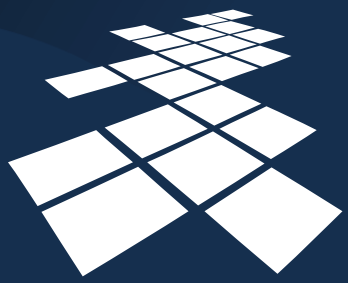
Bazy danych

Algorytmy zarządzania współbieżnym wykonywaniem transakcji
część II

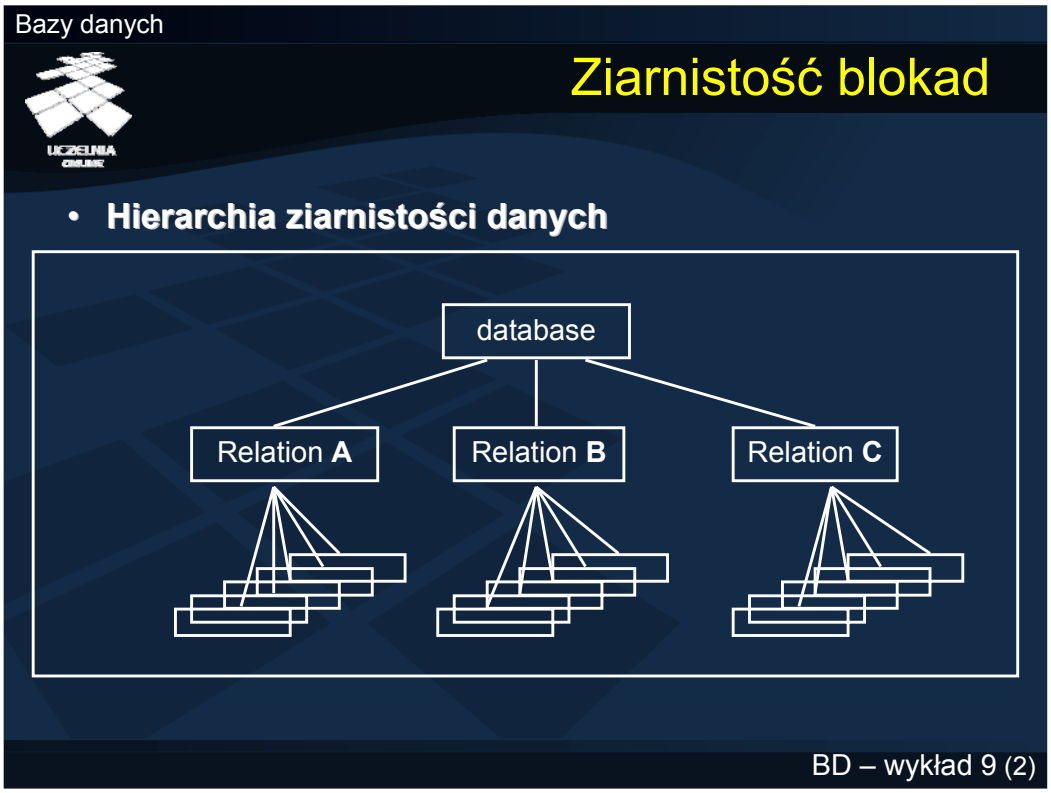
Wykład przygotował:
Tadeusz Morzy



UCZELNIA
ONLINE

BD – wykład 9

Kontynuujemy prezentację i omówienie algorytmów zarządzania współbieżnym wykonywaniem transakcji. Przedstawimy hierarchiczny algorytm blokowania dwu-fazowego, będący podstawowym algorytmem zarządzania współbieżnym wykonywaniem transakcji w komercyjnych systemach baz danych, a następnie, omówimy algorytm porządkowania transakcji według etykiet czasowych i algorytm optymistyczny.



Jak już wspomnieliśmy na zakończenie poprzedniego wykładu, nie istnieje żaden mechanizm blokowania, realizowany na poziomie blokad rekordów, który gwarantowałby rozwiązanie problemu rekordów „duchów". Rozwiązanie tego problemu wymaga wprowadzenia mechanizmu, który pozwalałby zakładać blokady na różnych poziomach organizacji danych – innymi słowy, pozwalałby zakładać blokady o różnej ziarnistości (baza danych, relacja, strona, rekord). Z drugiej strony może to rodzić pewien problem. Załóżmy, że dana jest relacja *Pracownicy* i że założyliśmy blokadę dla odczytu rekordu opisującego pracownika o nazwisku „Dziandziak". Z drugiej strony, inny użytkownik postanowił założyć blokadę do zapisu całej relacji *Pracownicy,* gdyż zamierza uaktualnić adres zamieszkania pracownika o nazwisku „Nowak". Czy te dwie blokady są kompatybilne czy tez nie? Rozwiązaniem tego problemu jest połączenie mechanizmu blokad fizycznych, zakładanych na poziomie rekordów (krotek) bazy danych, z blokadami intencyjnymi, nazywanymi również blokadami predykatowymi, zakładanymi na poziomie bazy danych i relacji. Algorytm, który łączy te dwa typy blokad nazywamy hierarchicznym algorytmem blokowania dwufazowego. Wykorzystuje on immanentną cechę organizacji bazy danych jaką jest hierarchia ziarnistości danych. Zauważmy, że baza danych posiada strukturę drzewa. Korzeniem tej struktury jest baza danych, która składa się ze zbioru relacji, które, z kolei, składają się ze zbioru krotek.

Hierarchiczna struktura bazy danych pozwala na stosunkowo efektywną implementację hierarchicznego algorytmu blokowania.

Jednostka blokowania

- Wybór jednostki blokowania jest kompromisem między stopniem współbieżności systemu, a narzutem systemowym
  - Współbieżność pracy systemu rośnie wraz ze zwiększaniem precyzji blokowania (zmniejszaniem liczby zablokowanych danych i zwiększaniem liczby blokad)
  - Precyzyjne blokowanie jest kosztowne dla złożonych transakcji, które wymagają długiego czasu utrzymywania dużej liczby blokad
  - Blokowanie dużych jednostek danych wspiera złożone transakcje o dużej liczbie operacji, kosztem prostych transakcji o niewielkiej liczbie operacji
- **Konkluzja:** potrzebny jest protokół, który będzie wspierał obydwa rodzaje transakcji, to znaczy taki, który będzie umożliwiał równoczesne zakładanie blokad na różnych jednostkach danych

