

Wydanie II

SQL

Praktyczny kurs

Opanuj język SQL w praktyce!

- **Poznaj**
modele baz danych i standardy
języka SQL
- **Naucz się**
korzystać z instrukcji pobierania
i modyfikacji danych
- **Dowiedz się,**
jak tworzyć i zmieniać strukturę bazy
oraz zarządzać jej użytkownikami

Danuta Mendrala
Marcin Szeliga



Helion

» Idź do

- Spis treści
- Przykładowy rozdział
- Skorowidz

» Katalog książek

- Katalog online
- Zamów drukowany katalog

» Twój koszyk

- Dodaj do koszyka

» Cennik i informacje

- Zamów informacje o nowościach
- Zamów cennik

» Czytelnia

- Fragmenty książek online

» Kontakt

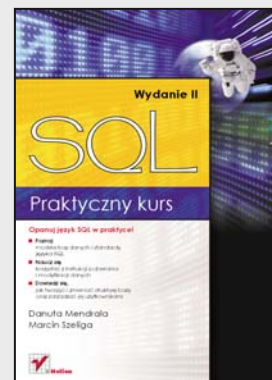
Helion SA
ul. Kościuszki 1c
44-100 Gliwice
tel. 32 230 98 63
e-mail: helion@helion.pl
© Helion 1991–2011

Praktyczny kurs SQL. Wydanie II

Autor: [Danuta Mendrala](#), Marcin Szeliga

ISBN: 978-83-246-3373-9

Format: 158×235, stron: 304



Poznaj modele baz danych i standardy języka SQL

- Naucz się korzystać z instrukcji pobierania i modyfikacji danych
- Dowiedz się, jak tworzyć i zmieniać strukturę bazy oraz zarządzać jej użytkownikami

Opanuj język SQL w praktyce!

Bazy danych są dosłownie wszędzie. Trudno sobie dziś bez nich wyobrazić funkcjonowanie nowoczesnej biblioteki, choćby najmniejszego sklepu internetowego, biura rachunkowego czy nawet niewielkiego serwisu WWW. Użytkownicy korzystający z baz danych często nie mają nawet pojęcia, w jaki sposób odbywa się dostęp do informacji i jaki mechanizm jest za to odpowiedzialny. Na ignorancję tę nie mogą sobie jednak pozwolić osoby odpowiedzialne za tworzenie, zarządzanie i konserwowanie baz danych. Powinny one znać przynajmniej jeden z popularnych serwerów bazodanowych i sprawnie posługiwać się językiem SQL stanowiącym standardowe narzędzie komunikacji z relacyjnymi bazami.

Jeśli pragniesz dołączyć do ekskluzywnego grona administratorów baz danych lub chcesz zostać programistą aplikacji bazodanowych, lecz przeszkadza Ci brak znajomości SQL-a, sięgnij po książkę „Praktyczny kurs SQL. Wydanie II”. W prosty i przystępny sposób prezentuje ona podstawowe pojęcia i zasady rządzące relacyjnym modelem baz danych, a także najważniejsze cechy i konstrukcje języka SQL oraz metody ich wykorzystywania. Lektura książki umożliwi Ci poznanie instrukcji odpowiedzialnych za odczytywanie danych z bazy i ich zapisywanie oraz modyfikację, jak również tworzenie baz i zmianę ich struktury. Poznasz też sposoby tworzenia ról i kont użytkowników oraz zarządzania ich uprawnieniami. Twoją wiedzę ugruntuują praktyczne zadania kończące każdy rozdział, a zamieszczone na końcu książki rozwiązania pomogą skorygować ewentualne błędy.

- Teoretyczne podstawy funkcjonowania baz danych
- Historia języka SQL i obowiązujące standardy zapytań
- Odczytywanie, przeszukiwanie, łączenie i grupowanie danych
- Korzystanie z podzapytań
- Zapisywanie, modyfikacja i usuwanie danych
- Transakcje i równoległy dostęp do danych
- Tworzenie baz danych i modyfikacja ich struktury
- Korzystanie z widoków i indeksów
- Zarządzanie użytkownikami, rolami i prawami dostępu do baz danych

Dowiedz się, jak tworzyć relacyjną bazę danych i zarządzać nią za pomocą języka SQL

Spis treści

Wstęp	9
Część I Trochę teorii, czyli modele i standardy	17
Rozdział 1. Relacyjny model baz danych	19
Tabele jako zbiory danych	19
Kolumny mają niepowtarzalne nazwy i zawierają określone typy danych	20
Wiersze powinny być unikalne	21
Kolejność kolumn jest bez znaczenia	21
Kolejność wierszy jest bez znaczenia	22
Bazy danych	22
Trzy modele baz danych: relacyjny, obiektowy i jednorodny	23
Model jednorodny	23
Model relacyjny	24
Model obiektowy	26
Założenia relacyjnego modelu baz danych	27
Postulaty Codda dotyczące struktury danych	27
Postulaty Codda dotyczące przetwarzania danych	28
Postulaty Codda dotyczące integralności danych	29
Normalizacja	29
Podsumowanie	31
Zadania	31
Rozdział 2. Standardy języka SQL	33
Strukturalny język zapytań	33
Przetwarzanie zbiorów a przetwarzanie pojedynczych danych	34
Język deklaratywny a język proceduralny	35
Język interpretowany a język kompilowany	35
Składnia języka SQL	37
Dialekty języka SQL	39
Standardy ANSI	40
Historia	40
SQL3	41
Podsumowanie	44
Zadania	44

Część II Pobieranie danych, czyli instrukcja SELECT 47

Rozdział 3. Odczytywanie danych z wybranej tabeli 49

Klauzula FROM	49
W pełni kwalifikowane nazwy obiektów	50
Wybieranie kolumn	51
Eliminowanie duplikatów	52
Wyrażenia	54
Operatory arytmetyczne	54
Łączenie danych tekstowych	55
Funkcje systemowe	55
Formatowanie wyników	58
Aliasy	59
Stałe (literały)	60
Sortowanie wyników	60
Sortowanie danych tekstowych	63
Podsumowanie	65
Zadania	65

Rozdział 4. Wybieranie wierszy 67

Logika trójwartościowa	67
Wartość NULL	68
Operatory logiczne	68
Klauzula WHERE	70
Standardowe operatory porównania	71
Operatory SQL	72
Złożone warunki logiczne	75
Klauzula TOP	78
Stronicowanie wierszy	79
Wydajne wyszukiwanie danych	80
W jaki sposób serwery bazodanowe odczytują dane?	81
W jakiej kolejności serwery bazodanowe wykonują poszczególne klauzule zapytań?	84
Argumenty SARG	85
Podsumowanie	87
Zadania	87

Rozdział 5. Łączenie tabel i wyników zapytań 89

Złączenia naturalne i nienaturalne	89
Klucze obce	90
Aliasy	93
Złączenia równościowe i nierównościowe	94
Złączenia zewnętrzne	96
Złączenie lewostronne	97
Złączenie prawostronne	97
Złączenie obustronne	98
Złączenie krzyżowe (iloczyn kartezjański)	98
Złączenia wielokrotne	100
Określanie kolejności złączeń	103
Złączenie tabeli z nią samą	104
Eliminacja duplikatów	105
Klucze obce w obrębie jednej tabeli	106

Łączenie wyników zapytań	107
Suma	108
Część wspólna	111
Różnica	111
Łączenie wierszy i wyników funkcji tabelarycznych	112
Operator APPLY	113
Podsumowanie	115
Zadania	115
Rozdział 6. Grupowanie wierszy	117
Funkcje grupujące	117
Funkcja COUNT()	118
Funkcje SUM() i AVG()	119
Funkcje MIN() i MAX()	120
Inne funkcje grupujące	120
Wyrażenia	121
Klauzula GROUP BY	122
Kolejność wykonywania klauzuli GROUP BY	125
Operatory CUBE i ROLLUP	126
Operator GROUPING SETS	129
Wydajne grupowanie danych	132
Niestandardowa klauzula OVER	132
Partycje	134
Funkcje rankingu	136
Niestandardowe operatory PIVOT i UNPIVOT	137
PIVOT	137
UNPIVOT	140
Klauzula HAVING	141
Podsumowanie	143
Zadania	144
Zadanie dodatkowe, do wykonania w bazie AdventureWorks	144
Rozdział 7. Podzapytania	145
Czym są podzapytania?	145
Podzapytania jako zmienne	146
Podzapytania niepowiązane	146
Podzapytania powiązane	151
Podzapytania jako źródła danych	156
Tabele pochodne	157
CTE	159
Wyznaczanie trendów	165
Operatory	169
Operator EXISTS	170
Operator ANY lub SOME	173
Operator ALL	177
Podsumowanie	178
Zadania	179
Zadanie dodatkowe, do wykonania w bazie AdventureWorks	179

Część III Modyfikowanie danych, czyli instrukcje INSERT, UPDATE, DELETE oraz MERGE 181

