Bazy Danych

Andrzej M. Borzyszkowski
Instytut Informatyki

Uniwersytetu Gdańskiego

materiały dostępne elektronicznie
http://inf.ug.edu.pl/~amb

Transakcje

© Andrzej M. Borzyszkowski
Bazy Danych

Współbieżność

- Przykład: przelewanie pieniędzy z konta A na konto B, początkowe stany obu kont równe 100, więc suma 200:

czas	użytkownik 1	użytkownik 2
0 min	czyta konto A, wynik 100	
1 min		odejmuje 50 z konta A
2 min		dodaje 50 do konta B
3 min	czyta konto B, wynik 150	

  - a więc użytkownik 1 sądzi, że na obu kontach jest razem 250

© Andrzej M. Borzyszkowski
Bazy Danych

Współbieżność, c.d.

- Przykład: każdy z użytkowników dodaje swój wkład do konta, stan początkowy 50:

czas	użytkownik 1	użytkownik 2
0 min	czyta stan konta, wynik 50	
1 min		czyta stan konta, wynik 50
2 min	nowa wartość konta 110	
3 min		nowa wartość konta 125

  - każdy z użytkowników sądzi, że nowa wartość konta jest powiększona o jego wpłatę, odp. 60 i 75 (i umożliwi w przyszłości wypłatę)

© Andrzej M. Borzyszkowski
Bazy Danych

Współbieżność, III

- Przykład: przelew raz jeszcze

czas	użytkownik 1	SYSTEM
1 min	odejmuje 50 z konta A	
2 min		AWARIA
1 godz	stan konta A pomniejszony o 50, stan konta B bez zmian	

  - tak więc suma obu kont będzie mniejsza niż przed awarią
  - byłaby większa, gdyby przelew zacząć od wpłaty

Współbieżność – wyzwania

- Trzy problemy
  - niespójna analiza
  - utracona modyfikacja
  - niezatwierdzona wartość

Transakcja – niepodzielna jednostka działań

  - albo wykonają się wszystkie operacja w transakcji, albo żadna
  - tzn. nowe wartości muszą być zatwierdzone
  - transakcja zajmuje zero czasu !

Współbieżność – niespójna analiza

- Przykład: przelewanie pieniędzy z konta A na konto B, początkowe stany obu kont równe 100, więc suma 200:

czas	użytkownik 1	użytkownik 2
0 min		odejmuje 50 z konta A
1 min		dodaje 50 do konta B
2 min	czyta konto A, wynik 50	
3 min	czyta konto B, wynik 150	

  - albo na odwrót, najpierw działa użytkownik 1, potem 2
  - w każdym przypadku użytkownik 1 uzyska sumę 200

Współbieżność – utracona modyfikacja

- Przykład: każdy z użytkowników dodaje swój wkład do konta, stan początkowy 50:

czas	użytkownik 1	użytkownik 2
0 min	czyta stan konta, wynik 50	
1 min	nowa wartość konta 110	
2 min		czyta stan konta, wynik 110
3 min		nowa wartość konta 185

  - albo na odwrót, najpierw działa użytkownik 2, gdy skończy działa użytkownik 1
  - zawsze późniejszy czyta dane aktualne, zmienione przez wcześniejszego

Współbieżność -- awaria

- Przykład: przelew raz jeszcze

czas	użytkownik 1	SYSTEM
1 min	odejmuje 50 z konta A	
2 min	dodaje 50 do konta B	
3 min		AWARIA
1 godz	stan konta A pomniejszony o 50, stan konta B powiększony o 50, suma prawidłowa	

  - gdyby awaria zdarzyła się przed zmianą konta B, uznamy, że zdarzyła się nawet przed zmianą konta A, tzn. konto A również pozostanie niezmienione w stanie początkowym

Operacje na danych

- Rodzaj **operacji**
  - odczyt – read(X)
  - zapis – write(X)
- Wielkość operacji
  - atomowa dana (komórka w tabeli)
  - wiersz tabeli (pojedyncza encja)
  - cała tabela
  - wielkość wyznaczona przez implementację (blok w systemie plików itp.)
- Również operacje zatwierdzenia (*commit*) i wycofania (*rollback*, *abort*)

Wycofanie transakcji – przyczyny

- Transakcja musi wykonać wszystkie operacje
  - a jeśli to niemożliwe, to musi wycofać już dokonane (*rollback, undo*)
- Przyczyny wycofania
  - przerwanie wykonania – błędy pamięci, przesyłania danych, spowodowane poza SZBD
  - błędy operacji z transakcji, jawna operacja wycofania
  - konieczność spowodowana współbieżnością (o tym będzie wykład)
  - również upływ czasu powoduje wycofanie niezatwierdzonej transakcji
  - awarie trwałych danych (pamięć dyskowa, ...)

