

WYKŁAD

Normalizacja schematów logicznych relacji

Plan wykładu

- Motywacja
- Normalizacja
- Postacie normalne
- Dekompozycje



Motywacja (1)

- Dana jest następująca relacja Dostawcy :

Unikalny
atrybut



Nazwisko	Adres	Produkt	Cena
Kowalski	ul. Krucza 10	chipsy	1,50
Kowalski	ul. Krucza 10	orzeszki	3,50
...
Kowalski	ul. Krucza 10	gruszki	4,50
Nowak	ul. Malwowa 4	chipsy	2,00
Nowak	ul. Malwowa 4	orzeszki	4,00
...

Motywacja (2)

- Założmy, że atrybut Nazwisko jest unikalny, tj. nie ma dwóch dostawców o tym samym nazwisku.
- Cechy relacji Dostawca:
 - redundancja danych - problem spójności danych
 - anomalia wprowadzania danych
 - anomalia usuwania danych
 - anomalia uaktualniania danych
- Rozwiązaniem: dekompozycja relacji Dostawca na dwie relacje: Dostawca i Dostawy

Nazwisko	Adres	Produkt	Cena
Kowalski	ul. Krucza 10	chipsy	1,50
Kowalski	ul. Krucza 10	orzeszki	3,50
...
Kowalski	ul. Krucza 10	gruszki	4,50
Nowak	ul. Malwowa 4	chipsy	2,00
Nowak	ul. Malwowa 4	orzeszki	4,00
...

Motywacja (3)

- Dekompozycja bez utraty informacji

Dostawca

Nazwisko	Adres
Kowalski	ul. Krucza 10
...	...
...	...
...	...
Nowak	ul. Malwowa 4
...	...

Dostawy

Nazwisko	Produkt	Cena
Kowalski	chipsy	1,50
Kowalski	orzeszki	3,50
Kowalski	gruszki	4,50
Nowak	chipsy	2,00
Nowak	orzeszki	4,00
...



Atrybut połączeniowy : NAZWISKO

Struktury danych (1)

1. Baza danych jest zbiorem relacji
2. Schemat relacji R, oznaczony przez $R(A_1, A_2, \dots, A_n)$, składa się z nazwy relacji R oraz listy atrybutów A_1, A_2, \dots, A_n
3. Liczbę atrybutów składających się na schemat relacji R nazywamy stopniem relacji
4. Każdy atrybut A_i schematu relacji R posiada domenę, oznaczoną jako $\text{dom}(A_i)$
5. Domena definiuje zbiór wartości atrybut relacji poprzez podanie typu danych
6. Relacją r o schemacie $R(A_1, A_2, \dots, A_n)$, oznaczoną $r(R)$, nazywamy zbiór n-tek (krotek) postaci $r=\{t_1, t_2, \dots, t_m\}$.
7. Pojedyncza krotka t jest uporządkowaną listą n wartości $t=\langle v_1, v_2, \dots, v_n \rangle$, gdzie $v_i, 1 < i < n$, jest elementem $\text{dom}(A_i)$ lub specjalną wartością pustą (NULL)
8. i-ta wartość krotki t, odpowiadająca wartości atrybutu A_i , będzie oznaczana przez $t[A_i]$
9. Relacja $r(R)$ jest relacją matematyczną stopnia n zdefiniowaną na zbiorze domen $\text{dom}(A_1), \text{dom}(A_2), \dots, \text{dom}(A_n)$ będącą podzbiorem iloczynu kartezjańskiego domen definiujących R: $r(R) \subseteq \text{dom}(A_1) \times \text{dom}(A_2) \times \dots \times \text{dom}(A_n)$

Zależności funkcyjne (1)

- **Zależność funkcyjna (FD)**

Dana jest relacja r o schemacie R . X, Y są podzbiorami atrybutów R . W schemacie relacji R , X wyznacza funkcyjnie Y , lub Y jest funkcyjnie zależny od X , co zapisujemy $X \rightarrow Y$, wtedy i tylko wtedy, jeżeli dla dwóch dowolnych krotek t_1, t_2 takich, że $t_1[X] = t_2[X]$ zachodzi zawsze $t_1[Y] = t_2[Y]$, gdzie $t_i[A]$ oznacza wartość atrybutu A krotki t_i

- **Przykłady:**

- 1. Nazwisko \rightarrow Adres
- 2. {Nazwisko, Towar} \rightarrow Cena

Nazwisko	Adres	Produkt	Cena
Kowalski	ul. Krucza 10	chipsy	1,50
Kowalski	ul. Krucza 10	orzeszki	3,50
...
Kowalski	ul. Krucza 10	gruszki	4,50
Nowak	ul. Malwowa 4	chipsy	2,00
Nowak	ul. Malwowa 4	orzeszki	4,00
...

Zależności funkcyjne (2)

- Zależność funkcyjna określa zależność pomiędzy atrybutami. Jest to własność semantyczna, która musi być spełniona dla dowolnych wartości krotek relacji.
- Relacje które spełniają nałożone zależności funkcyjne nazywamy instancjami legalnymi
- Zależność funkcyjna jest własnością schematu relacji R , a nie konkretnego wystąpienia relacji
- Z zależności funkcyjnej wynika, że jeżeli $t1[X] = t2[X]$ i $X \rightarrow Y$, to zachodzi zawsze $t1[Y] = t2[Y]$



Normalizacja

- Proces normalizacji relacji można traktować jako proces, podczas którego schematy relacji posiadające pewne niepożądane cechy są dekomponowane na mniejsze schematy relacji o pożądanych własnościach
- Proces normalizacji musi posiadać trzy dodatkowe własności:

Własność zachowania atrybutów - żaden atrybut nie zostanie zagubiony w trakcie procesu normalizacji

Własność zachowania informacji - dekompozycja relacji nie prowadzi do utraty informacji

Własność zachowania zależności - wszystkie zależności funkcyjne są reprezentowane w pojedynczych schematach relacji



Pojęcia podstawowe (1)

- Nadkluczem (superkluczem) schematu relacji $R = \{A_1, A_2, \dots, A_n\}$ nazywamy zbiór atrybutów $S \subseteq R$, który jednoznacznie identyfikuje wszystkie krotki relacji r o schemacie R . Innymi słowy, w żadnej relacji r o schemacie R nie istnieją dwie krotki t_1, t_2 takie, że $t_1[S] = t_2[S]$
- Kluczem K schematu relacji R nazywamy minimalny nadklucz, to znaczy taki nadklucz, że nie istnieje $K' \subset K$ będący nadkluczem schematu R
- Kluczem schematu Dostawy {Nazwisko, Produkt, Cena}

Dostawy

Nazwisko	Produkt	Cena
Kowalski	chipsy	1,50
Kowalski	orzeszki	3,50
Kowalski	gruszki	4,50
Nowak	chipsy	2,00
Nowak	orzeszki	4,00
...

**Klucz podstawowego schematu
relacji DOSTAWCA**

Pojęcia podstawowe (2)

- Klucze potencjalne (ang. *candidate keys*)
 - Klucz podstawowy (primary key)
 - Klucz drugorzędny (secondary key)
- Atrybuty:
 - atrybuty podstawowe: atrybut X jest podstawowy w schemacie R jeżeli należy do któregośkolwiek z kluczy schematu R
 - atrybuty wtórne: atrybut X jest wtórny w schemacie R jeżeli nie należy do żadnego z kluczy schematu R



normalizacja (1)

Zapewnienie, że każda informacja jest reprezentowana w modelu encji tylko raz

Relacja jest w pierwszej postaci normalnej (1PN 1NF) wtedy i tylko wtedy, gdy nie ma powtarzających się grup i każdy atrybut jest w postaci atomowej.

Relacja jest w drugiej postaci normalnej (2PN 2NF) wtedy i tylko wtedy, gdy: jest w pierwszej postaci normalnej i każdy niekluczowy atrybut jest zależny od wszystkich części klucza głównego.

Relacja jest w trzeciej postaci normalnej (3PN 3NF) wtedy i tylko wtedy, gdy: jest w drugiej postaci normalnej i żaden atrybut, nie będący kluczem, nie jest funkcjonalnie związany żadnym innym atrybutem, nie będącym również kluczem.



normalizacja – receptura (2)

1PN

Każdy atrybut musi mieć jedną wartość dla każdego wystąpienia jego encji w danym momencie czasu

Przejście z postaci 0PN do 1PN:

- usunięcie wielowartościowych atrybutów z encji w 0PN i utworzenie dla niej nowej encji
- skopiowanie unikalnego identyfikatora do nowej encji - będzie on prawdopodobnie stanowił część identyfikatora unikalnego dla tej nowej encji

2PN

Wartość każdego atrybutu musi zależeć od całego identyfikatora jego encji.

Przejście z postaci 1PN do 2PN:

- usunięcie wszystkich częściowo zależnych atrybutów i utworzenie dla nich nowej encji
- skopiowanie części identyfikatora z encji pierwotnej (od której zależne są usunięte atrybuty) do tej nowej encji

3PN

Wartość każdego atrybutu nie może zależeć od niczego innego poza identyfikatorem unikalnym

Przejście z postaci 2PN do 3PN:

- należy usunąć atrybuty niezależne i wtawić je do nowej encji

Uwaga: ta nowa encja potrzebuje identyfikatora unikalnego

pierwsza postać normalna 1NF (1)

Definicja

Schemat relacji R znajduje się w pierwszej postaci normalnej(1NF), jeżeli wartości atrybutów są atomowe (niepodzielne)

Tablica Pleć: 0 NF

Płeć	Imię
Męska	Jan, Piotr, Zenon
Żeńska	Anna, Eliza, Maria

Atrybut typu zbiorowego



Relacja Pleć w 1NF

Płeć	Imię
Męska	Jan
Męska	Piotr
Męska	Zenon
Żeńska	Anna
Żeńska	Eliza
Żeńska	Maria



pierwsza postać normalna 1NF (3)

- Pierwsza postać normalna zabrania definiowania złożonych atrybutów, które są wielowartościowe
- Relacje, które dopuszczają definiowanie złożonych atrybutów nazywamy relacjami zagnieżdzonymi (ang. *nested relations*)
- W relacjach zagnieżdzonych każda krotka może zawierać inną relację
 - *Pracownicy (idPrac, Nazwisko, {Projekty (nr, godziny)})*

Pracownicy

IdPrac	Nazwisko	Projekty	
		nr	godziny
1234567	Kowalski	1	32,5
		2	7,5
6655443	Nowak	3	40,5
		1	20
4343435	Kruczak	2	20
		1	10
3333333	Morzy	2	10
		3	10
		4	10

Relacja zewnętrzna

Relacja
zagnieżdzona



pierwsza postać normalna 1NF (4)

- Dana jest relacja R, zawierająca inną relację P
- Dekompozycja relacji R do zbioru relacji w 1NF:
 - Utwórz osobną relację dla relacji zewnętrznej
 - Utwórz osobną relację dla relacji wewnętrznej (zagnieżdżonej), do której dodaj klucz relacji zewnętrznej
 - Kluczem nowej relacji wewnętrznej (klucz relacji wewnętrznej + klucz relacji zewnętrznej)
- Dekompozycja relacji Pracownicy:
 - Pracownicy (IdPrac, Nazwisko)
 - Uczestnicy (IdPrac, Nr, Godziny)



druga postać normalna 2NF (1)

- Pełna zależność funkcyjna

Zbiór atrybutów Y jest w pełni funkcyjnie zależny od zbioru atrybutów X w schemacie R , jeżeli $X \rightarrow Y$ i nie istnieje podzbiór $X' \subset X$ taki, że $X' \rightarrow Y$

Zbiór atrybutów Y jest częściowo funkcyjnie zależny od zbioru atrybutów X w schemacie R , jeżeli $X \rightarrow Y$ i istnieje podzbiór $X' \subset X$ taki, że $X' \rightarrow Y$

- Druga postać normalna

Dana relacja r o schemacie R jest w drugiej postaci normalnej (2NF), jeżeli żaden atrybut wtórny tej relacji nie jest częściowo funkcyjnie zależny od żadnego z kluczy relacji r



druga postać normalna 2NF (2)

- Uczestnictwo

klucz	→	<table border="1"><tr><td>IdPrac</td><td>NrProj</td><td>Funkcja</td><td>Nazwisko</td><td>NazwaProj</td><td>Lokalizacja</td></tr></table>	IdPrac	NrProj	Funkcja	Nazwisko	NazwaProj	Lokalizacja
IdPrac	NrProj	Funkcja	Nazwisko	NazwaProj	Lokalizacja			

Zależności
od klucza

fd1: {IdPrac, NrProj} → Funkcja
fd2: {IdPrac, NrProj} → Nazwisko
fd3: {IdPrac, NrProj} → NazwaProj
fd4: {IdPrac, NrProj} → Lokalizacja

Atrybut
podstawowy

Zależności
niepełne

fd5: {IdPrac} → Nazwisko
fd6: {NrProj} → NazwaProj
fd7: {NrProj} → Lokalizacja

Atrybuty
wtórne

Zależności fd2, fd3, fd4 są zależnościami niepełnymi

Definicja 2NF !

druga postać normalna 2NF (3)

Uczestnictwo'

IdPrac	NrProj	Funkcja
--------	--------	---------

$fd1: \{IdPrac, NrProj\} \rightarrow Funkcja$

Pracownicy

IdPrac	Nazwisko
--------	----------

$fd5: \{IdPrac\} \rightarrow Nazwisko$

Projekty

NrProj	NazwaProj	Lokalizacja
--------	-----------	-------------

$fd6: \{NrProj\} \rightarrow NazwaProj$

$fd7: \{NrProj\} \rightarrow Lokalizacja$

$\{fd1, fd2, fd3, fd4, fd5, fd6, fd7\}^+ \equiv \{fd1, fd5, fd6, fd7\}^+$
bo:

$fd1 \Rightarrow fd2, fd3, fd4$, zgodnie z regułą poszerzenia



trzecia postać normalna 3NF (1)

Pracownicy-PP

klucz



Nazwisko	Instytut	Wydział
Brzeziński	I.Informatyki	Elektryczny
Morzy	I.Informatyki	Elektryczny
Koszlajda	I.Informatyki	Elektryczny
Królikowski	I.Informatyki	Elektryczny
...
Babij	ElektroEnerg.	Elektryczny
Kordus	ElektroEnerg.	Elektryczny
Sroczan	ElektroEnerg.	Elektryczny

Klucz: Nazwisko

Zależności funkcyjne: *Nazwisko → Instytut*

Nazwisko → Wydział

Instytut → Wydział



trzecia postać normalna 3NF (2)

- **Przechodnia zależność funkcyjna**

Zbiór atrybutów Y jest przechodnio funkcjnie zależny od zbioru atrybutów X w schemacie R , jeżeli $X \rightarrow Y$ i istnieje zbiór atrybutów Z , nie będący podzbiorem żadnego klucza schematu R taki, że zachodzi $X \rightarrow Z$ i $Z \rightarrow Y$

Zależność funkcyjna $X \rightarrow Y$ jest zależnością przechodnią jeżeli istnieje podzbiór atrybutów Z taki, że zachodzi $X \rightarrow Z$, $Z \rightarrow Y$ i nie zachodzi $Z \rightarrow X$ lub $Y \rightarrow Z$



trzecia postać normalna 3NF (3)

DEFINICJA

Dana relacja r o schemacie R jest w trzeciej postaci normalnej (3NF), jeżeli dla każdej zależności funkcyjnej $X \rightarrow A$ w R spełniony jest jeden z następujących warunków:

- X jest nadkluczem schematu R , lub
- A jest atrybutem podstawowym schematu R



trzecia postać normalna 3NF (4)

Pracownicy-PP-1

Nazwisko	Instytut
Brzeziński	I.Informatyki
Morzy	I.Informatyki
Koszlajda	I.Informatyki
Królikowski	I.Informatyki
...	...
Babij	ElektroEnerg.
Kordus	ElektroEnerg.
Sroczan	ElektroEnerg.

Pracownicy-PP-2

Instytut	Wydział
I.Informatyki	Elektryczny
...	Elektryczny
ElektroEnerg.	Elektryczny

ANOMALIA

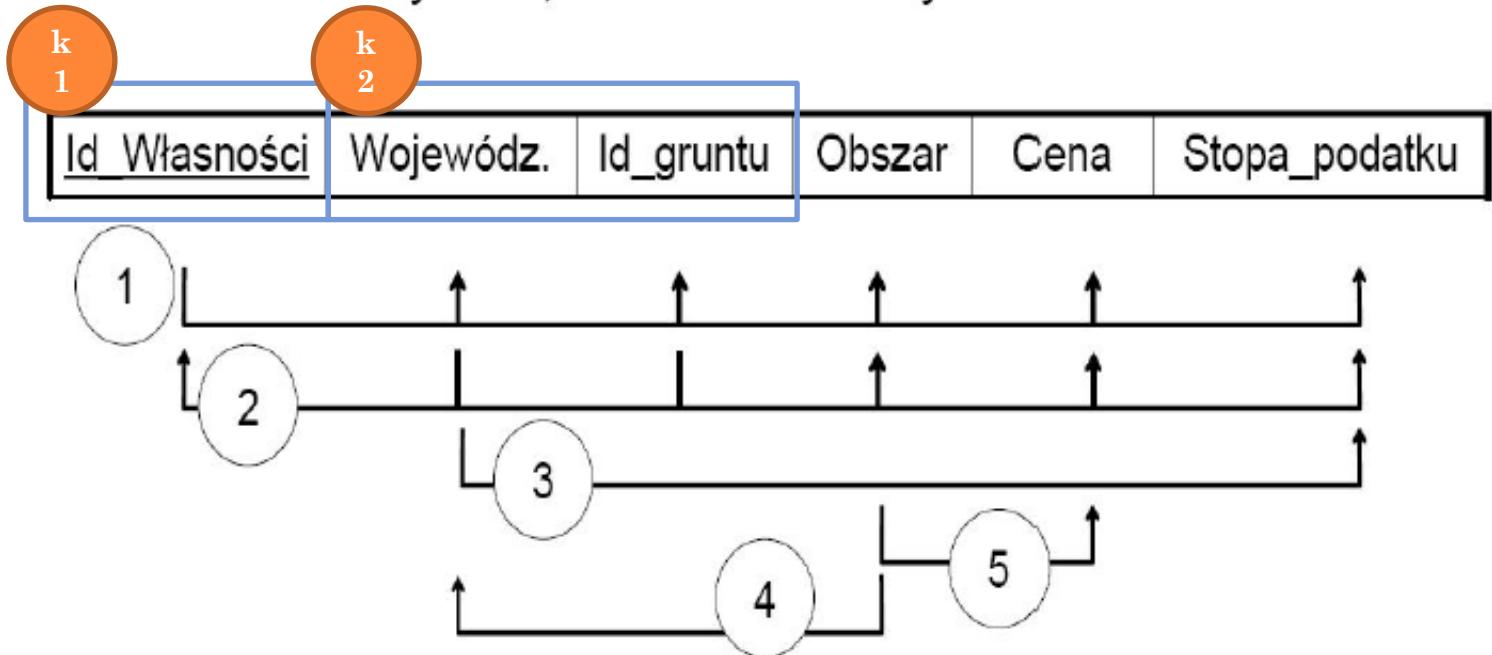
- Redundancji danych
- Wprowadzania danych
- Usuwania danych
- Uaktualniania danych

Przykład procesu normalizacji

UNF	1NF	2NF	3NF
	Relacja R jest w pierwszej postaci normalnej (1NF) wtedy i tylko wtedy, gdy wszystkie użyte dziedziny zawierają tylko atomowe wartości	Relacja R jest w drugiej postaci normalnej (2NF) wtedy i tylko wtedy gdy jest w postaci 1NF oraz każdy niekluczowy atrybut jest w pełni funkcjonalnie zależny od klucza głównego	Relacja R jest z trzeciej postaci normalnej (3NF) wtedy i tylko wtedy gdy jest w 2NF oraz każdy niekluczowy atrybut jest nietranzytywnie (tylko bezpośrednio) zależny od klucza głównego
Zapewnij, aby wszystkie encje były jednoznacznie identyfikowalne przez kombinację atrybutów i/lub ich związków	Usuń powtarzające się atrybuty lub grupy atrybutów i rozłoż atrybuty	Usuń wszystkie atrybuty, które zależą tylko od części jednoznacznego identyfikatora.	Usuń atrybuty zależne od atrybutów, które nie są częścią jednoznacznego identyfikatora.
UNF 1	1NF 2	2NF 3	3NF 7
LOT <u>data godzina</u> <u>nr lotu</u> nazwa linii lotniczej nazwa lotniska typ samolotu pojemność samolotu osoba 1 rola 1 osoba 2 rola 2 osoba 3 rola 3	LOT <u>data godzina</u> <u>nr lotu</u> nazwa linii lotniczej nazwa lotniska typ samolotu pojemność samolotu	LOT <u>data godzina</u> <u>nr lotu</u> nazwa linii lotniczej nazwa lotniska typ samolotu pojemność samolotu	TRASA <u>nr lotu</u> nazwa linii lotniczej nazwa lotniska typ samolotu pojemność samolotu
	ZAŁOGA <u>nazwisko</u> imię rola	ZAŁOGA <u>nazwisko</u> imię rola	ZAŁOGA rola
			OSOBA imię nazwisko

postać Boyce-Codd'a (1)

- Postać normalna Boyce-Codd'a stanowi warunek dostateczny 3NF, ale nie konieczny

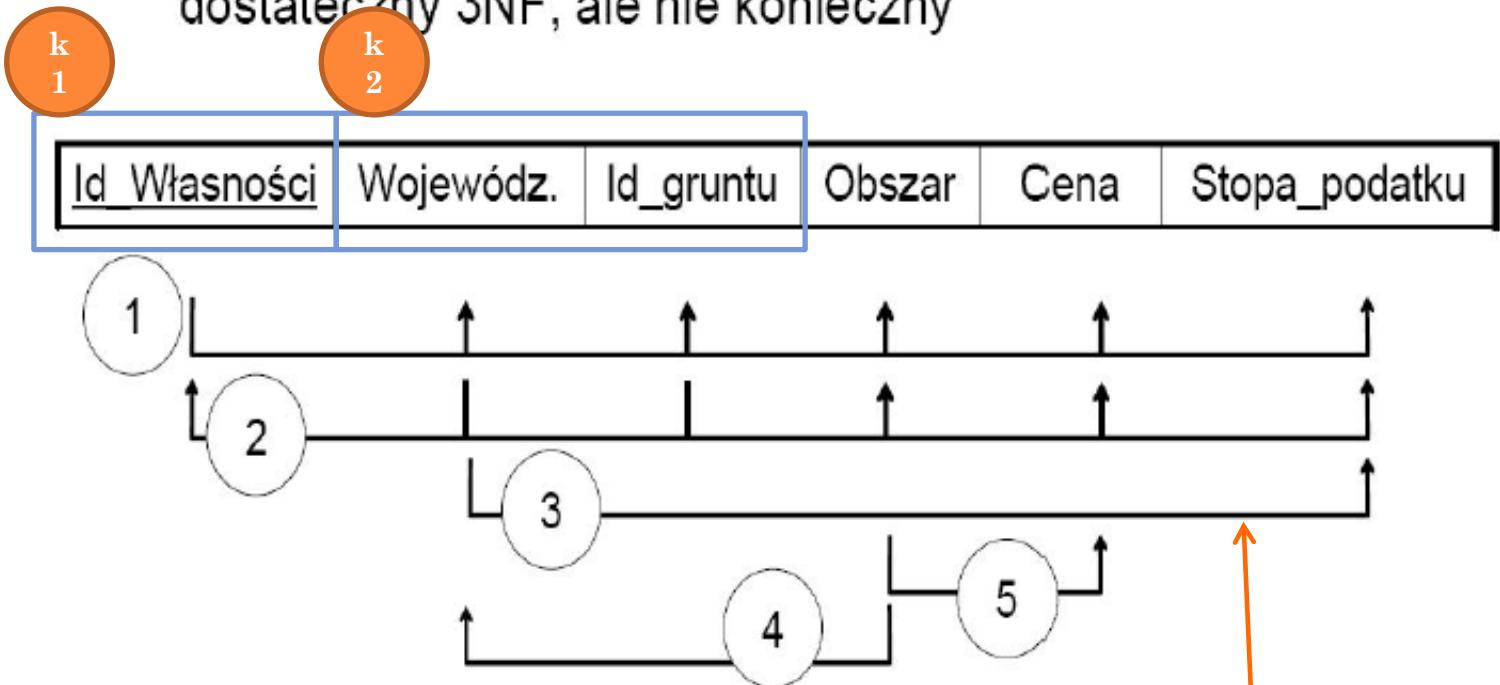


Schemat relacji jest w 1NF i posiada:

- Dwa klucze: K1 i K2
- Atrybuty podstawowe: Id_Własności, Województwo, Id_gruntu
- Atrybuty wtórne: Obszar, Cena, Stopa_podatku.

postać Boyce-Codd'a (1)

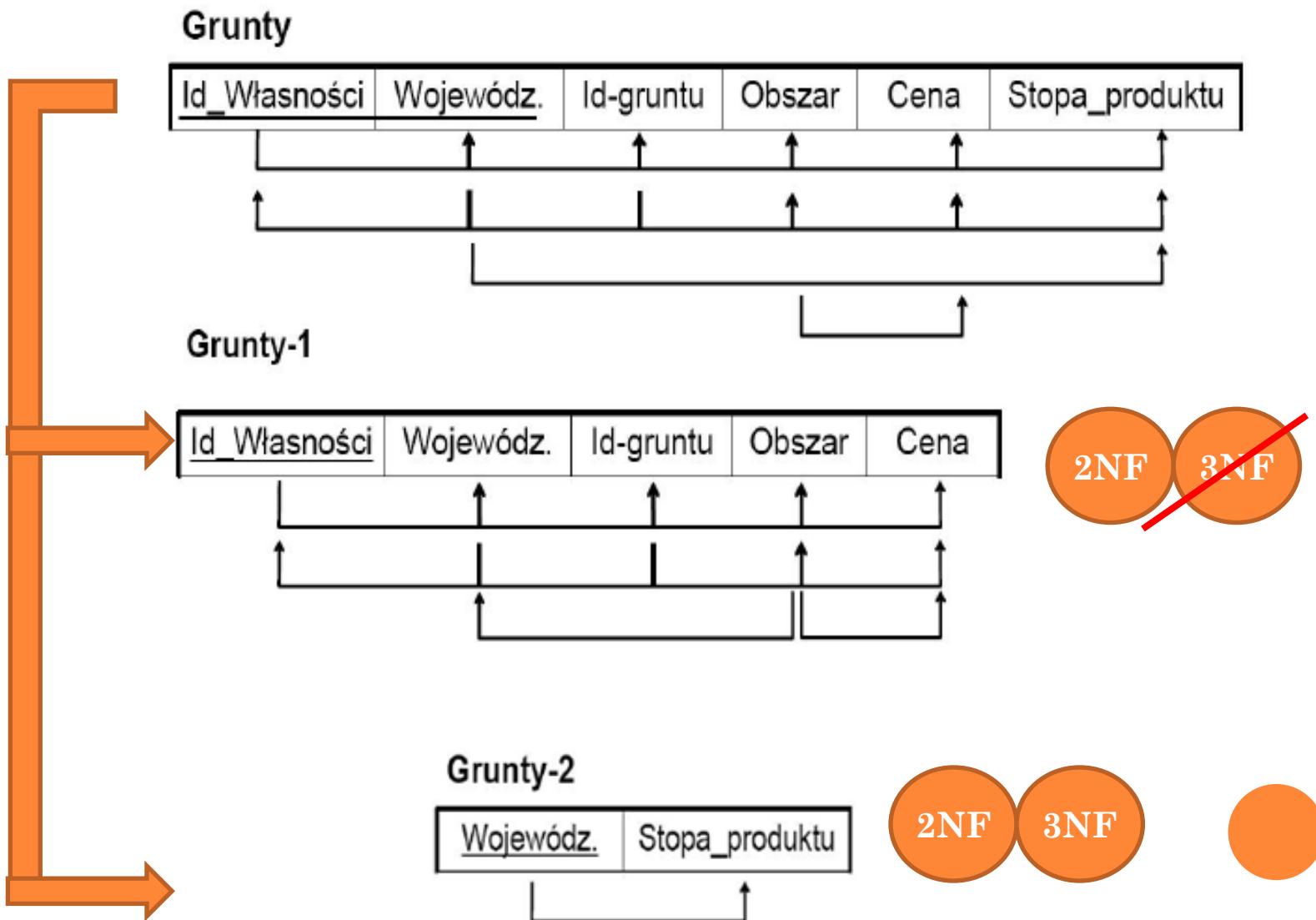
- Postać normalna Boyce-Codd'a stanowi warunek dostateczny 3NF, ale nie konieczny



- fd1: zależności od klucza
- fd2: zależności od klucza
- fd3: Wojewódz. → Stopa_podatku
- fd4: obszar → Wojewódz.
- fd5: obszar → Cena

2NF
Stopa_podatku (częściowo
funkcyjnie zależna)

postać Boyce-Codd'a (2)



postać Boyce-Codd'a (3)

Grunty-1

<u>Id_Własności</u>	Wojewódz.	Id-gruntu	Obszar	Cena

```
graph TD; A[Id_Własności] --> B[Wojewódz.]; A --> C[Id-grantu]; A --> D[Obszar]; A --> E[Cena]; B --> A; C --> A; D --> A; E --> A;
```

2NF
3NF

Grunty-1A

<u>Id_Własności</u>	Wojewódz.	Id-gruntu	Obszar

```
graph TD; A[Id_Własności] --> B[Wojewódz.]; A --> C[Id-grantu]; A --> D[Obszar]; B --> A; C --> A; D --> A;
```

2NF

3NF

Grunty-1B

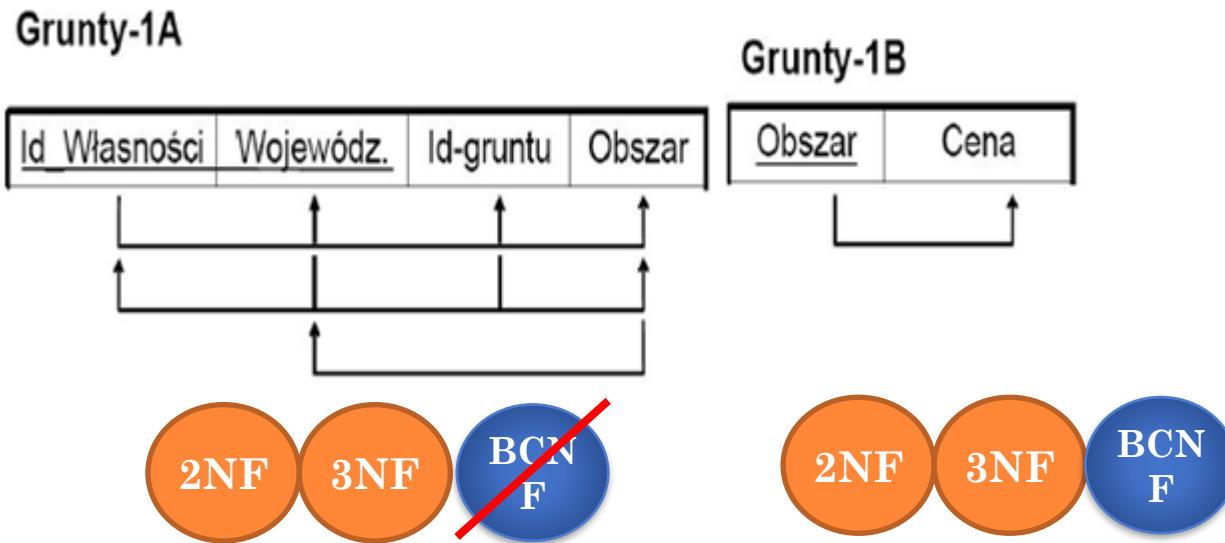
Obszar	Cena

```
graph TD; A[Obszar] --> B[Cena];
```

2NF

3NF

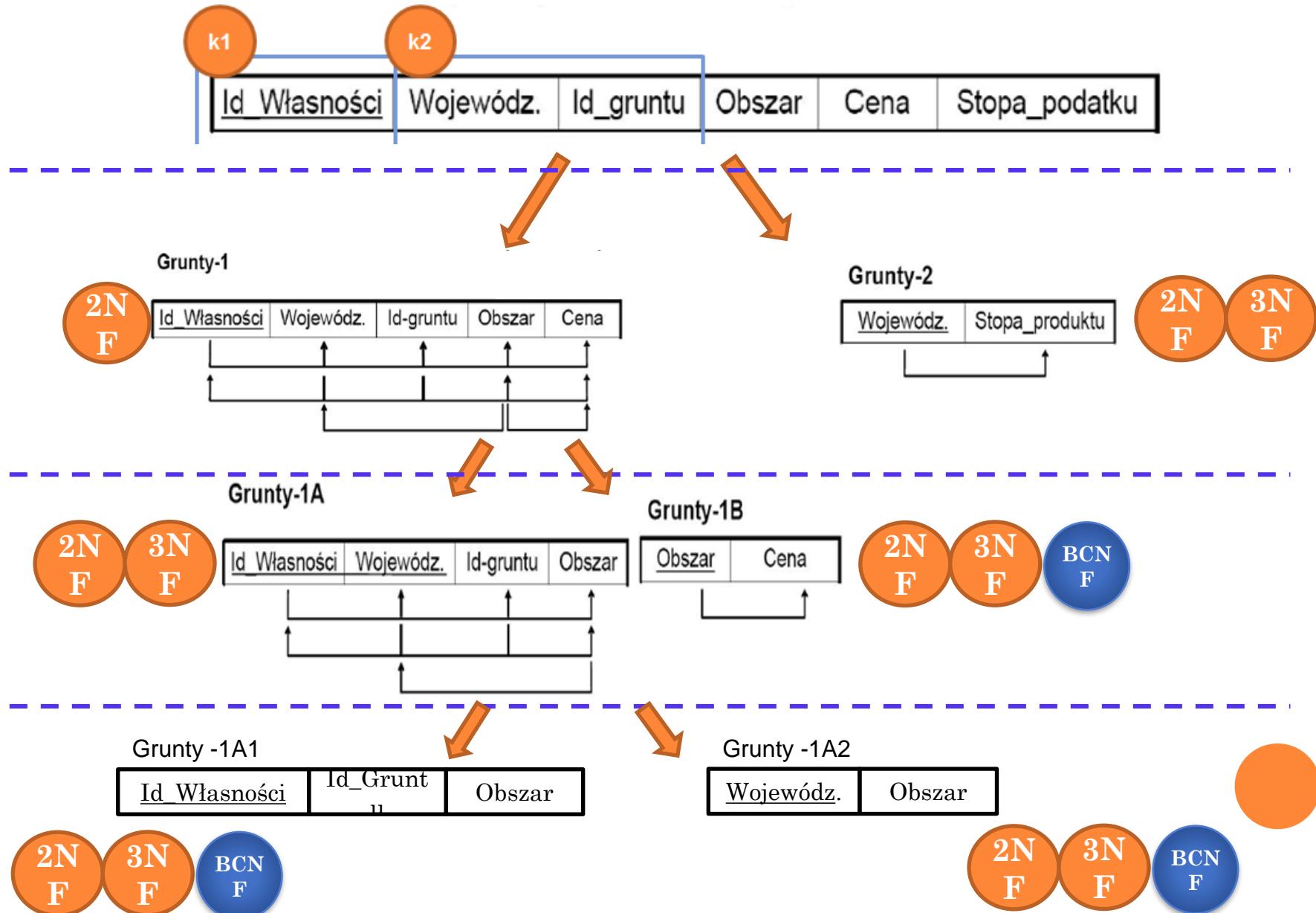
postać Boyce-Codd'a (3)



Dana relacja r o schemacie R jest w postaci normalnej Boyce'aCodd'a (BCNF), jeżeli dla każdej zależności funkcyjnej $X \rightarrow A$ w R spełniony jest następujący warunek: X jest nadkluczem schematu R .

W tym przypadku, zachodzi konieczność dekompozycji relacji Grunty-1A na dwa schematy relacji: Grunty1A1 (Id_Własności, Id_Gruntu, Obszar) oraz Grunty1A2 (Obszar, Województwo). 

postać Boyce-Codd'a



zależności wielowartościowe (1)

Loty

Lot	Dzień_tygodnia	Typ_samolotu
106	poniedziałek	134
106	czwartek	154
106	poniedziałek	154
106	czwartek	134
206	środa	747
206	piątek	767
206	środa	767
206	piątek	747

3NF

BCNF

Problem
modyfikacji !

