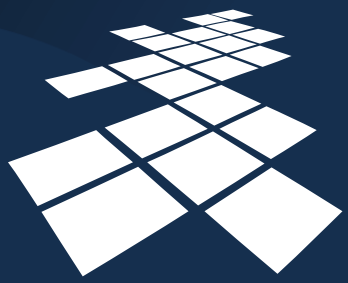
Systemy baz danych

Przetwarzanie transakcyjne

Wykład przygotował:
Tadeusz Morzy

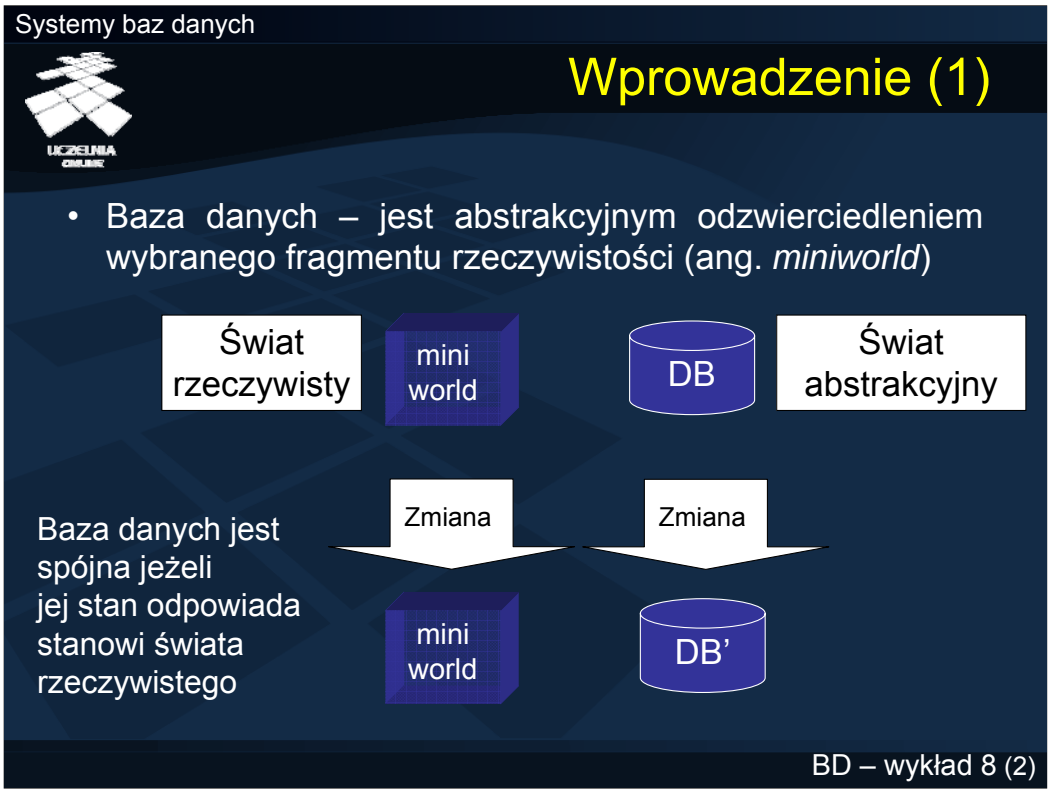


UCZELNIA
ONLINE

BD – wykład 8

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.



Jak wspomnieliśmy w jednym z pierwszych wykładów, baza danych jest abstrakcyjnym odzwierciedleniem wybranego fragmentu rzeczywistości. Fragment ten powinien być wiernie odzwierciedlany w bazie danych. Mówimy, że baza danych jest spójna jeżeli jej stan odpowiada stanowi świata rzeczywistego.

Wprowadzenie (2)

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

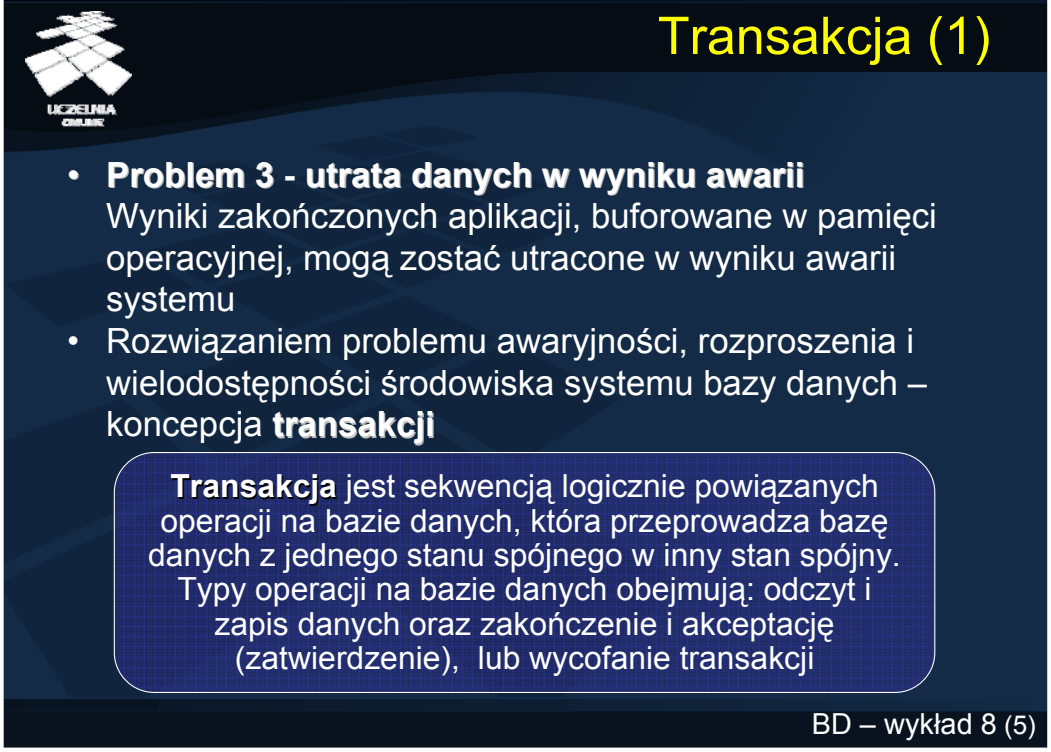
W celu zapewnienia tej spójności, zmiany zachodzące w świecie rzeczywistym muszą być zakodowane w postaci programu, który będzie transformował bazę danych z jednego stanu spójnego do innego stanu spójnego. Wykonanie tego programu powinno być odporne na wszelkiego rodzaju awarie sprzętowo-programowe. Ponadto, baza danych jest przeznaczona do użytkowania przez wielu równocześnie pracujących użytkowników. Taka równoległa praca może również wpływać na poprawność danych w bazie danych. W przypadku rozproszenia bazy danych na wiele węzłów sieci, należy zapewnić poprawność danych we wszystkich węzłach.

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 – współbieżny dostęp do danych**
  Operacje współbieżnie wykonywanych transakcji mogą naruszać spójność bazy danych, lub generować niepoprawne wyniki

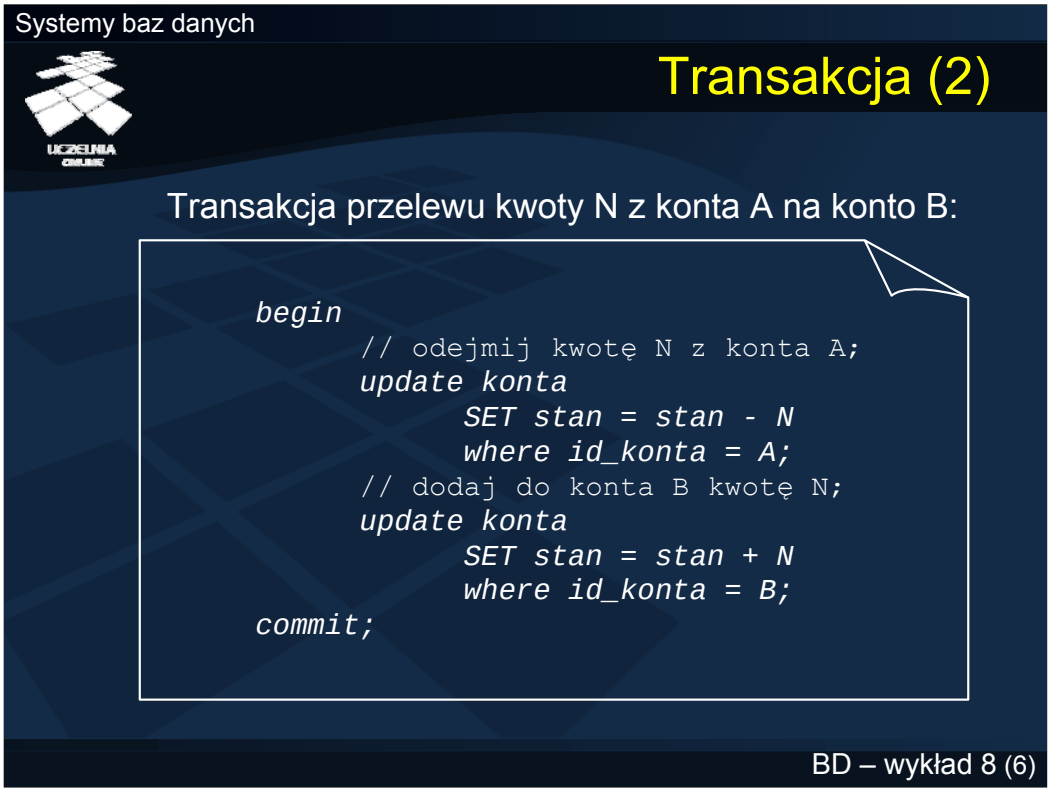
Jako przykład rozważmy system bankowy i aplikację przelewającą kwotę N z konta A na konto B. Załóżmy, że w czasie realizowania tej operacji, po pobraniu kwoty N z konta A, i zapisaniu tej aktualizacji do bazy danych, wystąpiła awaria systemu. W wyniku tej awarii wykonana została jedynie pierwsza część operacji przelewu, tj. kwota N została zdjęta z konta A, ale nie zdążyła ona wpłynąć na konto B.