Wycofanie transakcji – narzędzia

- Narzędzia wycofania
  - dziennik – zapis każdego ruchu (start(T), read(T,X), write(T,X,old,new), commit(T), abort(T))
  - mogą być prostsze dzienniki
  - w razie wycofania transakcji dziennik posłuży do odtworzenia poprzedniego stanu
  - po zatwierdzeniu transakcji i utrwaleniu jej wyników fragmenty dziennika są usuwane
- Wycofanie transakcji jest co najmniej tak czasochłonne jak sama transakcja, a raczej dużo bardziej

Wycofanie transakcji – wariacje

- Niektóre systemy przewidują ustawienie w transakcji punktów kontrolnych (savepoints)
  - wycofanie następuje do ostatniego takiego punktu
  - PostgreSQL od wersji 8.* posiada też to narzędzie
  - w zasadzie jest to sprzeczne z ideą atomowości transakcji

Wycofanie transakcji – wariacje 2

- Transakcja może obejmować kilka systemów (long transaction)
  - zatwierdzanie dwufazowe: każdy z systemów posiada dziennik pozwalający odtworzyć stan poprzedni i nowy
  - jeśli wszystkie systemy zakończyły pomyślnie ten etap, koordynator zaleca przyjęcie nowego stanu, wpp. odtworzenie poprzedniego stanu przez wszystkie systemy
  - idea transakcji zależy mocno od pewności, że koordynator będzie w stanie skutecznie porozumieć się, ze wszystkimi uczestnikami

Transakcja – własności ACID

- Cztery własności charakteryzujące transakcje
  - A – niepodzielność (atomic) – transakcji nie da się podzielić na podoperacje
  - C – spójność (consistency) – po zatwierdzeniu transakcji (i po wycofaniu transakcji) baza danych jest w stanie spójnym, tak jak była przed rozpoczęciem transakcji
  - I – odizolowanie (isolated) – transakcja przebiega tak, jak by w danym momencie była jedyną transakcją w systemie
  - D – trwałość (durable) – zatwierdzenie transakcji oznacza, że jej wyniki są trwale widoczne w bazie

Przebiegi transakcji

- Przebieg (wykonanie, historia) transakcji T1,...,Tn
  - ciąg operacji z T1, ..., Tn, t.że operacje z każdej transakcji występują w przebiegu w tej samej kolejności co w danej transakcji
- Przykład (T1 i T2, czytają i zapisują X i Y, a=rollback)
  - r1(X);r2(X);w1(X);r1(Y);w2(X);w1(Y);
  - r1(X);w1(X);r2(X);w2(X);r1(Y);a1;
- Operacje w konflikcie
  - jeśli należą do różnych transakcji, oraz
  - dotyczą tego samego obiektu, oraz
  - co najmniej jedna z operacji zapisuje ten obiekt
- Przebieg nie musi koniecznie być ciągiem
  - musi być ustalona kolejność operacji w konflikcie
  - oraz kolejność operacji w każdej transakcji

Przebiegi odtwarzalne

- Transakcja T2 czyta z transakcji T1 w danym przebiegu, jeśli
  - istnieje obiekt X, t.ż. r2(X) jest później niż w1(X)
- Przebieg jest odtwarzalny, jeśli
  - każda transakcja T2 czytająca z transakcji T1 jest zatwierdzona dopiero po zatwierdzeniu T1
  - kontrprzykład: r1(X);w1(X);r2(X);r1(Y);w2(X);c2;a1; – T2 odczytała X z T1, ale T1 została wycofana
- Problemy
  - utracona modyfikacja nadal możliwa
  - wycofania mogą powodować kolejne wycofania (kaskada)
  - r1(X);w1(X);r2(X);r1(Y);w2(X);a1;a2; – po wycofaniu T1 okazało się, że przeczytana wartość X jest nieaktualna

Przebiegi odtwarzalne, c.d.

- Przebieg jest bez kaskad, jeśli
  - żadna transakcja nie czyta obiektów zapisanych przez niezatwierdzone inne transakcje
- Przebieg jest ścisły, jeśli
  - żadna transakcja nie czyta ani nie zapisuje obiektów zapisanych przez niezatwierdzone inne transakcje
- Dla przebiegu ścisłego odtwarzanie jest łatwe, wystarczy przywrócić poprzednią wartość obiektów
  - dla innych przebiegów prosty pomysł nie wystarcza
  - przykład: X=1: w1(X,5);w2(X,7);a1;a2;
  - Najpierw old=1, new=5, potem old=5, new=7, po wycofaniu obu transakcji nie wiemy już, że X=1

Szeregowalność

- Przebieg jest sekwencyjny, jeśli transakcje wykonywane są po kolei, bez przeplatania operacji
  - przebieg jest szeregowalny, jeśli w pewnym sensie jest równoważny sekwencyjnemu
- Równoważność przebiegów
  - dają ten sam rezultat – ale jak to sprawdzić?
  - operacje w konflikcie wykonywane są w tej samej kolejności
  - jest jeszcze inna definicja, mniej ograniczająca
- Przykład: r1(X);w1(X);r1(Y);w1(Y);r2(X);w2(X); – T1 potem T2
  - r1(X);w1(X);r2(X);w2(X);r1(Y);w1(Y); – równoważny, T1<T2
- r1(X);r2(X);w1(X);w2(X);r1(Y);w1(Y); – nie jest szeregowalny, bo T2<T1<T2
- Przebieg jest szeregowalny, jeśli *nie ma cyklu* w grafie kolejności