Języki

Nazwisko	Język_obcy	Język_prog.
Nowak	angielski	Basic
Nowak	włoski	Fortran
Nowak	angielski	Fortran
Nowak	włoski	Basic
Nowak	czeski	Basic
Nowak	czeski	Fortran

3NF

BCNF

Problem modyfikacji
!

modyfikacja relacji

- Lot 106 będzie dodatkowo odbywał się w Środę i na tę linię wprowadzamy, dodatkowo, nowy typ samolotu – 104

Loty	Lot	Dzień-tygodnia	Typ-samolotu
	106	poniedziałek	134
	106	czwartek	154
	106	poniedziałek	154
	106	czwartek	134
5 nowych krotek	106	poniedziałek	104
	106	czwartek	104
	106	środa	134
	106	środa	154
	106	środa	104

Utrudniona pielęgnacja !

zależności wielowartościowe (1)

6 nowych krotek

Języki

Nazwisko	Język_obcy	Język_prog.
Nowak	angielski	Basic
Nowak	włoski	Fortran
Nowak	angielski	Fortran
Nowak	włoski	Basic
Nowak	czeski	Basic
Nowak	czeski	Fortran

Utrudniona pielęgnacja !

dekompozycja

Lot-1

Lot	Dzień-tygodnia
106	poniedziałek
106	czwartek
206	środa
206	piątek
106	środa

Lot-2

Lot	Typ-samolotu
106	134
106	154
206	747
206	767
106	104

Język-1

Nazwisko	Język_obcy
Nowak	angielski
Nowak	włoski
Nowak	czeski
Nowak	francuski

Język-2

Nazwisko	Język_prog.
Nowak	Basic
Nowak	Fortran
Nowak	C++



zależności wielowartościowe (2)

- Zależności wielowartościowe są konsekwencją wymagań pierwszej postaci normalnej, która nie dopuszcza, aby krotki zawierały atrybuty wielowartościowe
- Zależność wielowartościowa występuje w relacji $r(R)$ nie dlatego, że na skutek zbiegu okoliczności tak ułożyły się wartości krotek, lecz występuje ona dla dowolnej relacji r o schemacie R dlatego, że odzwierciedla ona ogólną prawidłowość modelowanej rzeczywistości

$Lot \rightarrow Dzień\text{-tygodnia}$

$Lot \rightarrow Typ\text{-samolotu}$

$Nazwisko \rightarrow Język\text{-obcy}$

$Nazwisko \rightarrow Język\text{-programowania}$

Zależności
wielowartościowe

Loty

Lot	Dzień_tygodnia	Typ_samolotu
106	poniedziałek	134
106	czwartek	154
106	poniedziałek	154
106	czwartek	134

Języki

Nazwisko	Język_obcy	Język_prog.
Nowak	angielski	Basic
Nowak	włoski	Fortran
Nowak	angielski	Fortran
Nowak	włoski	Basic
Nowak	czeski	Basic

zależności wielowartościowe (3)

- Wystąpienie zależności wielowartościowej $X \rightarrow\!\!\! \rightarrow Y$ w relacji o schemacie $R = XYZ$ wyraża dwa fakty:
 - Związek pomiędzy zbiorami atrybutów X i Y ;
 - Niezależność zbiorów atrybutów Y, Z . Zbiory te są związane ze sobą pośrednio poprzez zbiór atrybutów X

Lot-3	Lot	Dzień-tygodnia	Typ-samolotu
	106	poniedziałek	134
	106	czwartek	154
	106	czwartek	134
	206	środa	747
	206	piątek	767



czwarta postać normalna 4NF

Relacja r o schemacie R jest w czwartej postaci normalnej (4NF) względem zbioru zależności wielowartościowych MVD jeżeli jest ona w $3NF$ i dla każdej zależności wielowartościowej $X \rightarrow\!\!\rightarrow Y \in MVD$ zależność ta jest trywialna lub X jest nadkluczem schematu.

1. Zależność wielowartościowa $X \rightarrow\!\!\rightarrow Y$ w relacji $r(R)$ nazywamy zależnością trywialną, jeżeli
 - zbiór Y jest podzbiorem X , lub
 - $X \cup Y = R$
2. Zależność nazywamy trywialną, gdyż jest ona spełniona dla dowolnej instancji r schematu R



dekompozycja relacji na relacje bez utraty informacji (1)

Dekompozycja na relacje w 3NF

Dana jest relacja r o schemacie R , i dany jest zbiór F zależności funkcyjnych dla R . Niech relacje r_1 i r_2 o schematach, odpowiednio, R_1 i R_2 , oznaczają dekompozycję relacji $r(R)$. Dekompozycja ta jest dekompozycją bez utraty informacji, jeżeli co najmniej jedna z poniższych zależności funkcyjnych jest spełniona:

$$R_1 \cap R_2 \rightarrow R_1$$

$$R_1 \cap R_2 \rightarrow R_2$$



dekompozycja relacji na relacje bez utraty informacji (2)

Dekompozycja na relacje w 4NF

Dana jest relacja r o schemacie R . Niech relacje r_1 i r_2 o schematach, odpowiednio, R_1 i R_2 , oznaczają dekompozycję relacji $r(R)$. Dekompozycja ta jest dekompozycją bez utraty informacji, jeżeli co najmniej jedna z poniższych zależności wielowartościowych jest spełniona:

$$R_1 \cap R_2 \rightarrow\rightarrow (R_1 - R_2)$$

$$R_1 \cap R_2 \rightarrow\rightarrow (R_2 - R_1)$$



dekompozycja relacji na relacje bez utraty informacji (2)

Lot-1

Lot	Dzień-tygodnia
106	poniedziałek
106	czwartek
206	środa
206	piątek
...	...
106	środa

Lot-2

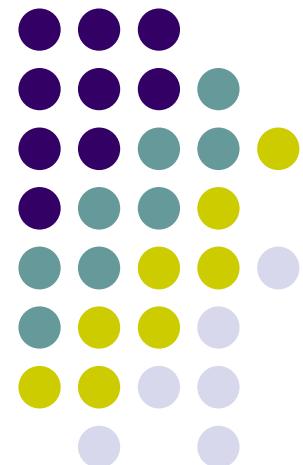
Lot	Typ-samolotu
106	134
106	154
206	747
206	767
...	...
106	104

Koniec wykładu



wykład

Indeksy





Plan wykładu

- Problematyka indeksowania
- Podział indeksów i ich charakterystyka
 - indeks podstawowy, zgrupowany, wtórny
 - indeks rzadki, gęsty
- Indeks wielopoziomowy statyczny (ISAM)
- Indeks wielopoziomowy dynamiczny (B+-drzewo)
- Algorytm wstawiania danych do indeksu B+-drzewo



wprowadzenie (1)

Problem:

- Dany jest plik zawierający uporządkowane lub nieuporządkowane rekordy danych
- W jaki sposób efektywnie zrealizować wyszukanie rekordu lub rekordów z zadанego zakresu wartości wybranego pola?

Rozwiązanie:

TAKI PLIK DODATKOWY NAZWA SIĘ INDEKSEM

- Utworzyć drugi plik, zdefiniowany na atrybucie wykorzystanym do specyfikacji kryterium poszukiwania
- Plik ten zawiera rekordy odpowiadające poszukiwanym wartościom pierwszych rekordów w poszczególnych blokach pliku danych
- Rekordy w dodatkowym pliku mają postać:
< pierwszy klucz w bloku, wskaźnik do bloku >
- Plik dodatkowy jest uporządkowany według wartości poszukiwanych



wprowadzenie (2)

Indeks - dodatkowa struktura fizyczna

Cel stosowania - przyśpieszenie dostępu do danych

- Zakładane na pojedynczych atrybutach lub zbiorach atrybutów relacji
 - atrybuty te są nazywane indeksowymi **ATRYBUTY INDEKSOWE**
- Model fizyczny indeksu
 - uporządkowany plik rekordów indeksu (*ang. data entry*) o stałej długości
 - rekord indeksu zawiera dwa pola **BUDOWA INDEKSU**
 - klucz reprezentujący jedną z wartości występujących w atrybutach indeksowych relacji
 - wskaźnik do bloku danych zawierający krotkę, której atrybut indeksowy równy jest kluczowi



wprowadzenie (3)

OZNACZENIE REKORDU INDEKSU

Rekord indeksu - **k***

- zawiera dostateczną informację umożliwiającą wyszukanie (jednego lub więcej) rekordów danych o wartości klucza k

Pytanie:

PROJEKTOWANIE STRUKTURY INDEKSU

- w jaki sposób rekordy indeksu powinny być zorganizowane, aby efektywnie wspierać wyszukiwanie rekordów o danej wartości klucza?
- co powinien zawierać rekord indeksu?

1?

2?



rekordy indeksu

Typy rekordów indeksu:

1. Rekord indeksu k^* jest rekordem danych (o wartości klucza k)
2. Rekord indeksu jest parą $\langle k, \text{rid} \rangle$, gdzie rid jest identyfikatorem rekordu danych o wartości klucza k
3. Rekord indeksu jest parą $\langle k, \text{rid-list} \rangle$, gdzie rid-list jest listą identyfikatorów rekordów danych o wartości klucza k
4. Rekord indeksu jest parą $\langle k, \text{bitmapa} \rangle$, gdzie bitmapa jest wektorem 0 i 1 reprezentującym zbiór rekordów danych

Rekord indeksu k^* umożliwia wyszukanie rekordów danych o wartości klucza k



rodzaje indeksów

charakterystyka atrybutu indeksowego

Indeks podstawowy (primary index)

- założony na atrybutie porządkującym unikalnym

Indeks zgrupowany (clustering index)

- założony na atrybutie porządkującym nieunikalnym

Indeks wtórny (secondary index)

- założony na atrybutie nieporządkującym

1

wskazania do pliku danych

Indeks gęsty (dense)

- posiada rekord indeksu dla każdego rekordu indeksowanego pliku danych

Indeks rzadki (sparse)

- posiada rekordy tylko dla wybranych rekordów indeksowanego pliku danych

2

liczba poziomów

Indeks jednopoziomowe

- jeden plik indeksu dla jednego pliku danych

Indeks wielopoziomowe

- indeks do indeksu

3



indeks podstawowy

Rekord indeksowy indeksu podst.

wskazania z rekordów
indeksowych → do
bloków danych

pole porządkujące
unikalne

Aaron, Ed		
Abbot, Diane		
...		
Acosta, Marc		

wartość
indeksowana wskażnik
do bloku

Aaron, Ed	
Adams, John	
Alexander, Ed	
Allen, Troy	
Anderson, Zach	
Arnold, Mack	
...	

...

...	
Wong, James	
Wright, Pam	

Adams, John		
Adams, Robin		
...		
Akers, Jan		

Alexander, Ed		
Alfred, Bob		
...		
Allen, Sam		

Allen, Troy		
Anders, Keith		
...		
Anderson, Rob		

Wright, Pam		
Wyatt, Charles		
...		
Zimmer, Byron		

BLOKI /pliki/
DANYCH

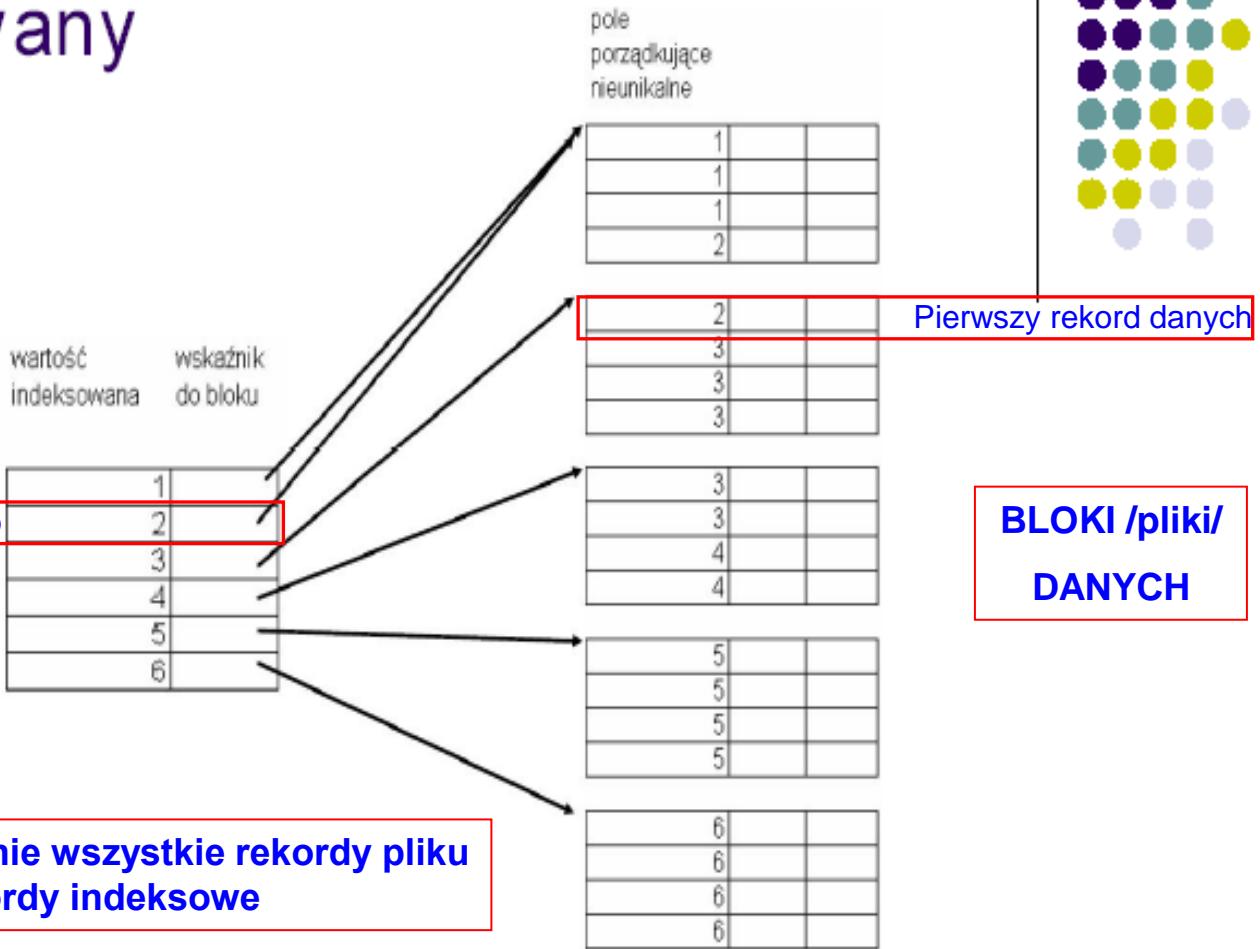
Jest indeksem rzadkim ponieważ nie wszystkie rekordy pliku
danych posiadają rekordy indeksowe



indeks zgrupowany

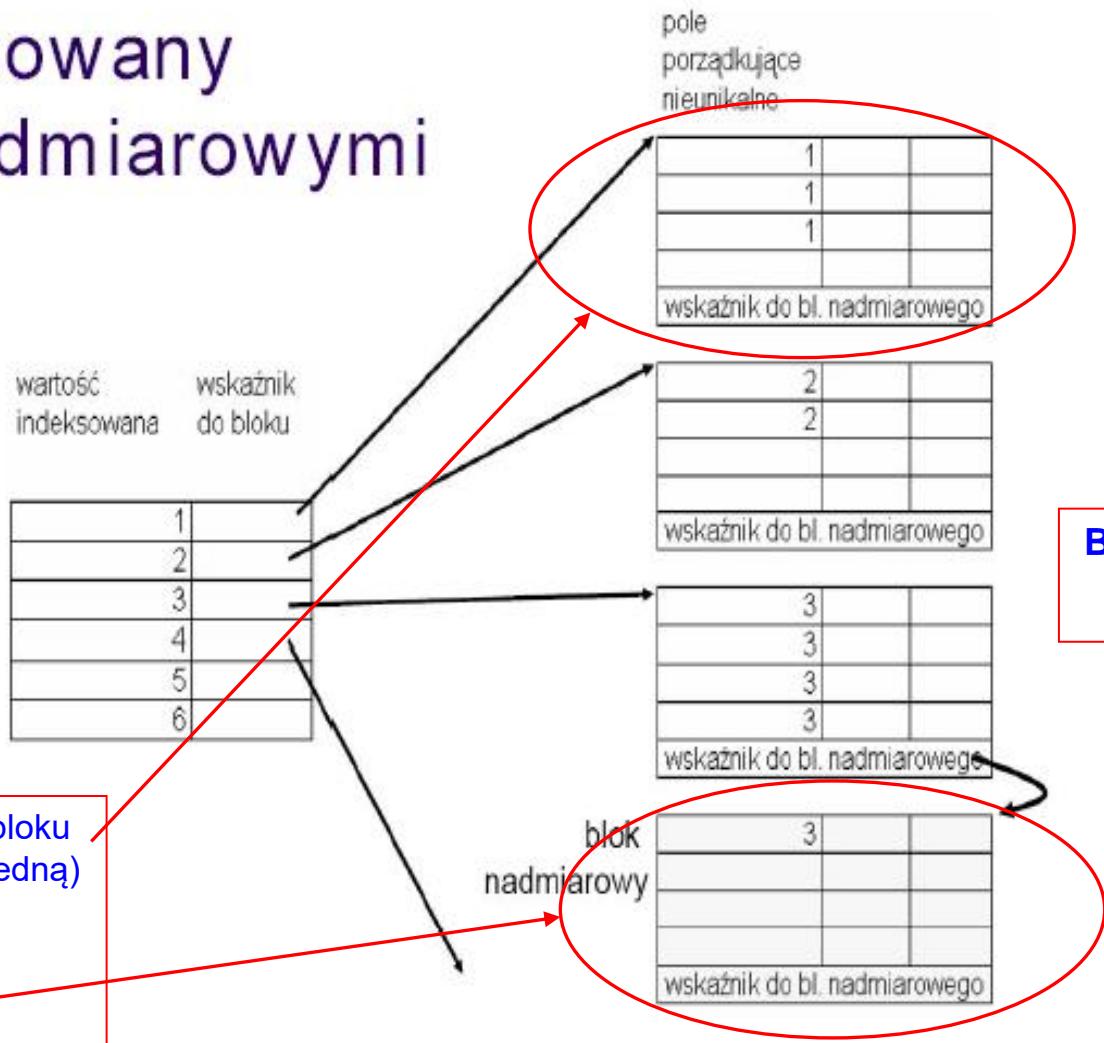
Rekord indeksowy indeksu zgrupowanego (dla wartości X) zawiera adres bloku danych, w którym znajduje się pierwszy rekord danych z wartością atrybutu indeksowego równą X

Rekord indeksowy ind. zgrupowanego





indeks zgrupowany z blokami nadmiarowymi





indeks wtórny GĘSTY

Każdy rekord pliku danych posiada swój odpowiednik w rekordzie indeksu

Rekord indeksu

wartość indeksowana wskaźnik do bloku

1	
2	
3	
4	
5	
6	
...	

Rekord pliku danych

pole nieporządkujące nieunikalne

9	
5	
13	
8	

6	
15	
3	
17	

21	
11	
16	
2	

24	
10	
4	
1	

...

Jest indeksem uporządkowanym zakładany na atrybucie indeksowym pliku danych, który nie jest atrybutem porządkującym tego pliku.

BLOKI /pliki/
DANYCH

indeks o kluczu złożonym

ZAPYTANIA PUNKTOWE



Klucz złożony Indeks<wiek; pensja>

1

32; 4300
35; 4500
42; 5000
46; 5800

nazwisko	wiek	pensja
Turecki	35	4500
Wolski	46	5800
Biedronka	42	5000
Maślak	32	4300

Klucz złożony Indeks<pensja; wiek>

2

4300; 32
4500; 35
5000; 42
5800; 46

Klucz Indeks<wiek>

3

32
35
42
46

Indeks<pensja>

Klucz

4

4300
4500
5000
5800

Atrybut wiodący – problem wyszukiwania

Indeksy na kluczach złożonych – wyszukiwanie rekordów spełniających warunki równościowe

Klucz indeksu może zawierać kilka atrybutów → tzw. KLUCZ ZŁOŻONY



indeks wielopoziomowy

1

- **ISAM** - Indexed Sequential Access Method (IBM)
 - poziom pierwszy:
 - indeks cylindrów <klucz, adres indeksu ścieżki>
 - poziom drugi:
 - indeks ścieżki <klucz, adres ścieżki>

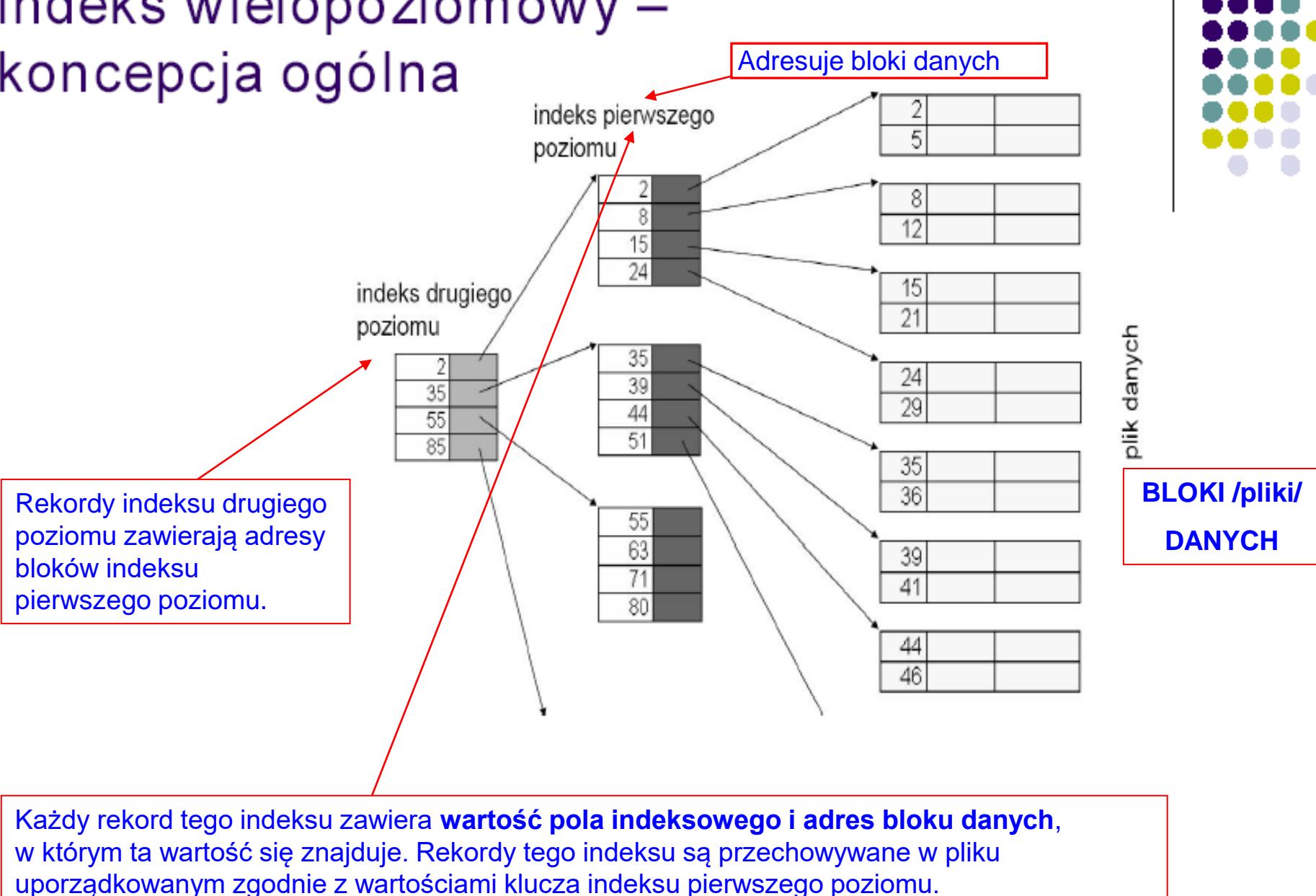
2

- **VSAM** - Virtual Sequential Access Method
 - rozwinięcie ISAM
 - niezależne od sprzętu

Indeks wielopoziomowy → którego efektywność przeszukiwania jest większa. Jedną z fundamentalnych koncepcji indeksu wielopoziomowego jest struktura ISAM (ang. Indexed Sequential Access Method), oryginalnie opracowana przez IBM.



indeks wielopoziomowy – koncepcja ogólna





Indeks statyczny

- modyfikowanie zawartości pliku degeneruje indeks - spada efektywność dostępu do danych
- powstają puste obszary po usuniętych rekordach
- wstawiane rekordy trafiają do bloków nadmiarowych

ISAM to indeks statyczny: nie posiada zaawansowanych mechanizmów modyfikowania struktury w sytuacji zmodyfikowania zawartości indeksowanego pliku.

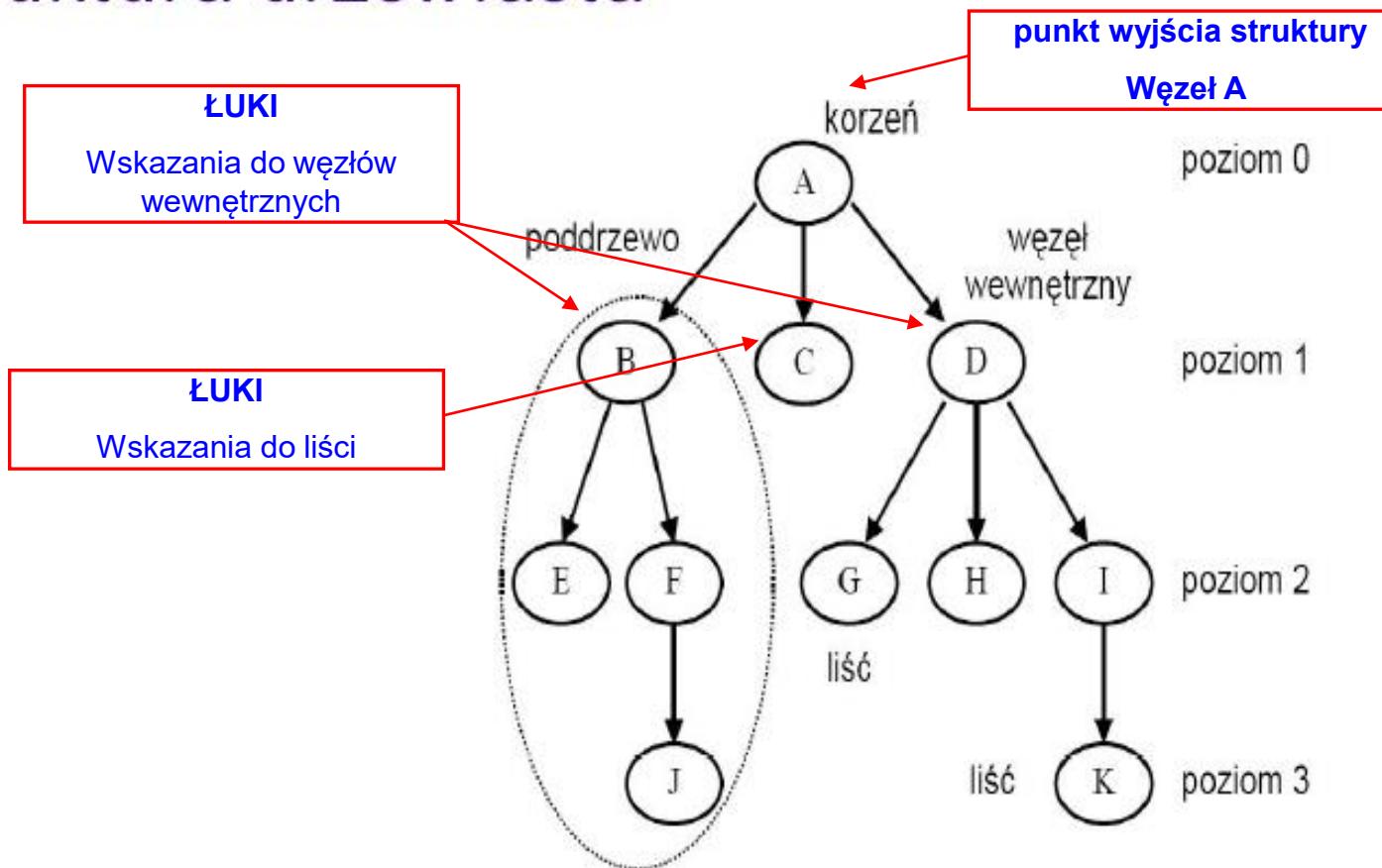
W konsekwencji indeks typu ISAM jest nieefektywny.

Rozwiązaniem tego problemu jest wprowadzenie indeksów dynamicznych.

Najpowszechniej stosowanymi indeksami dynamicznymi są indeksy drzewiaste, S - drzewa, B – drzewa oraz B⁺ drzewa.



struktura drzewiasta



Węzeł wewnętrzny posiada wskazania do innych węzłów. Liść nie posiada wskazań do innych węzłów. Jest więc elementem końcowym całej struktury. Przykładowy indeks ze slajdu składa się z 4 poziomów. Przy czym korzeń znajduje się na poziomie 0.



indeks B⁺-drzewo

Zrównoważona struktura drzewiasta

- wierzchołki wewnętrzne służą do wspomagania wyszukiwania
- wierzchołki liści zawierają rekordy indeksu ze wskaźnikami do rekordów danych

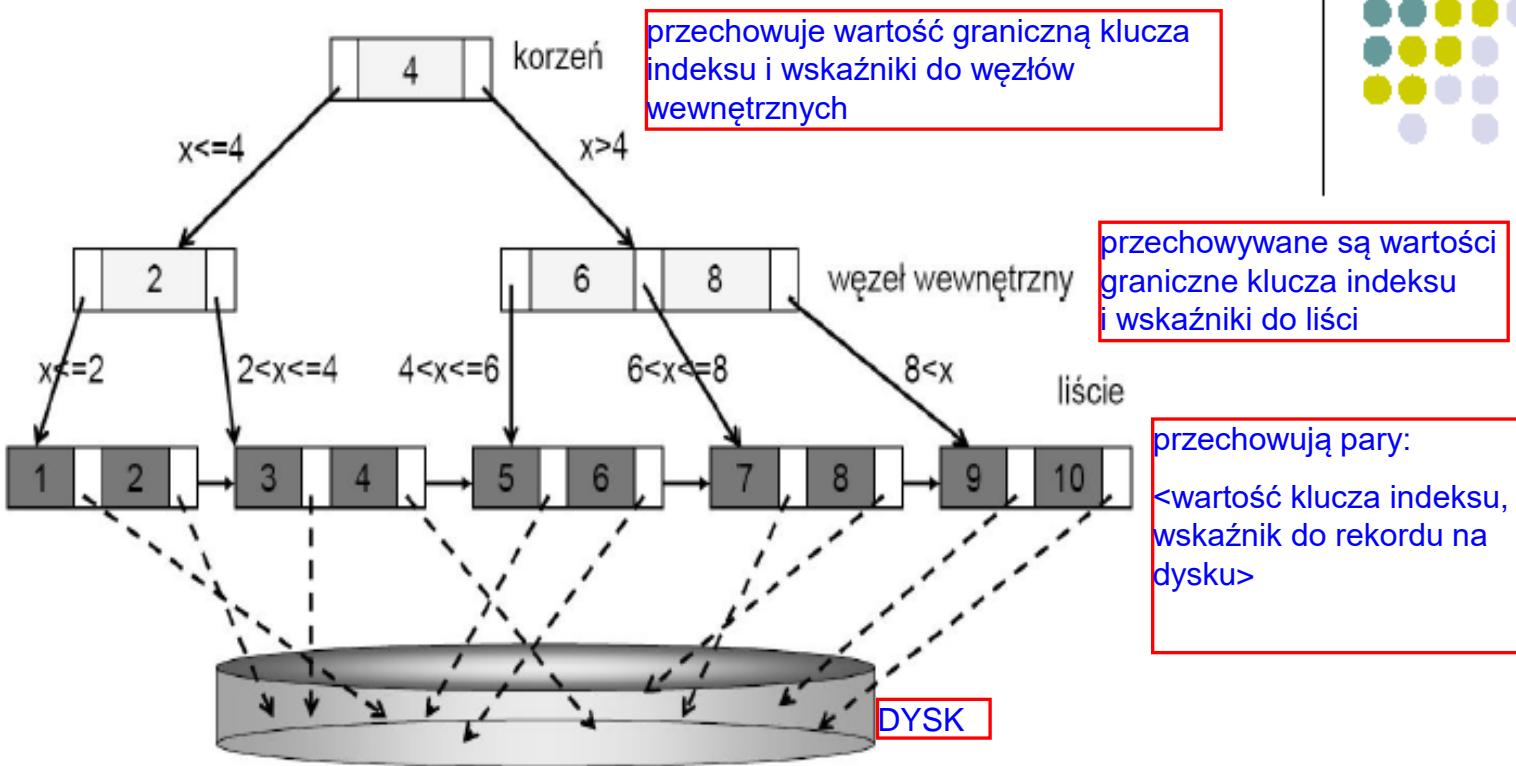
W celu zapewnienia odpowiedniej efektywności realizacji zapytań przedziałowych wierzchołki liści stanowią listę dwukierunkową

charakterystyka

- Operacje wstawiania i usuwania rekordów indeksu pozostawiają indeks zrównoważony
- Każdy wierzchołek jest wypełniony w co najmniej 50% (za wyjątkiem korzenia)
 - usuwanie rekordów może skutkować mniejszym wypełnieniem niż 50%
- Wyszukanie rekordu wymaga przejścia od korzenia do liścia
 - długość ścieżki od korzenia do dowolnego liścia nazywamy wysokością drzewa indeksu
- Indeks zrównoważony

indeks B⁺-drzewo

STRUKTURA TRZY - POZIOMOWA



W przykładzie wartością graniczną jest 4, a korzeń zawiera wskaźniki do dwóch węzłów wewnętrznych.



węzeł wewnętrzny (1)

- Struktura węzła wewnętrznego indeksu B⁺-drzewo rzędu p jest następująca

1

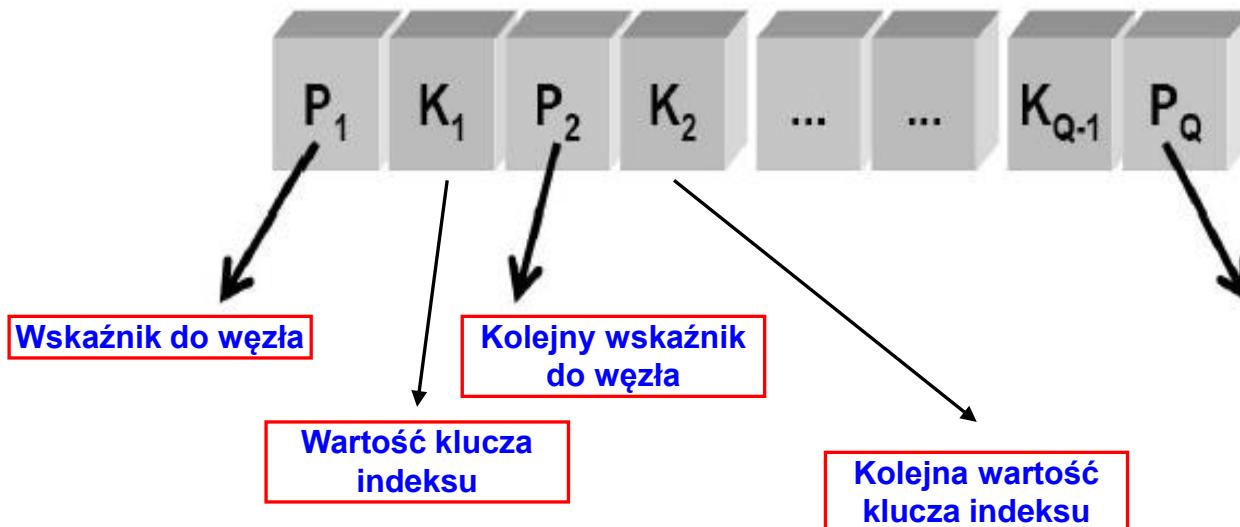
Węzeł wewnętrzny ma postać:

$\langle P_1, K_1, P_2, \dots, P_{Q-1}, K_{Q-1}, P_Q \rangle$

2

Dla każdego wierzchołka wewnętrznego zachodzi

$K_1 < K_2 < \dots < K_{Q-1}$



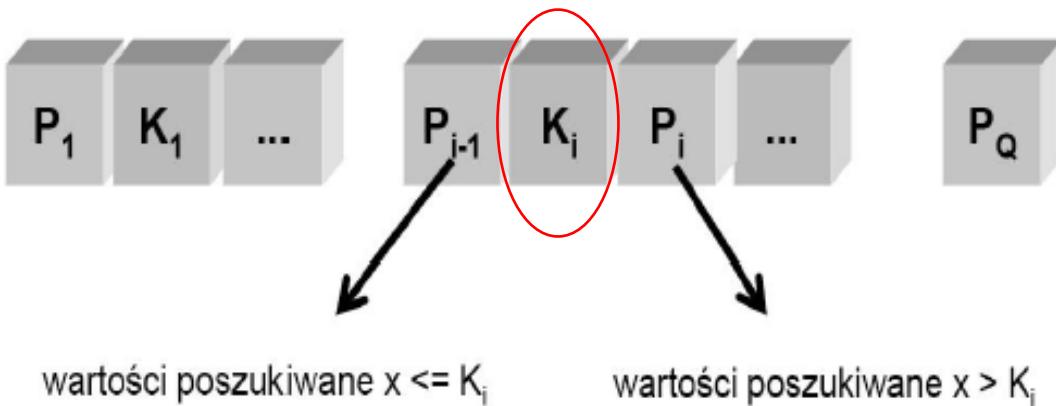
Wartości klucza indeksowego są uporządkowane (od lewej – wartości najmniejsze do prawej wartości największe).



węzeł wewnętrzny (2)

3

- Dla danej wartości K_i klucza w węźle zewnętrzny
 - lewy wskaźnik prowadzi do poddrzewa zawierającego wartości poszukiwane $\leq K_i$
 - prawy wskaźnik prowadzi do poddrzewa zawierającego wartości poszukiwane $> K_i$



Dla danej wartości K_i :
lewy wskaźnik $\leq K_i$
prawy wskaźnik $> K_i$

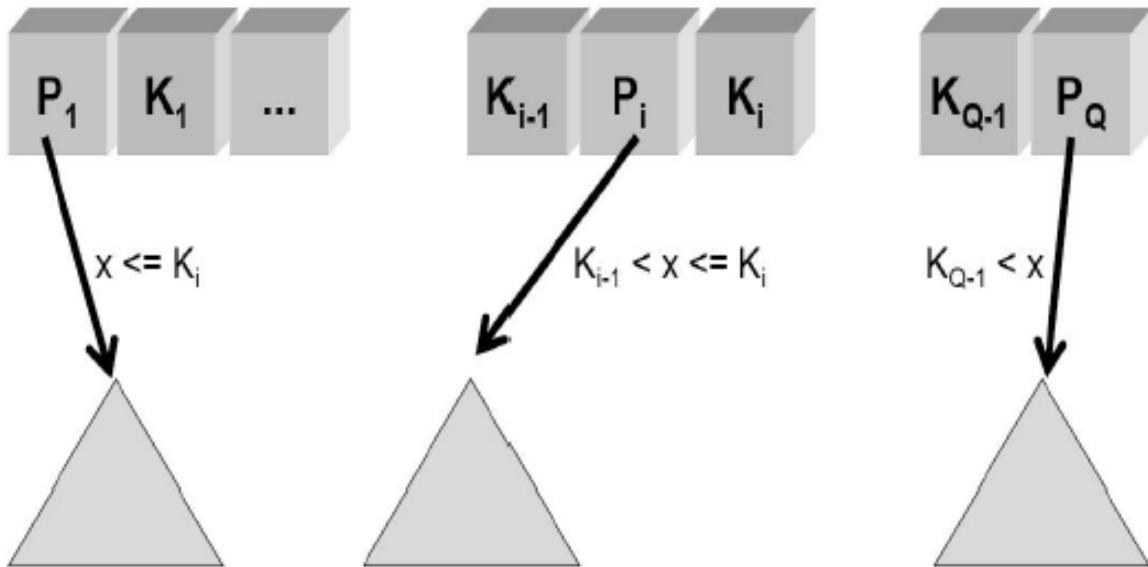


węzeł wewnętrzny (3)

- 4 Każdy wierzchołek wewnętrzny posiada co najwyżej p wskaźników do poddrzew
- 5 Każdy wierzchołek wewnętrzny, za wyjątkiem korzenia, posiada co najmniej ($p/2$) wskaźników do poddrzew
 - korzeń posiada co najmniej 2 wskaźniki do poddrzew
- 6 Każdy wierzchołek wewnętrzny o Q wskaźnikach posiada Q-1 wartości kluczy



węzeł wewnętrzny (4)



Każdy wierzchołek wewnętrzny posiada co najwyżej p wskaźników do poddrzew

Każdy wierzchołek wewnętrzny, za wyjątkiem korzenia, posiada co najmniej $(p/2)$ wskaźników do poddrzew: korzeń posiada co najmniej 2 wskaźniki do poddrzew

Każdy wierzchołek wewnętrzny o Q wskaźnikach posiada $Q-1$ wartości kluczy



liść

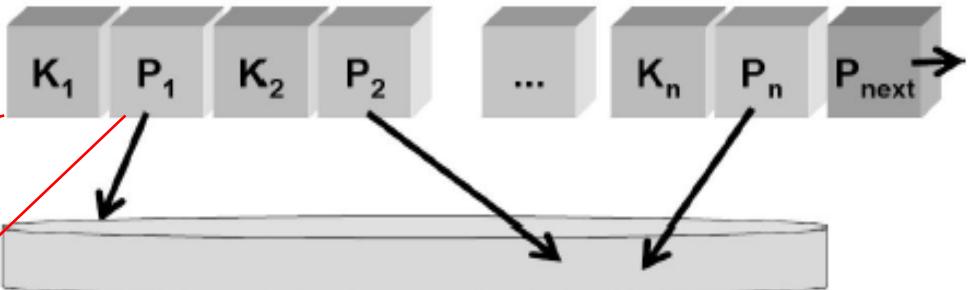
Struktura liścia indeksu B⁺-drzewo rzędu p jest następująca:

1. Liść ma postać:

$\langle\langle K_1, P_1 \rangle, \langle K_2, P_2 \rangle, \dots, \langle K_k, P_k \rangle, P_{next} \rangle$ **Zbiór par**

2. Dla każdego wierzchołka liścia zachodzi:

$K_1 < K_2 < \dots < K_k$



3. Każdy liść posiada co najmniej $(p/2)$ wartości kluczowe
4. Wszystkie liście znajdują się na tym samym poziomie (tej samej wysokości)



obliczanie rzędu indeksu

- Dane: rozmiar klucza V; rozmiar wskaźnika do bloku P; rozmiar bloku B; liczba rekordów w indeksowanym pliku danych r; liczba bloków pliku b
- Węzły wewnętrzne zawierają maksymalnie p wskaźników i p-1 kluczy
- Każdy z węzłów musi zmieścić się w pojedynczym bloku dyskowym – rząd B⁺-drzewa p jest największą liczbą całkowitą, dla której spełniona jest nierówność: $(p \cdot P) + [(p - 1) \cdot V] \leq B$
- Minimalna wysokość indeksu rzadkiego: $h = \log_p b$
- Minimalna wysokość indeksu gęstego: $h = \log_p r$

1

2

3

Węzły wewnętrzne zawierają maksymalnie p wskaźników i p-1 kluczy. Ponieważ każdy węzeł musi się zmieścić w pojedynczym bloku dyskowym, więc rząd B + drzewa jest największą liczbą całkowitą spełniającą nierówność (oznaczoną symbolem 1).



obliczanie rzędu indeksu - przykład

Obliczanie rzędu B⁺ - drzewa

Rozmiar pliku: $r = 30\ 000$ rekordów

Rozmiar bloku: $B = 1024B$

Rozmiar rekordu: $R = 100$ bajtów

Rozmiar klucza: $V = 9$

Rozmiar wskaźnika: $P = 6$

Rekordy mają stałą długość i nie są dzielone między bloki

Indeks wtórny

Liczba rekordów w bloku:

$$1 \quad rbl = \left\lfloor \frac{B}{R} \right\rfloor = \left\lfloor \frac{1024}{100} \right\rfloor = 10$$

$$2 \quad \text{Liczba bloków danych: } b = \left\lceil \frac{r}{rbl} \right\rceil = \left\lceil \frac{30000}{10} \right\rceil = 3000$$

Założenia dodatkowe:

Rekordy mają stałą długość i nie są dzielone między bloki.

