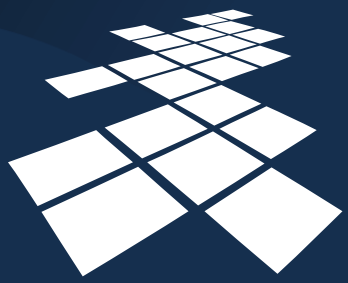
Bazy danych

Recovery – Transakcyjne odtwarzanie bazy danych po awarii

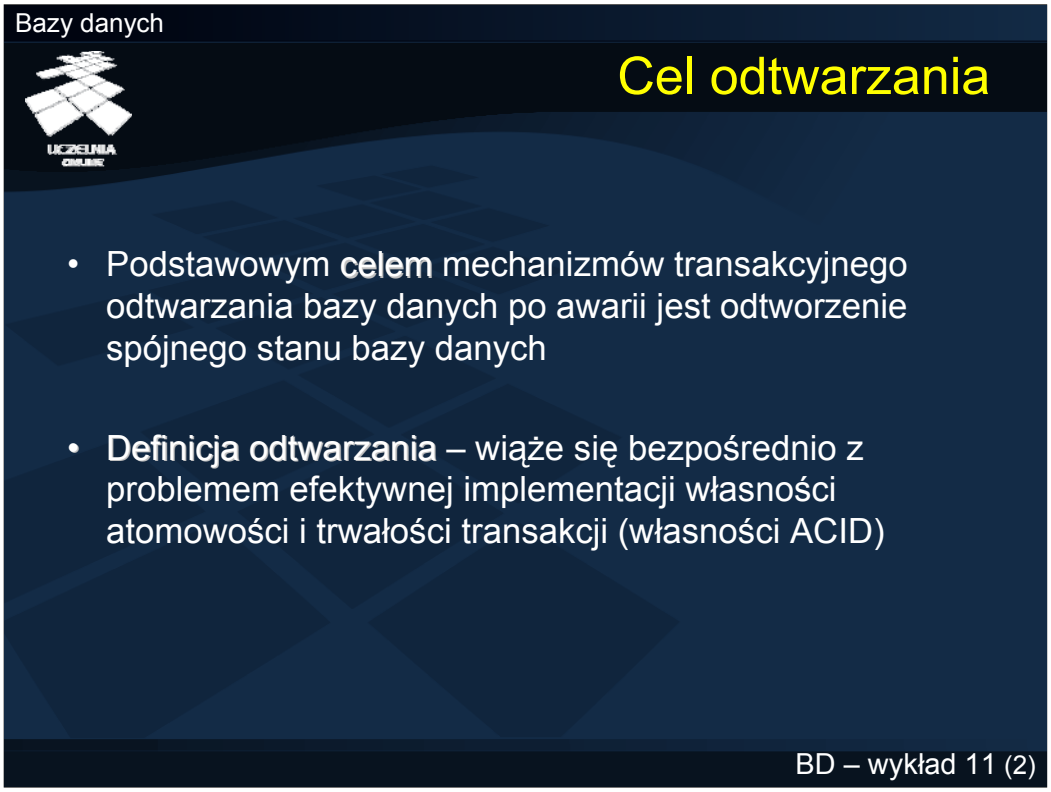
Wykład przygotował:
Tadeusz Morzy



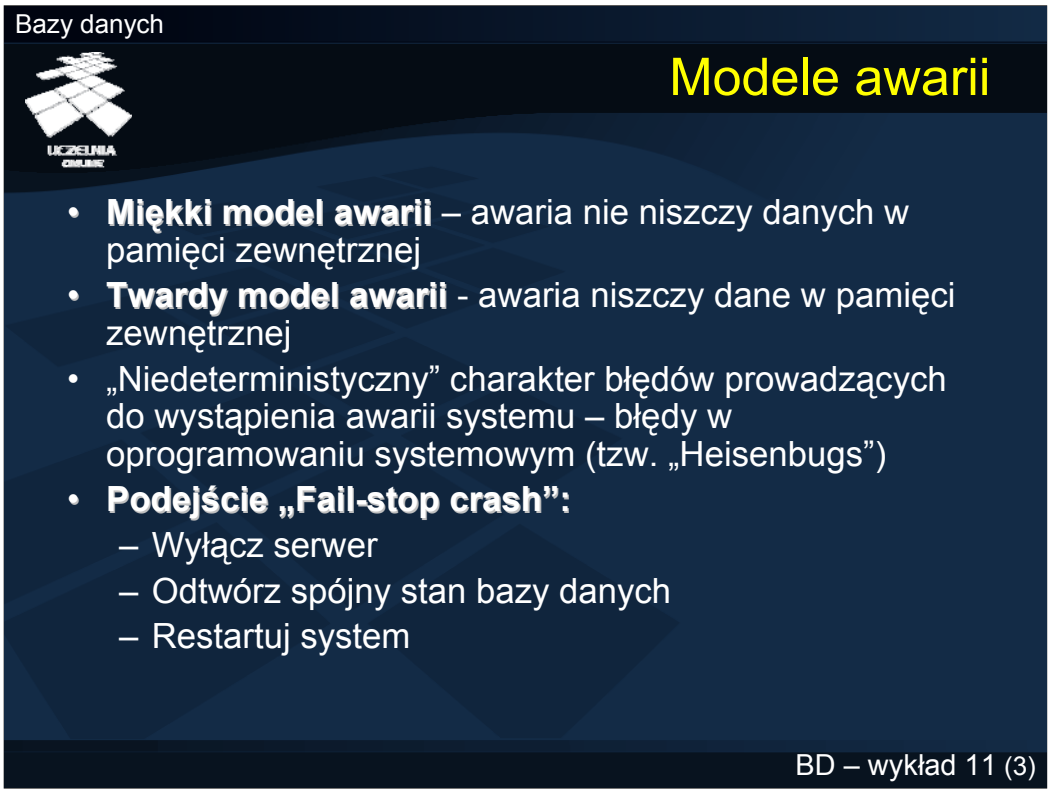
UCZELNIA
ONLINE

BD – wykład 11

Tematem wykładu jest problem odtwarzania spójnego stanu bazy danych po awarii. Rozpoczniemy od krótkiego wprowadzenia do modeli awarii. Następnie, przejdziemy do omówienia miary dostępności, jako podstawowej miary oceny efektywności mechanizmów odtwarzania, oraz do przedstawienia ogólnej architektury modułu odtwarzania. W kolejnej części, przedstawimy i omówimy generowanie rekordów logu oraz procedury Rollback oraz Roll Forward algorytmu odtwarzania spójnego stanu bazy danych. Przedyskutujemy problem poprawności przedstawionych procedur i omówimy strategię WAL, gwarantującą poprawność algorytmu odtwarzania. Na zakończenie, powiemy krótko o tak zwanych punktach kontrolnych pliku logu.



Podstawowym celem algorytmów transakcyjnego odtwarzania bazy danych po awarii jest odtworzenie spójnego stanu bazy danych sprzed awarii. Pierwsze pytanie, które się nasuwa, to jaki stan spójny chcemy odtworzyć? W trakcie wykonywania setek czy tysięcy transakcji, stan bazy danych, praktycznie, nigdy nie jest spójny! Odpowiedź na postawione pytanie wynika z definicji transakcji. Stan spójny bazy danych, który będzie odtworzony po awarii, to taki stan, który gwarantuje własności atomowości i trwałości wykonywanych transakcji (własności ACID). Do sformułowania poprawnego stanu bazy danych po awarii wrócimy później na wykładzie.



