3.14.3. Izolowanie transakcji[1]

Wiemy, że transakcja musi być izolowana, czyli próba odczytania danych z tabeli, na której przeprowadzana jest transakcja przez innego użytkownika, zakończy się niepowodzeniem. Dopiero pojawienie się polecenia *COMMIT* powoduje właściwą modyfikację danych, zdjęcie blokad z danych i umożliwia dostęp do nich innym użytkownikom.

W zależności od istniejącego na serwerze bazodanowym poziomu izolowania transakcji może pojawić się jeden z następujących problemów:

- Utrata aktualizacji (**ang. *lost or buried updates***)- problem pojawi się, gdy dwie transakcje modyfikują te same dane. Żadna z transakcji nie wie, że druga transakcja korzysta z tych samych danych. W efekcie transakcja, która zakończy się później, zmodyfikuje dane, niszcząc zmiany dokonane przez transakcję, która zakończyła się wcześniej. Domyślnie skonfigurowany serwer nie dopuści do utraty aktualizacji.
- Brudne odczyty **(ang. *dirty reads***)- problem pojawi się, gdy jedna transakcja dokonuje zmian danych, a druga w tym czasie odczytuje te dane. W efekcie transakcja odczytuje dane, które nie zostały jeszcze zatwierdzone i mogą zostać wycofane. Domyślnie skonfigurowany serwer nie dopuści brudnych odczytów.
- Niepowtarzalne odczyty (**ang. *non-repetable reads***)- występują, gdy powtórzenie w ramach transakcji tego samego odczytu daje inny wynik. Może to być spowodowane tym, że po pierwszym odczycie (a nie po zakończeniu transakcji) zostaną zdjęte blokady założone na odczytywane dane. Niezablokowane dane mogą zostać zmienione przez inne działania, a powtórne ich odczytanie da inny wynik. Domyślnie skonfigurowany serwer dopuszcza niepowtarzalne odczyty.

Odczyty widma (**ang. *phantom reads***)- występują, gdy pomiędzy dwoma odczytami tych samych danych w ramach jednej transakcji zmieni się liczba odczytywanych wierszy (na przykład w wyniku wykonania w międzyczasie instrukcji *INSERT* lub *DELETE* na tych danych). Domyślnie skonfigurowany serwer dopuszcza odczyty widma.

3.14.4. Poziomy izolowania transakcji

Blokad używamy w celu zabezpieczenia danych w trakcie współbieżnego wykonywania transakcji. Pozwala to na wykonywanie transakcji w pełnej izolacji od innych transakcji i zapewnia przez cały czas poprawność danych w bazie.

Dzięki temu możliwe jest wykonywanie kilku transakcji w tym samym czasie, tak jakby były wykonywane jedna po drugiej, czyli szeregowo.

Transakcje nie zawsze wymagają pełnej izolacji. Możemy wpływać na sposób zakładania blokad przez serwery bazodanowe, zmieniając poziom izolacji transakcji.

Poziom izolacji określa stopień, do którego dana transakcja musi być izolowana od innych transakcji. Najniższy poziom izolacji to maksymalny stopień współbieżności, ale jest on okupiony najmniejszym stopniem spójności danych. Natomiast wyższy stopień izolacji

[1] Projektowanie i administrowanie bazami danych, kwalifikacja inf.03, Helion, Jolanta Pokorska

transakcji zwiększa poprawność danych, ale zmniejsza stopień współbieżności. Najwyższy poziom izolacji gwarantuje najwyższy poziom spójności danych kosztem ograniczenia do minimum współbieżności.

Serwery bazodanowe pozwalają ustawiać na poziomie serwera, baz danych lub pojedynczych sesji poziom izolowania transakcji.