Rozdział 8. Modyfikowanie danych 183

Wstawianie danych	183
Klucze podstawowe	184
Wartości domyślne	185
Wartość NULL	186
Konstruktor wierszy	187
Wstawianie wyników zapytań	187
Usuwanie danych	189
Instrukcja DELETE	189
Instrukcja TRUNCATE TABLE	191
Aktualizowanie danych	191
Jednoczesne aktualizowanie wielu kolumn	192
Wyrażenia	193
Aktualizowanie danych wybranych na podstawie danych z innych tabel	193
Aktualizowanie danych za pomocą wyrażeń odwołujących się do innych tabel	194
Instrukcja MERGE	194
Podsumowanie	196
Zadania	196
Zadanie dodatkowe, do wykonania w bazie AdventureWorks	197

Rozdział 9. Transakcje i współbieżność 199

Właściwości transakcji	199
Transakcyjne przetwarzanie danych	201
Tryb jawnego zatwierdzania transakcji	202
Rozpoczynanie transakcji	203
Wycofywanie transakcji	204
Zatwierdzanie transakcji	205
Zagnieżdżanie transakcji	205
Punkty przywracania	206
Współbieżność	207
Blokady	207
Zakleszczenia	208
Poziomy izolowania transakcji	209
Model optymistyczny	213
Model pesymistyczny	214
Podsumowanie	215
Zadania	215

Część IV Tworzenie baz danych, czyli instrukcje CREATE, ALTER i DROP 217

Rozdział 10. Bazy danych i tabele 219

Tworzenie i usuwanie baz danych	219
Tworzenie i usuwanie tabel	222
Schematy	223
Zmiana struktury tabeli	223
Ograniczenia	224
NOT NULL	224
Klucz podstawowy	225
Niepowtarzalność	227

Wartość domyślna	227
Warunek logiczny	228
Klucz obcy	228
Ograniczenia a wydajność instrukcji modyfikujących i odczytujących dane	231
Podsumowanie	232
Zadania	233
Rozdział 11. Widoki i indeksy	235
Widoki	235
Tworzenie i usuwanie widoków	235
Modyfikowanie widoków	238
Korzystanie z widoków	238
Zalety widoków	243
Indeksy	243
Tworzenie, modyfikowanie i usuwanie indeksów	245
Porządkowanie indeksów	247
Podsumowanie	248
Zadania	249
Część V Uprawnienia użytkowników, czyli instrukcje GRANT i REVOKE	251
Rozdział 12. Nadawanie i odbieranie uprawnień	253
Konta użytkowników	253
Zakładanie i usuwanie kont użytkowników	254
Role	255
Tworzenie i usuwanie ról	255
Przypisywanie ról do użytkowników	255
Specjalna rola Public	256
Uprawnienia	256
Nadawanie i odbieranie uprawnień	257
Dziedziczenie uprawnień	258
Przekazywanie uprawnień	260
Zasada minimalnych uprawnień	261
Podsumowanie	261
Zadania	262
Zadanie dodatkowe, do wykonania w bazie AdventureWorks	262
Dodatek A Rozwiązania zadań	263
Zadania z rozdziału 1.	263
Zadanie 1.	263
Zadanie 2.	264
Zadanie 3.	264
Zadania z rozdziału 2.	265
Zadanie 1.	265
Zadanie 2.	265
Zadanie 3.	265
Zadania z rozdziału 3.	266
Zadanie 1.	266
Zadanie 2.	267
Zadanie 3.	267
Zadanie 4.	268
Zadanie 5.	270

Zadania z rozdziału 4.	271
Zadanie 1.	271
Zadanie 2.	272
Zadanie 3.	273
Zadanie 4.	274
Zadania z rozdziału 5.	275
Zadanie 1.	275
Zadanie 2.	275
Zadanie 3.	276
Zadanie 4.	276
Zadania z rozdziału 6.	277
Zadanie 1.	277
Zadanie 2.	278
Zadanie 3.	278
Zadanie 4.	279
Zadania z rozdziału 7.	280
Zadanie 1.	280
Zadanie 2.	281
Zadanie 3.	282
Zadanie 4.	284
Zadania z rozdziału 8.	285
Zadanie 1.	285
Zadanie 2.	286
Zadanie 3.	286
Zadanie 4.	288
Zadania z rozdziału 9.	289
Zadanie 1.	289
Zadanie 2.	290
Zadanie 3.	290
Zadania z rozdziału 10.	291
Zadanie 1.	291
Zadanie 2.	291
Zadanie 3.	292
Zadania z rozdziału 11.	293
Zadanie 1.	293
Zadanie 2.	293
Zadanie 3.	294
Zadania z rozdziału 12.	294
Zadanie 1.	294
Zadanie 2.	295
Zadanie 3.	295
Skorowidz	297

Rozdział 9.

Transakcje i współbieżność

- ◆ Czym są transakcje?
- ◆ Co oznacza skrót ACID?
- ◆ Jakie są zalety transakcyjnego przetwarzania danych?
- ◆ Na czym polega różnica pomiędzy transakcjami zagnieżdżonymi a zagnieżdżaniem transakcji?
- ◆ Co oznacza termin „współbieżność”?
- ◆ Po co serwery bazodanowe zakładają blokady?
- ◆ Kiedy dochodzi do zakleszczeń?
- ◆ Czy warto zmieniać domyślny poziom izolowania transakcji?
- ◆ W jakich sytuacjach optymistyczny model współbieżności jest lepszy niż pesymistyczny?

Właściwości transakcji

Transakcje gwarantują spójność modyfikowanych informacji. Typowym przykładem transakcyjnego przetwarzania danych jest przeniesienie pieniędzy z jednego konta na drugie. Taka operacja przebiega w dwóch etapach:

1. Zmniejszenie o pewną sumę stanu konta X.
2. Dodanie tej sumy do stanu konta Y.

Gdyby po wykonaniu pierwszej operacji wystąpił błąd uniemożliwiający wykonanie drugiej, z systemu zniknęłaby pewna suma pieniędzy. Równie nieprzyjemnym zaskoczeniem dla właściciela byłoby sprawdzenie przez niego stanu obu jego kont już po odjęciu danej sumy z pierwszego, ale przed jej dodaniem do drugiego konta.

Żeby temu zapobiec, transakcje muszą być:

1. Niepodzielne (ang. *Atomicity*). Niepodzielność oznacza, że zatwierdzane są wszystkie wchodzące w skład transakcji instrukcje albo nie jest zatwierdzana żadna z nich. Innymi słowy, wszystkie wchodzące w skład transakcji instrukcje muszą być wykonane poprawnie — jeżeli choć jedna z nich zgłosi błąd, wszystkie przeprowadzone w ramach transakcji zmiany zostaną wycofane.
2. Spójne (ang. *Consistency*). Ta cecha transakcji gwarantuje, że ich wykonanie nie doprowadzi, nawet w przypadku awarii serwera, do utraty spójności danych. Ponieważ wszystkie zmiany danych są wykonywane w ramach transakcji, przechowywane w bazach informacje zawsze będą spójne¹.
3. Izolowane (ang. *Isolation*). Izolowanie transakcji wymaga albo zablokowania modyfikowanych w ramach jednej z nich danych, albo utworzenia ich dodatkowej wersji. W zależności od obowiązującego w ramach serwera lub sesji klienckiej poziomu izolowania transakcji, może dojść do następujących sytuacji:
 - a) Utrata aktualizacji (ang. *Lost update*) ma miejsce, gdy dwa procesy modyfikują jednocześnie te same dane. Przykładowo jeden użytkownik zmienia cenę towaru na 100 zł, a drugi — na 200. W takim przypadku jedna ze zmian zostanie utracona (zastąpiona drugą modyfikacją). **Domyślnie skonfigurowane serwery bazodanowe nie dopuszczają do utraty aktualizacji.**
 - b) Brudne odczyty (ang. *Dirty reads*) — do takiej sytuacji dochodzi, gdy możliwe jest odczytanie zmian niezatwierdzonych jeszcze przez inny proces. Jeżeli proces odczytujący nie zażąda założenia blokady na odczytywanych danych, uzyska do nich dostęp nawet wtedy, kiedy właśnie będą modyfikowane. Gdyby proces modyfikujący wycofał wprowadzone zmiany, odczytane dane okazałyby się niespójne. **Domyślnie skonfigurowane serwery bazodanowe nie dopuszczają brudnych odczytów.**
 - c) Niepowtarzalne odczyty (ang. *Non-repeatable reads*) mają miejsce, gdy powtórzenie w ramach transakcji tego samego odczytu daje inny wynik. Różnice w wynikach są spowodowane tym, że natychmiast po zakończeniu odczytu (a nie po zakończeniu całej transakcji) proces odczytujący zdejmuje blokady założone na odczytywane dane. Niezablokowane dane mogą być zmienione przez inny proces, a więc ich powtórne odczytanie da inny (niespójny) wynik. **Domyślnie skonfigurowane serwery bazodanowe dopuszczają niepowtarzalne odczyty.**
 - d) Odczyty widma (ang. *Phantom reads*) — sytuacja taka ma miejsce, jeżeli pomiędzy dwoma wykonanymi w ramach transakcji odczytami zmieni się liczba odczytywanych wierszy. Jeśli np. podczas pierwszego odczytu w tabeli

¹ Przynajmniej w teorii. W praktyce bazy danych ulegają uszkodzeniu, choć bardzo rzadko z winy serwerów bazodanowych.