Zanim przejdziemy do dyskusji i prezentacji procedury odtwarzania bazy danych po awarii, wprowadzimy klasyfikację modeli awarii. Klasyfikacji awarii jest wiele – ta wprowadzona dla potrzeb wykładu wyróżnia dwa podstawowe typy awarii – tak zwane awarie miękkie i awarie twarde. Awaria miękka to awaria, która nie niszczy danych w pamięci zewnętrznej. Innymi słowy, awaria miękka to awaria, która nie niszczy mediów zewnętrznych. Awaria twarda to awaria, która niszczy dane w pamięci zewnętrznej. W dalszej części wykładu zakładamy model awarii miękkich, dla których opracowano systemowe mechanizmy przeciwdziałania. W przypadku awarii twardych, typowe rozwiązania sprowadzają się do replikacji danych i oprogramowania oraz sprzętu. Do grupy awarii miękkich zaliczamy te awarie, które prowadzą do utraty danych w pamięci operacyjnej. Są to, głównie, błędy oprogramowania systemowego i aplikacyjnego, które prowadzą do błędnego działania systemu, oraz zaniki napięcia, których skutki, z punktu wiedzenia działania systemu są podobne. Mówiąc o awariach, najczęściej, jako źródło awarii podaje się zanik napięcia. W praktyce ten rodzaj awarii występuje bardzo rzadko. Najczęstszym typem awarii są błędy w oprogramowaniu. Błędy te mają charakter niedeterministyczny. W przypadku, gdy występuje determinizm zdarzeń prowadzących do awarii, to ich źródło można usunąć. W przypadku błędów o charakterze niedeterministycznym mówimy często o tak zwanych błędach typu „Heisenbug" od nazwiska słynnego niemieckiego fizyka Wernera Heisenberga. Najlepszym sposobem rozwiązania problemów z błędami typu „Heisenbugs" jest podejście nazwane „fail_stop crash". Polega ono na: wyłączeniu serwera, odtworzeniu poprawnego (spójnego) stanu systemu w oparciu o mechanizm redo/undo w celu zapewnienia własności ACID, i na restarcie serwera.

Efektywność odtwarzania

- Konieczność minimalizacji czasu odtwarzania po awarii - w trakcie odtwarzania serwer i dane są niedostępne
- Miara efektywności odtwarzania – dostępność systemu, tj. prawdopodobieństwo, że próbkując losowo użytkownik znajduje system gotowy do realizacji żądań dostępu

$$\frac{MTTF}{MTTF + MTTR}$$

MTTF – średni czas do awarii
MTTR – średni czas odtworzenia po awarii

Mechanizm odtwarzania stanu spójnego powinien być poprawny, ale powinien być również efektywny. W trakcie odtwarzania systemu serwer i dane są niedostępne, stąd, konieczność minimalizacji czasu odtwarzania. Miarą efektywności procedury odtwarzania jest tak zwana dostępność systemu. Dostępność systemu definiuje się jako prawdopodobieństwo, że próbkując losowo system użytkownik znajduje ten system gotowy do realizacji żądań dostępu.

Dostępność

- Przykład: raz w miesiąca ma miejsce awaria serwera, odtworzenie stanu serwera zajmuje 2 godz., czas niedostępności serwera – 26 godz./rocznie
  dostępność = (720/722) = 99,7 %
- Przykład: awaria serwera ma miejsce raz na 48 godz., odtworzenie stanu serwera zajmuje 30 sek.
  dostępność = 99,98 %
- Wniosek: efektywność odtwarzania ma kluczowe znaczenie z punktu widzenia dostępności systemu
- Inna miara efektywności odtwarzania: ilość zasobów niezbędnych, podczas normalnego działania systemu, do tego, aby w razie awarii poprawnie odtworzyć stan systemu

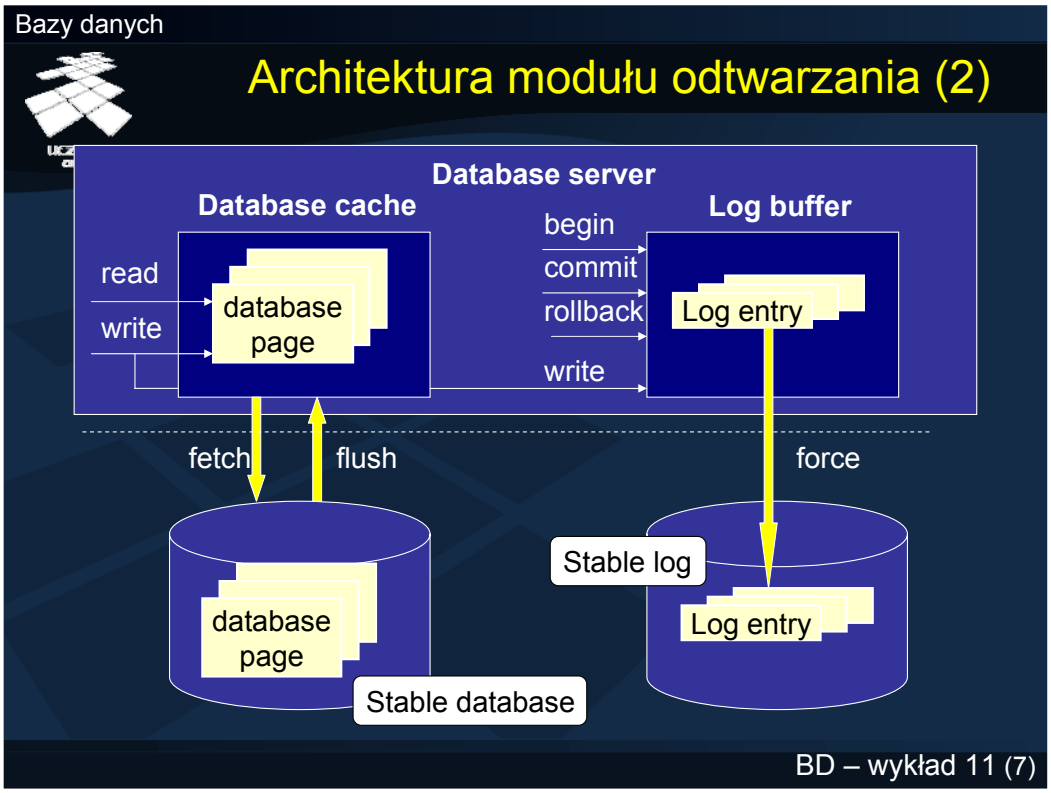
Od czego zależy dostępność systemu? Łatwo zauważyć, że podstawowym czynnikiem determinującym dostępność systemu jest czas odtworzenia systemu po awarii. Dla ilustracji rozważmy dwa przypadki. Załóżmy, że awaria serwera ma miejsce raz w miesiącu, natomiast czas odtworzenia poprawnego stanu serwera zajmuje 2 godz. Czas niedostępności serwera wynosi 26 godz. rocznie. Stąd, dostępność systemu wynosi = 99,7 %. Rozważmy inny przypadek. Załóżmy, że awaria serwera ma miejsce raz na 48 godz., odtworzenie stanu serwera zajmuje 30 sek. W takim przypadku dostępność wynosi= 99,98 %. Wniosek: efektywność odtwarzania ma kluczowe znaczenie z punktu widzenia dostępności systemu. Czasami przyjmuje się dodatkowe kryteria oceny efektywności procedury odtwarzania. Przykładowo - ilość zasobów niezbędnych, podczas normalnego działania systemu, do tego, aby w razie awarii poprawnie odtworzyć stan systemu.

Architektura modułu odtwarzania (1)

- Elementy składowe modułu odtwarzania :
  - Baza danych
  - Bufor bazy danych
  - Plik logu
  - Bufor logu
- Zawartość wymienionych składowych modułu odtwarzania całkowicie określa stan systemu
- Plik logu jako plik sekwencyjny typu „append-only”,
- Alternatywa – plik logu o dostępie swobodnym przechowujący zbiór wersji stron bazy danych (ang. shadow database)

Przejdziemy obecnie do przedstawienia podstawowych elementów składowych systemu odtwarzania. Zaliczamy do nich: bazę danych, bufor bazy danych, plik logu, oraz bufor logu.