Jeżeli w systemie bankowym będzie równocześnie działać wiele aplikacji przelewu (co jest typowe w rzeczywistości), wówczas ich równoczesna praca może powodować powstawanie danych niespójnych, czyli nieprawdziwych - mogą się pojawiać stany kont w rzeczywistości niewystępujące.



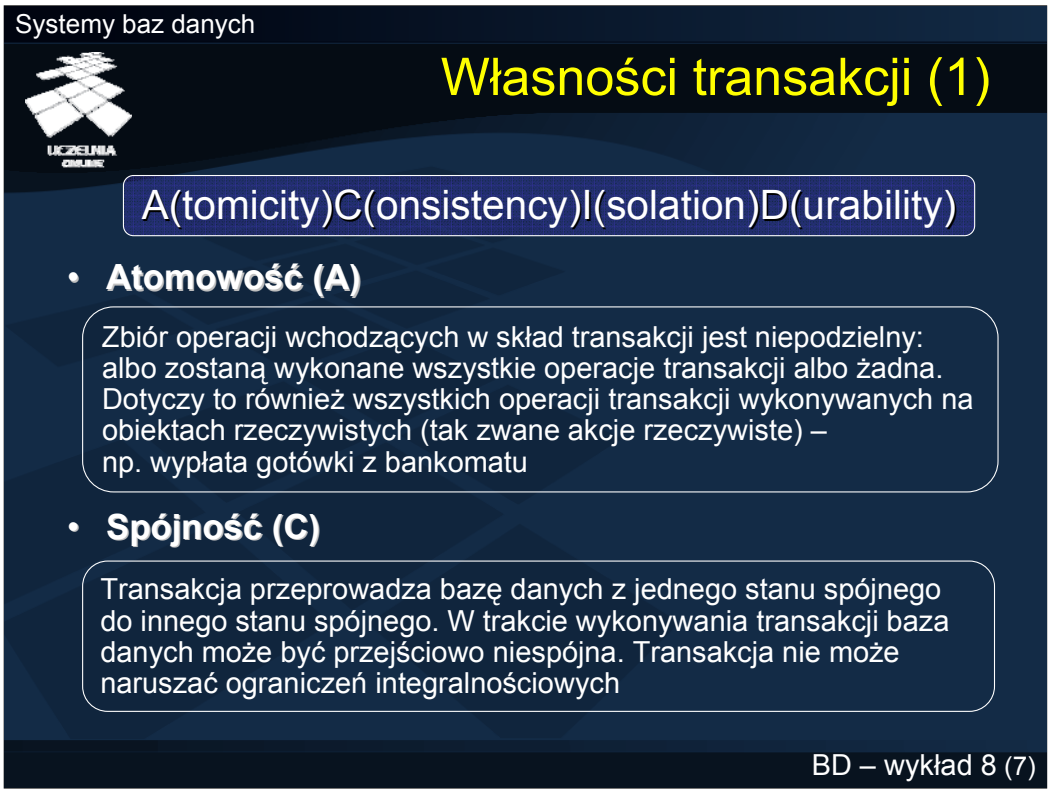
Kolejnym problemem jest niebezpieczeństwo utraty danych w wyniku awarii systemu. Jeżeli dane zmodyfikowane i wprowadzone przez zakończone aplikacje są buforowane w pamięci operacyjnej, to oznacza, że są one ulotne. Jakakolwiek awaria systemu spowoduje utratę tych danych.

Rozwiązaniem omówionych problemów jest wprowadzenie mechanizmu tzw. transakcji. 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.



Jako przykład, rozważmy transakcję przelewu kwoty N z konta A na konto B. Transakcja ta składa się z następujących operacji:

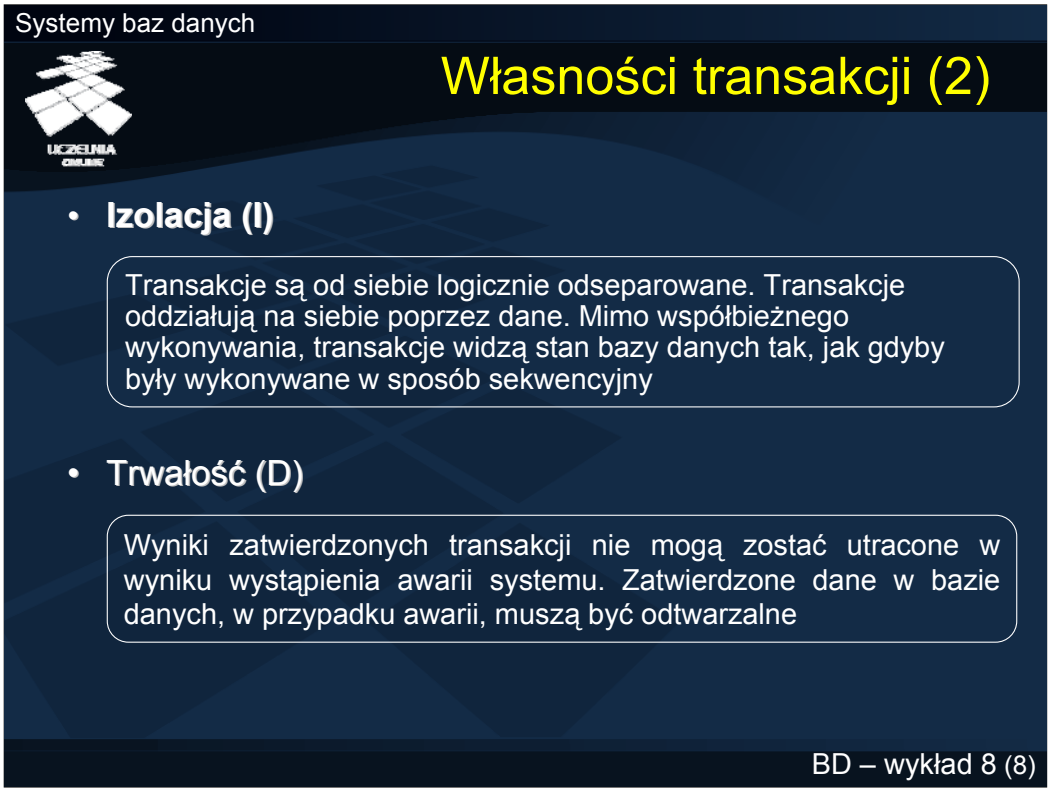
1. rozpoczęcie transakcji - begin,
2. pomniejszenie stanu konta A o kwotę N,
3. powiększenie stanu konta B o kwotę N,
4. zatwierdzenie transakcji - commit.



Każda transakcja posiada cztery cechy, tj. atomowiść (ang. **A**tomicity), spójność (ang. **C**onistency), izolacja (ang. **I**solation) i trwałość (ang. **D**urability). Cechy te są najczęściej oznaczane jako ACID, od angielskich nazw.

Atomowość oznacza, że zbiór operacji wchodzących w skład transakcji jest niepodzielny, to znaczy albo zostaną wykonane wszystkie operacje transakcji albo żadna. Dotyczy to również wszystkich operacji transakcji wykonywanych na obiektach rzeczywistych (tak zwane akcje rzeczywiste) – np. wypłata gotówki z bankomatu.

Spójność oznacza, że transakcja przeprowadza bazę danych z jednego stanu spójnego do innego stanu spójnego. W trakcie wykonywania transakcji baza danych może być przejściowo niespójna. Transakcja nie może naruszać ograniczeń integralnościowych.



Izolacja oznacza, że transakcje są od siebie logicznie odseparowane. Transakcje oddziałują na siebie poprzez dane. Mimo współbieżnego wykonywania, transakcje widzą stan bazy danych tak, jak gdyby były wykonywane w sposób sekwencyjny.

Trwałość oznacza, że wyniki zatwierdzonych transakcji nie mogą zostać utracone w wyniku wystąpienia awarii systemu. Zatwierdzone dane w bazie danych, w przypadku awarii, muszą być odtwarzalne.

Transakcja (3)

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

Transakcja przelewu z naszego przykładu jest:

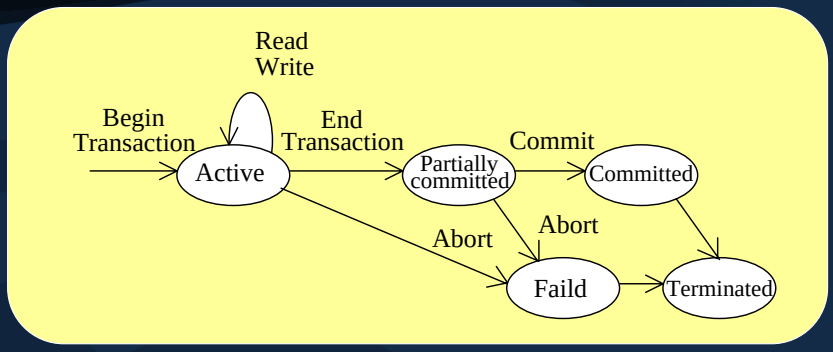
- atomowa jeżeli pieniądze zostaną poprawnie przetransferowane z konta A do B;

- spójna jeżeli kwota odjęta z konta A jest równa kwocie dodanej do konta B;

- izolowana jeżeli inne transakcje wykonywane współbieżnie, czytające i modyfikujące konta A i B, nie mają wpływu na tę transakcję;

- 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

Każda realizowana transakcja posiada zbiór ściśle określonych stanów i zbiór ściśle określonych przejść z jednego stanu do drugiego. Stany te są następujące:

- Active: transakcja jest aktywna, jest w czasie realizowania swoich operacji;
- Partially committed: transakcja jest częściowo zatwierdzona;
- Committed: transakcja została zatwierdzona;
- Failed: transakcja została wycofana;
- Terminated: transakcja zakończyła się zatwierdzeniem lub wycofaniem.