Produkty znajdowało się 100 produktów o cenach niższych niż 10 zł, instrukcja `SELECT * FROM Produkty WHERE Cena <10` zwróciłaby 100 wierszy. W trakcie trwania transakcji możliwa jest jednak zmiana pozostałych wierszy tabeli, w tym obniżenie ceny jakiegoś produktu poniżej 10 zł. Możliwe jest również wstawienie do tej tabeli nowego produktu o cenie np. 7 zł. Z tego powodu drugie wykonanie tego samego zapytania zwróciłoby już 102 wiersze.

Domyślnie skonfigurowane serwery bazodanowe dopuszczają odczyty widma.

4. **Trwałe** (ang. *Durability*). Trwałość transakcji gwarantuje, że efekty zatwierdzonych transakcji będą zapisane w bazie, nawet w przypadku awarii serwera baz danych. Do przywrócenia spójności danych serwery bazodanowe z reguły używają jakiejś formy dziennika transakcyjnego.



Pierwsze litery cech transakcji (A — *Atomicity*, C — *Consistency*, I — *Isolation*, D — *Durability*) tworzą skrót ACID, powszechnie używany do opisywania reguł przetwarzania danych, których muszą przestrzegać serwery bazodanowe, żeby mogły być nazwane transakcyjnymi lub relacyjnymi.

Transakcyjne przetwarzanie danych

Serwery bazodanowe mogą działać w trybie niejawnego zatwierdzania transakcji (w serwerze SQL 2011 taki tryb jest trybem domyślnym). Oznacza to, że użytkownicy nie muszą samodzielnie rozpoczynać transakcji, bo serwer robi to za nich.

W trybie niejawnego zatwierdzania transakcji wykonanie każdej instrukcji języka SQL składa się z trzech etapów:

1. Serwer bazodanowy automatycznie rozpoczyna nową transakcję.
2. Wykonywana jest pojedyncza instrukcja SQL.
3. Jeżeli instrukcja została wykonana z powodzeniem, transakcja jest zatwierdzana, w przeciwnym razie jest wycofywana.



Taki sposób działania oznacza, że użytkownicy nie mogą samodzielnie zatwierdzać lub wycofywać automatycznie rozpoczętych transakcji. Dlatego nazywa się on trybem niejawnego zatwierdzania transakcji.

Poniższy przykład ilustruje działanie trybu niejawnego zatwierdzania transakcji za pomocą funkcji systemowej @@TRANCOUNT zwracającej liczbę otwartych, aktywnych w danym momencie transakcji:

```
SELECT @@TRANCOUNT;  
UPDATE dbo.Produkty  
SET [Koszt standardowy]=3  
WHERE [Kod produktu]='NWTC-82';
```

```
SELECT @@TRANCOUNT;
```

```
-----
0
0
```

Jak widać, przed rozpoczęciem i po zakończeniu wykonywania instrukcji UPDATE nie było żadnych otwartych transakcji.

Tryb jawnego zatwierdzania transakcji

W niektórych serwerach bazodanowych (np. w serwerze Oracle) domyślnym trybem transakcyjnego przetwarzania danych jest tryb ich jawnego zatwierdzania. W tym trybie wykonanie każdej instrukcji języka SQL przebiega następująco:

1. Serwer bazodanowy automatycznie rozpoczyna nową transakcję.
2. Wykonywana jest pojedyncza instrukcja SQL.
3. Użytkownik samodzielnie musi zatwierdzić lub wycofać otwartą przez serwer transakcję.

Działanie tego trybu można zasymulować w serwerze SQL 2011, ustawiając opcję sesji `IMPLICIT_TRANSACTIONS`:

```
SET IMPLICIT_TRANSACTIONS ON;
SELECT @@TRANCOUNT;
UPDATE dbo.Produkty
SET [Koszt standardowy]=3
WHERE [Kod produktu]='NWTG-82';
SELECT @@TRANCOUNT;
```

```
-----
0
1
```

Tym razem przed rozpoczęciem instrukcji UPDATE również nie było otwartych transakcji, ale niejawnie rozpoczęta transakcja nie została po jej wykonaniu automatycznie zamknięta. Musi to zrobić sam użytkownik — albo zatwierdzając wprowadzone zmiany, albo je wycofując.

Przed przejściem do dalszych ćwiczeń zakończ transakcję i wyłącz omawiany tryb:

```
COMMIT TRAN;
SET IMPLICIT_TRANSACTIONS OFF;
```



Wskazówka

Tryb jawnego zatwierdzania transakcji pozwala wycofywać przypadkowe lub błędne modyfikacje, ale zatwierdzanie transakcji, która nie została przez nas rozpoczęta, jest mało intuicyjne.

Rozpoczynanie transakcji

Mechanizm transakcyjnego przetwarzania danych pokażemy, jawnie rozpoczynając i kończąc transakcje. Pozwoli nam to wykonać w ramach poszczególnych transakcji dowolną liczbę instrukcji oraz samodzielnie sterować czasem rozpoczęcia i zakończenia poszczególnych transakcji.

Żeby rozpocząć transakcję, należy wykonać instrukcję `BEGIN TRAN`²:

```
BEGIN TRAN;
SELECT @@TRANCOUNT;
-----
1
```

Jeżeli teraz w ramach tej samej sesji (czyli w tym samym oknie edytora SQL) zaktualizujemy ceny wybranych towarów i sprawdzimy liczbę aktywnych transakcji, dowiemy się, że rozpoczęta przez nas transakcja nadal jest otwarta:

```
UPDATE dbo.Produkty
SET [Cena katalogowa]=1
WHERE Kategoria='Zupy'
SELECT @@TRANCOUNT;
-----
1
```

Dopóki transakcja, w ramach której przeprowadziliśmy dowolne zmiany, jest otwarta, możemy je albo wycofać, albo zatwierdzić. Ponieważ serwer bazodanowy nie jest w stanie przewidzieć naszej decyzji, a jedną z cech transakcji jest jej odizolowanie, próba odczytania danych z tabeli `dbo.Produkty` w ramach tej samej sesji skończy się zupełnie inaczej niż ta sama próba wykonana przez innego użytkownika.

Żeby się o tym przekonać:

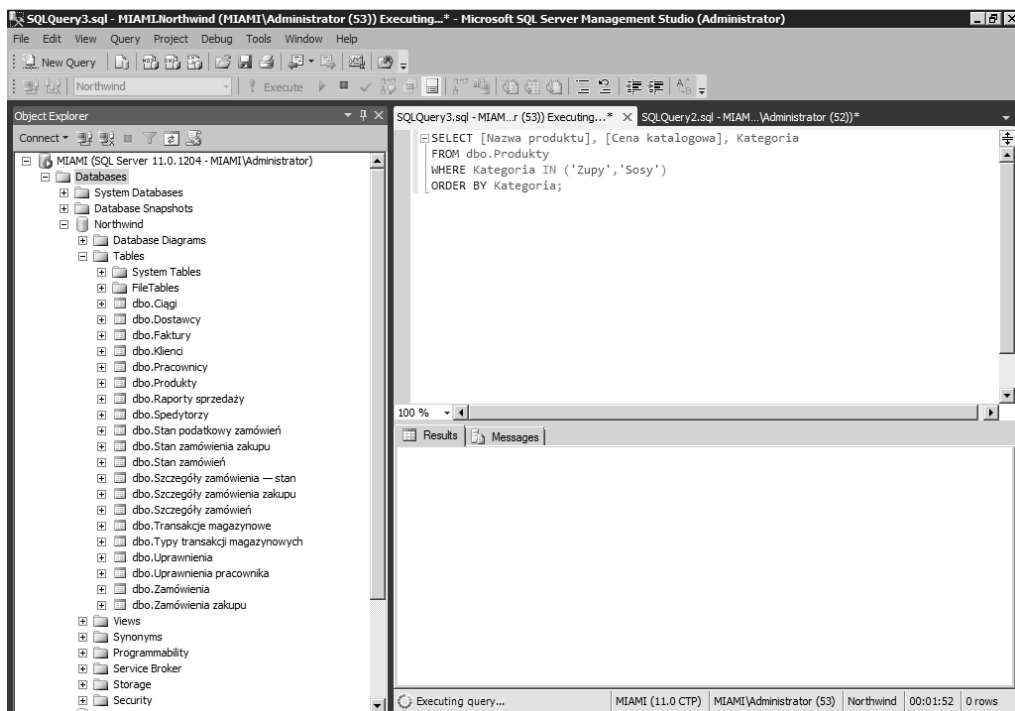
1. W tym samym oknie kodu SQL wykonaj zapytanie:

```
SELECT [Nazwa produktu], [Cena katalogowa], Kategoria
FROM dbo.Produkty
WHERE Kategoria IN ('Zupy','Sosy')
ORDER BY Kategoria;
-----
Northwind Traders Hot Pepper Sauce    21,05    Sosy
Northwind Traders Tomato Sauce        17,00    Sosy
Northwind Traders Curry Sauce         30,00    Sosy
Northwind Traders Clam Chowder         1,00     Zupy
Northwind Traders Vegetable Soup       1,00     Zupy
Northwind Traders Chicken Soup         1,00     Zupy
```

2. Zostało ono natychmiast wykonane, a cena każdej zupy wynosi 1.

3. Aby wykonać to samo zapytanie jako inny użytkownik, otwórz nowe okno edytora SQL³ i skopiuj do niego powyższą instrukcję `SELECT` (rysunek 9.1).

