Zarządzanie współbieżnością transakcji

Tomasz Koszlajda
Instytut Informatyki Politechniki Poznańskiej
e-mail: Tomasz.Koszlajda@cs.put.poznan.pl

1. Wstęp

Jednym z potencjalnych źródeł błędów danych przechowywanych w bazach danych są niepożądane interferencje między asynchronicznymi operacjami współbieżnie wykonywanych transakcji. Teoria systemów baz danych już w latach siedemdziesiątych i osiemdziesiątych zidentyfikowała problem zarządzania współbieżnością transakcji i zaproponowała kilka alternatywnych algorytmów synchronizujących. Najbardziej znane z tych algorytmów to: blokowanie dwufazowe, metoda znaczników czasowych i wielowersyjność danych. Algorytmy te rozwiązują problem poprawności współbieżnie wykonywanych transakcji.

W większości komercyjnych systemów zarządzania bazami danych zaimplementowano mechanizmy synchronizacji wzorowane na jednej z tych metod. Pełna automatyzacja procesu synchronizacji transakcji ma na celu odciążenie programistów budujących aplikacje baz danych od konieczności samodzielnego rozwiązywania trudnych problemów zarządzania współbieżnością dostępu do danych.

Jednak z powodu ograniczania przez algorytmy synchronizujące efektywności przetwarzania danych, producenci systemów komercyjnych próbują znaleźć kompromis między poprawnością, a efektywnością procesu przetwarzania transakcyjnego. Zazwyczaj domyślnym trybem pracy tych systemów jest tryb nie gwarantujący pełnej poprawności przetwarzania współbieżnego, ale za to charakteryzujący się większą wydajnością. Dla zagwarantowania pełnej poprawności procesu przetwarzania niezbędne jest zastosowanie dodatkowych mechanizmów systemowych. Wymaga to od konstruktorów aplikacji baz danych przeanalizowania specyfiki danego zastosowania i określenia właściwego trybu synchronizacji transakcji. Zignorowanie tego problemu przez programistów może doprowadzić do błędów w bazie danych i trudnych do oszacowania strat ich użytkowników.

2. Poprawność realizacji współbieżnych transakcji

2.1. Definicja transakcji

Jednym z podstawowych wymogów stawianych bazom danych jest ich spójność, polegająca na poprawności składowanych w nich danych. Jednak nawet dobrze napisane i poprawnie użytkowane aplikacje bazy danych, działające w zawodnym, współbieżnym i rozproszonym środowisku sprzętowo-programowym, nie gwarantują spójności bazy danych. Głównymi zagrożeniami dla poprawności przechowywanych i przetwarzanych danych są: anormalne przerwanie procesu przetwarzania, uszkodzenie nośnika danych oraz interferencje między równoległymi sesjami kilku użytkowników. Remedium na powyższe zagrożenia są transakcje.

Transakcja jest sekwencją logicznie powiązanych operacji wykonywanych na bazie danych w ramach sesji użytkownika. Przykładem takich logicznie powiązanych operacji są operacje: umniejszenia stanu konta i wypłaty przez bankomat odpowiedniej sumy pieniędzy. Te dwie operacje są od siebie wzajemnie zależne, muszą być wykonane razem. Osobno, żadna z nich - wypłata pieniędzy bez umniejszania stanu konta lub umniejszenie tanu konta bez wypłaty pieniędzy - nie implementuje poprawnej operacji bankowej.

W uproszczeniu można przyjąć, że transakcje wykonują na bazie danych cztery podstawowe rodzaje operacji: odczytu, zapisu, wstawienia i usunięcia danych. Podział sesji użytkowników na transakcje jest realizowany za pomocą pary dwóch dodatkowych operacji: zatwierdzenia (ang. commit) i wycofania (ang. rollback) transakcji. Operacja zatwierdzenia jest pomyślnym zakończeniem transakcji równoważnym zatwierdzeniu wszystkich wykonanych przez nią modyfikacji stanu bazy danych. Operacja wycofania oznacza niepomyślne zakończenie transakcji i unieważnienie wszystkich modyfikacji wprowadzonych przez transakcję. Wycofanie transakcji jest równoważne sytuacji, nie wystąpienia tej transakcja w historii bazy danych. Transakcje mogą być wycofywane na życzenie użytkownika lub automatycznie przez system zarządzania bazą danych, w przypadku anormalnego przerwania procesu przetwarzania transakcji.

Aby uodpornić bazę danych na wymienione zagrożenia spójności bazy danych, transakcje muszą posiadać ściśle określone cechy. Są nimi: atomowość, spójność, izolacja i trwałość, znane razem pod akronimem ACID (od ang. **A**tomicity, **C**onsistency, **I**solation, **D**urability). Atomowość transakcji oznacza, że w wypadku anormalnego przerywania procesu przetwarzania danych, operacje wchodzące w skład transakcji muszą wykonać się wszystkie razem lub żadna z nich. Spójność transakcji oznacza, że transakcja gwarantuje spójność bazy danych jako całość. W czasie wykonywania transakcji, a przed jej zakończeniem stan bazy danych może pozostawać przejściowo niespójny. Izolacja transakcji oznacza brak interakcji między równoległymi transakcjami. Mimo faktycznej współbieżności transakcje „muszą mieć złudzenie", że są wykonywane w sposób sekwencyjny. Trwałość transakcji oznacza z kolei, że modyfikacje stanu bazy danych wprowadzone przez poprawnie zakończone transakcje muszą mieć charakter trwały, mimo ewentualnych uszkodzeń nośników danych. Własności ACID powinny być implementowane przez system zarządzania bazą danych, w sposób transparentny dla programistów i użytkowników aplikacji bazy danych.

