Bazy danych

Andrzej Łachwa, UJ, 2015
andrzej.lachwa@uj.edu.pl

10/14

Przechowywanie danych

Wykorzystanie systemu plików, dostępu do plików za pośrednictwem systemu operacyjnego i proste rozwiązanie polegające na przechowywaniu każdej tabeli w jednym pliku, informacji o strukturze tabeli w innym pliku, a wyników operacji na tabelach w kolejnym pliku - nie nadaje się do stosowania w bazach danych!

Oto kilka powodów:
– modyfikacja jednego wiersza może spowodować konieczność przesunięcia wszystkich wierszy w pliku,
– przeszukiwanie może być bardzo kosztowne, bo np. warunek zapytania będzie sprawdzany dla każdego wiersza,
– działanie takie jak złączenie dwóch tabel będzie realizowane jako utworzenie nowej tabeli zawierającej złączone wiersze,
– wszystkie dane będą cały czas przechowywane w plikach na urządzeniu pamięci zewnętrznej,
– użytkownicy nie będą mogli jednocześnie modyfikować tej samej tabeli (tego samego pliku),
– w wyniku awarii urządzenia można utracić dane bądź wyniki operacji na danych.

W rezultacie nawet bardzo proste operacje wymagałyby nieustannego odczytywania i zapisywania zwykle bardzo dużych plików (plików, które nie mieszczą się w pamięci operacyjnej).

System zarządzania bazami danych musi wykorzystywać hierarchię pamięci i stosować różne strategie prowadzące do zwiększania wydajności.

Przykładem takiej strategii jest szeregowanie żądań dostępu algorytmem windy. Głowice dysku przesuwają się w kierunku od najbardziej wewnętrznego cylindra do najbardziej zewnętrznego i z powrotem. Gdy osiągną cylinder, którego dotyczy żądanie, wykonują żądane zapisy i odczyty, i przesuwają się dalej. Gdy osiągną położenie dla którego nie ma już żądań dla dalszych cylindrów, to zatrzymują się i rozpoczynają ruch w przeciwnym kierunku.

Innym przykładem jest wykorzystanie hierarchii pamięci przez wstępne ładowanie bloków. System przewiduje kolejność żądań dostępu do bloków i przesyła bloki do szybszej pamięci, zanim pojawi się żądanie.

Najmniejsza, najszybsza i najdroższa jest pamięć podręczna (*cache*) procesora. Pracuje ona z szybkością procesora.

Pamięć podręczna poziomu pierwszego (*L1-cache*) jest zintegrowana w jednym układzie scalonym z mikroprocesorem. Pamięć podręczna poziomu drugiego (*L2-cache*) stanowi odrębny układ scalony. Kolejno wprowadzono pamięci *L3-cache* i *L4-cache*. Pamięci kolejnych poziomów są coraz większe, ale dostęp do danych jest coraz wolniejszy.

Dla przykładu, procesor *Intel Core i7-5960X* ma pamięci:

- cache L1: 8 x 32kB + 8 x 32 kB
- cache L2: 8 x 256 kB
- cache L3: 20 MB

i pracuje z szybkością od 3,0 do 3,5 GHz.

Większa, wolniejsza ale tańsza jest pamięć operacyjna. Każdej wartości z pamięci podręcznej odpowiada dokładnie jedna wartość w pamięci operacyjnej. Zmiana wartości w pamięci podręcznej często wymaga natychmiastowego skopiowania tej wartości do pamięci operacyjnej; szczególnie w systemach wieloprocesorowych.

Obecnie pamięci operacyjne *DDR3 DIMM* (zwykle 1600 MHz) w komputerach osobistych mają wielkość 8, 16 lub 32 GB, czyli są nawet 1000 razy większe od pamięci *cache*, ale dostęp do danych jest około 10 razy wolniejszy niż do *cache* procesora.

Podobnie pamięci DDR4 (dwukrotnie większa częstotliwość, rzędu 4000 MHz).

