

DANUTA MENDRALA
MARCIN SZELIGA

WYDANIE III

SQL



PRAKTYCZNY
KURS

Helion 

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz wydawca dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz wydawca nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Redaktor prowadzący: Małgorzata Kulik

Projekt okładki: Studio Gravite / Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki

Grafika na okładce została wykorzystana za zgodą Shutterstock.com

Helion S.A.

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 231 22 19, 32 230 98 63

e-mail: helion@helion.pl

WWW: <https://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<https://helion.pl/user/opinie/pksq3v>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Kody źródłowe wybranych przykładów dostępne są pod adresem:

<https://ftp.helion.pl/przyklady/pksq3v.zip>

ISBN: 978-83-283-9236-6

Copyright © Helion S.A. 2015, 2022

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Spis treści

- Wstęp 9**
 - Serwery bazodanowe 10
 - O książce 10
 - SQL Server firmy Microsoft 11
 - Instalacja 12
 - Przykładowa baza danych 16
 - Konwencje i oznaczenia 17
- Część I Trochę teorii, czyli modele i standardy 19**
- Rozdział 1. Relacyjny model baz danych 21**
 - Tabele jako zbiory danych 21
 - Kolumny mają niepowtarzalne nazwy i zawierają określone typy danych 22
 - Wiersze powinny być unikatowe 23
 - Kolejność kolumn jest bez znaczenia 23
 - Kolejność wierszy jest bez znaczenia 24
 - Bazy danych 24
 - Trzy modele baz danych: relacyjny, obiektowy i jednorodny 24
 - Model jednorodny 25
 - Model relacyjny 25
 - Model obiektowy 29
 - Założenia relacyjnego modelu baz danych 30
 - Postulaty Codd’a dotyczące struktury danych 31
 - Postulaty Codd’a dotyczące przetwarzania danych 31
 - Postulaty Codd’a dotyczące integralności danych 32
 - Normalizacja 32
 - Podsumowanie 35
 - Zadania 36
- Rozdział 2. Standardy języka SQL 37**
 - Strukturalny język zapytań 37
 - Przetwarzanie zbiorów a przetwarzanie pojedynczych danych 38
 - Język strukturalny a język proceduralny 39
 - Język interpretowany a język kompilowany 39
 - Składnia języka SQL 41
 - Dialekty języka SQL 43

Standardy ANSI	44
Historia	44
SQL3	46
Podsumowanie	50
Zadania	50
Część II Pobieranie danych, czyli instrukcja SELECT	51
Rozdział 3. Odczytywanie danych z wybranej tabeli	53
Klauzula FROM	53
W pełni kwalifikowane nazwy obiektów	54
Wybieranie kolumn	55
Eliminowanie duplikatów	57
Wyrażenia	58
Operatory arytmetyczne	59
Łączenie danych tekstowych	60
Funkcje systemowe	60
Formatowanie wyników	64
Aliasy	64
Stałe (literały)	65
Sortowanie wyników	66
Sortowanie danych tekstowych	69
Podsumowanie	70
Zadania	70
Rozdział 4. Wybieranie wierszy	73
Logika trójwartościowa	73
Wartość NULL	74
Operatory logiczne	74
Klauzula WHERE	76
Standardowe operatory porównania	77
Operatory SQL	78
Złożone warunki logiczne	82
Klauzula TOP	85
Stronicowanie wierszy	87
Podsumowanie	88
Zadania	89
Rozdział 5. Łączenie tabel i wyników zapytań	91
Złączenia naturalne i nienaturalne	91
Klucze obce	92
Aliasy	95
Złączenia równościowe i nierównościowe	96
Złączenia zewnętrzne	98
Złączenie lewostronne	99
Złączenie prawostronne	99
Złączenie obuustronne	99
Złączenie krzyżowe (iloczyn kartezjański)	100
Złączenia wielokrotne	102
Określanie kolejności złączeń	104
Złączenie tabeli z nią samą	106
Eliminacja duplikatów	107
Klucze obce w obrębie jednej tabeli	108

Łączenie wyników zapytań	109
Suma	109
Część wspólna	112
Różnica	112
Łączenie wierszy i wyników funkcji tabelarycznych	113
Operator APPLY	115
Podsumowanie	116
Zadania	117
Rozdział 6. Grupowanie wierszy	119
Funkcje grupujące	119
Funkcja COUNT()	120
Funkcje SUM() i AVG()	121
Funkcje MIN() i MAX()	122
Inne funkcje grupujące	123
Wyrażenia	124
Klauzula GROUP BY	125
Kolejność wykonywania klauzuli GROUP BY	128
Operatory CUBE i ROLLUP	129
Operator GROUPING SETS	132
Operatory PIVOT i UNPIVOT	134
PIVOT	134
UNPIVOT	137
Klauzula HAVING	138
Podsumowanie	141
Zadania	141
Rozdział 7. Partycjonowanie wierszy oraz funkcje rankingu, analityczne i okienkowe	143
Partycjonowanie	143
Klauzula OVER	144
Partycjonowanie danych	147
Porządkowanie danych	149
Funkcje rankingu	149
Okienka	151
Funkcje okienkowe	154
Funkcje analityczne	156
Podsumowanie	158
Zadania	158
Rozdział 8. Podzapytania	161
Czym są podzapytania?	161
Podzapytania jako zmienne	162
Podzapytania niepowiązane	162
Podzapytania powiązane	168
Podzapytania jako źródła danych	173
Tabele pochodne	174
CTE	176
Wyznaczanie trendów	182
Operatory	185
Operator EXISTS	186
Operator ANY lub SOME	189
Operator ALL	193
Podsumowanie	195
Zadania	195

Rozdział 9. Wydajność zapytań	197
Wykonywanie zapytań przez serwery bazodanowe	197
Kolejność wykonywania klauzul zapytania	198
Plany wykonania zapytań	199
Wydajne wyszukiwanie danych za pomocą argumentów SARG	203
Poprawa wydajności złączeń	207
Wydajne grupowanie i partycjonowanie danych	208
Podsumowanie	209
Zadania	209

Część III Modyfikowanie danych, czyli instrukcje INSERT, UPDATE, DELETE oraz MERGE 211

Rozdział 10. Modyfikowanie danych	213
Wstawianie danych	213
Klucze podstawowe	214
Wartości domyślne	215
Wartość NULL	216
Konstruktor wierszy	217
Wstawianie wyników zapytań	218
Usuwanie danych	221
Instrukcja DELETE	221
Instrukcja TRUNCATE TABLE	223
Aktualizowanie danych	224
Jednoczesne aktualizowanie wielu kolumn	224
Wyrażenia	225
Aktualizowanie danych wybranych na podstawie danych z innych tabel	226
Aktualizowanie danych za pomocą wyrażeń odwołujących się do innych tabel	227
Instrukcja MERGE	227
Podsumowanie	229
Zadania	230

Rozdział 11. Transakcje i współbieżność	231
Właściwości transakcji	231
Transakcyjne przetwarzanie danych	233
Tryb jawnego zatwierdzania transakcji	234
Rozpoczynanie transakcji	234
Wycofywanie transakcji	236
Zatwierdzanie transakcji	237
Zagnieżdżanie transakcji	237
Punkty przywracania	238
Współbieżność	239
Blokady	239
Zakleszczenia	240
Poziomy izolowania transakcji	241
Model optymistyczny	246
Model pesymistyczny	247
Podsumowanie	248
Zadania	248

Część IV Tworzenie baz danych, czyli instrukcje CREATE, ALTER i DROP 249

Rozdział 12. Bazy danych i tabele	251
Tworzenie i usuwanie baz danych	251
Tworzenie i usuwanie tabel	254
Schematy	255
Zmiana struktury tabeli	256
Ograniczenia	256
NOT NULL	257
Klucz podstawowy	257
Niepowtarzalność	259
Wartość domyślna	260
Warunek logiczny	260
Klucz obcy	261
Ograniczenia a wydajność instrukcji modyfikujących i odczytujących dane	264
Podsumowanie	265
Zadania	266

Rozdział 13. Widoki i indeksy	267
Widoki	267
Tworzenie i usuwanie widoków	267
Modyfikowanie widoków	270
Korzystanie z widoków	270
Zalety widoków	275
Indeksy	276
Tworzenie, modyfikowanie i usuwanie indeksów	278
Porządkowanie indeksów	281
Podsumowanie	281
Zadania	282

Część V Uprawnienia użytkowników, czyli instrukcje GRANT i REVOKE 283

Rozdział 14. Nadawanie i odbieranie uprawnień	285
Konta użytkowników	285
Zakładanie i usuwanie kont użytkowników	286
Role	287
Tworzenie i usuwanie ról	287
Przypisywanie ról do użytkowników	287
Specjalna rola Public	288
Uprawnienia	288
Nadawanie i odbieranie uprawnień	289
Dziedziczenie uprawnień	290
Przekazywanie uprawnień	292
Zasada minimalnych uprawnień	293
Podsumowanie	293
Zadania	294

Dodatki

Dodatek A Rozwiązania zadań	297
Skorowidz	333

Rozdział 9.

