

Tadeusz Pankowski  
Jarosław Ciszak  
Andrzej Sikorski  
Katedra Automatyki Robotyki i Informatyki  
Politechnika Poznańska  
pankowsk@sol.put.poznan.pl

## Zarządzanie transakcjami w SQL-owych systemach baz danych

**Streszczenie.** Omawiane są teoretyczne i praktyczne problemy zarządzania transakcjami w relacyjnych systemach baz danych zgodnych ze standardem SQL2. Przedstawiono problemy związane ze stosowaniem różnych poziomów izolacji - w szczególności ich wpływ na współbieżność i zachowanie spójności bazy danych. Omówiono elementy języka według standardu SQL2 związane ze sterowaniem transakcjami. Rozważania zilustrowano przykładami z systemu Sybase Anyware 5.0 i MS SQL Server.

### 1. Wprowadzenie

Wykonywanie dowolnego programu operującego na bazie danych znajdującej się pod nadzorem systemu zarządzania bazą danych (serwera bazy danych), może być postrzegane jako realizacja *procesu* złożonego z wielkiej liczby *transakcji*. Każda transakcja realizuje logicznie spójny zestaw operacji w bazie danych (na przykład wprowadzenie jednej pozycji faktury, odczytanie zestawu danych o pracowniku, obliczenie zarobków wszystkich pracowników). System musi gwarantować, że wykonanie transakcji zakończy się poprawnie albo, że - w przypadku awarii – żadna zmiana wykonana przez (częściowo zrealizowaną) transakcję nie będzie widoczna w bazie danych. W rzeczywistych systemach baz danych wykonywanych jest jednocześnie (współbieżnie) bardzo wiele transakcji, które jednocześnie próbują odczytywać i modyfikować te same dane. Na przykład, przy sprzedaży biletów kolejowych, z wielu stanowisk kasowych mogą być kierowane żądania rezerwacji tego samego miejsca w przedziale; w hurtowni w czasie przyjmowania i sprzedawania towarów, z wielu stanowisk komputerowych wysyłane są żądania aktualizowania stanów towarów lub wprowadzania nowych.

Zarządzanie transakcjami współbieżnymi stanowi jeden z podstawowych czynników wpływających na efektywność i funkcjonalność systemu informatycznego. Twórcy mechanizmów zarządzania transakcjami mają przede wszystkim na względzie *wiarygodność* systemu, tzn. to, aby system zawsze udostępniał dane pewne i to niezależnie od licznych konfliktów, jakie między transakcjami mogą zachodzić, jak i od możliwych awarii (sprzętu, zasilania czy oprogramowania systemowego). Zapewnienie pełnej wiarygodności (mimo, że i tak w praktyce nieosiągalne) wiąże się z ograniczeniem współbieżności, co w praktyce objawia się opóźnieniami w realizacji transakcji – jedne transakcje muszą czekać na zakończenie innych. Dopuszcza się więc możliwość „osłabienia” twardych rygorów wykonywania transakcji wprowadzając różne „poziomy izolacji”, dzięki czemu twórcy oprogramowania, a także użytkownicy interaktywni mogą świadomie godzić się na nie do końca wiarygodne dane zyskując w zamian zwiększenie szybkości realizacji swoich zadań. (mogą na przykład dopuścić możliwość czytania danych pracownika, które są w trakcie aktualizacji i część z nich jest w nowej, a część w starej wersji).

Użytkownicy profesjonalni, ale również doraźni nieprofesjoniści, powinni być świadomi faktu, że ich polecenia kierowane do bazy danych podlegają zasadom zarządzania transakcjami. Użyteczna jest wtedy znajomość tych środków języka SQL [2, 3], które mogą usprawnić korzystanie z bazy danych.

Układ niniejszej pracy jest następujący: W rozdz. 2. podano podstawowe pojęcia związane z transakcjami i historiami ich przetwarzania. Różne poziomy izolacji, przy których wykonywane są transakcje, a także problemy związane z wyborem każdego z nich, omówione są z teoretycznego punktu widzenia w rozdz. 3. O protokołach stosowanych przy zarządzaniu transakcjami współbieżnymi jest mowa w rozdz. 4. Rozdz. 5 omawia te cechy języka SQL, według standardu SQL2, które bezpośrednio związane są z wykonywaniem transakcji. W rozdz. 6 przytoczone są przykłady współbieżnego wykonywania transakcji w komercyjnych systemach zarządzania bazami danych.

## 2. Przetwarzanie transakcji

W SZBD stosuje się pojęcie *transakcji* jako jednostki operowania na bazie danych podlegającej sterowaniu i kontroli. Celem systemu zarządzania transakcjami jest takie sterowanie operacjami w bazie danych, aby były one wykonywane z możliwie wysokim współczynnikiem współbieżności i aby przeciwdziałać naruszeniu spójności bazy danych. Realizacja tego celu została osiągnięta za pomocą odpowiednich *protokołów zarządzania transakcjami*.

### 2.1. Transakcje

*Transakcja* definiowana jest jako ciąg operacji na wspólnej bazie danych [1]. Do operacji tych należą:

$r_i[x]$  - czytanie danej  $x$  przez transakcję  $T_i$ ;

$w_i[x]$  - zapisanie danej  $x$  przez transakcję  $T_i$ ;

$a_i$  - odrzucenie (wycofanie) transakcji  $T_i$ ;

$c_i$  - zatwierdzenie transakcji  $T_i$ .