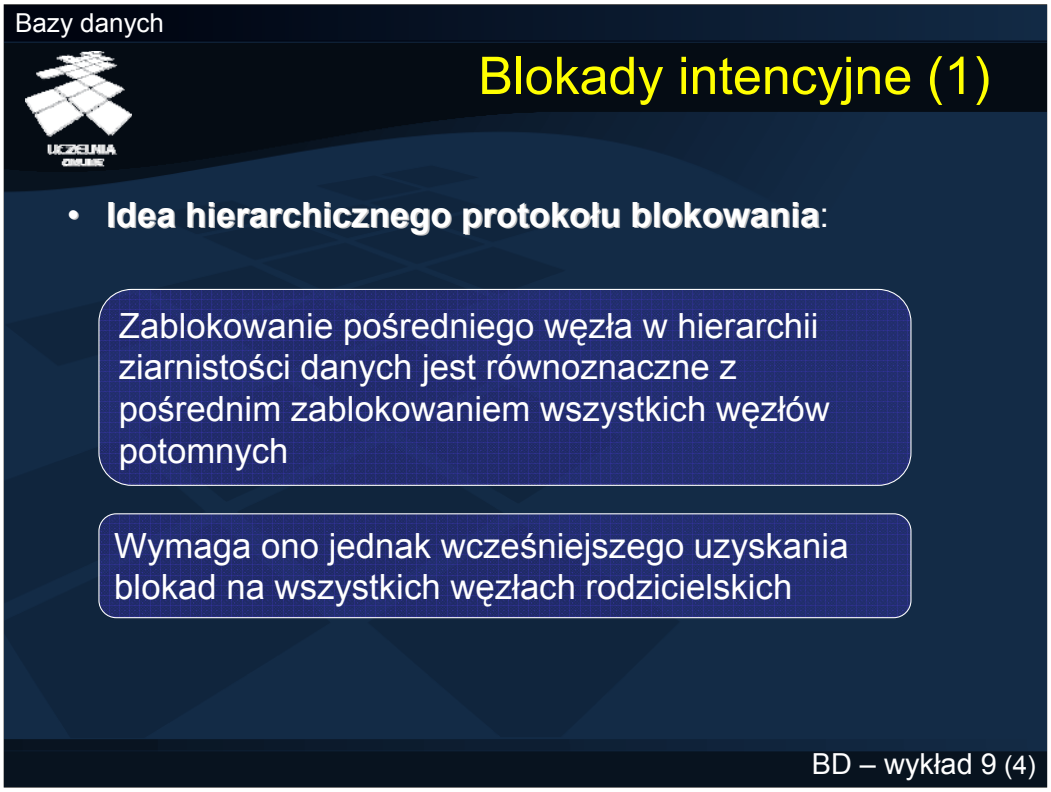
Jeszcze innym problemem związanym z wyborem jednostki blokowania jest problemem efektywności działania systemu. Podstawową miarą oceny działania systemu bazy danych jest przepustowość systemu mierzona liczbą transakcji na sekundę. Rozważmy dwa przykładowe scenariusze wykonywania transakcji na relacji Pracownicy. Załóżmy, że relacja Pracownicy zawiera 50 tys. krotek. Dana jest transakcja, która aktualizuje zarobki wszystkich pracowników o 1%. Ta transakcja odczytuje wszystkie krotki relacji Pracownicy, a następnie, aktualizuje wartość atrybutu *zarobki* dla każdej krotki. Oznacza to konieczność założenia 50 tys. blokad do odczytu, które, następnie, są konwertowane do blokad do zapisu (50tys. konwersji blokad). Po wykonaniu transakcji, system wykonuje 50 tys. operacji odblokowania. Gdyby jednostką blokowania była cała relacja, wówczas wykonanie transakcji wymagałoby założenia jednej blokady na całej relacji, jednej operacji konwersji blokady, i jednej operacji zdjęcia blokady. Z drugiej strony, rozważmy współbieżne wykonanie dwóch transakcji, z których pierwsza odczytuje krotkę r1 relacji Pracownicy, natomiast druga aktualizuje krotkę r2 tej relacji. Gdyby obie transakcje zakładały blokady na poziomie całej relacji, to, oczywiście, wystąpiłby konflikt blokad pomiędzy transakcjami, co pociągnęłoby konieczność zawieszenia wykonywania jednej transakcji do czasu zdjęcia blokady przez drugą transakcję. Z punktu widzenia tego scenariusza, lepszym rozwiązaniem byłoby, przyjęcie jako jednostki blokowania, pojedynczej krotki bazy danych. Wybór jednostki blokowania jest zatem kompromisem między stopniem współbieżności systemu, a narzutem systemowym związanym z implementacją algorytmu blokowania:

– przepustowość systemu rośnie wraz ze zwiększaniem precyzji blokowania (zmniejszaniem liczby zablokowanych danych i zwiększaniem liczby blokad),

– precyzyjne blokowanie jest kosztowne dla złożonych transakcji, które wymagają długiego czasu utrzymywania dużej liczby blokad,

– blokowanie dużych jednostek danych wspiera złożone transakcje o dużej liczbie operacji, kosztem prostych transakcji o niewielkiej liczbie operacji.

Reasumując, potrzebny jest protokół, który będzie wspierał obydwa typy transakcji, to znaczy taki, który będzie umożliwiał równoczesne zakładanie blokad na różnych jednostkach danych.



Idea hierarchicznego algorytmu blokowania dwufazowego jest następująca. Zablokowanie pośredniego węzła w hierarchii ziarnistości danych jest równoznaczne z pośrednim zablokowaniem wszystkich węzłów potomnych. Wymaga ono jednak wcześniejszego uzyskania blokad na wszystkich węzłach rodzicielskich.

Bazy danych

Blokady intencyjne (2)

- ***IR*** - transakcja zamierza uzyskiwać blokady typu ***R*** na poszczególnych obiektach składowych
- ***IW*** - transakcja zamierza uzyskiwać blokady typu ***W*** na poszczególnych obiektach składowych
- ***RIW*** - transakcja blokuje wszystkie obiekty składowe w blokadą typu ***R***, a na niektórych zamierza uzyskiwać blokady typu ***W***

Blokada żądana \ Blokada uzyskana	IR	IW	R	RIW	W
IR	✓	✓	✓	✓	-
IW	✓	✓	-	-	-
R	✓	-	✓	-	-
RIW	✓	-	-	-	-
W	-	-	-	-	-

BD – wykład 9 (5)

Hierarchiczny algorytm blokowania dwufazowego wprowadza dwa rodzaje blokad: blokady fizyczne oraz blokady intencyjne. Wyróżniamy trzy blokady intencyjne:

IR - transakcja zamierza uzyskać blokady typu ***R*** na poszczególnych obiektach składowych;

IW - transakcja zamierza uzyskać blokady typu ***W*** na poszczególnych obiektach składowych;

RIW - transakcja blokuje wszystkie obiekty składowe blokadą typu ***R***, a na niektórych elementach składowych zamierza uzyskać blokady typu ***W***. (blokada wspierająca aktualizację).

Znaczenie blokad fizycznych pozostaje bez zmian. Macierz kompatybilności rozszerzonego zbioru blokad przedstawiono na slajdzie.

Reguły blokowania intencyjnego

- Żądanie uzyskania blokady ***W*** na danym obiekcie jest pośrednim żądaniem ustawienia blokad ***W*** na wszystkich obiektach składowych
- Żądanie uzyskania blokady ***R*** lub ***RIW*** na danym obiekcie jest pośrednim żądaniem ustawienia blokad ***R*** na wszystkich obiektach składowych
- Zanim transakcja uzyska blokadę typu ***IR*** lub ***R*** na danym obiekcie, musi uzyskać blokadę typu ***IR*** na co najmniej jednym zawierającym go obiekcie wyższego poziomu
- Zanim transakcja uzyska blokadę typu ***IW***, ***RIW*** lub ***W*** na danym obiekcie, musi uzyskać blokadę typu ***IW*** na wszystkich zawierających go obiektach wyższego poziomu
- Zanim transakcja zwolni blokadę na danym obiekcie musi wpierw zwolnić wszystkie blokady z obiektów składowych

Reguły blokowania hierarchicznego algorytmu 2PL można zdefiniować następująco:

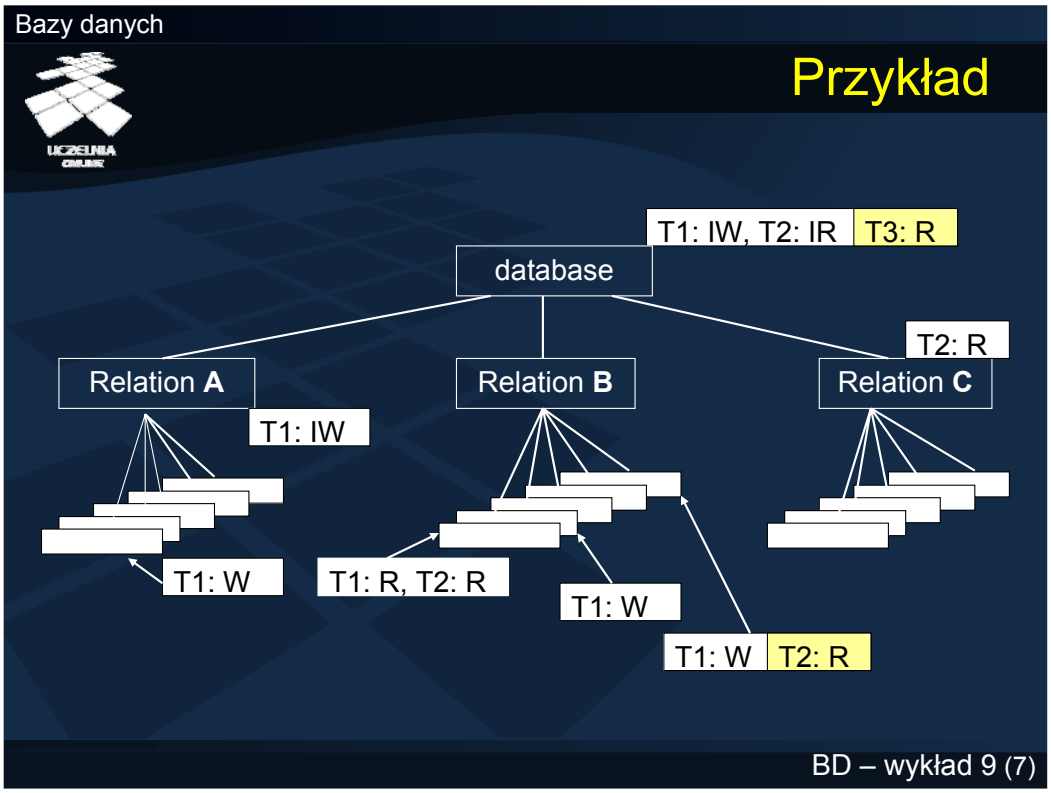
Żądanie uzyskania blokady ***W*** na danym obiekcie jest pośrednim żądaniem ustawienia blokad ***W*** na wszystkich obiektach składowych.