Wydajność zapytań

- ♦ W jaki sposób serwery bazodanowe wykonują zapytania?
- ♦ W jakiej kolejności wykonywane są poszczególne klauzule zapytań?
- ♦ Czym jest plan wykonania zapytania i jak go odczytać?
- ♦ Co oznacza akronim SARG?
- ♦ Jakie indeksy są przydatne do wyszukiwania wierszy?
- ♦ Co to znaczy, że indeks zawiera jakieś zapytanie?
- ♦ Które kolumny powinny być poindeksowane w celu poprawy wydajności złączeń?
- ♦ Jak poprawić wydajność zapytań grupujących lub partycjonujących wiersze?
- ♦ Co oznacza akronim POC?

Wykonywanie zapytań przez serwery bazodanowe

Chociaż szczegóły sposobów, w jakie poszczególne serwery bazodanowe wykonują zapytania, są różne, główne etapy tego procesu wyglądają podobnie. Ponieważ podstawowa wiedza na ten temat jest niezbędna do pisania wydajnych zapytań, poniżej przedstawione zostały poszczególne operacje wykonywane przez SQL Server:

1. Aplikacja kliencka łączy się z serwerem bazodanowym. Po udanym połączeniu nawiązana zostaje dwukierunkowa sesja, w ramach której odbywać się będzie komunikacja pomiędzy serwerem a klientem.
2. Klient wysyła do serwera instrukcję języka SQL.

3. Instrukcja ta zostaje odebrana, a następnie serwer bazodanowy musi opracować plan jej wykonania. Przeprowadzana w tym celu optymalizacja jest bardzo skomplikowanym i zależnym od danego serwera (a nawet od jego wersji) procesem, którego omówienie wykracza poza zakres tej książki. Powinienes jednak wiedzieć, że optymalizacja jest bardzo czasochłonnym, silnie obciążającym procesor i wymagającym dużej ilości pamięci procesem, a więc wiele serwerów bazodanowych przechowuje w pamięci raz zoptymalizowane plany wykonania zapytań w celu ich ponownego użycia.
4. Dysponując znalezionym planem wykonania, serwer bazodanowy może przystąpić do wykonywania instrukcji. Prawie zawsze wiąże się to z koniecznością odczytania pewnych danych.
5. Serwery bazodanowe przechowują dane w specjalnych jednostkach, w przypadku serwera SQL są to ośmiokilobajtowe strony. Strony są też jednostkami odczytu i zapisu danych, a więc serwer zapisuje oraz odczytuje jedną lub więcej stron, a nie poszczególne wiersze czy całe tabele. Do wykonania odebranej od klienta instrukcji serwer będzie więc musiał odczytać zawierające niezbędne do wykonania zapytania strony.
6. Jeżeli strony te znajdowały się już w buforze (pamięci RAM), wykonana zostanie operacja logicznego odczytu. W przeciwnym wypadku strony zostaną wczytane z dysku do bufora — taki odczyt nazywany jest odczytem fizycznym. Ponieważ pamięć jest znacznie szybsza od dysków, fizyczne odczyty są bardzo mało wydajne¹. W związku z tym, żeby zapewnić jak najwyższą wydajność, należy zminimalizować liczbę fizycznych odczytów poprzez wyposażenie serwera w odpowiednią (wystarczającą do zbuforowania wszystkich danych) ilość pamięci RAM.
7. Wynik wykonania instrukcji języka SQL (w przypadku zapytania będzie to zbiór wierszy) jest wysyłany do aplikacji klienckiej.

Kolejność wykonywania klauzul zapytania

Chociaż serwery bazodanowe optymalizują zapytania przed ich wykonaniem, proces ten nie może mieć wpływu na wynik zapytania (wykonanie zapytania według różnych planów, np. poprzez złączenie tabel w różnej kolejności albo poprzez pogrupowanie wierszy, a następnie złączenie otrzymanych w ten sposób grup czy też złączenie tabel, a następnie pogrupowanie wierszy, musi zawsze skutkować zwróceniem tego samego wyniku).

Poszczególne klauzule instrukcji SELECT są wykonywane zawsze w tej samej kolejności. Jeżeli któraś z opcjonalnych klauzul nie występuje, dany krok jest po prostu pomijany.

¹ Automatyczne buforowanie danych jest powodem, dla którego ponowne wykonanie tego samego zapytania może być znacznie szybsze.

Rezultatem wykonania każdego kroku jest zbiór pośredni — wirtualna tabela, która nie jest dostępna dla użytkownika. **Kolejny krok jest wykonywany tylko w oparciu o zbiór pośredni będący rezultatem wykonania poprzedniego kroku**, a zbiór pośredni, rezultat wykonania ostatniego kroku, jest zwracany użytkownikowi.

Tak więc chociaż faktyczny sposób, w jaki serwer bazodanowy wykona nasze zapytanie, może być (i najczęściej będzie) inny niż przedstawiona poniżej ogólna kolejność wykonywania zapytań, warto znać tę logiczną kolejność wykonywania poszczególnych klauzul.

W przypadku zapytań niegrupujących danych ich klauzule wykonywane są w następującej kolejności:

1. Najpierw serwer musi pobrać potrzebne do wykonania zapytania dane, a więc wykona klauzulę FROM oraz, jeżeli istnieją, klauzule JOIN i operatory APPLY.
2. Następnie wybrane zostaną wiersze spełniające warunki klauzuli WHERE.
3. Dla otrzymanych w wyniku wykonania dwóch poprzednich punktów wierszy wykonane zostaną wyrażenia zdefiniowane w klauzuli SELECT.
4. Na końcu otrzymane wiersze zostaną posortowane, o ile w zapytaniu wystąpiła klauzula ORDER BY.

Wykonanie zapytania grupującego wymaga wykonania dodatkowych operacji:

1. Punkty pierwszy i drugi są wykonywane w taki sam sposób jak dla zapytań niegrupujących.
2. Przed wykonaniem klauzuli SELECT wiersze kandydujące są grupowane (wykonywana jest klauzula GROUP BY).
3. Otrzymane w wyniku grupowania wiersze są filtrowane, o ile w zapytaniu wystąpiła klauzula HAVING.
4. Dla otrzymanych w wyniku wykonania poprzednich punktów wierszy wykonane zostaną wyrażenia zdefiniowane w klauzuli SELECT.
5. Na końcu otrzymane wiersze zostaną posortowane, o ile w zapytaniu wystąpiła klauzula ORDER BY.

Plany wykonania zapytań

Umiejętność czytania i analizowania planów wykonania zapytań jest niezbędna każdemu, kto chce pisać nie tylko poprawne (zwracające właściwe wyniki), ale również wydajne (zwracające te wyniki w najszybszy możliwy sposób) zapytania. Wynika to z faktu, że to samo zapytanie może być wykonane na bardzo dużo różnych sposobów (liczba możliwych sposobów wykonania tego samego zapytania zależy od samego zapytania i od serwera bazodanowego, ale nawet nieskomplikowane zapytania odczytujące dane z kilku tabel mogą być wykonane na kilkadziesiąt, a nawet kilkaset

różnych sposobów). Zadaniem serwera bazodanowego jest znalezienie wystarczająco dobrego sposobu (planu) wykonania zapytania. Żeby mu to umożliwić, powinniśmy nie tylko unikać typowych błędów (takich jak używanie argumentów uniemożliwiających efektywne skorzystanie z indeksów), ale również pisać zapytania w sposób ułatwiający znalezienie optymalnych planów ich wykonania. Najprostszym sposobem sprawdzenia, czy nasze zapytanie wykonywane jest wydajnie, jest właśnie analiza planu jego wykonania.