Przyjmuje się, że transakcje i protokoły zarządzania transakcjami muszą spełniać postulat *ACID* (*Atomicity* - atomowość, *Consistency* - spójność, *Isolation* - odizolowanie, *Duration* - trwałość). Postulat ten rozumiany jest następująco:

1. *Atomowość* - oznacza, że każda transakcja stanowi pojedynczą i niepodzielną jednostkę przetwarzania (a także odtwarzania) - w transakcji nie ma więc podtransakcji. Każda transakcja jest bądź wykonana w całości, bądź też żaden jej efekt nie jest widoczny w bazie danych.
2. *Spójność* - transakcja rozpoczynając się w spójnym stanie bazy danych pozostawia bazę danych w stanie spójnym (tym samym lub innym). Jeśli transakcja narusza więzy spójności bazy danych, to zostaje odrzucona.
3. *Odizolowanie* - zmiany wykonywane przez transakcję nie zatwierdzoną nie są widziane przez inne transakcje (chyba, że przyjęty *poziom izolacji* na to zezwala).
4. *Trwałość* - zmiany w bazie danych dokonane przez transakcję zatwierdzoną są trwałe w bazie danych, tzn. nawet w przypadku awarii systemu musi być możliwość ich odtworzenia.

Warto wspomnieć, że w najnowszych koncepcjach systemów zarządzania bazami danych (systemy obiektowe, systemy wspomagające pracę grupową, systemy kooperacyjne), postulaty

*ACID* są modyfikowane. Na przykład w przypadku tzw. *transakcji długotrwałych* wprowadza się pojęcie podtransakcji.

Istotne z punktu widzenia stosowanego protokołu (algorytmu) zarządzania transakcjami jest przyjęcia pojęcia *poziomu konfliktowości* operacji, tzn. przyjęcia, jakie operacje są konfliktowe, a jakie nie. Przyjmuje się, że jeśli dwie operacje są konfliktowe, to druga (późniejsza) z nich może być wykonana dopiero wtedy, gdy zostanie zakończona (zatwierdzona lub odrzucona) transakcja, z której pochodzi pierwsza z tych operacji. Mówiąc o *współbieżnym* wykonywaniu operacji mamy na myśli wykonywanie niekonfliktowych operacji pochodzących z różnych transakcji i to w czasie, gdy obydwie te transakcje są aktywne. Transakcje, których dowolne operacje wykonywane są współbieżnie, nazywamy *transakcjami współbieżnymi*.

Przyjmujemy następujące poziomy konfliktowości operacji (im niższy poziom konfliktowości, tym większa jest współbieżność transakcji): dwie operacje  $p_i[x]$  i  $q_j[y]$  nazywamy operacjami *konfliktowymi* jeśli:

- 1)  $i \neq j$ , tzn. operacje pochodzą z różnych transakcji,
- 2)  $x = y$ , tzn. operacje dotyczą tej samej danej,

oraz zachodzi jeden z poniższych warunków (określający *poziom konfliktowości*):

- 3.0) obydwie są operacjami zapisu (*poziom 0*);
- 3.1) operacją zapisu jest operacja, która wystąpiła wcześniej (*poziom 1*);
- 3.2) co najmniej jedna z operacji jest operacją zapisu (*poziom 2*);
- 3.3) co najmniej jedna z operacji jest operacją zapisu, albo  $x$  i  $y$  są różne, ale spełniają ten sam warunek  $E$ , i co najmniej jedna z operacji jest operacją zapisu (w tym dołączania).

## 2.2. Historie przetwarzania transakcji

Z punktu widzenia analizy poprawności protokołów zarządzania transakcjami istotne jest analizowanie *historii przetwarzania transakcji*. Historia taka znana jest oczywiście dopiero po wykonaniu danego zbioru transakcji. Nam chodzi natomiast o sformułowanie wymagań odnośnie cech jakie powinny posiadać historie dopuszczane (generowane) przez protokoły. Jeśli jesteśmy w stanie stwierdzić, że stosując określony protokół *zawsze* otrzymamy przetwarzanie, którego historia posiada pożądane cechy, to możemy orzec, że protokół ten jest właściwy.

Niech  $\Sigma = \{T_1, T_2, \dots, T_n\}$  będzie zbiorem transakcji. Ciąg

$$H = (o_1, o_2, \dots, o_m)$$

operacji pochodzących z transakcji należących do zbioru  $\Sigma$  nazywamy *historią przetwarzania transakcji ze zbioru  $\Sigma$* . Jeśli operacja  $o$  poprzedza operację  $o'$  w historii  $H$ , to stosować będziemy zapis  $o < o'$ .

Mówimy, że transakcja  $T'$  czyta z transakcji  $T$  daną  $x$ , jeśli  $T$  jest ostatnią transakcją, która zapisała  $x$  i nie została odrzucona do czasu czytania  $x$  przez  $T'$ . Transakcja  $T'$  zapisuje w transakcji  $T$  daną  $x$ , jeśli  $T$  jest transakcją, która wczytała  $x$  i nie została odrzucona do czasu zapisu  $x$  przez  $T'$ .

### 3. Problemy przetwarzania przy różnych poziomach izolacji

Przyjęcie określonego poziomu izolacji wiąże się z określonymi problemami – zbyt niski poziom zapewni nam zwiększenie współczynnika współbieżności, ale może doprowadzić do niekorzystnych cech związanych z odtwarzalnością. Poziom zbyt wysoki może powodować nieuzasadnione opóźnianie transakcji. W kolejnych podpunktach omawiamy problemy związane z przyjęciem poszczególnych poziomów izolacji.