Przejścia z jednego stanu do drugiego są opisane tzw. diagramem stanów transakcji przedstawionym na slajdzie. Rozpoczęcie transakcji (Begin Transaction) uruchamia transakcję, która jest aktywna. Każda operacja zapisu lub odczytu danych w ramach tej transakcji dokonuje się w stanie aktywnym transakcji. Kończenie transakcji z jej wycofaniem (Abort) przeprowadza transakcję ze stanu Active do stanu Failed, a następnie Terminate. Kończenie transakcji z jej zatwierdzeniem przeprowadza ją ze stanu Active do Partially committed - transakcja jest gotowa do zatwierdzenia. Z tego stanu można jeszcze transakcję wycofać, np. w sytuacji awarii systemu. Ostateczne zatwierdzenie transakcji przeprowadza ją do stanu Committed, a następnie do Terminated, co kończy działanie transakcji.

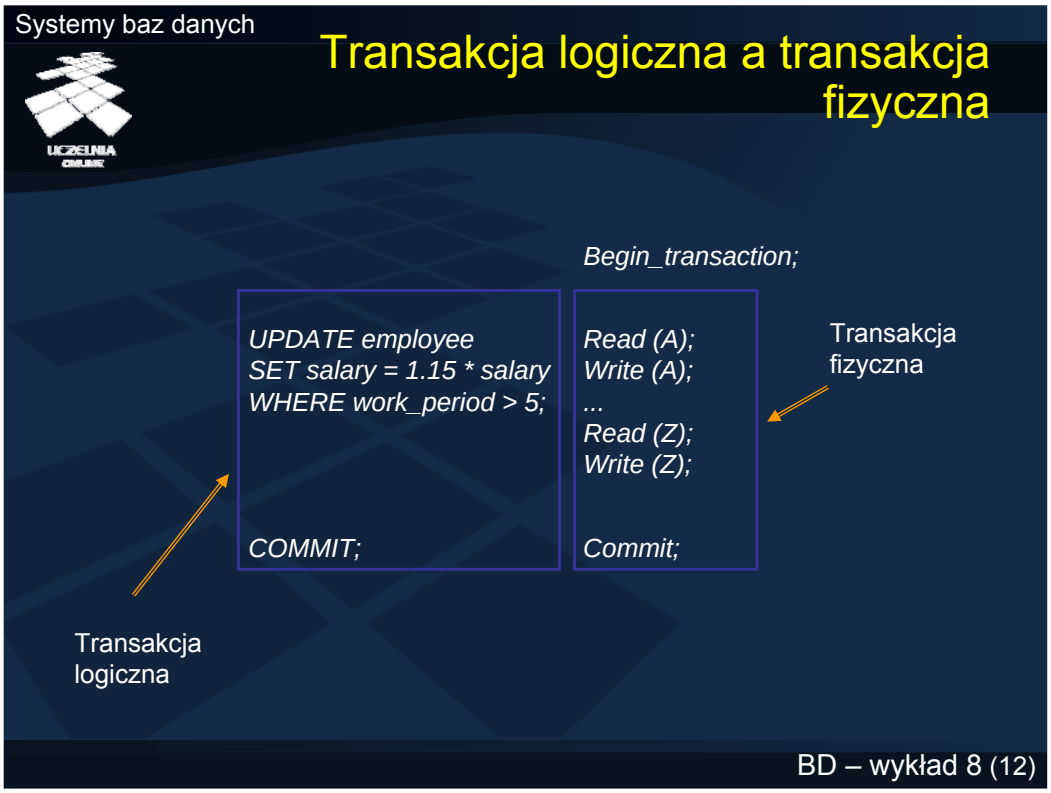
Zakończenie transakcji

- **End_transaction:** koniec transakcji oznacza, ze 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

End_transaction oznacza, ze 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 oznacza zatwierdzenie (akceptacja transakcji), czyli pomyślne zakończenie transakcji - zmiany wprowadzone przez transakcję mają być wprowadzone do bazy danych.

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



Z punktu widzenia użytkownika, transakcja jest zbiorem poleceń języka SQL, tj. select, insert, update, delete, commit, rollback. Mówimy tu o tzw. transakcji logicznej. Na poziomie systemu zarządzania bazą danych mówimy o tzw. transakcji fizycznej, która jest zarządzana przez odpowiedni moduł SZBD. Transakcja fizyczna składa się z elementarnych operacji rozpoczęcia transakcji, operacji zaalokowania zasobów systemowych dla transakcji, blokowania danych (przy pewnych rozwiązaniach synchronizacji transakcji), operacji na samych danych, kończenia transakcji i zwalniania zasobów systemowych.

Model transakcji (1)

- **Transakcją** T_i nazywamy uporządkowaną parę:

  gdzie: $T_j = (\overline{T_i} < T_j)$

  $\overline{T_i}$ = { o_j : 1 ≤ j ≤ n_i}, oznacza zbiór operacji na bazie danych: { **R** - odczyt, **W** - zapis, **C** – zatwierdzenie transakcji, **A** - wycofanie}

  $<T_j$ jest relacją częściowego porządku na zbiorze $\overline{T}_i$

  Przyjmiemy następującą notację:

- $r_i(x)$ lub $r_i(x, wartość)$
- $w_i(x)$ lub $w_i(x, wartość)$
- c_i lub a_i

Formalny model transakcji przedstawiono na slajdzie. Transakcją T_i nazywamy uporządkowaną parę: <zbiór operacji na bazie danych, relacja częściowego porządku na zbiorze tych operacji>. Zbiór operacji zawiera: odczyt (R), zapis (W), zatwierdzenie transakcji (C), wycofanie transakcji (A).

W dalszej części wykładu będziemy stosowali następującą notację:

- $r_i(x)$ oznacza odczyt danej x przez i-tą transakcję;

- $r_i(x, wartość)$ oznacza odczyt danej x przez i-tą transakcję, przy czym 'wartość' jest aktualnie odczytaną wartością danej x;

- $w_i(x)$ oznacza zapis danej x przez i-tą transakcję;

- $w_i(x, wartość)$ oznacza zapis danej x przez i-tą transakcję, przy czym 'wartość' jest aktualnie zapisywaną wartością danej x;

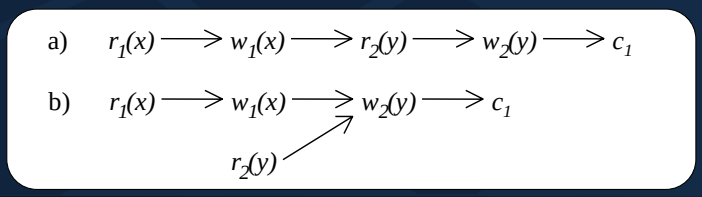
- c_i oznacza zatwierdzenie i-tej transakcji;

- a_i oznacza wycofanie i-tej transakcji.

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:

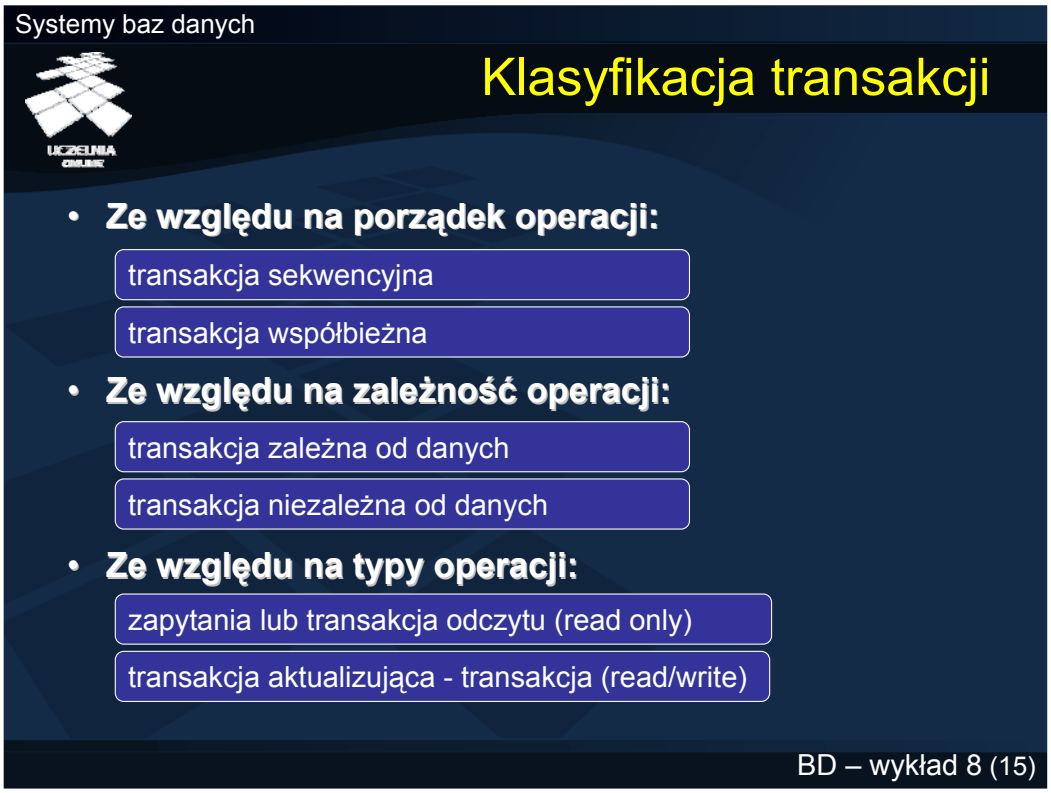


Każda transakcja może być reprezentowana przez graf skierowany: G = (V, A), gdzie:

- V jest zbiorem węzłów odpowiadających operacjom transakcji T_i;

- A jest zbiorem krawędzi reprezentujących porządek na zbiorze operacji.