SQL Server pozwala odczytywać plany wykonania zapytań w formie tekstowej oraz graficznej. Dodatkowo możliwe jest poznanie szacowanych oraz faktycznych planów wykonania. Ponieważ szczegółowe przedstawienie kwestii analizy planów wykonania zapytań wykracza poza zakres tej książki, ograniczymy się do przedstawiania faktycznych planów wykonania zapytań w formie graficznej.

Zacznijmy od wyjaśnienia tego, czym jest plan wykonania zapytania. Plan wykonania reprezentuje opracowaną przez serwer bazodanowy strategię odczytania i przetworzenia danych na potrzeby wykonania zapytania. Składa się ona z serii iteratorów, z których każdy wykonuje pojedynczą operację, taką jak odczytanie danych, posortowanie wierszy, złączenie tabel itd.

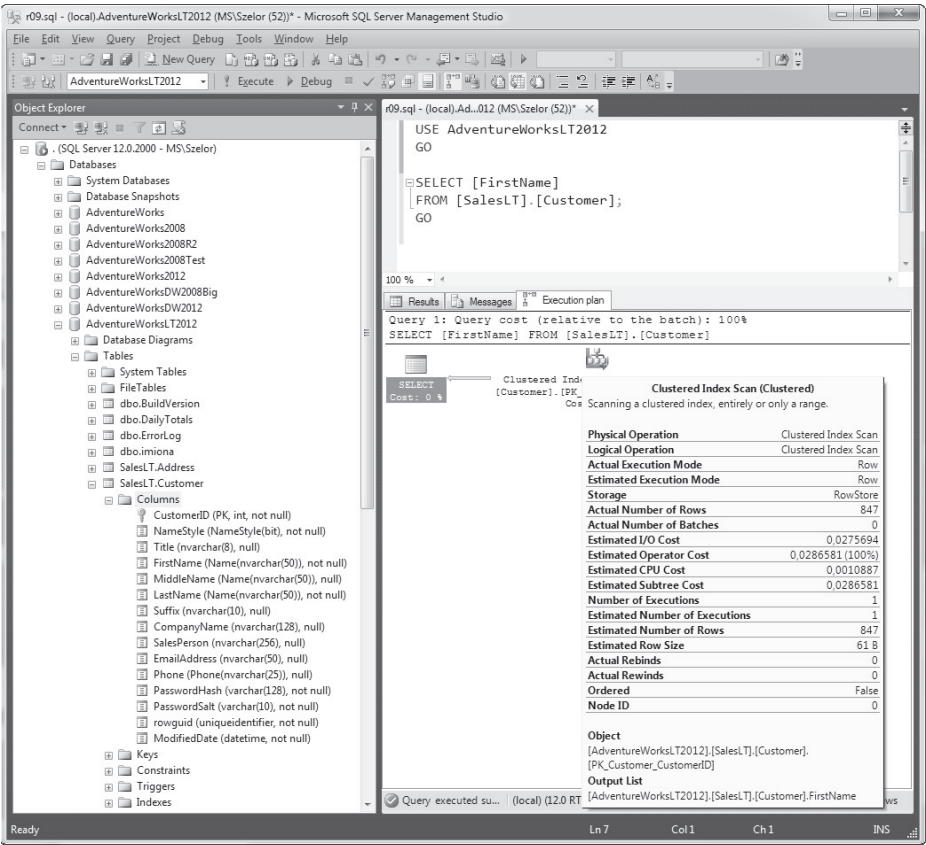
Plany wykonania zapytań można czytać w dwóch kierunkach:

1. Od lewa do prawa — ten kierunek reprezentuje logikę wykonywania zapytania;
2. Lub od prawa do lewa — ta kolejność repetuje przepływ danych pomiędzy kolejnymi iteratorami.

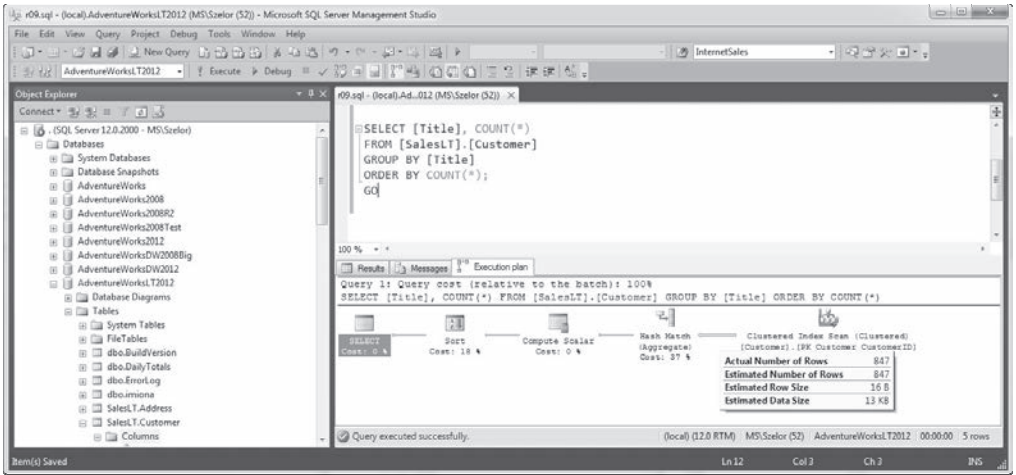
Żeby wyświetlić graficzny plan wykonania zapytania, należy kliknąć znajdujący się na pasku zadań konsoli SSMS przycisk *Include Actual Execution Plan* lub nacisnąć kombinację klawiszy *Ctrl+M*, a następnie uruchomić analizowane zapytanie (rysunek 9.1).

Przyjrzyjmy się nieco bardziej rozbudowanemu planowi wykonania zapytania. Przedstawiony na rysunku 9.2. plan wykonania należy przeczytać następująco:

1. Przeskanowany został indeks zgrupowany (tabela [SalesLT].[Customer]).
2. Odczytanych w ten sposób 847 wierszy (umieszczając kursor myszki nad reprezentującymi przepływ danych strzałkami, wyświetlimy dodatkowe informacje na temat liczby i rozmiaru wierszy) zostało pogrupowanych.
3. Dla każdej grupy wyliczona została funkcja COUNT.
4. Wiersze zostały posortowane.
5. Na końcu wiersze zostały zwrócone do aplikacji klienckiej.



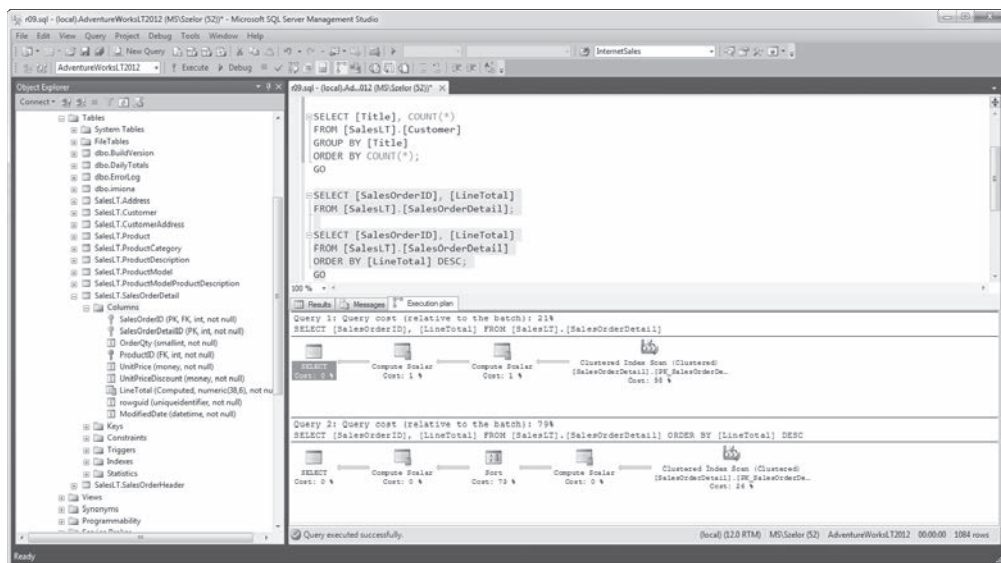
Rysunek 9.1. Najprostszy plan wykonania zapytania — potrzebne dane zostały odczytane za pomocą iteratora Clustered Index Scan (szczegóły operacji możemy poznać, umieszczając kursor myszki nad danym iteratorem), a następnie zwrócone do aplikacji klienckiej