² W niektórych serwerach bazodanowych transakcje rozpoczyna się instrukcjami `BEGIN TRANSACTION` lub `BEGIN WORK`.



Rysunek 9.1. Zapytanie wykonuje się już prawie dwie minuty, ale wciąż nie zwróciło żadnych danych



Wskazówka

Transakcyjne przetwarzanie danych polega na takim realizowaniu żądań klientów przez serwery bazodanowe, żeby każdy klient miał wrażenie, iż jest jedynym użytkownikiem serwera. Wymaga to opisanego w dalszej części rozdziału blokowania obiektów, do których w danym momencie odwołują się inni użytkownicy serwera.

Wycofywanie transakcji

Wycofanie transakcji oznacza przywrócenie danych do stanu sprzed jej rozpoczęcia i zdjęcie wszystkich założonych na potrzeby transakcji blokad. Jeżeli wrócimy do pierwszego okna edytora SQL (tego, w którym zapytanie zwróciło wyniki) i wykonamy w nim instrukcję `ROLLBACK TRAN4`, a następnie przełączymy się do drugiego okna edytora SQL, przekonamy się, że zapytanie wreszcie zostało wykonane i w dodatku ceny produktów z pierwszej podkategorii wcale nie wynoszą 1. Spowodowane jest to wycofaniem transakcji, w ramach której ceny były zmienione, i zdjęciem założonych na jej potrzeby blokad:

³ Można to zrobić, naciskając kombinację klawiszy `Ctrl+N` lub klikając przycisk *New Query*.

⁴ W niektórych serwerach bazodanowych transakcje wycofuje się instrukcjami `ROLLBACK TRANSACTION` lub `ROLLBACK WORK`.

```
SELECT [Nazwa produktu], [Cena katalogowa], Kategoria
FROM dbo.Produkty
WHERE Kategoria IN ('Zupy','Sosy')
ORDER BY Kategoria;
```

```
-----
Northwind Traders Hot Pepper Sauce    21,05    Sosy
Northwind Traders Tomato Sauce        17,00    Sosy
Northwind Traders Curry Sauce         30,00    Sosy
Northwind Traders Clam Chowder         7,2375   Zupy
Northwind Traders Vegetable Soup      1,4175   Zupy
Northwind Traders Chicken Soup        1,4625   Zupy
```

Zatwierdzanie transakcji

Zatwierdzenie transakcji oznacza utrwalenie wprowadzonych w jej trakcie zmian i zdjęcie wszystkich założonych na potrzeby transakcji blokad. Wspomniany na początku rozdziału przykład przelania pieniędzy z jednego konta na drugie mógłby być zaimplementowany w taki sposób:

```
BEGIN TRAN;
EXEC uspDodajDoKonta '123-456-78-90', 500;
EXEC uspOdjemijOdKonta '231-645-87-09', 500;
IF @@ERROR=0
    COMMIT TRAN;
ELSE
    ROLLBACK TRAN;
```

Po jawnym rozpoczęciu transakcji następuje wywołanie dwóch (nieistniejących w przykładowej bazie danych) procedur. Jeżeli żadna z nich nie zgłosi błędu, cała transakcja będzie zatwierdzona (zatwierdzić transakcję możemy, wykonując instrukcję COMMIT TRAN⁵), w przeciwnym razie zostanie ona wycofana.

Zagnieżdżanie transakcji

Większość serwerów bazodanowych pozwala zagnieżdżać transakcje, czyli wykonać instrukcję BEGIN TRAN w ramach wcześniej rozpoczętej transakcji. **Wynikiem takiej operacji jest zwiększenie licznika otwartych transakcji, a nie rozpoczęcie nowej (atomowej, niepodzielnej, trwałej i spójnej) transakcji.**

Działanie mechanizmu zagnieżdżania transakcji ilustruje poniższy przykład: wykonanie instrukcji BEGIN TRAN powoduje zwiększenie o jeden licznika otwartych transakcji, wykonanie instrukcji COMMIT TRAN zmniejsza wartość tego licznika o jeden, ale wykonanie instrukcji ROLLBACK TRAN zamyka transakcje i zeruje licznik otwartych transakcji:

```
BEGIN TRAN;
SELECT @@TRANCOUNT;
```

⁵ W niektórych serwerach bazodanowych transakcje zatwierdza się instrukcjami COMMIT TRANSACTION lub COMMIT WORK.

```

BEGIN TRAN;
SELECT @@TRANCOUNT;
BEGIN TRAN;
SELECT @@TRANCOUNT;
COMMIT TRAN;
SELECT @@TRANCOUNT;
ROLLBACK TRAN;
SELECT @@TRANCOUNT;

```

```

-----
1
2
3
2
0

```

Punkty przywracania

Większość serwerów bazodanowych pozwala wycofać nie tylko całą transakcję, lecz także jej część. W tym celu należy w trakcie transakcji wykonać instrukcję `SAVE TRAN`⁶, a następnie przywrócić ją do danego punktu:

```

BEGIN TRAN;
INSERT INTO dbo.Dostawcy(Firma)
VALUES ('TEST1');
SAVE TRAN PP1;
INSERT INTO dbo.Dostawcy(Firma)
VALUES ('TEST2');
SELECT @@TRANCOUNT;
ROLLBACK TRAN PP1;
SELECT @@TRANCOUNT;

```

```

-----
1
1

```

Ponieważ przywrócenie stanu transakcji do określonego punktu nie powoduje jej zakończenia (liczba otwartych transakcji nadal wynosi 1), musimy ją zatwierdzić lub wycofać:

```

SELECT ID, Firma
FROM dbo.Dostawcy
WHERE Firma LIKE 'TEST_';

```

```

-----
17  TEST1

```

Jako że druga instrukcja `INSERT` została wykonana po zdefiniowaniu punktu przywracania `PP1`, instrukcja `ROLLBACK TRAN PP1` przywróciła stan danych do momentu sprzed jej wykonania i w rezultacie tylko pierwszy wiersz został na trwałe wstawiony do tabeli.

⁶ W niektórych serwerach bazodanowych punkty przywracania tworzy się instrukcjami `SAVE TRANSACTION` lub `SAVE WORK`.

Współbieżność

Współbieżność to zdolność systemu do jednoczesnego realizowania wielu operacji, z reguły uzyskiwana poprzez uruchomienie osobnych procesów (robotników) na potrzeby obsługi poszczególnych żądań.



Współbieżność ma ogromny wpływ na skalowalność serwerów bazodanowych, czyli ich zdolność do coraz szybszego wykonywania transakcji dzięki rozbudowywaniu komputerów, na przykład zwiększaniu ich mocy obliczeniowej czy przepustowości dysków twardych.

Żeby każdy z kilkuset czy nawet kilku tysięcy jednoczesnych użytkowników serwera bazodanowego mógł pracować tak, jakby był jego jedynym użytkownikiem, konieczne jest odizolowanie od siebie poszczególnych transakcji. Umożliwiają to automatycznie zakładane blokady.

Blokady

Pomijając analizy wewnętrznych mechanizmów działania różnych serwerów bazodanowych, blokady można podzielić ze względu na ich tryb (sposób blokowania) i zakres (typ blokowanych zasobów).

Tryby blokad

Tryb blokady decyduje o tym, czy możliwe będzie jej założenie na zasobie wcześniej zablokowanym przez inny proces:

- 1. Blokady współdzielone S** (ang. *Shared*) są domyślnie zakładane na odczytywanych obiektach, takich jak tabele czy wiersze. Na obiekt zablokowany w trybie S inne procesy też mogą założyć blokadę S, czyli **odczytujący nie blokują innych odczytujących**. Blokady S domyślnie zakładane są tylko na czas wykonywania zapytania, a nie całej transakcji.
- 2. Blokady wyłączne X** (ang. *exclusive*) są zakładane na modyfikowanych obiektach. Blokady X są niekompatybilne z innymi blokadami, czyli modyfikujący blokują innych użytkowników. W przeciwieństwie do blokad współdzielonych blokady wyłączne domyślnie są utrzymywane do zakończenia całej transakcji, a nie pojedynczej operacji.

Zakresy blokad

Blokady mogą być zakładane na poziomie poszczególnych wierszy, kluczy indeksów, stron, zakresów lub całych tabel. Te obiekty tworzą naturalną hierarchię: tabela składa się z wielu stron, na każdej stronie zapisanych jest wiele wierszy itd. Z tego powodu serwery bazodanowe muszą analizować wszystkie istniejące blokady, zanim założą nową — jeżeli choć jeden wiersz tabeli jest zablokowany w trybie X, nie można na całej tabeli założyć innej blokady.