#### 3.1. Czytanie danych z transakcji nie zatwierdzonych (poziom izolacji 0)

Czytanie z transakcji nie zatwierdzonych możliwe jest przy przyjęciu poziomu konfliktowości 0, tzn. gdy za konfliktowe uważa się tylko parę operacji zapisu, a dwie operacje, z których jedna jest operacją odczytu nie są operacjami konfliktowymi. Można więc czytać dane zmieniane przez transakcję, która nie została jeszcze zatwierdzona. Tabela współbieżności operacji jest wtedy następująca (T oznacza, że operacje mogą być wykonywane współbieżnie – nie są konfliktowe, N – oznacza brak współbieżności, a więc konfliktowość):

	<i>Read</i>	<i>Write</i>
<i>Read</i>	T	T
<i>Write</i>	T	N

Przyjęcie tego rodzaju współbieżności operacji może doprowadzić do *przetwarzania nieodtwarzalnych* i z *kaskadą odrzuceń*. Zaletą jest jednak to, że wysoki jest współczynnik współbieżność transakcji.

#### Przetwarzania nieodtwarzalne

Przypuśćmy, że transakcja  $T$  zmieniła wartość danej  $x$ , następnie transakcja  $T'$  wczytała  $x$  i na podstawie jej wartości zmieniała wartość danej  $y$  w bazie danych. Przypuśćmy dalej, że transakcja  $T'$  została zatwierdzona, a po tym zdarzeniu powstała konieczność odrzucenia transakcji  $T$ . Należałoby więc wycofać wszystkie zmiany, jakie wprowadziła w bazie danych transakcja  $T$ , a także wszystkie konsekwencje tych zmian - w szczególności więc zmianę wartości danej  $y$ . Ta ostatnia operacja jest jednak niemożliwa, gdyż transakcja  $T'$ , która zmianę tę wykonała jest już zatwierdzona. Zaistniała więc sytuacja, w której baza danych jest *nieodtwarzalna*.

Powodem opisanej anomalii jest to, że transakcja czytająca dane z innej transakcji została zatwierdzona wcześniej niż transakcja, z której czytała. Aby sytuacji takiej uniknąć, należałoby czekać z zatwierdzeniem transakcji  $T'$  do czasu, aż zostanie zatwierdzona transakcja  $T$ . Przyjmujemy więc następującą definicję przetwarzania odtwarzalnego:

Historia  $H$  opisuje przetwarzanie *odtwarzalnym*, jeśli każda transakcja jest zatwierdzana po zatwierdzeniu wszystkich transakcji, z których czyta.

#### Przykład:

$$H1 : w_1[x] \ r_2[x] \ w_2[u] \ c_2 \ w_1[z] \ c_1$$

$H1$  opisuje przetwarzanie nie odtwarzalne, transakcja  $T_2$  czyta z  $T_1$ ,  $w_1[x] < r_2[x]$ , i  $T_2$  jest zatwierdzana przed zatwierdzeniem  $T_1$ ,  $c_2 < c_1$ . Operacja  $w_2[u]$  może oznaczać zapis wartości danej  $u$  wyznaczonej na podstawie wartości danej  $x$ .

## Przetwarzania z kaskadą odrzuceń

Przestrzeganie zasady odtwarzalności nie jest wystarczające. Mimo jej przestrzegania może dojść do sytuacji, gdy odrzucenie jednej transakcji pociągnie za sobą konieczność odrzucenia zależnej od niej (w jakimś sensie) innej transakcji, odrzucenie tej drugiej może spowodować konieczność odrzucenia trzeciej itd., co może prowadzić do *kaskady odrzuceń*.

Niech na przykład transakcja  $T'$  wczyta dane zmienione przez nie zatwierdzoną jeszcze transakcję  $T$ . Przypuśćmy, że transakcja  $T$  zostaje po tym zdarzeniu odrzucona. Konsekwencją tego jest także konieczność odrzucenia transakcji  $T'$ . Może to spowodować konieczność kaskadowego odrzucania wielu transakcji. Sytuacji tej można uniknąć, jeśli czytanie danych zmienionych przez transakcje jest dopuszczalne dopiero wtedy, gdy transakcje te zostały już zatwierdzone.

Historia  $H$  opisuje przetwarzania **bez kaskady odrzuceń**, jeśli transakcja czyta dane zapisane przez transakcje już zatwierdzone (a więc wymaga przyjęcia co najmniej konfliktowości poziomu 1).

### Przykład:

$$H2 : w_1[x] r_2[x] w_2[u] w_1[z] c_1 c_2.$$

$H2$  powstała z  $H1$  przez przesunięcie zatwierdzenia transakcji  $T_2$ , operacji  $c_2$ , na koniec historii.  $H2$  opisuje przetwarzanie odtwarzalne. Nie jest ono jednak przetwarzaniem bez kaskadowych odrzuceń, gdyż transakcja  $T_2$  czyta z  $T_1$ ,  $w_1[x] < r_2[x] < c_1$ , przed zatwierdzeniem transakcji  $T_1$ .

### 3.2. Zapisywanie danych w transakcjach nie zatwierdzonych (poziom izolacji 1)

Zapisywanie w transakcjach nie zatwierdzonych możliwe jest przy przyjęciu konfliktowości na poziomie 1, tzn. gdy za konfliktowe uważa się takie pary operacji, gdzie pierwsza jest operacją zapisu, a druga czytania, lub obydwie są operacjami zapisu. Dwie operacje, z których pierwsza jest operacją czytania nie są więc konfliktowe. Można zatem zapisywać dane, które zostały przeczytane przez transakcję jeszcze nie zatwierdzoną. Tabela współbieżności operacji jest wówczas następująca:

	<i>Read</i>	<i>Write</i>
<i>Read</i>	T	T
<i>Write</i>	N	N

Przyjęcie tego rodzaju współbieżności eliminuje anomalie związane z brakiem odtwarzalności i z kaskadą odrzuceń. Nie chroni jednak przed anomalią związaną z powtórным czytaniem tych samych danych.