Dwa przykłady grafu transakcji przedstawiono na slajdzie. W pierwszym z nich, pierwsza operacja transakcji pierwszej odczytuje daną x ($r_1(x)$), następnie zapisuje/modyfikuje tę daną ($w_1(x)$). Pierwszą operacją drugiej transakcji jest operacja odczytu danej y ($r_2(y)$), następną operacją jest zapis danej y ($w_2(y)$) przez transakcję drugą. Ostatnią jest operacja zatwierdzenia transakcji pierwszej (c_1). Pierwszy przykład reprezentuje sekwencyjnie wykonywane transakcje. Drugi przykład reprezentuje współbieżnie wykonywane transakcje.



Transakcje można podzielić ze względu na wiele kryteriów. Na potrzeby wykładu wprowadzimy trzy kryteria podziału:

- porządek operacji,

- zależność operacji,

- typ operacji.

Zgodnie z pierwszym kryterium wyróżnia się transakcje realizowane sekwencyjnie (tj. jedna po drugiej) i transakcje realizowane współbieżnie (tj. równocześnie). Zgodnie z drugim kryterium wyróżnia się transakcje zależne od danych i transakcje niezależne od danych. W transakcji zależnej do danych, zbiór danych adresowanych przez transakcję może nie być w całości znany w momencie rozpoczęcia transakcji. Zbiór ten jest określany dynamicznie w trakcie pracy transakcji, zależnie od danych przetworzonych przez wcześniejsze polecenia. Ze względu na trzecie kryterium wyróżnia się transakcje wyłącznie odczytujące dane i transakcje modyfikujące dane.

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, ..., 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 $<T_i$

W praktyce, w jednym systemie bazy danych działa równocześnie wiele transakcji. Należy zapewnić, aby transakcje te były wykonane w takiej kolejności, która nie wprowadzi danych niepoprawnych.

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, ..., 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 częściowy porządek w zbiorze operacji transakcji T_i.

Realizacje transakcji (2)

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

- gdzie:

1. $\overline{T}_r(\tau)$ zbiór operacji wszystkich transakcji należących do zbioru τ
2. $< r$ relacja częściowego porządku na zbiorze $\overline{T}_r(\tau)$,
3. Dla dowolnej pary operacji o_i, $o_j \in \overline{T}_r(\tau)$, 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 $o_j <_r o_i$

Formalną notację realizacji **S** zbioru transakcji (oznaczonego jako TAU) przedstawiono na slajdzie. Jest to para: zbiór operacji wszystkich transakcji należących do zbioru TAU i relacja częściowego porządku na zbiorze operacji należących do transakcji ze zbioru TAU. Dla dowolnej pary operacji o_i, o_j należących do zbioru operacji transakcji ze zbioru τ 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 tyko realizacji spełniających powyższy warunek)

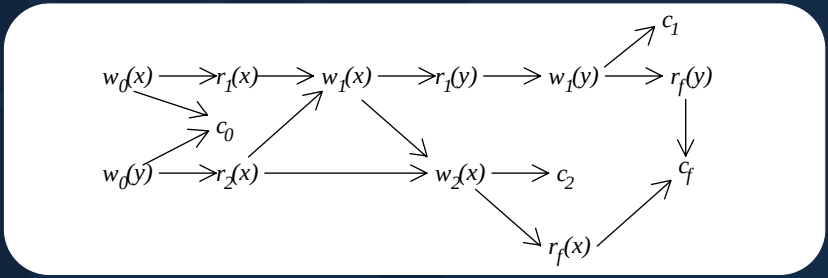
Przykład:
r: $w_0(x)$, $w_0(y)$, c_0, $r_1(x)$, $r_2(x)$, $w_1(x)$, $r_1(y)$, $w_2(x)$, c_2, $w_1(y)$, c_1, $r_f(x)$, $r_f(y)$, c_f;

Realizacja zawierająca tylko operacje zatwierdzonych transakcji nazywana jest zaakceptowaną projekcją. Dalsze rozważania dotyczyć będą tyko realizacji spełniających ten warunek.

Jako przykład zaakceptowanej projekcji rozważmy realizację przedstawioną na slajdzie. Transakcja T_0 zapisuje daną x ($w_0(x)$), następnie daną y ($w_0(y)$) i zatwierdza te operacje (c_0). Następnie transakcje T_1 i T_2 są realizowane współbieżnie. T_1 odczytuje daną x ($r_1(x)$), następnie T_2 ($r_2(x)$) odczytuje tę samą daną x. Kolejne operacje to: zapis danej x przez T_1 ($w_1(x)$), odczyt danej y przez T_1 ($r_1(y)$), zapis danej x przez T_2 ($w_2(x)$), zatwierdzenie T_2 (c_2), zapis danej y przez T_1 ($w_1(y)$), zatwierdzenie T_1, odczyt danej x przez T_f, odczyt danej y przez T_f i zatwierdzenie T_f.

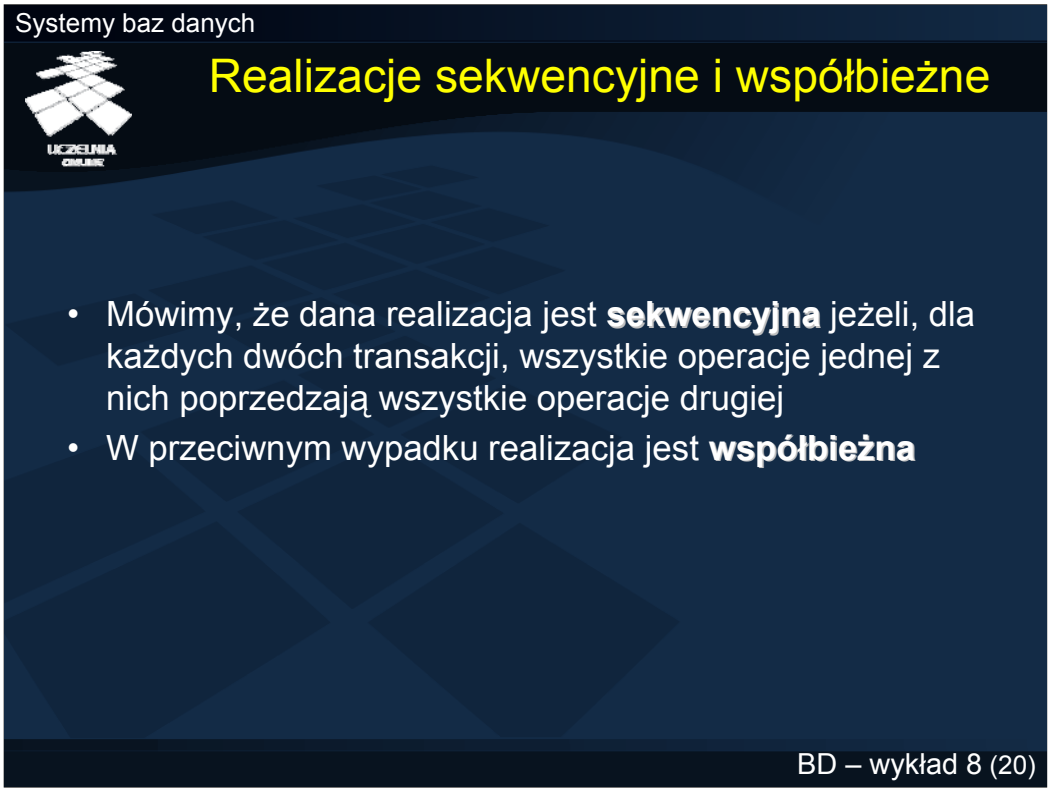
Realizacje transakcji (4)

- Dowolną realizację można przedstawić w postaci grafu skierowanego, nazywanego grafem realizacji, *GR(s(τ)) = (V, A)*. Węzły grafu odpowiadają operacjom ze zbioru $\overline{T}_r(\tau)$, natomiast krawędzie grafu reprezentują relację częściowego porządku < *r*
- **Przykład:**

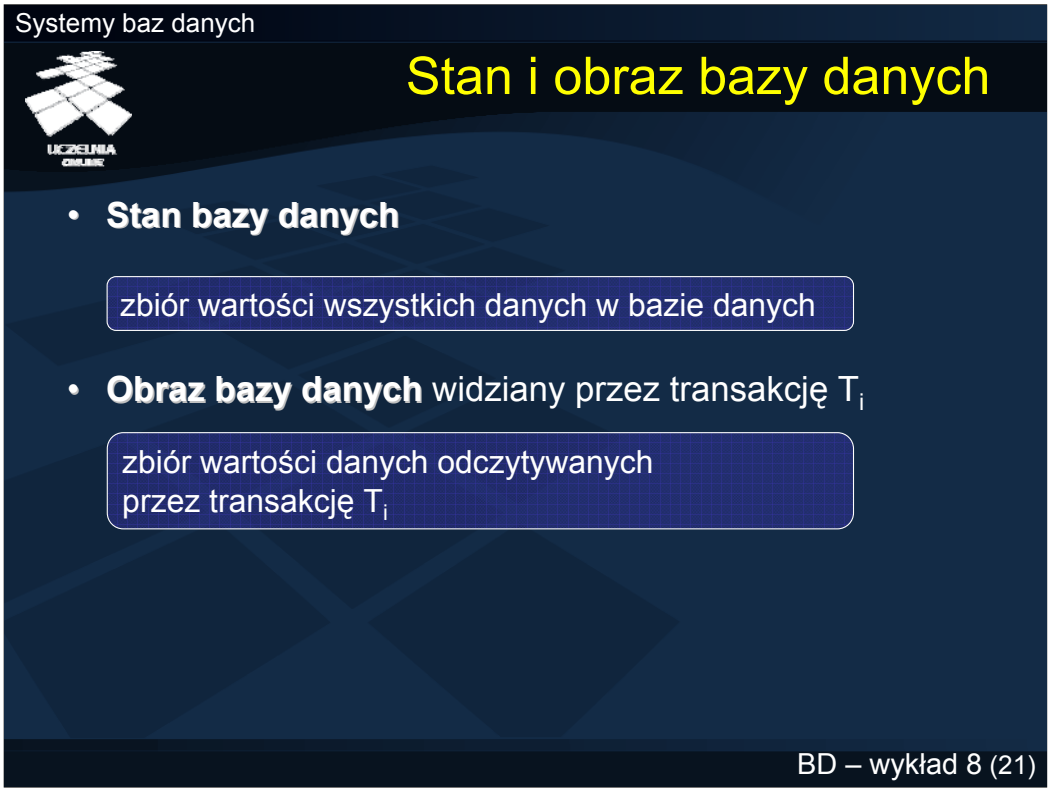


Dowolną realizację można przedstawić w postaci grafu skierowanego, nazywanego grafem realizacji. Węzły grafu odpowiadają operacjom ze zbioru operacji należących do transakcji natomiast krawędzie grafu reprezentują relację częściowego porządku na zbiorze tych operacji.