Im większe obiekty są blokowane, tym mniejsza współbieżność (bo użytkownicy muszą dłużej czekać na dostęp do zablokowanych zasobów), ale również tym mniejsza liczba blokad, którymi musi zarządzać serwer bazodanowy (zostanie założona jedna blokada na całą tabeli zamiast miliona blokad na poszczególnych wierszach).

Zakleszczenia

Zakleszczenie (ang. *DeadLock*) ma miejsce, gdy różne procesy blokują się nawzajem w taki sposób, że żaden z nich nie jest w stanie założyć blokad wymaganych do ukończenia już rozpoczętych operacji.

Najczęściej występują dwa typy zakleszczeń:

1. Zakleszczenia cykliczne, wynikające z tego, że dwa procesy w różnych kolejnościach próbują uzyskać dostęp do tych samych zasobów.
2. Zakleszczenia konwersji blokad, związane ze zmianą wcześniej założonej blokady współdzielonej (wiele procesów może jednocześnie zablokować ten sam zasób w trybie S) na blokadę wyłączną (tylko jeden proces może założyć na tym samym obiekcie blokadę X).

Serwery bazodanowe automatycznie wykrywają zakleszczenia i przerywają działanie jednego procesu. Na ofiarę zakleszczenia wybierany jest proces o niższym priorytecie, a jeżeli oba procesy mają ten sam priorytet, ofiarą zakleszczenia zostaje ten, którego wycofanie jest mniej kosztowne.

Mechanizm wykrywania i usuwania zakleszczeń pokazuje poniższy przykład:

Pierwszy użytkownik w ramach jawnie rozpoczętej transakcji modyfikuje kilka danych w tabeli `dbo.Dostawcy`:

```
BEGIN TRAN;
UPDATE dbo.Dostawcy
SET Firma = UPPER(Firma)
WHERE ID<5;;
```

(4 row(s) affected)

Następnie inny użytkownik w ramach jawnie rozpoczętej przez siebie transakcji modyfikuje znacznie więcej danych w tabeli `Transakcje magazynowe`⁷:

```
BEGIN TRAN;
UPDATE dbo.[Transakcje magazynowe]
SET Ilość += 1
WHERE [ID transakcji] <135;
```

(100 row(s) affected)

⁷ Możemy zasymulować jednoczesną pracę dwóch użytkowników, otwierając nowe okno edytora SQL — każde z okien nawiązuje własną sesję z bazą danych.

W dalszej kolejności pierwszy użytkownik próbuje odczytać zawartość tabeli zablokowanej już przez drugą sesję (okno wyników może pokazać kilka wierszy, ale i tak użytkownik będzie musiał czekać na możliwość zablokowania w trybie S pozostałych wierszy tabeli Transakcje magazynowe):

```
SELECT *
FROM dbo.[Transakcje magazynowe];
```

W tym momencie nie wystąpiło jeszcze zakleszczenie — wystarczyłoby, żeby drugi użytkownik zakończył swoją transakcję. Ale jeżeli w ramach drugiej sesji użytkownik spróbuje odczytać zawartość tabeli zmodyfikowanej przez pierwszego użytkownika, oba procesy się zakleszczą:

```
SELECT ID, Firma, [Tytuł zawodowy]
FROM dbo.Dostawcy;
```

```
-----
1      Dostawca A   Kierownik ds. sprzedaży
2      Dostawca B   Kierownik ds. sprzedaży
3      Dostawca C   Przedstawiciel handlowy
4      Dostawca D   Kierownik ds. marketingu
5      Dostawca E   Kierownik ds. sprzedaży
...
```

Po chwili drugie zapytanie zostało jednak wykonane, co więcej — nazwy firm nie zostały przekonwertowane na wielkie litery. Żeby przekonać się, dlaczego tak się stało, wystarczy przełączyć się do okienka pierwszej sesji. Znajdziemy w nim poniższy komunikat błędu:

```
Transaction (Process ID 52) was deadlocked on lock resources with another process
➤ and has been chosen as the deadlock victim. Rerun the transaction.
```

Jeżeli sprawdzimy liczbę otwartych w ramach pierwszej sesji transakcji, okaże się, że jawnie rozpoczęta przez pierwszego użytkownika transakcja została — zgodnie z komunikatem błędu — wycofana:

```
SELECT @@TRANCOUNT;
```

```
-----
0
```

Ponieważ wycofanie transakcji wiąże się ze zdjęciem założonych na jej potrzeby blokad, druga sesja mogła z powodzeniem zakończyć operacje i odczytać tabelę dbo.Dostawcy. Liczba transakcji otwartych w ramach drugiej sesji nadal wynosi 1 — żeby zakończyć ćwiczenie i wycofać zmiany, należy wykonać w tym oknie edytora SQL instrukcję ROLLBACK TRAN.

Poziomy izolowania transakcji

Możemy wpływać na sposób zakładania blokad przez serwery bazodanowe, zmieniając poziom izolowania transakcji. Większość serwerów pozwala ustawić (na poziomie serwera, bazy danych lub poszczególnych sesji) jeden z czterech poziomów izolowania transakcji, przedstawionych przez nas od najmniej restrykcyjnego, w którym maksymalna współbieżność jest okupiona występowaniem największej liczby typów niespójności danych, do najbardziej restrykcyjnego, który kosztem ograniczenia współbieżności gwarantuje najwyższy poziom spójności danych.

Read Uncommitted

W trybie niezatwierdzonego odczytu (ang. *Read Uncommitted*) odczyt danych nie powoduje założenia blokady współdzielonej. **Na tym poziomie występują brudne odczyty, niepowtarzalne odczyty i odczyty widma (jedynym niekorzystnym zjawiskiem niewystępującym na tym poziomie jest utrata aktualizacji).**

Żeby się o tym przekonać:

1. W jednej sesji (oknie edytora SQL) rozpoczniemy transakcję i zaktualizujemy adres klienta:

```
BEGIN TRAN;
UPDATE dbo.Klienci
SET Adres = 'ZmianaWToku'
WHERE ID=1;
```

(1 row(s) affected)

2. W drugiej sesji zmienimy poziom izolowania transakcji na Read Uncommitted i spróbujemy odczytać modyfikowane przez innego użytkownika dane:

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
GO
SELECT Adres
FROM dbo.Klienci
WHERE ID=1;
```

ZmianaWToku

Udało nam się odczytać dane, pomimo że osoba, która je zmieniała, nie zatwierdziła jeszcze transakcji, a więc w każdej chwili może ją wycofać. **W tym trybie** (często wymuszonym na poziomie poszczególnych instrukcji za pomocą specyficznych dla danego serwera bazodanowego dyrektyw optymalizatora) **można odczytywać dane, o których wiemy, że nie będą w tym samym czasie modyfikowane.**

Kończąc ćwiczenie, zamknij bez zatwierdzania otwartej transakcji i na nowo otwórz oba okna edytora SQL — w ten sposób kolejne ćwiczenie rozpoczniemy, pracując w domyślnym trybie izolowania transakcji.

Read Committed

Tryb odczytu zatwierdzonego (ang. *Read Committed*) **jest domyślnym poziomem izolowania transakcji.** Na tym poziomie odczyt danych wymaga założenia na nich blokady współdzielonej. Ponieważ zakładana na czas zmiany blokada X jest niekompatybilna z innymi blokadami, w tym z blokadą S, eliminuje to brudne odczyty. Jednak **na tym poziomie nadal występują niepowtarzalne odczyty i odczyty widma.**

Zjawisko niepowtarzalnego odczytu pokazuje poniższy przykład:

1. W pierwszym oknie edytora SQL ustawiamy tryb odczytów zatwierdzonych⁸, jawnie rozpoczynamy transakcję i odczytujemy adres wybranego klienta:

⁸ Ponieważ ten tryb jest trybem domyślnym, instrukcja SET jest dodana tylko w celach demonstracyjnych.

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
BEGIN TRAN;
SELECT Adres
FROM dbo.Klienci
WHERE ID=1;
```

ul. Jasna 123

2. W tym momencie transakcja jest nadal otwarta, a my w drugim oknie edytora SQL zmienimy adres tego klienta:

```
UPDATE dbo.Klienci
SET Adres = 'OdczytWToku'
WHERE ID=1;
```

(1 row(s) affected)

3. Jeżeli pierwszy użytkownik w ramach tej samej transakcji ponownie odczyta adres tego klienta, uzyska inny wynik:

```
SELECT Adres
FROM dbo.Klienci
WHERE ID=1;
COMMIT TRAN;
```

OdczytWToku

Repeatable Read

W trybie powtarzalnego odczytu (ang. *Repeatable Read*) blokady współdzielone typu S są utrzymywane do czasu zakończenia całej transakcji. Dzięki temu inny proces nie może zmodyfikować odczytywanych w jej ramach danych, co eliminuje niepowtarzalne odczyty. Z niekorzystnych zjawisk związanych z izolowaniem transakcji **na tym poziomie występują tylko odczyty widma**.