Standard SQL3 definiuje cztery poziomy izolacji transakcji:

- *Read Uncommitted,*
- *Read Committed,*
- *Repeatable Read,*
- *Serializable.*

Read Uncommitted

Read Uncommitted to tryb niezatwierdzonego odczytu. Jest to najniższy poziom izolacji transakcji. Odczyt danych powoduje założenia blokady współdzielonej. Tylko fizycznie uszkodzone dane nie będą odczytywane. Na tym poziomie pojawiają się brudne odczyty, niepowtarzalne odczyty i odczyty widma. Nie występuje jedynie problem z utratą aktualizacji. Ten tryb może być stosowany do odczytywania danych, o których wiemy, że w czasie odczytywania nie będą modyfikowane.

Ćwiczenie 3.18

W ramach jednej sesji w oknie edytora SQL rozpoczniemy transakcję, której zadaniem będzie zmodyfikowanie danych klienta w tabeli *Klient.*

```
BEGIN TRANSACTION;

UPDATE Klient

SET ulica = 'Mickiewicza' , nr_domu=94

WHERE nazwisko='Nowak' AND imie='Andrzej';
```

Nie zamykając tej transakcji, w ramach kolejnej sesji, jako inny użytkownik (nowe okno edytora SQL), zmienimy poziom izolowania transakcji i spróbujemy odczytać dane modyfikowane przez pierwszego użytkownika:

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMITTED;