Rysunek 9.2. Składający się z pięciu iteratorów plan wykonania zapytania grupującego dane

SQL Server ocenia i podaje koszt wykonania każdej operacji — np. w poprzednim przykładzie koszt pogrupowania wierszy wyniósł 37% kosztów wykonania całego zapytania. Ta wiedza pozwala nam na wdrożenie względnie prostej, a jednocześnie skutecznej metody optymalizacji zapytań, polegającej na eliminowaniu z planów wykonania najkosztowniejszych operacji.

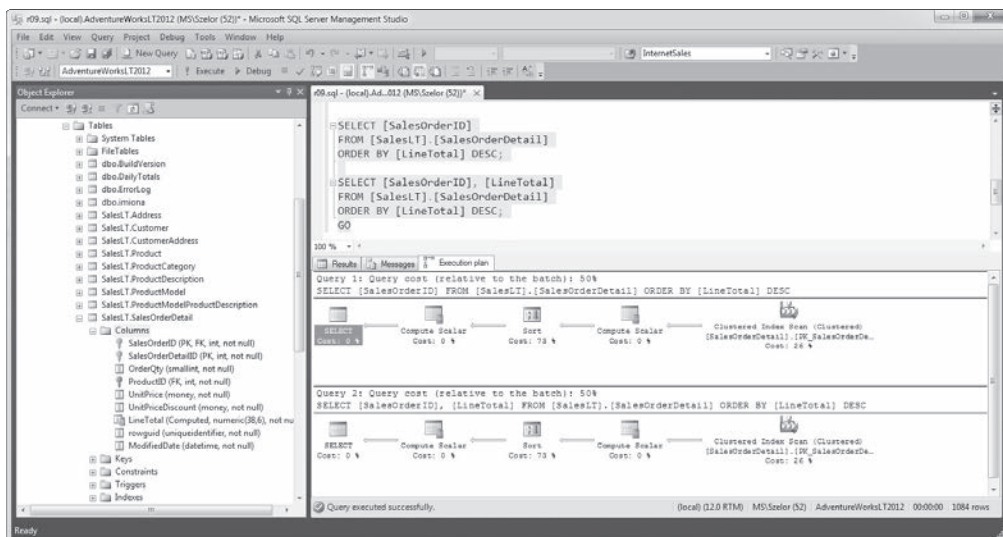
Co więcej, jeżeli jednocześnie uruchomimy kilka zapytań (np. kilka wersji tego samego zapytania), serwer SQL zwróci informację o procentowym rozkładzie kosztów wykonania całego wsadu (zbiór jednocześnie wysłanych do serwera zapytań nazywa się wsadem). Rysunek 9.3 ilustruje tę funkcjonalność.



Rysunek 9.3. Drugie zapytanie (to z klauzulą `ORDER BY`) okazało się cztery razy bardziej kosztowne od pierwszego. Różnica wynika z tego, że wykonując drugie zapytanie, serwer musiał dodać kosztowny iterator `SORT`

Przekonał się właśnie, że **sortowanie może wielokrotnie wydłużyć czas wykonania zapytania**, a więc jeżeli wynik zapytania nie musi być posortowany, nie należy umieszczać w nim klauzuli `ORDER BY`.

Porównajmy jeszcze koszty wykonania zapytania, w którym użyta do sortowania kolumna nie została dołączona do wyniku z zapytaniem zawierającym tę kolumnę (rysunek 9.4) — okaże się, że są one identyczne, a więc **niewymienienie w klauzuli `SELECT` kolumn użytych do sortowania nie skraca czasu wykonania zapytania**.



Rysunek 9.4. *Chociaż analiza planów wykonania zapytań przez konkretny serwer bazodanowy nie jest tematem tej książki, to warto wiedzieć, że kolumny użyte do sortowania muszą być odczytane tak samo jak kolumny wymienione w klauzuli SELECT. Z tego powodu większość serwerów bazodanowych wykona oba powyższe zapytania w tym samym czasie, a koszt wykonania każdego z nich przez serwer SQL wyniósł dokładnie 50% kosztu wykonania całego wsadu*

Wydajne wyszukiwanie danych za pomocą argumentów SARG

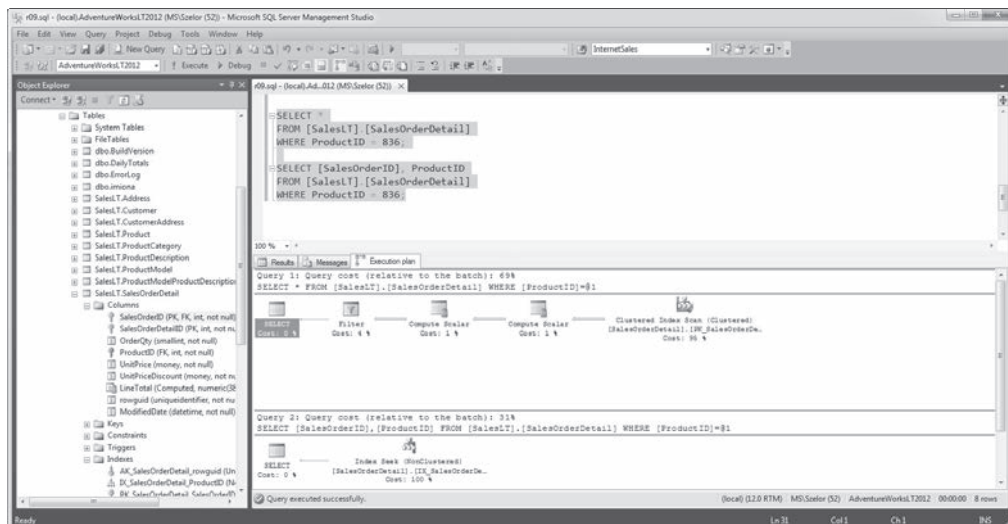
Dane mogą być zapisane:

1. W tabelach — dla uproszczenia i zgodnie z relacyjnym modelem baz danych przyjmijmy, że kolejność wierszy tabeli jest nieistotna, a więc nie są one w żaden sposób posortowane².
2. W indeksach — aby uniknąć analizowania różnic pomiędzy poszczególnymi serwerami bazodanowymi, przyjmijmy, że indeks przypomina skorowidz książki — dołączaną na jej końcu, alfabetycznie uporządkowaną listę haseł z odnośnikami do numerów stron, na których dane hasło jest opisane. W przypadku baz danych odpowiednikiem hasła będzie wybrana kolumna (lub kolumny) tabeli, czyli tzw. klucz indeksu, a odnośnikiem wskaźnik³. Podsumowując, **każdy klucz indeksu posiada wskaźnik do wiersza tabeli zawierającego wartości pozostałych (niekluczowych) kolumn, a klucze indeksu są zawsze posortowane.**

² Wiele serwerów bazodanowych porządkuje wiersze tabeli według wartości klucza podstawowego. Taka posortowana tabela nazywana jest indeksem zgrupowanym.

³ Więcej informacji na temat indeksów znajduje się w rozdziale 13.