Na pliku danych jest zakładany indeks wtórny.



obliczanie rzędu indeksu - przykład

Rząd węzła 3

$$(p \cdot P) + [(p-1) \cdot V] \leq B \rightarrow p \leq \frac{V+B}{P+V}$$

Minimalna wysokość indeksu:

5
$$h = \lceil \log_p r \rceil$$

\downarrow

$$h = \lceil \log_{68} 30000 \rceil = 3$$

\downarrow

$$p \leq \frac{9+1024}{6+9}$$
$$p = 68$$

Rozmiar bloku: $B = 1024B$

Rozmiar rekordu: $R = 100$ bajtów

Rozmiar klucza: $V = 9$

Rozmiar wskaźnika: $P = 6$

wstawianie danych do indeksu – przykład (1)



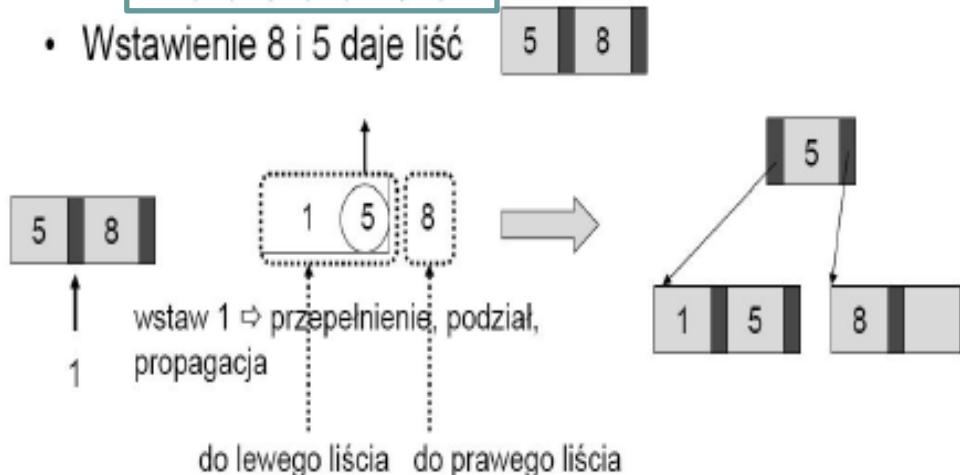
Idea działania
algorytmu
modyfikowania
struktury indeksu

- Założenia
 - indeks B⁺-drzewo
 - rząd drzewa: 3
 - sekwencja danych wstawianych do indeksu:
 - 8, 5, 1, 7, 3, 12, 9, 6
- Wstawienie 8 i 5 daje liść

Zarządzanie strukturą indeksów B⁺ - drzewo:
wstawianie, usuwanie, modyfikowanie wartości
atrybutu indeksowego

Każdy węzeł posiada minimalnie 2 i maksymalnie 3
wskaźniki

Liść posiada od 1 do 2 wartości atrybutu
indeksowego

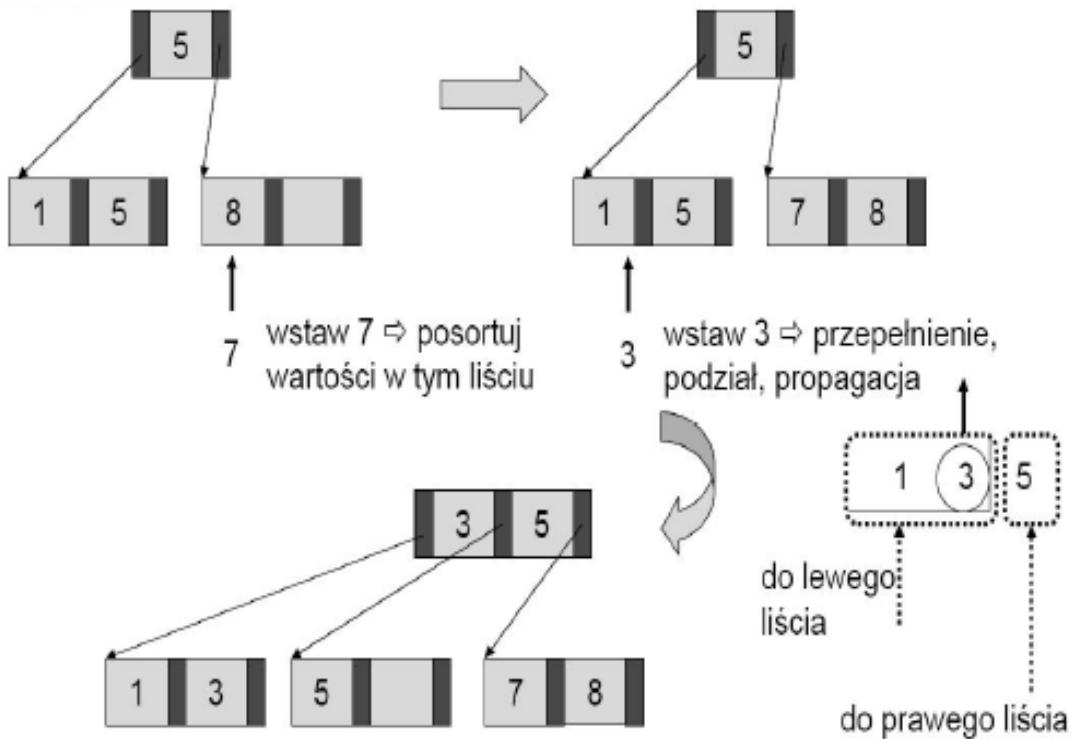




wstawianie danych do indeksu – przykład (2)

sekwencja danych wstawianych do indeksu:

- 8, 5, 1, 7, 3, 12, 9, 6



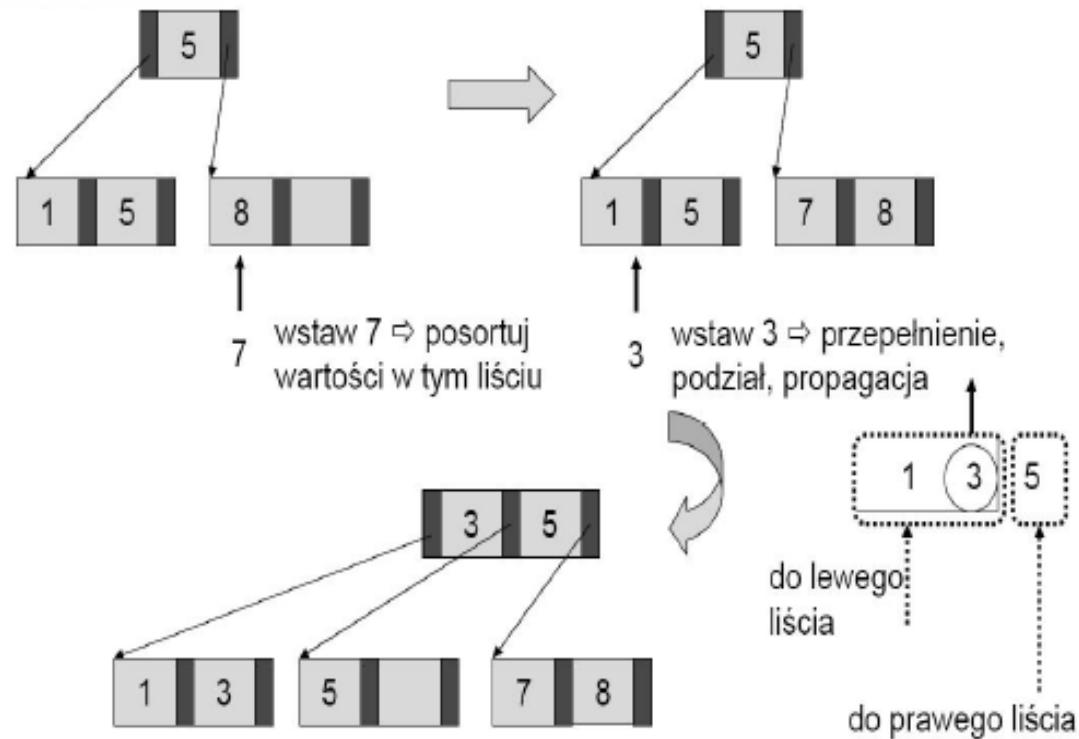
Wartość 7 musi być wstawiona do prawego liścia, który posiada miejsce na jedną wartość.



wstawianie danych do indeksu – przykład (2)

sekwencja danych wstawianych do indeksu:

- 8, 5, 1, 7, 3, 12, 9, 6



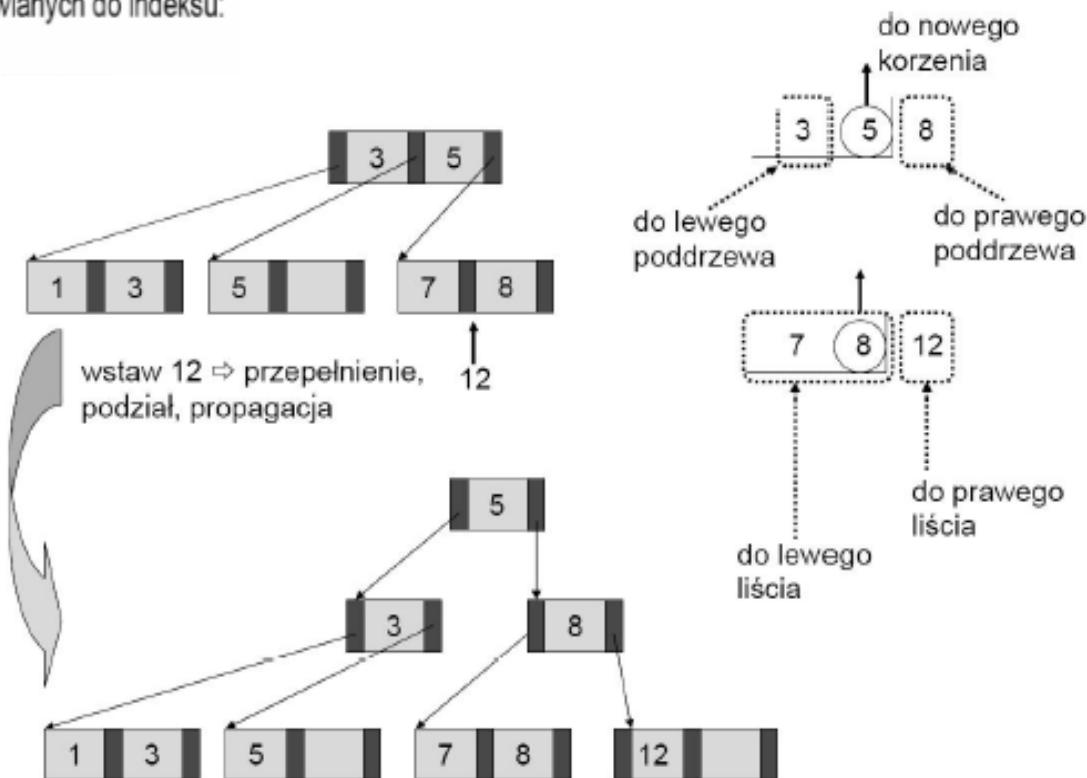
Wartość 3 musi być wstawiona do lewego liścia.



wstawianie danych do indeksu – przykład (3)

sekwencja danych wstawianych do indeksu:

- 8, 5, 1, 7, 3, 12, 9, 6



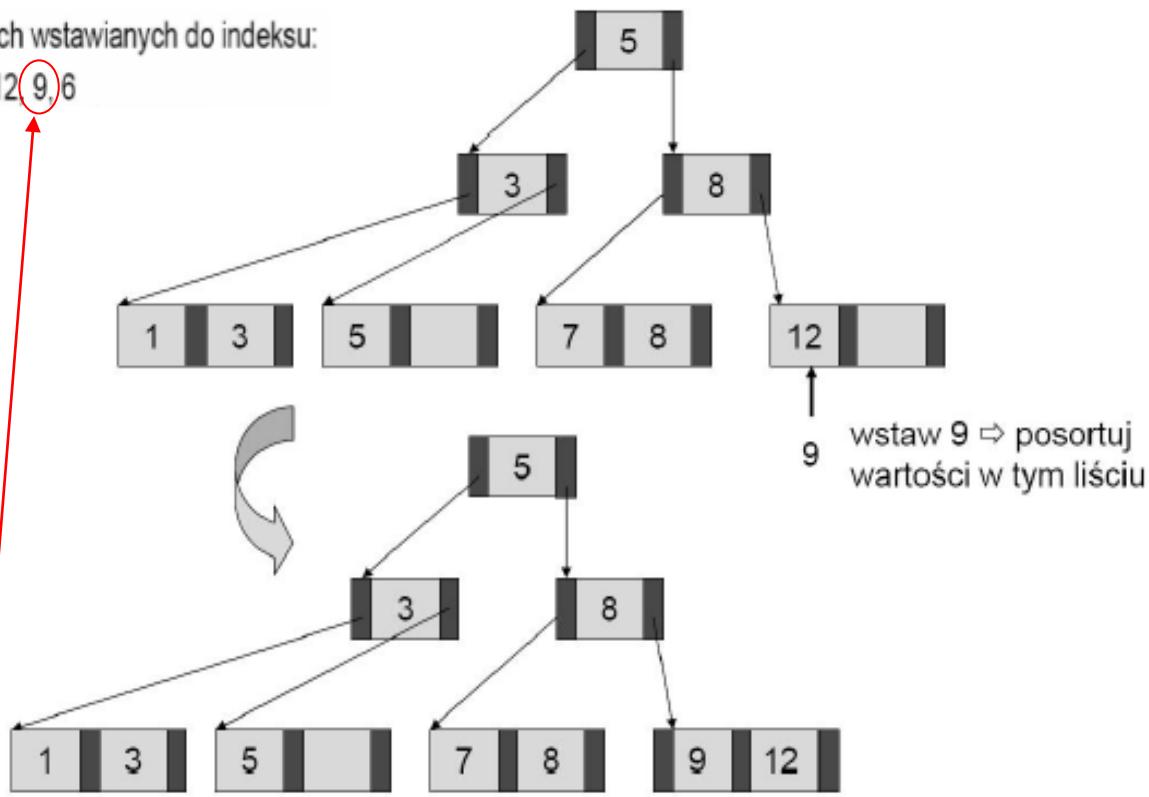
Wartość 12 musi być wstawiona do skrajnie prawego liścia.



wstawianie danych do indeksu – przykład (4)

sekwencja danych wstawianych do indeksu:

- 8, 5, 1, 7, 3, 12, 9, 6



9 wstaw 9 \Rightarrow posortuj
wartości w tym liściu

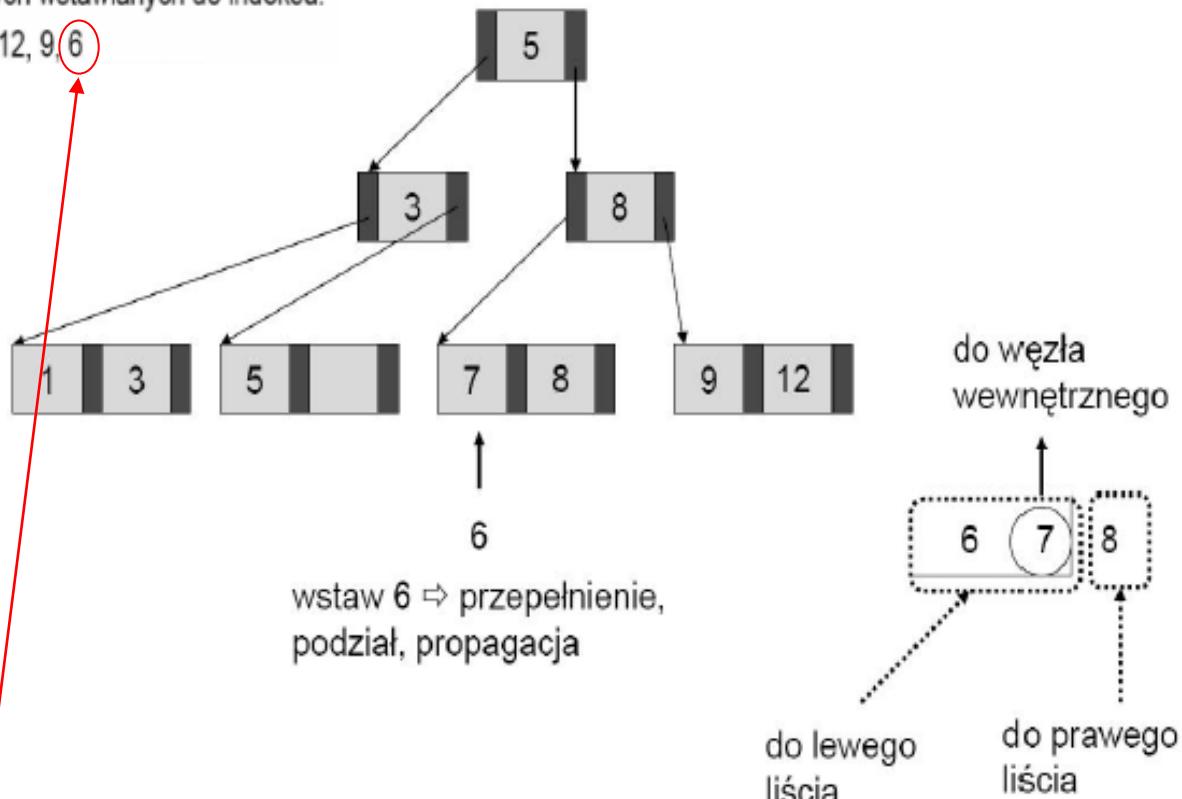
Wartość 9 musi być wstawiona do skrajnie prawego liścia, który posiada miejsce na jedną wartość.
Wartości w tym liściu należy posortować.



wstawianie danych do indeksu – przykład (5)

sekwencja danych wstawianych do indeksu:

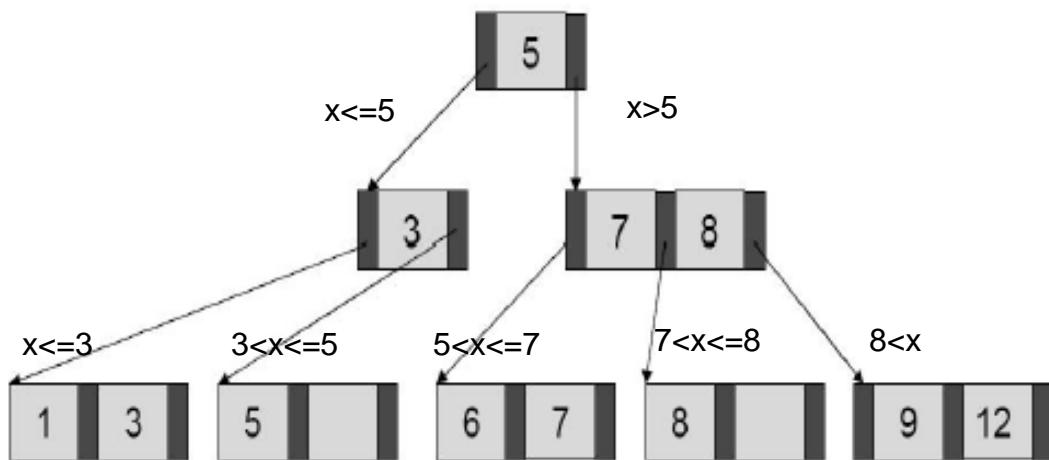
- 8, 5, 1, 7, 3, 12, 9, 6



Wartość 6 musi być wstawiona do trzeciego od lewej liścia.



wstawianie danych do indeksu – przykład (6)



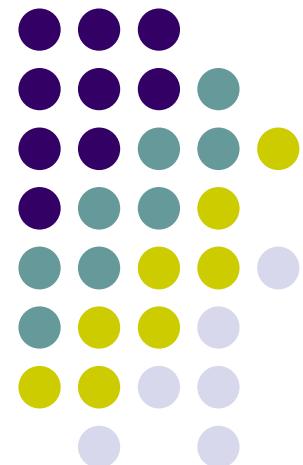
Ostateczna struktura indeksu jest przedstawiona powyżej



Koniec wykładu

wykład

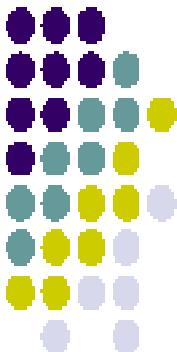
Organizacja plików





Plan wykładu

- Struktura przechowywania danych i organizacja rekordów w blokach
- Rodzaje organizacji plików
 - pliki nieuporządkowane
 - pliki uporządkowane
 - pliki haszowe



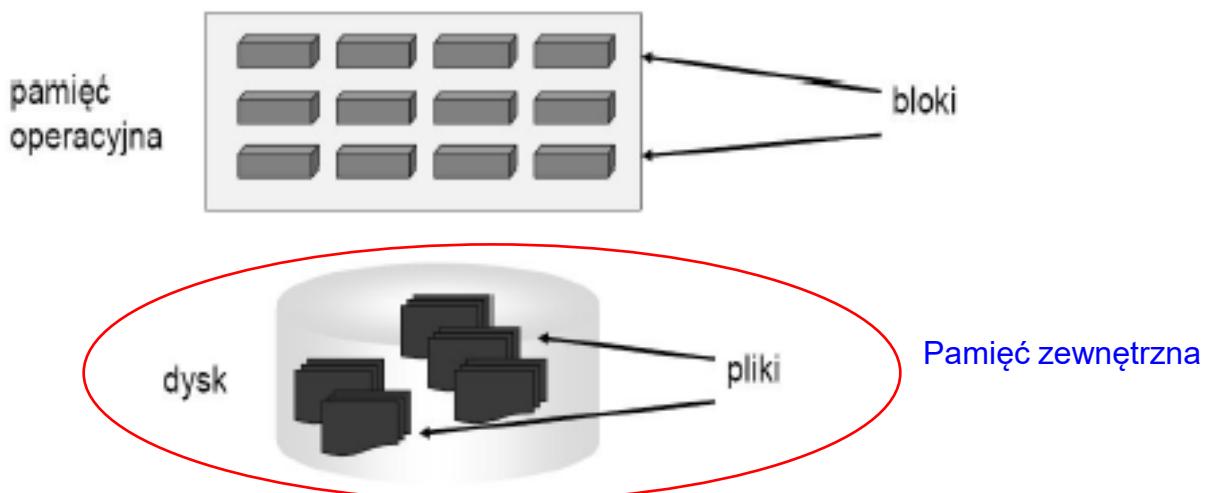
Wprowadzenie (1)

- Organizacja pliku
 - sposób uporządkowania rekordów w pliku przechowywanym na dysku
 - wspiera wykonywanie operacji na pliku
- Wybór odpowiedniej organizacji zależy od sposobu użytkowania danego pliku
 - wyszukiwanie rekordów opisujących zatrudnionych pracowników w porządku alfabetycznym ⇒ sortowanie pliku według nazwisk
 - wyszukiwanie rekordów opisujących zatrudnionych, których zarobki są w podanym zakresie ⇒ sortowanie nie jest właściwe
 - wybór odpowiedniej organizacji dla każdego pliku ⇒ administrator



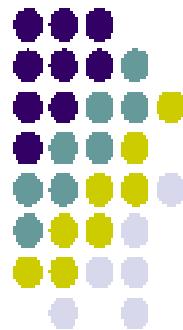
Wprowadzenie (2)

- Media fizyczne tworzą hierarchię pamięci składającą się z:
 - pamięci operacyjnej o organizacji blokowej
 - pamięci zewnętrznej o organizacji plikowej



Pamięć zewnętrzna → organizacja plikowa.

Pamięć operacyjna → organizacja blokowa.



Wprowadzenie (3)

- Trwałe dane w BD są przechowywane w pamięci zewnętrznej:
 - ze względu na rozmiar danych
 - odporność pamięci zewnętrznej na awarie
 - niski koszt przechowywania
- Buforowanie bloków dyskowych
- Alokacja danych w plikach \Rightarrow rekordy
 - rekordy proste / złożone
 - rekordy o stałej / zmiennej długości
- Na poziomie dyskowym rekordy przechowywane w blokach dyskowych

Bufory bazy danych.

STRUKTURA PROSTA REKORDU: wartością każdego pola rekordu jest wartość elementarna (liczba, łańcuch znaków, data, ciąg bitów).

STRUKTURA ZŁOŻONA REKORDU: wartością pola rekordu jest inny rekord. Rekordy mogą mieć stałą długość lub zmienną.

Stała długość rekordu długość oznacza, że rekord zawsze zajmuje tyle samo miejsca na dysku, niezależnie od rzeczywistych rozmiarów przechowywanych w nim danych.

Zmienna długość rekordu: rekordy o zmiennej długości przyjmują taki rozmiar jaki faktycznie przyjmują przechowywane w nich dane.



Współczynnik blokowania

- W bloku dyskowym lub bloku pamięci operacyjnej mieści się niecałkowita liczba rekordów
- Współczynnik blokowania \Rightarrow bfr (blocking factor)
 - rozmiar bloku B
 - rozmiar rekordu R

Współczynnik blokowania: $b_{fr} = \frac{B}{R}$

1

Wolny obszar w bloku: $f_{space} = B - (b_{fr} \cdot R)$

2

Współczynnik blokowania bfr jest określany jako największa liczba całkowita mniejsza od ilorazu rozmiaru bloku i rozmiaru rekordu.

Wolny obszar, jaki pozostaje w każdym z bloków, jest wyrażony wzorem 2:

B - (bfr * R).

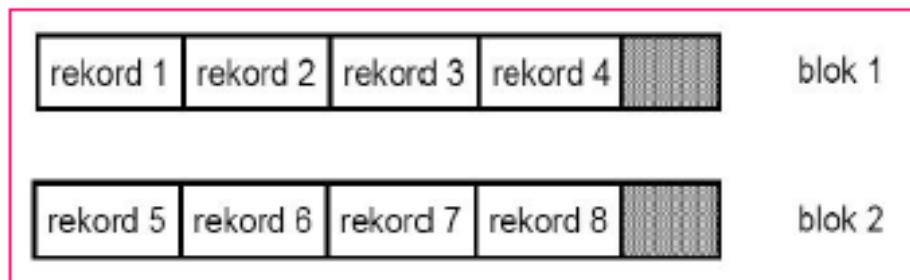
W przypadku rekordów o zmiennej długości jest to obszar minimalny.

Organizacje rekordów w blokach



*organizacja dzielona
(spanned)*

1



*organizacje niepodzielna
(unspanned)*

2

Rekordy w blokach: **dzielone** (ang. *spanned*) oraz **niepodzielne** (ang. *unspanned*).

Rekord dzielony: który cały nie mieści się w bloku jest dzielony, a jego części są przechowywane w kilku blokach.

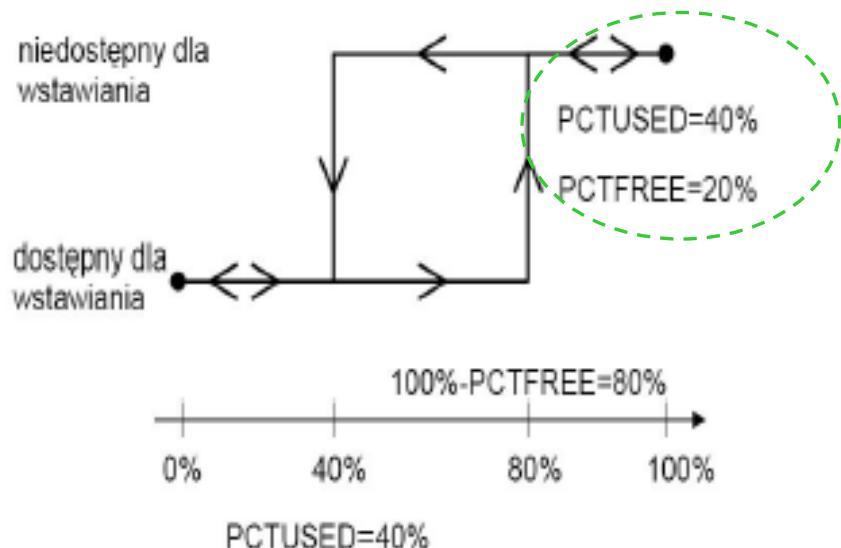
Rekord niepodzielny: rekord, który nie mieści się w całości w bloku jest przenoszony do takiego bloku, w którym zmieści się cały.

Organizacja niepodzielna powoduje, że w blokach pozostaje niewykorzystane miejsce.



zarządzanie rozmiarem bloku danych

Utrzymywanie wolnej pamięci w bloku dla potencjalnych modyfikacji



Definicja parametrów: **PCTFREE** oraz **PCTUSED**.

PCTFREE: określa ile procent rozmiaru bloku pozostanie wolne.

PCTUSED: określa, kiedy do bloku można wstawić rekordy. Parametr ten jest wyrażony również procentowym rozmiarem bloku. Jeśli zajętość bloku przewyższa wartość PCTUSED, wówczas blok nie jest wykorzystywany do wstawiania nowych rekordów. Jeśli natomiast zajętość bloku jest niższa niż PCTUSED, wówczas do bloku można wstawić rekordy.



operacje na plikach

- Operacje dostępu do pojedynczego rekordu (record-at-a-time)
 - wyszukiwanie: Find, FindFirst, FindNext
 - usuwanie: Delete
 - aktualizacja: Update
 - wstawianie: Insert
- Operacje dostępu do zbioru rekordów (set-at-a-time)
 - wyszukiwanie wszystkich rekordów spełniających kryterium: FindAll
 - wyszukiwanie z sortowaniem: FindOrdered
 - reorganizacja pliku: Reorganize

Ponieważ rekordy są fizycznie przechowywane w plikach, więc dostęp do rekordów musi być realizowany za pomocą dedykowanych i zoptymalizowanych operacji.

Wyróżnia się operacje na **pojedynczym rekordzie** (ang. record-at-a-time) i operacje na **zbiorze rekordów** (ang. set-at-a-time).



model kosztów

Model kosztów

- Notacja i założenia:
 - $N \Rightarrow$ liczba bloków
 - każdy blok zawiera R rekordów
 - średni czas odczytu/zapisu bloku dyskowego wynosi D
 - średni czas przetwarzania rekordu wynosi C
 - dla plików haszowych czas obliczenia wartości funkcji haszowej wynosi H
- Typowe wartości wynoszą:
 - $D = 15 \text{ ms}$
 - C i H od 1 do $10 \mu\text{s}$
 - dominuje koszt I/O

Operacja na pliku posiada swój tzw. **koszt**, który jest zależny od organizacji wewnętrznej pliku. Koszt jest konkretną wartością, której miarą może być np. czas wykonania, liczba dostępów do dysku.

Koszt jest wartością wynikającą z tzw. **modelu kosztów** (ang. *cost model*).



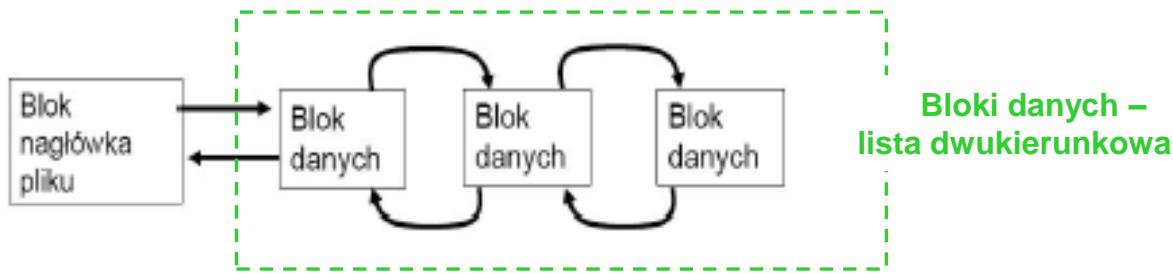
rodzaje organizacji plików

- 1 • Pliki nieuporządkowane (unordered files, heap files)
- 2 • Pliki uporządkowane (ordered files)
- 3 • Pliki haszowe (hash files)



pliki nieuporządkowane

- Nagłówek pliku zawierający wskaźnik do bloku danych
- Blok danych zawiera wskaźnik do bloku następnego i poprzedniego
- Rekordy wstawiane na koniec pliku



Charakterystyka

- Efektywne wstawianie pojedynczych rekordów i dużych zbiorów rekordów
- Efektywne pozostałe operacje w przypadku plików o rozmiarze kilku bloków
- Struktura właściwa dla odczytu wszystkich rekordów
- Struktura stosowana z innymi strukturami dostępu do danych (np. indeksy)

Plik nieuporządkowany: rekordy są przechowywane w kolejności ich wstawiania. Nagłówek pliku zawiera: [wskaźnik do pierwszego bloku danych](#).