### Anomalia powtórnego czytania

Przypuśćmy, że transakcja  $T$  czyta daną  $x$ , a następnie transakcja  $T'$  zapisuje nową wartość danej  $x$  i jest zatwierdzana. Jeśli teraz transakcja  $T$  ponownie przeczyta daną  $x$ , to okazuje się, że dana ta ma inną wartość. Transakcja  $T$  dysponuje więc dwiema różnymi wartościami tej samej danej. Może zdarzyć się też sytuacja, że transakcja  $T'$  usunie daną  $x$ . Wówczas przy próbie ponownego czytania, transakcja  $T$  ma informację, że danej  $x$  nie ma w bazie danych.

Historię  $H$  nazywamy historią **bez anomalii powtórnego czytania**, jeśli transakcja nie może zapisywać danych czytanych przez transakcje jeszcze nie zatwierdzone.

**Przykład:**

$$H3 : w_1[x] r_2[y] w_1[y] w_1[z] c_1 r_2[y] c_2$$

W  $H3$  występuje anomalia powtórnego czytania, gdyż  $r_2[y] < w_1[y] < r_2[y]$ .

### 3.3. Zakaz czytania i zapisywania danych w transakcjach nie zatwierdzonych (poziom izolacji 2)

Zakaz czytania i zapisywanie w transakcjach nie zatwierdzonych związany jest z przyjęciem konfliktowości na poziomie 2, tzn. gdy za konfliktowe uważa się takie pary operacji, gdzie co najmniej jedna jest operacją zapisu. Zatem niekonfliktowe są tylko operacje czytania. Jeśli więc transakcja nie zatwierdzona czytała jakąś daną, to dana ta może być tylko czytana przez inną transakcję. Jeśli natomiast transakcja nie zatwierdzona zapisała jakąś daną, to nie można jej ani odczytać ani tym bardziej zapisać, dopóki transakcja ta nie zostanie zatwierdzona. Tabela współbieżności operacji jest więc wówczas następująca:

	<i>Read</i>	<i>Write</i>
<i>Read</i>	T	N
<i>Write</i>	N	N

Zgodnie z tymi założeniami, dana zablokowana do odczytu jest dostępna do odczytu również innym transakcjom. Dana zablokowana do odczytu lub zapisu jest dostępna do zapisu dopiero po zatwierdzeniu transakcji. Przyjęcie tego rodzaju współbieżności eliminuje anomalie powtórnego czytania. Może jednak wystąpić tzw. *zjawisko fantomu*.

#### Fantomy

Przypuśćmy, że transakcja  $T$  wczytała z tabeli  $R$  zbiór rekordów spełniających warunek  $E$ . Następnie transakcja  $T'$  dołączyła do  $R$  nowy rekord  $r$  spełniający warunek  $E$  i została zatwierdzona. Jeśli  $T$  ponownie odwoła się do rekordów tabeli  $R$  spełniających warunek  $E$ , to okaże się, że tym razem zbiór ten jest inny.

Problem ten wymaga wprowadzenia specjalnego blokowania rekordów. Mianowicie *blokowania według warunku*, tzn. jeśli zablokowane są rekordy spełniające warunek  $E$ , to zakładana jest blokada na dołączanie rekordów spełniających ten warunek, a także na operacje zapisu, w wyniku których mogłyby się zmienić zbiór rekordów spełniających warunek  $E$ . Można to rozumieć jak zablokowanie rekordu *EOF* (*end-of-file*) i jako zablokowanie indeksu klucza głównego na operacje modyfikacji rekordów spełniających warunek  $E$ .

**Przykład:**

$$H4 : w_1[x] r_2[u] w_1[z] c_1 r_2[u] c_2$$

W historii  $H4$  może pojawić się problem fantomów. Jeśli bowiem operacja  $r_2[u]$  wczytuje zbiór rekordów spełniających warunek  $E$ , a operacja  $w_1[z]$  spowoduje, że zbiór takich rekordów ulegnie zmianie (na przykład tak zostaną zmienione pola rekordu  $z$ , że po zmianie rekord  $z$  będzie spełniał warunek  $E$ ), to powtórne wykonanie operacji  $r_2[u]$  zwróci inny zbiór rekordów.

### 3.4. Historie szeregowe (poziom izolacji 3)

Historia  $H$  opisuje *przetwarzanie szeregowe* (jest *historią szeregową*), jeśli dla dowolnych dwóch transakcji  $T$  i  $T'$ , *wszystkie operacje* transakcji  $T$  poprzedzają wszystkie operacje transakcji  $T'$ , lub wszystkie operacje transakcji  $T'$  poprzedzają wszystkie operacje transakcji  $T$ . W przetwarzaniu szeregowym nie ma więc "przeplatających" się operacji pochodzących z różnych transakcji. Tym samym nie ma żadnej współbieżności, ale dzięki temu nie może dojść do żadnych konfliktów między transakcjami. Z punktu widzenia realnych zastosowań takie przetwarzania są jednak bezużyteczne.