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

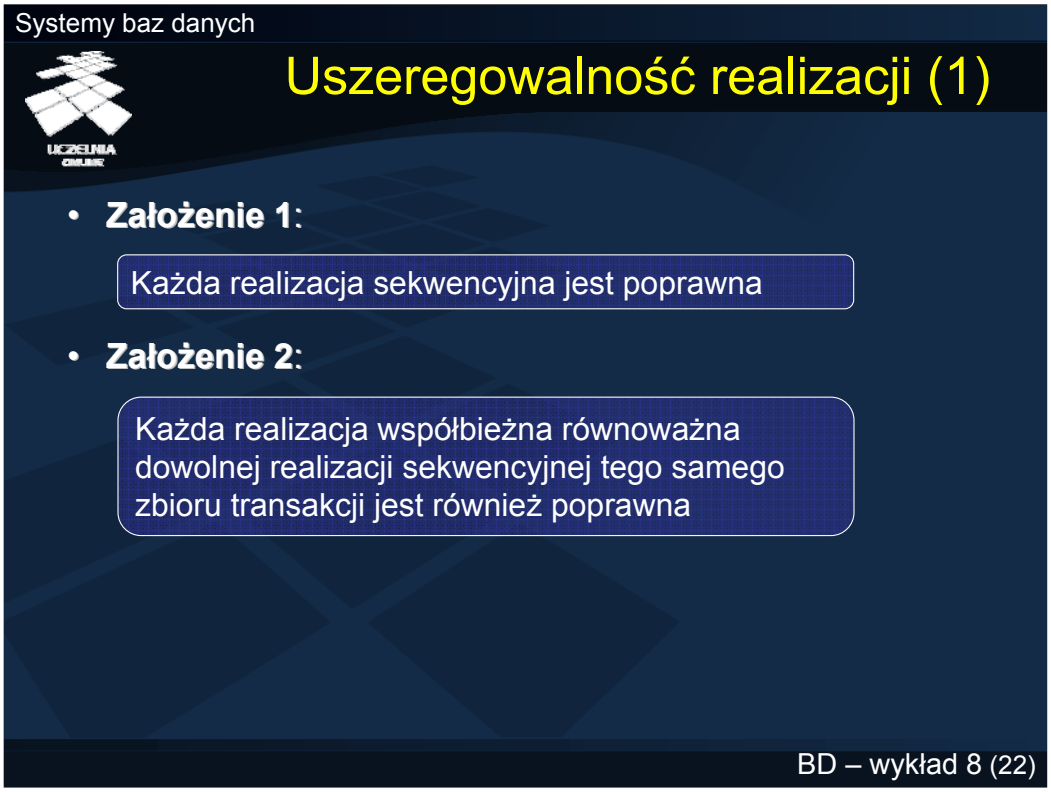


Mówimy, że dana realizacja jest sekwencyjna jeżeli, dla każdych dwóch transakcji, wszystkie operacje jednej z nich poprzedzają wszystkie operacje drugiej. W przeciwnym wypadku realizacja jest współbieżna.



W kontekście zarządzania transakcjami należy wprowadzić pojęcie stanu bazy danych i obrazu bazy danych.

Stan bazy danych reprezentuje zbiór wartości wszystkich danych w bazie. Obraz bazy danych widziany przez transakcję T_i jest zbiorem wartości danych odczytywanych przez transakcję T_i.



Przejdziemy teraz do omówienia problematyki uszeregowalności realizacji. Przyjmiemy następujące założenia:

- każda realizacja sekwencyjna jest poprawna;

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

Uszeregowalność realizacji (2)

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?

Jako przykład rozważmy transakcję T1, która sumuje wartości konta a i konta b i transakcję T2, która przelewa 30 z konta a na konto b. Załóżmy, że początkowa wartość konta a=50 i knota b=50. Przedstawiona na slajdzie realizacja nie jest poprawna ponieważ obraz bazy danych widziany przez transakcję T1 to a+b=70, zamiast 100.

Uszregowalność realizacji (3)

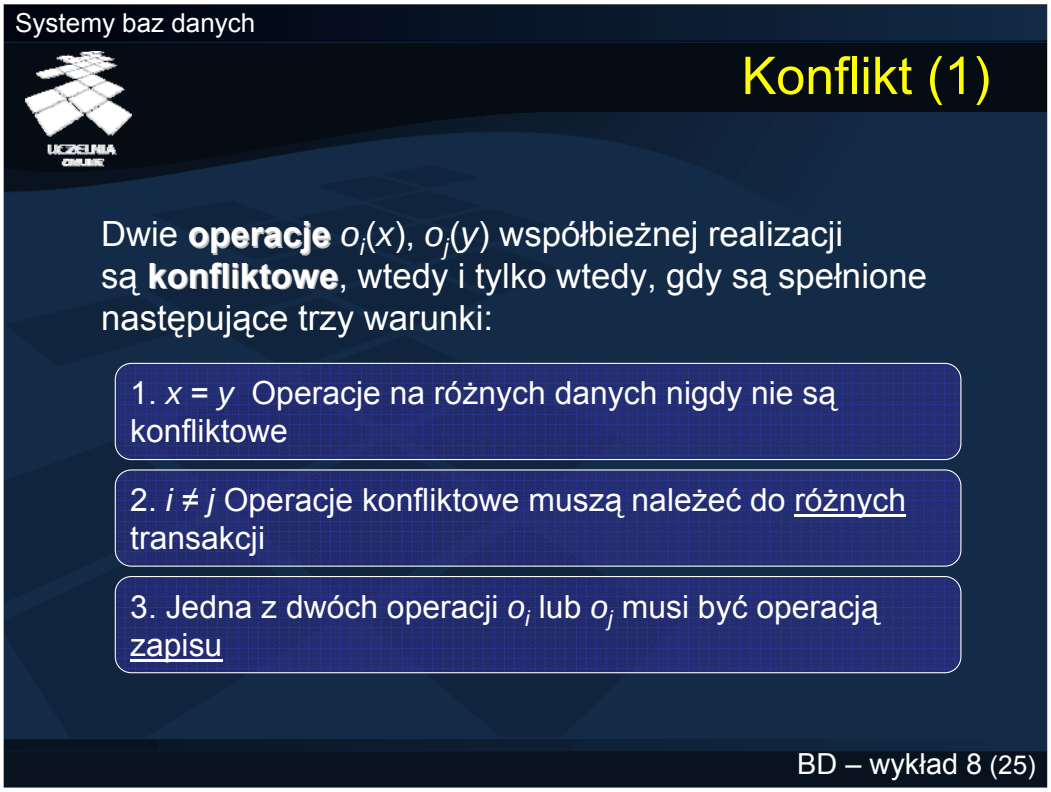
Realizacje sekwencyjne transakcji T1 i T2:

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.



Jeżeli przynajmniej dwie operacje należące do różnych transakcji realizują dostęp do tej samej danej i przynajmniej jedna z tych operacji jest modyfikacją/zapisem danej, wówczas występuje konflikt w dostępie do tej danej. Bardziej formalnie: mówimy, że dwie operacje $o_i(x)$, $o_j(y)$ współbieżnej realizacji są konfliktowe, wtedy i tylko wtedy, gdy są spełnione następujące trzy warunki.

Po pierwsze, gdy dotyczą tej samej danej. Innymi słowy, operacje na różnych danych nigdy nie są konfliktowe.

Po drugie, operacje konfliktowe muszą należeć do różnych transakcji.

Po trzecie, jedna z dwóch operacji o_i lub o_j musi być operacją zapisu.

Konflikt (2)

- 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 $o_i(x)\ o_j(y)$, jeżeli operacje te są konfliktowe i $o_i(x) <_r o_j(y)$
- Następujące pary operacji mogą znajdować się w konflikcie:
- $r_i(x)\ w_j(x)$
- $w_i(x)\ r_j(x)$
- $w_i(x)\ w_j(x)$

Pojęcie konfliktu można rozszerzyć na zbiór transakcji. Mówimy, że dwie transakcje Ti, Tj są konfliktowe, jeżeli zawierają wzajemnie konfliktowe operacje. Wprowadzimy obecnie pojęcie relacji poprzedzania operacji w realizacji r(TAU). Mówimy, że operacja oi(x) znajduje się w relacji poprzedzania z operacją oj(y) w realizacji r(TAU), co zapisujemy jako oi(x) -> oj(y), jeżeli operacje te są konfliktowe i operacja oi(x) poprzedza w realizacji r(TAU) operację oj(y).