[Bloki danych](#) tworzą listę dwukierunkową.

pliki nieuporządkowane – operacje (1)

1



- Podstawową organizacją pliku nieuporządkowanego jest stóg (ang. heap)
- Operacje:
 - wstawianie rekordu
 - rekord jest wstawiany do ostatniego bloku pliku; blok ten jest zapisywany na dysk
 - koszt = $2D+C$
 - wyszukiwanie rekordu
 - konieczność liniowego przeszukiwania wszystkich bloków
 - średni koszt = $0.5N(D+RC)$
 - maksymalny koszt = $N(D+RC)$ (przejrzenie całego pliku)

1

2

pliki nieuporządkowane – operacje (2)



1

– przeglądanie pliku

- koszt = $N(D + RC)$

1

- odczyt N stron z kosztem D

- dla każdej strony przetworzyć R rekordów z kosztem C

– wyszukiwanie z przedziałem wartości

- odczyt wszystkich bloków

- koszt = $N(D + RC)$

2

– usuwanie rekordu

- liniowe przeszukiwanie i zapis bloku na dysk

- koszt wyszukania + koszt ($C + D$)

- średnio: $0.5N(D+RC)+(C+D)$

3

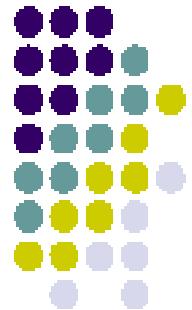
- maksymalnie: $N(D+RC)+(C+D)$

4

- pozostaje zwolniona pamięć \Rightarrow konieczność okresowej reorganizacji pliku

pliki nieuporządkowane – operacje (3)

1



– sortowanie

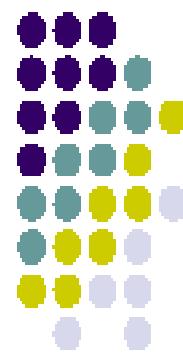
- trudne
- stosowane tzw. sortowanie zewnętrzne
 - plik sortowany fragmentami mieszczącymi się w pamięci operacyjnej
 - fragmenty są łączone w większe w kolejnych przebiegach algorytmu sortowania

Sortowanie → wykonywane w pamięci operacyjnej. Techniką sortowania stosową najczęściej

Sortowanie zewnętrzne (ang. *external sorting*) → sortowanie pliku fragmentami, które mieszczą się w pamięci operacyjnej. Każdy posortowany fragment jest w drugiej fazie sortowania łączony z innymi fragmentami. Łączenie to wymaga zwykle wielu przebiegów.

pliki nieuporządkowane – charakterystyka

1



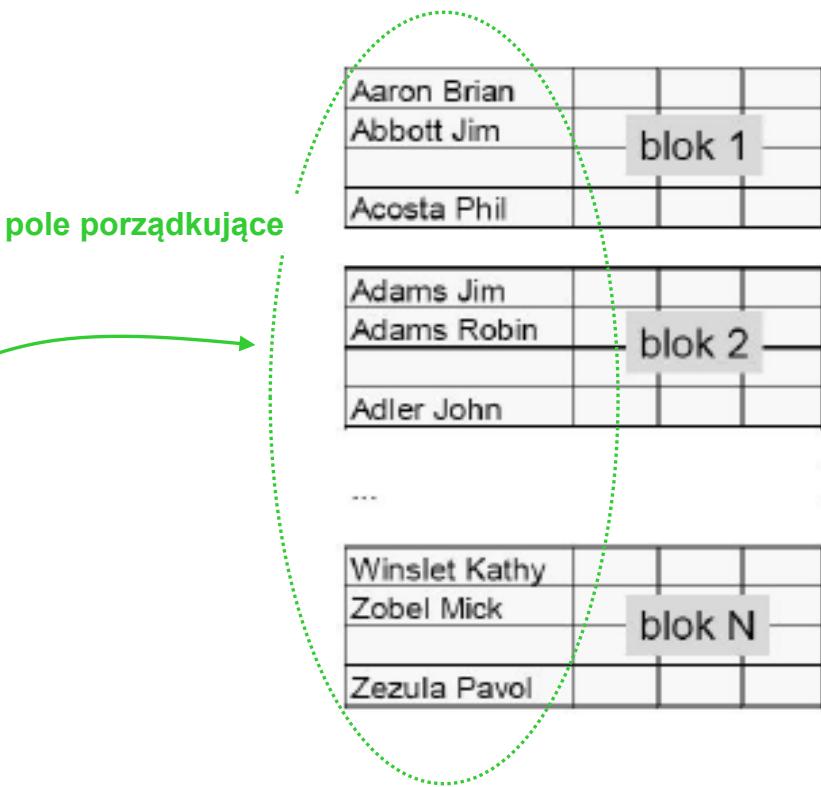
- Efektywne wstawianie pojedynczych rekordów i dużych zbiorów rekordów
- Efektywne pozostałe operacje w przypadku plików o rozmiarze kilku bloków
- Struktura właściwa dla odczytu wszystkich rekordów
- Struktura stosowana z innymi strukturami dostępu do danych (np. indeksy)



2

pliki uporządkowane

Struktura organizacji plików uporządkowanych



W pliku uporządkowanym rekordy składowane są uporządkowane wg wartości tzw. pola porządkującego (ang. *ordering field*).

W przykładowym pliku ze slajdu, rekordy zostały uporządkowane wg wartości **nazwiska i imienia**.

pliki uporządkowane – operacje (1)

2



Operacje:

– przeglądanie pliku

- koszt = $N(D+RC)$ 1

- ponieważ wszystkie strony muszą być odczytane

– wyszukiwanie rekordu

- wyszukiwanie binarne

- $\text{koszt} = D \cdot \log_2 B + C \cdot \log_2 R$ 2

– wyszukiwanie z przedziałem wartości

- koszt wyszukania pierwszego rekordu + koszt selekcji zbioru rekordów

$$\text{koszt} = D \cdot \log_2 B + C \cdot \log_2 R + ND \quad 3$$



Operacje:

– wstawianie rekordu

- koszt wyszukania miejsca wstawienia rekordu
- koszt przepisania średnio połowy rekordów
- koszt całkowity = $2*(0.5N(D+RC))$ 1

– usuwanie rekordu

- koszt znalezienia usuwanego rekordu
- koszt przepisania średnio połowy rekordów
- identyczny jak koszt wstawiania
- koszt całkowity = $2*(0.5N(D+RC))$ 2



plik uporządkowany

- rozwiązanie problemu wstawiania rekordów

- Przesuwanie średnio połowy rekordów \Rightarrow nieefektywne
- Pozostawienie wolnej pamięci w każdym bloku na wstawiane rekordy
 - wstawienie rekordu wymaga przesunięcia rekordów tylko w ramach bieżącego bloku
- Tworzenie nieuporządkowanego pliku rekordów, a następnie łączenie go z plikiem głównym
 - efektywne wstawianie
 - nieefektywne wyszukiwanie

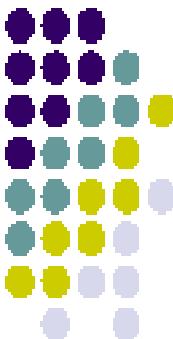
Pierwsze rozwiązanie: przesuwanie średnio połowy rekordów w pliku, w celu zagwarantowania właściwego porządku rekordów w pliku.

Drugie rozwiązanie: pozostawienie w każdym bloku dyskowym pustego miejsca na wstawiane rekordy. W tym przypadku przesunięcia rekordów należy dokonać tylko w tym bloku, do którego jest wstawiany rekord.

Trzecie rozwiązanie: wykorzystanie nieuporządkowanego tymczasowego pliku, tzw.: nadmiarowego (ang. *overflow file*) lub transakcyjnego (ang. *transaction file*).

plik uporządkowany

- modyfikowanie rekordu



PROCEDURA REALIZACJI

- Znalezienie i odczyt modyfikowanego rekordu do bufora
 - połowienie binarne dla warunku na polu porządkującym
 - przeszukanie całego pliku dla innego warunku
- Modyfikowanie wartości atrybutu nieporządkującego
 - zmodyfikowanie w buforze i zapis na dysk w to samo miejsce
- Modyfikowanie wartości atrybutu porządkującego
 - zmiana pozycji rekordu w pliku
 - usunięcie rekordu + wstawienie nowego

Modyfikowanie rekordu wymaga jego znalezienia w pliku i odczytania do bufora.

plik uporządkowany – zalety i wady

2



ZALETY:

- Efektywny odczyt rekordów w kolejności pola porządkującego \Rightarrow posortowane
- Znalezienie następnego rekordu, według określonego porządku, jest bardzo proste
- Metoda połowienia binarnego do wyszukiwania rekordu z warunkiem opartym o pole porządkujące

WADY:

- Uporządkowanie pliku jest nieprzydatne, gdy wyszukiwanie jest realizowane według wartości pola nie porządkującego pliku danych,
- Kosztowne wstawianie i usuwanie rekordów ze względu na konieczność zachowania porządku w pliku

Zalety: Operacja odczytu i sortowania \rightarrow efektywna, znalezienie następnego rekordu, według określonego porządku \rightarrow bardzo proste.

Wady: Uporządkowanie pliku jest nieprzydatne, gdy wyszukiwanie jest realizowane według wartości pola nie porządkującego pliku danych.



Pliki nieuporządkowane

□ Wstawianie rekordu

➤ $K = 2D + C = 2 \cdot 15 \cdot 10^{-3} + 5 \cdot 10^{-6} = ??? \text{ s}$

□ Wyszukiwanie rekordu

➤ $K_{sr} = 0,5 \cdot N(D + RC) = 0,5 \cdot 3000(15 \cdot 10^{-3} + 30000 \cdot 5 \cdot 10^{-6}) = ??? \text{ s}$

➤ $K_{max} = N(D + RC) = 3000(15 \cdot 10^{-3} + 30000 \cdot 5 \cdot 10^{-6}) = ??? \text{ s}$

□ Przeglądanie pliku

➤ $K = N(D + RC) = 3000(15 \cdot 10^{-3} + 30000 \cdot 5 \cdot 10^{-6}) = ??? \text{ s}$

□ Wyszukiwanie z przedziałem wartości

➤ $K = N(D + RC) = 3000(15 \cdot 10^{-3} + 30000 \cdot 5 \cdot 10^{-6}) = ??? \text{ s}$

□ Usuwanie rekordu

➤ $K_{sr} = 0,5 \cdot N(D + RC) + (C + D) = 0,5 \cdot 3000(15 \cdot 10^{-3} + 30000 \cdot 5 \cdot 10^{-6}) + (15 \cdot 10^{-3} + 5 \cdot 10^{-6}) = 0,5 \cdot 495 \approx ??? \text{ s}$

➤ $K_{max} = N(D + RC) + (C + D) = 3000(15 \cdot 10^{-3} + 30000 \cdot 5 \cdot 10^{-6}) + (15 \cdot 10^{-3} + 5 \cdot 10^{-6}) \approx ??? \text{ s}$



Pliki uporządkowane

□ Wstawianie rekordu

$$\begin{aligned} \succ K &= 2 \cdot (0,5 \cdot N(D + RC)) = 2 \cdot (0,5 \cdot 3000(15 \cdot 10^{-3} + 30000 \cdot 5 \cdot 10^{-6})) = 2 \cdot 0,5 \cdot 495 = ??? \text{ s} \end{aligned}$$

□ Wyszukiwanie rekordu

$$\begin{aligned} \succ K &= D \cdot \log_2 B + C \cdot \log_2 R = 15 \cdot 10^{-3} \cdot \log_2 1024 + 5 \cdot 10^{-6} \cdot \log_2 30000 \approx ??? \text{ s} \end{aligned}$$

□ Przeglądanie pliku

$$\begin{aligned} \succ K &= N(D + RC) = 3000(15 \cdot 10^{-3} + 30000 \cdot 5 \cdot 10^{-6}) = ??? \text{ s} \end{aligned}$$

□ Wyszukiwanie rekordu z przedziałem wartości

$$\begin{aligned} \succ K &= D \cdot \log_2 B + C \cdot \log_2 R + ND = 15 \cdot 10^{-3} \cdot \log_2 1024 + 5 \cdot 10^{-6} \cdot \log_2 30000 + 3000 \cdot 15 \cdot 10^{-3} \approx ??? \text{ s} \end{aligned}$$

□ Usuwanie rekordu

$$\begin{aligned} \succ K &= 2 \cdot (0,5 \cdot N(D + RC)) = 2 \cdot (0,5 \cdot 3000(15 \cdot 10^{-3} + 30000 \cdot 5 \cdot 10^{-6})) = 2 \cdot 0,5 \cdot 495 = ??? \text{ s} \end{aligned}$$

Pliki uporządkowane



Pliki nieuporządkowane

□ Wstawianie rekordu

$$\begin{aligned} \gg K &= 2 \cdot 0,5 \cdot N(D + RC) = 2 \cdot 0,5 \cdot \\ &3000(15 \cdot 10^{-3} + 30000 \cdot 5 \cdot 10^{-6}) = \\ &2 \cdot 0,5 \cdot 495 = ??? \text{ s} \end{aligned}$$

□ Wyszukiwanie rekordu

$$\begin{aligned} \gg K &= D \cdot \log_2 B + C \cdot \log_2 R = 15 \cdot 10^{-3} \cdot \\ &\log_2 1024 + 5 \cdot 10^{-6} \cdot \log_2 30000 \approx \\ &??? \text{ s} \end{aligned}$$

□ Przeglądanie pliku

$$\begin{aligned} \gg K &= N(D + RC) = 3000(15 \cdot 10^{-3} + \\ &30000 \cdot 5 \cdot 10^{-6}) = ??? \text{ s} \end{aligned}$$

□ Wyszukiwanie rekordu z przedziałem wartości

$$\begin{aligned} \gg K &= D \cdot \log_2 B + C \cdot \log_2 R + ND = 15 \cdot \\ &10^{-3} \cdot \log_2 1024 + 5 \cdot 10^{-6} \cdot \\ &\log_2 30000 + 3000 \cdot 15 \cdot 10^{-3} \approx ??? \text{ s} \end{aligned}$$

□ Usuwanie rekordu

$$\begin{aligned} \gg K &= 2 \cdot 0,5 \cdot N(D + RC) = 2 \cdot 0,5 \cdot \\ &3000(15 \cdot 10^{-3} + 30000 \cdot 5 \cdot 10^{-6}) = \\ &2 \cdot 0,5 \cdot 495 = ??? \text{ s} \end{aligned}$$

□ Wstawianie rekordu

$$\begin{aligned} \gg K &= 2D + C = 2 \cdot 15 \cdot 10^{-3} + 5 \cdot 10^{-6} \\ &= ??? \text{ s} \end{aligned}$$

□ Wyszukiwanie rekordu

$$\begin{aligned} \gg K_{\text{śr}} &= 0,5 \cdot N(D + RC) = 0,5 \cdot 3000(15 \cdot 10^{-3} + \\ &30000 \cdot 5 \cdot 10^{-6}) = ??? \text{ s} \\ \gg K_{\text{max}} &= N(D + RC) = 3000(15 \cdot 10^{-3} + 30000 \cdot 5 \cdot \\ &10^{-6}) = ??? \text{ s} \end{aligned}$$

□ Przeglądanie pliku

$$\begin{aligned} \gg K &= N(D + RC) = 3000(15 \cdot 10^{-3} + 30000 \cdot 5 \cdot \\ &10^{-6}) = ??? \text{ s} \end{aligned}$$

□ Wyszukiwanie z przedziałem wartości

$$\begin{aligned} \gg K &= N(D + RC) = 3000(15 \cdot 10^{-3} + 30000 \cdot 5 \cdot \\ &10^{-6}) = ??? \text{ s} \end{aligned}$$

□ Usuwanie rekordu

$$\begin{aligned} \gg K_{\text{śr}} &= 0,5 \cdot N(D + RC) + (C + D) = 0,5 \cdot 3000(15 \cdot 10^{-3} + \\ &30000 \cdot 5 \cdot 10^{-6}) + (15 \cdot 10^{-3} + 5 \cdot 10^{-6}) \approx ??? \text{ s} \\ \gg K_{\text{max}} &= N(D + RC) + (C + D) = 3000(15 \cdot 10^{-3} + 30000 \cdot \\ &5 \cdot 10^{-6}) + (15 \cdot 10^{-3} + 5 \cdot 10^{-6}) \approx ??? \text{ s} \end{aligned}$$



plik haszowy

Plik o strukturze wykorzystującej technikę haszowania nazywa się plikiem haszowym (bezpośrednim)

- Porządek rekordów w pliku określony na podstawie tzw. pola haszowego
- Koncepcja
 - zdefiniowanie funkcji haszowej (ang. hash function) z argumentem, którym jest wartość pola haszowego
 - wynikiem funkcji haszowej jest adres bloku dyskowego, w którym powinien znaleźć się dany rekord
- Haszowanie
 - wewnętrzne
 - zewnętrzne

Koncepcja organizacji tego pliku polega na zdefiniowaniu **funkcji haszowej** (ang. *hash function*). W praktyce stosuje się dwie techniki haszowania, tj. haszowanie wewnętrzne i haszowanie zewnętrzne.



3

haszowanie wewnętrzne

Dana jest tablica rekordów o M szczelinach, których adresy odpowiadają indeksom tablicy haszowej.

Indeks tablicy: 0 \Rightarrow M-1

Funkcja haszowa

$H(K=\text{wartość pola haszowego}) \rightarrow \{0, \dots, M-1\}$

Najczęściej spotykana funkcja haszowa

$H(K)=K \bmod M$

indeksy tablicy
haszowej

	Id_prac	Nazwisko	Etat	Płaca
0	450	Nowak	kierownik	2000
1	120	Kuzio	portier	1000
2	220	Misiek	z-ca kier.	1800
M-2	100	Dziubek	dyrektor	5000
M-1	110	Gulczas	portier	800

Koncepcja haszowania wewnętrznego

Dana jest tablica rekordów o M szczelinach. Adresy szczelin odpowiadają indeksom tablicy haszowej. Indeksy te przyjmują wartości od 0 do M-1.

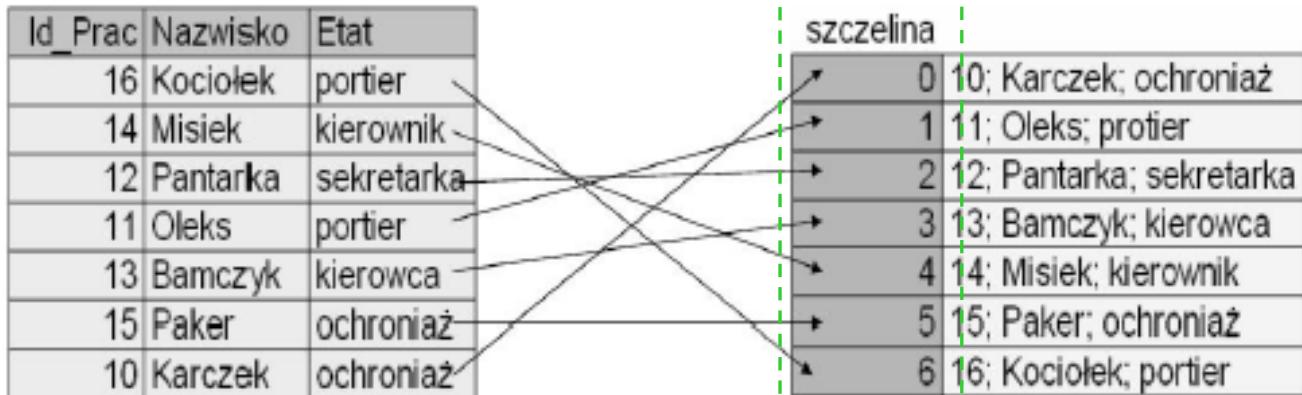
Funkcja haszowa ma postać: $H(K=\text{wartość pola haszowego}) > \{0, \dots, M-1\}$.

Funkcja ta transformuje wartość pola haszowego do liczby całkowitej z zakresu od 0 do M-1.



3

haszowanie przykład



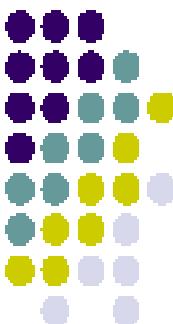
$$H(\text{Id_Prac}) = \text{Id_Prac} \bmod 10$$

Przykład haszowania wewnętrznego: rekordy pracowników z polami:

Id_Prac, Nazwisko, Etat.

Przyjmijmy tablicę haszową ze szczelinami o numerach od 0 do 6.

Definicja funkcji haszowej: **H(Id_Prac)=Id_Prac MOD 10**



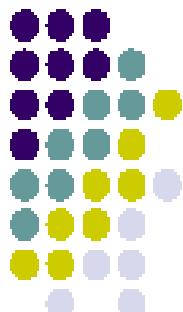
proces haszowania

3

Proces haszowania składa się z dwóch kroków:

1. transformacji (pole nienumeryczne są transformowane do liczb całkowitych)
2. haszowania (przykładowy algorytm transformacji)

```
temp := 1;  
for i = 1 to 20 do  
    temp := temp * code(K[i])  
hash_address := temp MOD M;
```



kolizja - alternatywa

3

- Kolizja \Rightarrow wartość funkcji haszowej dla danej wartości pola haszowego nowego rekordu odpowiada zajętemu już adresowi szczeliny
- Źródło kolizji \Rightarrow funkcja haszowa generująca te same adresy szczeliny dla różnych wartości atrybutu haszowego
- Rozwiązywanie kolizji \Rightarrow procedura znajdowania innej lokalizacji dla wprowadzanego rekordu

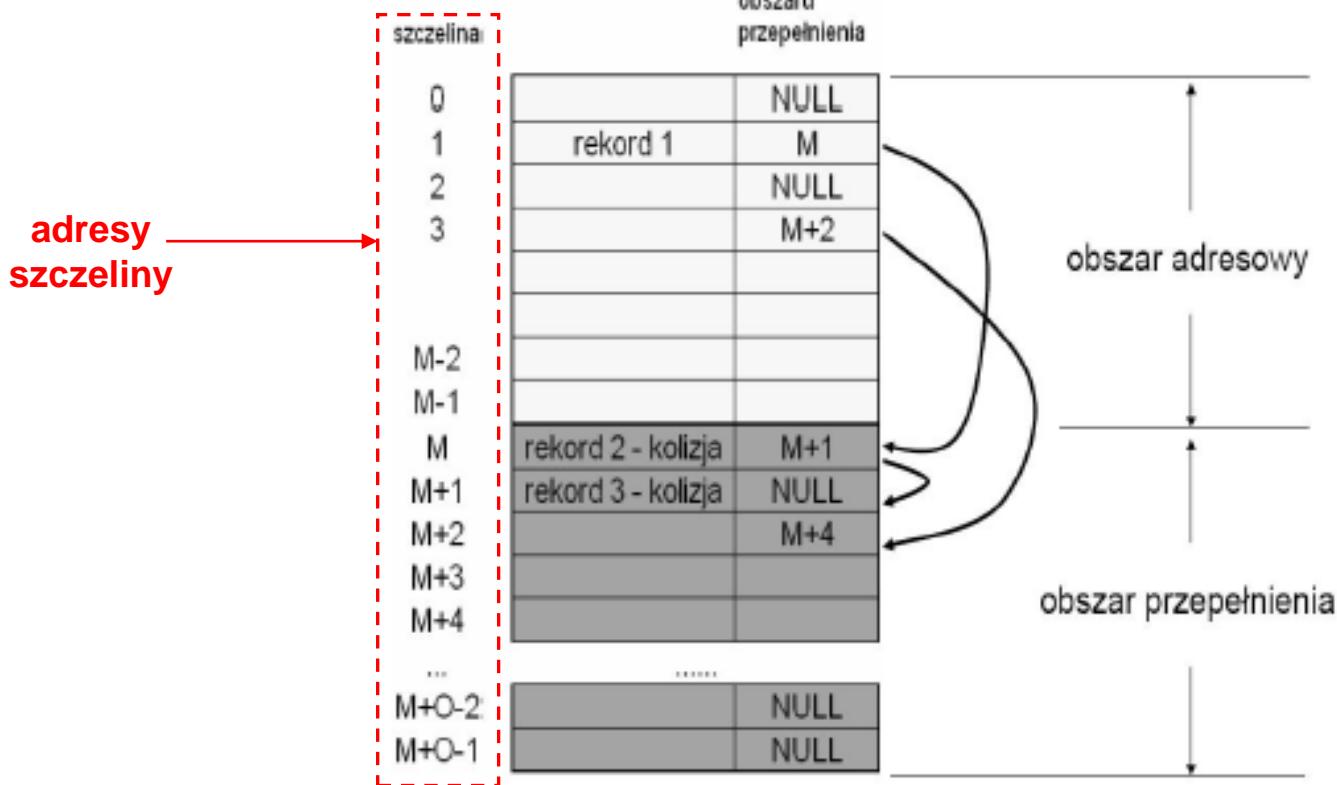
METODY ROZWIĄZYWANIA KOLIZJI

- Adresowanie otwarте (open addressing) – następna wolna lokalizacja
- Łącuchowanie (chaining) – wskaźniki do nowych lokalizacji
- Haszowanie wielokrotne (multiple hashing) – nowa funkcja haszowa



3

Łańcuchowanie



Technika łańcuchowania polega na przechowywaniu w szczelinie dodatkowo wskaźnika do tzw. obszaru przepełnienia (ang. *overflow space*). Służy on do przechowywania wszystkich rekordów ulegających kolizji w danej szczelinie. Rekordy w obszarze przepełnienia tworzą listę.

haszowanie zewnętrzne (1)

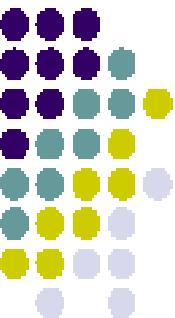
3



- Wartościami tablicy haszowej są adresy logicznych obszarów dyskowych (LOD) (block buckets)
- Liczba LOD jest stała i równa liczbie szczelin w tablicy haszowej \Rightarrow jest to tzw. haszowanie statyczne (static hashing)
- LOD jest albo pojedynczym blokiem dyskowym albo zbiorem kolejnych (leżących obok siebie) bloków dyskowych
- Funkcja haszowa odwzorowuje wartość atrybutu haszowego w numer LOD
- Plik dyskowy zawiera tablicę konwersji numerów LOD w fizyczne adresy bloków dyskowych

W haszowaniu zewnętrznym wartościami tablicy haszowej są adresy logicznych obszarów dyskowych (LOD) (ang. *block buckets*).

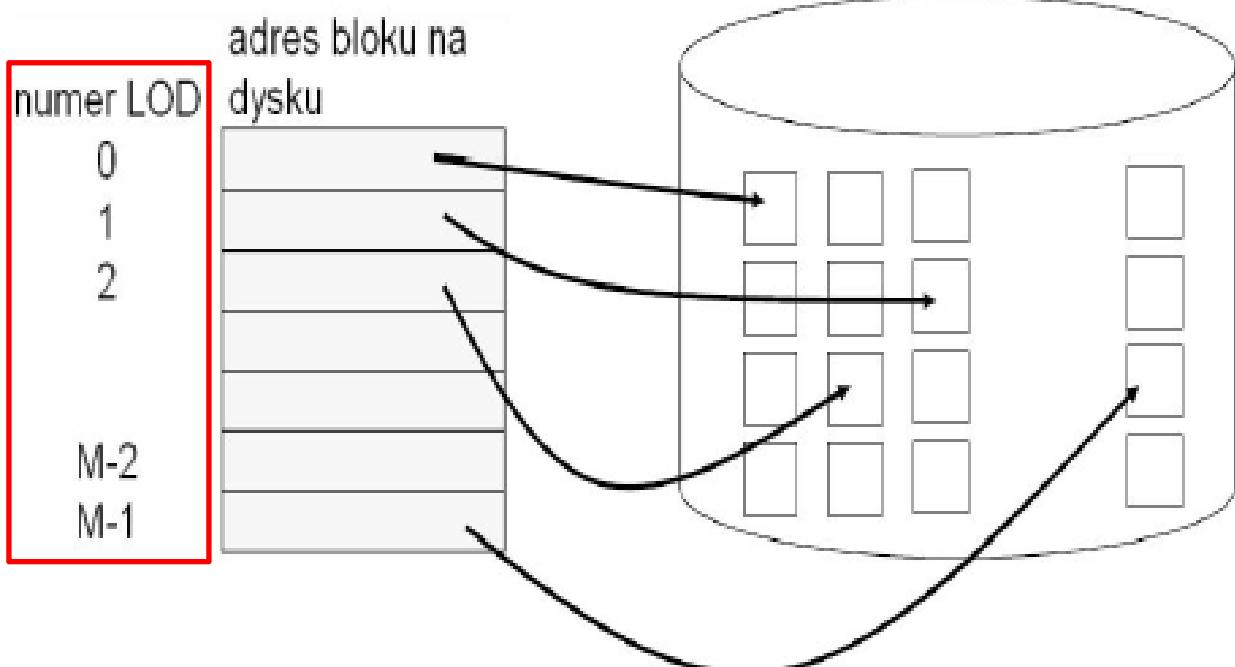
Funkcja haszowa odwzorowuje wartość atrybutu haszowego w numer LOD. Plik dyskowy zawiera tablicę konwersji numerów LOD w fizyczne adresy bloków dyskowych.



haszowanie zewnętrzne (2)

3

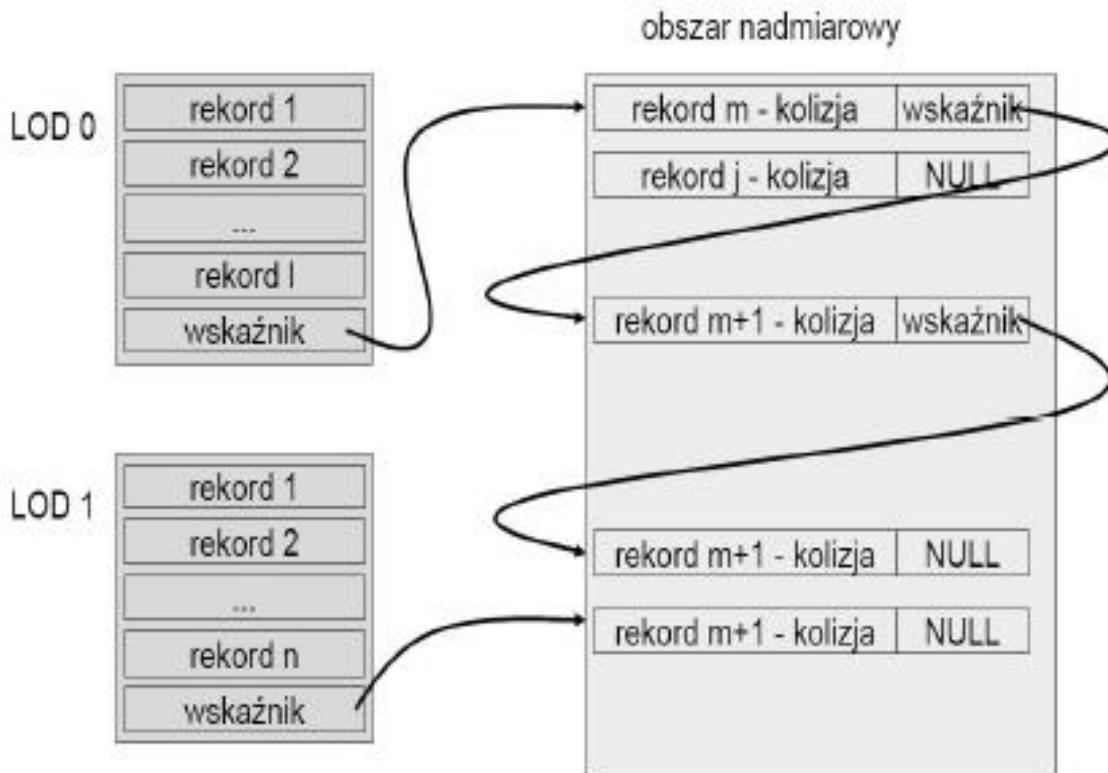
tablica konwersji





3

haszowanie zewnętrzne - kolizje



W haszowaniu zewnętrznym → kolizje rzadsze, (ten sam LOD, którego numer jest wynikiem działania funkcji haszowej, może pomieścić wiele rekordów).

Kolizja może jednak wystąpić po zapełnieniu się wszystkich bloków dyskowych wchodzących w skład danego LOD.



haszowanie zewnętrzne - operacje (1)

3

- Poszukiwanie rekordu z warunkiem nałożonym na pole inne niż haszowe
 - przeszukanie całego pliku i obszaru nadmiarowego
- Poszukiwanie rekordu z warunkiem nałożonym na pole haszowe
 - funkcja haszowa
 - w przypadku kolizji przeszukanie obszaru nadmiarowego
- Usunięcie rekordu z LOD
 - wyszukanie rekordu (funkcja haszowa + tablica konwersji) i zwolnienie szczeliny
 - przesunięcie pierwszego rekordu z obszaru nadmiarowego do LOD
- Usunięcie rekordu z obszaru nadmiarowego
 - wyszukanie rekordu (funkcja haszowa) + przeszukanie listy rekordów w obszarze nadmiarowym
 - zwolnienie szczeliny
 - utrzymywanie listy wolnych szczelin

Poszukiwanie w pliku haszowym rekordu z warunkiem nałożonym na pole inne niż haszowe wymaga przeszukania całego pliku i obszaru nadmiarowego.

Usunięcie rekordu z pliku haszowego przebiega w dwóch wariantach.



haszowanie zewnętrzne - operacje (1)

3

- Poszukiwanie rekordu z warunkiem nałożonym na pole inne niż haszowe
 - przeszukanie całego pliku i obszaru nadmiarowego
- Poszukiwanie rekordu z warunkiem nałożonym na pole haszowe
 - funkcja haszowa
 - w przypadku kolizji przeszukanie obszaru nadmiarowego
- Usunięcie rekordu z LOD
 - wyszukanie rekordu (funkcja haszowa + tablica konwersji) i zwolnienie szczeliny
 - przesunięcie pierwszego rekordu z obszaru nadmiarowego do LOD
- Usunięcie rekordu z obszaru nadmiarowego
 - wyszukanie rekordu (funkcja haszowa) + przeszukanie listy rekordów w obszarze nadmiarowym
 - zwolnienie szczeliny
 - utrzymywanie listy wolnych szczelin

W wariancie drugim, usuwany rekord znajduje się w obszarze nadmiarowym. Jego usunięcie polega na znalezieniu szczeliny (z wykorzystaniem funkcji haszowej i tablicy konwersji) w LOD.

haszowanie zewnętrzne - operacje (3)

3



- Wstawienie rekordu
 - odczyt adresu szczeliny (funkcja haszowa)
 - w przypadku kolizji zaalokowanie szczeliny w obszarze nadmiarowym
- Zmodyfikowanie wartości pola haszowego
 - odczytanie rekordu (funkcja haszowa)
 - rekord zmienia szczeliny
- Usunięcie rekordu + wstawienie rekordu z nową wartością
- Zmodyfikowanie wartości pola nie-haszowego
 - odczytanie rekordu (funkcja haszowa)
 - zmodyfikowanie wartości
 - rekord nie zmienia szczeliny

Wstawienie rekordu do pliku haszowego polega na odczytaniu adresu szczeliny z wykorzystaniem funkcji haszowej i zapisaniu rekordu do tej szczeliny.

Zmodyfikowanie wartości pola haszowego polega na odczytaniu rekordu z wykorzystaniem funkcji haszowej.

funkcja haszowa

3



- Cechą dobrej funkcji haszowej jest zapewne równomiernego rozkładu rekordów w obrębie przestrzeni adresowej tablicy haszowej
- Zalecany rozmiar tablicy haszowej
 $r/M \in (0,7 \div 0,9)$, $r \Rightarrow$ liczba rekordów, $M \Rightarrow$ liczba bloków

Charakterystyka

- Problem porządkowania pliku oraz wyszukiwania rekordów w porządku wartości pola haszowego
- Problem stałego rozmiaru przestrzeni adresowej przydzielonej plikowi – częste kolizje

Praktyka pokazuje, że tablica haszowa powinna być zajęta w 70% - 90%

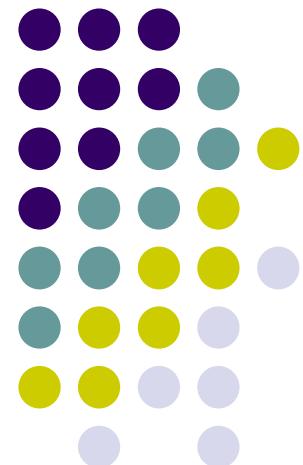
Pliki haszowe są nieefektywne dla operacji odczytu danych z ich porządkowaniem zgodnie z wartością pola haszowego.



KONIEC WYKŁADU

wykład

Przetwarzanie transakcyjne
Cz.1



Technologia baz danych (2)



2. Przetwarzanie transakcyjne (spójność bazy danych)

- dostęp do bazy danych za pomocą transakcji o własnościach ACID
- metody synchronizacji transakcji (2PL, znaczniki czasowe, wielowersyjność danych)
- metody odtwarzania spójności bazy danych (plik logu, odtwarzanie i wycofywanie operacji, punkty kontrolne)
- archiwizacja bazy danych i odtwarzanie po awarii



Plan wykładu

Celem niniejszego wykładu jest omówienie problematyki związanej z transakcjami w bazie danych.