Stan wymienionych składowych systemu odtwarzania całkowicie określa stan systemu. Podstawowym elementem składowym systemu odtwarzania jest plik logu. Plik logu jest implementowany, najczęściej, jako plik sekwencyjny typu „append-only”, jednakże prezentowany model odtwarzania jest na tyle ogólny, że dopuszcza również alternatywne implementacje logu (np. wersje stron bazy danych przechowywane w pliku o dostępie swobodnym – tzw. ***shadow database***)



System odtwarzania składa się z dwóch komplementarnych par elementów: bazy danych i bufora bazy danych, z jednej strony, z drugiej, z pliku logu i bufora logu. Baza danych jest przechowywana w pamięci zewnętrznej, która jest pamięcią nie ulotną. Nieformalnie, baza danych jest zbiorem plików, z których każdy składa się z sekwencji stron. Bufor bazy danych jest fragmentem pamięci operacyjnej (pamięci ulotnej), o ograniczonym rozmiarze. Jest to zbiór tzw. ramek, z których każda może pomieścić jedną stronę. Procesy systemu zarządzania bazą danych realizują dostęp do danych (odczyt i zapis rekordów znajdujących się na stronach) via bufor danych. Żądanie procesu Q dostępu do danej strony jest przesyłane do zarządcy bufora, który sprawdza, czy dana strona jest w buforze. Jeżeli tak, to adres strony w buforze jest zwracany do procesu. Jeżeli nie, to zarządca bufora sprowadza daną stronę z bazy danych do bufora (operacja fetch) i, podobnie jak poprzednio, zwraca adres strony w buforze do procesu. Ściągniecie strony z bazy danych do bufora wymaga, najczęściej, usunięcia jakieś strony z bufora (operacja flush) – tak zwana wymiana stron na żądanie. Usuwana strona z bufora jest zapisywana na dysk do bazy danych. Każda operacja aktualizacji zawartości strony w buforze danych, oraz operacje begin_transaction, commit_transaction oraz rollback_transaction, generują rekord logu. Rekordy logu są umieszczane w buforze logu, który, co pewien czas, jest zapisywany do pliku logu.

Operacje (1)

- Operacje transakcji:
  - Begin(T)
  - Commit(T)
  - Rollback(T)
  - Save(T)
  - Restore (T,s)
- Operacje na stronach bazy danych:
  - read[pageno, T]
  - write[pageno, T]
  - full-write[pageno, T] (blind-write, real action)
  - exec[op, obj, T]

Zbiór operacji wykonywanych w ramach systemu odtwarzania podzieliliśmy na operacje poszczególnych elementów składowych systemu. Operacje transakcji, które powodują generacje rekordów logu to: Begin(T) (początek transakcji), Commit(T) (akceptacja transakcji), Rollback(T) (wycofanie transakcji), Save(T) (zawieszenie wykonywania transakcji), oraz Restore (T,s) (odwieszenie wykonywania transakcji).

Operacje, które są wykonywane na stronach bazy danych w buforze to: read[pageno, T] (odczyt strony o zadanym numerze pageno), write[pageno, T] (zapis strony o zadanym numerze pageno), full-write[pageno, T] (zapis nowej strony o numerze pageno – strona pusta), oraz exec[op, obj, T] (wykonanie wskazanej operacji op na wskazanym obiekcie obj, który znajduje się na stronie w buforze).

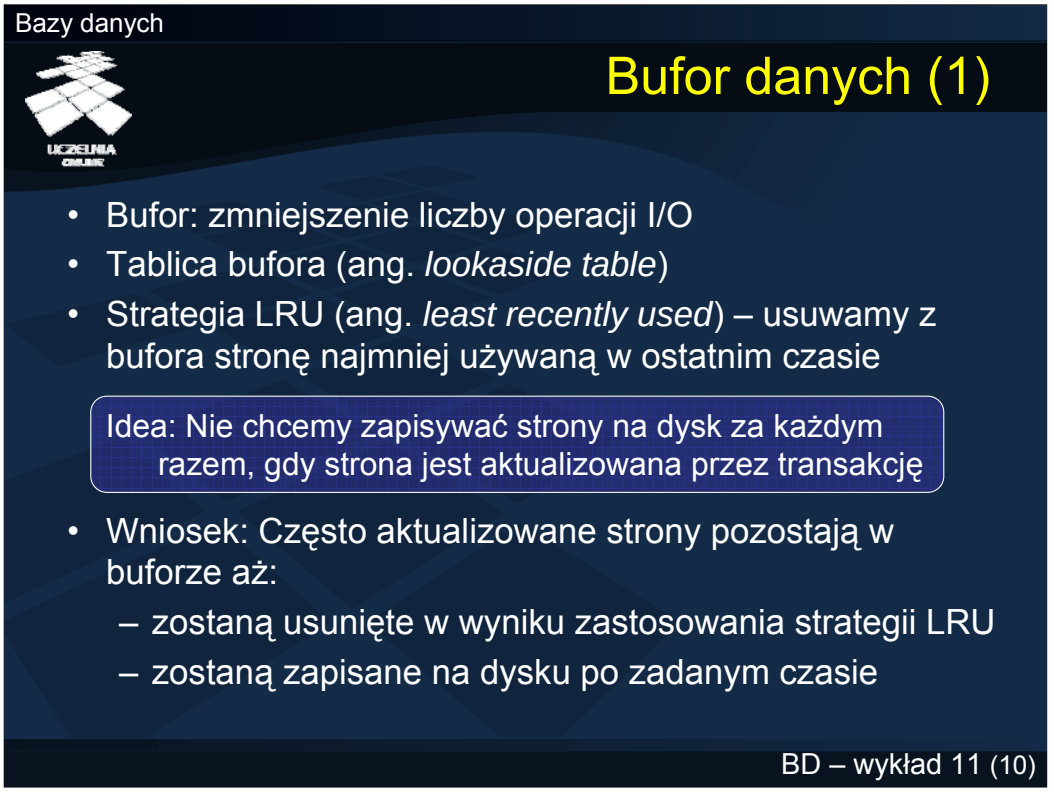
Operacje (2)

- Operacje na buforze danych:
  - **Fetch** [pageno] : pobierz stronę o identyfikatorze *pageno* z bazy danych do bufora danych,
  - **Flush** [pageno] : zapisz stronę o identyfikatorze *pageno* z bufora danych do bazy danych
- Operacje na pliku logu:
  - Force () : zapisz rekordy logu z bufora logu do pliku logu

Operacje na buforze danych:

Fetch [pageno] - pobierz stronę o identyfikatorze *pageno* z bazy danych do bufora danych,

Flush [pageno] - zapisz stronę o identyfikatorze *pageno* z bufora danych do bazy danych. Wreszcie, operacja na buforze logu: Force () : zapisz rekordy logu z bufora logu do pliku logu.



Jak już wspomnieliśmy, bufor bazy danych stanowi interfejs pomiędzy procesami zarządzającymi transakcjami a bazą danych. Dostęp do danych jest realizowany wyłącznie w obszarze bufora bazy danych. Ponieważ cała komunikacja transakcji z bazą danych jest realizowana poprzez bufor, efektywność procedur zarządzania buforem ma istotny wpływ na efektywność całego systemu bazy danych. Podstawową miarą oceny efektywności zarządzania buforem jest liczba operacji we/wy. Minimalizując liczbę operacji we/wy optymalizujemy efektywność działania systemu. Jaka jest zatem podstawowa strategia zarządzania buforem? Żądanie procesu Q dostępu do danej strony jest przesyłane do zarządcy bufora, który sprawdza, czy dana strona jest w buforze. W tym celu wykorzystuje tablicę haszową bufora (tzw. lookaside table). Jeżeli żądana strona jest w buforze, to adres strony w buforze jest zwracany do procesu. Jeżeli nie, to zarządca bufora sprowadza daną stronę z bazy danych do bufora (operacja fetch) i, podobnie jak poprzednio, zwraca adres strony w buforze do procesu. Jak wspomnieliśmy, wczytanie strony do bufora wymaga, najczęściej, usunięcia innej strony z bufora (mówimy o wymianie stron na żądanie). Bufor w trakcie normalnego działania pracuje w nasyceniu, co oznacza, ze wszystkie ramki bufora są zajęte przez strony. Najpopularniejszą strategią wymiany stron na żądanie jest strategia LRU (ang. least recently used) – usuwamy z bufora stronę najmniej używaną w ostatnim czasie. Idea strategii LRU bazuje na lokalności odwołań – zarządca bufora nie usuwa strony z bufora po zakończeniu aktualizacji tej strony w nadziei, że kolejny proces może zażądać dostępu do tej strony. Pozostawiając zaktualizowaną stronę w buforze minimalizujemy liczbę operacji we/wy. W konsekwencji często aktualizowane strony pozostają w buforze bardzo długo, aż: zostaną usunięte w wyniku zastosowania strategii LRU lub zostaną zapisane na dysku po zadanym czasie.