Jeżeli odczytywana tabela nie jest poindeksowana, serwer bazodanowy wykonując dowolne odwołujące się do tej tabeli zapytanie, będzie musiał odczytać ją w całości. Tego typu sytuacja zachodzi też, gdy odczytujemy wszystkie kolumny tabeli za pomocą symbolu *. Żeby się o tym przekonać, wystarczy porównać plany wykonania dwóch pokazanych na rysunku 9.5 zapytań:



Rysunek 9.5. Wyszukanie w indeksie kluczy spełniających warunek z klauzuli WHERE i odczytanie tylko czterech wierszy tabeli okazało się w tym przypadku trzy razy szybsze niż odczytanie całej tabeli i wybranie czterech wierszy spełniających podany warunek

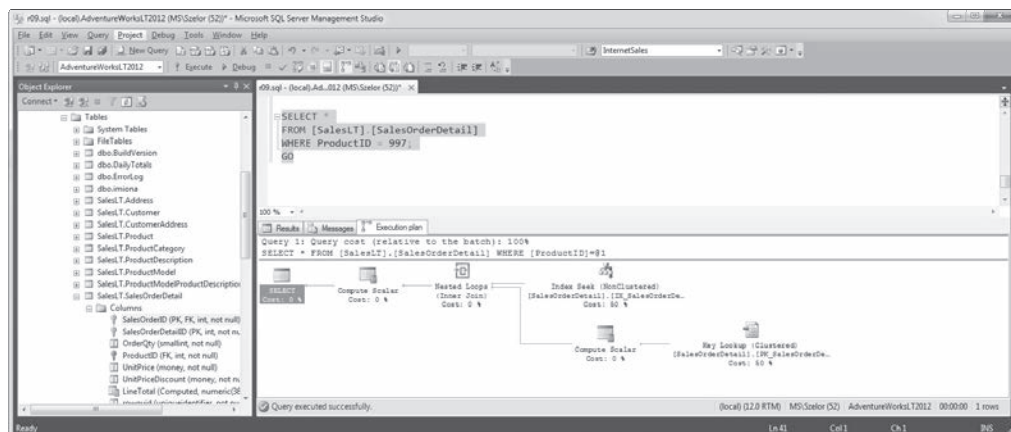
Podsumujmy — tabela [SalesLT].[SalesOrderDetail] ma założony indeks na kolumnie ProductID. Indeks ten zawiera, oprócz identyfikatorów produktów, identyfikatory zamówień. Nie zawiera natomiast pozostałych kolumn tej tabeli. Z tego powodu wykonanie pierwszego, zwracającego tylko cztery wiersze, zapytania wymagało odczytania całej tabeli, podczas gdy do znalezienia odpowiednich danych w indeksie wystarczyło go przeszukać. O takich zapytaniach jak te drugie mówimy, że zawierają się one w indeksie, ponieważ wszystkie potrzebne do ich wykonania dane zapisane są w założonym dla tabeli indeksie.



Wskazówka

Utworzenie zawierającego zapytanie indeksu jest najprostszym i najskuteczniejszym sposobem na poprawę wydajności tego zapytania. Nie oznacza to jednak, że powinniśmy tworzyć indeksy zawierające wszystkie zapytania. Ponieważ serwer bazodanowy automatycznie synchronizuje dane w indeksach z danymi w tabeli, liczba indeksów zdefiniowanych dla pojedynczej tabeli nie powinna przekraczać 10.

Jeżeli istnieje indeks zawierający zapytanie, serwer bazodanowy zawsze go użyje. Zupełnie inaczej wygląda użycie przez serwery bazodanowe indeksów podczas wykonywania zapytań, które się w nich nie zawierają (rysunek 9.6).



Rysunek 9.6. Plan wykonania zapytania poprzez przeszukiwanie indeksu i pobranie brakujących w tym indeksie danych z tabeli

Na planie wykonania drugiego zapytania widoczny jest operator Key Lookup symbolizujący odczytywanie danych z tabeli [SalesLT].[SalesOrderDetail]. Operacja ta musiała być przeprowadzona, bo indeks zawierał tylko identyfikatory produktów i zamówień, a zapytanie miało zwrócić wszystkie dane wybranego zamówienia. Takie sięganie po dane jest na tyle kosztowne, że **jeżeli zapytanie zwraca więcej niż kilka procent wierszy tabeli, serwery bazodanowe przestają używać indeksów i odczytują całą tabelę, tak jakby użyta do wyszukiwania kolumna nie była zindeksowana.**



Wskazówka

Wiedząc, w jaki sposób serwery bazodanowe odczytują dane, możemy teraz uzasadnić uwagę z jednego z poprzedniego rozdziałów, w której napisaliśmy, że **używając symbolu ***, **zmuszamy serwer bazodanowy do odczytania wszystkich kolumn tabeli, co może wielokrotnie wydłużyć czas wykonywania zapytania.** Ponieważ większość zapytań zwraca więcej niż 1% wierszy tabeli, posługując się symbolem *, powodujemy, że serwery bazodanowe nie korzystają z istniejących indeksów.

Skoro koszt (a więc i czas) wykonania zapytań z użyciem indeksów jest wielokrotnie niższy niż koszt odczytania tych samych danych z tabel, **powinniśmy tak pisać zapytania, żeby serwery bazodanowe korzystały z indeksów przy ich wykonywaniu.**



Wskazówka

Skrót SARG (ang. *Search ARGuments*) oznacza warunki wyszukiwania, czyli takie warunki logiczne, które pozwalają serwerom bazodanowym na wybranie właściwych wierszy poprzez przeszukiwanie indeksu, a nie odczytanie całej tabeli.

Ponieważ użycie indeksów może być bardzo kosztowne⁴, serwery bazodanowe, zanim się na nie zdecydują, szacują koszt wykonania zapytania, używając do tego danych statystycznych. Jeżeli nie będą w stanie oszacować tego kosztu, wybiorą najbezpieczniejsze rozwiązanie, czyli odczytają tabelę.

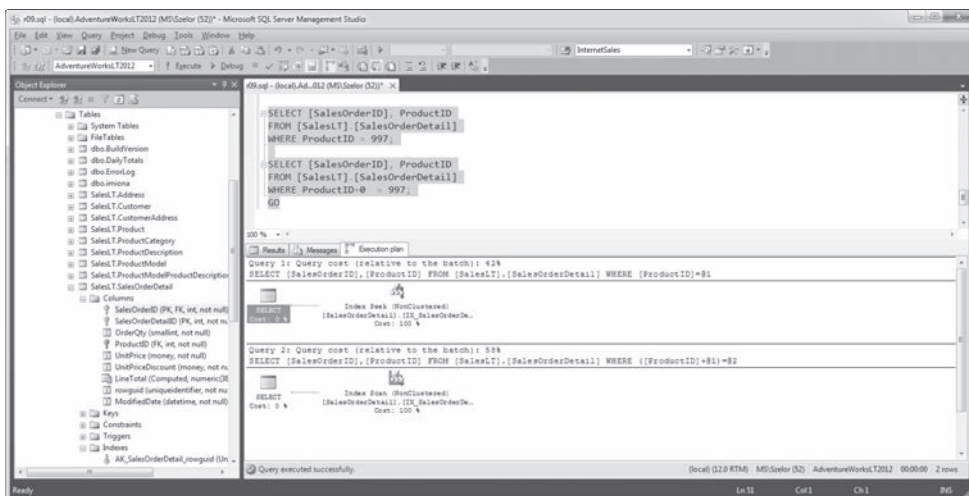
⁴ Z powodu opisanej wcześniej operacji *Key Lookup*, polegającej na odczytywaniu dla każdego klucza indeksu odpowiadającego mu wiersza tabeli.