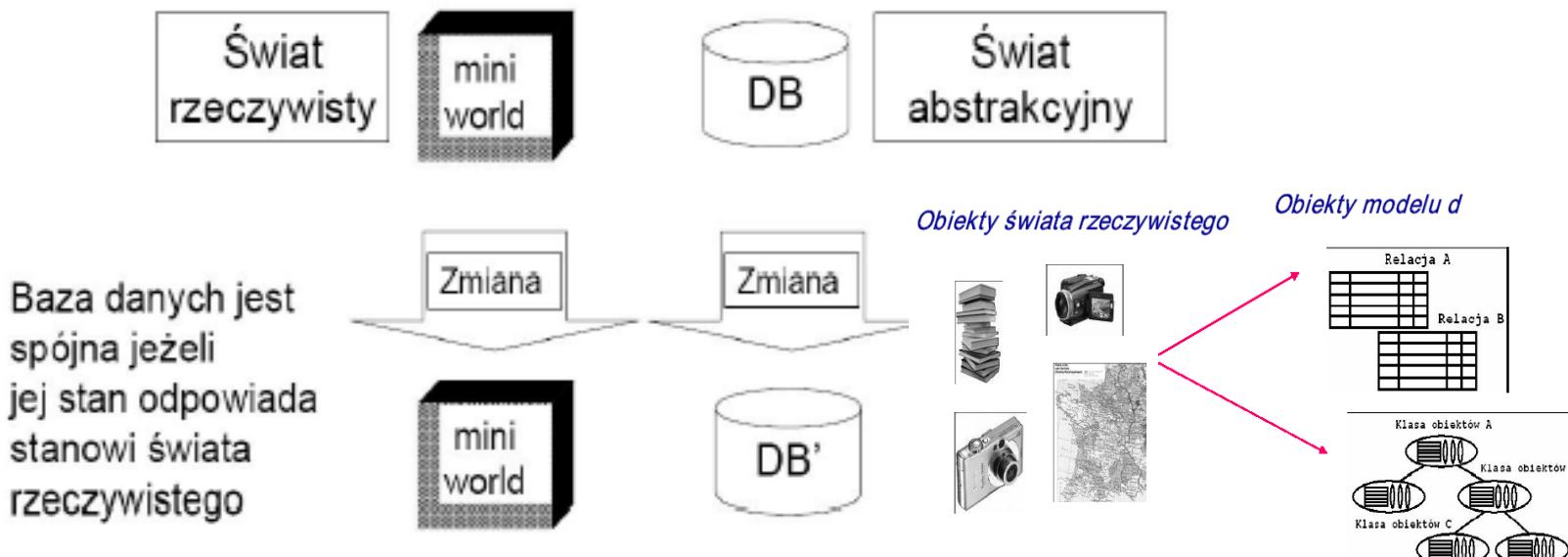
W szczególności zostaną omówione:

- transakcja i jej własności,
- formalny model transakcji,
- sekwencyjne i współbieżne realizacje zbioru transakcji,
- uszeregowalność transakcji.



wprowadzenie (1)

- Baza danych – jest abstrakcyjnym odzwierciedleniem wybranego fragmentu rzeczywistości (ang. *miniworld*)



Baza danych jest spójna, jeżeli jej stan odpowiada stanowi świata rzeczywistego.



wprowadzenie (2)

1. Zmiany zachodzące w świecie rzeczywistym muszą być zakodowane w postaci programu, który będzie transformował bazę danych z jednego stanu spójnego do innego stanu spójnego
2. Niebezpieczeństwa związane z realizacją programu transformującego bazę danych
 1. Awaryjność środowiska sprzętowo-programowego
 2. Współbieżny dostęp do danych
 3. Rozproszenie baz danych

1. **spójności BD** → transformacja BD z jednego stanu spójnego do innego stanu spójnego.
 2. Odporność na awarie sprzętowo-programowe.
 3. Obsługa **równoległa**.

problemy przygotowania aplikacji



Przykład: Napisać aplikację przelewu kwoty N z konta A na konto B

Problem 1 – awaria systemu

Po pobraniu kwoty N z konta A, i zapisaniu tej aktualizacji do bazy danych, wystąpiła awaria systemu. W wyniku awarii systemu wykonana została jedynie część operacji składających się na daną aplikację

Problem 2 – wspólnie dostępny dostęp do danych

Operacje współbieżnie wykonywanych transakcji mogą naruszać spójność bazy danych, lub generować niepoprawne wyniki

1

2

Przykład: system bankowy i aplikacja przelewu kwoty N z konta A na konto B.





transakcja (1)

Problem 3 - utrata danych w wyniku awarii

Wyniki zakończonych aplikacji, buforowane w pamięci operacyjnej, mogą zostać utracone w wyniku awarii systemu

Rozwiązaniem problemu awaryjności, rozproszenia i wielodostępności środowiska systemu bazy danych – koncepcja transakcji

3

Transakcja jest sekwencją logicznie powiązanych operacji na bazie danych, która przeprowadza bazę danych z jednego stanu spójnego w inny stan spójny. Typy operacji na bazie danych obejmują: odczyt i zapis danych oraz zakończenie i akceptację (zatwierdzenie), lub wycofanie transakcji

DEFINICJA



transakcja (2)

Transakcja przelewu kwoty N z konta A na konto B:

begin

// odejmij kwotę N z konta A;
update konta
 SET stan = stan - N
 where id_konta = A;
// dodaj do konta B kwotę N;
update konta
 SET stan = stan + N
 where id_konta = B;

commit;



własności transakcji (1)

A(tomicity)C(onsistency)I(solation)D(urability)

ATOMOWOŚĆ - Atomicity

SPÓJNOŚĆ - Consistency

ACID

własności transakcji (2)

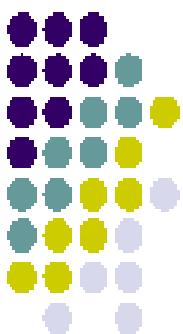


A(tomicity)C(onsistency)I(solation)D(urability)

IZOLACJA - Isolation

TRWAŁOŚĆ - Durability

ACID



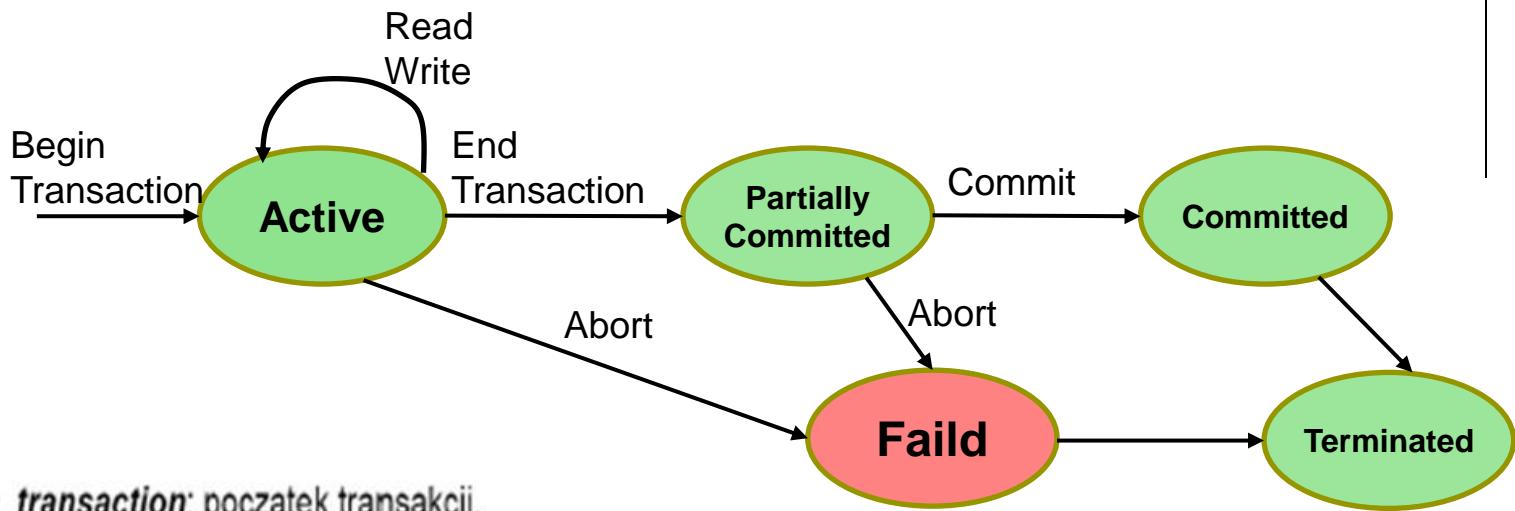
transakcja (3)

Transakcja jest:

Cechy transakcji

1. Atomowa: jeżeli pieniądze zostaną poprawnie przetransferowane z konta A do B
2. Spójna: jeżeli kwota odjęta z konta A jest równa kwocie dodanej do konta B
3. Izolowana: jeżeli inne transakcje wykonywane współbieżnie, czytające i modyfikujące konta A i B, nie mają wpływu na transakcję
4. Trwała: jeżeli po zakończeniu transakcji, baza danych trwale odzwierciedla nowe stany kont A i B

diagram stanów transakcji



Begin_transaction: początek transakcji.

Read, Write: operacje odczytu i zapisu danych w bazie danych.

End_transaction: koniec transakcji:

Commit: zatwierdzenie (akceptacja) wyników transakcji.

Rollback: wycofanie wyników transakcji

Stany transakcji

Active

Partially Committed

Committed

Failed

Terminated

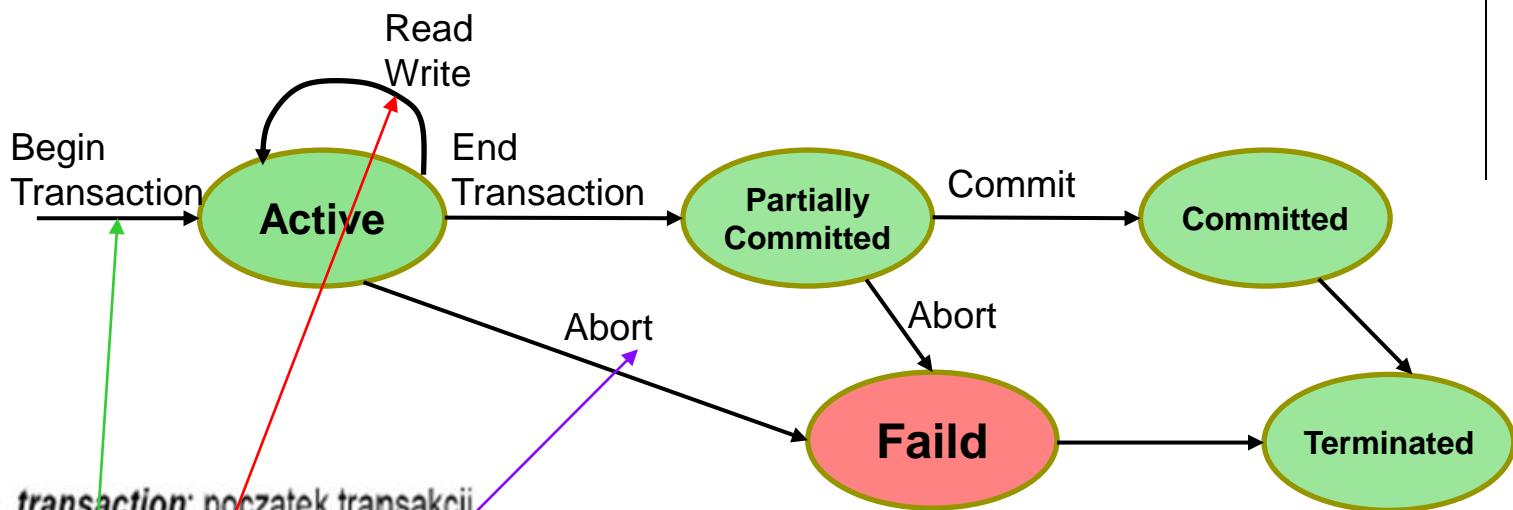
Stany transakcji

Save point:

Release savepoint

Rollback to savepoint

diagram stanów transakcji



Begin_transaction: początek transakcji.

Read, Write: operacje odczytu i zapisu danych w bazie danych.

End_transaction: koniec transakcji:

Commit: zatwierdzenie (akceptacja) wyników transakcji.

Rollback: wycofanie wyników transakcji

1. (**Begin Transaction**) uruchamia transakcję, która jest aktywna.
2. (**Read, Write**) dokonuje się w stanie aktywnym transakcji.
3. (**Abort**) przeprowadza transakcję ze stanu **Active** do stanu **Failed**, a następnie **Terminate**.
4. Kończenie transakcji z jej zatwierdzeniem przeprowadza ją ze stanu **Active** do **Partially committed** transakcja jest gotowa do zatwierdzenia.



zakończenie transakcji

End_transaction: koniec transakcji oznacza, że wszystkie operacje odczytu i/lub zapisu transakcji zostały wykonane. W tym momencie, zachodzi konieczność podjęcia decyzji, czy zmiany wprowadzone przez transakcję mają być wprowadzone do bazy danych (zatwierdzenie transakcji) czy też mają być wycofane z bazy danych

Commit: zatwierdzenie (akceptacja transakcji) oznacza pomyślne zakończenie transakcji - zmiany wprowadzone przez transakcję mają być wprowadzone do bazy danych

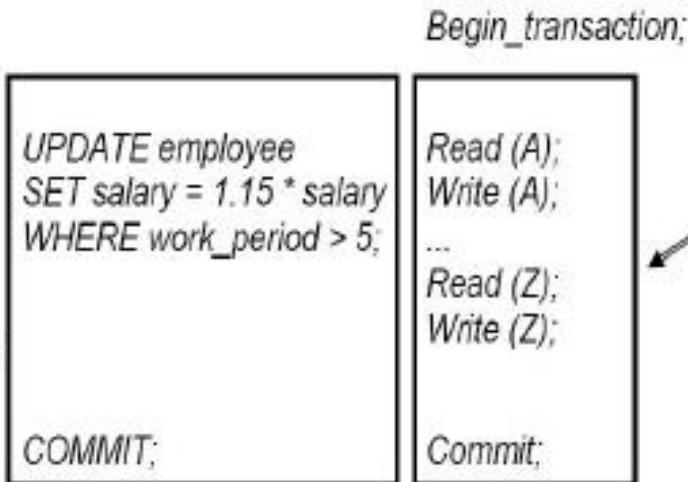
Rollback: wycofanie transakcji oznacza niepoprawne zakończenie transakcji i konieczność wycofania z bazy danych wszystkich ewentualnych zmian wprowadzonych przez transakcję



transakcja logiczna a fizyczna



Transakcja
logiczna



Transakcja
fizyczna



Poziom użytkownika: zbiór poleceń języka SQL, tj. **select, insert, update, delete, commit, rollback** - tzw. [transakcja logiczna](#).

Poziom SZBD - [transakcja fizyczna](#), która jest zarządzana przez odpowiedni moduł **SZBD**.



model transakcji (1)

Transakcją T_i nazywamy uporządkowaną parę:

gdzie:

$$T_j = (\overline{T}_i, < T_j)$$

$\overline{T}_i = \{o_j : 1 \leq j \leq n_j\}$ - oznacza zbiór operacji na bazie danych:

{ R - odczyt, W - zapis, C – zatwierdzenie transakcji, A - wycofanie }

$< T_j$ - jest relacją częściowego porządku na zbiorze T_i

Zbiór operacji zawiera:

- odczyt (**R**),
- zapis (**W**),
- zatwierdzenie transakcji (**C**),
- wycofanie transakcji (**A**).



model transakcji (1)

Transakcją T_i nazywamy uporządkowaną parę:

gdzie:

$$T_j = (\overline{T}_i, < T_j)$$

$\overline{T}_i = \{o_j : 1 \leq j \leq n_j\}$ - oznacza zbiór operacji na bazie danych:

{ R - odczyt, W - zapis, C – zatwierdzenie transakcji, A - wycofanie }

$< T_j$ - jest relacją częściowego porządku na zbiorze T_i

Dalsza notacja: Przyjmiemy następującą notację:

- $r_i(x)$ lub $r_i(x, \text{wartość})$
- $w_i(x)$ lub $w_i(x, \text{wartość})$
- c_i lub a_i



model transakcji (2)

Każda transakcja może być reprezentowana przez graf skierowany:

G = (V, A), gdzie:

V - jest zbiorem węzłów odpowiadających operacjom transakcji T_i

A - jest zbiorem krawędzi reprezentujących porządek na zbiorze operacji

Przykład:

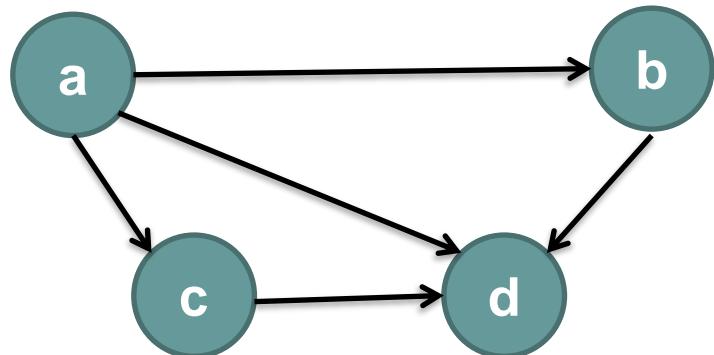
$r_1(x) \rightarrow w_1(x) \rightarrow r_2(y) \rightarrow w_2(y) \rightarrow c_1$

sekwencyjne wykonywanie
transakcji

$r_1(x) \rightarrow w_1(x) \rightarrow w_2(y) \rightarrow c_1$

$r_2(y)$

współbieżne wykonywanie
transakcji



Graf skierowany, digraf - DG



klasyfikacja transakcji

Ze względu na porządek operacji:

- transakcja sekwencyjna
- transakcja współbieżna

Ze względu na zależność operacji:

- transakcja zależna od danych
- transakcja niezależna od danych

Ze względu na typy operacji:

- zapytania lub transakcja odczytu (read only)
- transakcja aktualizująca - transakcja (read/write)

Trzy kryteria podziału transakcji: **porządek operacji, zależność operacji, typ operacji.**

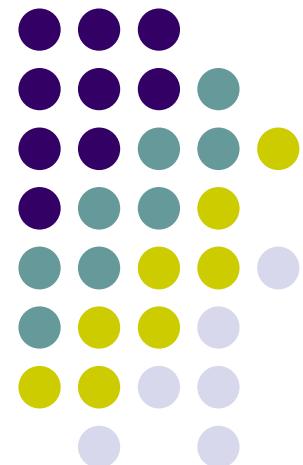


KONIEC WYKŁADU

Cz. 1

wykład

Przetwarzanie transakcyjne
Cz.2



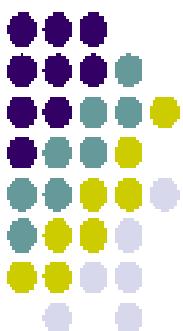


Plan wykładu

Celem niniejszego wykładu jest omówienie problematyki związanej z transakcjami w bazie danych.

W szczególności zostaną omówione:

- transakcja i jej własności,
- formalny model transakcji,
- sekwencyjne i współbieżne realizacje zbioru transakcji,
- uszeregowalność transakcji.



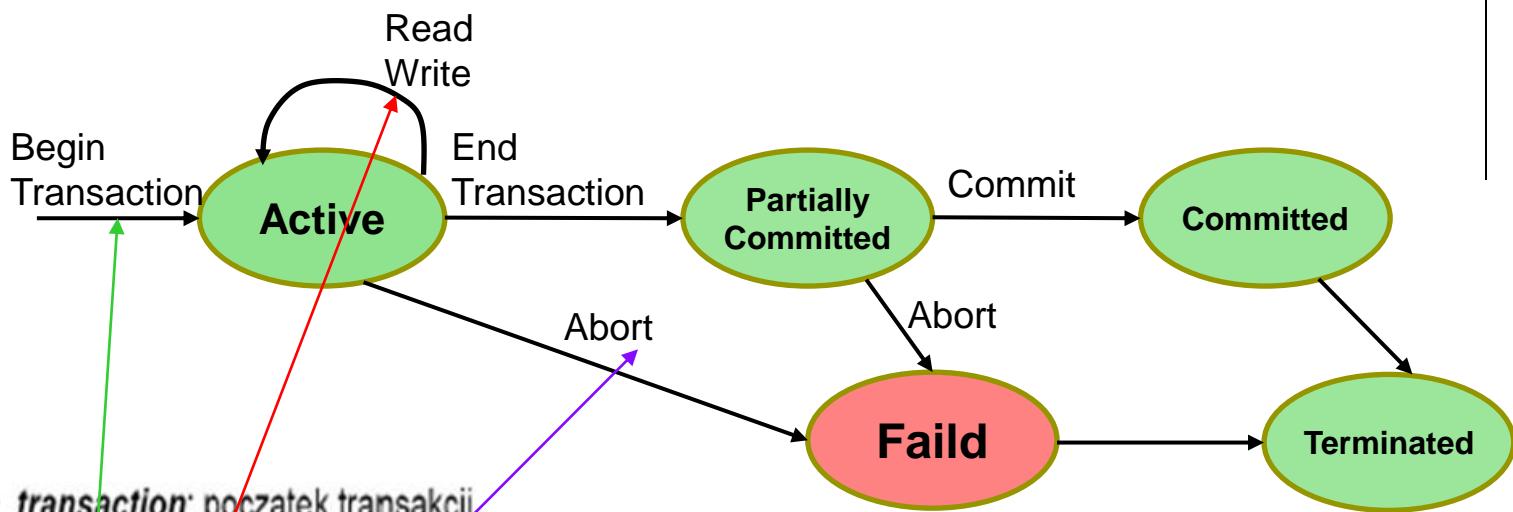
transakcja (3)

Transakcja jest:

Cechy transakcji

1. Atomowa: jeżeli pieniądze zostaną poprawnie przetransferowane z konta A do B
2. Spójna: jeżeli kwota odjęta z konta A jest równa kwocie dodanej do konta B
3. Izolowana: jeżeli inne transakcje wykonywane współbieżnie, czytające i modyfikujące konta A i B, nie mają wpływu na transakcję
4. Trwała: jeżeli po zakończeniu transakcji, baza danych trwale odzwierciedla nowe stany kont A i B

diagram stanów transakcji



Begin_transaction: początek transakcji.

Read, Write: operacje odczytu i zapisu danych w bazie danych.

End_transaction: koniec transakcji:

Commit: zatwierdzenie (akceptacja) wyników transakcji.

Rollback: wycofanie wyników transakcji

1. (**Begin_Transaction**) uruchamia transakcję, która jest aktywna.
2. (**Read, Write**) dokonuje się w stanie aktywnym transakcji.
3. (**Abort**) przeprowadza transakcję ze stanu **Active** do stanu **Failed**, a następnie **Terminate**.
4. Kończenie transakcji z jej zatwierdzeniem przeprowadza ją ze stanu **Active** do **Partially committed** transakcja jest gotowa do zatwierdzenia.



model transakcji (2)

Każda transakcja może być reprezentowana przez graf skierowany:

G = (V, A), gdzie:

V - jest zbiorem węzłów odpowiadających operacjom transakcji T_i ;

A - jest zbiorem krawędzi reprezentujących porządek na zbiorze operacji.

Przykład:

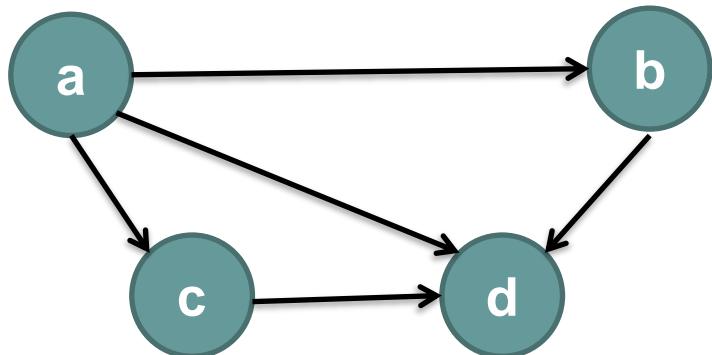
$r_1(x) \rightarrow w_1(x) \rightarrow r_2(y) \rightarrow w_2(y) \rightarrow c_1$

sekwencyjne wykonywanie
transakcji

$r_1(x) \rightarrow w_1(x) \rightarrow w_2(y) \rightarrow c_1$

$r_2(y)$

współbieżne wykonywanie
transakcji



Graf skierowany, digraf - DG



klasyfikacja transakcji

Ze względu na porządek operacji:

- transakcja sekwencyjna
- transakcja współbieżna

1

Ze względu na zależność operacji:

- transakcja zależna od danych
- transakcja niezależna od danych

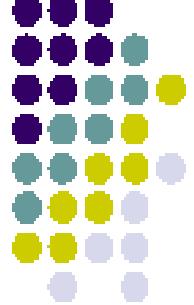
2

Ze względu na typy operacji:

- zapytania lub transakcja odczytu (read only)
- transakcja aktualizująca - transakcja (read/write)

3

Trzy kryteria podziału transakcji: **porządek operacji, zależność operacji, typ operacji.**



realizacje transakcji (1)

- Częściowo uporządkowaną sekwencją operacji należących do zbioru współbieżnie wykonywanych transakcji nazywamy realizacją (historią). Realizacja modeluje, formalnie, współbieżne wykonanie zbioru transakcji
- Formalnie, realizacją S zbioru n transakcji T_1, T_2, \dots, T_n nazywamy takie uporządkowanie operacji współbieżnie wykonywanych transakcji, w którym, dla każdej transakcji T_i w realizacji S , porządek wykonania operacji transakcji T_i jest taki sam jak porządek $<_{Ti}$

W praktyce, w jednym systemie bazy danych działa równocześnie wiele transakcji.



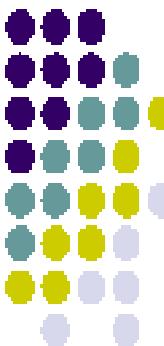
realizacje transakcji (2)

Formalna notacja realizacji S zbioru transakcji.

$$S(\tau) = (\bar{T}_r(\tau), < r)$$

POS: *partially ordered set*

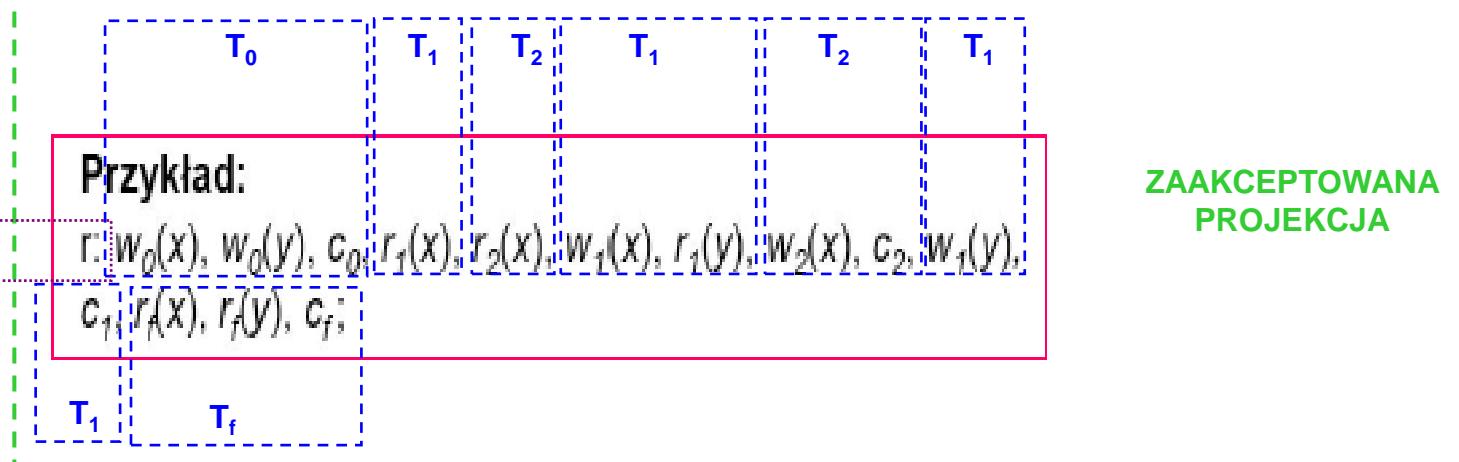
- gdzie:
 1. $\bar{T}_r(\tau)$ zbiór operacji wszystkich transakcji należących do zbioru τ
 2. $< r$ relacja częściowego porządku na zbiorze $\bar{T}_r(\tau)$,
 3. Dla dowolnej pary operacji $o_i, o_j \in \bar{T}_r(\tau)$, takich, że żądają one dostępu do tej samej danej i co najmniej jedna z nich jest operacją zapisu, zachodzi $o_i < r o_j$ lub $o_j < r o_i$



realizacje transakcji (3)

Realizacja zawierająca tylko operacje zatwierdzonych transakcji nazywana jest *zaakceptowaną projekcją*

(Dalsze rozważania dotyczyć będą tylko realizacji spełniających powyższy warunek)



[Przykład zaakceptowanej projekcji](#)

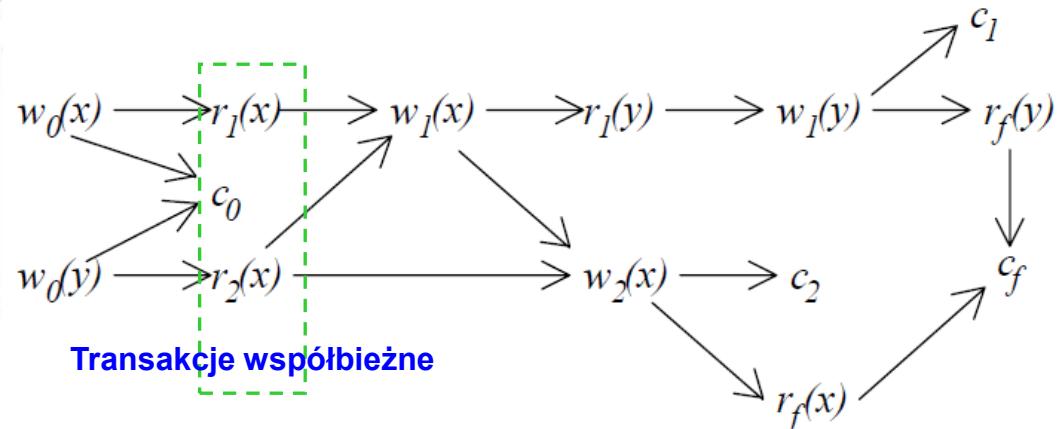


realizacje transakcji (4)

- Dowolną realizację można przedstawić w postaci grafu skierowanego, nazywanego grafem realizacji,

$GR(s(\tau)) = (V, A)$. Węzły grafu odpowiadają operacjom ze zbioru $T_r(\tau)$, natomiast krawędzie grafu reprezentują relację częściowego porządku $< r$

- Przykład:



Przykład:

$r: w_0(x), w_0(y), c_0, r_1(x), r_2(x), w_1(x), r_1(y), w_1(y), c_2, w_2(x), c_1, r_f(x), r_f(y), c_f;$

Przykład grafu realizacji dla transakcji T_0, T_1, T_2, T_f przedstawiono na slajdzie.

Przykład zaakceptowanej projekcji



realizacje sekwencyjne i współbieżne

1. Mówimy, że dana realizacja jest sekwencyjna jeżeli, dla każdych dwóch transakcji, wszystkie operacje jednej z nich poprzedzają wszystkie operacje drugiej
2. W przeciwnym wypadku realizacja jest współbieżna

Stan i obraz bazy danych

Stan bazy danych - zbiór wartości wszystkich danych w bazie danych

Obraz bazy danych - widziany przez transakcję T_i zbiór wartości danych odczytywanych przez transakcję T_i



uszeregowalność realizacji (1)

Założenie 1

każda realizacja sekwencyjna jest poprawna

Założenie 2

każda realizacja współbieżna równoważna dowolnej realizacji sekwencyjnej tego samego zbioru transakcji jest również poprawna

Przykład:

Dane (początkowe wartości): $a=50; b=50$

Transakcja T1: sumuje konta a i b

Transakcja T2: przelewa 30 z konta a na konto b

Dana realizacja postaci:

s: ...r2(a, 50) w2(a, 20) r1(a,20) r1(b, 50) r2(b,50)
w2(b, 80) c1 c2

Czy dana realizacja jest poprawna?



Czy realizacja jest poprawna?



uszeregowalność realizacji (1)

Przykład:

Dane (początkowe wartości): $a=50$; $b=50$

Transakcja T1: sumuje konta a i b

Transakcja T2: przelewa 30 z konta a na konto b

Dana realizacja postaci:

s: ...r2(a, 50) w2(a, 20) r1(a,20) r1(b, 50) r2(b,50)
w2(b, 80) c1 c2

Czy dana realizacja jest poprawna?

Przedstawiona na slajdzie realizacja **nie jest poprawna** ponieważ obraz bazy danych widziany przez transakcję **T₁** to $a+b=70$, zamiast **100**.



uszeregowalność realizacji (1)

Realizacje sekwencyjne transakcji T1 i T2:

```
s1...r1(a, 50) r1(b, 50) c1 r2(a, 50) w2(a, 20)  
r2(b, 50) w2(b, 80) c2 ....
```

końcowy stan bazy danych: a= 20; b= 80

obraz bazy danych widziany przez T2: a = 50; b = 50

obraz bazy danych widziany przez T1: a = 50; b = 50

W przykładzie ze slajdu transakcje **T₁** i **T₂** są realizowane sekwencyjnie.

W tym przypadku obraz bazy danych widziany przez obie transakcje **jest poprawny**.



konflikt (1)

pojęcia konfliktu dwóch operacji

Dwie **operacje** $o_i(x)$, $o_j(y)$ współbieżnej realizacji są **konfliktowe**, wtedy i tylko wtedy, gdy są spełnione następujące trzy warunki:

1. $x = y$ Operacje na różnych danych nigdy nie są konfliktowe

1

2. $i \neq j$ Operacje konfliktowe muszą należeć do różnych transakcji

2

3. Jedna z dwóch operacji o_i lub o_j musi być operacją zapisu

3



konflikt (2)

Rozszerzenie pojęcia konfliktu na zbiór (dwie) transakcji

- Dwie transakcje T_i, T_j są konfliktowe, jeżeli zawierają wzajemnie konfliktowe operacje
- Mówimy, że operacja $o_i(x)$ poprzedza operację $o_j(y)$ w realizacji $r(\tau)$, co zapisujemy jako $o_i(x) \rightarrow o_j(y)$, jeżeli operacje te są konfliktowe i operacja $o_i(x)$ poprzedza $o_j(y)$ w $r(\tau)$.
- Następujące pary operacji mogą znajdować się w konflikcie:
 - $r(x) \quad w(x)$
 - $w(x) \quad r(x)$
 - $w(x) \quad w(x)$

Pojęcie konfliktu można rozszerzyć na zbiór transakcji.

Dwie transakcje T_i oraz T_j są konfliktowe, jeżeli zawierają wzajemnie konfliktowe operacje.



konfliktowa równoważność

Rozszerzenie relacji poprzedzania na zbiór (dwie) transakcji

- Mówimy, że transakcja T_i poprzedza transakcję T_j w realizacji $r(\tau)$, co zapisujemy jako $T_i \rightarrow T_j$ jeżeli zawierają odpowiednio operacje $o_i(x)$ i $o_j(y)$, między którymi zachodzi związek poprzedzania
- Mówimy, że dwie realizacje $r(\tau) = (T_r(\tau), < r)$ i $r'(\tau) = (T_{r'}(\tau), < r')$ są konfliktowo równoważne, jeżeli dla każdej pary operacji $o_i(x)$ i $o_j(y)$ w realizacji $r(\tau)$, takich, że $\rightarrow o_i(x) \rightarrow o_j(y)$ w realizacji $r(\tau)$

2. $i \neq j$ Operacje konfliktowe muszą należeć do różnych transakcji

Relacje poprzedzania można rozszerzyć na zbiór transakcji.



konfliktowa równoważność

Pojęcie równoważności dwóch realizacji

- Mówimy, że transakcja T_i poprzedza transakcję T_j w realizacji $r(\tau)$, co zapisujemy jako $T_i \rightarrow T_j$ jeżeli zawierają odpowiednio operacje $o_i(x)$ i $o_j(y)$, między którymi zachodzi związek poprzedzania

- Mówimy, że dwie realizacje $r(\tau) = (T_r(\tau), < r)$ i $r'(\tau) = (T_{r'}(\tau), < r')$ są konfliktowo równoważne, jeżeli dla każdej pary operacji $o_i(x)$ i $o_j(y)$ w realizacji $r(\tau)$, takich, że $o_i(x) \rightarrow o_j(y)$, zachodzi również $o_i(x) \rightarrow o_j(y)$ w realizacji $r'(\tau)$

Po wprowadzeniu relacji poprzedzania, można formalnie zdefiniować pojęcie równoważności dwóch realizacji.

Obecnie, sformułowane zostanie kryterium poprawności współbieżnej realizacji zbioru transakcji nazywane **kryterium konfliktowej uszeregowalności**.



kryterium konfliktowej uszeregowalności

Kryterium konfliktowej uszeregowalności

Realizacja $r(\tau)$ zbioru transakcji τ jest **konfliktowo uszeregowalna** wtedy i tylko wtedy, gdy jest ona konfliktowo równoważna dowolnej sekwencyjnej realizacji τ

definicja

**Weryfikacja
konfliktowej
uszeregowalności**

Grafem konfliktowej-uszeregowalności realizacji $r(\tau)$

nazywamy skierowany graf $\text{CSRG}(r(\tau)) = (V, A)$, taki, w którym zbiór wierzchołków V odpowiada transakcjom ze zbioru , natomiast zbiór krawędzi $A = \{(T_i, T_j) : T_i \rightarrow T_j\}$

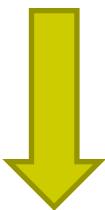
graf konfliktowej uszeregowalności realizacji



twierdzenie konfliktowej uszeregowalności

Realizacja $r(\tau)$ zbioru transakcji jest
konfliktowo-uszeregowalna

wtedy i tylko wtedy, gdy jej graf konfliktowej
uszeregowalności $\text{CSRG}(r(\tau))$ jest acykliczny



Korzystając z grafu konfliktowej uszeregowalności można sformułować twierdzenie,
pozwalające
w sposób algorytmiczny weryfikować, czy dana realizacja wspólnie jest poprawna, tj.
konfliktowo uszeregowalna.