Pamięci zewnętrzne w postaci dysków magnetycznych mają pojemność 1 TB lub większą, czyli są około 100 razy większe od pamięci operacyjnych. Czas dostępu (do bloku) jest rzędu >10 ms, podczas gdy dla pamięci *cache* są to nanosekundy.

Dyski SSD mają czas dostępu rzędu >0,1 ms, ale nie oznacza to, że są 100 razy szybsze. Dla przykłady transfery danych zmierzone dla szybkich dysków HDD i SDD wyniosły obecnie odpowiednio 160 MB/s i 400 MB/s. Transfer danych dla SSD jest więc tylko 2,5 raza większy.

Pamięci dyskowe zaliczamy do drugorzędnych. Niekiedy wciąż stosowane są jeszcze pamięci trzeciorzędne (pamięci te wymagają znalezienia odpowiedniego nośnika i załadowania go do urządzenia). Pojemności są oczywiście dowolnie duże, ale czas dostępu liczy się tutaj w sekundach, a nawet w minutach.

Algorytm wstępnego ładowania bloków przenosi do pamięci operacyjnej całą ścieżkę lub cały cylinder dysku, zanim nastąpi żądanie dostępu do zapisanych tam danych. Podobnie działa wstępne ładowanie danych do pamięci podręcznej. Dane są tam przenoszone, zanim pojawi się żądanie ich odczytu.

Przyczytaj:

http://wazniak.mimuw.edu.pl/, wykład 6

Garcia-Molina, Ullman, Widom...: rozdział 11

WSPÓŁBIEŻNOŚĆ

Serwer bazodanowy nie może obsługiwać klientów sekwencyjnie: wszyscy musieli by czekać na obsłużenie przez serwer w kolejce. W przypadku większości systemów komunikujących się z bazami danych (np. internetowe programy rezerwacji, systemy bankowe, kasy w supermarketach) wymaga się krótkich czasów odpowiedzi dla setek równolegle pracujących użytkowników. Nie można ich ustawić w kolejkę!

Sterowanie współbieżnością to proces zapewniający możliwość przetwarzania opartego na współistnieniu wielu procesów operujących na wspólnych (współdzielonych) danych w SZBD.

Transakcją nazywamy grupę instrukcji, które muszą być wykonane, aby odpowiednie zmiany zostały zapisane w bazie. Jeżeli chociaż jedna instrukcja z takiej grupy zakończy się niepowodzeniem, wówczas działanie wszystkich jest odwoływane.

BEGIN TRANSACTION – rozpoczęcie transakcji

COMMIT – zakończenie transakcji z zaakceptowaniem wszystkich zmian

ROLLBACK – zakończenie transakcji z wycofaniem wszystkich zmian

ROLLBACK TO SAVEPOINT *NazwaPunktuZapisu* – zakończenie transakcji z wycofaniem wszystkich tych zmian, które nastąpiły od zdefiniowanego (w czasie działania transakcji) punktu zapisu *NazwaPunktuZapisu*.

Transakcja powinna spełniać właściwości określane jako *ACID*:

— niepodzielność (*Atomicity*)
— spójność (*Consistency*)
— izolacja (*Isolation*)
— trwałość (*Durability*)

Atomowość to niepodzielność transakcji: albo wszystkie modyfikacje danych zakończą się sukcesem, albo żadna modyfikacja nie nastąpi. Zatem, jeżeli z jakiegoś powodu transakcja nie może być zakończona, to tzw. mechanizm odtwarzania musi zapewnić wycofanie wszystkich zmian wprowadzzonych już przez tę transakcję w bazie danych.

Spójność oznacza, że po zakończeniu transakcji baza musi być w stanie spójnym, tj. muszą być zachowane wszystkie więzy integralności, a wewnętrzne struktury bazy (np. indeksy) powinny być doprowadzone do prawidłowego stanu.

Własność izolacji mówi, że modyfikacje przeprowadzane przez daną transakcję muszą być odizolowane od innych działających transakcji, nie może kolidować ze współbieżnym wykonywaniem innych transakcji.