Historia  $H$  opisuje *przetwarzanie szeregowe* (jest *historią szeregową*), jeśli dla dowolnych dwóch transakcji  $T$  i  $T'$  oraz dla każdej pary  $(o, o')$  operacji konfliktowych pochodzących odpowiednio z  $T$  i z  $T'$ , zawsze  $o < o'$  lub  $o' < o$ . Zauważmy, że każda przetwarzanie szeregowe jest równoważne pewnemu przetwarzaniu szeregowemu (z punktu widzenia wyniku przetwarzania). Daje ponadto możliwość współbieżnego wykonywania pewnych operacji. Tak rozumiana szeregowość jest zgodna z definicją konfliktowości na poziomie 3. Jest to sytuacja najbardziej bezpieczna, żadna z omówionych anomalii nie może wówczas wystąpić.

Dla każdej historii  $H$  można zbudować *graf szeregowości*  $GSz(H)$ . Zbiorem wierzchołków w takim grafie jest zbiór  $\Sigma$  wszystkich transakcji, a krawędź  $T \rightarrow T'$  należy do zbioru krawędzi grafu wtedy i tylko wtedy, gdy istnieją operacje konfliktowe  $o$  i  $o'$ , pochodzące odpowiednio z  $T$  i  $T'$  takie, że  $o < o'$ . Historia  $H$  jest szeregowość wtedy i tylko wtedy, gdy jej graf szeregowości  $GSz(H)$  nie zawiera cykli.

## 4. Protokoły zarządzania transakcjami współbieżnymi

Zarządzaniem transakcjami zajmuje się wyspecjalizowany moduł *planisty* [1]. Planista związany jest z każdym menadżerem danych ( $MD$ ) zarządzającym danymi określonej lokalnej bazy danych na określonym węźle sieci. Zatem tylu jest planistów ile lokalnych baz danych w systemie. Nie ma natomiast żadnego centralnego planisty, który koordynowałby dostęp do lokalnych baz danych. (Taki centralny nadzorca jest natomiast potrzebny do rozwiązywania globalnych zakleszczeń.) Każda transakcja może kierować swoje operacje odczytu/zapisu do wielu lokalnych baz danych. Planista przyjmując operację  $o$  ma do dyspozycji następujące działania:

- 1) przekazanie operacji  $o$  do wykonania menadżerowi danych,
- 2) umieszczenie operacji  $o$  w kolejce (jeśli znajduje się ona w konflikcie z inną operacją i należy poczekać na zakończenie tej operacji), lub zatwierdzenie transakcji, od której operacja ta pochodzi - wykrywanie takich konfliktów dostępu realizowane jest za pomocą analizy (poszukiwania cykli) w dynamicznie tworzonym grafie szeregowości;
- 3) odrzucenie operacji i tym samym całej transakcji, jeśli na przykład konieczne jest to z punktu widzenia rozwiązywania konfliktów prowadzących do zakleszczeń.

Opisana powyżej strategia działania planisty jest w komercyjnych SZBD realizowana za pomocą tzw. *protokołu blokowania dwufazowego (B2F)*. Planista realizujący tę strategię nazywany jest *planistą blokowania dwufazowego (planistą B2F)*. Następujące reguły definiują tę strategię, przy czym trzecia z nich (*reguła B2F*) odgrywa zasadniczą rolę:

1. Jeśli operacja  $p_i[x]$  może być wykonana, to planista zakłada blokadę (odpowiednio do odczytu lub do zapisu) na daną  $x$  dla transakcji  $T_i$  i operację tę przekazuje  $MD$  do wykonania; jeśli operacja ta nie może być wykonana, to umieszczana jest w kolejce.
2. Zdjęcie założonej blokady może nastąpić najwcześniej wtedy, gdy  $MD$  powiadomi planistę o zakończeniu wykonywania operacji.
3. (Reguła  $B2F$ ) Jeśli nastąpiło zdjęcie jakiegokolwiek blokady założonej dla transakcji  $T$ , to dla tej transakcji nie można już założyć żadnej innej blokady.

Ostatnia z powyższych reguł uzasadnia nazwę "blokowanie dwufazowe", gdyż w procesie wykonywania transakcji można wyróżnić dwie fazy: fazę *zakładania blokad* i fazę *zdejmowania blokad*. W zależności od tego, kiedy planista zdejmuje blokady założone na dane dla poszczególnych transakcji, możemy mówić o blokowaniu z różnymi *poziomami izolacji*:

0.  $B2F$  z *poziomem izolacji 0*: zdejmowanie blokad następuje natychmiast po zakończeniu operacji w sposób następujący: dana zablokowana na czas realizacji operacji *Read* jest natychmiast odblokowywana po zakończeniu tej operacji. Natomiast dana zablokowana na czas realizacji operacji *Write* zmienia blokadę na *non-exclusive*, to znaczy może na niej być realizowana operacja *Read*, ale nie operacja *Write*. Blokada dla operacji *Write* zdejmowana jest całkowicie dopiero po zatwierdzeniu transakcji.
1.  $B2F$  z *poziomem izolacji 1*: dana zablokowana na czas realizacji operacji *Read* jest natychmiast odblokowywana po zakończeniu tej operacji. Natomiast dana zablokowana na czas realizacji operacji *Write* odblokowywana jest dopiero po zatwierdzeniu transakcji.
2.  $B2F$  z *poziomem izolacji 2*: zdejmowanie blokad następuje dopiero po zatwierdzeniu transakcji.
3.  $B2F$  z *poziomem izolacji 3*: dodatkowo utrzymywane są blokady sterowane przez warunki, tzn. zablokowana jest możliwość takich aktualizacji (w tym dołączania), które wpływają na rekordy spełniające określony warunek.