Realizacja $r(\tau)$ zbioru transakcji T jest konfliktowo uszeregowalna wtedy i tylko
wtedy, gdy jej graf konfliktowej uszeregowalności $\text{CSRG}(r(\tau))$ jest acykliczny.

- 1 każda realizacja sekwencyjna zbioru transakcji zachowuje spójność bazy danych
- 2 z definicji grafu konfliktowej uszeregowalności wynika, że graf ten, dla dowolnej realizacji
sekwencyjnej, musi być acykliczny
- 3 z definicji równoważności realizacji wynika, że graf konfliktowej uszeregowalności realizacji
współbieżnej musi być również acykliczny



realizacje odtwarzalne (1)

- Czy własność uszeregowalności gwarantuje wolność od anomalii?

Przykład:

$H = r_1[x] w_1[x] r_1[y] r_2[x] w_1[y] r_2[y] c_2 r_1[z] w_1[z] <\text{crash}> c_1$

- Historia H jest uszeregowalna, ale nie jest wolna od anomalii (brudny odczyt). Po restarcie systemu transakcja T2 nie zostanie poprawnie odtworzona

Czy własność uszeregowalności gwarantuje poprawność dowolnej realizacji transakcji, w szczególności, czy gwarantuje wolność od anomalii współbieżnego wykonywania transakcji?

1 awaria

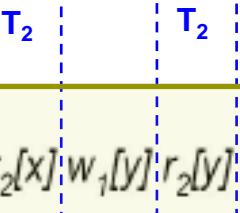
2 wycofanie

3 zakleszczenie



realizacje odtwarzalne (1)

- Czy własność uszeregowalności gwarantuje wolność od anomalii ?



Przykład:

$H = r_1[x] w_1[x] r_1[y] r_2[x] w_1[y] r_2[y] c_2 r_1[z] w_1[z] <\text{crash}> c_1$

- Historia H jest uszeregowalna, ale nie jest wolna od anomalii (brudny odczyt). Po restarcie systemu transakcja T2 nie zostanie poprawnie odtworzona

Jeżeli rozważamy realizacje, które zawierają operacje wycofywanych transakcji, to odpowiedź na postawione na wstępie pytanie **jest negatywna**.



definicje

- Potrzebna jest definicja nowych własności realizacji wykluczających anomalie będące wynikiem awarii systemu
- Mówimy, że transakcja T_i czyta daną x z transakcji T_j w realizacji H jeżeli

$$w_j[x] < r_i[x]$$

$$a_j < r_i[x]$$

jeżeli istnieje operacja $w_k[x]$ taka, że $w_j[x] < w_k[x] < r_i[x]$,
wtedy $a_k < r_i[x]$

Jeżeli rozważamy szerszą klasę realizacji, które zawierają operacje zatwierdzonych jak i wycofywanych, na skutek awarii, transakcji, **potrzebne są definicje nowych własności realizacji, wykluczających anomalie będące wynikiem awarii systemu.**



realizacje odtwarzalne (2)



- Realizacja H jest **odtwarzalna** (ang. *recoverable*) (RC) wówczas, jeżeli transakcja T_i czyta z transakcji T_j ($i \neq j$) w realizacji H i $c_i \in H$, to $c_j < c_i$



- Realizacja H unika **kaskadowych wycofań** (ang. *avoids cascading aborts*) (ACA) wówczas, jeżeli transakcja T_i czyta z transakcji T_j ($i \neq j$), to $c_j < r_i[x]$



- Realizacja H jest **ścisła** (ang. *strict*) (ST) wówczas, jeżeli $w_j[x] < o_i[x]$ ($i \neq j$), zachodzi $a_j < o_i[x]$ lub $c_j < o_i[x]$, gdzie $o_i[x]$ jest jedną z operacji $r_i[x]$ lub $w_i[x]$



realizacje uszeregowalne

- Realizacja z zbioru transakcji jest poprawna (uszeregowalna) jeżeli jest ona obrazowo i stanowo równoważna jakiejkolwiek sekwencyjnej realizacji tego zbioru transakcji. Realizację taką nazywamy realizacją uszeregowalną (SR)

Przedstawiona na poprzednich slajdach definicja kryterium konfliktowej uszeregowalności stanowi zmodyfikowaną wersję podstawowego kryterium poprawności współbieżnej realizacji transakcji, które nosi nazwę kryterium uszeregowalności.

Zasadnicza różnica pomiędzy definicją kryterium uszeregowalności a kryterium konfliktowej uszeregowalności kryje się w definicji równoważności realizacji transakcji.



graf uszeregowalności (1)

- **Grafem uszeregowalności** realizacji $r(\tau)$ nazywamy skierowany graf $SG(r(\tau)) = (V, A)$, taki, w którym zbiór wierzchołków V odpowiada transakcjom ze zbioru τ , natomiast zbiór krawędzi jest zdefiniowany następująco:
 - Jeżeli istnieje dana x , i operacje $T_i : r(x), T_j : w(x) \in T_r(\tau)$, takie, że $T_i : r(x)$ czyta wartość danej x zapisanej przez operację $T_j : w(x)$, to:
 1. $(T_j, T_i) \in A$
 2. Jeżeli $T_j \neq T_0, T_i \neq T_f$ i istnieje operacja $T_k : w(x) \in T_r(\tau)$, $T_k \neq T_0$, to $(T_k, T_j) \in A$ lub $(T_j, T_k) \in A$
 3. Jeżeli $T_j \neq T_0$, to $(T_0, T_j) \in A$

W celu weryfikacji uszeregowalności realizacji konstrujemy graf uszeregowalności realizacji.



graf uszeregowalności (2)

4. Jeżeli $T_j = T_0$, $T_i \neq T_f$ i istnieje operacja $T_k : w(x) \in T_r(\tau)$, $T_k \neq T_0$, to $(T_i, T_k) \in A$;
5. Jeżeli $T_i = T_f$, i istnieje operacja $T_k : w(x) \in T_r(\tau)$, to $(T_k, T_j) \in A$

Dana realizacja $r(\tau)$ jest uszeregowalna wtedy i tylko wtedy, gdy można skonstruować dla niej acykliczny skierowany graf uszeregowalności $SG(r(\tau))$

definicja

2. Jeżeli $T_j \neq T_0$, $T_i \neq T_f$ i istnieje operacja $T_k : w(x) \in T_r(\tau)$, $T_k \neq T_0$, to $(T_k, T_j) \in A$ lub $(T_j, T_k) \in A$

Można pokazać, że dana realizacja $r(T)$ jest uszeregowalna wtedy i tylko wtedy, gdy można skonstruować dla niej **acykliczny skierowany graf uszeregowalności** $SG(r(T))$.



Problem NP zupełny



graf uszeregowalności (2)

4. Jeżeli $T_j = T_0$, $T_i \neq T_f$ i istnieje operacja $T_k : w(x) \in T_r(\tau)$, $T_k \neq T_0$, to $(T_i, T_k) \in A$;
5. Jeżeli $T_i = T_f$, i istnieje operacja $Tk : w(x) \in T_r(\tau)$, to $(T_k, T_j) \in A$

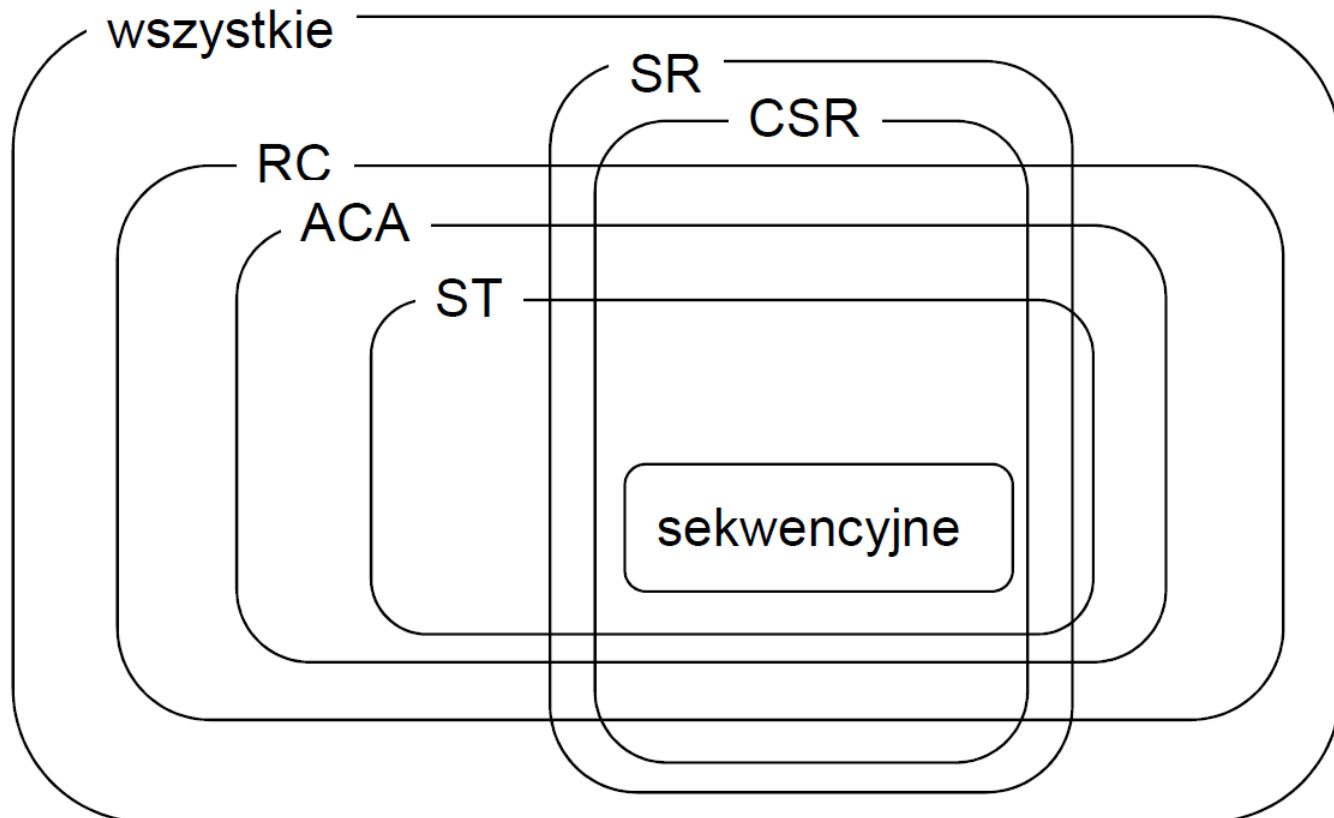
Dana realizacja $r(\tau)$ jest uszeregowalna wtedy i tylko wtedy, gdy można skonstruować dla niej acykliczny skierowany graf uszeregowalności $SG(r(\tau))$



Z teorii grafów wynika, że weryfikacja czy dany poligraf zawiera cykl jest problemem NP zupełnym.



Uszeregowalność transakcji - klasyfikacja



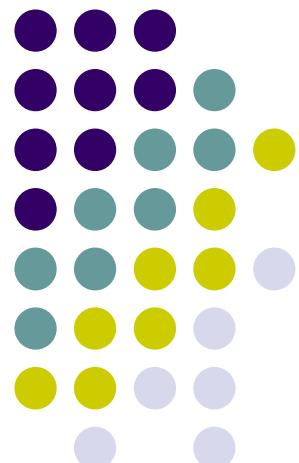


KONIEC WYKŁADU
Cz. 2

wykład

Zarządzanie współbieżnym
wykonywaniem transakcji

Cz.1





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.

1/2

2/2



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 (O)
 - 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 uzyskana	R	W
Blokada żądana	R	✓
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 uzyskana	R	W
Blokada żądana	✓	-
R	✓	-
W	✓	✓

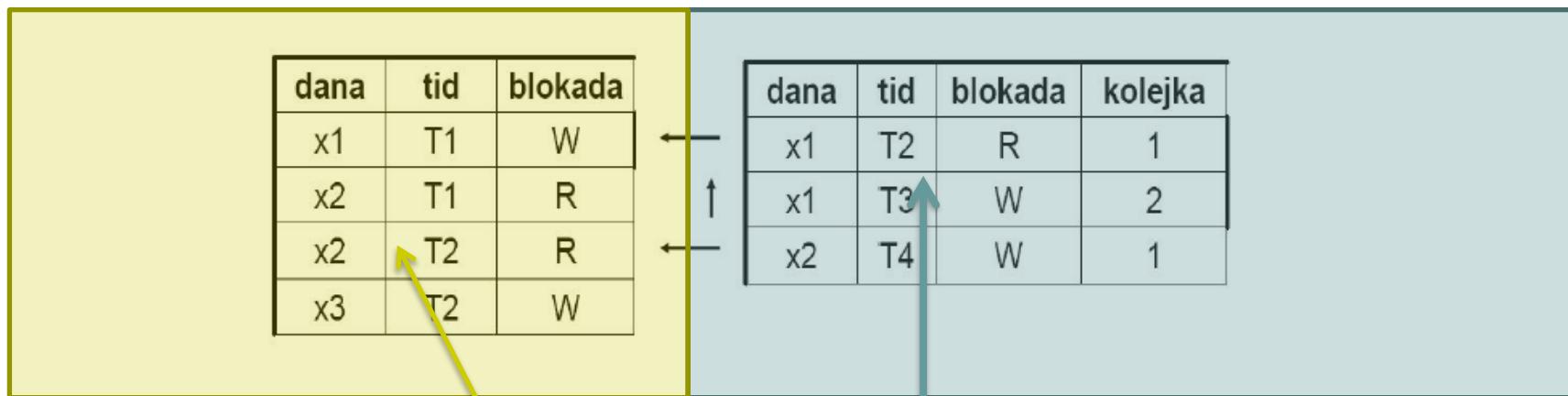
konwersja niedopuszczalna

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.**



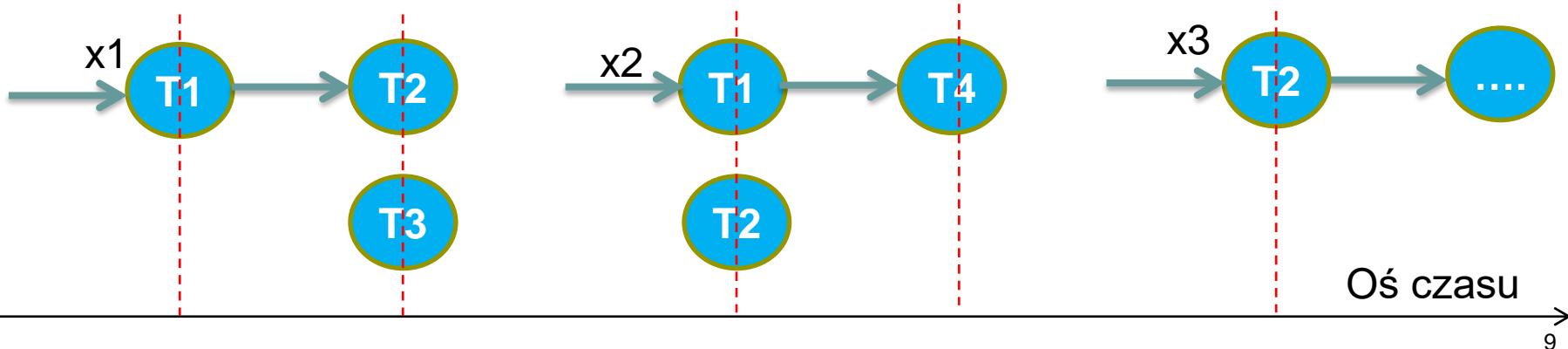
implementacja algorytmów blokowania (1)

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)

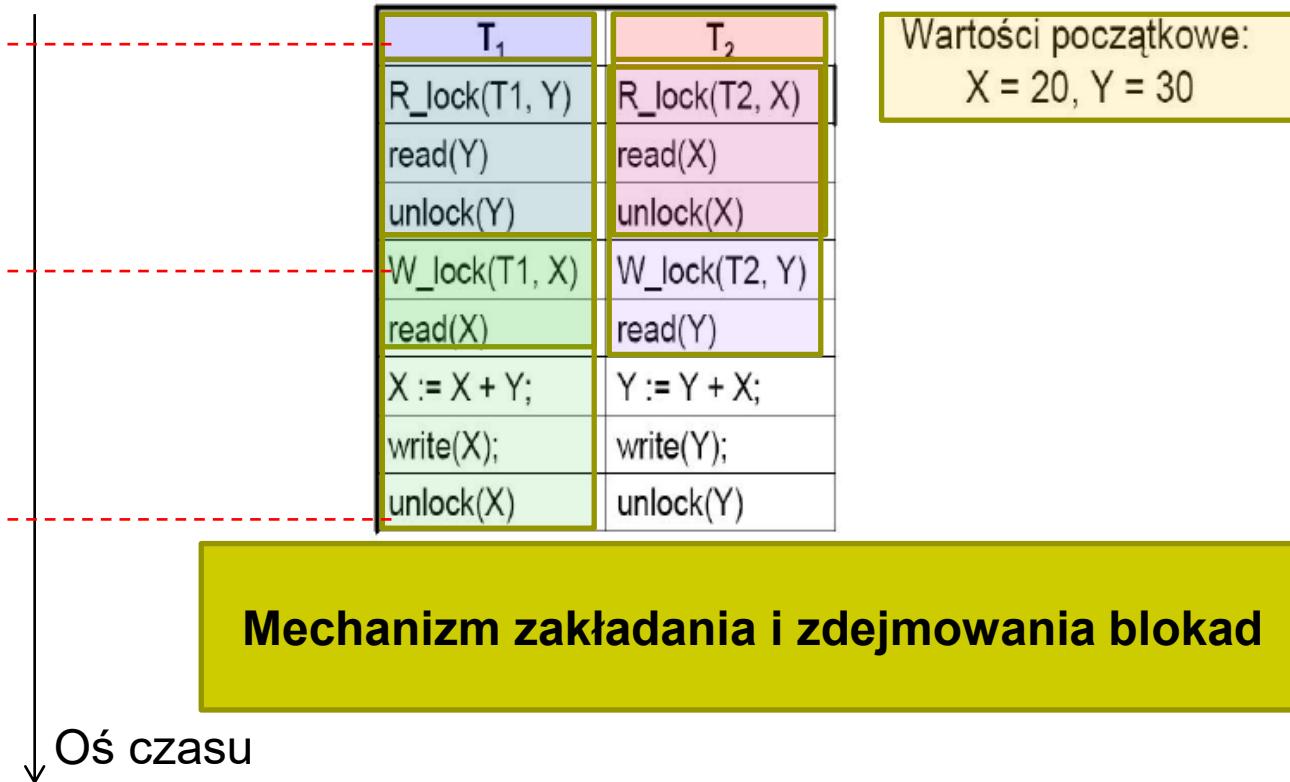
```
Unlock(X, tid) begin
    if LOCK(X, tid) = W
        then begin
            LOCK(X, tid) ← 0;
            < wake up one of the waiting
                transactions, if any >;
        end;
    else if LOCK(X, tid) = R
        then begin
            LOCK(X, tid) ← 0;
            if (number_of_read_locks_on_X=0)then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

Algorytm odblokowania danej X przez transakcję tid

3



algorytm blokowania (1)

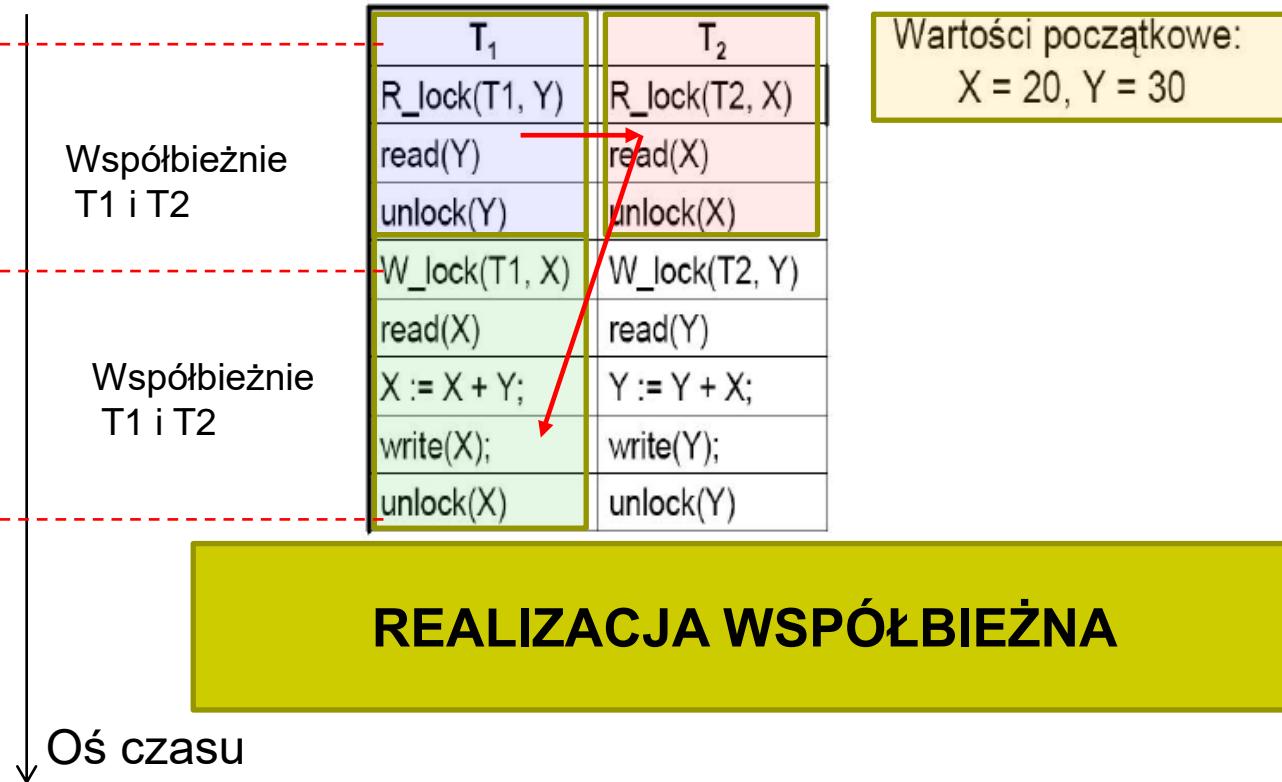


Przykład działania mechanizmu zakładania i zdejmowania blokad

Założmy, że wartości początkowe danych X i Y wynoszą, odpowiednio, 20 i 30.



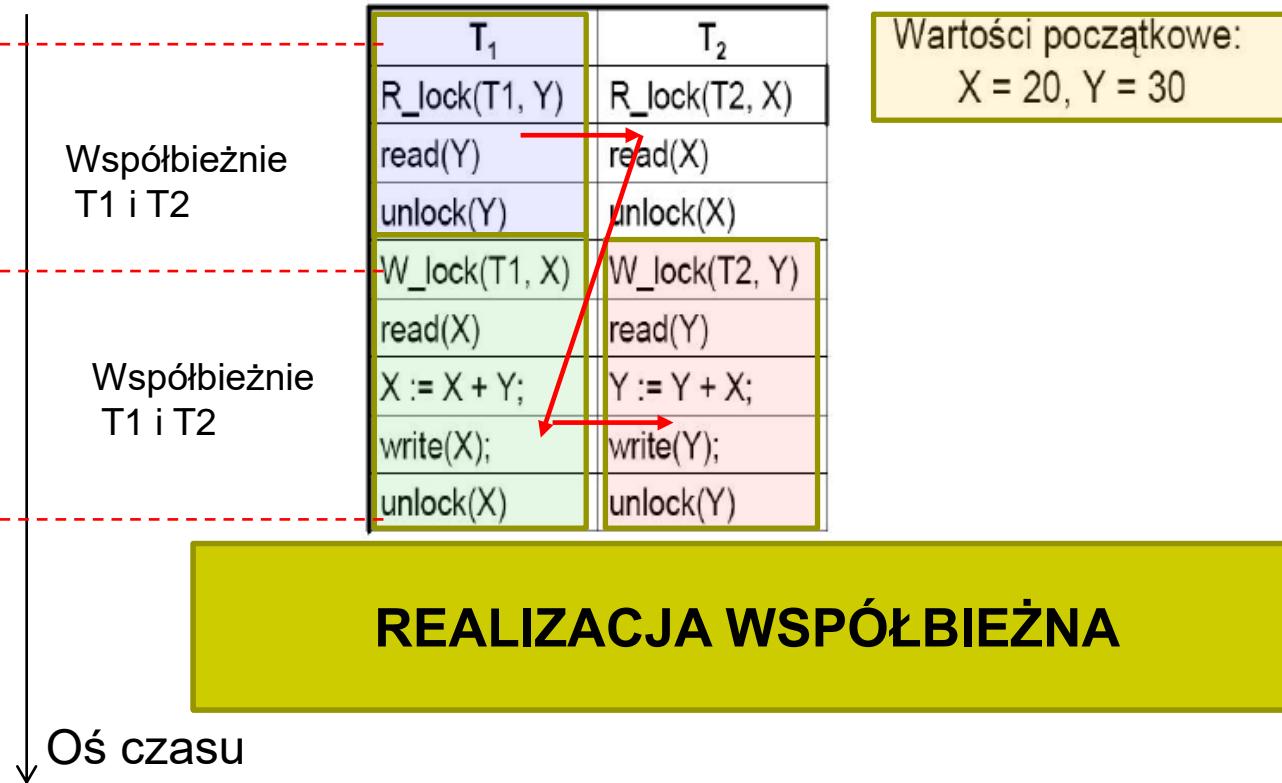
algorytm blokowania (1)



Przykład działania mechanizmu zakładania i zdejmowania blokad



algorytm blokowania (1)



Przykład działania mechanizmu zakładania i zdejmowania blokad



algorytm blokowania (1)

	T ₁	T ₂
Współbieżnie T1 i T2	R_lock(T1, Y)	R_lock(T2, X)
	read(Y)	read(X)
	unlock(Y)	unlock(X)
Współbieżnie T1 i T2	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

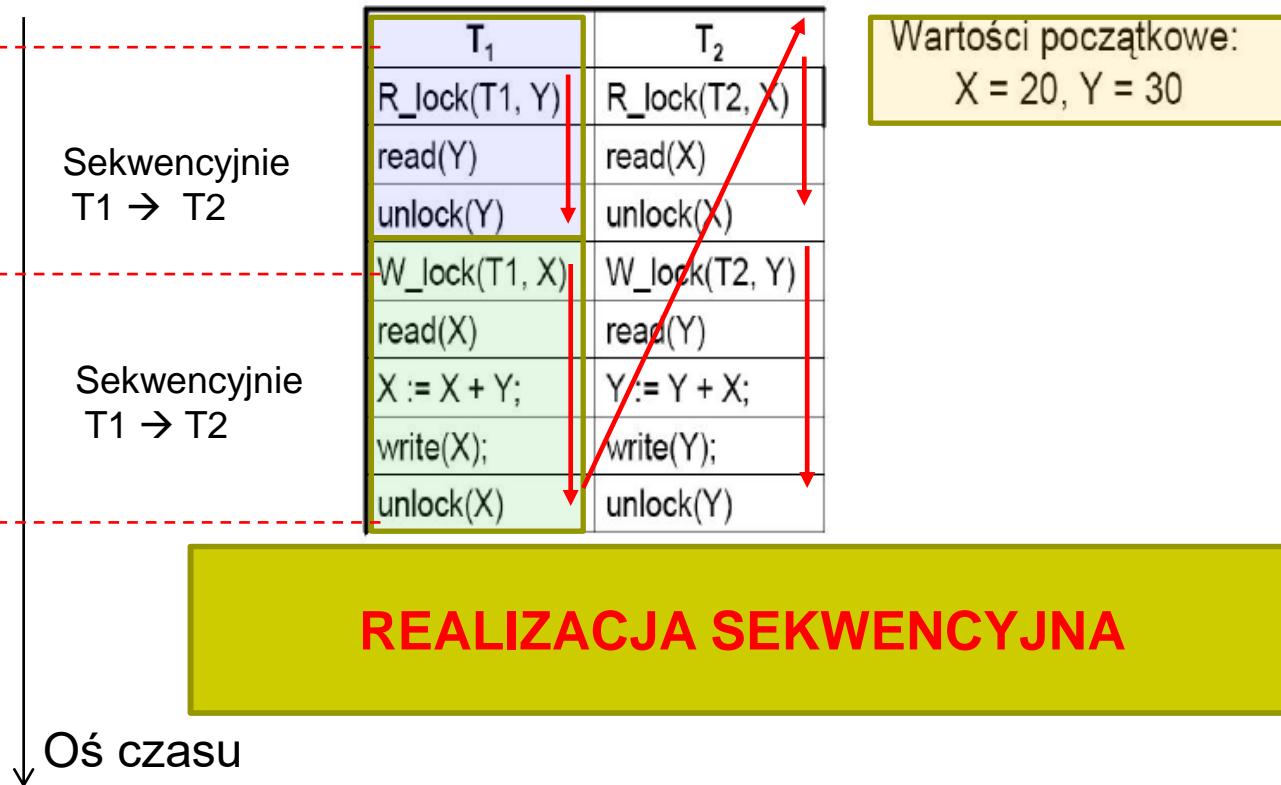
REALIZACJA WSPÓŁBIEŻNA - WYNIK

Oś czasu

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



algorytm blokowania (1)



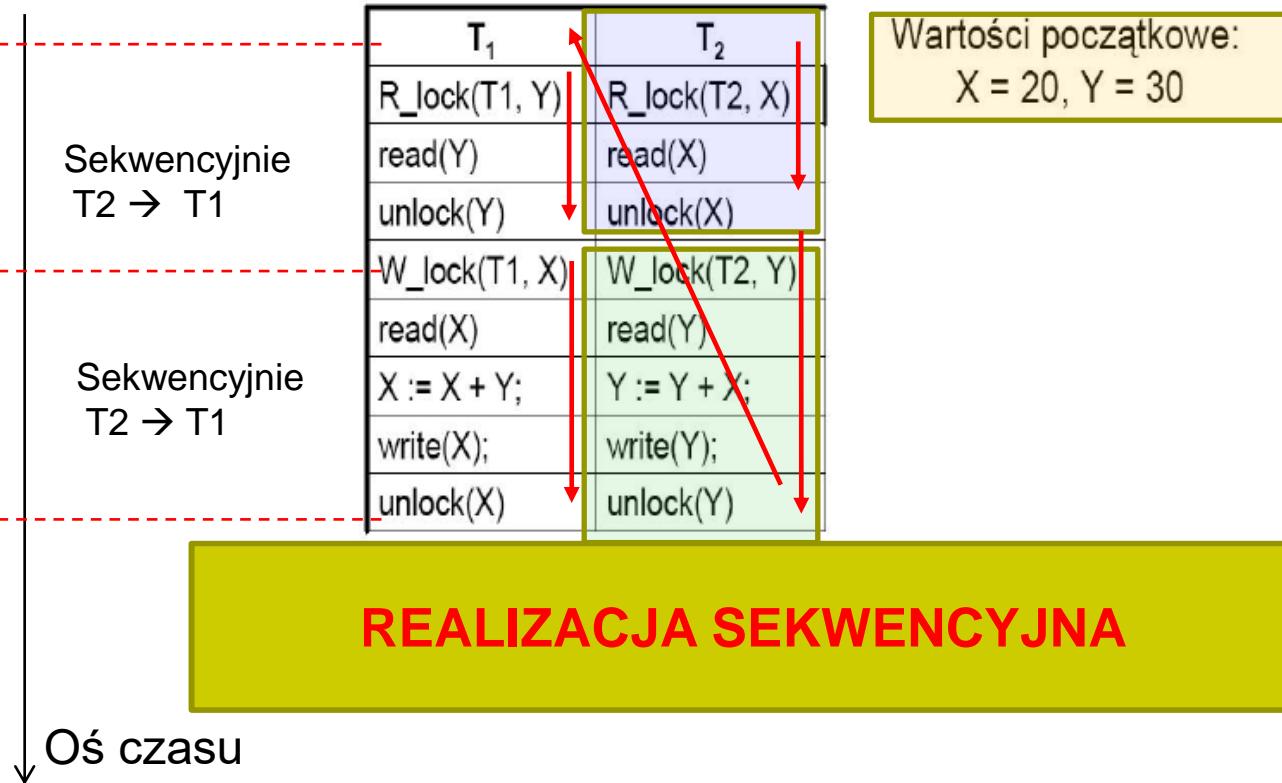
Realizacja sekwencyjna:

$T_1 \rightarrow T_2: X=50 \text{ i } Y=80$

1



algorytm blokowania (1)



Realizacja sekwencyjna:
 $T_2 \rightarrow T_1: X=70 \text{ i } Y=50$

2



algorytm blokowania (1)

	T ₁	T ₂
Sekwencyjnie	R_lock(T1, Y)	R_lock(T2, X)
	read(Y)	read(X)
	unlock(Y)	unlock(X)
Sekwencyjnie	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

↓ Oś czasu

Realizacja sekwencyjna:

T1 → T2: X=50 i Y=80

1

Realizacja sekwencyjna:

T2 → T1: X=70 i Y=50

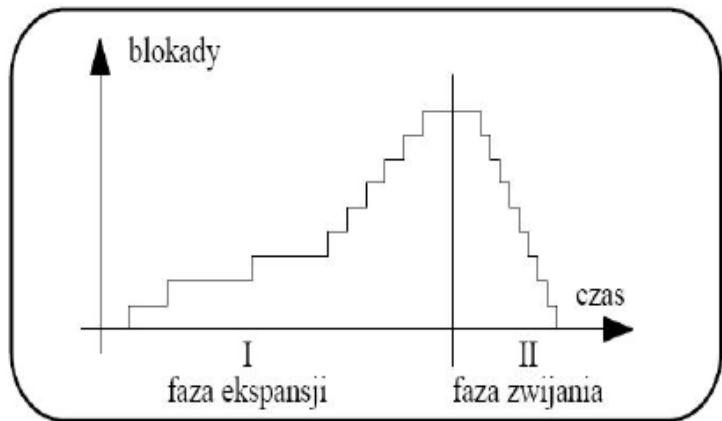
2

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:

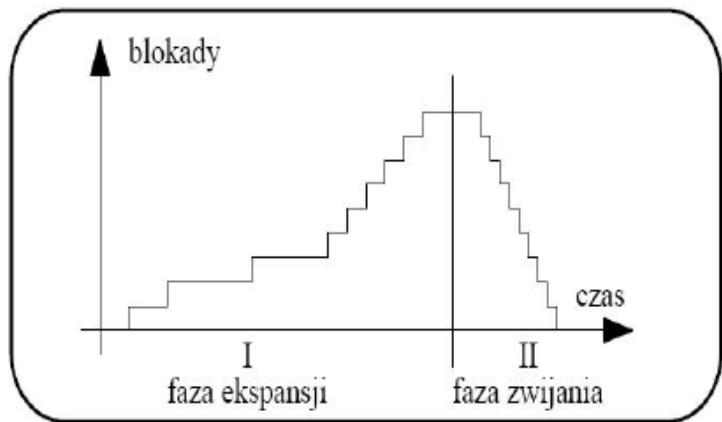
1. Każda operacja $read(X)$ danej transakcji T musi być poprzedzona operacją $R_lock(X, T)$ lub $W_lock(X, T)$
2. Każda operacja $write(X)$ danej transakcji T musi być poprzedzona operacją $W_lock(X, T)$
3. Operacje $unlock(x, T)$ dla danej transakcji T są wykonywane po zakończeniu wszystkich operacji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

Podstawowy algorytm blokowania:

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



algorytm blokowania dwufazowego (1)



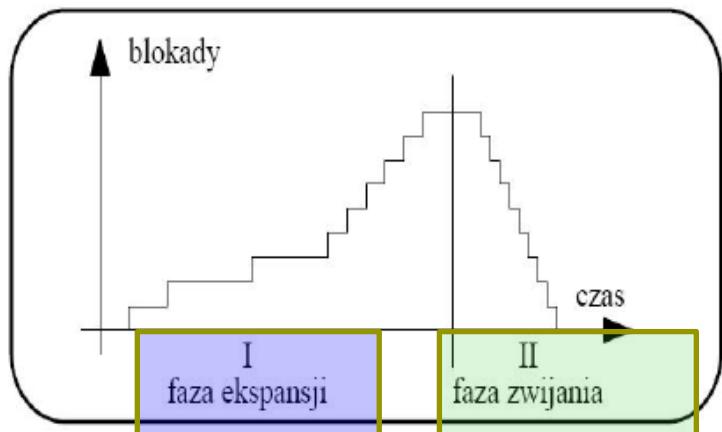
Algorytm podstawowy:

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)

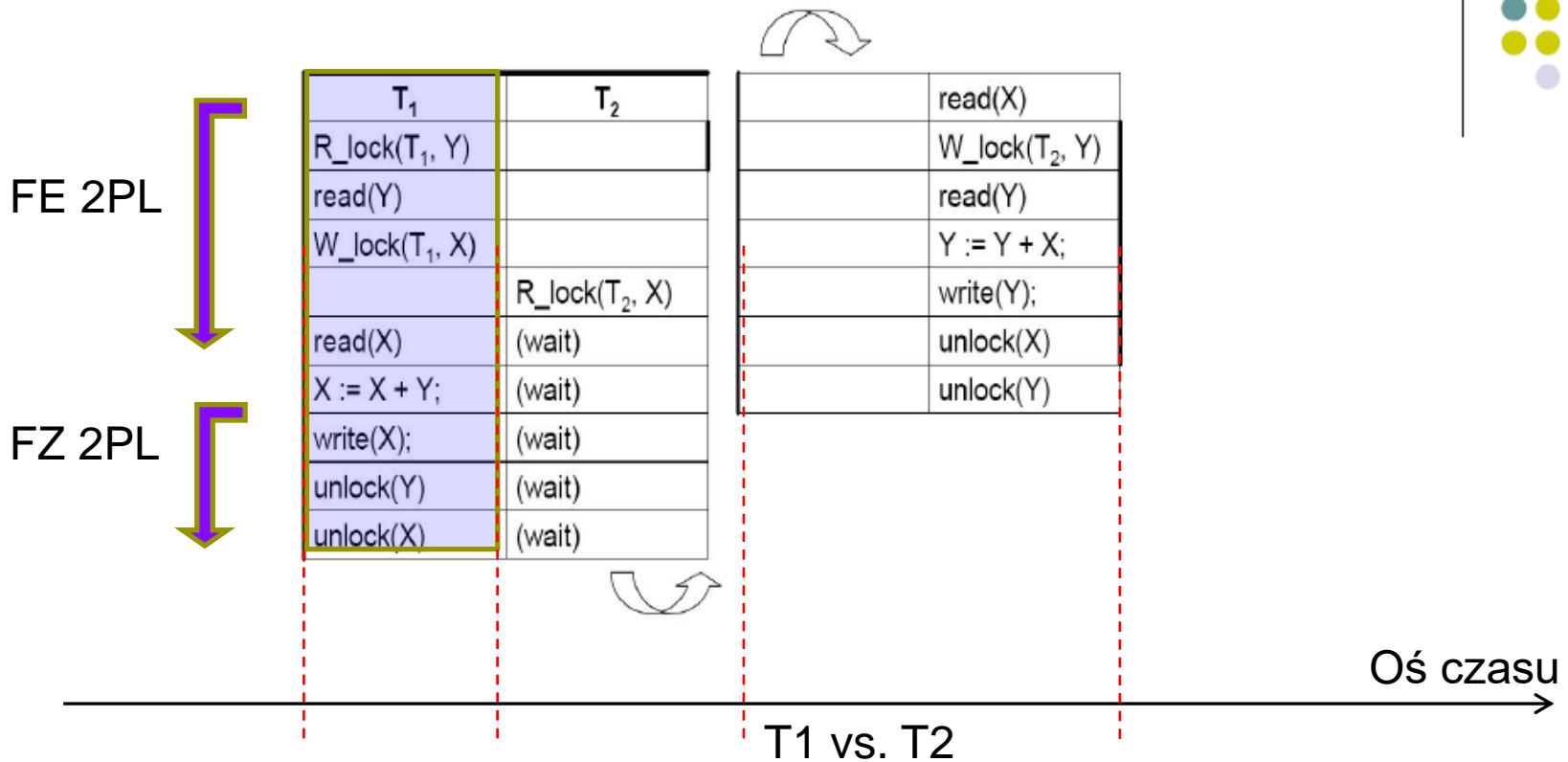
1

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

2



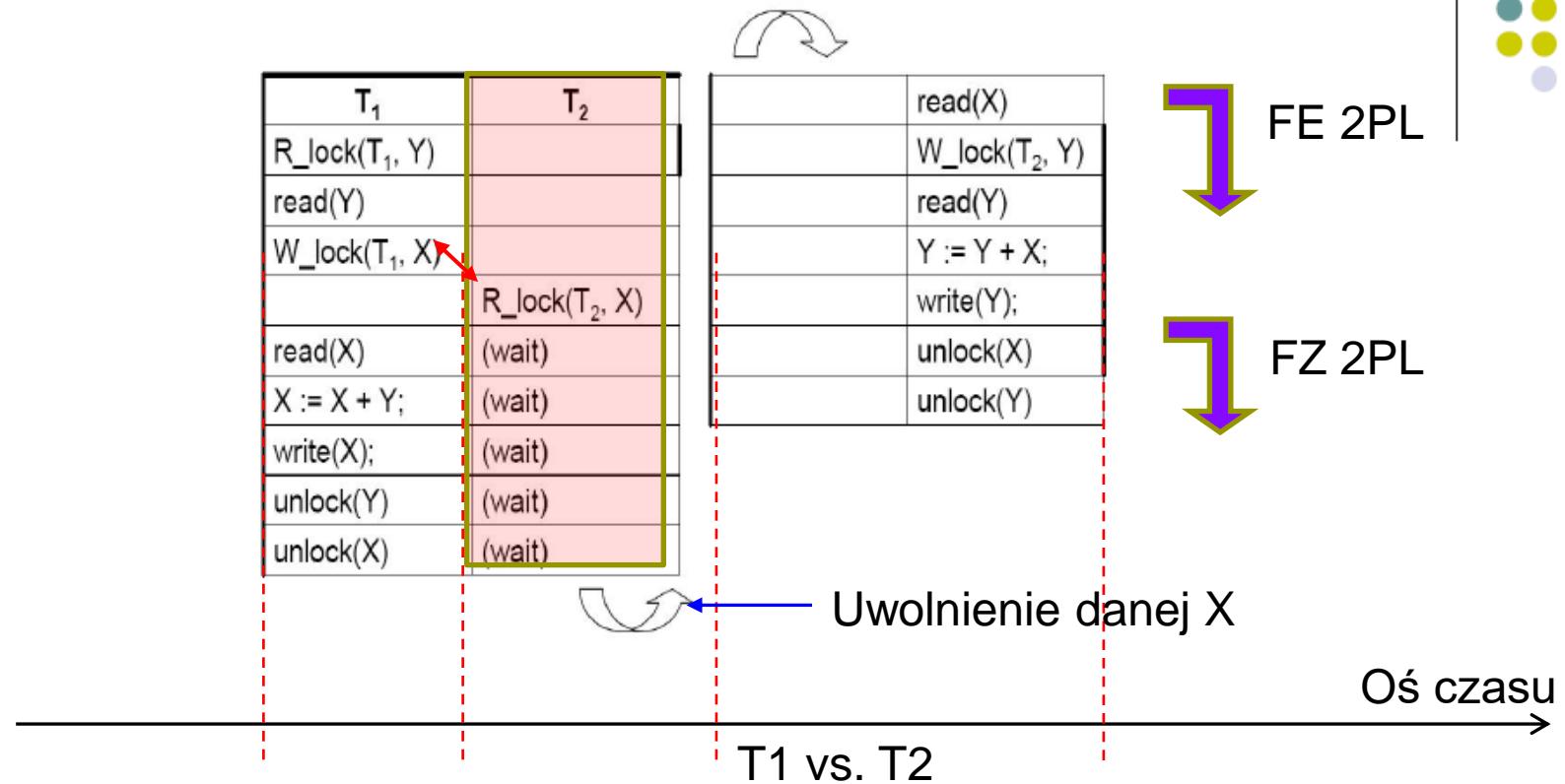
algorytm blokowania dwufazowego (3)



**Działanie algorytmu blokowania dwufazowego:
przykładowa realizacja transakcji T₁ i T₂**



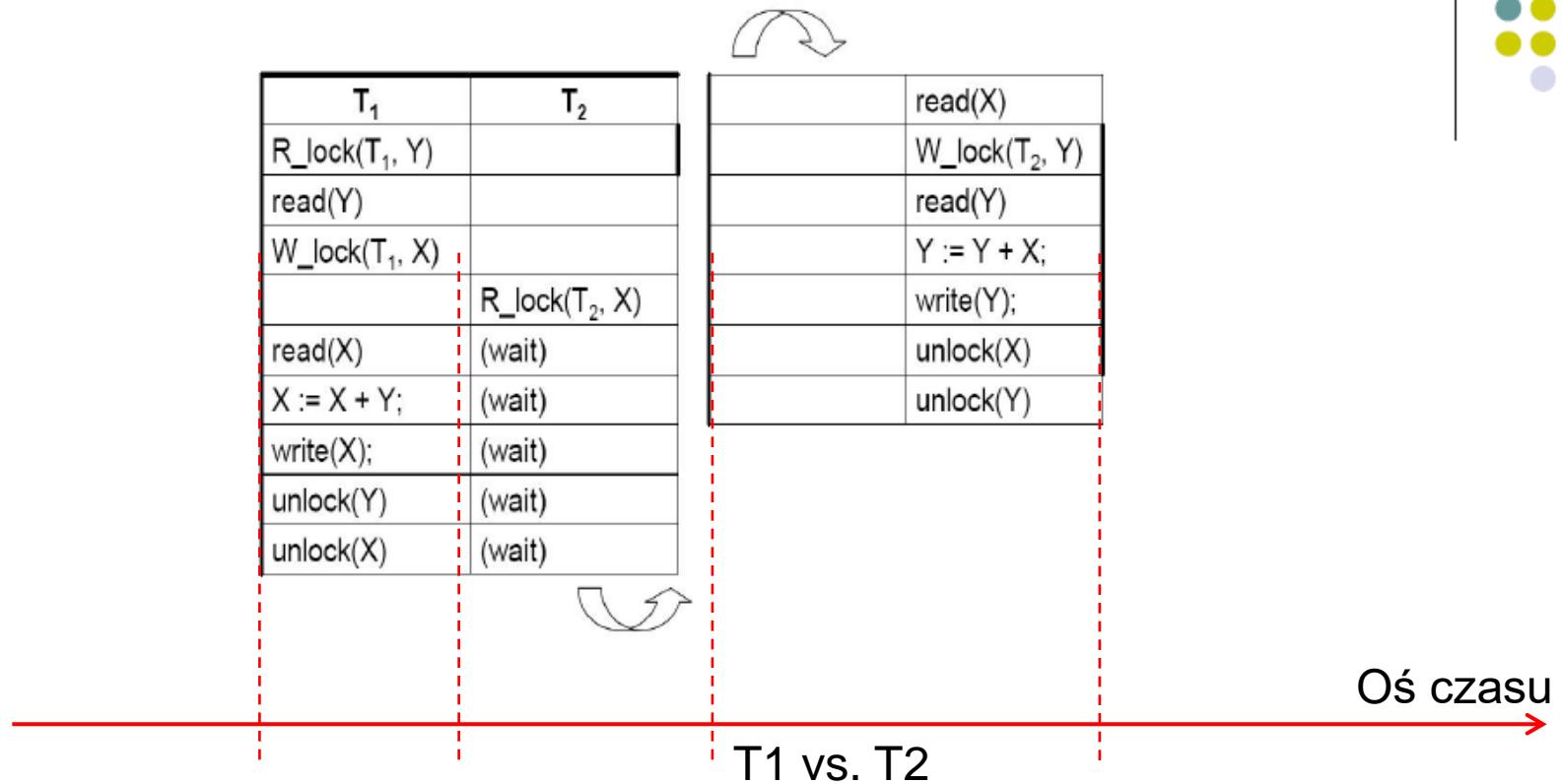
algorytm blokowania dwufazowego (3)



Działanie algorytmu blokowania dwufazowego: przykładowa realizacja transakcji T₁ i T₂



algorytm blokowania dwufazowego (3)



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

WNIOSK:

Przedstawiona realizacja współbieżna transakcji jest uszeregowalna

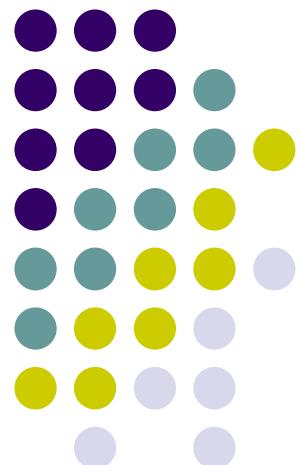


**KONIEC WYKŁADU
Cz. 1**

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	
Współbieżnie Sekwencyjnie	R_lock(T1, Y)	R_lock(T2, X)	
	read(Y)	read(X)	
	unlock(Y)	unlock(X)	
Współbieżnie Sekwencyjnie	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)	

Oś
czasu

Realizacja sekwencyjna:

T1 → T2: X=50 i Y=80

1

Realizacja sekwencyjna:

T2 → T1: X=70 i Y=50

2

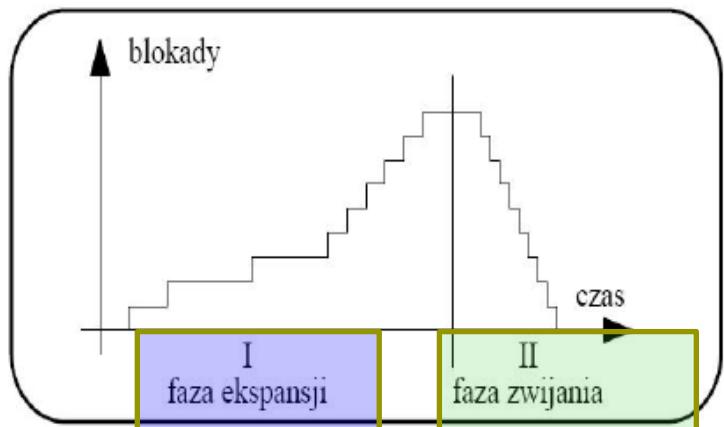
Realizacja współbieżna:

X=50 i Y=50

3



algorytm blokowania dwufazowego



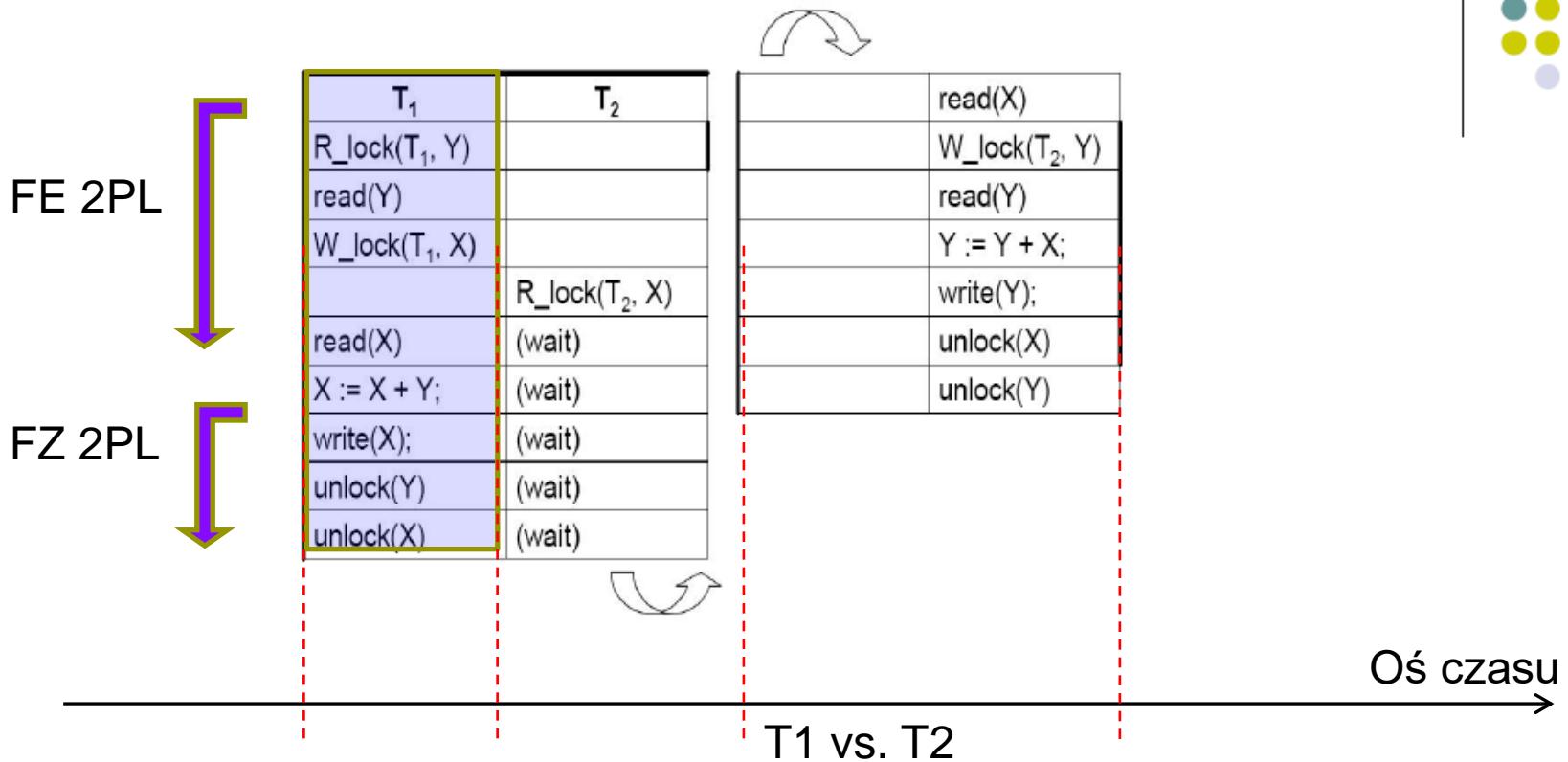
Algorytm podstawowy:

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**.



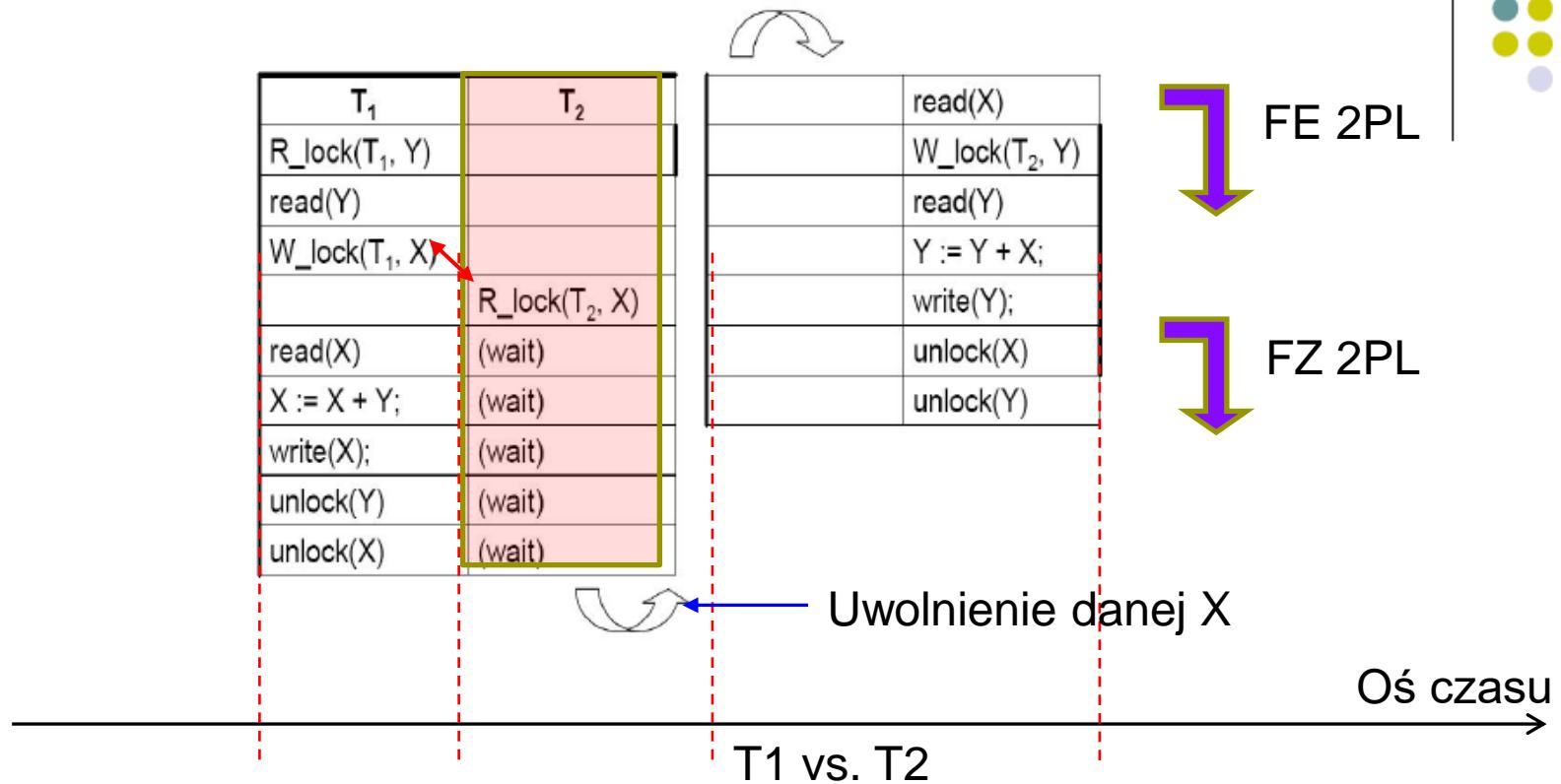
algorytm blokowania dwufazowego



**Działanie algorytmu blokowania dwufazowego:
przykładowa realizacja transakcji T1 i T2**



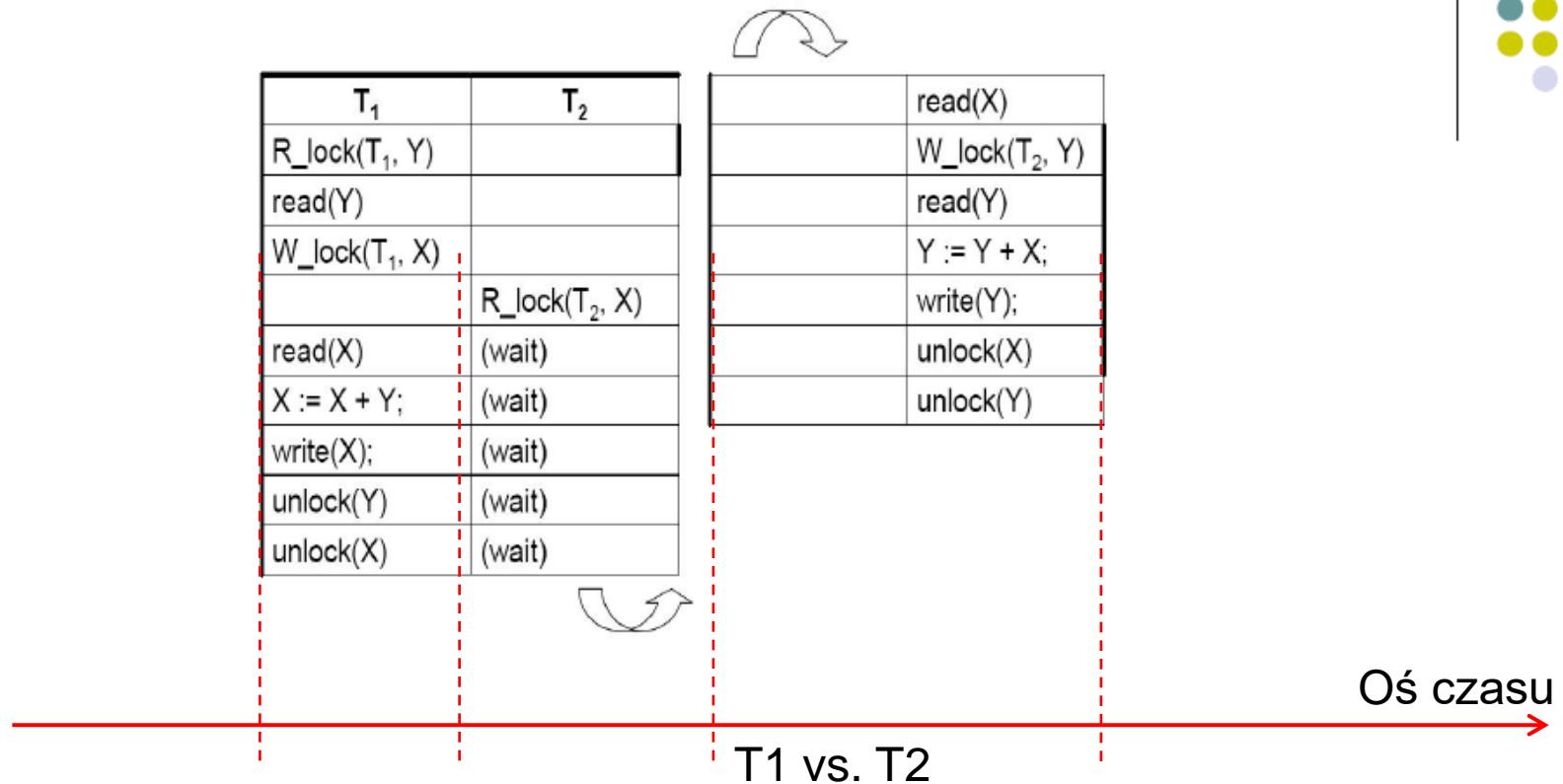
algorytm blokowania dwufazowego



Działanie algorytmu blokowania dwufazowego: przykładowa realizacja transakcji T₁ i T₂



algorytm blokowania dwufazowego (3)



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

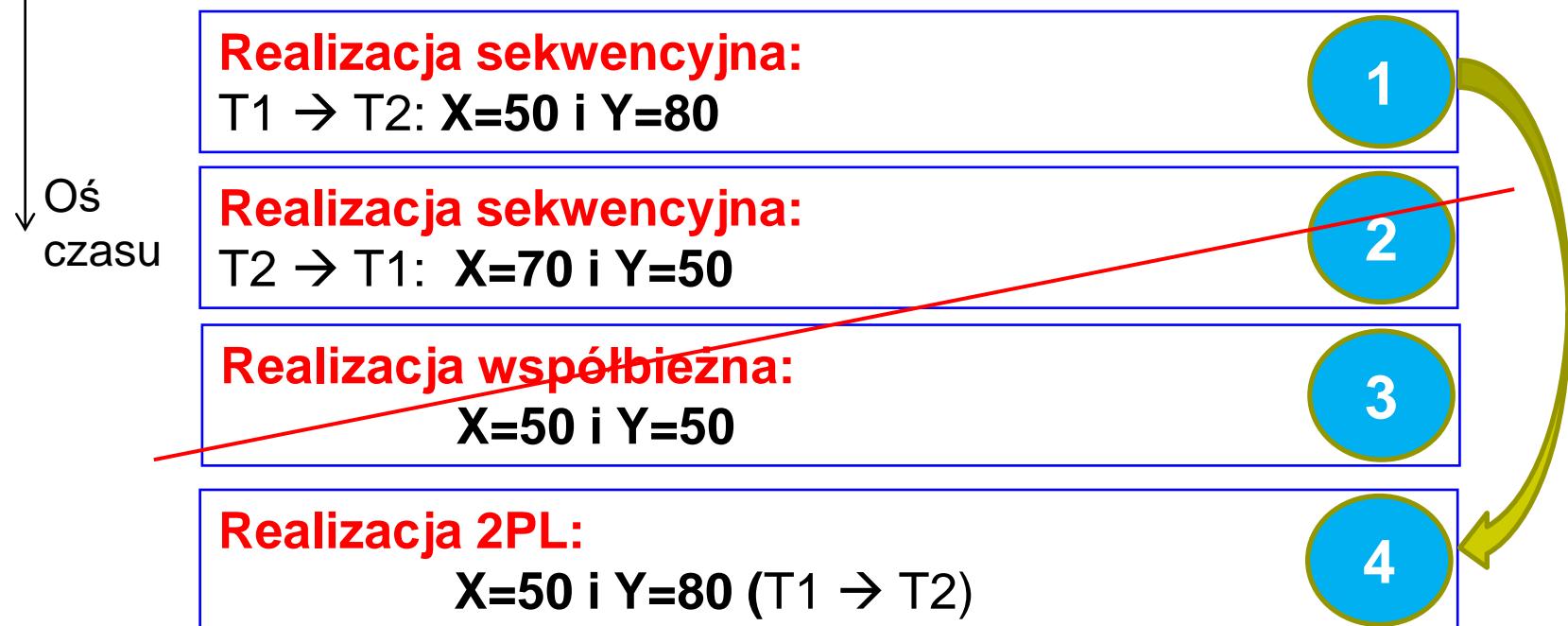
WNIOSK:

Przedstawiona realizacja współbieżna transakcji jest uszeregowalna



algorytm blokowania

	T ₁	T ₂	
Współbieżnie Sekwencyjnie	R_lock(T1, Y)	R_lock(T2, X)	
	read(Y)	read(X)	
	unlock(Y)	unlock(X)	
Współbieżnie Sekwencyjnie	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)	



zakleszczenie transakcji

Realizacja współbieżna
T1 i T2



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

Oś czasu

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

Oś czasu

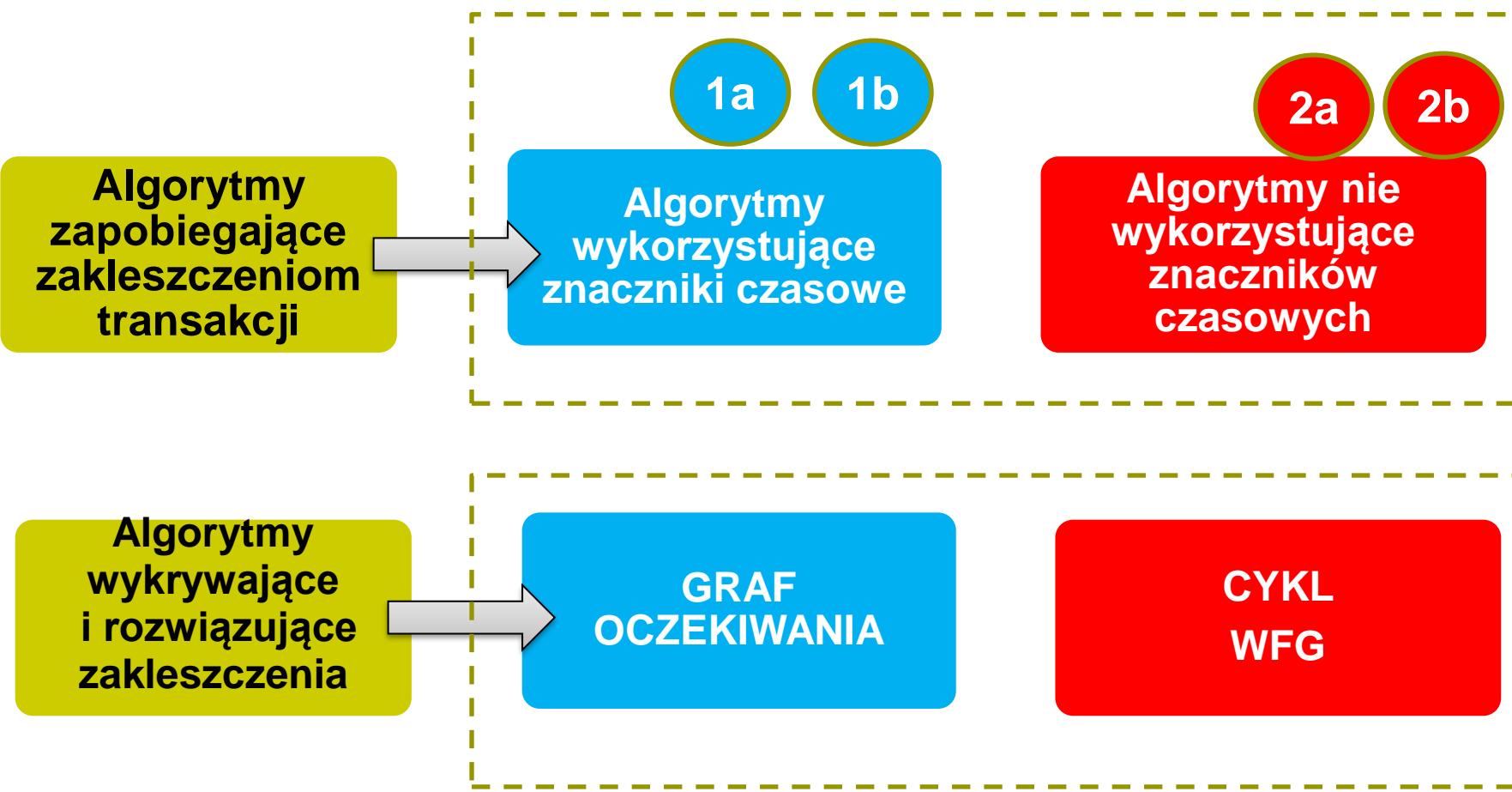
Dwa podejścia do problemu zakleszczenia transakcji:

- 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
- 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_i będzie wycofana i restartowana z tym samym znacznikiem czasowym. W przeciwnym wypadku T_i będzie czekać na zwolnienie blokady

1a

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
- 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_i będzie wycofana i restartowana z tym samym znacznikiem czasowym. W przeciwnym wypadku T_i będzie czekać na zwolnienie blokady

1a

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.
- **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

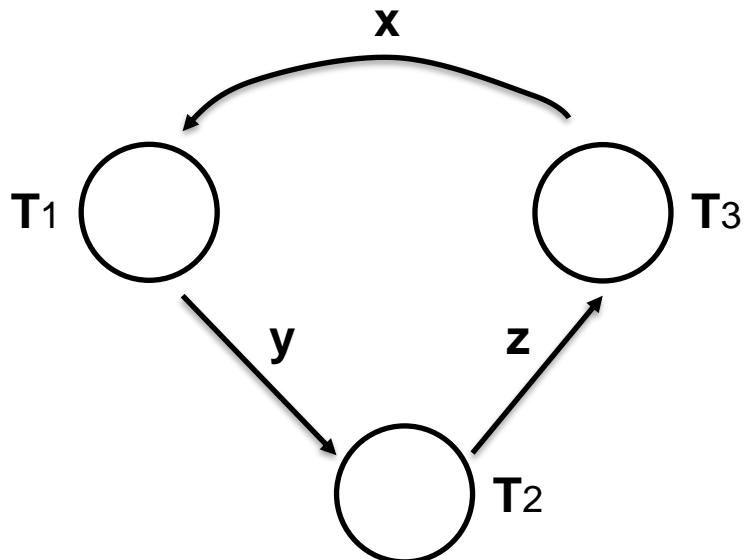
2a

2b



metody wykrywania i rozwiązywania zakleszczeń

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



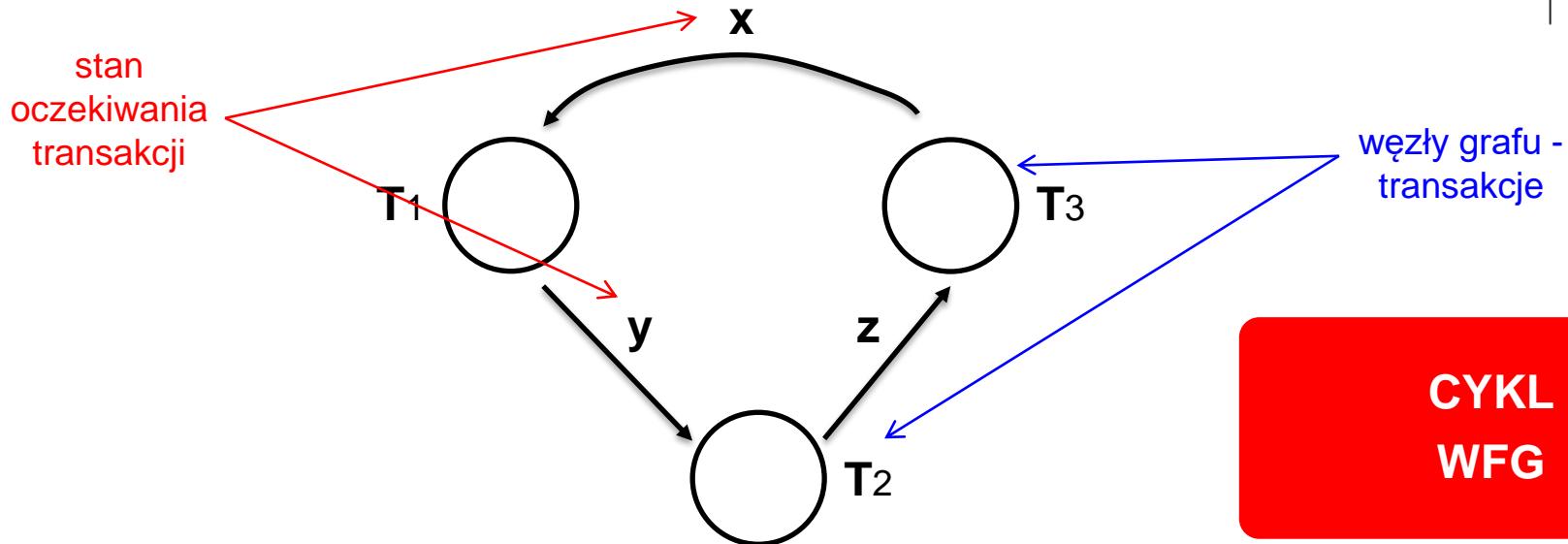
**GRAF
OCZEKIWANIA**

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.

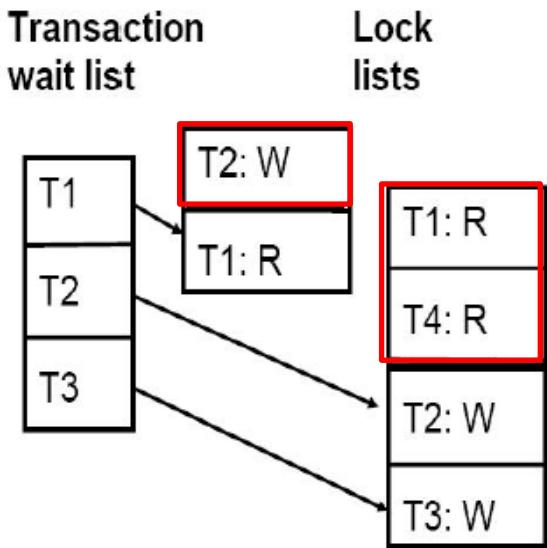
Obie listy mają postać $((T_i, m_i), \dots)$, gdzie T_i – oznacza transakcję, natomiast m_i – oznacza rodzaj blokady.

Do grafu WFG dodaje się łuk $T_i \rightarrow T_j$, jeżeli zachodzi warunek:

- transakcja T_j należy do listy transakcji, które uzyskały blokadę, natomiast T_i – jest na liście transakcji oczekujących, lub
- transakcja T_j jest przed transakcją T_i na liście transakcji oczekujących
- blokady „ m_i ” oraz „ m_j ” są niekompatybilne.



procedura wykrywania zakleszczenia (2)



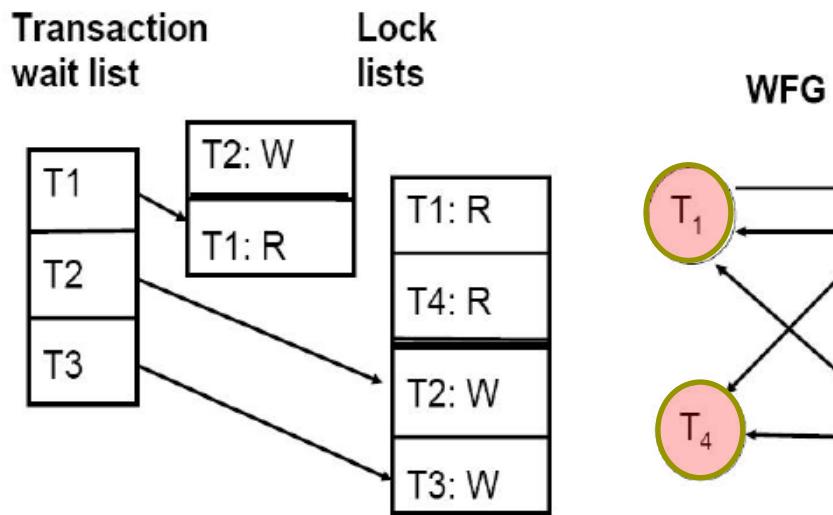
Działanie procedury wykrywania zakleszczenia

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.

- Założymy, 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 zauważyc, że w grafie uszeregowalności wystąpi cykl świadczący o nieuszeregowalności realizacji.



problem duchów (2)

- Łatwo zauważyc, 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

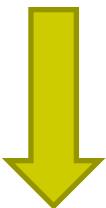
- ❑ Rozwiążanie problemu „duchów” wymaga wprowadzenia **blokad hierarchicznych** !

```
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/

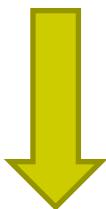
Wykorzystuje immanentną cechę organizacji bazy danych jaką jest hierarchia ziarnistości danych

- **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 precyzyji blokowania (zmniejszaniem liczby zablokowanych danych i zwiększeniem 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**

SQL

Structured Query Language

Karina Burnicka Viktoria Ługowska

19.04.2021

Część I

Omówienie języków zapytań

Definicja

Język zapytań (ang.) query language – język stosowany do formułowania zapytań w odniesieniu do baz danych. W odpowiedzi uzyskuje się zestawienia danych, zwane raportami. Najbardziej znane języki zapytań to SQL oraz xBase

Omówienie języków zapytań

Najpopularniejszy język zapytań

SQL

Strukturalny język zapytań używany do tworzenia, modyfikowania baz danych oraz do umieszczania i pobierania danych z baz danych. Opracowany przez IBM, wydany w 1974r. Od 1986 stał się oficjalnym standardem wspieranym przez ISO.

Inne języki zapytań

- ① Language INtegrated Query (LINQ)
- ② Query By Example (QBE)
- ③ XQuery (ang. XML query language)
- ④ xBase
- ⑤ POSTQUEL
- ⑥ Informix-4GL
- ⑦ System zarządzania relacyjną bazą danych (RDBMS)

Inne języki zapytań

- ① Language INtegrated Query (LINQ)
- ② Query By Example (QBE)
- ③ XQuery (ang. XML query language)
- ④ xBase
- ⑤ POSTQUEL
- ⑥ Informix-4GL
- ⑦ System zarządzania relacyjną bazą danych (RDBMS)

Inne języki zapytań

- ① Language INtegrated Query (LINQ)
- ② Query By Example (QBE)
- ③ XQuery (ang. XML query language)
- ④ xBase
- ⑤ POSTQUEL
- ⑥ Informix-4GL
- ⑦ System zarządzania relacyjną bazą danych (RDBMS)

Inne języki zapytań

- ① Language INtegrated Query (LINQ)
- ② Query By Example (QBE)
- ③ XQuery (ang. XML query language)
- ④ xBase
- ⑤ POSTQUEL
- ⑥ Informix-4GL
- ⑦ System zarządzania relacyjną bazą danych (RDBMS)

Inne języki zapytań

- ① Language INtegrated Query (LINQ)
- ② Query By Example (QBE)
- ③ XQuery (ang. XML query language)
- ④ xBase
- ⑤ POSTQUEL
- ⑥ Informix-4GL
- ⑦ System zarządzania relacyjną bazą danych (RDBMS)

Inne języki zapytań

- ① Language INtegrated Query (LINQ)
- ② Query By Example (QBE)
- ③ XQuery (ang. XML query language)
- ④ xBase
- ⑤ POSTQUEL
- ⑥ Informix-4GL
- ⑦ System zarządzania relacyjną bazą danych (RDBMS)

Inne języki zapytań

- ① Language INtegrated Query (LINQ)
- ② Query By Example (QBE)
- ③ XQuery (ang. XML query language)
- ④ xBase
- ⑤ POSTQUEL
- ⑥ Informix-4GL
- ⑦ System zarządzania relacyjną bazą danych (RDBMS)

Część II

Omówienie standardów ISO

2 Standardy ISO SQL

- ➊ SQL:86 – pierwszy oficjalny standard
- ➋ SQL:89 – ograniczenia integralności
- ➌ SQL:92 – typy danych, operatory, tworzenie tabel tymczasowych

- ① SQL:86 – pierwszy oficjalny standard
- ② SQL:89 – ograniczenia integralności
- ③ SQL:92 – typy danych, operatory, tworzenie tabel tymczasowych

- ① SQL:86 – pierwszy oficjalny standard
- ② SQL:89 – ograniczenia integralności
- ③ SQL:92 – typy danych, operatory, tworzenie tabel tymczasowych

- ➊ SQL:99 – obiektowość, typy tablicowe i strukturalne, zapytania rekurencyjne, wyrażenia regularne, osadzenie w Javie
- ➋ SQL:2003, SQL:2006 – obsługa danych XML
- ➌ SQL:2008 – klauzula FETCH, TRUNCATE TABLE
- ➍ SQL:2011 – nowe opcjonalne elementy składniowe SQL
- ➎ SQL:2016 - obsługa plików JSON, polimorficznych funkcji tabelowych
- ➏ SQL:2019 – tablice wielowymiarowe SQL/MDA

- ➊ SQL:99 – obiektowość, typy tablicowe i strukturalne, zapytania rekurencyjne, wyrażenia regularne, osadzenie w Javie
- ➋ SQL:2003, SQL:2006 – obsługa danych XML
- ➌ SQL:2008 – klauzula FETCH, TRUNCATE TABLE
- ➍ SQL:2011 – nowe opcjonalne elementy składniowe SQL
- ➎ SQL:2016 - obsługa plików JSON, polimorficznych funkcji tabelowych
- ➏ SQL:2019 – tablice wielowymiarowe SQL/MDA

- ① SQL:99 – obiektowość, typy tablicowe i strukturalne, zapytania rekurencyjne, wyrażenia regularne, osadzenie w Javie
- ② SQL:2003, SQL:2006 – obsługa danych XML
- ③ SQL:2008 – klauzula FETCH, TRUNCATE TABLE
- ④ SQL:2011 – nowe opcjonalne elementy składniowe SQL
- ⑤ SQL:2016 - obsługa plików JSON, polimorficznych funkcji tabelowych
- ⑥ SQL:2019 – tablice wielowymiarowe SQL/MDA

- ① SQL:99 – obiektowość, typy tablicowe i strukturalne, zapytania rekurencyjne, wyrażenia regularne, osadzenie w Javie
- ② SQL:2003, SQL:2006 – obsługa danych XML
- ③ SQL:2008 – klauzula FETCH, TRUNCATE TABLE
- ④ SQL:2011 – nowe opcjonalne elementy składniowe SQL
- ⑤ SQL:2016 - obsługa plików JSON, polimorficznych funkcji tabelowych
- ⑥ SQL:2019 – tablice wielowymiarowe SQL/MDA

- ① SQL:99 – obiektowość, typy tablicowe i strukturalne, zapytania rekurencyjne, wyrażenia regularne, osadzenie w Javie
- ② SQL:2003, SQL:2006 – obsługa danych XML
- ③ SQL:2008 – klauzula FETCH, TRUNCATE TABLE
- ④ SQL:2011 – nowe opcjonalne elementy składniowe SQL
- ⑤ SQL:2016 - obsługa plików JSON, polimorficznych funkcji tabelowych
- ⑥ SQL:2019 – tablice wielowymiarowe SQL/MDA

- ① SQL:99 – obiektowość, typy tablicowe i strukturalne, zapytania rekurencyjne, wyrażenia regularne, osadzenie w Javie
- ② SQL:2003, SQL:2006 – obsługa danych XML
- ③ SQL:2008 – klauzula FETCH, TRUNCATE TABLE
- ④ SQL:2011 – nowe opcjonalne elementy składniowe SQL
- ⑤ SQL:2016 - obsługa plików JSON, polimorficznych funkcji tabelowych
- ⑥ SQL:2019 – tablice wielowymiarowe SQL/MDA

Część III

Podstawowe zasady i postaci języka SQL

DDL

Język Definicji Danych – DDL (ang. Data Definition Language)

DML

Język Manipulacji Danymi – DML (ang. Data Manipulation Language)

DCL

Język Kontroli Danych - DCL (ang. Data Control Language)

DQL

Język Zapytań do Danych - DQL (ang. Data Query Language)

Język Definicji Danych - DDL

Podstawowe zasady i postaci języka SQL

CREATE

definiowanie obiektów w bazie danych

ALTER

modyfikowanie obiektów w bazie danych

DROP

usuwanie obiektów z bazy danych

Język Manipulacji Danymi - DML

Podstawowe zasady i postaci języka SQL

INSERT

wstawianie do tabeli nowych wierszy

UPDATE

modyfikowanie wierszy w tabeli

DELETE

usuwanie wierszy z tabeli

Język Kontroli Danych - DCL

Podstawowe zasady i postaci języka SQL

GRANT

przydzielanie prawa do danych

REVOKE

pozbawianie prawa do danych

DENY

bezwarkowe pozbawienie prawa do danych

SELECT

pobieranie danych z tabel

① Zapytania/Instrukcje

② Klauzule

③ Identyfikatory

④ Wyrażenia

⑤ Operatory

⑥ Predykaty

① Zapytania/Instrukcje

② Klauzule

③ Identyfikatory

④ Wyrażenia

⑤ Operatory

⑥ Predykaty

① Zapytania/Instrukcje

② Klauzule

③ Identyfikatory

④ Wyrażenia

⑤ Operatory

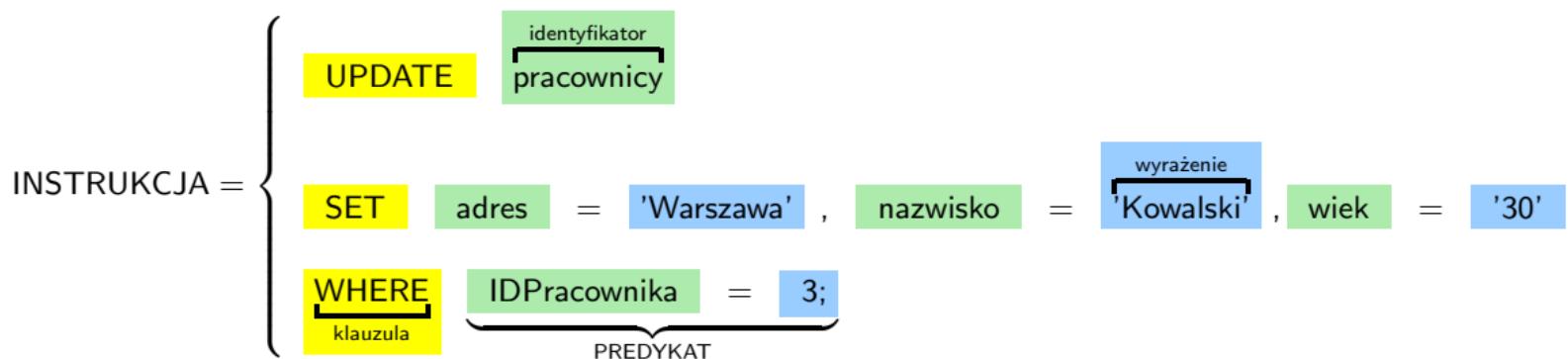
⑥ Predykaty

- ① Zapytania/Instrukcje
- ② Klauzule
- ③ Identyfikatory
- ④ Wyrażenia
- ⑤ Operatory
- ⑥ Predykaty

- ① Zapytania/Instrukcje
- ② Klauzule
- ③ Identyfikatory
- ④ Wyrażenia
- ⑤ Operatory
- ⑥ Predykaty

- ① Zapytania/Instrukcje
- ② Klauzule
- ③ Identyfikatory
- ④ Wyrażenia
- ⑤ Operatory
- ⑥ Predykaty

Elementy języka SQL



Część IV

Omówienie typów danych SQL

Rodzaj dopuszczalnych danych

Każda kolumna tabeli posiada ograniczenie co do typu wpisywanych do niej danych (zapobiega to umieszczeniu danych tekstowych w kolumnie numerycznej). Ułatwia to również poprawne sortowanie i pełni ważną rolę w optymalizacji przestrzeni zajmowanej na dysku. Dlatego bardzo ważne jest staranne dobieranie typów kolumn

Należy pamiętać !

Nie istnieją dwa identyczne SZBD

Typy danych

Typy danych znacznie różnią się między poszczególnymi SZBD (systemami zarządzania bazą danych). Dochodzi nawet do tego iż typ o takiej samej nazwie ma różne znaczenia w SZBD różnych producentów. Szczegółów zawsze należy szukać w dokumentacji bazy danych.

Tekstowe typy danych

Tekstowe typy danych nadają się do przechowywania informacji takich jak : nazwy, adresy, numery telefonów i kody pocztowe można podzielić je na typy o stałej długości i o zmiennej długości

- ① CHAR – tekst o stałej długości od 1 do 255 znaków (rozmiar musi zostać określony w trakcie definiowania tablicy)
- ② NCHAR – specjalna postać CHAR zaprojektowana dla znaków wielobajtowych lub unicode
- ③ NVARCHAR – specjalna postać typu TEXT dla znaków wielobajtowych lub unicode
- ④ TEXT – nazywany również: VARCHAR, LONG lub MEMO, tekst o zmiennej długości

Tekstowe typy danych

Tekstowe typy danych nadają się do przechowywania informacji takich jak : nazwy, adresy, numery telefonów i kody pocztowe można podzielić je na typy o stałej długości i o zmiennej długości

- ① CHAR – tekst o stałej długości od 1 do 255 znaków (rozmiar musi zostać określony w trakcie definiowania tablicy)
- ② NCHAR – specjalna postać CHAR zaprojektowana dla znaków wielobajtowych lub unicode
- ③ NVARCHAR – specjalna postać typu TEXT dla znaków wielobajtowych lub unicode
- ④ TEXT – nazywany również: VARCHAR, LONG lub MEMO, tekst o zmiennej długości

Tekstowe typy danych

Tekstowe typy danych nadają się do przechowywania informacji takich jak : nazwy, adresy, numery telefonów i kody pocztowe można podzielić je na typy o stałej długości i o zmiennej długości

- ① CHAR – tekst o stałej długości od 1 do 255 znaków (rozmiar musi zostać określony w trakcie definiowania tablicy)
- ② NCHAR – specjalna postać CHAR zaprojektowana dla znaków wielobajtowych lub unicode
- ③ NVARCHAR – specjalna postać typu TEXT dla znaków wielobajtowych lub unicode
- ④ TEXT – nazywany również: VARCHAR, LONG lub MEMO, tekst o zmiennej długości

Tekstowe typy danych

Tekstowe typy danych nadają się do przechowywania informacji takich jak : nazwy, adresy, numery telefonów i kody pocztowe można podzielić je na typy o stałej długości i o zmiennej długości

- ① CHAR – tekst o stałej długości od 1 do 255 znaków (rozmiar musi zostać określony w trakcie definiowania tablicy)
- ② NCHAR – specjalna postać CHAR zaprojektowana dla znaków wielobajtowych lub unicode
- ③ NVARCHAR – specjalna postać typu TEXT dla znaków wielobajtowych lub unicode
- ④ TEXT – nazywany również: VARCHAR, LONG lub MEMO, tekst o zmiennej długości

Numeryczne typy danych

Numeryczne typy danych przechowują liczby

- ① **BIT** – wartość jednobitowa (0 lub 1)
- ② **DECIMAL/NUMERIC** - wartość stało- lub zmiennoprzecinkowa z różnym poziomem precyzji
- ③ **FLOAT / NUMBER** – wartość zmiennoprzecinkowa
- ④ **INT / INTIGER** – czterobajtowa wartość całkowita obsługująca liczby z przedziału od 2147483648 do 2147483647
- ⑤ **REAL** – czterobajtowa wartość zmiennoprzecinkowa
- ⑥ **SMALLINT** – dwubajtowa wartość całkowita obsługująca liczby z przedziału 32768 do 32767
- ⑦ **TINYINT** – jednobajtowa wartość całkowita obsługująca liczby z przedziału od 0 do 255

Numeryczne typy danych

Numeryczne typy danych przechowują liczby

- ① BIT – wartość jednobitowa (0 lub 1)
- ② DECIMAL/NUMERIC - wartość stało- lub zmiennoprzecinkowa z różnym poziomem precyzji
- ③ FLOAT/ NUMBER – wartość zmiennoprzecinkowa
- ④ INT/ INTIGER – czterobajtowa wartość całkowita obsługująca liczby z przedziału od 2147483648 do 2147483647
- ⑤ REAL – czterobajtowa wartość zmiennoprzecinkowa
- ⑥ SMALLINT – dwubajtowa wartość całkowita obsługująca liczby z przedziału 32768 do 32767
- ⑦ TINYINT – jednobajtowa wartość całkowita obsługująca liczby z przedziału od 0 do 255

Numeryczne typy danych

Numeryczne typy danych przechowują liczby

- ① BIT – wartość jednobitowa (0 lub 1)
- ② DECIMAL/NUMERIC - wartość stało- lub zmiennoprzecinkowa z różnym poziomem precyzji
- ③ FLOAT/ NUMBER – wartość zmiennoprzecinkowa
- ④ INT/ INTIGER – czterobajtowa wartość całkowita obsługująca liczby z przedziału od 2147483648 do 2147483647
- ⑤ REAL – czterobajtowa wartość zmiennoprzecinkowa
- ⑥ SMALLINT – dwubajtowa wartość całkowita obsługująca liczby z przedziału 32768 do 32767
- ⑦ TINYINT – jednobajtowa wartość całkowita obsługująca liczby z przedziału od 0 do 255

Numeryczne typy danych

Numeryczne typy danych przechowują liczby

- ① BIT – wartość jednobitowa (0 lub 1)
- ② DECIMAL/NUMERIC - wartość stało- lub zmiennoprzecinkowa z różnym poziomem precyzji
- ③ FLOAT/ NUMBER – wartość zmiennoprzecinkowa
- ④ INT/ INTIGER – czterobajtowa wartość całkowita obsługująca liczby z przedziału od 2147483648 do 2147483647
- ⑤ REAL – czterobajtowa wartość zmiennoprzecinkowa
- ⑥ SMALLINT – dwubajtowa wartość całkowita obsługująca liczby z przedziału 32768 do 32767
- ⑦ TINYINT – jednobajtowa wartość całkowita obsługująca liczby z przedziału od 0 do 255

Numeryczne typy danych

Numeryczne typy danych przechowują liczby

- ① BIT – wartość jednobitowa (0 lub 1)
- ② DECIMAL/NUMERIC - wartość stało- lub zmiennoprzecinkowa z różnym poziomem precyzji
- ③ FLOAT/ NUMBER – wartość zmiennoprzecinkowa
- ④ INT/ INTIGER – czterobajtowa wartość całkowita obsługująca liczby z przedziału od 2147483648 do 2147483647
- ⑤ REAL – czterobajtowa wartość zmiennoprzecinkowa
- ⑥ SMALLINT – dwubajtowa wartość całkowita obsługująca liczby z przedziału 32768 do 32767
- ⑦ TINYINT – jednobajtowa wartość całkowita obsługująca liczby z przedziału od 0 do 255

Numeryczne typy danych

Numeryczne typy danych przechowują liczby

- ① BIT – wartość jednobitowa (0 lub 1)
- ② DECIMAL/NUMERIC - wartość stało- lub zmiennoprzecinkowa z różnym poziomem precyzji
- ③ FLOAT/ NUMBER – wartość zmiennoprzecinkowa
- ④ INT/ INTIGER – czterobajtowa wartość całkowita obsługująca liczby z przedziału od 2147483648 do 2147483647
- ⑤ REAL – czterobajtowa wartość zmiennoprzecinkowa
- ⑥ SMALLINT – dwubajtowa wartość całkowita obsługująca liczby z przedziału 32768 do 32767
- ⑦ TINYINT – jednobajtowa wartość całkowita obsługująca liczby z przedziału od 0 do 255

Numeryczne typy danych

Numeryczne typy danych przechowują liczby

- ① BIT – wartość jednobitowa (0 lub 1)
- ② DECIMAL/NUMERIC - wartość stało- lub zmiennoprzecinkowa z różnym poziomem precyzji
- ③ FLOAT/ NUMBER – wartość zmiennoprzecinkowa
- ④ INT/ INTIGER – czterobajtowa wartość całkowita obsługująca liczby z przedziału od 2147483648 do 2147483647
- ⑤ REAL – czterobajtowa wartość zmiennoprzecinkowa
- ⑥ SMALLINT – dwubajtowa wartość całkowita obsługująca liczby z przedziału 32768 do 32767
- ⑦ TINYINT – jednobajtowa wartość całkowita obsługująca liczby z przedziału od 0 do 255

Typy danych daty i czasu

- ① DATE – wartość daty
- ② DATETIME/ TIMESTAMP – wartość daty i czasu
- ③ SMALLDATETIME – wartość daty i czasu z dokładnością co do minuty
- ④ TIME – wartość czasu

Typy danych daty i czasu

- ① DATE – wartość daty
- ② DATETIME/ TIMESTAMP – wartość daty i czasu
- ③ SMALLDATETIME – wartość daty i czasu z dokładnością co do minuty
- ④ TIME – wartość czasu

Typy danych daty i czasu

- ① DATE – wartość daty
- ② DATETIME/ TIMESTAMP – wartość daty i czasu
- ③ SMALLDATETIME – wartość daty i czasu z dokładnością co do minuty
- ④ TIME – wartość czasu

Typy danych daty i czasu

- ① DATE – wartość daty
- ② DATETIME/ TIMESTAMP – wartość daty i czasu
- ③ SMALLDATETIME – wartość daty i czasu z dokładnością co do minuty
- ④ TIME – wartość czasu

Binarne typy danych

(dowolne dane takie jak grafika, multimedia czy dokumenty) są one najmniej przenośne między SZBD i nie korzysta się z nich często

- ➊ **BINARY** – dane binarne o stałej długości od 255 lub 8000 bajtów w zależności od implementacji
- ➋ **VARBINARY** – dane binarne o zmiennej długości od 255 do 8000 bajtów w zależności od implementacji
- ➌ **RAW** – dane binarne o stałej długości maksymalnie do 255 bajtów
- ➍ **LONG RAW** – dane binarne o zmiennej długości maksymalnie do 2GB

Binarne typy danych

(dowolne dane takie jak grafika, multimedia czy dokumenty) są one najmniej przenośne między SZBD i nie korzysta się z nich często

- ① **BINARY** – dane binarne o stałej długości od 255 lub 8000 bajtów w zależności od implementacji
- ② **VARBINARY** – dane binarne o zmiennej długości od 255 do 8000 bajtów w zależności od implementacji
- ③ **RAW** – dane binarne o stałej długości maksymalnie do 255 bajtów
- ④ **LONG RAW** – dane binarne o zmiennej długości maksymalnie do 2GB

Binarne typy danych

(dowolne dane takie jak grafika, multimedia czy dokumenty) są one najmniej przenośne między SZBD i nie korzysta się z nich często

- ① **BINARY** – dane binarne o stałej długości od 255 lub 8000 bajtów w zależności od implementacji
- ② **VARBINARY** – dane binarne o zmiennej długości od 255 do 8000 bajtów w zależności od implementacji
- ③ **RAW** – dane binarne o stałej długości maksymalnie do 255 bajtów
- ④ **LONG RAW** – dane binarne o zmiennej długości maksymalnie do *2GB*

Binarne typy danych

(dowolne dane takie jak grafika, multimedia czy dokumenty) są one najmniej przenośne między SZBD i nie korzysta się z nich często

- ① **BINARY** – dane binarne o stałej długości od 255 lub 8000 bajtów w zależności od implementacji
- ② **VARBINARY** – dane binarne o zmiennej długości od 255 do 8000 bajtów w zależności od implementacji
- ③ **RAW** – dane binarne o stałej długości maksymalnie do 255 bajtów
- ④ **LONG RAW** – dane binarne o zmiennej długości maksymalnie do 2GB

Table: produkty

Columns:

prod_id	char(10)
dost_id	char(10)
prod_nazwa	char(255)
prod_cena	decimal(8,2)
prod_opis	text

Część V

Tworzenie, usuwanie i modyfikacja schematu tabel i danych

CREATE

Query X

File Edit View Insert Tools Window Help

CREATE TABLE czytelnicy (IDCzytelnika int,
Nazwisko VARCHAR(200),
Imie VARCHAR(200),
PRIMARY KEY(IDCzytelnika));

SCHEMAS

Filter objects

biblioteka

Tables

czytelnicy

- Columns
- Indexes
- Foreign Keys
- Triggers

Views

Stored Procedures

Functions

Table: **czytelnicy**

Columns:

IDCzytelnika	int PK
Nazwisko	varchar(200)
Imie	varchar(200)

CREATE

Query 1 x

CREATE TABLE ksiazki (sygnatura smallint PRIMARY KEY,
autor VARCHAR(100),
tytul VARCHAR(100),
rodzaj_literacki VARCHAR(50));

SCHEMAS

Filter objects

biblioteka

Tables

- czytelnicy
- ksiazki
 - Columns
 - Indexes
 - Foreign Keys
 - Triggers

Views

Stored Procedures

Functions

Table: ksiazki

Columns:

sygnatura	smallint PK
autor	varchar(100)
tytul	varchar(100)
rodzaj_literacki	varchar(50)

INSERT

Query 1 x

The screenshot shows a MySQL Workbench interface. The title bar says "Query 1". Below it is a toolbar with various icons for file operations, search, and navigation. A dropdown menu "Limit to 1000 rows" is open. The main area contains the following SQL code:

```
1 •  INSERT INTO czytelnicy(IDCzytelnika, Nazwisko, Imie)
2      VALUES ( 1, 'Kowal', 'Piotr'),
3              ( 2, 'Czajka', 'Karolina'),
4              ( 3, 'Kot', 'Tomasz');
```

INSERT

Query 1 ×

The screenshot shows a MySQL Workbench interface with a query editor titled "Query 1". The query itself is:

```
1 • INSERT INTO ksiazki
2     VALUES (1, 'H. Sienkiewicz', 'Potop', 'powieść historyczna'),
3             (2, 'J. Słowacki', 'Balladyna', 'dramat'),
4             (3, 'A. Mickiewicz', 'Pan Tadeusz', NULL),
5             (4, NULL, 'Ferydydurka', 'powieść awangardowa');
```

Below the query, the results are displayed in a table:

sygnatura	autor	tytul	rodzaj_literacki
1	H. Sienkiewicz	Potop	powieść historyczna
2	J. Słowacki	Balladyna	dramat
3	A. Mickiewicz	Pan Tadeusz	NULL
4	NULL	Ferydydurka	powieść awangardowa
NULL	NULL	NULL	NULL

SELECT

Query ×

Limit to 1000 rows

1 • SELECT * FROM czytelnicy;

	IDCzytelnika	Nazwisko	Imie
▶	1	Kowal	Piotr
	2	Czajka	Karolina
	3	Kot	Tomasz
*	NULL	NULL	NULL

SELECT

Query 1 ×

The screenshot shows a MySQL Workbench interface. The top bar has tabs for 'Query 1' and 'Results'. Below the tabs is a toolbar with various icons for database management. The main area contains a SQL query and its execution results.

```
1 •   SELECT * FROM ksiazki
2 WHERE autor IN ('J. Słowacki', 'H. Sienkiewicz');
```

	sygnatura	autor	tytuł	rodzaj_literacki
▶	1	H. Sienkiewicz	Potop	powieść historyczna
▶	2	J. Słowacki	Balladyna	dramat
*	NULL	NULL	NULL	NULL

UPDATE

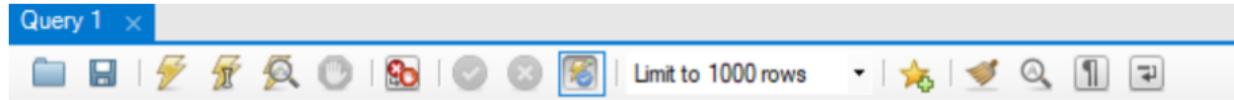
Query 1 ×

The screenshot shows a MySQL Workbench interface. The top bar has a 'Query 1' tab and various toolbar icons. Below the toolbar, there is a dropdown menu set to 'Limit to 1000 rows'. The main area contains an SQL query and its execution results.

```
1 • UPDATE czytelnicy
2      SET Nazwisko = 'Wróbel'
3      WHERE IDCzytelnika = 2;
```

	IDCzytelnika	Nazwisko	Imię
▶	1	Kowal	Piotr
	2	Wróbel	Karolina
	3	Kot	Tomasz
*	NULL	NULL	NULL

UPDATE



```
1 • UPDATE ksiazki
2     SET autor = 'W. Gombrowicz'
3     WHERE sygnatura > 3;
```

DELETE

Query X

1 • `DELETE FROM czytelnicy`
2 `WHERE IDCzytelnika < 2;`

	IDCzytelnika	Nazwisko	Imie
▶	2	Wróbel	Karolina
●	3	Kot	Tomasz
*	NULL	NULL	NULL

DELETE

Query 1 ×

The screenshot shows a MySQL Workbench interface with a query editor titled "Query 1". The query is:

```
1 •    DELETE FROM ksiazki
2      WHERE sygnatura BETWEEN 2 AND 4;
```

The results pane displays a table with four columns: "sygnatura", "autor", "tytul", and "rodzaj_literacki". There are two rows of data:

	sygnatura	autor	tytul	rodzaj_literacki
▶	1	H. Sienkiewicz	Potop	powieść historyczna
*	NULL	NULL	NULL	NULL

ALTER

The screenshot shows a MySQL Workbench interface with a 'Query' tab selected. The toolbar contains various icons for database management. The SQL editor window displays the following code:

```
1 •  ALTER TABLE czytelnicy
2      ADD data_urodzenia date;
3
4 •  ALTER TABLE czytelnicy
5      ADD adres VARCHAR(50);
```

Table: **czytelnicy**

Columns:

IDCzytelnika	int PK
Nazwisko	varchar(200)
Imie	varchar(200)
data_urodzenia	date
adres	varchar(50)

Table: wydawnictwa

Columns:

ID_Wydawnictwa	tinyint PK
Nazwa_wydawnictwa	varchar(100)
Kod_pocztowy	varchar(5)
miasto	varchar(50)
ulica	varchar(50)

DROP

The screenshot shows the MySQL Workbench interface with a 'Query' tab selected. The query editor contains the command: `1 • DROP TABLE czytelnicy;`. Below the editor, the 'Action Output' pane displays the execution results:

#	Time	Action	Message	Duration / Fetch
1	10:28:49	DROP TABLE czytelnicy;	0 row(s) affected	0.016 sec

Część VI

Zakładanie i usuwanie indeksów

Co to są indeksy i do czego służą?

Zapytanie	Czas (ms)	Data Output
SELECT * FROM person;	3562	10000000000
SELECT * FROM person WHERE last_name = 'Smith';	4261	100520
SELECT * FROM person WHERE first_name = 'Emma';	4066	49767
SELECT * FROM person WHERE first_name = 'Julie' AND last_name = 'Andrews';	514	46

Co to są indeksy i do czego służą?

Zapytanie	Czas (ms)	INDEKS
SELECT * FROM person WHERE first_name = 'Emma';	508	CREATE INDEX person_first_name_idx ON person (first_name);
SELECT * FROM person WHERE firts_name = 'Julie' AND last_name = 'Andrews';	26	CREATE INDEX person_first_name_last_name_idx ON person (first_name, last_name);

Zakładanie indeksów

Query X

Limit to 1000 rows

1 • CREATE INDEX identyfikator ON czytelnicy(IDCzytelnika);

SCHEMAS

Filter objects

biblioteka

- Tables
 - czytelnicy
 - Columns
 - Indexes
 - PRIMARY
 - identyfikator
 - Foreign Keys
 - Triggers
 - Views
 - Stored Procedures
 - Functions

Index: identyfikator

Definition:

Type	BTREE
Unique	No
Visible	Yes
Columns	IDCzytelnika

Karina Burnicka Viktoria Ługowska

SQL

19.04.2021 43 / 54

Zakładanie unikalnych indeksów

Query X

|   |   |   |  | Limit to 1000 rows |  |  |  |  |  |  | 

1 • CREATE UNIQUE INDEX dane_osobowe ON czytelnicy(Nazwisko, Imie);

SCHEMAS

Filter objects

biblioteka

- Tables
 - czytelnicy
 - Columns
 - Indexes
 - PRIMARY
 - dane_osobowe
 - identyfikator
 - Foreign Keys
 - Triggers
 - Views
 - Stored Procedures
 - Functions

Index: dane_osobowe

Definition:

Type	BTREE
Unique	Yes
Visible	Yes
Columns	Nazwisko Imie

Usuwanie indeksów

The screenshot shows a MySQL Workbench interface. The title bar says "Query". The toolbar has various icons for database management. The main area contains the following SQL code:

```
1 • ALTER TABLE czytelnicy
2   DROP INDEX identyfikator;
```

Część VII

Omówienie więzów spójności

Więzy spójności

dzięki więzom spójności nie można tak zmodyfikować danych by straciły one spójność. Są zbiorem zasad nałożonych na tabele w bazie danych

① Primary Key

- ② Unique
- ③ NOT NULL
- ④ Check
- ⑤ Foreign key
- ⑥ DEFAULT
- ⑦ INDEX

- ① Primary Key
- ② Unique
- ③ NOT NULL
- ④ Check
- ⑤ Foreign key
- ⑥ DEFAULT
- ⑦ INDEX

- ① Primary Key
- ② Unique
- ③ NOT NULL
- ④ Check
- ⑤ Foreign key
- ⑥ DEFAULT
- ⑦ INDEX

Charakterystyka więzów spójności

- ① Primary Key
- ② Unique
- ③ NOT NULL
- ④ Check
- ⑤ Foreign key
- ⑥ DEFAULT
- ⑦ INDEX

Charakterystyka więzów spójności

- ① Primary Key
- ② Unique
- ③ NOT NULL
- ④ Check
- ⑤ Foreign key
- ⑥ DEFAULT
- ⑦ INDEX

- ① Primary Key
- ② Unique
- ③ NOT NULL
- ④ Check
- ⑤ Foreign key
- ⑥ DEFAULT
- ⑦ INDEX

- ① Primary Key
- ② Unique
- ③ NOT NULL
- ④ Check
- ⑤ Foreign key
- ⑥ DEFAULT
- ⑦ INDEX

Przykład więzów spójności

```
CREATE TABLE Pracownicy(
    ID_prac NUMERIC(7,0) PRIMARY KEY,
    pesel NUMERIC(11,0) UNIQUE NOT NULL,
    Imię VARCHAR(25) NOT NULL,
    Nazwisko VARCHAR(25) NOT NULL,
    Szef NUMERIC(7,0)
        CONSTRAINT Szef_ko REFERENCES Pracownicy,
    Nazwa_działu VARCHAR(35),
    Miejsce VARCHAR(35),
    Zarobki NUMERIC(2,8),
    Premia NUMERIC(8,2),
    CONSTRAINT Dz_ko FOREIGN KEY (Nazwa_działu, Miejsce)
        REFERENCES Działy (Nazwa, Miasto),
    CONSTRAINT P_ck CHECK(0.1*Zarobki <= Premia AND Premia <=0.5*Zarobki)
);
```

Część VIII

Omówienie zatwierdzania zmian w bazie

Commit

Commit – ostateczne zatwierdzenie tymczasowo dokonanych zmian. Najpopularniejszym użyciem jest zakończenie transakcji.

ROLLBACK

W czasie trwania transakcji, w przeciwnieństwie do commit poleceniem służącym do anulowania niezatwierdzonych zmian jest ROLLBACK

Commit używany jest też w systemach kontroli wersji. Oznacza wtedy zatwierdzenie tymczasowo wprowadzonych zmian w kodzie źródłowym i wprowadzenie ich do systemu.

Część IX

Literatura



Sebastian Lachecinski

Składowanie i przetwarzanie danych temporalnych w świetle wymagań standardu języka SQL ISO
<https://www.researchgate.net/publication/344692829>



Konrad Zdanowski

Bazy danych – wykład ósmy Indeksy
https://www.impan.pl/~kz/DB/KZ_BD_w08.pdf



Ben Forta

SQL Server i T-SQL w mgieniu oka
Wydawnictwo Helion



Internet

<http://dev.cdur.pl/Artykuly/Zakladanie-indeksu-w-SQL-Server>
<https://jsystems.pl/blog/artykul.html?id=427>



SQL Index — Indexes in SQL — Database Index

<https://www.youtube.com/watch?v=fsG1XaZEa78&t=65s>

Dziękujemy za uwagę!

Karina Burnicka Viktoria Ługowska