Trwałość – po zakończeniu transakcji jej efekty muszą pozostać w bazie na stałe. Nie mogą zostać utracone w wyniku jakiejkolwiek awarii.

PROBLEMY Z TRANSAKCJAMI

1. Problem utraconej modyfikacji (*lost update*). Modyfikacja wykonana w t_6 przez transakcję A zostaje utracona w momencie t_8.

Czas	Transakcja A	Transakcja B
t1	-	-
t2	*retrieve p*	-
t3	-	-
t4	-	*retrieve p*
t5	-	-
t6	*update p*	-
t7	-	-
t8	-	*update p*
t9	-	-

2. Problem czytania brudnopisu (*dirty read*): Transakcja A odczytuje w czasie t4 wartość zmodyfikowaną wcześniej (w czasie t2) przez inną transakcję jeszcze nie zatwierdzoną. W momencie cofnięcia transakcji B odczyt w czasie t4 staje się fałszywy.

Czas	Transakcja A	Transakcja B
t1	-	-
t2	-	*update p*
t3	-	-
t4	*retrieve p*	-
t5	-	-
t6	-	*rollback p*
t7	-	-

3. Problem niespójnej analizy (*non-repeatable read*): Transakcja A sumuje salda 3 rachunków (r_1, r_2, r_3) o wartościach w czasie t1: 200 zł, 300 zł i 4000 zł. Równolegle transakcja B wykonuje przelew 1000 zł z rachunku r_3 na r_1. Gdyby A wykonać ponownie w czasie t10–t12 to wynik byłby inny.

Czas	Transakcja A	Transakcja B
t1	suma ← r1	-
t2	suma ← suma+r2	-
t3	-	*retrieve r3 (4000)*
t5	-	*update r3 (4000-1000)*
t6	-	*retrieve r1 (200)*
t7	-	*update r1 (200+1000)*
t8	-	*commit*
t9	suma ← suma+r3	-

4. Wiersze widma (*phantom reads*): Transakcja A odczytuje rekordy, które spełniają pewne kryterium wyboru. Druga transakcja (B) wstawia nowe rekordy do tej samej tabeli, przy czym niektóre z nich spełniają kryterium sprawdzane przez A. Transakcja B może również modyfikować wiersze w taki sposób, że dodatkowe rekordy zaczną spełniać to kryterium. Mogą zatem pojawić się takie rekordy (fantomy), które spełniają dane kryterium, a które nie zostały odczytane przez A.

Lekarstwo na problemy: BLOKADY

– blokady wyłączne (typu X, blokady do zapisu)
– blokady wspólne (typu S, blokady do odczytu)

1. Jeśli transakcja założy blokadę X na krotkę p, to próba założenia jakiejkolwiek blokady przez inną transakcję na tej samej krotce zostanie oddalona.
2. Jeśli transakcja A założy blokadę S na krotkę p, to:
– próba założenia blokady X przez transakcję B na tej samej krotce zostanie oddalona,
– próba założenia blokady S przez transakcję B na tej samej krotce zostanie zaakceptowana, czyli obie transakcje będą blokować p.

Propozycja protokołu dostępu do danych

1. Transakcja, która chce uzyskać dostęp do krotki, musi najpierw uzyskać blokadę S na tej krotce.

2. Transakcja, która chce modyfikować krotkę, musi najpierw uzyskać blokadę X na tej krotce. Jeśli transakcja wcześniej założyła blokadę S, to musi ona zwiększyć poziom blokady z S do X.

3. Jeżeli żądanie blokady od transakcji B zostanie odrzucone ze względu na to, że jest w konflikcie z blokadą założoną wcześniej przez transakcję A, to B przechodzi w stan oczekiwania aż ta blokada zostanie zdjęta (system powinien dbać o to, by transakcja B nie czekała w nieskończoność, tj. by nie nastąpiło zagłodzenie).

