza rodzaj blokady.

Do grafu WFG dodaje się łuk Ti → Tj, jeżeli zachodzi warunek:

- transakcja Tj należy do listy transakcji, które uzyskały blokadę, natomiast Ti – jest na liście transakcji oczekujących, lub
- transakcja Tj jest przed transakcją Ti na liście transakcji oczekujących
- blokady „mi” oraz „mj” są niekompatybilne.

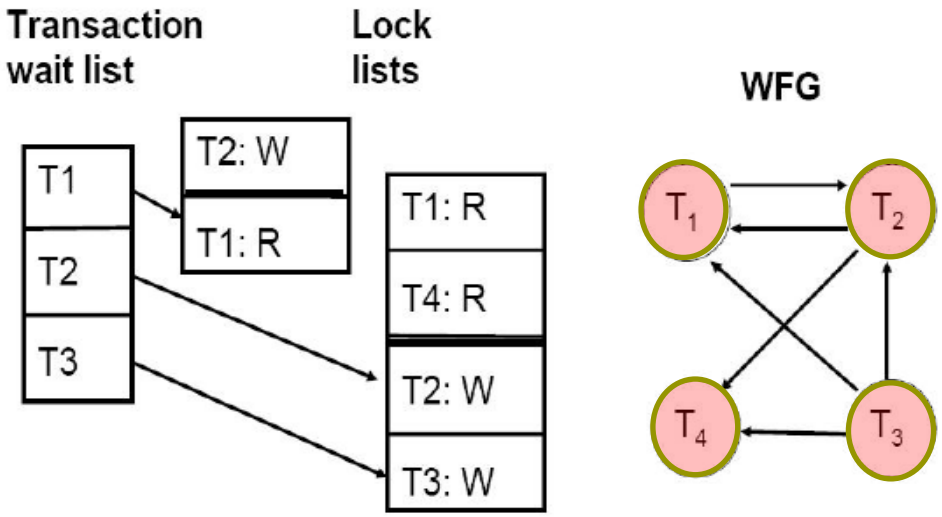
procedura wykrywania zakleszczenia (2)



Działanie procedury wykrywania zakleszczenia

Pierwsza z danych jest blokowana przez transakcję **T2** : T2:W
Druga z danych jest blokowana przez transakcje T1 i **T4:** T1:R T4:R

procedura wykrywania zakleszczenia (2)



- Łuk (T1,T2) reprezentuje oczekiwanie transakcji T1 na T2 w odniesieniu do pierwszego zasobu.
- Łuk (T2,T1) reprezentuje oczekiwanie transakcji T2 na T1 w odniesieniu do drugiego zasobu.
- Łuk (T2,T4) reprezentuje oczekiwanie transakcji T2 na T4 w odniesieniu do drugiego zasobu.
- Podobnie, łuki (T3,T4) oraz (T3,T1).
- Łuk (T3,T2) reprezentuje sytuację, gdy obie transakcje T3 i T2 znajdują się kolejce transakcji oczekujących na uwolnienie zasobu, ale ich blokady są niekompatybilne.

W przykładowym grafie WFG występuje cykl obejmujący wierzchołki T1 i T2.

problem „duchów” (1)

```
T1: select * from emp
where eyes =”blue” and hair=”red”;
T2: insert into emp
where eyes =”blue” and hair=”red”;
```

T1 → T2

Problem duchów: Jest konsekwencją przyjęcia określonej jednostki blokowania.

- Załóżmy, że transakcje T1 i T2 są wykonywane sekwencyjnie.
- Blokowanie realizowane na poziomie rekordów bazy danych: ⇒ niebezpieczeństwo nieuszeregowalności realizacji.

problem „duchów” (1)

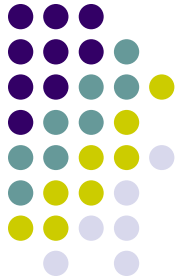
```
T1: select * from emp
where eyes =”blue” and hair=”red”;
T2: insert into emp
where eyes =”blue” and hair=”red”;
```

T2 → T1

W odniesieniu do innej relacji, kolejność operacji transakcji T1 i T2 może być odwrotna.