Zjawisko odczytu widma pokazuje poniższy przykład:

1. W ramach pierwszej sesji zmienimy poziom izolowania transakcji na Repeatable Read i w ramach jawnie rozpoczętej transakcji odczytamy nazwy towarów o kodach kończących się cyfrą 6:

```
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
BEGIN TRAN;
SELECT [Nazwa produktu]
FROM dbo.Produkty
WHERE [Kod produktu] LIKE '%6';
```

Northwind Traders Boysenberry Spread
Northwind Traders Marmalade
Northwind Traders Gnocchi
Northwind Traders Tomato Sauce
Northwind Traders Cake Mix
Northwind Traders Smoked Salmon

2. Podczas gdy pierwsza transakcja jest wciąż otwarta, w drugim oknie edytora SQL zmienimy kod jednego z pozostałych, niezwróconych przez pierwsze zapytanie produktu na NWT00-6, a więc na kod spełniający warunki pierwszego zapytania:

```
UPDATE dbo.Produkty
SET [Kod produktu] = 'NWTC0-6'
WHERE [Kod produktu] = 'NWTC0-3';
```

(1 row(s) affected)

3. Jeżeli pierwszy użytkownik raz jeszcze wykona, w ramach tej samej transakcji, to samo zapytanie, tym razem jego wynik będzie liczył więcej wierszy — pojawi się w nim wiersz widmo:

```
SELECT [Nazwa produktu]
FROM dbo.Produkty
WHERE [Kod produktu] LIKE '%6';
```

Northwind Traders Syrup
Northwind Traders Boysenberry Spread
Northwind Traders Marmalade
Northwind Traders Gnocchi
Northwind Traders Tomato Sauce
Northwind Traders Cake Mix
Northwind Traders Smoked Salmon

4. Jeśli jednak w ramach drugiej sesji spróbujemy zmienić dane odczytywane w ramach nadal otwartej pierwszej transakcji (czyli doprowadzić do niepowtarzalnego odczytu), instrukcja będzie oczekiwać, aż pierwsza transakcja zostanie zakończona, a założone dla niej blokady zdjęte:

```
UPDATE dbo.Produkty
SET [Kod produktu] = 'NWTC0-1'
WHERE [Kod produktu] = 'NWTJP-6';
```

5. Żeby powyższa aktualizacja została wykonana, w pierwszym oknie edytora SQL wykonaj instrukcję COMMIT TRAN.

W trybie Repeatable Read należy odczytywać te dane, które w ramach transakcji są odczytywane kilkakrotnie i mogą być zmieniane w tym samym czasie przez innych użytkowników. Sytuacja taka ma miejsce np. w różnego rodzaju zestawieniach i raportach zbiorczych, w których odczytując te same dane, za każdym razem musimy otrzymać te same wyniki, inaczej zestawienie lub raport będą niespójne.

Serializable

W trybie szeregowania transakcje odwołujące się do tych samych tabel są wykonywane jedna po drugiej. Blokowanie całych obiektów, a nie tylko odczytywanych danych, na czas trwania transakcji pozwala wyeliminować odczyty widma, ale powoduje, że odczytując nawet jeden wiersz tabeli, możemy uniemożliwić pozostałym użytkownikom zmodyfikowanie przechowywanych w niej danych.

Żeby się o tym przekonać:

1. W pierwszym oknie edytora SQL przełączymy się do trybu szeregowania, jawnie rozpoczniemy transakcję i odczytamy informacje o wybranym towarze:

```
SELECT [Nazwa produktu]
FROM dbo.Produkty
```

```
WHERE [Kod produktu] LIKE '%6';
```

```
-----  
Northwind Traders Syrup  
Northwind Traders Gnocchi  
Northwind Traders Tomato Sauce  
Northwind Traders Cake Mix  
Northwind Traders Smoked Salmon
```

2. Jeżeli teraz w drugim oknie edytora SQL spróbujemy zmienić dane dowolnego, również niezwróconego przez pierwsze zapytanie produktu, okaże się, że aktualizacja została zablokowana i będzie wykonana dopiero po zakończeniu pierwszej transakcji:

```
UPDATE dbo.Produkty  
SET [Kod produktu] = 'NWTCA-48'  
WHERE [Kod produktu] = 'NWTCA-49';
```

3. Kończąc ćwiczenie, zamknij oba okna edytora SQL bez zatwierdzania rozpoczętej w jednym z nich transakcji.

W trybie *Serializable* mamy gwarancję, że odczytywane w ramach transakcji dane zawsze będą takie same — serwer bazodanowy nie dopuści nie tylko do ich zmiany, lecz także do pojawienia się nowych danych. Jednak przez ten czas pozostali użytkownicy nie będą mogli modyfikować zablokowanych tabel. W większości przypadków powoduje to tak znaczne wydłużenie czasu reakcji serwera, że lepiej jest skopiować odczytywane dane⁹, a jeżeli zmian nie jest zbyt dużo, przełączyć się do modelu optymistycznego.

Model optymistyczny

W modelu optymistycznym tylko modyfikujący blokują innych modyfikujących, czyli różni użytkownicy mogą jednocześnie modyfikować i odczytywać te same dane.

Serwery bazodanowe zapewniają spójność modyfikowanych w tym modelu danych poprzez ich wersjonowanie. Zakładając (optymistycznie), że w czasie gdy jeden użytkownik odczytuje dane, inni raczej nie będą ich modyfikować, serwery te są w stanie na bieżąco zarządzać dodatkowymi wersjami danych.

Jeżeli to założenie jest prawdziwe, czyli jeżeli jednoczesne modyfikacje i odczyty tych samych danych nie zachodzą zbyt często, możemy znacznie skrócić czas reakcji serwera¹⁰, przełączając bazę do optymistycznego modelu współbieżności. Żeby się o tym przekonać:

1. W pierwszym oknie edytora SQL wykonamy poniższe instrukcje, przełączając bazę *Northwind* do modelu optymistycznego:

```
USE master;  
ALTER DATABASE Northwind  
SET READ_COMMITTED_SNAPSHOT ON
```

⁹ Niektóre serwery bazodanowe pozwalają utworzyć migawkę (ang. *Snapshot*) danych.

¹⁰ Niektóre serwery bazodanowe, np. serwer Oracle, domyślnie działają w optymistycznym modelu współbieżności.

```
WITH ROLLBACK IMMEDIATE;
```

```
-----  
Command(s) completed successfully.
```

2. W tym samym oknie edytora SQL połączymy się z bazą *Northwind* i w ramach jawnie rozpoczętej transakcji zmienimy dane dwóch pracowników:

```
USE Northwind;  
BEGIN TRAN;  
UPDATE dbo.Pracownicy  
SET Nazwisko = 'X'  
WHERE ID <3;
```

```
-----  
(2 row(s) affected)
```

3. W nowym oknie edytora SQL odczytamy dane o kilku pracownikach:

```
SELECT ID, Nazwisko  
FROM dbo.Pracownicy  
WHERE ID <3;
```

```
-----  
1      Ciesielska  
2      Czupta
```

4. Okazuje się, że tym razem zapytanie zostało wykonane natychmiast, ale z zachowaniem wymogów domyślnego trybu izolowania transakcji, czyli trybu Read Committed — **pozostali użytkownicy serwera odczytają ostatnią zatwierdzoną wersję danych**. Gdyby rozpoczęta w ramach pierwszej sesji transakcja została zatwierdzona, to ponowne wykonanie tego samego zapytania zwróciłoby najnowszą, zatwierdzoną wersję ze zmienionymi danymi dwóch pierwszych pracowników.

Model pesymistyczny

W modelu pesymistycznym odczytujący są blokowani przez modyfikujących (serwer będzie czekał z założeniem blokady S, aż zdjęta zostanie blokada X), a modyfikujący przez odczytujących (założenie blokady X wymaga zdjęcia blokady S).

Ponieważ koszt zarządzania wieloma wersjami tych samych danych rośnie wraz ze wzrostem wersjonowanych danych, w tym modelu zakłada się (pesymistycznie), że odczytywane dane będą w tym samym czasie regularnie modyfikowane.

Żeby przywrócić pesymistyczny (domyślny) model współbieżności bazy *Northwind*, należy wykonać poniższe instrukcje:

```
USE master;  
ALTER DATABASE Northwind  
SET READ_COMMITTED_SNAPSHOT OFF  
WITH ROLLBACK IMMEDIATE;
```

```
-----  
Nonqualified transactions are being rolled back. Estimated rollback completion:  
↪100%.
```


Podsumowanie

- ♦ Serwery bazodanowe przeprowadzają wszystkie zmiany danych w ramach jawnie lub niejawnie rozpoczętych transakcji.
- ♦ Transakcje powinny być otwierane jak najpóźniej i zamykane jak najwcześniej.
- ♦ Transakcje powinny zawierać tylko powiązane ze sobą instrukcje.
- ♦ Przerwane (czy to z powodu awarii klienta, czy też serwera) transakcje będą wycofane.
- ♦ Na czas trwania transakcji pewne obiekty bazy danych są automatycznie blokowane.
- ♦ Serwery bazodanowe automatycznie wykrywają zakleszczenia i usuwają je poprzez wycofanie jednej z zakleszczonych transakcji.
- ♦ Odizolowanie, jedną z czterech cech ACID transakcji, uzyskuje się za pomocą automatycznie zakładanych i zwalnianych blokad.
- ♦ Można sterować sposobem zakładania i czasem trwania blokad, zmieniając poziom izolowania transakcji.
- ♦ W modelu optymistycznym serwery bazodanowe wersjonują dane, co poprawia współbieżność kosztem większego obciążenia serwera.