Bufor danych (2)

- Mówimy, że strona w buforze jest „brudna”, jeżeli została uaktualniona przez transakcję od czasu jest ostatniego zapisu na dysk
- „Brudne” strony mogą pozostawać w buforze jeszcze długo po tym, jak uaktualniające je transakcje zostały zaakceptowane

Problem: awaria. Jeżeli aktualizowane strony nie były zapisywane na dysk, to tracimy informacje o dokonanych aktualizacjach

- W momencie aktualizacji, system zapisuje ten fakt w postaci tzw. rekordu logu (ang. *log entry*) w buforze logu
- Okresowo, bufor logu jest zapisywany na dysk do pliku logu (ang. *log file*)

Wprowadźmy pojęcie tak zwanej „brudnej” strony. Mówimy, że strona w buforze jest „*brudna*”, jeżeli zastała uaktualniona przez transakcję od czasu jest ostatniego zapisu na dysk. Z tego co powiedzieliśmy poprzednio wynika, że „brudne” strony mogą pozostawać w buforze jeszcze długo po tym, jak uaktualniające je transakcje zostały zaakceptowane. Jest to ważny element mechanizmu zapewniającego odpowiednią efektywność działania systemu bazy danych. Jednakże rodzi to bardzo poważny problem: w przypadku wystąpienia awarii, jeżeli aktualizowane strony nie były zapisywane w międzyczasie na dysk, to tracimy informacje o dokonanych aktualizacjach. Jak rozwiązać ten problem? Okazuje się również, że rozwiązanie polegające na natychmiastowym zapisie uaktualnionych stron na dysk nie rozwiązuje poprawnie problemu utarty informacji. Rozwiązaniem jest mechanizm logu.

W momencie aktualizacji dowolnej strony, system zapisuje ten fakt w postaci tzw. *rekordu logu* (ang. log entry) w *buforze logu*. Okresowo, bufor logu jest zapisywany na dysk do *pliku logu* (ang. log file). Współdziałanie bufora danych, bazy danych, bufora logu i pliku logu pozwala na poprawną implementacje procedur odtwarzania spójnego stanu bazy danych po awarii

Format logu (1)

- Rozważmy następującą realizację transakcji:

r1 = T1:r(a, 50) T1:w(a, 20) T2:r(c, 100)
T2:w(c,50) T2:c T1:r(b, 50) T1:w(b, 80) T1:c.....

- Załóżmy, że w bazie danych zdefiniowano ograniczenie integralnościowe postaci: a + b = 100 i załóżmy, że zapis na dysk jest realizowany zgodnie ze strategią LRU
- Wystąpiła awaria – załóżmy, że aktualne wartości danych na dysku są następujące: a = 50, b = 80, c = 100
- Dane niespójne

Przejdźmy obecnie do formalnego zdefiniowania problemu odtwarzania bazy danych po awarii. Zacznijmy od fundamentalnego pytania: czy można podać taką strategię zarządzania buforem danych, aby dane w bazie danych były zawsze spójne? Innymi słowy, czy konieczny jest osobny mechanizm gwarantujący poprawne odtworzenie bazy danych po awarii? Odpowiedź na pierwsze pytanie brzmi – NIE, i, oczywiście, odpowiedź na pytanie drugie brzmi –TAK. Rozpocznijmy od prostego przykładu. Dana jest współbieżna realizacja zbioru dwóch transakcji T1 i T2 przedstawiona na slajdzie. Załóżmy dodatkowo, że w bazie danych zdefiniowano ograniczenie integralnościowe postaci: a + b = 100 i załóżmy, że zapis na dysk jest realizowany zgodnie ze strategią LRU. Przyjmijmy, że początkowo, stan bazy danych wynosił: a=50, b=50 i c=100. Strony a i c, uaktualnione przez transakcje T1 i T2, pozostały w buforze (nowe wartości a=20 i c=50), natomiast strona b, uaktualniona przez transakcję T1, została zapisana na dysk w wyniku zadziałania strategii LRU. Jeżeli po wykonaniu operacji T1:w(b, 80) wystąpi awaria w systemie, to wartości danych a, b i c będą następujące: a=50 (poprzednia wartość), b=80 (nowa wartość zapisana przez T1), c=100 (poprzednia wartość). Łatwo zauważyć, że stan bazy danych jest niepoprawny – dane są niespójne. Wniosek: zakładając, że zapis stron z bufora danych na dysk jest realizowany wyłącznie w oparciu o strategię LRU, nie możemy zapewnić poprawnego stanu bazy danych.

Format logu (2)

- Załóżmy, że strategia zapisu jest następująca: strona jest zapisywana na dysk natychmiast po jej aktualizacji w buforze
- Realizacja jak wyżej - awaria wystąpiła po wykonaniu T2:c
- Wartości danych na dysku są następujące: a = 20, b = 80, c = 50 – dane niespójne
- Wniosek: strategia LRU jak i strategia zapisu natychmiastowego aktualizacji na dysk może prowadzić do niespójności bazy danych

Przyjmijmy w takim razie inną strategię działania bufora danych. Załóżmy mianowicie, że strategia zapisu stron na dysk jest następująca: strona jest zapisywana na dysk natychmiast po jej aktualizacji w buforze. Być może to zapewni poprawność działania systemu? Odpowiedź jest negatywna. Rozważmy realizację przedstawioną na poprzednim slajdzie. Załóżmy, że awaria wystąpiła po wykonaniu operacji T2:c. Zakładając, że każda strona, po aktualizacji, jest natychmiast zapisywana na dysk, otrzymujemy następujący stan bazy danych: a = 20, b = 80, c = 50. Dane są niespójne. Wniosek: strategia LRU jak i strategia natychmiastowego zapisu aktualizacji na dysk może prowadzić do niespójności bazy danych.

Stan spójny bazy danych (1)

- Celem odtwarzania jest zapewnienie własności atomowości i trwałości transakcji
- Procedura odtwarzania bazy danych (ang. database recovery) po awarii, korzystając z informacji zawartej w rekordach logu i stanu bazy danych przechowywanego na dysku, przeprowadza bazę danych do stanu, który odzwierciedla wszystkie lub żadną operacje aktualizacji transakcji aktywnych w momencie wystąpienia awarii
- System transakcyjny, inicjowany ponownie po awarii, nie pamięta intencji logiki transakcji, które były aktywne w momencie awarii - stąd, system musi wycofać wszystkie zmiany w bazie danych wprowadzone przez aktywne transakcje

Celem odtwarzania jest zapewnienie własności atomowości i trwałości transakcji. Procedura odtwarzania bazy danych po awarii (ang. database recovery), korzystając z informacji zawartej w rekordach logu i stanu bazy danych przechowywanego na dysku, przeprowadza bazę danych do stanu, który odzwierciedla wszystkie lub żadną operacje aktualizacji transakcji aktywnych w momencie wystąpienia awarii. System transakcyjny, inicjowany ponownie po awarii, nie pamięta intencji logiki transakcji, które były aktywne w momencie awarii - stąd, system musi wycofać wszystkie zmiany w bazie danych wprowadzone przez aktywne transakcje.

Stan spójny bazy danych (2)

- Dla realizacji (awaria w momencie realizacji operacji T1:c):

  r1 = T1:r(a, 50) T1:w(a, 20) T2:r(c, 100)
  T2:w(c,50) T2:c T1:r(b, 50) T1:w(b, 80) T1:c.....