2.2. Przykłady błędów wynikających ze współbieżności transakcji

Interferencje między współbieżnymi transakcjami polegają na równoczesnym, konfliktowym dostępie różnych transakcji do tych samych danych. Wynikiem takich interferencji są błędy pozostawione przez transakcje w bazie danych lub błędny obraz bazy danych widziany przez transakcje, czyli błędne raporty, wykresy lub wartości odczytywane przez użytkowników bazy danych. Sklasyfikowano kilka podstawowych typów takich interferencji nazywanych anomaliami.

Brudny zapis (ang. *dirty write*) jest anomalią polegającą na tym, że po zapisaniu danej x przez transakcję T_1 (operacja $w_1(x)$) i przed jej zakończeniem - pomyślnym (operacja c_1) bądź niepomyślnym (operacja a_1) - następuje zapis tej samej danej przez inną transakcję T_2 (operacja $w_2(x)$). Anomalią jest więc sekwencja operacji: $w_1(x)$, $w_2(x)$, ..., $\{c_1 \mid a_1\}$. Załóżmy, że początkowa wartość danej x wynosiła 100, transakcja pierwsza nadała jej wartość 200, a transakcja druga wartość 300; i dodatkowo transakcja pierwsza zakończyła się operacją wycofania. Dla takiej sekwencji nie można jednoznacznie stwierdzić jaka powinna być poprawna wartość danej x po wycofaniu transakcji pierwszej. Czy należy przywrócić wartość 100, pozostawić 300, czy może odjąć różnicę między 200 i 100?

Brudny odczyt (ang. *dirty read*) jest anomalią polegającą na tym, że po zapisaniu danej x przez transakcję T_1 i przed jej zakończeniem, następuje odczyt tej danej przez inną transakcję T_2. W przypadku zatwierdzenia transakcji T_2 i jednoczesnym wycofaniu transakcji T_1, obraz bazy danych (stan danej x) widziany przez transakcję T_2 jest niepoprawny. W związku z wycofaniem transakcji T_1, transakcja T_2 odczytała stan danej x, który formalnie nigdy nie zaistniał. Anomalią jest sekwencja operacji: $w_1(x)$, $r_2(x)$, ...,(w dowolnej kolejności a_1 lub c_2).

Rozmyty odczyt (ang. *fuzzy read* lub *unrepeatable read*) jest anomalią polegającą na tym, że po odczytaniu danej x przez transakcję T_1, następuje zapis lub usunięcie tej danej przez transakcję T_2, która następnie zostaje zatwierdzona. Jeżeli transakcja T_1 będzie próbowała jeszcze raz odczytać x, otrzyma ona inną wartość tej danej lub odkryje, że dana ta została usunięta. Anomalią jest sekwencja operacji: $r_1(x)$, ..., $w_2(x)$, ..., c_2, ... , $r_1(x)$.

Utracona modyfikacja (ang. *lost update*) jest anomalią polegającą na tym, że po odczytaniu danej x przez transakcje T_1 i T_2, następuje zapis zmodyfikowanej wartości danej x przez transakcję T_1, która następnie może zostać zatwierdzona. Jeżeli teraz transakcja T_2 również zapisze zmodyfikowaną wartość danej x, modyfikacja wprowadzona przez transakcję T_1 zostanie utracona. Anomalią jest więc sekwencja operacji: $r_1(x)$, ..., $r_2(x)$, ..., $w_1(x)$, ..., c_1, ... , $w_2(x)$.

Fantom (ang. *phantom*) jest anomalią polegającą na tym, że po odczytaniu przez transakcję T_1, zbioru danych X spełniających warunek logiczny p, następuje wstawienie przez transakcję T_2 danej x, która również spełnia warunek p, lub modyfikacja istniejącej danej x, która do tej pory nie spełniała warunku p, ale w wyniku tej modyfikacji będzie go spełniała. Jeżeli transakcja T_1 będzie próbowała jeszcze raz odczytać dane, które spełniają warunek p, otrzyma ona w wyniku zbiór danych X' równy zbiorowi X powiększonemu o daną x. O danej x mówi się w tym wypadku, że jest fantomem, który pojawił się w wyniku powtórzonego zapytania. Anomalią jest sekwencja operacji: $r_1(X)$, ..., $w_2(x)$, ..., , $r_1(X')$.

2.3. Definicja izolacji (uszeregowalności – ang. serializability) transakcji

Operacje należące do różnych transakcji, uporządkowane w kolejności ich wykonywania są nazywane *historiami* lub *realizacjami* tego zbioru transakcji. W zbiorze wszystkich potencjalnych historii danego zbioru transakcji znajdują się *historie sekwencyjne*, czyli takie, w których operacje żadnej z transakcji nie przeplatają się z operacjami innych transakcji, oraz *historie współbieżne*, w których operacje różnych transakcji przeplatają się. Historie sekwencyjne z definicje są wolne od błędów wynikających ze współbieżności. Historie współbieżne mogą być zarówno poprawne, to jest wolne od wszystkich (nie tylko tych wymienionych powyżej) anomalii jak i niepoprawne. Definicja poprawnych historii współbieżnych jest oparta na pojęciu *uszeregowalności transakcji* (ang. serializability).

Wzorcem poprawności współbieżnych historii transakcji są historie sekwencyjne. Historia współbieżna, która pozostawia taki sam stan bazy danych i, w której poszczególne transakcje odczytując dane widzą takie same wartości jak te same transakcje w sekwencyjnej historii tego zbioru transakcji, jest historią równoważną tej historii sekwencyjnej. Historie współbieżne równoważne w powyższy sposób choć jednej sekwencyjnej historii tych samych transakcji są nazywane historiami *uszeregowalnymi*. Brak uszeregowalności historii współbieżnych jest przyczyną niespójności baz danych lub przyczyną błędów w odczycie stanu bazy danych.

Niech $T_1=(r_1(x), w_1(x-100), r_1(y), w_1(y+100), c_1)$ i $T_2=(r_2(x), r_2(y), c_2)$ będą dwoma danymi transakcjami. Transakcja T_1 odczytuje, a następnie modyfikuje dane x i y. Transakcja T_2 odczytuje stan danych x i y wyświetla ich sumę użytkownikowi. Poniżej przedstawiono trzy przykładowe historie transakcji T_1 i T_2. Historia H_1:

$$H_1=(r_1(x), w_1(x), r_1(y), w_1(y), c_1, r_2(x), r_2(y), c_2)$$

jest przykładem historii sekwencyjnej. Transakcja T_2 odczytuje stan danych x i y po zakończeniu transakcji T_1. Historia H_2:

$$H_2=(r_1(x), r_2(x), w_1(x), r_1(y), r_2(y), w_1(y), c_1, c_2)$$

jest współbieżną historią uszeregowalną. Historia ta jest równoważna historii sekwencyjnej, w której transakcja T_2 poprzedza transakcję T_1. Transakcja T_2 odczytuje stan danych x i y sprzed zakończenia transakcji T_1. Historia H_3:

$$H_3=(r_1(x), r_2(x), w_1(x), r_1(y), w_1(y), r_2(y), c_1, c_2)$$

jest z kolei historią współbieżną nieuszeregowalną. Nie jest ona równoważna żadnej z dwóch możliwych sekwencji transakcji T_1 i T_2. Transakcja T_2 odczytuje stan danej x sprzed transakcji T_1, a stan danej y po zakończeniu T_1.

Niektóre historie uszeregowalne posiadają w dalszym ciągu pewne niepożądane cechy. Na przykład historia H_4:

$$H_4=(r_1(x), w_1(x), r_2(x), r_1(y), w_1(y), r_2(y) , c_2, r_1(z), w_1(z), c_1)$$

jest uszeregowalna, bo jest równoważna historii sekwencyjnej, w której T_1 poprzedza T_2. Rozważmy jednak przypadek awarii systemu, która wystąpi po operacji zatwierdzenia transakcji T_2:

$$H_4=(r_1(x), w_1(x), r_2(x), r_1(y), w_1(y), r_2(y) , c_2, <awaria>)$$

i wymusi tym samym wycofanie przez system transakcji T_1 (aby zagwarantować jej atomowość), która nie została jeszcze pomyślnie zakończona. Zakończona transakcja T_2 musi zostać również wycofana, ponieważ odczytała ona stan bazy danych wprowadzony przez T_1, która formalnie nie wystąpiła w historii bazy danych. Jest to sprzeczne z zasadą trwałości transakcji. Historie transakcji, które na wypadek awarii nie wymagają kaskadowego wycofywania już zatwierdzonych transakcji, są nazywane historiami *odtwarzalnymi* (ang. recoverable).

Celem zarządzania współbieżnością transakcji jest takie sterowanie kolejnością operacji w spontanicznych, współbieżnych historiach transakcji, aby w bazie danych realizowane były jedynie uszeregowalne i odtwarzalne historie transakcji.

2.4. Poziomy izolacji

Większość komercyjnych systemów baz danych umożliwia użytkownikom baz danych korzystanie z trybu pracy systemu, w którym dla podniesienia wydajności przetwarzania ogranicza się poprawność generowanych historii. Rozwiązanie to umożliwia elastyczne dopasowanie poziomu izolacji współbieżnych transakcji do wymaganego w danym zastosowaniu modelu przetwarzania. Poziomy izolacji definiuj się w oparciu o anomalie. Dany poziom izolacji określa jakie anomalie są w nim dopuszczalne, a jakie zabronione.

Standardy i komercyjne systemy baz danych definiują różne, często pokrywające się poziomy izolacji. Wynika to z różnic w zastosowanych metodach synchronizacji. Każdy z algorytmów synchronizacji ma naturalne trudności z usunięciem określonych typów anomalii. Na przykład, w metodzie blokowania dwu-fazowego trudna jest eliminacja anomalii typu *fantom*, a w algorytmach wielowersyjnych złożona jest eliminacja anomalii *utraconego zapisu*.

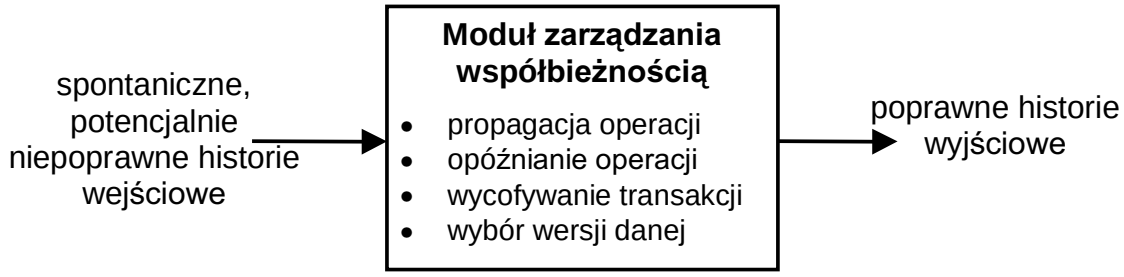
Jedną z definicji poziomów izolacji jest standard SQL92. Definiuje on trzy poziomy izolacji bazujące na anomaliach: *brudnego odczytu*, *rozmytego odczytu* i *fantomu*. Definicję tę przedstawiono w tabeli poniżej:

Dopuszczalne anomalie / **Poziom izolacji**	**Brudny zapis**	**Brudny odczyt**	**Rozmyty odczyt**	**Fantom**
Read uncommitted	nie	tak	tak	tak
Read committed	nie	nie	tak	tak
Repeatable read	nie	nie	nie	tak
Serializable	nie	nie	nie	nie

Nazwa poziomu izolacji: *Serializable* jest myląca, ponieważ z jej definicji nie wynika jednoznacznie, że uszeregowalność nie dopuszcza jakichkolwiek anomalii współbieżnego wykonania, a nie tylko czterech wymienionych. Definicja ta zakłada milcząco, że stosowaną metodą synchronizacji jest blokowanie dwufazowe. Dla tej metody, eliminacja rozmytego odczytu i fantomów przy okazji eliminuje również inne nie wymienione anomalie. Dla innych metod synchronizacji transakcji założenie to nie jest prawdziwe.

3. Algorytmy synchronizacji współbieżnych transakcji

Zarządzanie współbieżnością transakcji ma na celu transformację wejściowych, powstałych w spontaniczny sposób, niepoprawnych historii transakcji, w historie uszeregowalne i odtwarzalne. Wszystkie poprawne wejściowe historie powinny pozostać niezmienione i wykonywać się bez opóźnienia. Cel działania modułu zarządzania współbieżnością transakcji przedstawiono symbolicznie na rysunku poniżej.



Metody synchronizacji współbieżnych transakcji różnią się między sobą sposobami zmiany kolejności operacji w spontanicznych historiach wejściowych. Metody wykorzystujące blokady zmieniają kolejność wejściowych sekwencji operacji poprzez opóźnianie wykonywania operacji konfliktowych. Metody znaczników czasowych wykrywają niepoprawne historie i zmieniają je przez wycofywanie transakcji naruszających uszeregowalność transakcji. Metody wykorzystujące wielowersyjność danych unikają konfliktów między operacjami przez wykonywanie konfliktowych operacji na różnych wersjach danych. Ponadto poszczególne metody różnią się stopniem ograniczania współbieżnego wykonywania transakcji.

3.1. Synchronizacja za pomocą blokowania dostępu do danych

Najpowszechniej stosowanym mechanizmem synchronizacji jest metoda blokowania dwufazowego. W metodzie tej każda operacja na danych musi być poprzedzona ustawieniem odpowiedniej blokady na tej danej. W przypadku uzyskania wymaganej blokady dana operacja jest bezzwłocznie wykonywana. Niemożność ustawienia blokady ze względu na konflikt z inną już założoną blokadą, zawiesza wykonanie związanej z nią operacji aż do momentu zdjęcia konfliktowej blokady. Stosowane są dwa rodzaje blokad: blokada wyłączna do zapisu X i blokada współdzielona do odczytu S. Konfliktowość blokad przedstawiona została w tabeli poniżej. Znak „✓” w tabeli oznacza kompatybilność blokad; takie blokady mogą być równolegle ustawione na tej samej danej. Natomiast znak „-” oznacza konfliktowość blokad; ustawienie żądanej blokady (i wykonanie powiązanej z nią operacji) będzie odroczone aż do momentu zdjęcia blokady konfliktowej.

Blokada założona / Blokada żądana	Brak blokad	Blokada **S**	Blokada **X**
Blokada **S**	✓	✓	-
Blokada **X**	✓	-	-

Metoda blokowania dwufazowego zakłada i zdejmuje blokady dla każdej transakcji w dwóch rozłącznych fazach. Blokady związane z poszczególnymi operacjami są zakładane w pierwszej fazie transakcji, a zdejmowane w fazie drugiej. Fazy transakcji są rozdzielone przez tak zwany *punkt akceptacji* transakcji. Jest to moment tuż po wykonaniu przez transakcję wszystkich operacji,

ale jeszcze przed jej zakończeniem. Tak działający algorytm nie dopuszcza do powstania większości anomalii.

Sprawdźmy to na przykładzie nieuszeregowalnej historii H_3. Historia ta po synchronizacji przez algorytm blokowania dwufazowego zostanie zmodyfikowana do poprawnej historii H_3':

$$H_3'=(r_1(x), r_2(x), \mathbf{(1)}, r_2(y), c_2, \mathbf{(2)}, w_1(x), r_1(y), w_1(y), c_1).$$

W punkcie (1) nastąpiło zawieszenie wykonywania operacji $w_1(x)$ i całej transakcji T_1, ze względu na konflikt założonej blokady do odczytu transakcji T_2 i żądanej blokady do zapisu transakcji T_1. W punkcie (2) zakończenie transakcji T_2 powoduje zdjęcie wszystkich jej blokad i w konsekwencji odwieszenie wykonywania transakcji T_1.

W celu podniesienia efektywności działania dla dużej liczby zakładanych blokad, algorytm blokowania dwufazowego może być rozszerzony o możliwość blokowania danych o różnej ziarnistości. Zamiast zakładania indywidualnych blokad na wielu danych można zwiększyć ziarno blokowania do całego zbioru danych. Większymi ziarnami blokowania mogą być bloki danych, pliki, relacje, schematy lub całe bazy danych. Jednostki te tworzą hierarchię lub graf ziarnistości blokowania.

O ile pojedyncze dane są blokowane automatycznie przez system zarządzania bazą danych, to większe jednostki mogą być blokowane jawnie przez użytkowników lub programistów. Synchronizacja blokad na różnych poziomach ziarnistości wymaga wprowadzenia nowego typu blokad zwanych intencyjnymi. Uzyskanie blokady na niższym poziomie ziarnistości musi być poprzedzone założeniem blokad intencyjnych na wszystkich jednostkach wyższego poziomu.

Algorytm blokowania dwufazowego jest powszechnie stosowany ze względu na prostotę jego implementacji. Posiada on jednak szereg wad. Należą do nich: niski stopień współbieżności transakcji, występowanie zakleszczeń oraz brak zapewnienia uszeregowalności transakcji.

Niski stopień współbieżności transakcji wynika z tego, że algorytm ten jest mało selektywny. Powoduje on opóźnienia w realizacji nie tylko niepoprawnych, ale również poprawnych historii transakcji. Na przykład uszeregowalna i odtwarzalna historia H_5:

$$H_5=(r_1(x), w_2(x), r_1(y), c_1, c_2),$$

zostanie przez algorytm blokowania zmodyfikowana do historii H_5', w której wykonywanie transakcji T_2 zostanie zawieszone do czasu zakończenia transakcji T_1:

$$H_5'=(r_1(x), r_1(y), c_1, w_2(x), c_2).$$

W klasycznej wersji algorytm blokowania dwufazowego nie gwarantuje również pełnej uszeregowalności transakcji. Jest on nieodporny na anomalię fantomów. Przykładem niech będzie historia H_6:

$$H_6=(r_1(X), i_2(y\in X), c_2, r_1(X'=X\cup y), c_1).$$

Duże X w $r_1(X)$ oznacza odczyt całego zbioru danych spełniających warunek logiczny zapytania. Operacja $i_2(y\in X)$ jest operacją wstawienia danej y, która również spełnia ten warunek. Po zakończeniu transakcji drugiej kolejne zapytanie zwraca inny wynik obejmujący również fantom - daną y. Historia H_6 jest nieuszeregowalna. Transakcja T_1 widzi w pierwszym zapytaniu stan bazy danych sprzed wykonania transakcji T_2, a w drugim stan po wykonaniu T_2. Jednak algorytm blokowania dwufazowego nie zmodyfikuje tej historii. Nie umie on zsynchronizować operacji odczytu zbioru danych i operacji wstawiania danych do tego zbioru. Komercyjne systemy baz danych (na przykład produkt IBM – DB2) rozwiązują problem fantomów przez bardziej restrykcyjne blokowanie. Podczas operacji odczytu lub zapisu blokowane są nie pojedyncze dane, ale całe zbiory danych. Jednak ze względu na radykalne obniżenie stopnia współbieżności takie rozwiązanie jest traktowane jako opcjonalne.