W poniższych punktach zostały opisane przypadki, w których serwer bazodanowy może nie mieć możliwości oszacowania kosztu wykonania zapytania, a więc nie skorzysta z indeksów i czas wykonania zapytań będzie wielokrotnie dłuższy:

1. Zanegowanie warunku logicznego (użycie operatora NOT albo operatora <>), o ile serwer bazodanowy nie potrafi automatycznie przekształcić warunku logicznego w równoważny warunek pozytywny. Przykładem takiego przekształcenia może być zastąpienie przez serwer SQL warunku NOT CustomerID >3 warunkiem CustomerID <= 3:

```
SELECT *
FROM [SalesLT].[SalesOrderDetail]
WHERE ProductID <> 999;
```

2. Umieszczenie nazwy kolumny w ramach dowolnego wyrażenia, nawet jeżeli to wyrażenie nie zmienia wyniku testu logicznego. Z tego powodu, choć oba pokazane na rysunku 9.7 zapytania zwracają te same dane, to pierwsze zostanie wykonane czterokrotnie szybciej.



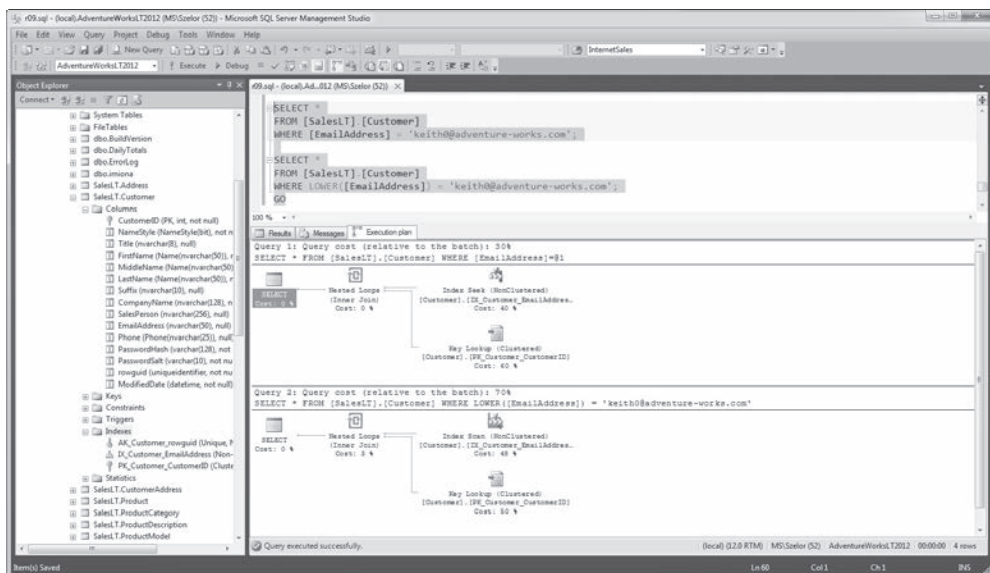
Rysunek 9.7. Użycie nazwy kolumny jako części dowolnego wyrażenia oznacza, że serwer bazodanowy będzie musiał odczytać całą indeks (albo nawet całą tabelę), żeby mieć pewność, że zwrócił wszystkie wiersze spełniające tego typu warunek

3. Użycie nazwy kolumny jako argumentu dowolnej funkcji (rysunek 9.8).
4. Użycie do wyboru wierszy wzorca zaczynającego się od symbolu wieloznacznego, np. Name LIKE N'%b'.



Wskazówka

Przykładowa baza danych jest bardzo mała i powyższe zapytania odwołują się do tabel liczących zaledwie kilkadziesiąt wierszy każda. Tylko dlatego różnice w wydajności poprawnie napisanych (umożliwiających przeszukiwanie indeksu) i nieoptymalnych zapytań są tak małe. Tymczasem swoją prawdziwą siłę indeksy pokazują w przypadku dużych, liczących tysiące czy miliony wierszy, tabel. Wtedy znalezienie wybranych wierszy w indeksie nadal wymaga odczytania takiej samej ilości danych, podczas gdy odczytanie całej tabeli czy indeksu może zająć nawet kilka minut.



Rysunek 9.8. Umieszczenie nazwy kolumny jako argumentu dowolnej (również systemowej) funkcji ma dokładnie ten sam wpływ na użycie indeksów co użycie nazwy kolumny w wyrażeniu

Poprawa wydajności złączeń

Łączenie tabel, szczególnie tych dużych (liczących wiele wierszy i kolumn), jest kosztowną operacją — wybranie pasujących do warunku złączenia (warunku z klauzuli ON) wierszy w najgorszym wypadku jest operacją o złożoności obliczeniowej rzędu $O(n^2)$, gdzie n jest liczbą wierszy. Oznacza to, że jeżeli koszt złączenia 100 wierszy wyniósł 10 000, to koszt złączenia 200 wierszy wyniesie już 40 000, a więc cztery razy więcej niż w przypadku dwukrotnie mniejszych tabel.

Zmniejszyć ten koszt możemy, zakładając indeksy na używanych do łączenia tabel kolumnach — jeżeli obie użyte do złączenia kolumny są poindeksowane, złączenie tabel będzie operacją o złożoności obliczeniowej rzędu $O(n)$. Ponieważ prawie zawsze tabele są łączone na podstawie pary kolumn klucz podstawowy – klucz obcy, to na tych kolumnach należy założyć indeksy.

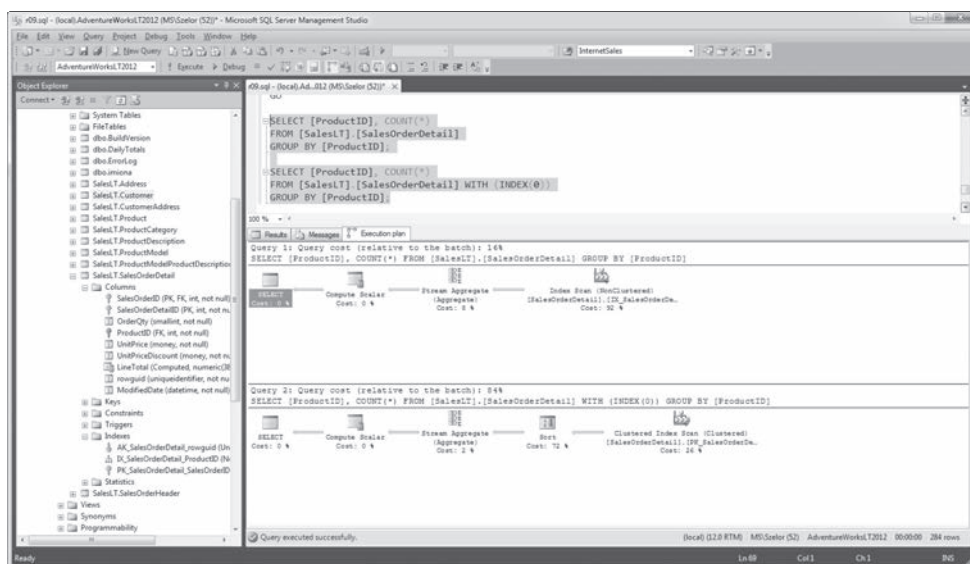
Większość serwerów bazodanowych, w tym SQL Server, automatycznie indeksuje kolumny klucza podstawowego. Oznacza to, że aby poprawić wydajność złączeń, wystarczy założyć indeksy na kolumnach kluczy obcych wszystkich tabel bazy danych.

Wydajne grupowanie i partycjonowanie danych

Grupowanie, sortowanie i wyszukiwanie danych może być bardzo kosztowne. Najprostszym i najskuteczniejszym sposobem na poprawę wydajności grupowania danych, tak samo jak sortowania czy wyszukiwania na podstawie argumentów SARG, jest użycie indeksów.

W przypadku grupowania zindeksowana powinna być każda lub co najmniej pierwsza kolumna wymieniona w klauzuli GROUP BY. Dzięki temu serwer bazodanowy będzie mógł odczytać odpowiednio posortowane dane, co znacznie uprości i przyspieszy ich pogrupowanie (grupowanie posortowanych danych wymaga wielokrotnie mniej czasu procesora i pamięci niż grupowanie nieposortowanych danych)⁵. Ponadto klucze indeksu są z reguły znacznie mniejsze niż wiersze tabeli, a więc zmniejszy się liczba odczytywanych z dysku danych.

Żeby się o tym przekonać, wykonamy dwa zapytania zwracające dokładnie tak samo pogrupowane i posortowane dane, za drugim razem zabraniając serwerowi bazodanowemu skorzystania z indeksu⁶ (rysunek 9.9).