- transakcja T2 zaakceptowana; T1 wycofana. Stąd, stan spójny po awarii: a = 50, b = 50, c = 50
- **Rekordy logu** – rekordy logu są generowane i zapisywane do bufora logu dla każdej operacji aktualizacji obiektu bazy danych i operacji inicjacji transakcji

Reasumując, stan spójny po awarii to taki stan, który odzwierciedla wszystkie aktualizacje wykonane przez zaakceptowane transakcje; natomiast zmiany w bazie danych, wprowadzone do niej, przez transakcje aktywne w momencie wystąpienia awarii, muszą zostać wycofane.

Dla realizacji przedstawionej na slajdzie (awaria w momencie realizacji operacji T1:c), poprawny stan bazy danych po awarii powinien odzwierciedlać w bazie danych aktualizacje transakcji T2 (T2 zaakceptowana); oraz powinien wycofać z bazy danych zmiany wprowadzone, być może, przez transakcję (T1 aktywna w momencie awarii). Stąd, stan spójny po awarii powinien być następujący: a = 50, b = 50, c = 50.

Rekordy logu (1)

- Format rekordu pliku logu:
  - **(S, i):** rekord inicjacji transakcji Ti;
  - **(W, i, a, x, y):** rekord zapisu danej a przez transakcje Ti, poprzednia wartość danej x (before image), nowa wartość danej y (after image);
  - **(C, i):** rekord akceptacji transakcji Ti;
- Realizacja:

r1 = T1:r(a, 50) T1:w(a, 20) T2:r(c, 100)
T2:w(c,50) T2:c T1:r(b, 50) T1:w(b, 80) T1:c.....

Jak już wspomnieliśmy, mechanizm odtwarzania po awarii opiera się na współdziałaniu czterech elementów. Kluczowymi elementami systemu odtwarzania są bufor i plik logu. Rekordy logu są generowane i zapisywane do bufora logu dla każdej operacji aktualizacji obiektu bazy danych i operacji inicjacji transakcji.

W dalszej części wykładu przyjmiemy następującą notację zapisu formatu rekordów logu:

–(S, i): oznacza rekord inicjacji transakcji Ti;

–(W, i, a, x, y): oznacza rekord zapisu danej a przez transakcje Ti, poprzednia wartość danej x (before image), nowa wartość danej y (after image);

–(C, i): oznacza rekord akceptacji transakcji Ti;

Rekordy logu (2)

- Rekordy logu wygenerowane przez moduł odtwarzania dla przedstawionej realizacji:

Operacja	Rekord logu
1. T1:r(a, 50)	(S, 1) rekord logu początku transakcji T1, nie wpisuje się rekordów logu dla operacji odczytu, w tym przypadku jest to operacja początku transakcji
2. T1:w(a, 20)	(W, 1, a, 50, 20) – rekord logu dla operacji aktualizacji atrybutu a. Wartość a=50 before image, a=20- after image

Dla rozważanej realizacji ze slajdu 15 sekwencja rekordów logu generowana przez zarządcę bufora logu zostanie przedstawiona na kolejnych slajdach.

Operacja

1. T1:r(a, 50) – generowany rekord (S, 1) - rekord logu początku transakcji T1 (generalnie, system nie generuje rekordów logu dla operacji odczytu, w tym przypadku jest to operacja początku transakcji)

2. T1:w(a, 20) – generowany rekord (W, 1, a, 50, 20) – rekord logu dla operacji aktualizacji atrybutu a. Wartość a=50 before image, a=20 – after image

Rekordy logu (3)

3. T2:r(c, 100)	(S, 2) – rekord logu początku transakcji T2
4. T2:w(c, 50)	(W, 2, c, 100, 50) – rekord logu dla operacji aktualizacji atrybutu c. Wartość c=100 (before image), c=50 – after image
5. T2:c	(C, 2) – rekord akceptacji transakcji T2 – (**zapis bufora logu na dysk**)
6. T1:r(b, 50)	

Operacja

3. T2:r(c, 100) - generowany rekord (S, 2) - rekord logu początku transakcji T2

4. T2:w(c, 50) - generowany rekord (W, 2, c, 100, 50) – rekord logu dla operacji aktualizacji atrybutu c. Wartość c=100 (before image), c=50 – after image

5. T2:c - generowany rekord (C, 2) – rekord akceptacji transakcji T2 (następuje **zapis bufora logu na dysk**)

6. T1:r(b, 50) brak generacji rekordu logu (operacja odczytu)

Rekordy logu (4)

7. T1:w(b, 80)	(W, 1, b, 50, 80) – rekord logu dla operacji aktualizacji atrybutu b. Wartość b=50 – before image, b=80 – after image
8. T1:c	(C, 2) – rekord akceptacji transakcji T1 – (**zapis bufora logu na dysk**)

- Bufor logu jest zapisywany do pliku logu w dwóch sytuacjach:
  - w momencie akceptacji transakcji
  - w momencie przepełnienia bufora (*double-buffered disk write*)

Operacja

7. T1:w(b, 80) - generowany rekord (W, 1, b, 50, 80) – rekord logu dla operacji aktualizacji atrybutu b. Wartość b=50 (before image), b=80 – after image

8. T1:c - generowany rekord (C, 2) – rekord akceptacji transakcji T1 (następuje **zapis bufora logu na dysk**)

Zauważmy, że bufor logu jest zapisywany do pliku logu w momencie akceptacji transakcji. Oczywiście, bufor logu jest również zapisywany do pliku logu w momencie przepełnienia bufora. Stosowana jest architektura tzw. double-buffered disk write, tj. w momencie, gdy cześć bufora logu jest zapisywana na dysk do pliku logu, druga cześć bufora jest dostępna dla nowo generowanych rekordów logu.

Poprawność odtwarzania (1)

- Czy informacje zapisane w buforze logu są wystarczające do odtworzenia spójnego stanu bazy danych?
- Mogą wystąpić dwa problemy:
- może się okazać, że zapisaliśmy na dysk strony uaktualnione przez nie zaakceptowane transakcje (**UNDO**)
- może się okazać, że nie zapisaliśmy na dysk stron uaktualnionych przez zaakceptowane transakcje (**REDO**)

Czy informacje zapisane w buforze logu są wystarczające do odtworzenia spójnego stanu bazy danych po awarii? Aby odpowiedzieć na to pytanie, należy rozważyć dwa przypadki, które mogą stwarzać potencjalnie pewne problemy.

–może się okazać, że zapisaliśmy na dysk strony uaktualnione przez nie zaakceptowane transakcje – musimy wówczas wycofać wszystkie zmiany wprowadzone przez nie zaakceptowane transakcje,

–może się okazać, że nie zapisaliśmy na dysk stron uaktualnionych przez zaakceptowane transakcje – musimy zagwarantować, że zmiany wprowadzone przez zaakceptowane transakcje znajdą się w bazie danych.

Poprawność odtwarzania (2)

- Realizacja (scenariusz nr 1)

r1 = T1:r(a, 50) T1:w(a, 20) T2:r(c, 100)
T2:w(c,50) T2:c T1:r(b, 50) T1:w(b, 80) T1:c.....

- Załóżmy, że wystąpiła awaria systemu po wykonaniu operacji T1:w(b, 80) ((W, 1, b, 50, 80) ostatni zapis do bufora logu)
- Bufor logu został zapisany na dysk w momencie akceptacji transakcji T2
- Transakcja T2 została zaakceptowana; transakcja T1 nie.
- Po ponownej inicjacji systemu po awarii, operacje aktualizacji wykonane przez transakcję T1 muszą zostać wycofane! (brak danych umożliwiających wycofanie)

Aby zilustrować pierwszy z wymienionych problemów rozważmy ponownie przykładową realizację ze slajdu 15. Załóżmy, że awaria systemu wystąpiła po wykonaniu operacji T1:w(b, 80) ((W, 1, b, 50, 80) - ostatni zapis do bufora logu). Przypomnijmy, że bufor logu został zapisany na dysk w momencie akceptacji transakcji T2. W momencie wystąpienia awarii transakcja T2 była zaakceptowana; natomiast transakcja T1 była aktywna. Należy pamiętać, że awaria systemu niszczy nie tylko dane znajdujące się w buforze danych, ale również bufor logu. Jedyna informacja, które może być wykorzystana do odtworzenia stanu spójnego bazy danych znajduje się na dysku zewnętrznym w pliku logu. Po ponownej inicjacji systemu po awarii, operacje aktualizacji wykonane przez transakcję T1 muszą zostać wycofane! Jeżeli na skutek działania strategii LRU strona b, zmodyfikowana przez transakcję T1, została zapisana na dysk, brak jest danych umożliwiających wycofanie tej aktualizacji. Rekord logu (W, 1, b, 50, 80) nie został zapisany do pliku logu. Do tego problemu wrócimy w późniejszej części wykładu.