Żądanie uzyskania blokady ***R*** lub ***RIW*** na danym obiekcie jest pośrednim żądaniem ustawienia blokad ***R*** na wszystkich obiektach składowych.

Zanim transakcja uzyska blokadę typu ***IR*** lub ***R*** na danym obiekcie, musi uzyskać blokadę typu ***IR*** na co najmniej jednym zawierającym go obiekcie wyższego poziomu.

Zanim transakcja uzyska blokadę typu ***IW***, ***RIW*** lub ***W*** na danym obiekcie, musi uzyskać blokadę typu ***IW*** na wszystkich zawierających go obiektach wyższego poziomu.

Zanim transakcja zwolni blokadę na danym obiekcie musi wpierw zwolnić wszystkie blokady z obiektów składowych.



Dla ilustracji działania hierarchicznego algorytmu 2PL rozważmy prosty przykład przedstawiony na slajdzie. Dane są trzy transakcje T1, T2 i T3. Transakcja T1 zapisuje pierwszą krotkę relacji A, odczytuje pierwszą krotkę relacji B, i aktualizuje drugą i ostatnią krotkę relacji B. Transakcja T2 jest zapytaniem, które odczytuje pierwszą i ostatnią krotkę relacji B, oraz wszystkie krotki relacji C. Transakcja T3 odczytuje wszystkie relacje A, B i C. Zgodnie z algorytmem hierarchicznego blokowania, transakcja T1 zakłada blokadę intencyjną zapisu IW na bazie danych, oraz intencyjne blokady do zapisu IW na relacjach A i B. Następnie, zakłada fizyczna blokadę W na zapisywanej krotce relacji A, blokadę R na pierwszej krotce relacji B oraz blokadę W na drugiej i ostatniej krotce relacji B. Transakcja T2 zakłada blokadę intencyjną do odczytu IR na bazie danych. Blokady IW i IR są kompatybilne. Następnie, transakcja T2 zakłada blokadę R na pierwszej krotce relacji B oraz próbuje założyć blokadę R na ostatniej krotce relacji B. Ponieważ krotka ta jest zablokowana do zapisu przez T1, operacja transakcji T2 założenia blokady do odczytu tej krotki zostaje umieszczona w kolejce transakcji oczekujących na uwolnienie blokady tej krotki. Ponieważ transakcja T2 odczytuje wszystkie krotki relacji C, zakłada blokadę fizyczną R całej relacji C.

Transakcja T3 zamierza odczytać wszystkie relacje A, B i C, stąd zakłada blokadę fizyczną R na całej bazie danych. Blokada fizyczna R jest niekompatybilna z blokadą intencyjną do zapisu IW, stąd żądanie transakcji T3 założenia blokady R nie może być spełnione.

Tryb blokady aktualizacji (1)

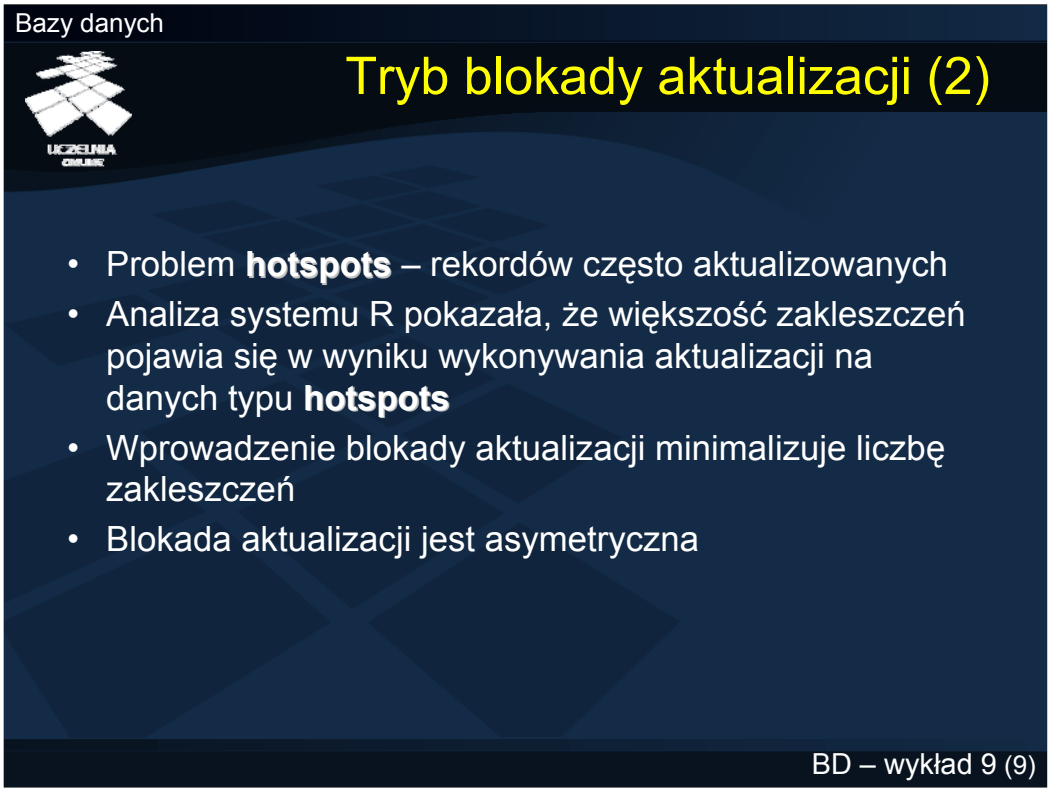
- Blokada aktualizacji (update mode) została wprowadzona w celu minimalizacji prawdopodobieństwa wystąpienia zakleszczeń

```
update emp
set salary = salary * 1.1
where name = ”Morzy”;
```

- Aktualizacja jest wykonywana następująco: blokada dla odczytu jest zakładana na wszystkich rekordach odczytywanych przez zapytanie (rekordy relacji *emp*), a następnie, na wszystkich rekordach, które spełniają predykat aktualizacji, jest zakładana blokada dla zapisu

W późniejszym czasie, zbiór typów blokad fizycznych został poszerzony o blokadę aktualizacji (update mode). Blokada aktualizacji została wprowadzona w celu minimalizacji prawdopodobieństwa wystąpienia zakleszczeń. Rozważmy następującą transakcję aktualizacji, która podnosi zarobki pracownikowi „Morzy” o 10%.

```
update emp
set salary=salary*1.1
where name=”Morzy”;
```



Realizacja tej transakcji, zgodnie z hierarchicznym algorytmem 2PL, przebiega następująco. Na relacji *Emp* jest zakładana blokada intencyjna RIW, następnie, na wszystkich rekordach odczytywanych przez transakcję jest zakładana blokada do odczytu, a następnie, na wszystkich rekordach, które spełniają predykat aktualizacji, jest zakładana blokada do zapisu. Jeżeli, współbieżnie, inna transakcja aktualizuje zarobki pracownika „Dziandziak", istnieje duże prawdopodobieństwo wystąpienia zakleszczenia. Prawdopodobieństwo wystąpienia zakleszczenia rośnie dla rekordów często aktualizowanych. Takie rekordy w bazie danych nazywamy „hotspots". Jak pokazała analiza działania systemu R, prototypowej wersji systemu DB2, większość zakleszczeń pojawia się w wyniku wykonywania aktualizacji na danych typu hotspots. W celu rozwiązania tego problemu wprowadzono blokadę aktualizacji. Cechą charakterystyczną tej blokady jest jej asymetria w stosunku do blokady R. Jeżeli transakcja T1 założyła blokadę do odczytu danej X, to inna transakcja T2 może założyć blokadę aktualizacji U tej danej. Jeżeli natomiast, transakcja T2 założyła blokadę U danej Y, to transakcja T1 nie może założyć już blokady R danej Y.

Tryb blokady aktualizacji (3)

- Pełna macierz kompatybilności blokad ma następującą postać:

Blokada żądana \ Blokada uzyskana	IR	IW	R	RIW	U	W
IR	✓	✓	✓	✓	-	-
IW	✓	✓	-	-	-	-
R	✓	-	✓	-	-	-
RIW	✓	-	-	-	-	-
U	-	-	✓	-	-	-
W	-	-	-	-	-	-

Pełną macierz kompatybilności blokad, z uwzględnieniem blokady aktualizacji, przedstawiono na slajdzie.