Rysunek 9.9. Pogrupowanie danych nawet małej, liczącej mniej niż tysiąc wierszy, tabeli bez indeksu okazało się pięć razy bardziej kosztowne niż użycie w tym celu odpowiedniego indeksu

⁵ Niektóre serwery bazodanowe potrafią grupować tylko posortowane dane. Jeżeli brakuje im przydatnego indeksu, przed grupowaniem dodatkowo sortują dane.

⁶ Serwery bazodanowe umożliwiają określenie sposobu, w jaki zapytanie ma być wykonane — służą do tego specyficzne dla danego serwera dyrektywy optymalizatora. Dyrektywa `WITH (INDEX(0))` zabrania serwerowi SQL użycia jakiegokolwiek indeksu.

Zapytania partycjonujące dane (a więc zapytania, w których występuje klauzula `OVER`) mają większe wymagania co do indeksów. W ich przypadku najlepszymi indeksami są indeksy POC (ang. *Partitioning, Ordering, Covering*), czyli takie, w których pierwsze kolumny są kolumnami użytymi do partycjonowania danych (w klauzuli `PARTITION BY`), po nich występują kolumny użyte do ustalenia porządku wierszy (w klauzuli `ORDER BY`), a indeks zawiera też wszystkie pozostałe (wymienione w klauzuli `SELECT`) kolumny.

Podsumowanie

- ♦ Kolejność, w jakiej wpisuje się poszczególne klauzule zapytania, nie odpowiada kolejności, w której są one wykonywane przez serwery bazodanowe.
- ♦ Serwery bazodanowe optymalizują sposoby wykonania zapytań.
- ♦ Sposobem na sprawdzenie, czemu dane zapytanie działa wolniej, niż się spodziewaliśmy, jest sprawdzenie planu jego wykonania.
- ♦ Indeksy są najprostszym i wyjątkowo skutecznym sposobem na poprawę wydajności zapytań.
- ♦ Samo istnienie indeksu nie wystarczy, żeby serwer bazodanowy mógł z niego efektywnie skorzystać — zapytanie musi być poprawnie napisane, np. do wybierania wierszy powinno się używać argumentów typu `SARG`, a zapytanie nie powinno odczytywać więcej kolumn, niż jest to wymagane.
- ♦ Uniwersalna strategia indeksowania polega na tworzeniu indeksów zawierających zapytania, a więc indeksów wielokolumnowych. Dla zapytań grupujących lub partycjonujących dane należy tworzyć indeksy typu POC.

Zadania

1. Poniższe zapytanie zwracające nazwy produktów i kategorii działa zbyt wolno. Jak można poprawić jego wydajność?

```
SELECT [Name]
FROM [SalesLT].[Product]
UNION
SELECT [Name]
FROM [SalesLT].[ProductCategory];
```

2. Wykonanie poniższego zapytania wymaga przeskanowania (odczytania w całości) indeksu założonego na kolumnie `UnitPrice`. Przepisz to zapytanie tak, żeby używało argumentu typu `SARG`.

```
SELECT [SalesOrderID]
FROM [SalesLT].[SalesOrderDetail]
WHERE [UnitPrice]*.77 > 900;
```

3. Znajdź optymalny indeks dla poniższego zapytania:

```
SELECT [DueDate], [SalesOrderID], [TotalDue],  
       LAG([TotalDue]) OVER (PARTITION BY [DueDate] ORDER BY [DueDate]) as  
       PreviousTotalDue  
FROM [SalesLT].[SalesOrderHeader]  
ORDER BY [DueDate];
```

Skorowidz

A

Access, 10
aktualizowanie danych, 223–226
aliasy, 64, 95
ANSI, 44
argumenty SARG, 203

B

baza danych, 24, 251
 model jednorodny, 25
 model obiektowy, 29
 model relacyjny, 25
blokady, 239
 współdzielone, 239
 wyłączne X, 239
brudne odczyty, Dirty reads, 232

C

CTE, Common Table
 Expressions, 176
 proste, 177
 rekurencyjne, 179
część wspólna zapytań, 112
czwarta postać normalna, 35

D

DB2, 10
dialekty języka SQL, 43
dostęp do tabel, 113
druga postać normalna, 33
duplikat, 57
dziedziczenie uprawnień, 290

E

eliminowanie duplikatów,
 57, 107

F

Firebird, 10
formatowanie wyników, 64
funkcja, 113
 AVG(), 121
 CHECKSUM_AGG(), 123
 CASE, 63
 COUNT(), 120
 COUNT_BIG(), 123
 GROUPING, 131
 GROUPING_ID, 131
 MAX(), 122
 MIN(), 122
 STDEV(), 123
 STDEVP(), 123
 SUM(), 121
 VAR(), 124
 VARP(), 124
funkcje
 analityczne, 156
 arytmetyczne, 61
 daty i czasu, 62
 grupujące, 119, 123
 okienkowe, 151, 154
 rankingu, 149
 systemowe, 60
 użytkownika, 42
 znakowe, 61

G

generowanie kluczy
 podstawowych, 258
grupowanie
 danych, 208, 272
 wierszy, 119

H

hierarchia operatorów, 82

I

identyfikatory, 41
iloczyn kartezjański, 100
indeksy, 276
 modyfikowanie, 278
 opcje, 280
 tworzenie, 278
 uporządkowanie, 281
 usuwanie, 278
instalacja SQL Server, 12
instrukcja
 ALTER, 253
 BEGIN TRAN, 237
 CREATE, 251
 CREATE SCHEMA:, 255
 DELETE, 221
 DROP, 254
 GRANT, 283
 INSERT, 213
 INSERT INTO ... SELECT,
 218
 MERGE, 227
 REVOKE, 283

instrukcja
 ROLLBACK TRAN, 236, 237
 SELECT, 51
 SELECT ... INTO, 218
 TRUNCATE TABLE, 223
 UPDATE, 224

instrukcje
 automatycznie rozszerzane,
 270
 Connection Statements, 46
 Control Statements, 46
 Data Statements, 46
 Diagnostics Statements, 46
 Schema Statements, 46
 Session Statements, 46
 Transaction Statements, 46
 InterBase, 10
 izolowanie transakcji, 241

J

jawne zatwierdzanie transakcji,
 234
 język
 interpretowany, 39
 kompilowany, 39
 proceduralny, 39
 SQL, 19, 37
 strukturalny, 39

K

klasy instrukcji, 46
 klauzula
 FROM, 53
 GROUP BY, 125, 128
 HAVING, 138
 ORDER BY, 268
 OVER, 144
 TOP, 85
 WHERE, 76

klucz
 kompozytowy, 259
 obcy, 92, 108, 261
 podstawowy, 214
 kolejność
 kolumn, 23
 wierszy, 24
 wykonywania klauzul, 128
 wykonywania zapytań, 198
 złączeń, 104
 kolumny, 22, 55
 komentarze, 43

kompozytowe klucze
 podstawowe, 259
 konstruktor wierszy, 217
 konto użytkownika, 285, 286
 konwersja typów, 62

L

literały, 42, 65
 logika trójwartościowa, 73

Ł

łączenie
 danych tekstowych, 60
 tabel, 91
 wierszy, 113
 wyników zapytań, 109

M

model
 optymistyczny, 246
 pesymistyczny, 247
 relacyjny, 21
 modele baz danych, 24
 modyfikowanie
 danych, 213, 272
 indeksów, 278
 widoków, 270
 MySQL, 10

N

nadawanie uprawnień, 285, 289
 nazwy obiektów, 54
 niepowtarzalne odczyty, Non-
 repeatable reads, 232
 niepowtarzalność, 259
 normalizacja, 32
 NOT NULL, 257

O

odbieranie uprawnień, 285, 289
 odczytywanie
 widma, Phantom reads, 232
 danych, 53, 271
 ograniczenia, 256, 264
 okienka, 151
 opcje indeksów, 280
 operator, 42
 ALL, 193