Poprawność odtwarzania (3)

- Procedura odtwarzania po awarii jest wykonywana w dwóch fazach: **ROLLBACK** i **ROLL FORWARD**
- W fazie **ROLLBACK**, rekordy pliku logu są odczytywane w odwrotnej kolejności (od końca). W fazie tej procedura odtwarzania wykonuje operacje UNDO wszystkich operacji aktualizacji transakcji, które nie zostały zaakceptowane
- W fazie **ROLL FORWARD**, rekordy pliku logu są odczytywane w oryginalnej kolejności (od początku). W fazie tej procedura odtwarzania wykonuje operacje REDO wszystkich operacji aktualizacji transakcji, które zostały zaakceptowane

Procedura odtwarzania bazy danych po awarii jest wykonywana w dwóch fazach: fazie ROLLBACK i fazie ROLL FORWARD. W fazie ROLLBACK, rekordy pliku logu są odczytywane w odwrotnej kolejności (od końca) pliku logu. W fazie tej procedura odtwarzania wykonuje operacje UNDO wszystkich operacji aktualizacji transakcji, które nie zostały zaakceptowane. W fazie ROLL FORWARD, rekordy pliku logu są odczytywane w oryginalnej kolejności (od początku) pliku logu. W fazie tej procedura odtwarzania wykonuje operacje REDO wszystkich operacji aktualizacji transakcji, które zostały zaakceptowane.

Faza Rollback

• Rekord logu	Rollback
5. (C, 2)	wstaw T2 do listy transakcji zaakceptowanych
4. (W,2,c,100,50)	T2 na liście transakcji zaakceptowanych – nie rób nic
3. (S, 2)	T2 na liście transakcji zaakceptowanych – nie rób nic
2. (W,1,a,50,20)	T1 – aktywna, wykonaj UNDO operacji aktualizacji – dana a przyjmuje wartość 50
1. (S, 1)	Transakcja T1 pasywna. Zakończ fazę ROLLBACK

Przedstawimy obecnie, nieco dokładniej, działanie procedury Rollback w odniesieniu do przykładu ze slajdu 21. Następujące rekordy logu zostały zapisane do pliku logu w momencie akceptacji transakcji T2 (od początku pliku logu): .(S, 1), .(W,1,a,50,20), .(S, 2), .(W,2,c,100,50), .(C, 2). Procedura Rollback odczytuje rekordy pliku logu w odwrotnej kolejności. Omówimy kolejno operacje wykonywane w fazie Rollback:

5. (C, 2) wstaw T2 do listy transakcji zaakceptowanych

4. (W,2,c,100,50) T2 na liście transakcji zaakceptowanych – nie rób nic

3. (S, 2) T2 na liście transakcji zaakceptowanych – nie rób nic

2. (W,1,a,50,20) T1 – aktywna, wykonaj UNDO operacji aktualizacji – dana a przyjmuje wartość 50 (before image)

1.(S, 1) Transakcja T1 pasywna. Zakończ fazę ROLLBACK

W fazie Rollback została wykonana operacja UNDO dla transakcji T1, która była aktywna w momencie wystąpienia awarii. Przejdziemy do przedstawienia fazy Roll Forward.

Faza Roll Forward

• Rekord logu	Rollback
1. (S, 1)	brak akcji (T1 aktywna)
2. (w,1,a,50,20)	brak akcji (T1 aktywna)
3. (S, 2)	brak akcji
4. (w,2,c,100,50)	Transakcja T2 na liście transakcji zaakceptowanych. Wykonaj REDO operacji aktualizacji – dana c przyjmuje wartość 50
5. (C,2)	brak akcji Zakończono przeglądanie pliku logu – zakończ procedurę odtwarzania

Podobnie jak poprzednio, działanie procedury ROLL FORWARD przedstawimy w odniesieniu do przykładu ze slajdu 21. W fazie ROLL FORWARD, rekordy pliku logu są odczytywane w oryginalnej kolejności (od początku) pliku logu. W fazie tej procedura odtwarzania wykonuje operacje REDO wszystkich operacji aktualizacji transakcji, które zostały zaakceptowane.

1. (S, 1) - dla pierwszego rekordu logu transakcja T1 jest inicjowana i system nie podejmuje żadnej akcji;

2. (w,1,a,50,20) - dla drugiego rekordu logu transakcja T1 jest aktywna i system nie podejmuje żadnej akcji;

3. (S, 2) - podobnie jest dla trzeciego rekordu logu, który opisuje inicjowanie transakcji T2;

4. (w,2,c,100,50) - dla czwartego rekordu logu, transakcja T2 znajduje się na liście transakcji zaakceptowanych; system wykonuje operację REDO operacji aktualizacji – dana c przyjmuje wartość 50 (after image);

5. (C,2) - piąty rekord logu oznacza zatwierdzenie transakcji drugiej, a system nie podejmuje żadnej akcji. W tym momencie kończy się przeglądanie pliku logu, co kończy procedurę odtwarzania.

W fazie Roll Forward została wykonana operacja REDO dla transakcji T2, która została zaakceptowana wcześniej, przed wystąpieniem awarii.

Poprawność odtwarzania (4)

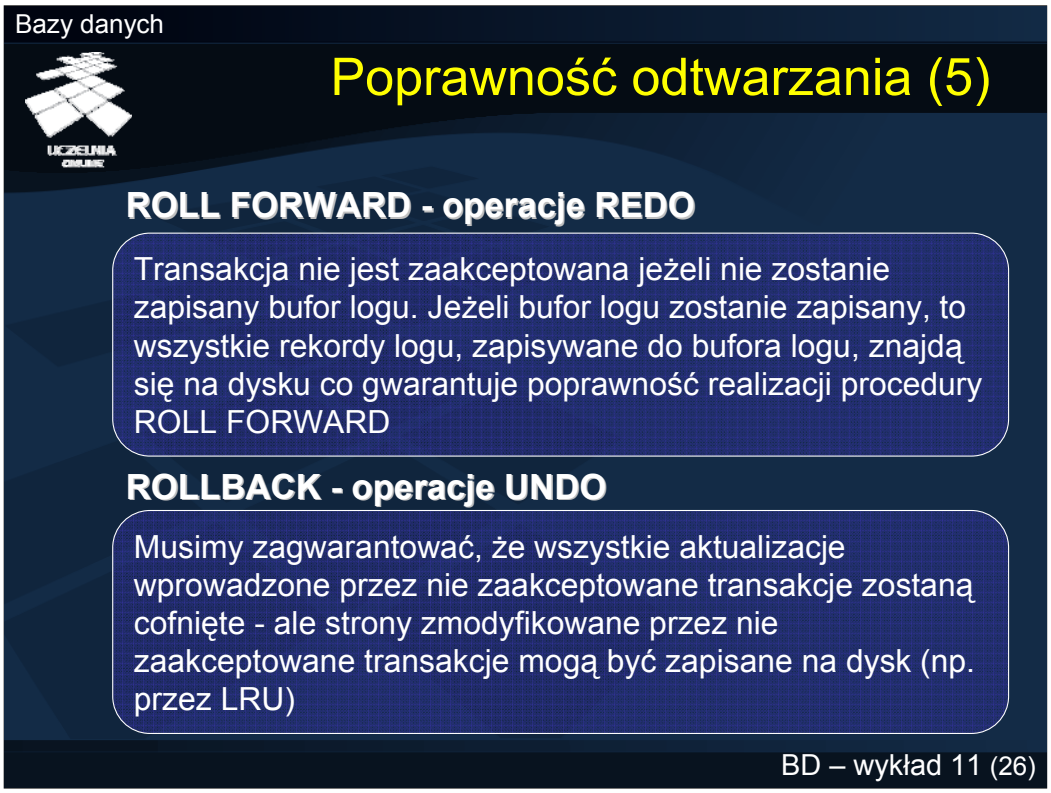
- W jaki sposób możemy zapewnić, że wszystkie rekordy logu konieczne do zapewnienia poprawności procedury odtwarzania zostaną zapisane na dysku?
- Założenia dodatkowe:
  - zapis na dysk jest realizowany w sposób atomowy - „read after write”
  - akceptacja transakcji + zapis bufora logu - akcja atomowa
- Poprawność procedur ROLLBACK i ROLL FORWARD