Zadania

1. Twoim zadaniem jest przygotowanie raportu podsumowującego roczną sprzedaż. Wyliczając sumy i średnie wartości sprzedaży produktów, kilkukrotnie musisz odczytać tabelę `dbo.Produkty`. Jak zagwarantujesz poprawność wyników raportu?
2. Po przerwie na lunch użytkownicy zgłaszają, że próby dalszej pracy z bazą danych kończą się chwilowym „zawieszeniem” programu i wreszcie komunikatem błędu mówiącym, iż serwer bazodanowy jest niedostępny. Po sprawdzeniu okazuje się, że serwer i sieć działają normalnie, a baza nie została uszkodzona. Co jest najbardziej prawdopodobną przyczyną problemu?
3. W ramach tworzonej procedury modyfikujesz duże ilości danych zapisanych w kilkunastu tabelach oraz wstawiasz jeden wiersz, informujący o wykonaniu wszystkich operacji, do tabeli znajdującej się w bazie danych na zdalnym serwerze. Połączenie między serwerami jest mocno obciążone i zdarza się, że czas nawiązywania sesji i przesyłania danych pomiędzy serwerami wielokrotnie się wydłuża. Co zrobić, aby w przypadku zgłoszenia przez procedurę błędu braku połączenia ze zdalnym serwerem nie trzeba było ponownie wykonywać długotrwałych modyfikacji danych?

Skorowidz

A

- access, 10
- ACID, 201
- aktualizacja
 - automatyczna wartości kluczy, 230
 - danych, 191
 - danych wybranych na podstawie danych z innych tabel, 193
 - danych za pomocą wyrażeń odwołujących się do innych tabel, 194
 - utrata, 200
 - wielu kolumn jednocześnie, 192
- alias, 59, 93
- apostrof, 74
- argument
 - SARG, 85, 132

B

- baza danych, 9, 22, 219
 - Northwind, 89, 100, 124
 - relacyjna, 19, 26
- blokada, 207
 - tryb, 207
 - współdzielona, 207
 - wyłączna X, 207
 - zakres, 207

C

- Codd Edgar Frank, 29
- Connection Statements, 41
- Control Statements, 41
- CTE
 - proste, 160, 162
 - rekurencyjne, 162
- czcionka o stałej szerokości, 15

D

- dane
 - aktualizacja, 191
 - aktualizacja na podstawie danych z innych tabel, 193
 - aktualizacja za pomocą wyrażeń odwołujących się do innych tabel, 194
 - grupowanie
 - wydajne, 132
 - przetwarzanie pojedyncze, 34
 - tekstowe
 - generowanie, 99
 - łączenie, 55
 - sortowanie, 63
 - typ, 20, 42
 - usuwanie, 189
 - wstawianie, 183
- Data Statements, 41
- DB2, 10
- Diagnostics Statements, 41
- dialekt
 - języka SQL, 39
- dokument XML, 42
 - XML, 42
- duplikat, 52
 - eliminacja, 105
- dyrektywa GO, 85

F

- funkcja
 - arytmetyczna
 - ABS, 56
 - CEILING, 56
 - FLOOR, 56
 - POWER, 56

funkcja

arytmetyczna

RAND, 56
 ROUND, 56
 SQRT, 56

CASE, 58

CAST, 57

czas

DATEADD, 57

data

DATEADD, 57
 DAY, 57
 GETDATE, 57
 MONTH, 57
 YEAR, 57

GROUPING, 128

GROUPING_ID, 128

grupująca

AVG, 119
 CHECKSUM_AGG, 121
 COUNT, 118
 COUNT_BIG, 121
 MAX, 120
 MIN, 120
 STDEV, 121
 STDEVP, 121
 SUM, 119
 VAR, 121
 VARP, 121
 zagnieżdżanie, 122

rankingu, 136

systemowa, 55

tabelaryczna

łączenie, 112

znakowa

LEN, 56
 LOWER, 56
 LTRIM, 56
 REPLACE, 56
 REPLICATE, 56
 RTRIM, 56
 SUBSTRING, 56
 UPPER, 56

I

identyfikator, 24, 37, 50

spedytora, 146

iloczyn kartezjański, 98

indeks, 22, 81, 87, 235, 243

modyfikowanie, 245

odtworzenie, 247

opcje, 247

tworzenie, 245

uporządkowanie kluczy, 248

usunięcie, 247

usuwanie, 245

instrukcja

ALTER, 223

ALTER DATABASE, 221

ALTER VIEW, 238

CALL, 11

COMMIT TRAN, 205

CREATE ROLE, 255

CREATE SCHEMA, 223

CREATE TABLE, 222

CREATE USER, 254

CREATE VIEW, 235

CREATE., 219

DCL, 40

DDL, 40

DELETE, 189, 191

DENY, 257, 258

DML, 40

DROP, 221, 232

DROP ROLE., 255

DROP TABLE, 222

EXEC, 11

IF ... THEN ... ELSE, 58

INSERT INTO ... SELECT, 188

klasa, 41

MERGE, 194

modyfikująca, 231

REVOKE, 257, 258

SELECT, 54, 236

SELECT ... INTO, 187

TRUNCATE TABLE, 189, 191

UPDATE, 191, 192

InterBase, 10

InterBase Firebird, 10

J

język

deklaratywny, 35

interpretowany, 35

kompilowany, 35

proceduralny, 35

SEQUEL, 33

K

kaskadowe usuwanie, 230

katalog, 22

klasa instrukcji, 41

klaster, 22

klauszula

CHECK OPTION, 242
FROM, 49, 50, 59, 80
GROUP BY, 122, 124, 129
HAVING, 141
ON DELETE, 230
ON UPDATE, 230
ORDER BY, 62, 236
ORDER BY,, 80
OVER, 132
SELECT, 59, 80
SET, 193
TOP, 78, 79, 80
WHEN, 58
WHERE, 70, 80, 91

klucz

indeksu, 81
obcy, 24, 90, 106, 228
podstawowy, 21, 24, 90, 184, 225
kompozytowy, 226

kolumna, 21, 24, 51

aktualizacja wielu kolumn jednocześnie, 192

komentarz, 37, 39

kompilator, 35

konsola SSMSE, 14

konstruktor

wiersz, 187

konto użytkownika, 253

usuwanie, 254

zakładanie, 254

konwencja, 15

kursywa, 15

L

lista wartości, 149

literał, 37, 38, 60

logika trójwartościowa, 67

Ł

łączenie

danych tekstowych, 55

M

model

jednorodny, 23
obiektyowy, 26
optymistyczny, 213
pesymistyczny, 214
relacyjna baza danych, 19
relacyjny, 24

MySQL, 10

N

niepowtarzalność, 227

niezgodność

składni, 26

typów, 26

użycia, 27

normalizacja, 29

NOT NULL, 224

O

obiekt, 20, 37

odczyt

brudny, 200

niepowtarzalny, 200

niezatwierdzony, 210

powtarzalny, 211

widma, 200

zatwierdzony, 210

ograniczenia, 224, 231

operator, 37, 38

ALL, 170, 177

AND, 68, 69, 75

ANY, 173, 175

ANY lub SOME, 169

APPLY, 113

arytmetyczny, 38, 54

BETWEEN ... AND, 73

CROSS APPLY, 114

CUBE, 126

część wspólna, 111

EXCEPT, 111

EXISTS, 169, 170, 171

GROUPING SETS, 129, 131

IN, 72, 150, 153

INTERSECT, 111

IS NULL, 74

JOIN ... ON, 91

Key Lookup, 82, 85

konkatenacji, 55

LIKE, 74

logiczny, 38, 68

NATURAL JOIN, 92

NOT, 68, 69, 85

OR, 68, 69, 75

PIVOT, 137

porównania, 38, 71

mniejszy lub równy, 71

mniejszy niż, 71

równy, 71

różny, 71

większy lub równy, 71

większy niż, 71

operator

- ROLLUP, 126
- różnica, 111
- SOME, 173
- SQL, 72
- suma, 108
- UNION, 109
- UNION ALL, 109
- UNPIVOT, 137, 140
- znakowy, 38

optymalizacja, 36

Oracle Database, 10

ORM, 27

zabezpieczenia przed modyfikacjami

- przeprowadzanymi za pomocą języków
- proceduralnych, 28

poziom

- izolowania transakcji, 209
- zgodności, 43
 - pełny, 40
 - podstawowy, 40
 - pośredni, 40

przetwarzanie

- pojedynczych danych, 34
- zbioru, 34

punkt przywracania, 206

P

partycja, 134

PL/pgSQL, 39

PL/SQL, 39

podzapytanie, 145, 146, 147, 156

- jako źródła danych, 156
- niepowiązane, 146
- niezwracające żadnych wartości, 150
- powiązane, 151
- usuwanie wyników, 190
- zagnieżdżanie, 151
- zwracające listę wartości, 149

pogrubienie, 15

postać

- normalna
 - druga, 30
 - pierwsza, 29
 - trzecia, 30

sterty, 243

uporządkowana struktura, 243

PostgreSQL, 10

postulat

- dostępu, 28
- fizycznej niezależności danych, 28
- informacyjny, 28
- logicznej niezależności danych, 28
- modyfikowania bazy danych przez widoki, 28
- modyfikowania danych na wysokim poziomie abstrakcji, 29
- niezależności dystrybucyjnej, 28
- niezależności ograniczeń, 29
- pełnego języka danych, 28
- postulat Codd'a
 - dotyczące integralności danych, 29
 - dotyczące przetwarzania danych, 28
 - dotyczące struktury danych, 27
- słownika danych, 29
- wartości, 29