AND, 75
 ANY, 189, 192
 APPLY, 115
 BETWEEN ... AND, 79
 CUBE, 129
 EXCEPT, 112
 EXISTS, 186
 GROUPING SETS, 132
 IN, 78
 INTERSECT, 112
 IS NULL, 81
 LIKE, 80
 NOT, 75
 OR, 75
 PIVOT, 134
 ROLLUP, 129
 SOME, 189
 UNION, 110
 UNION ALL, 111
 UNPIVOT, 137

operatory
 arytmetyczne, 59
 logiczne, 74
 porównania, 77
 SQL, 78
 Oracle Database, 10

P

partycjonowanie
 danych, 147, 208
 wierszy, 143
 pierwsza postać normalna, 33
 PK, Primary Key, 257
 PL/pgSQL, 44
 PL/SQL, 44
 pobieranie danych, 51
 podzapytania, 161
 a złączenia, 172
 jako zmienne, 162
 jako źródła danych, 173
 niepowiązane, 162
 niezwracające żadnych
 wartości, 166
 powiązane, 168
 zwracające listę wartości, 165
 porządkowanie danych, 149
 postać Boyce'a-Codda, 34
 PostgreSQL, 10
 postulaty Codda
 integralność danych, 32
 przetwarzanie danych, 31
 struktura danych, 31
 poziomy zgodności, 48

procedura, 113
procedury użytkownika, 42
przekazywanie uprawnień, 292
przetwarzanie
 pojedynczych danych, 38
 zbiorów, 38
przykładowa baza danych, 16
przypisywanie ról, 287
punkty przywracania, 238

R

Read Committed, 242
Read Uncommitted, 242
relacyjne bazy danych, 21, 30
Repeatable Read, 243
rola Public, 288
role, 287
rozpoczynanie transakcji, 234
różnica zapytań, 112

S

SARG, 203
schemat, 255
Serializable, 245
serwery bazodanowe, 10, 197, 256
składnia języka SQL, 41
słowa kluczowe, 43
sortowanie
 danych tekstowych, 69
 wyników, 66
SQL, 19
SQL PL, 44
SQL Server, 10, 11
SQL3, 46, 49
stałe, 65
standardy ANSI, 44
stronicowanie wierszy, 87
struktura tabeli, 256
strukturalny język zapytań, 37
suma zapytań, 109

T

tabele, 21, 251
 ograniczenia, 256
 pochodne, 174
 tworzenie, 254
 usuwanie, 254
 złączenia, 91–109
 zmiana struktury, 256

transakcje, 231
 izolowane, Isolation, 232
 niepodzielne, Atomicity, 232
 poziomy izolowania, 241
 rozpoczynanie, 234
 spójne, Consistency, 232
 trwale, Durability, 233
 wycofywanie, 236
 zagnieżdżanie, 237
 zatwierdzanie, 234, 237
transakcyjne przetwarzanie
 danych, 233
trendy, 182
tryb
 niezatwierdzonego odczytu,
 242
 odczytu zatwierdzonego, 242
 powtarzalnego odczytu, 243
 szeregowania, 245
tryby blokad, 239
trzecia postać normalna, 33
T-SQL, 44
tworzenie
 baz danych, 249, 251
 indeksów, 278
 ról, 287
 schematu, 255
 tabel, 254
 widoków, 267
typy danych, 46

U

uprawnienia, 285
 dziedziczenie, 290
 na serwerze SQL, 289
 nadawanie, 289
 obiektowe, 288
 odbieranie, 289
 przekazywanie, 292
 systemowe, 288
 zasady, 293
uprawnienia użytkowników, 283
usuwanie
 baz danych, 251, 254
 danych, 221
 indeksów, 278
 kaskadowe, 263
 kont użytkowników, 286
 ról, 287
 tabel, 254
 widoków, 267
 wyników podzapytań, 222
utrata aktualizacji,
 Lost update, 232

W

wartości domyślne, 215, 260
wartość NULL, 47, 74, 216, 257
warunek logiczny, 260
widok, View, 42, 113, 267
 grupujący dane, 272
 modyfikowanie, 270
 modyfikujący dane, 272
 odczytywanie danych, 271
 tworzenie, 267
 usuwanie, 267
 zagnieżdżony, 271
 zalety, 275
wiersze, 23, 73, 119
właściwości transakcji, 231
współbieżność, 231, 239
wstawianie
 danych, 213
 wyników zapytań, 218
wybieranie
 kolumn, 55
 wierszy, 73
wycofywanie transakcji, 236
wydajność
 instrukcji, 264
 zapytań, 197
 złączeń, 207
wyniki funkcji tabelarycznych,
 113
wyniki zapytań, 91, 109
 część wspólna, 112
 różnica, 112
 suma, 109
wyrażenia, 58, 124, 225
wyszukiwanie danych, 203
wyznaczanie trendów, 182
wyzwalacz, Trigger, 42

Z

zagnieżdżanie
 funkcji grupujących, 124
 podzapytań, 167
 transakcji, 237
zakleszczenie, DeadLock, 240
zakładanie kont użytkowników,
 286
zakresy blokad, 239
zapytania, 197
 kolejność wykonywania, 198
 plany wykonania, 199

zasada minimalnych uprawnień,
293
zatwierdzanie transakcji, 234, 237
zliczanie wierszy, 121
złączenia, 207
 krzyżowe, 100

lewostronne, 99
naturalne, 91
nienaturalne, 91
nierównościowe, 96
obustronne, 99
prawostronne, 99

równościowe, 96
wielokrotne, 102
 zewnętrzne, 98
złączenie tabeli z nią samą, 106
złożone warunki logiczne, 82
zmiana struktury tabeli, 256

PROGRAM PARTNERSKI

— GRUPY HELION —

- 
1. ZAREJESTRUJ SIĘ
 2. PREZENTUJ KSIĄŻKI
 3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA
Helion 



SQL to dziś właściwie jedyny poważny język używany do tworzenia i obsługi relacyjnych baz danych, niezależnie od tego, czy są to bazy Microsoftu, Oracle czy Sun Microsystems. I choć nie wszędzie działa dokładnie tak samo, bez jego znajomości nie ma co marzyć o swobodnym korzystaniu z bazy, nie mówiąc już o jej stworzeniu. Jeśli więc taka baza jest Ci potrzebna, jeśli chcesz zorientować się, jak ułożyć dane w sposób najwygodniejszy dla siebie albo precyzyjnie wysegregować to, czego akurat szukasz, musisz opanować SQL — inaczej serwer bazodanowy nijak Cię nie zrozumie.

Trzecie wydanie tej książki traktuje o języku SQL w wersji dla SQL Server firmy Microsoft. Autorzy szybko przeprowadzą Cię od instalacji serwera bazodanowego, przez najróżniejsze operacje na przykładowej, niewielkiej bazie AdventureWorksLT, aż po kwestie związane z tworzeniem własnej bazy i nadawaniem uprawnień jej użytkownikom. Ponadto znajdziesz tu ważne (i nowe!) informacje o partycjonowaniu danych i wydajności zapytań. Bezcennym wsparciem w trakcie nauki będą dla Ciebie zadania — ich rozwiązanie pozwoli Ci poczuć się pewniej i sprawdzić swoje wiadomości w praktyce. Jeśli chcesz rozpocząć przygodę z bazami danych albo odświeżyć wiedzę, trafiłeś doskonale!

- Teoria baz danych i standardy języka SQL
- Odczytywanie danych z wybranej tabeli
- Wybieranie i grupowanie wierszy
- Łączenie tabel i wyników zapytań
- Partycjonowanie wierszy oraz funkcje rankingu, analityczne i okienkowe
- Podzapytania i wydajność zapytań
- Transakcje i współbieżność
- Bazy danych i tabele
- Widoki i indeksy
- Nadawanie i odbieranie uprawnień

Stwórz bazę danych z językiem SQL!

Helion

helion.pl

HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 250 98 63
helion@helion.pl

Sprawdź nasze szkolenia!

SZKOLENIA

AKADEMIA IT & BUSINESS

HELIONSZKOLENIA.PL

KOD KORZYŚCI
Sięgnij po więcej!

ISBN 978-83-283-9236-6

9 788328 392366