Wadą planisty  $B2F$  jest to, że może prowadzić do powstania zakleszczeń. Zakleszczenie pojawia się wtedy gdy w *grafie oczekiwań*  $GO(H)$  wystąpi cykl. Jeśli  $\Sigma$  jest zbiorem transakcji, a  $H$  jest historią wykonywania tych transakcji, to zbiór wierzchołków grafu  $GO(H)$  składa się z elementów zbioru  $\Sigma$ , a krawędź  $T \rightarrow T'$  należy do grafu, gdy założenie jakiejś blokady dla transakcji  $T$  wymaga oczekiwania na zdjęcie blokady założonej dla transakcji  $T'$ . Wystąpienie cyklu w grafie oczekiwań oznacza, że powstało zakleszczenie. Aby je rozwiązać należy odrzucić jedną z transakcji powodującą cykl. Planista musi więc dynamicznie tworzyć i analizować graf oczekiwań. Problem staje się bardziej złożony w systemie rozproszonym. Wówczas co prawda każdy z planistów jest w stanie poradzić sobie z zakleszczeniami lokalnymi, ale nie jest w stanie kontrolować zakleszczeń globalnych. Do kontroli i rozwiązywania zakleszczeń globalnych konieczny jest centralny nadzorca, który otrzymuje informacje potrzebne mu do tworzenia globalnego grafu oczekiwań od planistów lokalnych.

Inną metodą zarządzania transakcjami współbieżnymi jest metoda oparta na wykorzystaniu *znaczników czasowych*. Podstawową zaletą tej metody jest to, że nie prowadzi do powstawania zakleszczeń. Nie jest jednak powszechnie wykorzystywana w komercyjnych SZBD. Jeszcze inną klasę stanowią tzw. *optymistyczne* strategie zarządzania transakcjami. Dopuszcza się w nich maksymalne zrównoleglenie operacji, a dopiero w fazie zatwierdzania transakcji sprawdzane jest czy nie dochodzi do konfliktów i wówczas niektóre z transakcji są odrzucane. Metoda ta ma zastosowanie przede wszystkim w systemach czasu rzeczywistego, gdzie czas wykonywania transakcji jest elementem krytycznym [3,5].

## 5. Zarządzanie transakcjami wg standardu SQL2

Podstawowym zadaniem systemu zarządzania transakcjami jest takie ich wykonywanie, które zapewni zachowanie spójności bazy danych. Spójność (integralność) oznacza poprawność, niesprzeczność danych w bazie danych [4]. Standard SQL [2] zawiera środki do definiowania więzów integralności (ang. *integrity constraints*), np. w obrębie zdania CREATE TABLE. Każda próba naruszenia zdefiniowanych więzów integralności (podczas modyfikacji bazy danych) zostaje udaremniona - operacja ją podejmująca jest odrzucana, a baza danych pozostaje nie zmieniona.

Środki definiowania więzów integralności:

- UNIQUE - dla określenia, że kolumna lub zestaw kolumn ma mieć unikalną wartość w tabeli (jest *kluczem alternatywnym*);
- PRIMARY KEY - specjalny przypadek UNIQUE dla zdefiniowania klucza głównego, dodatkowo wymaga się, aby wszystkie wartości klucza głównego były różne od *NULL*;
- FOREIGN KEY - dla zdefiniowania zależności referencyjnych (zależności odniesień);
- CHECK - dla określenia, że kolumna lub zestaw kolumn (z jednej lub kilku tabel) mają mieć wartości spełniające określony warunek.

Do środków definiowania integralności należy również określenie typu kolumny (np. DECIMAL, CHARACTER) oraz fraza WITH CHECK OPTION w CREATE VIEW (używana w przypadku modyfikowalnych perspektyw).

Tryb sprawdzania więzów integralności może być:

- natychmiastowy (ang. *immediate*) - zaraz po zakończeniu operacji SQL-owej,
- opóźniony (ang. *deferred*) - po zakończeniu całej transakcji.

W standardzie SQL przyjmuje się następujące pojęcia związane z zarządzaniem transakcjami:

1. Transakcja jest ciągiem operacji stanowiących jedną całość z punktu widzenia odtwarzania (*atomowość transakcji*). Każda transakcja kończy się operacją COMMIT (normalne zakończenie) - zatwierdzenie transakcji, lub ROLLBACK (nienormalne zakończenie) - odrzucenie transakcji.
2. Każda transakcja przekształca spójny stan bazy danych w stan spójny (spójność transakcji).
3. Aktualizacje wykonane przez transakcję  $T_1$  nie są widoczne dla żadnej innej transakcji  $T_2$  dopóty, dopóki  $T_1$  nie wykona COMMIT (izolacja transakcji). Wyjątkiem jest przetwarzanie  $T_2$  z poziomem izolacji 0, READ UNCOMMITTED.
4. Zmiany wykonane przez zatwierdzoną transakcję są trwałe w bazie danych nawet w przypadku awarii (trwałość zmian).

Transakcja ma więc, omawianą poprzednio, właściwość *ACID*. Odtwarzanie realizowane jest przez system bazy danych po awaryjnym przerwaniu działania systemu. Ma na celu powrót bazy danych do ostatniego spójnego stanu. Przerwane transakcje mogą wymagać powtórnego wykonania.

Transakcja rozpoczyna się w chwili wydania przez agenta odpowiedniego polecenia "inicjującego transakcję". Większość wyrażeń SQL inicjuje transakcję.

Określenie charakterystyk transakcji (SET TRANSACTION i SET CONSTRAINTS):

SET TRANSACTION *lista-opcji*

*opcje:*

tryb dostępu : READ ONLY, READ WRITE

rozmiar obszaru diagnostyk : DIAGNOSTICS SIZE *n*

poziom izolacji : ISOLATION LEVEL *izolacja*

*izolacja:* SERIALIZABLE (domyślna),

REPEATABLE READ,

READ COMMITTED

READ UNCOMMITTED

SET CONSTRAINTS {*lista-więzów* | ALL}

{DEFERRED| IMMEDIATE}

ustala tryb sprawdzania więzów integralności na natychmiastowy (IMMEDIATE) lub opóźniony (DEFERRED).