GO

SELECT miejscowosc, ulica, nr_domu

FROM klient

WHERE nazwisko='Nowak' AND imie='Andrzej';
```

Mimo że pierwszy użytkownik nie zamknął transakcji, udało się odczytać zmienione dane klienta. Kończąc przykład, należy zamknąć transakcję bez zatwierdzania zmian.

Read Committed

Read Committed to tryb odczytu zatwierdzonego. Jest to domyślny poziom izolacji stosowany przez MS SQL Server. Podczas odczytu danych zostanie nałożona na nie blokada współdzielona. Założona blokada eliminuje brudne odczyty. Natomiast nadal występują niepowtarzalne odczyty oraz odczyty widma.

Ćwiczenie 3.19

Otwieramy dwa okna edytora SQL. W pierwszym oknie ustawiamy tryb odczytu zatwierdzonego, rozpoczynamy transakcję i odczytujemy klienta:
SET TRANSACTION ISOLATION LEVEL READ COMMITTED

BEGIN TRANSACTION;

SELECT miejscowosc, ulica, nr_domu

FROM Klient

WHERE nazwisko=’Nowak AND imie=’Andrzej’;

Pozostawiamy otwartą transakcję i w drugim oknie zmieniamy adres klienta:

UPDATE Klient

SET ulica = ‘Sienkiewicza’, nr_domu=21

WHERE nazwisko=’Nowak’ AND imie=’Andrzej’;

Ponownie wracamy do okna pierwszego. W ramach tej samej transakcji drugi raz odczytujemy dane klienta:

SELECT miejscowosc, ulica, nr_domu

FROM Klient

WHERE nazwisko=’Nowak’ AND imie=’Andrzej’;

COMMIT;

Adres ponownie odczytany jest inny niż odczytany wcześniej. Wystąpił efekt niepowtarzalnych odczytów.

Repeatable Read

Repeatable Read to tryb powtarzalnego odczytu. Założona na dane blokada współdzielona jest utrzymywana aż do zakończenia całej transakcji. Dzięki temu, że inny proces nie może zmodyfikować odczytywalnych danych, zostały wyeliminowane niepowtarzalne odczyty. Natomiast nadal występują odczyty widma.

Ćwiczenie 3.20

Otwieramy dwie sesje edytora SQL. W pierwszej ustawiamy tryb powtarzalnego odczytu, rozpoczynamy transakcję i odczytujemy nazwiska klientów mieszkających w Poznaniu:

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

BEGIN TRANSACTION;

SELECT nazwisko, imie

FROM Klient

WHERE miejscowosc=’Poznań’;

Pozostawiamy otwartą transakcję i w drugiej sesji zmieniamy na Poznań miejsce zamieszkania jednego z klientów, który nie mieszkał w Poznaniu:

UPDATE Klient

SET miejscowosc = ‘Poznań’

WHERE nazwisko=’Kowalski’ AND imie=’Jan’;

Ponownie wracamy do pierwszej sesji. W ramach tej samej transakcji drugi raz odczytujemy nazwiska klientów mieszkających w Poznaniu:

SELECT nazwisko, imie

FROM Klient

WHERE miejscowosc=’Poznań’;

Tym razem liczba odczytywanych wierszy uległa zmianie. Jest ich więcej, Pojawił się wiersz widmo.

Zmiana Danych w drugiej sesji była możliwa, ponieważ dotyczyła danych , które nie były odczytywane w transakcji otwartej w pierwszej sesji.

Na zakończeni e przykładu należy wykonać instrukcję zamknięcia transakcji COMMIT.

Sytuacja uległa zmianie, gdy w drugiej sesji będziemy próbowali zmienić dane odczytywanie w pierwszej sesji.

Ćwiczenie 3.21

Otwieramy dwie sesje edytora SQL. W pierwszej ustawiamy tryb powtarzalnego odczytu, rozpoczynamy transakcję i odczytujemy nazwiska klientów mieszkających w Gdańsku.

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

BEGIN TRANSACTION;

SELECT nazwisko, imie

PROM Klient

WHERE miejscowosc = ‘Gdańsk’;

Pozostawiamy otwartą transakcję i w drugiej sesji zamieniamy na Kraków miejsce zamieszkania jednego z klientów, który mieszkał w Gdańsku:

UPDATE Klient

SET miejscowosc = ‘Kraków’

WHERE miejscowosc = ‘Gdańsk’;

Ta instrukcja będzie oczekiwać na zakończenie transakcji w pierwszej sesji i zdjęcie blokady z danych.

Aby aktualizacja danych była możliwa, w pierwszej sesji należy wykonać instrukcję zamknięcia transakcji COMMIT. Efekt niepowtarzalnego odczytu został wyeliminowany.

Serializable

Serializable to tryb szeregowania. Transakcje odwołujące się do tych samych danych są szeregowane, czyli wykonywane jedna po drugiej. Jest to najwyższy poziom izolacji transakcji, gdzie transakcje są w pełni izolowane od siebie. Na czas trwania transakcji ulegają zablokowaniu całe obiekty, a nie tylko odczytywane dane, co pozwala wyeliminować odczyty widma, ale powoduje, że inni użytkownicy nie mogą modyfikować przechowywanych w obiekcie danych. Na przykład odczytując jeden wiersz tabeli, blokujemy możliwość modyfikowania danych w całej tabeli.

Ćwiczenie 3.22

Otwieramy dwie sesje edytora SQL. W pierwszej ustawiamy tryb szeregowania, rozpoczynamy transakcję i odczytujemy informacje o wybranym kliencie:

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

BEGIN TRANSITION;

SELECT miejscowosc, ulica, nr_domu

FROM Klient

WHERE nazwisko=’Adamek’ AND imie=’Marek’;

Jeżeli w drugiej sesji spróbujemy zmienić dane innego klienta, to okaże się, że aktualizacja została zablokowana:

UPDATE Klient

SET miejscowosc = ‘Warszawa’

WHERE nazwisko=’Borewicz’ AND imie=’Adam’;

Aktualizacja będzie możliwa dopiero po zakończeniu transakcji w pierwszej sesji.

Na zakończenie przykładu należy zamknąć obydwie sesje bez zatwierdzania transakcji.

W tym trybie odczytywane dane zawsze są takie same. Jednak w trakcie transakcji pozostali użytkownicy nie mogą modyfikować zablokowanych tabel. Powoduje to znaczne wydłużenie czasu dostępu do danych i w praktyce, jeśli zmian nie jest dużo, zaleca się przełączenie do modelu optymistycznego.