4. Blokady S i X są utrzymywane do końca działania transakcji (tj. do polecenia COMMIT lub ROLLBACK).

Zakleszczenia

Zakleszczenie to sytuacja, w której dwie lub więcej transakcji oczekuje na zwolnienie „wzajemnej” blokady.

Strategie rozwiązywania zakleszczeń:

1. cofnąć losowo wybraną transakcję
2. cofnąć transakcję, które najdłużej trwa
3. cofnąć najkrócej trwającą transakcję

Protokół dwufazowego blokowania (*2 Phase Locking*)

1.Zanim transakcja rozpocznie działanie na pewnym obiekcie w bazie danych, musi założyć na ten obiekt blokadę.
2. Po zwolnieniu blokady transakcja nie może zakładać żadnej nowej blokady na jakimkolwiek obiekcie.

Twierdzenie

Jeśli wszystkie transakcje spełniają protokół dwufazowego blokowania, to wszystkie przeplatane porządki (współbieżne) są szeregowalne (poprawne).

Przeczytaj w podręczniku R.Elmasri, Sh.Navathe:

Rozdz. 17. Wprowadzenie do problematyki i teorii przetwarzania transakcji (571-600)

Rozdz. 18. Techniki sterowania współbieżnego (601-628)

Transakcje i zasady ACID; poziomy izolacji.

Transakcją nazywamy grupę instrukcji, które muszą być wykonane, aby odpowiednie zmiany zostały zapisane w bazie. Jeżeli chociaż jedna instrukcja z takiej grupy zakończy się niepowodzeniem, wówczas działanie wszystkich jest odwoływane.

Transakcja powinna spełniać właściwości określane jako *ACID*:

— niepodzielność (*Atomicity*)
— spójność (*Consistency*)
— izolacja (*Isolation*)
— trwałość (*Durability*)

Z transakcjami związane są blokady dostępu i poziomy izolacji.

PYTANIE NA EGZAMIN LICENCJACKI

Transakcja może zakładać blokady na elementy danych, zmieniać swoje blokady i zdejmować swoje blokady. Blokada do odczytu (*read lock*, *shared lock*) pozwala na współużytkowanie tego elementu danych przez inne transakcje. Blokada pełna (odczytu i zapisu, na wyłączność, *read-write lock*, *exclusive lock*) nie pozwala na dostęp do elementu danych innym transakcjom. Transakcja może rozszerzyć blokadę, zawęzić blokadę i znieść blokadę.

Transakcja jest zgodna z protokołem blokowania dwufazowego jeżeli występują dwie fazy. W pierwszej transakcja może zakładać i rozszerzać blokady. W drugiej transkacja może zawężać i znosić blokady, ale nie może już ani rozszerzać ani zakładać nowych blokad. Jeżeli każda transakcja w harmonogramie jest zgodna z opisanym protokołem, to harmonogram jest szeregowalny.

Mechanizm blokad prowadzi do tworzenia kolejek transakcji oczekujących na możliwość założenia blokady, a to może doprowadzić do powstawania zakleszczeń. Musimy więc stosować różne protokoły zapobiegania zakleszczeniom.

Poziom izolacji to ustalona przez programistę cecha transakcji. Poziom ten określa się przy użyciu instrukcji ISOLATION LEVEL <X>, gdzie <X> może przyjąć cztery wartości.

Serializable – to zwykle poziom domyślny. Gwarantuje, że dane odczytywane to dane utworzone wyłącznie przez zatwierdzone transakcje oraz że wartość żadnej danej odczytywanej lub zapisywanej przez daną transakcję nie zostanie zmieniona przez inną transakcję do momentu zakończenia danej transakcji. Innymi sło-

wy system zarządzania transakcjami nie dopuści do odczytów zmodyfikowanych, odczytów niepowtarzalnych i fantomów.

Uwaga: Nie jest to dokładnie to samo co szeregowalność!

Repeatable read – możliwe wystąpienie fantomów.

Read Committed – możliwe wystąpienie fantomów i odczytów niepowtarzalnych

Read Uncommitted – możliwe wszystkie naruszenia.