Algorytmy znaczników czasowych (ang. timestamp ordering)

- **Znacznik czasowy** (ang. *timestamp*) transakcji *T* - *TS*(*T*), jest jej unikalnym identyfikatorem. Znaczniki są przydzielone transakcjom w kolejności, w której transakcje pojawiają się w systemie zarządzania bazą danych
- Z każdą daną (*x*) w bazie danych związane są dwie wartości znaczników czasowych:
  - ***Read_TS(x)*** - największy znacznik czasowy spośród wszystkich transakcji, które pomyślnie odczytały tę daną
  - ***Write_TS(x)*** - największy znacznik czasowy spośród wszystkich transakcji, które pomyślnie zapisały tę daną

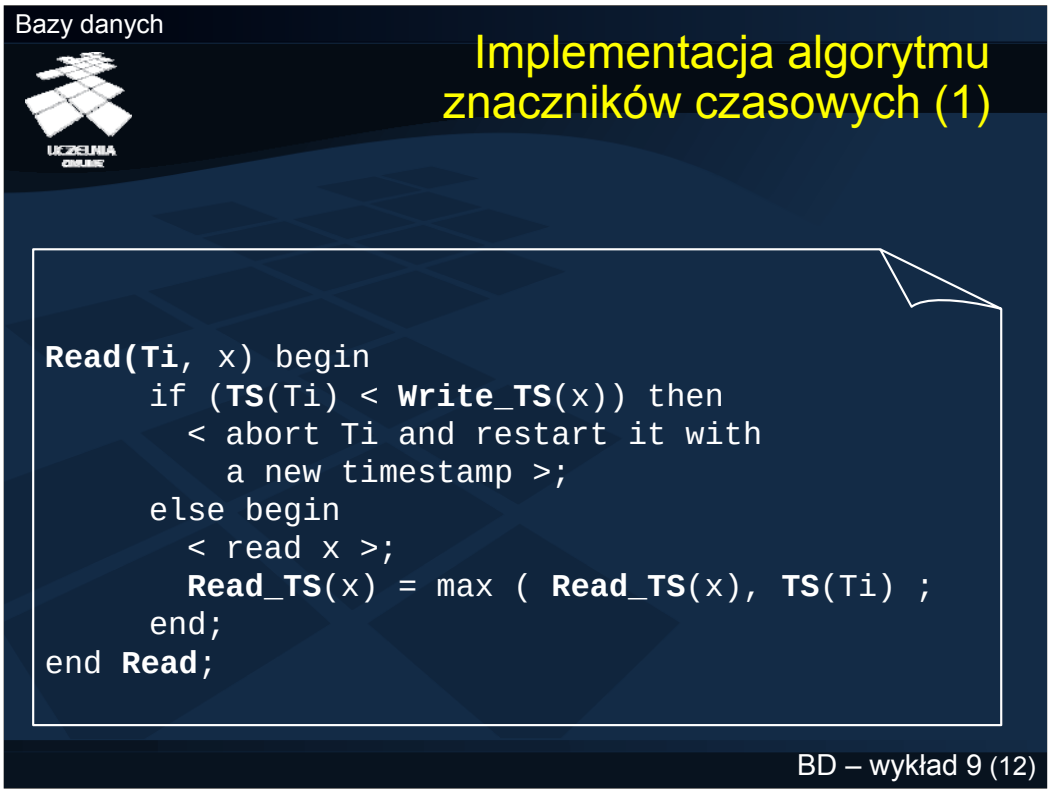
Przejdziemy obecnie do przedstawienia drugiej ważnej grupy algorytmów zarządzania współbieżnym wykonywaniem transakcji, a mianowicie, algorytmów znaczników czasowych (algorytmy T/O) (inna popularna nazwa tej grupy algorytmów to – algorytmy porządkowania transakcji wg. etykiet czasowych). W algorytmach blokowania, uszeregowanie transakcji będących w konflikcie wynika z kolejności, w jakiej te transakcje uzyskały blokady danych. Mechanizm dwufazowości zakładania blokad gwarantuje uszeregowalność realizacji transakcji. W przypadku algorytmów znaczników czasowych, uszeregowanie transakcji będących w konflikcie wynika z porządku znaczników czasowych przydzielanych transakcjom w momencie ich inicjacji. Porządek jest więc predefiniowany, a nie ustalany dynamicznie, jak w przypadku algorytmów blokowania.

Znacznik czasowy (ang. timestamp) transakcji *T* (*TS*(*T*)), jest unikalnym identyfikatorem transakcji. Znaczniki są przydzielone transakcjom w kolejności, w której transakcje pojawiają się w systemie bazy danych.

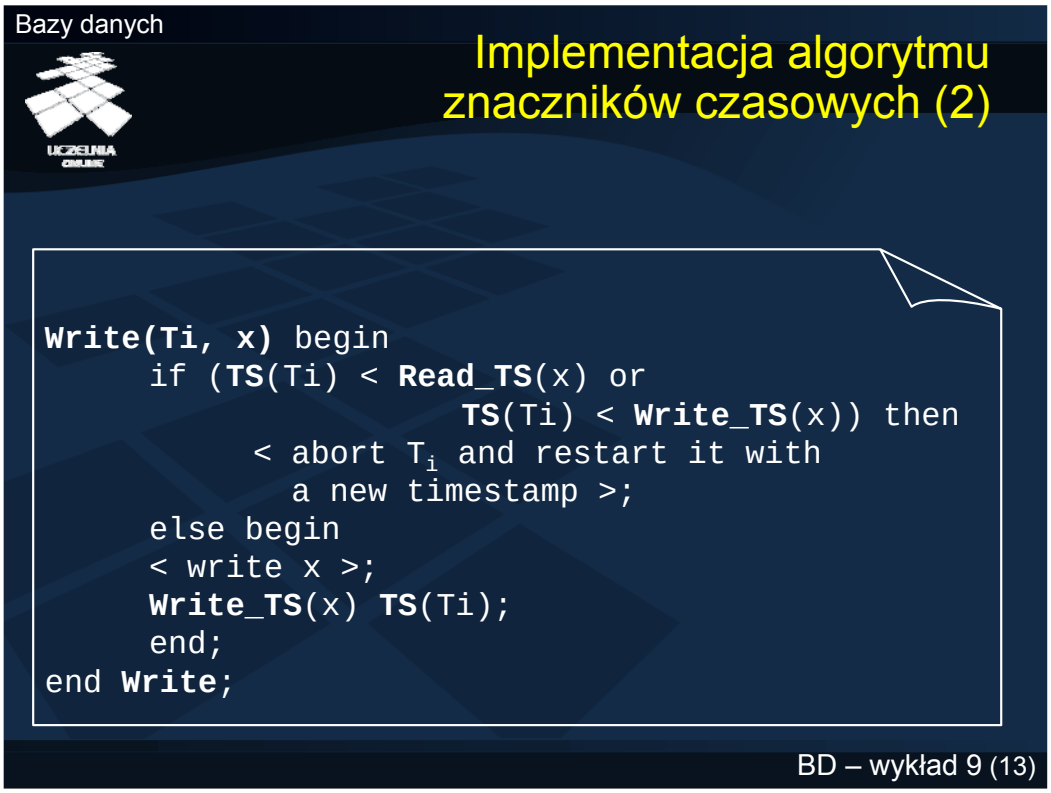
Z każdą daną (*x*) w bazie danych związane są dwie wartości znaczników czasowych:

Read_TS(x) - największy znacznik czasowy spośród wszystkich transakcji, które pomyślnie odczytały tę daną.

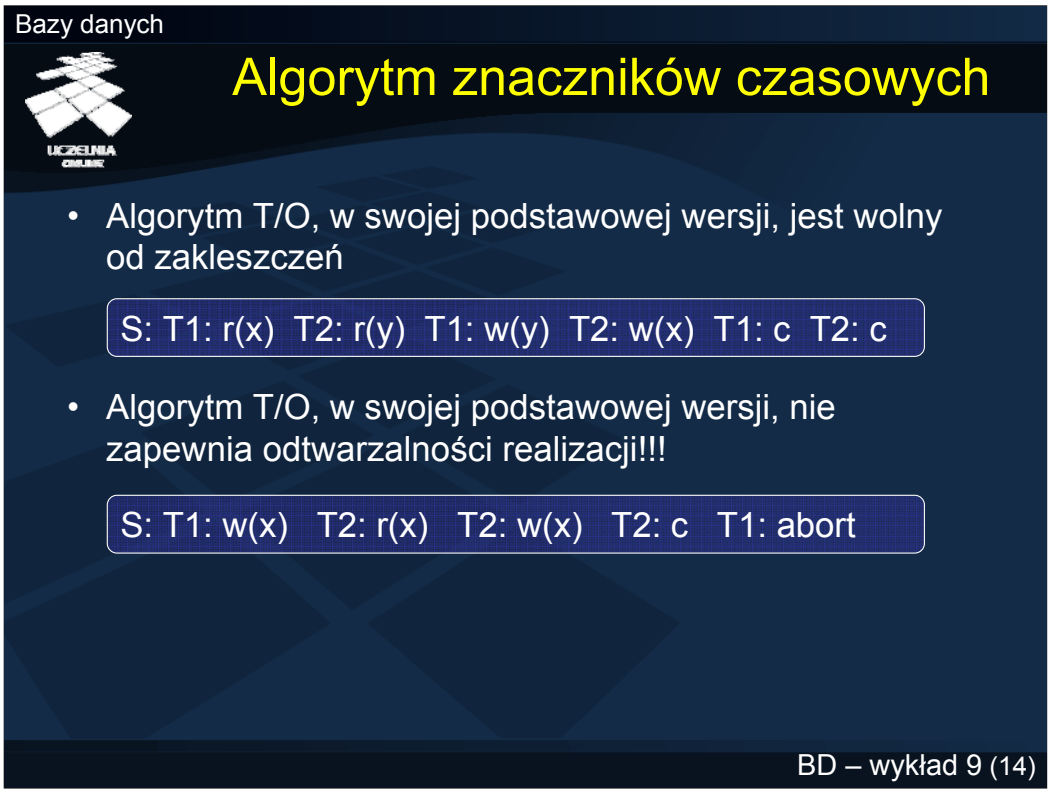
Write_TS(x) - największy znacznik czasowy spośród wszystkich transakcji, które pomyślnie zapisały tę daną.



Implementacja podstawowego algorytmu znaczników czasowych składa się z dwóch procedur: procedury odczytu danej i procedury zapisu danej. Prezentowany slajd przedstawia procedurę odczytu danej x przez transakcję Ti. Jeżeli znacznik czasowy transakcji Ti (TS(Ti)) jest mniejszy od znacznika czasowego zapisu danej x (Write_TS(x)), wówczas transakcja Ti jest wycofywana i restartowana ponowne z nowym, późniejszym, znacznikiem czasowym. W przeciwnym razie, transakcja odczytuje wartość danej x i, ewentualnie, uaktualnia znacznik czasowy odczytu danej x (Read_TS(x)).



Kolejny slajd przedstawia procedurę zapisu danej x przez transakcję Ti. Jeżeli znacznik czasowy transakcji Ti (TS(Ti)) jest mniejszy od znacznika czasowego zapisu danej x (Write_TS(x)) lub znacznika czasowego odczytu danej x (Read_TS(x)), wówczas transakcja Ti jest wycofywana i restartowana ponowne z nowym, późniejszym, znacznikiem czasowym. W przeciwnym razie, transakcja zapisuje nową wartość danej x i aktualizuje jej znacznik czasowy zapisu (Write_TS(x)).



Algorytm znaczników czasowych, w swojej podstawowej wersji, jest wolny od zakleszczeń. Rozważmy realizację transakcji T1 i T2 przedstawioną na slajdzie.

S: T1: r(x) T2: r(y) T1: w(y) T2: w(x) T1: c T2: c

W przypadku algorytmu blokowania, przedstawiona realizacja prowadzi do wystąpienia zakleszczenia. W przypadku algorytmu znaczników czasowych, zakładając, że TS(T1) < TS(T2), transakcja T1 zostanie wycofana i restartowana ponownie (na skutek odrzucenia operacji zapisu T1: w(y)). Zasadniczą wadą algorytmu jest to, że, w swojej podstawowej wersji, algorytm nie zapewnia odtwarzalności realizacji!!! Rozważmy realizację transakcji T1 i T2 przedstawioną na slajdzie:

S: T1: w(x) T2: r(x) T2: w(x) T2: c T1: abort

Jak łatwo zauważyć, przedstawiona realizacja jest nie odtwarzalna, gdyż transakcja T2 odczytała stan danej x zapisany przez transakcję T1, która została wycofana.

Algorytm znaczników czasowych z buforowaniem operacji

- Aby zapewnić odtwarzalność realizacji generowanych przez algorytm T/O konieczna jest modyfikacja algorytmu polegająca na buforowaniu operacji odczytu i zapisu danych, aż do momentu zatwierdzenia transakcji
- Transakcja T1 zapisuje daną x: aktualizowana jest wartość znacznika Write_TS(x), zmiana wartości danej x jest odsunięta w czasie do momentu zatwierdzenia transakcji T1
- Transakcja T2 odczytuje daną x aktualizowaną przez transakcję T1: znacznik transakcji TS(T2) jest porównywany ze znacznikiem Write_TS(x), jeżeli warunek odczytu jest spełniony, to odczyt jest odsunięty w czasie do momentu akceptacji transakcji T1, w przeciwnym razie transakcja T2 jest wycofywana

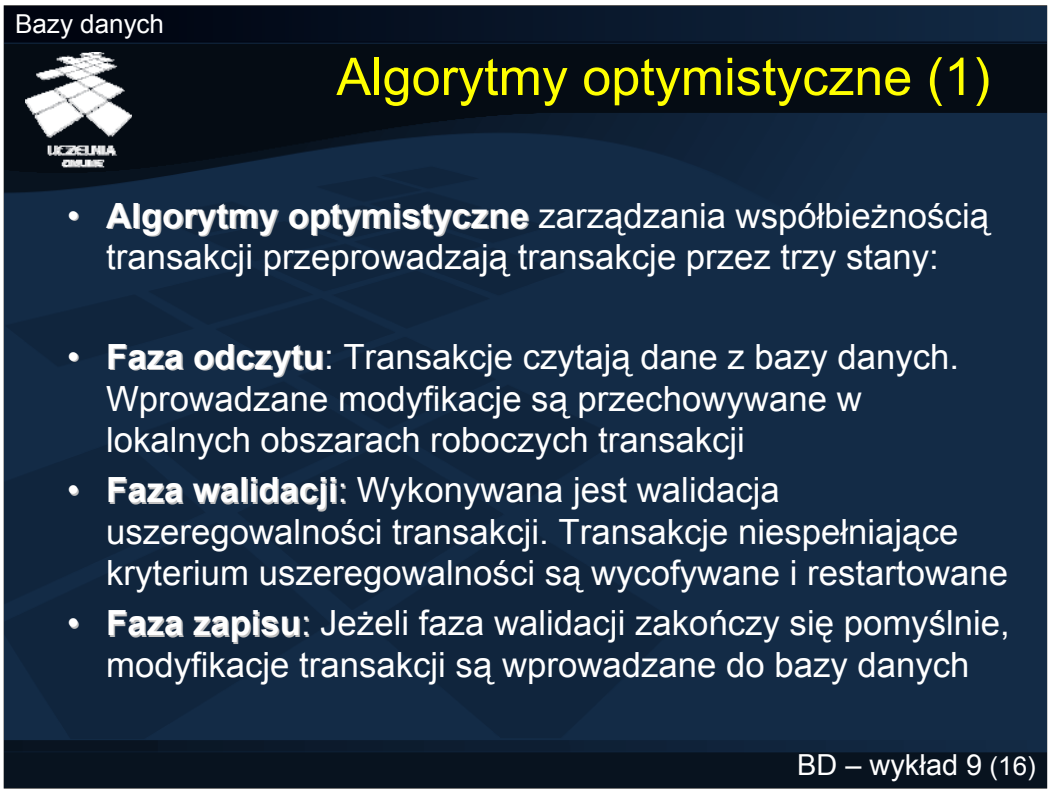
Efekt buforowania jest podobny do efektu zakładania blokad dla zapisu na danych !!!

Aby zapewnić odtwarzalność realizacji generowanych przez algorytm znaczników czasowych konieczna jest modyfikacja algorytmu polegająca na buforowaniu operacji odczytu i zapisu danych, aż do momentu zatwierdzenia transakcji.

Transakcja T1 zapisuje daną x: aktualizowana jest wartość znacznika Write_TS(x), zmiana wartości danej x jest odsunięta w czasie do momentu zatwierdzenia transakcji T1.

Transakcja T2 odczytuje daną x aktualizowaną przez transakcję T1: znacznik transakcji TS(T2) jest porównywany ze znacznikiem Write_TS(x), jeżeli warunek odczytu jest spełniony, to odczyt jest odsunięty w czasie do momentu akceptacji transakcji T1, w przeciwnym razie transakcja T2 jest wycofywana.

Efekt buforowania jest podobny do efektu zakładania blokad dla zapisu na danych !!!