R

Read Committed, 210

Read Uncommitted, 210

rekord, 20

- selekcja, 70

relacja, 24

Repeatable Read, 211

rola, 255

- przypisanie ról do użytkowników, 255
- public, 256
- tworzenie, 255
- usuwanie, 255

S

Schema Statements, 42

schemat, 22, 223

serwer

- bazodanowy, 9, 10, 22, 35, 50, 81, 82, 84, 132, 145, 153, 184, 201, 203, 206, 208, 223, 245, 256
- SQL 2011, 92, 137, 253, 257

Session Statements, 42

słowo kluczowe, 37, 38

- DESC, 61

- DISTINCT, 53, 118

SQL

- dialekt języka, 39
- SQL PL, 40
- SQL Server, 10
- SQL Server 2011, 10

standard

- ANSI, 40
- ANSI SQL, 11
- ANSI SQL3, 11
- ANSI SQL99, 9, 22
- SQL3, 41, 44, 111, 122, 260
- SQL99, 39

stronicowanie, 80
wierszy, 79
symbol, 15, 52, 83
nadużywanie, 184

T

tabela, 81, 95, 219
 łączenie, 89
 łącznikowa, 30
 pochodna, 157
 słownikowa, 30
 tworzenie, 222
 usuwanie, 222
Transaction Statements, 42
transact-SQL, 11
transakcja, 199, 200, 203
 cechy, 201
 izolowana, 200
 niepodzielna, 200
 poziom izolowania, 209
 przetwarzanie danych, 201, 204
 spójna, 200
 trwała, 201
 tryb jawnego zatwierdzenia, 202
 tryb niejawnego zatwierdzenia, 201
 właściwości, 199
 wycofanie, 204
 zagnieżdżanie, 205
 zatwierdzanie, 205
trend
 wyznaczanie, 165
tryb
 jawnego zatwierdzania transakcji, 202
 niejawnego zatwierdzania transakcji., 201
 odczytu niezatwierdzonego, 210
 odczytu zatwierdzonego, 210
 powtarzalnego odczytu, 211
 Repeatable Read, 212
 serializable, 212
 szeregowania, 212
T-SQL, 39
typ
 binarny, 42
 BINARY, 42
 BLOB, 42
 VARBINARY, 42
czas, 42
 TIME, 42
danych, 20, 42
data, 42
 DATE, 42
konwersja, 57
liczba, 42

INTEGER, 42
NUMERIC, 42
REAL, 42
SMALLINT, 42
znak, 42
CHAR, 42
NCHAR, 42
NVARCHAR, 42
VARCHAR, 42

U

uprawnienie
DELETE, 257
do modyfikowania kont użytkowników, 256
do modyfikowania ról, 256
do odczytywania metadanych obiektów, 256
do przejmowania obiektów na własność, 256
do tworzenia funkcji, 256
do tworzenia procedur, 256
do tworzenia schematów, 256
do tworzenia tabel, 256
do wykonywania kopii zapasowych baz
 danych, 256
dziedziczenie, 258
EXECUTE, 257
INSERT, 257
nadawanie, 253, 257
obiektywne, 257
odbieranie, 253, 257
odbieranie uprawnień w serwerze SQL 2011, 257
przekazywanie, 260
REFERENCE, 257
SELECT, 257
systemowe, 256
UPDATE, 257

W

wartość
CASCADE, 230
domyślna, 185, 227
FALS, 69
nieznana, 42, 150
NO ACTION, 230
NULL, 42, 68, 186
odczytanie, 146
przypisanie, 146
SET DEFAULT, 230
SET NULL, 230
TRUE, 69
UNKNOWN, 68, 69

- warunek
 - logiczny, 228
 - złożony, 75
 - złączenia, 96
- widok, 235, 243
 - grupujący dane, 240
 - modyfikowanie, 238
 - modyfikowanie danych, 240
 - tworzenie, 235
 - usuwanie, 235
 - zagnieżdżony, 239
- wiersz, 20
 - grupowanie, 117
 - łączenie, 112
 - stronicowanie, 79
 - wybór, 67
 - zliczanie, 118
- wskaźnik, 81
- współbieżność, 199, 207
- wynik
 - formatowanie, 58
 - sortowanie, 60
- wyrażenie, 54, 121, 193
 - tabelaryczne, 159

Z

- zakleszczenie, 208
- cykliczne, 208
- konwersji, 208

- zapytanie
 - łączenie wyników, 107
 - wewnętrzne, 146
 - wstawianie wyników, 187
 - zagnieżdżone, 145
- zasada
 - minimalnych uprawnień, 261
 - sprzeczności, 67
 - tożsamości, 67
 - wyłączonego środka, 67
- zbiór
 - przetwarzanie, 34
 - rekordów, 20
- złączenie, 155
 - FULL OUTER JOIN, 98
 - krzyżowe, 98
 - LEFT OUTER JOIN, 97
 - lewostronne, 97
 - naturalne, 89, 92
 - nienaturalne, 89, 92
 - nierównościowe, 94, 95
 - obustronne, 98
 - określenie kolejności, 103
 - prawostronne, 97
 - RIGHT OUTER JOIN, 97
 - równościowe, 94
 - tabeli z nią samą, 104
 - wielokrotne, 100
 - zewnętrzne, 96
- zmienna, 148

Bazy danych są dosłownie wszędzie. Trudno sobie dziś bez nich wyobrazić funkcjonowanie nowoczesnej biblioteki, choćby najmniejszego sklepu internetowego, biura rachunkowego czy nawet niewielkiego serwisu WWW. Użytkownicy korzystający z baz danych często nie mają nawet pojęcia, w jaki sposób odbywa się dostęp do informacji i jaki mechanizm jest za to odpowiedzialny. Na ignorancję tę nie mogą sobie jednak pozwolić osoby odpowiedzialne za tworzenie i konserwowanie baz danych oraz zarządzanie nimi. Powinny one znać przynajmniej jeden z popularnych serwerów bazodanowych i sprawnie posługiwać się językiem SQL stanowiącym standardowe narzędzie komunikacji z relacyjnymi bazami.

Jeśli pragniesz dołączyć do ekskluzywnego grona administratorów baz danych lub chcesz zostać programistą aplikacji bazodanowych, lecz przeszkadza Ci brak znajomości SQL-a, sięgnij po książkę „Praktyczny kurs SQL. Wydanie II”. W prosty i przystępny sposób prezentuje ona podstawowe pojęcia i zasady rządzące relacyjnym modelem baz danych, a także najważniejsze cechy i konstrukcje języka SQL oraz metody ich wykorzystywania. Lektura książki umożliwi Ci poznanie instrukcji odpowiedzialnych za odczytywanie danych z bazy i ich zapisywanie oraz modyfikację, jak również tworzenie baz i zmianę ich struktury. Poznasz też sposoby tworzenia ról i kont użytkowników oraz zarządzania ich uprawnieniami. Twoją wiedzę ugruntuj praktyczne zadania kończące każdy rozdział, a zamieszczone na końcu książki rozwiązania pomogą skorygować ewentualne błędy.

- **Teoretyczne podstawy funkcjonowania baz danych**
- **Historia języka SQL i obowiązujące standardy zapytań**
- **Odczytywanie, przeszukiwanie, łączenie i grupowanie danych**
- **Korzystanie z podzapytań**
- **Zapisywanie, modyfikacja i usuwanie danych**
- **Transakcje i równoległy dostęp do danych**
- **Tworzenie baz danych i modyfikacja ich struktury**
- **Korzystanie z widoków i indeksów**
- **Zarządzanie użytkownikami, rolami i prawami dostępu do baz danych**

Dowiedz się, jak tworzyć relacyjną bazę danych i zarządzać nią za pomocą języka SQL.

Nr katalogowy: **6224**



Księgarnia internetowa:
<http://helion.pl>



Zamówienia telefoniczne:
0 801 339900
0 601 339900



Helion

Sprawdź najnowsze promocje:

☛ <http://helion.pl/promocje>

Książki najchętniej czytane:

☛ <http://helion.pl/bestsellery>

Zamów informacje o nowościach:

☛ <http://helion.pl/nnowosci>

Helion SA

ul. Kościuszki 1c, 44-100 Gliwice

tel.: 32 230 98 63

e-mail: helion@helion.pl

<http://helion.pl>

helion.pl
księgarnia
internetowa

Cena 47,00 zł

ISBN 978-83-246-3373-9



9 788324 163373