3.2. Synchronizacja za pomocą znaczników czasowych

Zdecydowanie mniej popularna metodą synchronizacji transakcji są algorytmy, których działanie opiera się na etykietowaniu danych znacznikami czasowymi transakcji. W metodzie tej jako poprawną przyjmuje się kolejność wykonywania transakcji zgodną z czasami pojawiania się transakcji w systemie. Każda z transakcji w momencie rozpoczęcia otrzymuje unikalny znacznik czasowy (może być to na przykład kolejny numer sekwensera), a każdą dana przechowuje dwa dodatkowe atrybuty pamiętające znacznik czasowy najmłodszej transakcji, która ją odczytała (RTS: read time stamp) i znacznik czasowy ostatniej transakcji, która ją zmodyfikowała (WTS: write time stamp). Transakcje przetwarzając dane pozostawiają w tych atrybutach ślady swojego działania. Mechanizm synchronizacji polega na wycofywaniu transakcji, których znacznik czasowy jest wcześniejszy od znaczników związanych z odczytywaną lub zapisywaną daną.

Załóżmy, że w bazie danych zawierającej dane x (RTS=5, WTS=4) i y (RTS= 3, WTS=1) wykonywana jest następująca nieuszeregowalna historia transakcji T_1 i T_2:

$$H_3=(r_1(x), r_2(x), w_1(x), r_1(y), w_1(y), r_2(y), c_1, c_2).$$

Transakcji T_1 nadano znacznik TS=6, a transakcji T_2 znacznik TS=7. Historia ta zostanie zmodyfikowana do następującej historii uszeregowalnej H_3':

$$H_3'=(r_1(x), r_2(x), \mathbf{(1)}\ a_1, r_2(y), \mathbf{(2)}, c_2, r_1(x), w_1(x), r_1(y), w_1(y), c_1).$$

Odczyty danej x przez transakcje T_1 i T_2 modyfikują znacznik czasowy transakcji odczytującej kolejno do wartości RTS= 6 i RTS=7. W punkcie (1) w związku z tym, że znacznik czasowy transakcji T_1 (TS=6) jest mniejszy od znacznika czasowego RTS=7 co oznacza brak synchronizacji (transakcja wcześniejsza wykonuje operację konfliktową po transakcji późniejszej), transakcja T_1 jest wycofana. W punkcie (2) następuje restart transakcji T_1 z nowym znacznikiem czasowym TS=8.

Rzadkie zastosowanie metody znaczników czasowych w komercyjnych systemach baz danych wynika z kosztownego sposobu synchronizacji jakim jest wycofywanie transakcji.

3.3. Algorytmy wielowersyjne

W wielowersyjnych algorytmach synchronizacji współbieżnych transakcji operacje zapisu nie powodują nadpisania starej wartości danych, lecz tworzą nową wersję danej, przy zachowaniu starej wersji. W przypadku wystąpienia operacji konfliktowych system może uniknąć konfliktu przez wykonanie tych operacji na różnych wersjach tej samej danej. Na przykład w sytuacji gdy transakcja chce oczytać daną zmodyfikowaną przez inną nie zakończoną transakcję, odczyt ten może dotyczyć starszej wersji tej danej. W ten sposób, wielowersyjne algorytmy synchronizacji znacznie podnoszą współbieżność synchronizowanych transakcji.

Załóżmy, że w wielowersyjnej bazie danych zawierającej dane wielowersyjne $x=\{x_0\}$ i $y=\{y_0\}$ wykonywana jest następująca nieuszeregowalna historia H_3:

$$H_3=(r_1(x), r_2(x), w_1(x), r_1(y), w_1(y), r_2(y), c_1, c_2).$$

Za pomocą algorytmu wielowersyjnego, historia ta zostanie zmodyfikowana do postaci:

$$H_3'=(r_1(x_0), r_2(x_0), \mathbf{(1)}\ w_1(x_1), r_1(y_0), w_1(y_1), \mathbf{(2)}\ r_2(y_0), c_1, c_2).$$

Historia ta jest uszeregowalna. Jest ona równoważna wersji sekwencyjnej, w której transakcja T_2 poprzedza transakcję T_1. Transakcja T_2 widzi bowiem stan danych x i y sprzed rozpoczęcia transakcji T_1. W punkcie (1) transakcja T_1 tworzy nową wersję x_1 danej x. W punkcie (2) algorytm wybiera do odczytu ze zbioru wersji danej $x=\{x_0, x_1\}$ starszą niekonfliktową wersję x_0.

Istnieje wiele wariantów algorytmów wielowersyjnych. Bazują one na klasycznych jednowersyjnych algorytmach synchronizujących. W wielowersyjnym algorytmie blokowania dwufazowego mogą istnieć tylko dwie wersje jednej danej. Operacje odczytu i zapisy nie są konfliktowe, a więc blokady do zapisu i odczytu są kompatybilne. W przypadku równoległego wykonywania przez różne transakcje operacji odczytu i zapisu, odczytywana jest starsza wersja danej, a operacja zapisu tworzy nową wersję. Dwie równoległe operacje zapisu pozostają konfliktowe. Na zakończenie transakcji, która utworzyła nową wersję danej następuje usunięcie starej wersji.