Na zakończenie prezentacji algorytmów zarządzania współbieżnym wykonywaniem transakcji, przedstawimy, krótko, ostatnią z wymienionych wcześniej metod – metodę optymistyczną.

Algorytmy blokowania nazywane są często algorytmami pesymistycznymi zarządzania współbieżnym wykonywaniem transakcji, gdyż zakładają one pesymistyczny scenariusz konfliktów pomiędzy transakcjami – wystąpienie konfliktu jest traktowane jako potencjalne źródło nieuszeregowalności realizacji. Stąd, konflikt pomiędzy transakcjami jest rozwiązywany poprzez blokady lub wycofanie transakcji. Jednakże, w systemach, w których obciążenia danych są niskie, prawdopodobieństwo, że wystąpi konflikt pomiędzy współbieżnie wykonywanymi transakcjami jest niewielkie, a jeszcze mniejsze jest prawdopodobieństwo, że konflikt ten spowoduje nieuszeregowalność realizacji. Z drugiej strony narzut systemowy związany z mechanizmem zakładania i zdejmowania blokad jest wysoki. Podstawowym założeniem metod optymistycznych jest założenie, że konflikty pomiędzy transakcjami występują stosunkowo rzadko (stąd nazwa metody), stąd algorytm powinien być maksymalnie mało restryktywny jeżeli chodzi o ograniczenie dostępu do danych dla współbieżnie wykonywanych transakcji.

Wykonywanie transakcji w metodzie optymistycznej przebiega w trzech fazach:

Fazie odczytu: Transakcje czytają dane z bazy danych. Wprowadzane modyfikacje są przechowywane w lokalnych obszarach roboczych transakcji.

Fazie walidacji: Wykonywana jest walidacja uszeregowalności transakcji. Transakcje nie spełniające kryterium uszeregowalności są wycofywane i restartowane.

Fazie zapisu: Jeżeli faza walidacji zakończy się pomyślnie, modyfikacje transakcji są wprowadzane do bazy danych.

Algorytmy optymistyczne (2)

- Z każdą transakcją T_i związane są czasy rozpoczęcia poszczególnych faz transakcji oraz utrzymywane są zbiory danych odczytywanych R_i i modyfikowanych W_i
- W fazie walidacji transakcji T_i, dla każdej transakcji T_j, która została zatwierdzona lub znajduje się w fazie walidacji sprawdzane jest czy spełniony jest jeden z warunków:

1. Transakcja T_j zakończyła swoją fazę zapisu zanim transakcja T_i rozpoczęła swoją fazę odczytu
2. $(R_i \cup W_i) \cap W_j = \varnothing$ i T_i zakończyła swoją fazę odczytu zanim T_i zakończyła swoją fazę odczytu

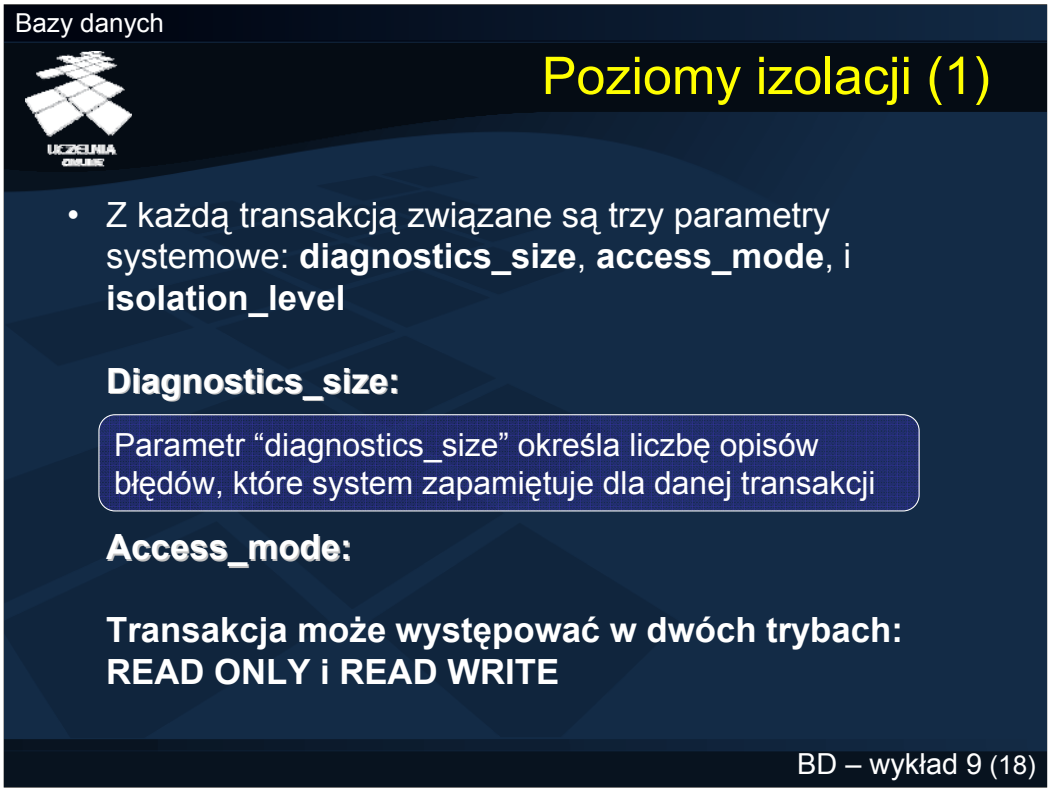
Z każdą transakcją *Ti* związane są czasy rozpoczęcia poszczególnych faz transakcji oraz utrzymywane są zbiory danych odczytywanych *Ri* i modyfikowanych *Wi*.

W fazie walidacji transakcji *Ti*, dla każdej transakcji *Tj*, która została zatwierdzona lub znajduje się w fazie walidacji sprawdzane jest czy spełniony jest jeden z warunków:

1. Warunek pierwszy: transakcja *Tj* zakończyła swoją fazę zapisu zanim transakcja *Ti* rozpoczęła swoją fazę odczytu.

2. Warunek drugi: wyrażenie nr 2 ze slajdu daje w wyniku zbiór pusty ((*Ri* SUMA *Wi*) ILOCZYN *Wj* = ZBIÓR PUSTY) i *Ti* zakończyła swoją fazę odczytu zanim *Tj* zakończyła swoją fazę odczytu.

Jeżeli spełniony jest którykolwiek z powyższych warunków, transakcja Ti jest zatwierdzana i przechodzi do fazy zapisu. W przeciwnym przypadku, transakcja Ti jest wycofywana i restartowana ponownie. Łatwo zauważyć, że test walidacji jest stosunkowo prosty i może być efektywnie zaimplementowany. Algorytm optymistyczny został zastosowany jako mechanizm zarządzania współbieżnością, dla specyficznych danych, w systemie DB2.



Paradoksalnie, komercyjne systemy zarządzania bazami danych jak i standard SQL, nie zapewniają, automatycznie, uszeregowalności realizacji transakcji. Wprowadzają one koncepcję tak zwanych poziomów izolacji. Zanim przedstawimy koncepcję poziomów izolacji transakcji przedstawioną w standardzie SQL, krótko omówimy inne elementy specyfikacji transakcji w tym standardzie.

Z każdą transakcją, zgodnie ze standardem SQL, związane są trzy parametry systemowe: diagnostics_size, access_mode, i isolation_level.

Parametr “diagnostics_size” określa liczbę opisów błędów, które system zapamiętuje dla danej transakcji. Parametr „access_mode” określa tryb wykonywania transakcji. Transakcja może występować w dwóch trybach: READ ONLY i READ WRITE.

Poziomy izolacji (2)

- Jeżeli transakcja występuje w trybie READ ONLY oznacza to, że transakcja jest zapytaniem i nie może modyfikować bazy danych. Dla transakcji w trybie READ ONLY, system zakłada wyłącznie blokady dla odczytu - prowadzi to do zwiększenia stopnia współbieżności.

- Domyślnie, trybem wykonywania transakcji jest tryb READ WRITE, który pozwala na wykonywanie wszystkich typów operacji na bazie danych

Jeżeli transakcja występuje w trybie READ ONLY oznacza to, że transakcja jest zapytaniem i nie może modyfikować bazy danych. Dla transakcji w trybie READ ONLY, system zakłada wyłącznie blokady dla odczytu - prowadzi to do zwiększenia stopnia współbieżności.

Domyślnie, trybem wykonywania transakcji jest tryb READ WRITE, który pozwala na wykonywanie wszystkich typów operacji na bazie danych.

Poziomy izolacji (3)

Większość systemów zarządzania bazami danych nie zapewnia automatycznie uszeregowalności realizacji!!

Celem prowadzenia tak zwanych poziomów izolacji jest znalezienie kompromisu pomiędzy poprawnością wykonywania transakcji a zapewnieniem jak największego stopnia współbieżności wykonywanych transakcji.

Standard SQL-92 definiuje cztery poziomy izolacji:
- READ UNCOMMITTED
- READ COMMITTED
- REPEATABLE READ
- SERIALIZABLE

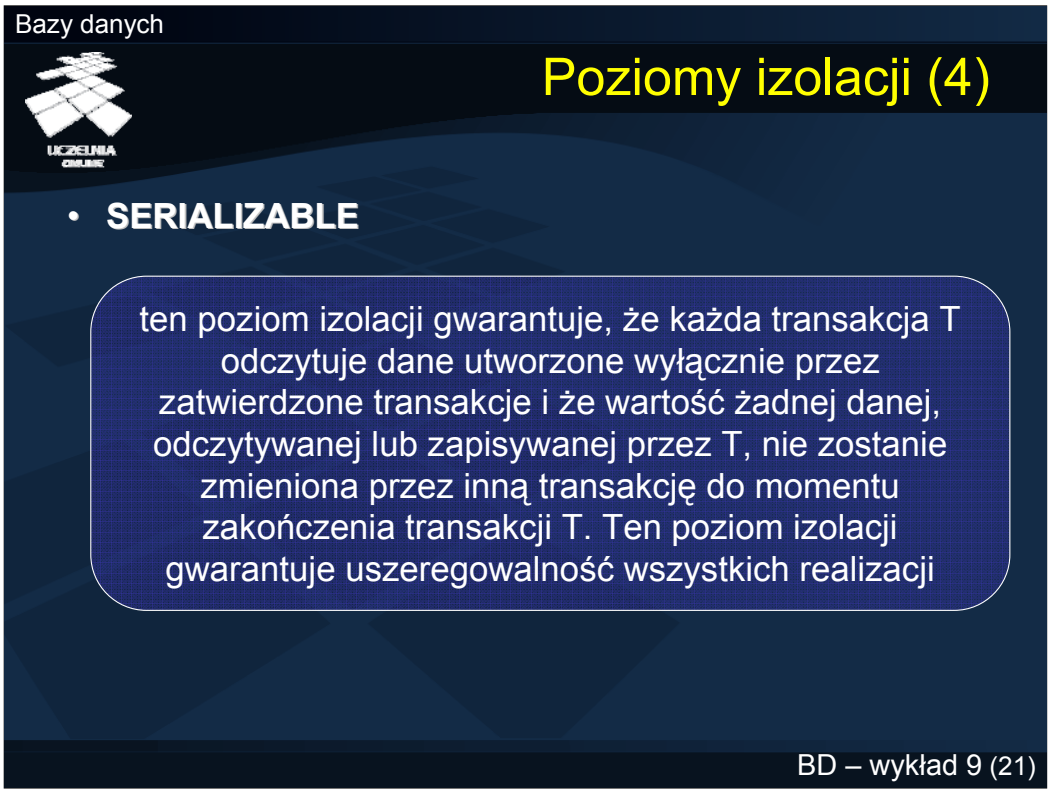
Jak już wspomnieliśmy, systemy zarządzania bazami danych nie zapewniają automatycznie uszeregowalności realizacji!! Wprowadzają one koncepcję tak zwanych poziomów izolacji transakcji.

Celem wprowadzenia poziomów izolacji transakcji jest znalezienie kompromisu pomiędzy poprawnością wykonywania transakcji a zapewnieniem jak największego stopnia współbieżności wykonywanych transakcji.

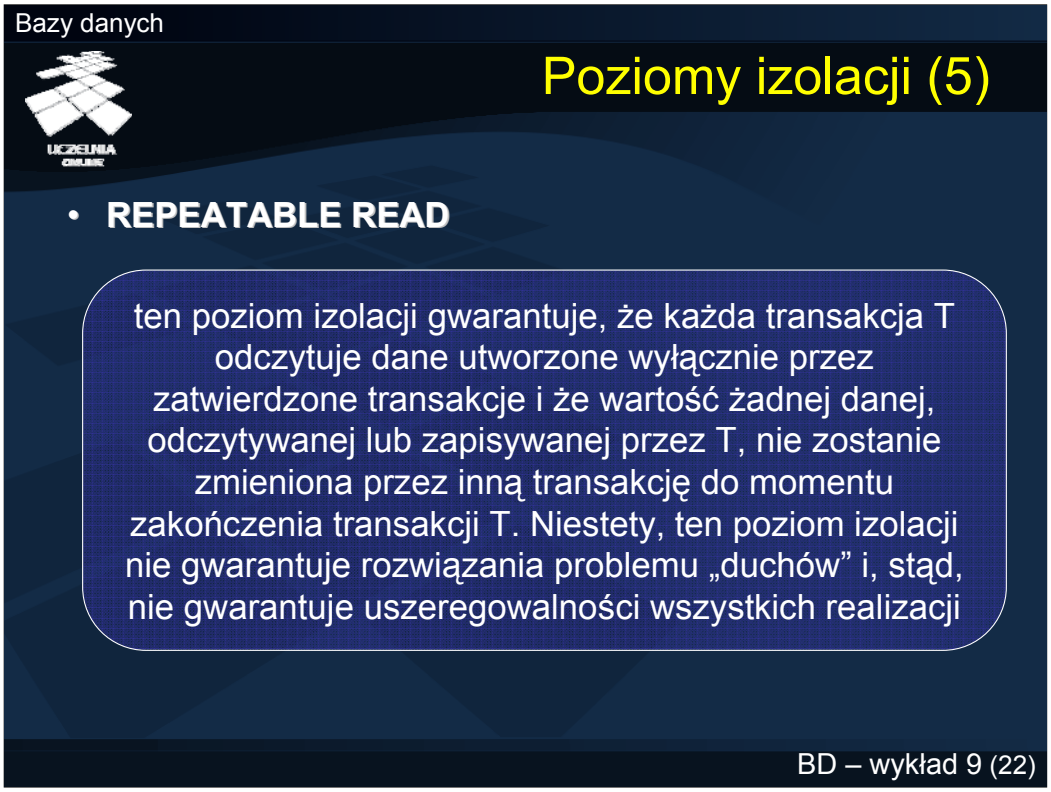
Standard SQL-92 definiuje cztery poziomy izolacji:

- READ UNCOMMITTED (poziom izolacji 0)
- READ COMMITTED (poziom izolacji 1)
- REPEATABLE READ (poziom izolacji 2)
- SERIALIZABLE (poziom izolacji 3)

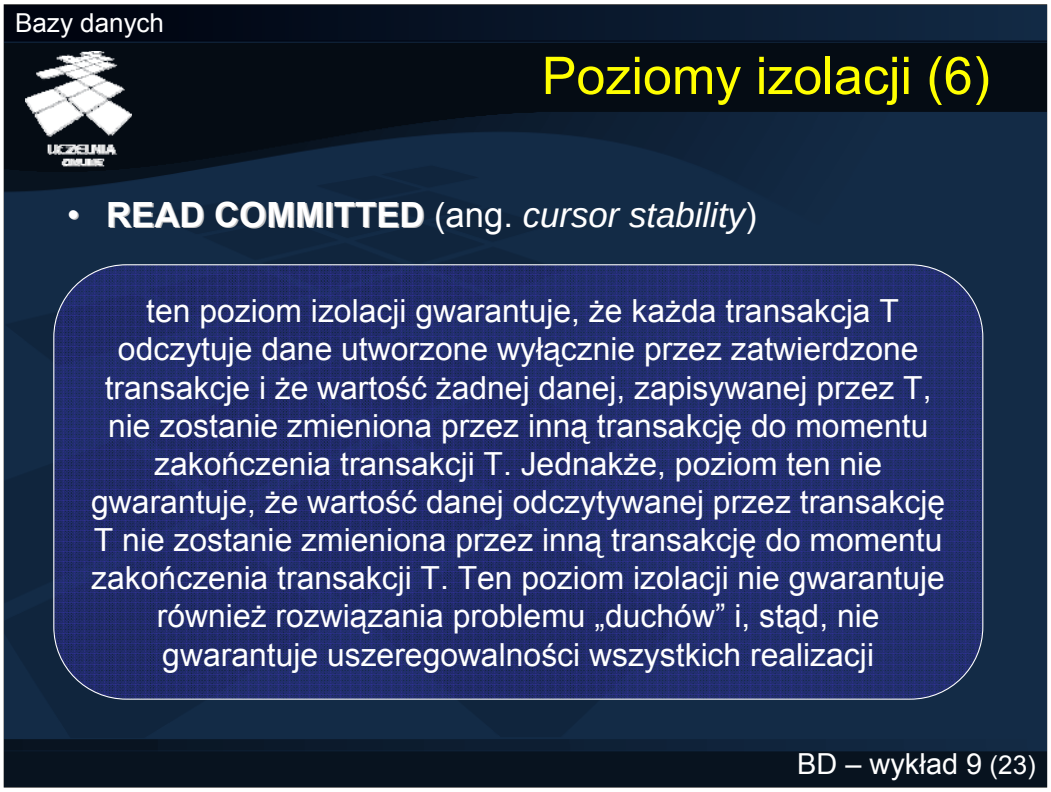
Jedynie poziom izolacji 3 zapewnia pełną uszeregowalność realizacji transakcji. W kolejnych slajdach, krótko omówimy poszczególne poziomy izolacji.