Przyjęcie określonego poziomu izolacji może być źródłem problemów omawianych w rozdz.2. W poniższej tabelicy "T" oznacza, że dany problem występuje przy rozważanym poziomie izolacji - "N", że nie występuje.

<i>Poziom izolacji</i>	<i>Czytanie danych nie zatwierdzonych</i>	<i>Anomalia powtórnego czytania</i>	<i>Fantomy</i>
0 : READ UNCOMMITTED	T	T	T
1 : READ COMMITTED	N	T	T
2 : REPEATABLE READ	N	N	T
3 : SERIALIZABLE	N	N	N

Problemy związane z przetwarzaniem bazy danych przy różnych poziomach izolacji zilustrujemy teraz przykładami z systemu Sybase Anyware 5.0 i MS SQL Server 6.0.

## 6. Wykonywanie transakcji współbieżnych w komercyjnych systemach baz danych

Wszystkie przykładowe transakcje podane w tym rozdziale zawierają składnię zgodną z językiem SQL Watcom i systemem Sybase Anyware 5.0, gdyż w tym przypadku występuje duża zgodność ze standardem SQL2.

Przypuśćmy, że w bazie danych istnieje tabela Towar:

Nazwa	Cena	Stan
200MMX	320	20
233MMX	370	50

Przy każdym poziomie izolacji operacje zapisu są konfliktowe. W tab.1 pokazano historię przetwarzania dwóch transakcji operujących na tabeli Towary i próbujących jednocześnie aktualizować ten sam zbiór wierszy tabeli. Występuje wówczas konflikt zapisu.

Tab. 1. Historia przetwarzania transakcji dla dowolnego poziomu izolacji przy konflikcie zapisu.

Transakcja 1 (T1)	Transakcja 2 (T2)	Uwagi
set transaction isolation level [0   1   2   3]	set transaction isolation level [0   1   2   3]	Dowolny poziom izolacji
update Towar set Cena=300 where Nazwa='200MMX'		
	update Towar set Cena=290 where Nazwa='200MMX'	T2 czeka na zakończenie T1.
commit		
		Wykonanie operacji „update” transakcji T2
	commit	

W tab.2. pokazano historię przetwarzania, gdy aktualizacje dotyczą rozłącznych zbiorów wierszy. Nie ma wówczas konfliktu zapisu.

Tab. 2. Historia przetwarzania transakcji, gdy nie ma konfliktu zapisu

Transakcja 1 (T1)	Transakcja 2 (T2)	Uwagi
set transaction isolation level [0   1   2   3]	set transaction isolation level [0   1   2   3]	Dowolny poziom izolacji
update Towar set Cena=300 where Nazwa='200MMX'		
	update Towar set Cena=350 where Nazwa='233MMX'	Wykonanie operacji T2, T1 i T2 aktualizują różne wiersze.
commit		
	commit	

W niektórych serwerach baz danych jednostką blokowania dostępu jest cała tabela, a nie poszczególne jej wiersze. Wówczas operacje „update” zamieszczone w powyższym przykładzie są konfliktowe i nie mogą być wykonane współbieżnie. Tak jest na przykład (w ogólnym przypadku) w systemie MS SQL Server 6.0.

W dalszym ciągu rozpatrzmy historie przetwarzania przy różnych poziomach izolacji. Ograniczymy się przy tym tylko do tych operacji, które są zgodne ze standardem SQL2.

### **Poziom izolacji 0 (READ UNCOMMITTED)**

Przy poziomie izolacji 0 możliwe jest czytanie danych zmienionych przez transakcje jeszcze nie zatwierdzone. Mówi się wówczas o „brudnym czytaniu” (ang. *dirty read*). Dopuszczenie takiego czytania bardzo zwiększa współbieżność przetwarzania, ale jednocześnie może doprowadzić do udostępniania nieprawdziwych danych z bazy danych (co ilustruje przykład z tab. 3).

Tab. 3. Historia przetwarzania transakcji na poziomie izolacji 0: zapis – odczyt

Transakcja 1 (T1)	Transakcja 2 (T2)	Uwagi
set transaction isolation level 0	set transaction isolation level 0	
update Towar set Cena=300 where Nazwa='200MMX'		
	select Cena from Towar where Nazwa='200MMX'	T2 czyta zmienioną cenę
rollback		T1 wycofuje zmiany. T2 posiada niepoprawną informację o cenie.
	commit	

Konsekwencje tego mogą być bardzo poważne, jeśli na podstawie takich nieprawdziwych danych zostaną wprowadzone zmiany w bazie danych, te zmiany mogą powodować kolejne zmiany itd. Zerowy poziom izolacji może więc być stosowany tylko w takich transakcjach, o których wiemy, że nawet w przypadku błędnych danych nie spowodują poważnych negatywnych konsekwencji. Możemy go stosować na przykład dla transakcji, których zadaniem jest udzielanie informacji z bazy danych. Prawdopodobieństwo wystąpienia takiej sytuacji, jaką ilustruje nasz przykład jest bowiem w praktyce niewielkie.

Historia przetwarzania pokazana w tab. 1. jest niedopuszczalna dla żadnych poziomów izolacji wyższych niż 0, gdyż dla nich wszystkich dwie operacje pochodzące z różnych transakcji i dotyczące tych samych danych (wierszy tabeli), i z których pierwsza jest operacją zapisu, są konfliktowe.