Łatwo zauważyć, że następujące pary operacji mogą znajdować się w konflikcie:

- *ri*(*x*) i *wj*(*x*),

- *wi*(*x*) i *rj*(*x*),

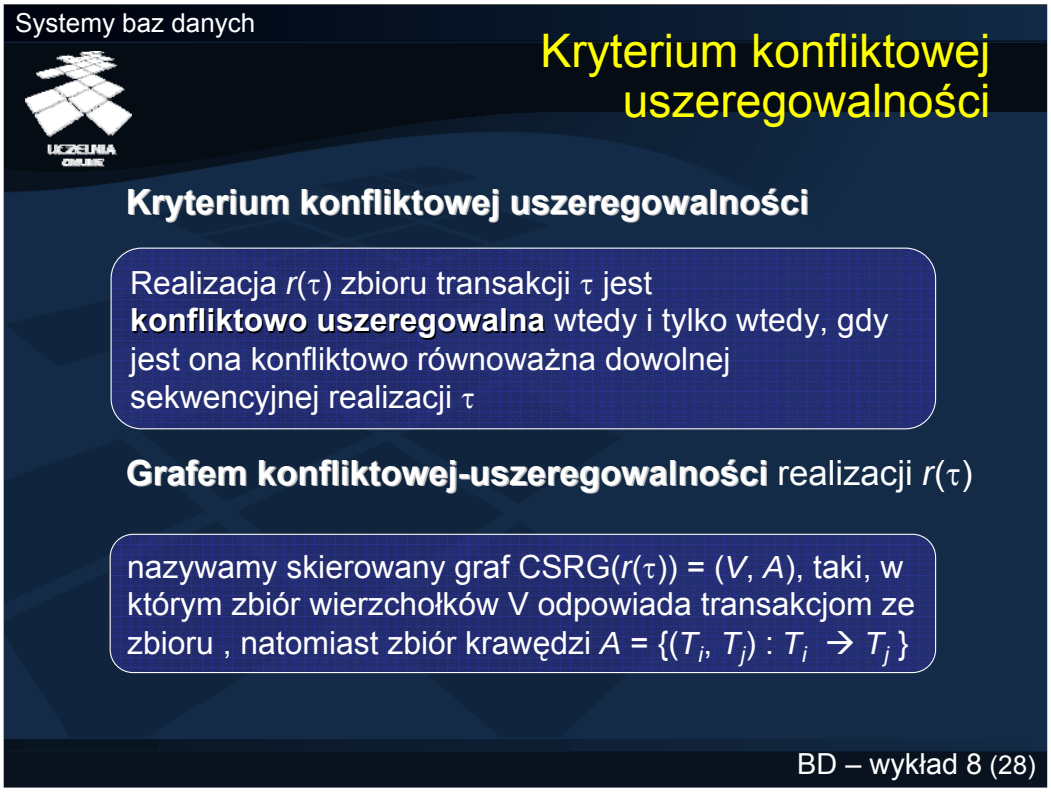
- *wi*(*x*) i *wj*(*x*).

Konfliktowa równoważność

- Mówimy, że **transakcja T_i poprzedza transakcję** T_j w realizacji $r(\tau)$, co zapisujemy jako $Ti \rightarrow Tj$, jeżeli zawierają odpowiednio operacje $o_i(x)$ i $o_j(x)$, między którymi zachodzi związek poprzedzania
- Mówimy, że dwie realizacje $r(\tau) = (\overline{T}_r(\tau) , <r)$ i $r'(\tau) = (\overline{T}_r(\tau) , <r')$ są **konfliktowo równoważne**, jeżeli dla każdej pary operacji $o_i(x)$ i $o_j(y)$ w realizacji $r(\tau)$, takich, że $o_i(x) \rightarrow o_j(x)$, zachodzi również $o_i(x) \rightarrow o_j(y)$ w realizacji $r'(\tau)$

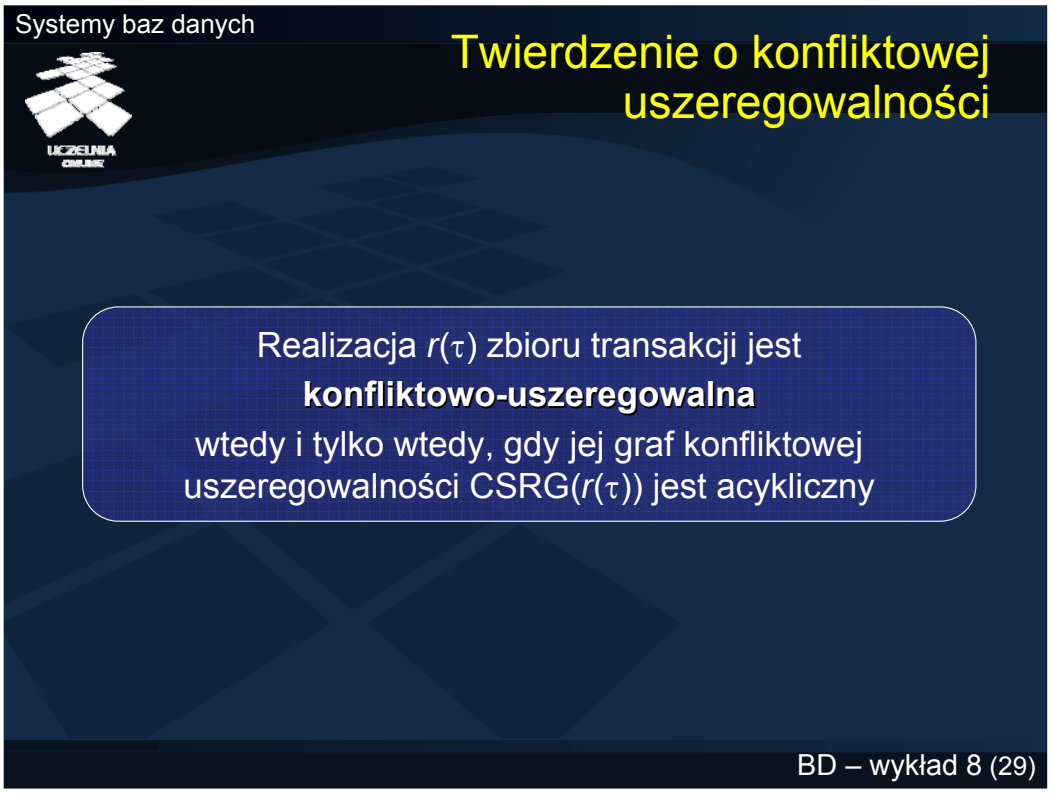
Relacje poprzedzania można również rozszerzyć na zbiór transakcji. Mówimy, że transakcja Ti jest w relacji poprzedzania z transakcją *Tj* w realizacji *r*(TAU), co zapisujemy jako *Ti* -> *Tj*, jeżeli transakcje te zawierają odpowiednio operacje *oi*(*x*) i *oj*(*x*), między którymi zachodzi relacja poprzedzania. Przypomnijmy, że zgodnie z założeniem 2, każda realizacja współbieżna równoważna dowolnej realizacji sekwencyjnej tego samego zbioru transakcji jest poprawna. Jak już wspominaliśmy, kluczowe, w powyższej definicji, jest pojęcie równoważności.

Obecnie, po wprowadzeniu relacji poprzedzania, możemy formalnie zdefiniować pojęcie równoważności dwóch realizacji. Mówimy, że dwie realizacje *r*(TAU) = (Tr(TAU) , <*r*) i *r'*(TAU) = (Tr(TAU) , <*r'*) są **konfliktowo równoważne**, jeżeli dla każdej pary operacji *oi*(*x*) i *oj*(*x*) w realizacji *r*(TAU), takich, że *oi*(*x*) -> *oj*(*y*), zachodzi również *oi*(*x*) -> *oj*(*y*) w realizacji *r'*(TAU). Obecnie, sformułujemy kryterium poprawności współbieżnej realizacji zbioru transakcji nazywane kryterium konfliktowej uszeregowalności.



Definicja kryterium konfliktowej uszeregowalności brzmi następująco. Realizacja *r*(TAU) zbioru transakcji T jest **konfliktowo uszeregowalna** wtedy i tylko wtedy, gdy jest ona konfliktowo równoważna dowolnej sekwencyjnej realizacji zbioru TAU.

W jaki sposób można zweryfikować czy dana realizacja współbieżna spełnia kryterium konfliktowej uszeregowalności? W celu weryfikacji konfliktowej uszeregowalności realizacji konstruujemy graf konfliktowej uszeregowalności realizacji. Grafem konfliktowej uszeregowalności realizacji *r*(TAU) nazywamy skierowany graf CSRG(*r*(TAU)) = (*V*, *A*), taki, w którym zbiór wierzchołków V odpowiada transakcjom ze zbioru T, natomiast zbiór krawędzi zawiera relacje poprzedzania transakcji Ti i Tj: *A* = {(*Ti*, *Tj*) : *Ti* -> *Tj*}.



Możemy obecnie, korzystając z grafu konfliktowej uszeregowalności sformułować twierdzenie, pozwalające w sposób algorytmiczny weryfikować, czy dana realizacja współbieżna jest poprawna, tj. konfliktowo-uszeregowalna. Realizacja *r*(TAU) zbioru transakcji T jest konfliktowo-uszeregowalna wtedy i tylko wtedy, gdy jej graf konfliktowej uszeregowalności CSRG(*r*(TAU)) jest acykliczny.