Łatwo zauważyć, że w grafie uszeregowalności wystąpi cykl świadczący o nieuszeregowalności realizacji.

problem duchów (2)

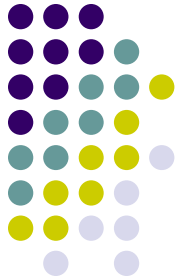


- **Łatwo zauważyć, że współbieżne wykonanie transakcji T1 i T2 może być nieuszeregowalne.**
- **Nie istnieje żaden mechanizm blokowania, realizowany na poziomie blokad rekordów, który zagwarantowałby rozwiązanie problemu „duchów”.**

- **Takie nowo-wprowadzane lub usuwane rekordy nazywane są „duchami”.**

T1: select * from ***emp***
where eyes =”blue” and hair=”red”;
T2: insert into ***emp***
where eyes =”blue” and hair=”red”;

problem duchów (3)

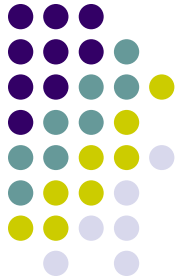


PYTANIE

- **W jaki sposób zapobiec, aby transakcja T2 nie wprowadzała nowych rekordów do relacji *emp* w trakcie realizacji transakcji T1?**

T1: select * from ***emp***
where eyes =”blue” and hair=”red”;
T2: insert into ***emp***
where eyes =”blue” and hair=”red”;

problem duchów (4)

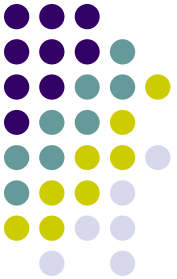


ODPOWIEDŹ

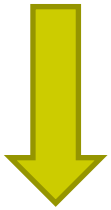
- **Rozwiązanie problemu „duchów” wymaga wprowadzenia blokad hierarchicznych !**

T1: select * from ***emp***
where eyes =”blue” and hair=”red”;
T2: insert into ***emp***
where eyes =”blue” and hair=”red”;

blokady hierarchiczne – ziarnistość blokad



- hierarchiczny algorytm blokowania dwufazowego → realizacja blokad o różnej ZIARNISTOŚCI

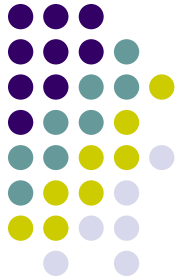


Łączy mechanizm blokad fizycznych z blokadami intencyjnymi /predykatowymi/

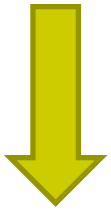
Wykorzystuje immanentną cechę organizacji bazy danych jaką jest hierarchia ziarnistości danych

- hierarchiczna struktura BD pozwala na efektywną implementację hierarchicznego algorytmu blokowania

Problem „duchów” - wybór jednostki blokowania (1)



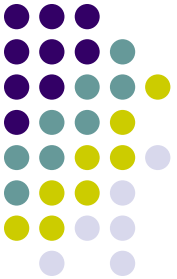
- Jest konsekwencją przyjęcia określonej jednostki blokowania → efektywność działania systemu



Efektywność systemu: liczba transakcji na sekundę

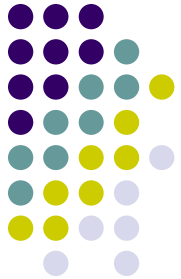
- Wybór jednostki blokowania jest kompromisem między stopniem współbieżności systemu, a narzutem systemowym związanym z implementacją algorytmu blokowania

Problem „duchów” - wybór jednostki blokowania (2)



- przepustowość systemu rośnie wraz ze zwiększaniem precyzji blokowania (zmniejszaniem liczby zablokowanych danych i zwiększaniem liczby blokad)
- precyzyjne blokowanie jest kosztowne dla złożonych transakcji, które wymagają długiego czasu utrzymywania dużej liczby blokad
- blokowanie dużych jednostek danych wspiera złożone transakcje o dużej liczbie operacji, kosztem prostych transakcji o niewielkiej liczbie operacji.

Potrzebny jest protokół, który będzie wspierał oba typy transakcji, tzn.: taki, który będzie umożliwiał równoczesne zakładanie blokad na różnych jednostkach danych.



KONIEC WYKŁADU
Cz. 2