Wróćmy obecnie do dyskusji nad poprawnością zaproponowanego mechanizmu odtwarzania spójności bazy danych po awarii. W jaki sposób możemy zapewnić, że wszystkie rekordy logu konieczne do zapewnienia poprawności procedury odtwarzania zostaną zapisane na dysku? System zarządzania bazą danych musi zapewnić dodatkowe mechanizmy:

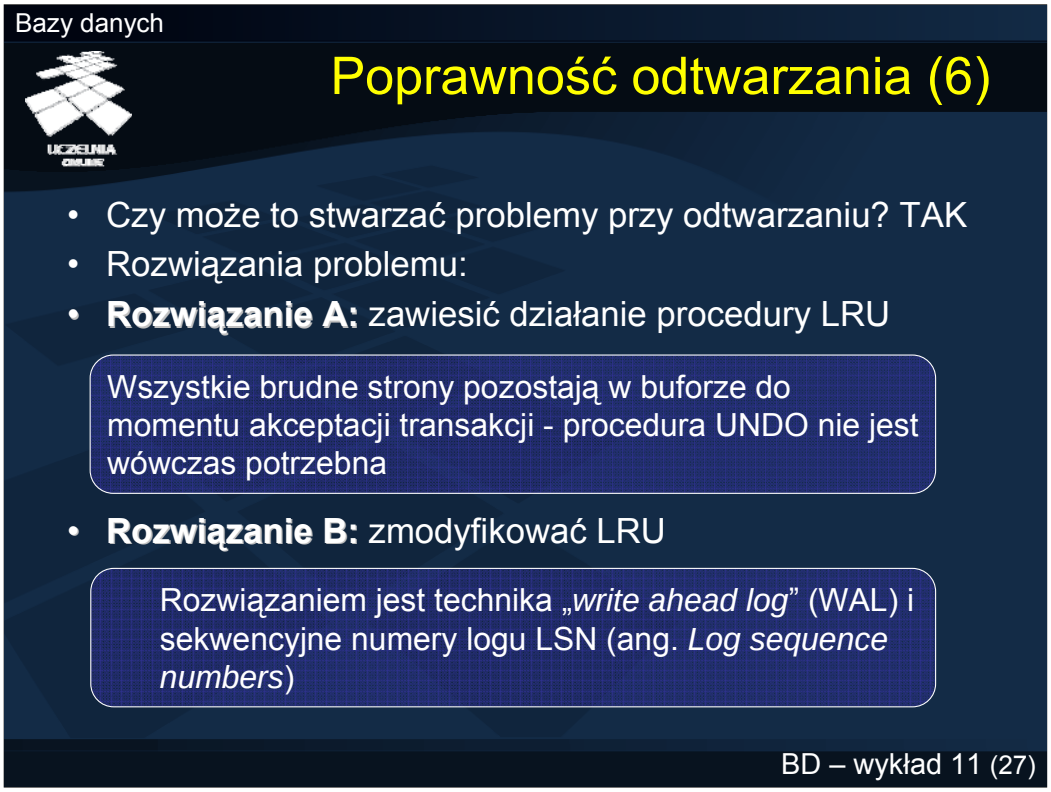
–zapis na dysk jest realizowany w sposób atomowy (np. mechanizm „read after write”)

–akceptacja transakcji + zapis bufora logu = akcja atomowa, tj. jeżeli z jakiś względów nie udało się zapewnić zapisu bufora logu do pliku logu, to oznacza to brak akceptacji transakcji.

Wróćmy do pytania, czy mechanizm odtwarzania oparty o procedury Rollback i ROLL FORWARD, uzupełniony o dwa wymienione wyżej mechanizmy, gwarantuje poprawność odtwarzania?



Rozpocznijmy od poprawności procedury ROLL FORWARD. Czy procedura ta pozwala zagwarantować, że wszystkie zmiany wprowadzone do bazy danych przez zaakceptowane transakcje znajdą się na dysku? Odpowiedź jest twierdząca. Transakcja nie jest zaakceptowana, jeżeli nie zostanie zapisany bufor logu. Jeżeli bufor logu zostanie zapisany, to wszystkie rekordy logu, zapisywane do bufora logu, znajdą się na dysku. To gwarantuje poprawność realizacji procedury ROLL FORWARD, tj. gwarantuje, że wszystkie zmiany wprowadzone do bazy danych przez zaakceptowane transakcje znajdą się na dysku (mechanizm REDO w fazie ROLL FORWARD). Drugie pytanie brzmi: czy procedura ROLLBACK gwarantuje, że wszystkie aktualizacje wprowadzone przez nie zaakceptowane transakcje zostaną cofnięte ? Niestety, odpowiedź jest negatywna - strony zmodyfikowane przez nie zaakceptowane transakcje mogą być zapisane na dysk (np. przez LRU), natomiast tych zmian nie można wycofać w oparciu o rekordy logu zapisane w pliku logu.



Możliwe są dwa rozwiązania tego problemu.

Rozwiązanie A: zawiesić działanie procedury LRU, to znaczy, przyjąć założenie, że wszystkie brudne strony pozostają w buforze do momentu akceptacji transakcji - procedura UNDO nie jest wówczas potrzebna.

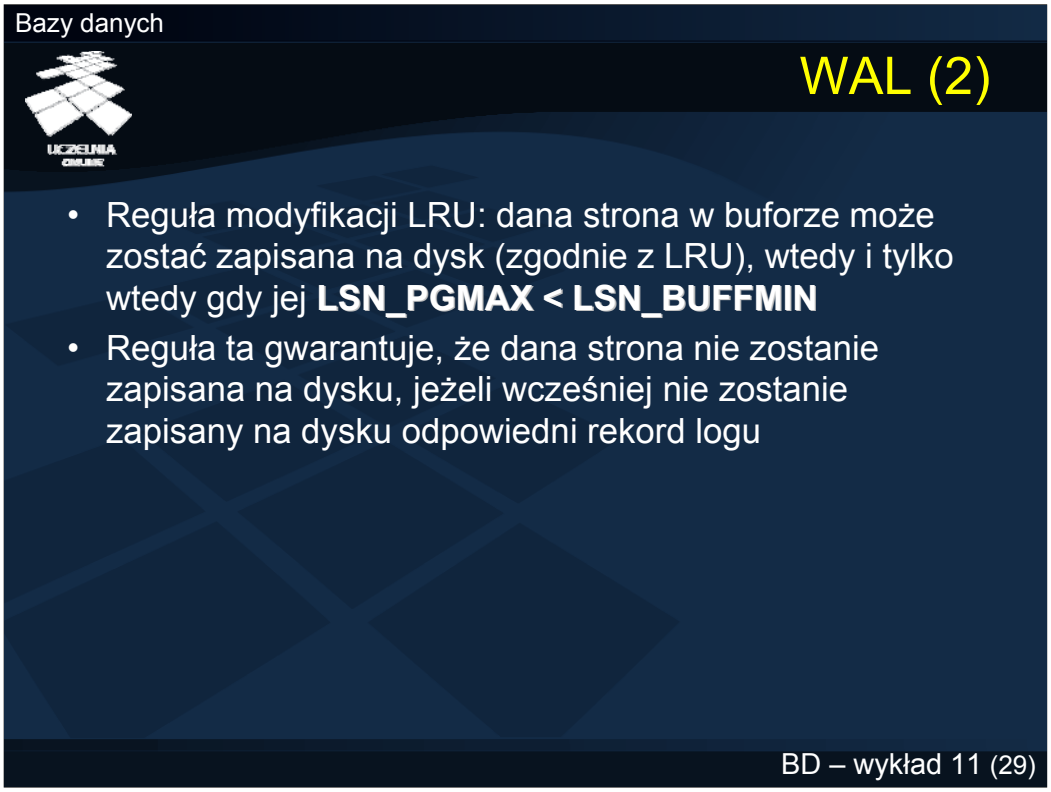
Rozwiązanie B: zmodyfikować strategię LRU.