### *Poziom izolacji 1 (READ COMMITTED)*

Cechą charakterystyczną tego poziomu izolacji jest to, że możliwe jest aktualizowanie przez transakcję T2 danych wczytanych przez nie zakończoną jeszcze transakcją T1. Po powtórnym odwołaniu się do tych samych danych transakcja T1 możemy uzyskać sprzeczne informacje. Ilustruje to przykład w tab. 4.

Tab. 4. Historia przetwarzania transakcji na poziomie izolacji 1: odczyt-zapis

Transakcja 1 (T1)	Transakcja 2 (T2)	Uwagi
set transaction isolation level 1	set transaction isolation level 1	
select Cena, Stan from Towar where Nazwa='200MMX'		T1 czyta cenę i stan towaru
	update Towar set Cena=310 where Nazwa='200MMX'	T2 zmienia cenę wczytaną przez T1
select sum(Cena*Stan) from Towar where Nazwa='200MMX'		T1 czeka na zakończenie T2
	commit	
		Wykonanie „select” dla T1. Wynik sprzeczny z poprzednim „select”-em
commit		

### Poziom izolacji 2 (**REPEATABLE READ**)

W przypadku poziomu izolacji 2 mamy zagwarantowane, że przy ponownym odwołaniu się przez transakcję do tych samych danych, dostajemy identyczne informacje. Uzyskuje się to przez uniemożliwienie aktualizowania danych wczytanych przez transakcję, która nie została jeszcze zakończona. Przetwarzanie transakcji przedstawione w tab. 4 będzie obecnie miało historię pokazaną w tab. 5.

Tab. 5. Historia przetwarzania transakcji na poziomie izolacji 2: odczyt-zapis-odczyt

Transakcja 1 (T1)	Transakcja 2 (T2)	Uwagi
set transaction isolation level 2	set transaction isolation level 2	
select Cena, Stan from Towar where Nazwa='200MMX'		T1 czyta cenę i stan towaru
	update Towar set Cena=310 where Nazwa='200MMX'	T2 czeka na zakończenie T1.
select sum(Cena*Stan) from Towar where Nazwa='200MMX'		T1 oblicza wartość towaru
commit		
		Wykonanie „update” dla T2.
	commit	

Poziom izolacji 2 zabezpiecza przed modyfikacją wczytanych danych, ale nie przed dołączaniem nowych wierszy. Po odczytaniu danych przez transakcję T1, transakcja T2 może dołączyć nowy wiersz spełniający warunek nałożony na odczytywany zbiór wierszy. Przypuśćmy, że T2 po dołączeniu wiersza zostaje zatwierdzona, a T1 ponownie odwołuje się do zbioru wierszy spełniających początkowy warunek. Może wówczas uzyskać informacje sprzeczne z pierwszym czytaniem. Przyczyną niezgodności jest nowo dołączony wiersz zwany „fantomem”. Sytuację tę ilustruje historia przetwarzania podana w tab. 6.

Tab. 6. Historia przetwarzania transakcji na poziomie izolacji 2: odczyt-dołączenie-odczyt

Transakcja 1 (T1)	Transakcja 2 (T2)	Uwagi
set transaction isolation level 2	set transaction isolation level 2	
select Cena, Stan from Towar where Nazwa='200MMX'		T1 czyta cenę i stan towaru
	insert into Towar values ('200MMX',250,10)	T2 dołącza nowy wiersz „fantom”
	commit	
select sum(Cena*Stan) from Towar where Nazwa='200MMX'		T1 oblicza wartość towaru. Wynik jest sprzeczny z poprzednim „select”-em
commit		

### Poziom izolacji 3 (**SERIALIZABLE**)

Przed niespodziewanym pojawieniem się fantomów chroni poziom izolacji 3. Przy tym poziomie izolacji przetwarzanie z tab.6 miałyby historię podaną w tab. 7.

Tab. 7. Historia przetwarzania transakcji na poziomie izolacji 3: odczyt-dołączenie-odczyt

Transakcja 1 (T1)	Transakcja 2 (T2)	Uwagi
set transaction isolation level 3	set transaction isolation level 3	
select Cena, Stan from Towar where Nazwa='200MMX'		T1 czyta cenę i stan towaru
	insert into Towar values ('200MMX',250,10)	T2 czeka na zakończenie T1
select sum(Cena*Stan) from Towar where Nazwa='200MMX'		T1 oblicza wartość towaru.
commit		
		Wykonanie „insert” przez T2
	commit	

Nie wszystkie serwery baz danych rozróżniają 2. i 3. poziom izolacji. Na przykład MS SQL Server utożsamia te dwa poziomy przyjmując, że operacja dołączania jest zwykłą operacją aktualizacji.

## LITERATURA

- [1] Bernstein P.A., V. Hadzilacos, N. Goodman, *Concurrency Control and Recovery in Database Systems*, Addison-Wesley, Reading Massachusetts, 1987.
- [2] Date C.J., Darwen H., *A Guide to the SQL Standard, Third Edition*, Addison-Wesley, Reading Massachusetts, 1994.
- [3] Elmasri R., Navathe S. B., *Fundamentals of Database Systems*, The Benjamin/Cummings, Redwood City, 1994.
- [4] Pankowski T., *Podstawy baz danych*, PWN, Warszawa, 1992.
- [5] Pankowski T., *Zarządzanie transakcjami w SQL-owych bazach danych czasu rzeczywistego*, Informatyka 9, 1995, ss. 21-26.