Poprawność powyższego twierdzenia wynika bezpośrednio z własności spójności transakcji. Zgodnie z własnością spójności, każda transakcja transformuje bazę danych z jednego stanu spójnego do innego stanu spójnego. Stąd wynika, że każda realizacja sekwencyjna zbioru transakcji zachowuje spójność bazy danych, gdyż jest ona sekwencją transformacji odwzorowujących bazę danych z jednego do innego stanu spójnego. Z definicji grafu konfliktowej uszeregowalności wynika, że graf ten, dla dowolnej realizacji sekwencyjnej, musi być acykliczny. Z definicji kryterium konfliktowej uszeregowalności wynika, że dowolna realizacja współbieżna jest poprawna, jeżeli jest ona równoważna dowolnej realizacji sekwencyjnego tego samego zbioru transakcji. Z definicji równoważności realizacji wynika, że graf konfliktowej uszeregowalności realizacji współbieżnej musi być również acykliczny, jeżeli realizacja ta jest konfliktowo-uszeregowalna. Co kończy skrótowy dowód poprawności podanego twierdzenia.

Systemy baz danych

Realizacje odtwarzalne (1)

- Czy własność uszeregowalności gwarantuje wolność od anomalii ?

  Przykład:
  $H = r_1[x]\ w_1[x]\ r_1[y]\ r_2[x]\ w_1[y]\ r_2[y]\ c_2\ r_1[z]\ w_1[z]$ <crash> c_1

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

Można sformułować następujące pytanie: Czy własność uszeregowalności gwarantuje poprawność dowolnej realizacji transakcji, w szczególności, czy gwarantuje wolność od anomalii współbieżnego wykonywania transakcji? Odpowiedź na to pytanie jest twierdząca, jeżeli rozważamy wyłącznie realizacje będące zaakceptowanymi projekcjami, tj. realizacje zawierające tylko operacje zatwierdzonych transakcji. W rzeczywistości, realizacje zawierają nie tylko operacje zatwierdzonych transakcji. Transakcje są wycofywane przez użytkowników, podlegają awariom, są wycofywane przez system np. na skutek wystąpienia zakleszczenia. Jeżeli rozważamy realizacje, które zawierają operacje wycofywanych transakcji, to odpowiedź na postawione na wstępie pytanie jest negatywna. Ilustruje to przykładowa realizacja przedstawiona na slajdzie. Realizacja ta jest uszeregowalna – po usunięciu operacji transakcji T1, której wykonanie zostało przerwane na skutek wystąpienia awarii w systemie, pozostała realizacja zawiera operacje zatwierdzonej transakcji T1. Niestety, realizacja ta, mimo, że uszeregowalna, nie jest wolna od anomalii brudnego odczytu.

Transakcja T2 odczytuje wartości danych x i y zapisane przez transakcję T1, która następnie, jest wycofywana. Zauważmy, że po restarcie systemu, transakcja T2 nie zostanie poprawnie odtworzona, gdyż powinna ona odczytać stan bazy danych sprzed wykonania transakcji T1.

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 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 $a_k < r_i[x]$

- Mówimy, że transakcja T_i ***czyta*** z transakcji T_j w realizacji H, jeżeli T_i **czyta** jakąś daną z transakcji T_j w realizacji H

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. Aby zdefiniować nowe niezbędne własności realizacji, konieczne jest wprowadzenie dodatkowych definicji.

Mówimy, że transakcja Ti czyta daną x z transakcji Tj w realizacji H jeżeli:

1. wj[x] < *ri*[x];

2. *aj* < *ri*[*x*]

3. jeżeli istnieje operacja wk[x] taka, że wj[x] < wk[x] < *ri*[*x*], wtedy *ak* < *ri*[*x*].

Mówimy, że transakcja Ti czyta z transakcji Tj w realizacji H, jeżeli Ti czyta dowolna daną z transakcji Tj w realizacji H.

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 $w_i[x]$

Korzystając z wprowadzonych pojęć, możemy obecnie zdefiniować nowe klasy realizacji transakcji.

Mówimy, że realizacja *H* jest odtwarzalna (ang. *recoverable*) (*RC*) wówczas, jeżeli transakcja *Ti czyta z* transakcji *Tj* (*i* różne od *j*) w realizacji *H* i *ci* należy do realizacji *H*, to *cj* < *ci*

Mówimy, że realizacja *H* unika kaskadowych wycofań (ang. *avoids cascading aborts*) (*ACA*) wówczas, jeżeli transakcja *Ti czyta z* transakcji *Tj* (*i* różne od *j*), to *cj* < *ri* [*x*]

Mówimy, że realizacja *H* jest ścisła (ang. *strict*) (*ST*) wówczas, jeżeli *wj* [*x*] < *oi* [*x*] (*i* różne od *j*), zachodzi *aj* < *oi* [*x*] lub *cj* < *oi* [*x*], gdzie *oi* [*x*] jest jedną z operacji *ri* [*x*] lub *wi* [*x*]

Realizacje odtwarzalne (3)

- Przykład:

$T_1 = w_1[x]\ w_1[y]\ w_1[z]\ c_1$
$T_2 = r_2[u]\ w_2[x]\ r_2[y]\ w_2[y]\ c_2$

$H_1 = w_1[x]\ w_1[y]\ r_2[u]\ w_2[x]\ r_2[y]\ w_2[y]\ c_2\ w_1[z]\ c_1$
$H_2 = w_1[x]\ w_1[y]\ r_2[u]\ w_2[x]\ r_2[y]\ w_2[y]\ w_1[z]\ c_1\ c_2$
$H_3 = w_1[x]\ w_1[y]\ r_2[u]\ w_2[x]\ w_1[z]\ c_1\ r_2[y]\ w_2[y]\ c_2$
$H_4 = w_1[x]\ w_1[y]\ r_2[u]\ w_1[z]\ c_1\ w_2[x]\ r_2[y]\ w_2[y]\ c_2$

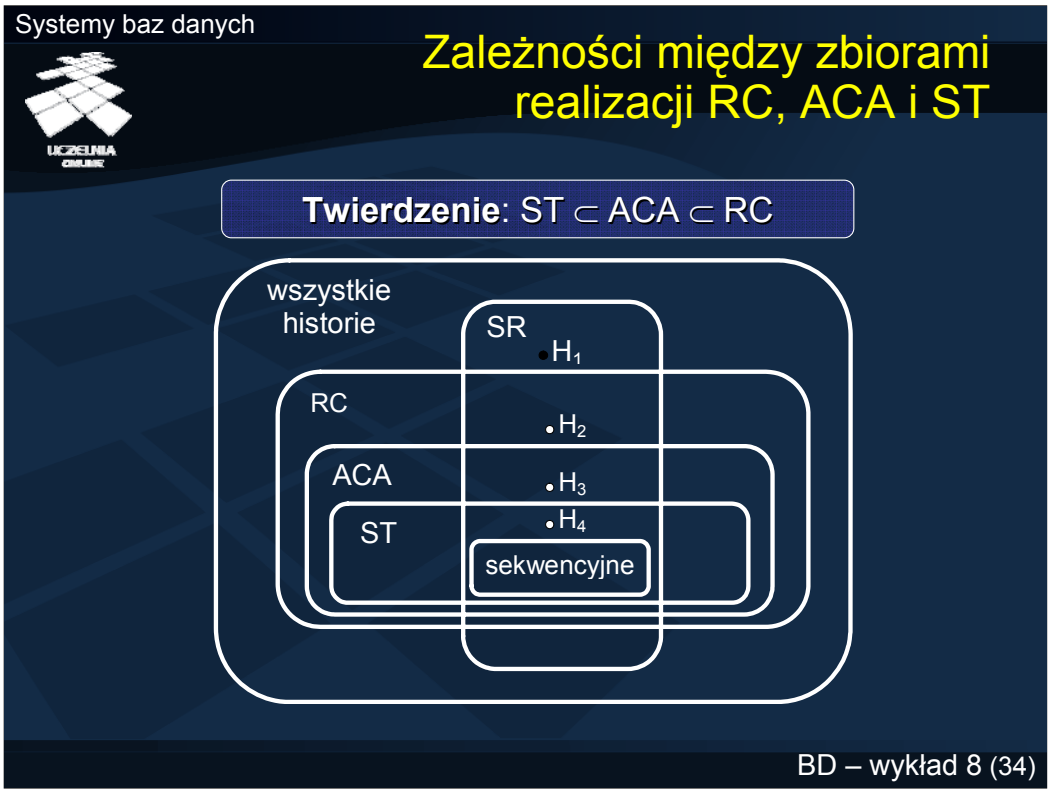
Dla ilustracji nowo wprowadzonych klas realizacji transakcji rozważmy przykładowe realizacje przedstawione na slajdzie. Dane są dwie transakcje T1 i T2 przedstawione na slajdzie.

Rozważmy realizację H1. Realizacja H1 jest realizacją konfliktowo-uszeregowalną, ale nie jest realizacja odtwarzalną. Transakcja T2 czyta daną y z transakcji T1, ale c2 < c1. Łatwo również zauważyć, że realizacja H1 nie należy do klasy realizacji ACA jak również ST.

Rozważmy realizację H2. Realizacja H2 jest realizacją konfliktowo-uszeregowalną. Ponadto, jest również realizacją odtwarzalną. Transakcja T2 czyta daną y z transakcji T1, ale tym razem c1 < c2. Realizacja H2 nie należy do klasy realizacji ACA jak również ST.