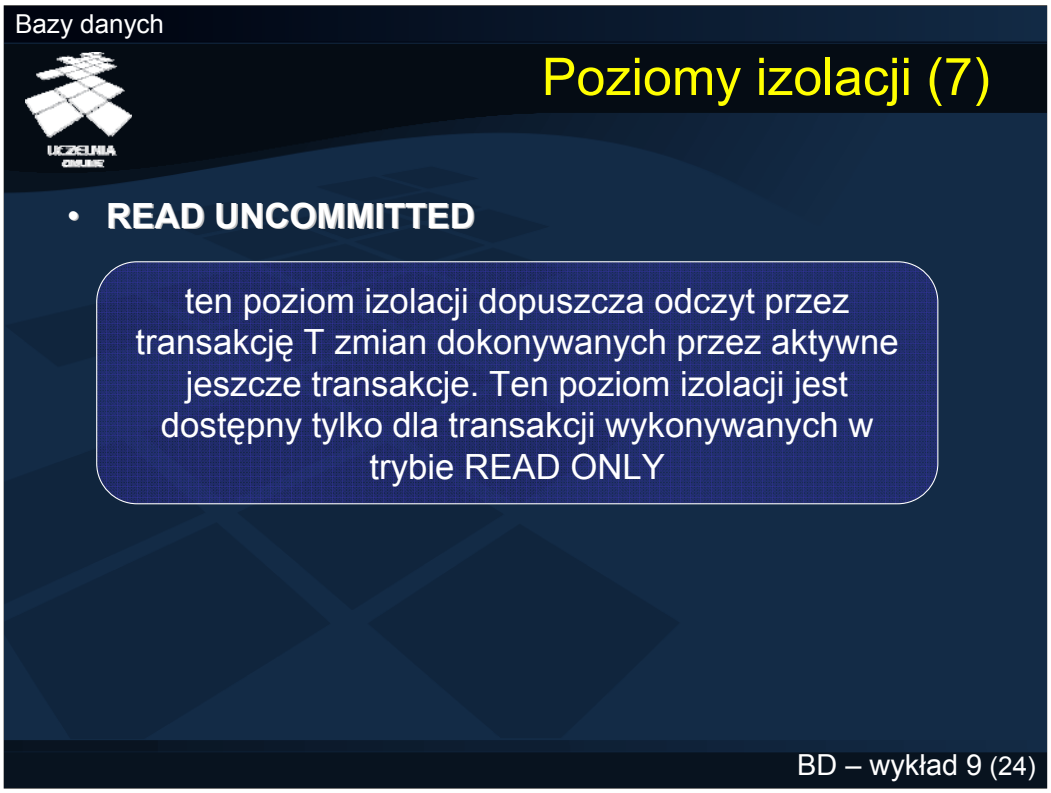
Poziom izolacji 3 - SERIALIZABLE. Ten poziom izolacji gwarantuje, że każda transakcja T odczytuje dane utworzone wyłącznie przez zatwierdzone transakcje i że wartość żadnej danej, odczytywanej lub zapisywanej przez T, nie zostanie zmieniona przez inną transakcję do momentu zakończenia transakcji T. Ten poziom izolacji gwarantuje uszeregowalność wszystkich realizacji transakcji T.



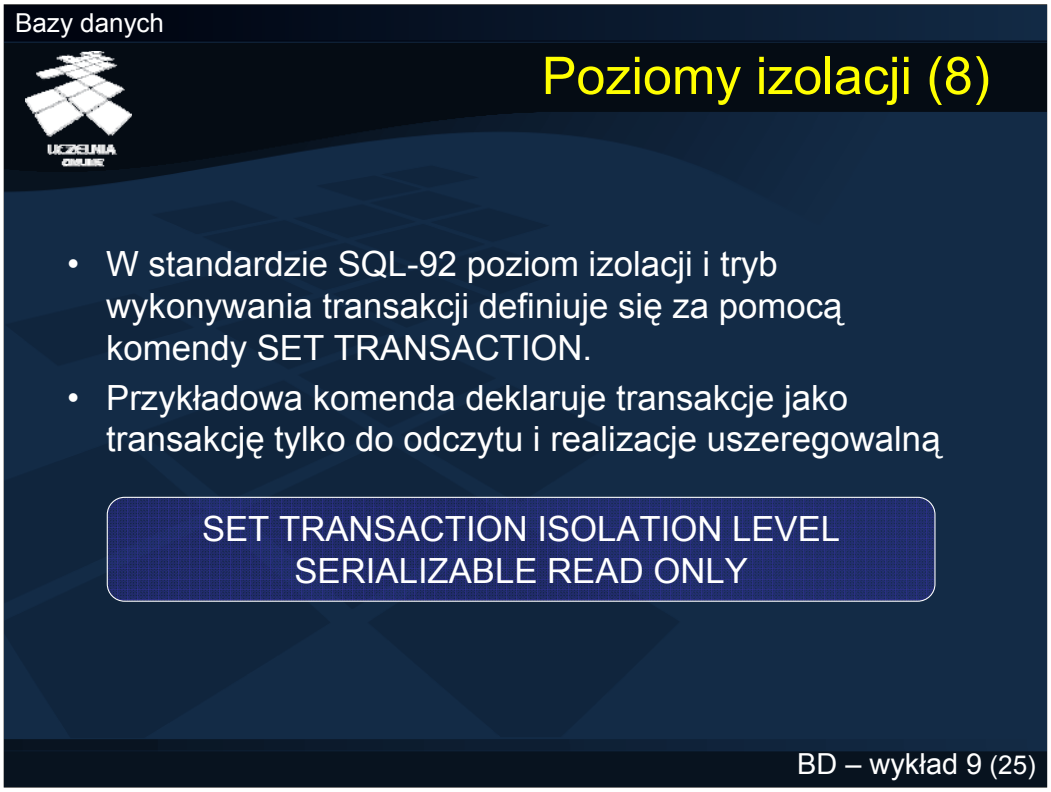
Poziom izolacji 2 - REPEATABLE READ. Ten poziom izolacji gwarantuje, że każda transakcja T odczytuje dane utworzone wyłącznie przez zatwierdzone transakcje i że wartość żadnej danej, odczytywanej lub zapisywanej przez T, nie zostanie zmieniona przez inną transakcję do momentu zakończenia transakcji T. Niestety, ten poziom izolacji nie gwarantuje rozwiązania problemu rekordów „duchów” i, stąd, nie gwarantuje uszeregowalności wszystkich realizacji.



Poziom izolacji 1 - READ COMMITTED. Ten poziom izolacji gwarantuje, że każda transakcja T odczytuje dane utworzone wyłącznie przez zatwierdzone transakcje i że wartość żadnej danej, zapisywanej przez T, nie zostanie zmieniona przez inną transakcję do momentu zakończenia transakcji T. Jednakże, poziom ten nie gwarantuje, że wartość danej odczytywanej przez transakcję T nie zostanie zmieniona przez inną transakcję do momentu zakończenia transakcji T. Ten poziom izolacji nie gwarantuje również rozwiązania problemu „duchów” i, stąd, nie gwarantuje uszeregowalności wszystkich realizacji



Poziom izolacji 0 - READ UNCOMMITTED. Ten poziom izolacji dopuszcza odczyt przez transakcję T zmian dokonywanych przez aktywne jeszcze transakcje. Ten poziom izolacji jest dostępny tylko dla transakcji wykonywanych w trybie READ ONLY



W standardzie SQL-92 poziom izolacji i tryb wykonywania transakcji definiuje się za pomocą polecenia systemowego SET TRANSACTION.

Przykładowe polecenie, przedstawione na slajdzie, deklaruje transakcję jako transakcję tylko do odczytu i realizację uszeregowalną.