Rozwiązaniem jest technika nazywana „write ahead log" (WAL).

WAL (1)

- System bazy danych utrzymuje licznik, który generuje rosnącą sekwencję LSN
- **LSN** jest liczbą całkowitą, przypisywaną do każdego rekordu logu zapisywanego do bufora logu
- **LSN_BUFFMIN** - SBD przechowuje również najmniejszą wartość LSN strony, od czasu ostatniego zapisu bufora na dysk
- **LSN_PGMAX** - dla każdej strony w buforze danych, system pamięta wartość ostatniej LSN operacji, która aktualizowała daną na tej stronie

System bazy danych utrzymuje specjalny licznik, który generuje rosnącą sekwencję tzw. log sequence number (LSN). LSN jest liczbą całkowitą, przypisywaną do każdego rekordu logu zapisywanego do bufora logu. SBD przechowuje również najmniejszą wartość LSN strony, od czasu ostatniego zapisu bufora na dysk. Oznaczamy ją LSN_BUFFMIN. Jest to unikalna liczba dla całego systemu. Ponadto, dla każdej strony w buforze danych, system pamięta wartość ostatniej operacji LSN, która aktualizowała daną na tej stronie. Oznaczamy ją przez LSN_PGMAX.

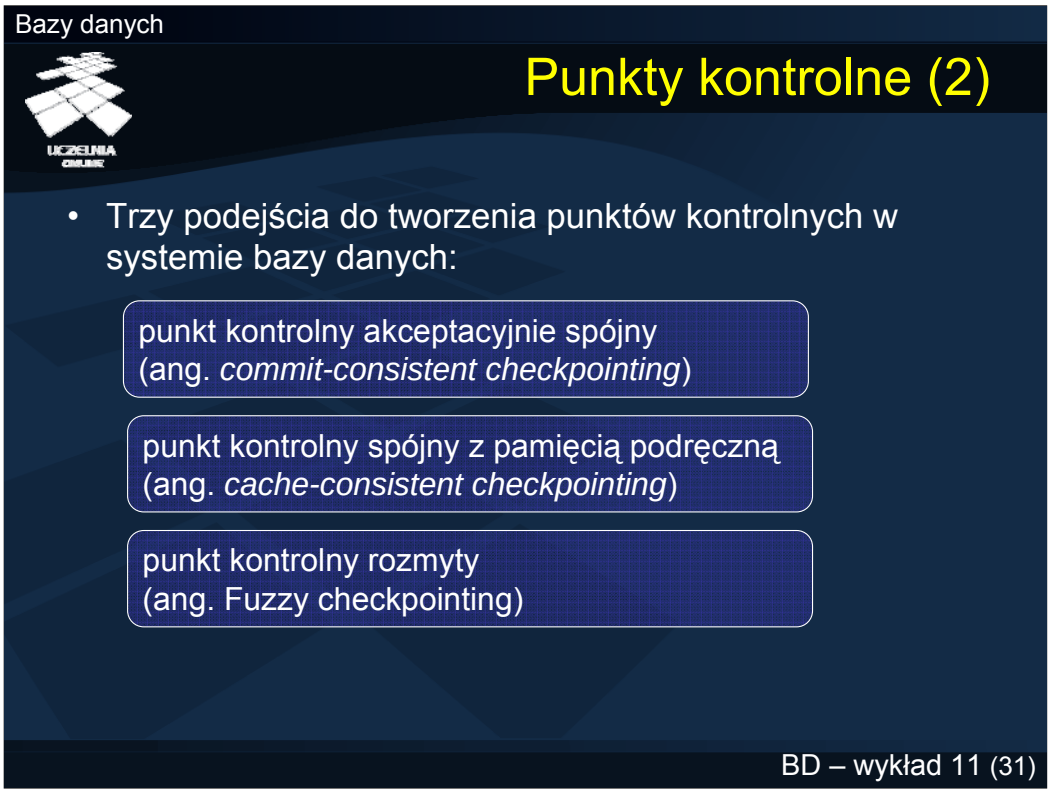


Reguła modyfikacji LRU jest następująca. Dana strona w buforze może zostać zapisana na dysk (zgodnie z LRU), wtedy i tylko wtedy gdy jej LSN_PGMAX < LSN_BUFFMIN. Reguła ta gwarantuje, że dana strona nie zostanie zapisana na dysku, jeżeli wcześniej nie zostanie zapisany na dysku odpowiedni rekord logu

Punkty kontrolne (1)

- Do jakiego stanu należy się cofnąć wykonując procedurę ROLLBACK?
  - do momentu inicjacji (startup) systemu
  - do stanu określonego przez użytkownika (administratora)
- Problem:
  - procedura ROLLBACK - większość transakcji to krótkie transakcje aktualizacji - stosunkowo efektywna
  - procedura ROLL FORWARD - konieczność przetworzenia całego pliku logu - niska efektywność

Na zakończenie wykładu, krótko wspomnimy o tak zwanych punktach kontrolnych. Do jakiego stanu należy się cofnąć w pliku logu wykonując procedurę ROLLBACK? do momentu inicjacji (startup) systemu, czy do stanu określonego przez użytkownika (administratora)? Problemem może być rozmiar pliku logu. Jak wygląda efektywność procedury ROLLBACK i ROLL FORWARD? Procedura ROLLBACK - większość transakcji to krótkie transakcje aktualizacji - jest stosunkowo efektywna. Procedura ROLL FORWARD - konieczność przetworzenia całego pliku logu - ma niską efektywność.



Zauważmy, że w trakcie wystąpienia awarii, tylko cześć transakcji była aktywna. Stąd, wycofanie zmian wprowadzonych przez te transakcje może być stosunkowo efektywnie zaimplementowane. W przypadku procedury ROLL FORWARD – zachodzi konieczność przetworzenia całego pliku logu! Musimy, de facto, wykonać wszystkie zaakceptowane transakcje raz jeszcze – operacja po operacji! Może to być bardzo kosztowne. Rozwiązaniem są tak zwane punkty kontrolne. Punkt kontrolny (ang. checkpoint) - punkt na osi czasu, od którego i do którego są realizowane procedury ROLLBACK i ROLL FORWARD. Punkty kontrolne są tworzone w trakcie działania systemu bazy danych i ograniczają konieczność przetwarzania całego pliku logu w procedurze odtwarzania. Po pierwsze, plik logu narasta bardzo szybko i możemy nie posiadać możliwości, aby przechowywać cały plik logu. Po drugie, przetwarzanie całego pliku logu byłoby niezmiernie kosztowne czasowo.

Trzy ogólne podejścia do tworzenia punktów kontrolnych w systemie bazy danych to:

- punkt kontrolny akceptacyjnie spójny (ang. commit-consistent checkpointing),
- punkt kontrolny spójny z pamięcią podręczną (ang. cache-consistent checkpointing),
- punkt kontrolny rozmyty (ang. fuzzy checkpointing).

Opis wymienionych punktów kontrolnych i sposobów ich tworzenia można znaleźć w podanej literaturze.