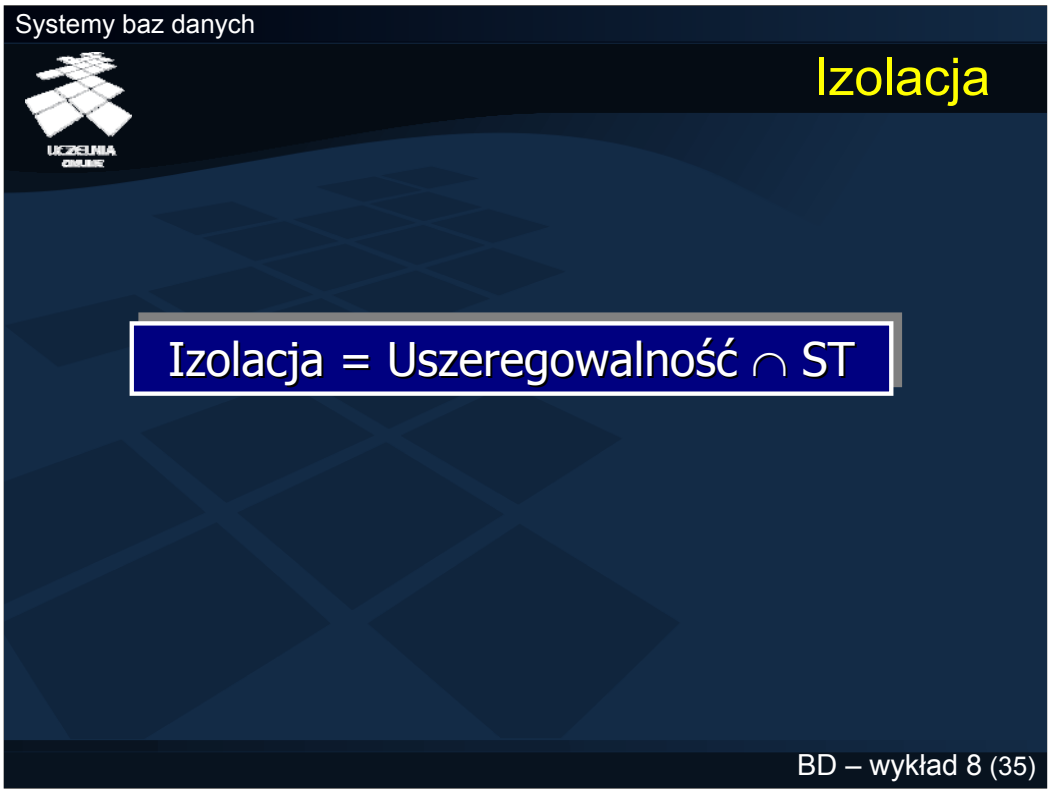
Rozważmy realizację H3. Realizacja H3 jest realizacją konfliktowo-uszeregowalną i odtwarzalną. Ponadto, jest ona również realizacja unikającą kaskadowych wycofań (należy do klasy ACA). Transakcja T2 czyta daną y z transakcji T1 i spełniony jest warunek c1 < r2[y]. Realizacja H3 nie jest natomiast realizacją ścisłą.

Wreszcie, rozważmy realizację H4. Realizacja H4 jest realizacją konfliktowo-uszeregowalną, odtwarzalną unikającą kaskadowych wycofań oraz realizacją ścisłą.

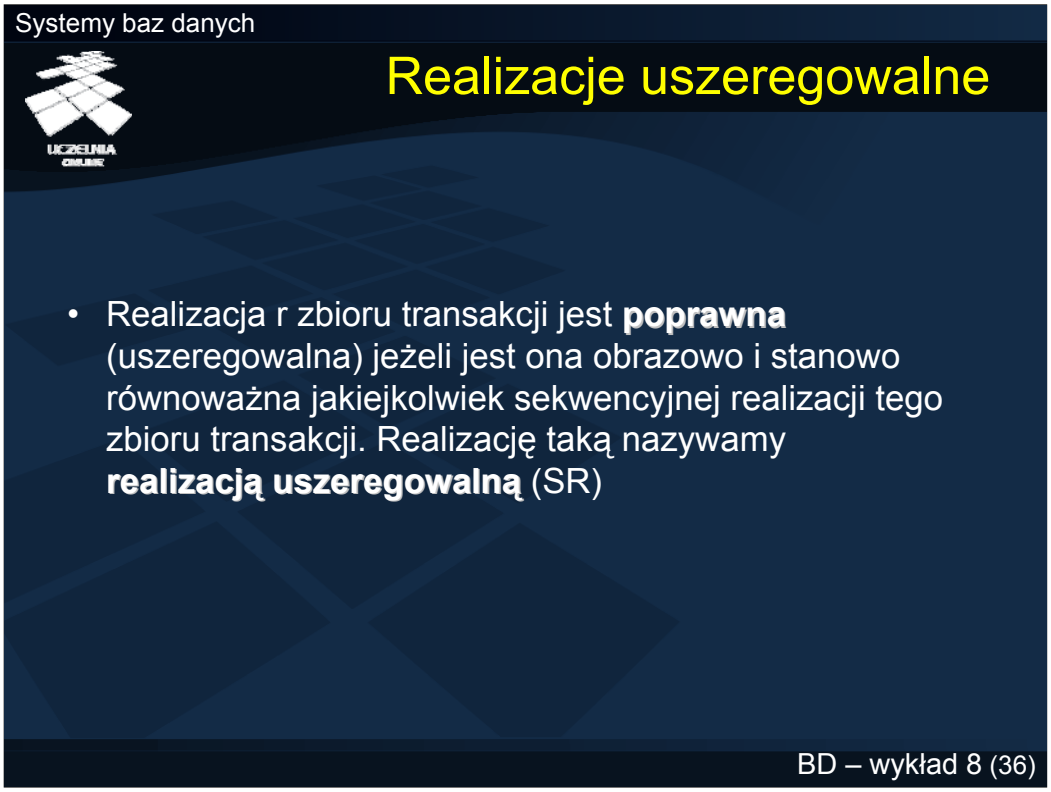


Można sformułować i udowodnić następujące twierdzenie określające zależności pomiędzy zbiorami realizacji RC, ACA i ST.

Zbiór realizacji ścisłych zawiera się w zbiorze realizacji unikających kaskadowych wycofań, który, z kolei, zawiera się w zbiorze realizacji odtwarzalnych. Diagram przedstawiony na slajdzie ilustruje zależności pomiędzy zbiorami realizacji RC, ACA i ST. Na diagramie zaznaczono również przynależność przykładowych realizacji H1, H2, H3 i H4.



Mówiąc o własnościach transakcji, stwierdziliśmy, że jedną z czterech własności transakcji jest własność izolacji. Korzystając z wprowadzonych dotychczas pojęć, własność izolacji transakcji możemy zdefiniować formalnie. Własność izolacji transakcji oznacza, że transakcja jest konfliktowo-uszeregowalna i ścisła.



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. Formalnie, definicja kryterium uszeregowalności brzmi następująco: realizacja r(T) zbioru transakcji T jest poprawna (uszeregowalna), jeżeli jest ona obrazowo i stanowo równoważna jakiejkolwiek sekwencyjnej realizacji zbioru transakcji T. Realizację r(T), spełniającą zdefiniowane powyżej kryterium, nazywamy realizacją uszeregowalną (należy do klasy SR). Jak łatwo zauważyć, równoważność konfliktowa realizacji została zastąpiona w kryterium uszeregowalności równoważnością obrazową i stanową realizacji. Kryterium uszeregowalności ma charakter bardziej egzystencjalny, jest natomiast mało przydatne w sensie konstrukcyjnym.

Graf uszergowalności (1)

- **Grafem uszeregowalności** realizacji r(τ) 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 Ti : r(x) czyta wartość danej x zapisanej przez operację Tj : 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 $(T_i, T_k) \in A$
3. Jeżeli $T_j \neq T_0$, to $(T_0, T_j) \in A$

W celu weryfikacji uszeregowalności realizacji konstruujemy graf uszeregowalności realizacji. Grafem uszeregowalności realizacji *r*(T) nazywamy skierowany graf SG(*r*(T)) = (*V*, *A*), taki, w którym zbiór wierzchołków V odpowiada transakcjom ze zbioru T, natomiast definicja zbioru krawędzi została przedstawiona na prezentowanych slajdach.

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 uszeregowlaności SG($r(\tau)$)

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

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 weryfikacji, czy dana realizacja jest uszeregowalna, jest problemem NP.-zupełnym, to znaczy, problemem obliczeniowo trudnym. Trudność weryfikacji krytetium uszeregowalności wynika z konstrukcji tego grafu. Z definicji grafu uszeregowalności wynika, że wynikowy graf jest poligrafem (patrz warunek 2). Z teorii grafów wynika, że weryfikacja czy dany poligraf zawiera cykl jest problemem NP.-zupełnym. Stąd, w praktyce, kryterium uszeregowalności zastąpiono łatwiejszym do weryfikacji kryterium konfliktowej uszeregowalności. Problem weryfikacji, czy dana realizacja jest realizacją konfliktowo-uszeregowalną jest problemem obliczeniowo łatwym (problem należy do klasy problemów P).

Uszeregowalność transakcji - klasyfikacja

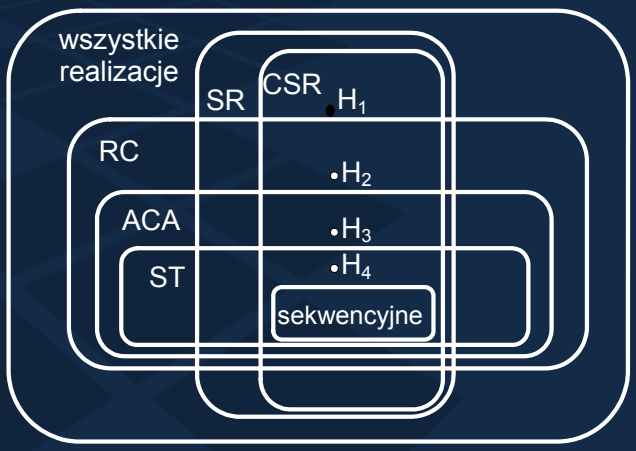


Diagram przedstawiony na slajdzie ilustruje zależności pomiędzy zbiorami realizacji RC, ACA i ST oraz zbiorami realizacji uszeregowalnych, konfliktowo uszeregowalnych oraz realizacji sekwencyjnych. Jak widać z diagramu, zbiór realizacji konfliktowo uszeregowalnych jest podzbiorem zbioru realizacji uszeregowalnych.