W wielowersyjnym algorytmie znaczników czasowych nie ograniczenia na liczbę wersji. Teraz znaczniki czasowe transakcji zapisujących i odczytujących są pamiętane dla każdej wersji danej. Dla operacji odczytu ze zbioru wersji wybierana jest ta, której znacznik czasowy transakcji, która ją utworzyła jest największym znacznikiem mniejszym niż znacznik danej transakcji odczytującej. Przykładowo z podanego poniżej zbioru wersji danej *x*, transakcja o znaczniku czasowym TS=12, odczyta wersję x_2:

$x = \{\mathbf{x_0}$:WTS=**1**, RTS=3, $\mathbf{x_1}$:WTS=**6**, RTS=6, $\mathbf{x_2}$:WTS=**9**, RTS=18, $\mathbf{x_3}$:WTS=**19**, RTS=21$\}$.

W wielowersyjnym algorytmie znaczników czasowych operacje zapisu nie są konfliktowe. Tworzą one nowe wersje danych. Jedyny przypadek konfliktu ma miejsce, gdy transakcja chce utworzyć nową wersję danych, która powinna być odczytana (zgodnie z porządkiem znaczników czasowych) przez już wykonaną operację odczytu. Dla danej x o zbiorze wersji z poprzedniego przykładu, taki konflikt wystąpi na przykład podczas próby operacji zapisu danej x przez transakcję o znaczniku czasowym TS=12, czyli podczas tworzenia nowej wersji x_4 ze znacznikami WTS=RTS=12. W realizacji uszeregowalnej nowa wersja powinna być odczytana przez transakcję o znaczniku czasowym TS=18. Tymczasem z historii danej x wynika, że operacja odczytu danej x przez transakcję o znaczniku TS=18 miała już miejsce i transakcja ta odczytała wersję x_2. Wersji x_4 transakcja o znaczniku TS=12 nie mogła odczytać, bo wersja ta w tym czasie jeszcze nie istniała.

4. Zarządzanie transakcjami w SZBD ORACLE

System zarządzania bazą danych produkcji firmy ORACLE realizuje funkcje zarządzania współbieżnością transakcji. Implementuje on algorytm synchronizacji, który jest złożeniem metod blokowania dwufazowego, wielowersyjnej metody znaczników czasowych, oraz reguły *pierwszy zatwierdzony wygrywa* (ang. first-committer-wins).

4.1. Metoda hierarchicznego blokowanie dwufazowego

Moduł zarządzania współbieżnością transakcji systemu ORACLE w domyślnym trybie pracy używa do synchronizacji operacji zapisu algorytmu blokowania dwufazowego o trzech poziomach ziarnistości: baza danych, relacja i krotka. Użytkownik może jawnie blokować całą bazę danych wybierając specjalny tryb otwarcia bazy danych. Jawne blokowanie relacji jest realizowane za pomocą polecenia: *LOCK TABLE nazwa_relacji IN typ_blokady*. Do wyboru jest pięć typów blokad: dwie zwykłe (współdzielona i wyłączna), dwie intencyjne (wyłączności i współdzielenia) i jedna mieszana.

Operacje odczytu nie są synchronizowane za pomocą blokad. System ORACLE nie zakłada blokad współdzielonych dla pojedynczych danych. Dzięki temu zapytania nie są blokowane przez system. Wyjątkiem są zapytania z klauzulą *FOR UPDATE OF ...* , dla których zakładane są blokady wyłączne i w związku z tym ich wykonanie może być opóźnione do czasu uzyskania blokady.

4.2. Wielowersyjna metoda znaczników czasowych

Do synchronizacji operacji odczytu stosowany jest wielowersyjny algorytm znaczników czasowych. Rolę znaczników czasowych pełni tak zwany SCN (ang. *System Change Number*) pobierany z systemowego sekwensera. Każdej transakcji w momencie jej zatwierdzania przypisywany jest unikalny znacznik czasowy SCN. Znacznik ten służy następnie do oznakowania wersji danych

utworzonych przez daną transakcję. Znacznik czasowy wykorzystywany do identyfikacji wersji danych właściwych dla danego zapytania jest pobierany z sekwensera SCN. Gwarantuje to spójność operacji odczytu. Wartość znacznika SCN może się zmieniać dla kolejnych zapytań tej samej transakcji.

Domyślny tryb pracy systemu ORACLE zapewnia poziom izolacji READ COMMITED. Jednostką synchronizacji operacji odczytu są pojedyncze zapytania, a nie cała transakcja. Tryb ten dopuszcza występowanie anomalii *rozmytego odczytu, utraconej modyfikacji* i *fantomu*.

4.3. Unikanie anomalii fantomów i utraty modyfikacji

Wyższe poziomy izolacji wykluczające anomalie rozmytego odczytu, utraconej modyfikacji i fantomu są dostępne opcjonalnie dla całych sesji lub pojedynczych transakcji. Istnieją dwa sposoby zwiększenia poziomu izolacji transakcji: ustawienia trybu transakcji tylko do odczytu oraz ustawienia poziomu izolacji SERIALIZABLE dla całej sesji lub pojedynczej transakcji.

Wybór jednej z tych opcji powoduje zmianę sposobu identyfikacji wersji obiektów przez transakcje. Transakcje w tym wypadku mają przydzielane dwa znaczniki czasowe SCN: z początku i końca transakcji. Końcowy znacznik czasowy jest wykorzystywany tak jak w trybie domyślnym, do znakowania tworzonych wersji danych. Początkowy znacznik czasowy jest wykorzystywany przez wszystkie zapytania transakcji do identyfikacji odczytywanych wersji danych. Eliminuje anomalie *rozmytego odczytu* i *fantomu*.

Dla usunięcia anomalii utraconej modyfikacji system ORACLE stosuje regułę *pierwszy zatwierdzony wygrywa*. Jeżeli dwie transakcje, których czasy realizacji określone przez znaczniki czasowe początku i końca transakcji pokrywają się, próbują modyfikować tę samą daną to transakcja, która jako pierwsza zainicjowała operację modyfikacji w wypadku pomyślnego zakończenie wymusza wycofanie drugiej transakcji. Druga transakcja nie będzie mogła wykonać swojej modyfikacji nawet po zwolnieniu blokady wyłącznej przez transakcję pierwszą. Wszelkie próby modyfikacji danej będą przez system odrzucone. Dzięki temu modyfikowane mogą być jedynie najświeższe wersje danych.

Zastosowanie trybu izolacji o mylącej nazwie SERIALIZABLE, nie gwarantuje pełnej izolacji transakcji. W dalszym ciągu istnieje wiele niestandardowych anomalii nie wykrywanych przez zastosowane mechanizmy synchronizacji.

Poniżej podano przykład nieuszeregowalnej historii transakcji błędnie synchronizowanej przez system ORACLE w trybie SERIALIZABLE. Załóżmy, że między danymi x i y zachodzi zależność $x > y$. Zależność ta jest zachowywana oddzielnie przez każdą z transakcji T_1 i T_2. Poniższa współbieżna realizacja może jednak spowodować niezachowanie tej zależności.

$$H_7 = (r_1(x), r_1(y), r_2(x), r_2(y), w_2(x), c_2, w_1(y), c_1).$$

Przyjmijmy, że wartości początkowe danych wynosiły $x=200$ i $y=150$. Transakcja T_1 odczytała wartości x i y, i postanowiła zmniejszyć wartość danej x o 40. Jednak zanim to nastąpiło transakcja T_2 również odczytała dane x i y, i postanowiła zwiększyć daną y o 30. Każda transakcja osobna zachowuje przyjętą zależność między danymi. Transakcja T_1 chce zostawić stan danych $x=160$ i $y=150$, a transakcja T_2 chce zostawić stan danych $x=200$ i $y=180$. Jednak współbieżne wykonanie tych transakcji pozostawia stan danych $x=160$ i $y=180$ niezgodny z przyjętą zależnością.

W systemie ORACLE uzyskanie pełnej uszeregowalności wymaga zakładania blokad przez zapytania poprzez stosowanie w zapytaniach klauzuli FOR UPDATE. Jednak wiąże się to z radykalnym ograniczeniem stopnia współbieżności systemu. Współbieżność jest w takim wypadku niższa niż w klasycznym algorytmie blokowania dwufazowego. Spowodowane jest to tym, że system ORACLE nie rozróżnia blokad do odczytu z intencją modyfikacji i zwykłych blokad do zapisu.

5. Podsumowanie

Zastosowanie systemowej synchronizacji współbieżnych transakcji miało rozwiązać problem błędów będących wynikiem współbieżnego dostępu do bazy danych. Jednak ze względu na konieczność znalezienia kompromisu między zapewnieniem poprawności danych, a wydajnością przetwarzania spowodowało, że niektóre komercyjne systemy baz danych nie gwarantują pełnej uszeregowalności transakcji. Przykładem jest system ORACLE, w którym zastosowano algorytmy synchronizujące zapewniające maksymalną efektywności procesu przetwarzania danych kosztem poprawności danych.

Powoduje to, konstrukcja oprogramowania aplikacyjnego musi uwzględniać problemy synchronizacji transakcji w sposób właściwy dla specyfiki danego zastosowania. Błędy będące wynikiem niepełnej synchronizacji transakcji realizowanej przez system zarządzania bazą danych, nieuwzględnione w kodzie aplikacji, mogą mieć trudne do oszacowania konsekwencje dla użytkowników aplikacji bazy danych. Tymczasem tematyka synchronizacji współbieżnych transakcji jest mało znana wśród programistów tworzących oprogramowanie aplikacyjne.

Literatura

1. ANSI X3.135-1992, *American National Standard for Information Systems – Database Language - SQL*, listopad 1992.
2. P. A. Bernstein, V. Hadzilacos, N. Goodman, *Concurrency Control and Recovery in Database Systems*, Addison_Wesley, 1987.
3. J. Gray, A. Reuter, *Transaction Processing: Concepts and Techniques*, Morgan Kaufmann 1993.
4. H. Berenson, P. Bernstein, J. Gray, J. Melton, E. O'Neil, P. O'Neil, *A critique of ANSI SQL Levels*, SIGMOD 1995, San Jose USA.
5. Dokumentacja techniczna Oracle8i.