SQL/ PostgreSQL

- BEGIN (można użyć BEGIN WORK) w SQL nie występuje, ponieważ każde wyrażenie SQL rozpoczyna transakcję
  - PostgreSQL transakcja rozciąga się na jedną instrukcję, jeśli ma być dłuższa, trzeba użyć BEGIN
- COMMIT (można użyć COMMIT WORK) zatwierdzenie – kończy transakcję pozytywnie, wszystkie dane od tego momentu należy uważać za zatwierdzone, w szczególności dostępne dla innych transakcji
- ROLLBACK (również w wersji ROLLBACK WORK) wycofanie – kończy transakcję niepowodzeniem, dane tymczasowe są przywrócone do poprzedniego stanu

Transakcje w PostgreSQL

- PostgreSQL działa domyślnie w trybie chained (niejawnych transakcji), instrukcja jest całą transakcją chyba, że jest częścią bloku BEGIN ROLLBACK/COMMIT
  - np. SQL server Microsoftu wymaga podania SET IMPLICIT_TRANSACTIONS
  - standard SQL wymaga jawnego zakończenia transakcji
- Nie wolno zagnieżdżać transakcji
  - tzn. BEGIN musi mieć do pary COMMIT albo ROLLBACK nim nastąpi następny BEGIN
- Transakcje powinny być w miarę krótkie
  - w szczególności należy pilnować, by częścią transakcji nie był dialog z użytkownikiem – najpierw dane, potem transakcja

Poziomy izolacji w/g ANSI/ISO

- Najwyższym poziomem jest założenie, że transakcja jest jedyną wykonywaną w danym momencie, tzn. wiele transakcji musi się uszeregować w kolejności (szeregowalność)
- Może to być zbyt mocne założenie, zbyt ograniczające wydajność bazy danych
- Standard ANSI/ISO wprowadza cztery poziomy izolacji
  - READ UNCOMMITTED
  - READ COMMITTED
  - REPEATABLE READ
  - SERIALIZABLE
- PostgreSQL domyślnie przyjmuje drugi poziom, można ustawić czwarty

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE

Poziomy izolacji: odczyt na brudno

- Odczyt na brudno (*dirty read*): odczyt danych jeszcze nie zatwierdzonych przez transakcję piszącą
  - transakcja być może będzie wycofana (ROLLBACK), należy przyjąć, że dane te nigdy nie istniały
- Poziom ANSI/ISO: READ UNCOMMITED
- PostgreSQL: nie dopuszcza do odczytu na brudno
  - *gdyby dopuszczał*: np. zmiana wartości konta z 100 na 200

czas	transakcja 1	transakcja 2	
0 min	SET konto=200		
1 min		SELECT konto...	odczyt 200
2 min	ROLLBACK		
3 min		SELECT konto...	odczyt 100

Poziomy izolacji: odczyt niepowtarzalny

- Odczyt niepowtarzalny (nonrepeatable read): odczyt danych nie dający się powtórzyć w ramach jednej transakcji
  - tzn. pozwolenie, by inna transakcja zmieniła odczytane dane
- Poziom ANSI/ISO: READ COMMITED
- PostgreSQL: domyślnie *dopuszcza* do odczytu niepowtarzalnego
  - np. wpłata na konto przez każdego z użytkowników
  - czyli możliwa jest niespójna analiza
  - oraz utracona modyfikacja

Odczyt niepowtarzalny, c.d.

- Np. wpłata na konto przez każdego z użytkowników

czas	użytkownik 1	użytkownik 2
0 min	BEGIN WORK	BEGIN WORK
1 min	czyta stan konta, wynik 50	
2 min		czyta stan konta, wynik 50
3 min	pisze wartość konta 110	
4 min		nie może zmienić stanu konta
5 min	COMMIT WORK	
6 min	(gdyby jeszcze raz czytał, byłaby wartość 110)	
		pisze wartość konta 125
7 min		COMMIT WORK

Poziomy izolacji: odczyt widmo

- Odczyt widmo (phantom): odczyt danych nie istniejących wcześniej w danej transakcji
  - tzn. pozwolenie, by inna transakcja wstawiła wiersz do przeczytanej tabeli
- Poziom ANSI/ISO: REPEATABLE READ
- PostgreSQL: domyślnie *dopuszcza* do odczytu widm

Czas	transakcja 1	transakcja 2
0 min	BEGIN	BEGIN
1 min	UPDATE towar SET cena=1	
2 min		INSERT INTO towar VALUES ...
3 min	SELECT cena FROM towar	
4 min	COMMIT	COMMIT

  - wstawiana wartość nie podległa globalnej zmianie w transakcji 1

Poziomy izolacji

Poziom izolacji	odczyt brudny	niepowt.	widmo
UNCOMMITTED	możliwy	możliwy	możliwy
COMMITED	niedopuszcz.	możliwy	możliwy
REPEATABLE	niedopuszcz.	niedopuszcz.	możliwy
SERIALIZABLE niedopusz	niedopuszcz.	niedopuszcz.	

PostgreSQL domyślnie przyjmuje drugi poziom, można ustawić czwarty

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE