

SPRAWDŹ NOWE MOŻLIWOŚCI T-SQL I SQL SERVER!

Apress

T-SQL dla zaawansowanych Przewodnik programisty

Wydanie IV

Miguel Cebollero, Michael Coles, Jay Natarajan

Helion 

Tytuł oryginału: Pro T-SQL Programmer's Guide, 4th Edition

Tłumaczenie: Andrzej Stefański

ISBN: 978-83-283-2247-9

Original edition copyright © 2015 by Miguel Cebollero, Jay Natarajan, and Michael Coles
All rights reserved.

Polish edition copyright © 2016 by HELION SA.
All rights reserved.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

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 Wydawnictwo HELION 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 Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/tsqlz4>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

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

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

Spis treści

O autorach	11
O korektorze merytorycznym	12
Podziękowania	13
Wprowadzenie	15
Rozdział 1. Podstawy T-SQL	21
Krótka historia T-SQL	21
Języki imperatywne i deklaratywne	22
Podstawy SQL	23
Wyrażenia	23
Bazy danych	25
Logi transakcyjne	26
Schematy	26
Tabele	27
Widoki	28
Indeksy	29
Procedury składowane	30
Funkcje użytkownika	30
Moduły SQL CLR	30
Podstawy stylu	31
Białe znaki	31
Konwencje nazewnictwa	32
Jedno wejście, jedno wyjście	33
Programowanie defensywne	35
Wyrażenie SELECT *	36
Inicjalizacja zmiennych	37
Podsumowanie	37

Rozdział 2. Narzędzia	39
SQL Server Management Studio	39
IntelliSense	40
Fragmenty kodu	41
Schematy skrótów klawiszowych	43
Debugowanie T-SQL	43
Opcje edycji w SSMS	45
Pomoc kontekstowa	46
Graficzna reprezentacja planów wykonania zapytań	48
Opcje do zarządzania projektami	48
Eksplorator obiektów	50
Narzędzie SQLCMD	52
SQL Server Data Tools	54
SQL Profiler	54
Extended Events	57
SQL Server Integration Services	58
BCP	58
SQL Server 2014 Books Online	60
Przykładowa baza danych AdventureWorks	61
Podsumowanie	61
Rozdział 3. Kod proceduralny	63
Logika trójwartościowa	63
Przepływ sterowania	65
Słowa kluczowe BEGIN i END	65
Wyrażenie IF ... ELSE	66
Wyrażenia WHILE, BREAK i CONTINUE	68
Wyrażenie GOTO	69
Wyrażenie WAITFOR	70
Wyrażenie RETURN	71
Wyrażenie CASE	72
Proste wyrażenie CASE	72
Przeszukiwane wyrażenie CASE	73
CASE i tabele przestawne	75
Wyrażenie IIF	79
CHOOSE	80
COALESCE i NULLIF	81
Kursory	82
Podsumowanie	89
Rozdział 4. Funkcje użytkownika	91
Funkcje skalarne	91
Rekurencja w skalnych UDF	93
Kod proceduralny w funkcjach użytkownika	96
Wielowyrażeniowe funkcje zwracające tabelę	104
Wbudowane funkcje zwracające tabelę	112
Ograniczenia funkcji definiowanych przez użytkownika	115
Funkcje niedeterministyczne	115
Stan bazy danych	116
Podsumowanie	117

Rozdział 5. Procedury składowane	119
Wprowadzenie	119
Odkrywanie metadanych	121
Natywnie kompilowane procedury składowane	122
Zarządzanie procedurami składowanymi	124
Najlepsze praktyki związane z tworzeniem procedur składowanych	125
Przykład procedury składowanej	127
Rekurencja w procedurach składowanych	132
Parametry tabelaryczne	140
Tymczasowe procedury składowane	142
Rekompilacja i pamięć podręczna	143
Statystyki procedur składowanych	143
Przechwytywanie parametrów	145
Rekompilacja	148
Podsumowanie	151
Rozdział 6. Obiekty pamięciowe	153
Czynniki napędzające technologie pamięciowe	153
Trendy sprzętowe	154
Podstawy obiektów pamięciowych	155
Krok 1. Dodanie nowej grupy plików optymalizowanej do operacji pamięciowych	156
Krok 2. Dodanie nowego kontenera optymalizowanego do operacji pamięciowych	157
Krok 3. Utwórz nową tabelę optymalizowaną do operacji pamięciowych	159
Ograniczenia tabel pamięciowych	163
Indeksy tabel pamięciowych OLTP	164
Indeksy typu hash	165
Indeksy zakresu	167
Natywnie kompilowane procedury składowane	169
Rozdział 7. Wyzwalacze	173
Wyzwalacze DML	173
Wiele wyzwalaczy	174
Kiedy używać wyzwalaczy DML	175
Wirtualne tabele inserted i deleted	177
Przeglądanie za pomocą wyzwalaczy DML	178
Wykorzystanie zapisywania modyfikacji danych	178
Udostępnianie danych wyzwalaczom	183
Wyzwalacze zagnieżdżone i rekurencyjne	183
Funkcje UPDATE() i COLUMNS_UPDATED()	184
Wyzwalacze na widokach	188
Wyzwalacze DDL	191
Typy zdarzeń DDL i grupy zdarzeń	191
Wyzwalacze logowania	195
Podsumowanie	197
Rozdział 8. Szyfrowanie	199
Hierarchia szyfrowania	199
Klucze główne usługi	200
Klucze główne bazy danych	201
Certyfikaty	203

Ograniczenia szyfrowania asymetrycznego	205
Klucze asymetryczne	207
Kopie zapasowe klucza asymetrycznego	210
Klucze symetryczne	210
Tymczasowe klucze symetryczne	212
Sól i uwierzytelnianie	216
Szyfrowanie bez kluczy	217
Tworzenie skrótów	217
Rozszerzone zarządzanie kluczami	218
Przeźroczyste szyfrowanie danych	219
Podsumowanie	221
Rozdział 9. Wyrażenia nazwane i funkcje okna	223
Wyrażenia nazwane	223
Wiele zapytań nazwanych	225
Zwiększona czytelność CTE	226
Rekurencyjne zapytania nazwane	227
Funkcje okna	231
Funkcja ROW_NUMBER	231
Stronicowanie zapytania za pomocą OFFSET/FETCH	233
Funkcje RANK i DENSE_RANK	234
Funkcja NTILE	239
Funkcje agregacyjne, analityczne i klauzula OVER	241
Przykłady funkcji analitycznych	245
CUME_DIST i PERCENT_RANK	245
PERCENTILE_CONT i PERCENTILE_DISC	247
LAG i LEAD	248
FIRST_VALUE i LAST_VALUE	250
Podsumowanie	251
Rozdział 10. Typy danych i zaawansowane typy danych	253
Podstawowe typy danych	253
Znaki	253
Typy danych max	254
Wartości numeryczne	256
Typy danych dla daty i czasu	258
UTC i czas wojskowy	261
Funkcje daty i czasu	262
Przesunięcia a strefy czasowe	265
Typ danych uniqueidentifier	265
Typ danych hierarchyid	267
Reprezentacja danych hierarchicznych	267
Przykład hierarchyid	269
Metody hierarchyid	273
Przestrzenne typy danych	274
(X, Y) czy (szerokość, długość)	277
Półkula i orientacja	278
Michigan i wielkie jeziora	279

Obsługa FILESTREAM	281
Włączanie obsługi FILESTREAM	282
Tworzenie grup plików FILESTREAM	283
Tabele korzystające z FILESTREAM	284
Korzystanie z danych FILESTREAM	285
Obsługa FileTable	286
Funkcje filetable	290
Wyzwalacze na filetable	294
Podsumowanie	295
Rozdział 11. Wyszukiwanie pełnotekstowe	297
Architektura FTS	297
Tworzenie katalogów pełnotekstowych i indeksów	298
Tworzenie katalogów pełnotekstowych	298
Tworzenie indeksów pełnotekstowych	300
Zapytania pełnotekstowe	305
Predykat FREETEXT	306
Optymalizacja wydajności FTS	307
Predykat CONTAINS	308
Funkcje FREETEXTTABLE i CONTAINSTABLE	311
Tezaury i stoplisty	313
Procedury składowane, dynamiczne widoki zarządcze i funkcje	317
Semantyka statystyczna	318
Podsumowanie	320
Rozdział 12. XML	321
XML dawniej	321
OPENXML	321
Formaty generowane przez OPENXML	325
Klauzula FOR XML	327
FOR XML RAW	327
FOR XML AUTO	329
FOR XML EXPLICIT	331
FOR XML PATH	332
Typ danych xml	334
Niestrukturyzowany xml	334
Ustrukturyzowany xml	335
Metody typu danych xml	337
Metoda query	338
Metoda value	339
Metoda exist	339
Metoda nodes	340
Metoda modify	341
Indeksy XML	343
Przekształcenia XSL	347
Ustawienia związane z bezpieczeństwem SQL CLR	350
Podsumowanie	352

Rozdział 13. XQuery i XPath	353
XPath i FOR XML PATH	353
Atrybuty XPath	355
Kolumny bez nazw i wieloznaczniki	356
Grupowanie elementów	356
Funkcja data	357
Testowanie węzłów i funkcje	358
XPath i NULL	359
Klauzula WITH XMLNAMESPACES	360
Testowanie węzłów	360
XQuery i typ danych xml	361
Wyrażenia i sekwencje	361
Metoda query	363
Ścieżki określające lokalizację	364
Testowanie węzłów	366
Przestrzenie nazw	367
Oznaczenia osi	369
Dynamiczne generowanie XML	371
Komentarze XQuery	373
Typy danych	373
Predykaty	374
Operatory porównujące wartości	374
Operatory porównań ogólnych	376
Format daty XQuery	377
Porównania węzłów	378
Wyrażenia warunkowe (if...then...else)	379
Wyrażenia arytmetyczne	379
Dzielenie liczb całkowitych w XQuery	380
Funkcje w XQuery	380
Konstruktory i rzutowanie	383
Wyrażenia FLWOR	384
Słowa kluczowe for i return	384
Słowo kluczowe where	387
Słowa kluczowe order by	387
Słowo kluczowe let	388
Obsługa UTF-16	389
Podsumowanie	391
Rozdział 14. Widoki katalogowe i dynamiczne widoki zarządcze	393
Widoki katalogowe	393
Metadane tabel i kolumn	394
Odpytywanie o pozwolenia	395
Dynamiczne widoki zarządcze i funkcje	397
Metadane indeksowe	398
Informacje o sesji	402
Informacje o połączeniu	403
Aktualnie wykonywany SQL	403
Pamięciowe widoki systemowe	405
Najdroższe zapytania	406

Przestrzeń tempdb	407
Zasoby serwera	409
Niewykorzystywane indeksy	411
Statystyki oczekiwania	413
Widoki INFORMATION_SCHEMA	413
Podsumowanie	416
Rozdział 15. Programowanie klienta .NET	417
ADO.NET	417
Klient SQL .NET	419
Połączenie ze źródłem danych	419
Odłączone zbiory danych	423
Zapytania parametryzowane	424
Zapytania niezwracające wyników, skalarne i XML	428
SqlBulkCopy	431
Wiele aktywnych zbiorów wyników	436
LINQ to SQL	439
Designer	440
Zapytania z LINQ to SQL	442
Podstawy zapytań LINQ to SQL	442
Klauzula where	444
Klauzula orderby	445
Klauzula join	446
Odroczone wykonanie zapytania	447
Od LINQ do Entity Framework	448
Odpytywanie elementów	452
Podsumowanie	456
Rozdział 16. Programowanie z CLR	459
Stary sposób	459
Rozwiązania ze zintegrowanym CLR	460
Pakiety CLR	461
Funkcje użytkownika	465
Procedury składowane	472
Funkcje agregujące użytkownika	476
Tworzenie prostej funkcji agregującej	476
Tworzenie zaawansowanych UDA	479
Typy użytkownika w zintegrowanych CLR	484
Wyzwalacze	491
Podsumowanie	495
Rozdział 17. Usługi danych	497
LocalDB z SQL Server 2014 Express	498
Programowanie asynchroniczne z ADO.NET 4.5	502
ODBC dla Linuksa	503
JDBC	508
Architektura SOA i usługi danych WCF	510
Tworzenie WCF Data Service	512
Definiowanie źródła danych	512

Tworzenie usługi danych	514
Tworzenie odbiornika WCF Data Service	518
Podsumowanie	522
Rozdział 18. Obsługa błędów i dynamiczny SQL	523
Obsługa błędów	523
Stare metody obsługi błędów	523
Wyrażenie RAISERROR	525
Obsługa wyjątków TRY...CATCH	526
TRY_PARSE, TRY_CONVERT i TRY_CAST	528
Wyrażenie THROW	529
Narzędzia do debugowania	530
Debugowanie za pomocą wyrażenia PRINT	530
Flagi śledzenia	531
Debugger zintegrowany z SSMS	532
Debugger T-SQL w Visual Studio	533
Dynamiczny SQL	536
Wyrażenie EXECUTE	536
Wstrzykiwanie SQL i dynamiczny SQL	537
Usuwanie problemów z dynamicznym SQL	539
Procedura składowana sp_executesql	540
Dynamiczny SQL i zasięg	540
Parametryzacja po stronie klienta	541
Podsumowanie	542
Rozdział 19. Poprawianie wydajności	543
Pamięć masowa w SQL Server	543
Pliki i grupy plików	543
Alokacja przestrzeni	544
Partycje	549
Kompresja danych	550
Kolumny rzadkie	555
Indeksy	559
Sterty	559
Indeksy klastrowe	560
Indeksy nieklastrowe	562
Indeksy filtrowane	565
Optymalizacja zapytań	566
Czytanie planów zapytań	566
Metodologia	570
Oczekiwanie	571
Extended Events	573
Podsumowanie	577
Dodatek A Odpowiedzi do ćwiczeń	579
Dodatek B Typy danych XQuery	589
Dodatek C Słowniczek	595
Dodatek D Krótki przewodnik po SQLCMD	609
Skorowidz	617

ROZDZIAŁ 9



Wyrażenia nazwane i funkcje okna

SQL Server 2014 kontynuuje obsługę wyjątkowo użytecznych wyrażeń nazwanych, które po raz pierwszy pojawiły się w SQL Server 2005. CTE mogą uprościć Twoje zapytania, co sprawi, że staną się łatwiejsze do analizy i utrzymania. SQL Server obsługuje też odwołania do tego samego CTE, co pozwala na tworzenie bardzo potężnych zapytań rekurencyjnych.

Dodatkowo SQL Server obsługuje funkcje okna, które pozwalają na dzielenie wyników i dodawanie numerowania oraz wartości rankingowych do wierszy w uzyskanych za ich pomocą zbiorach wyników. Rozdział ten zaczyna się od omówienia możliwości i korzyści powiązanych ze stosowania CTE, a kończy opisem dostępnych w SQL Server funkcji okna.

Wyrażenia nazwane

Wyrażenia nazwane (CTE — ang. *common table expressions*) są użytecznym dodatkiem do SQL Server. CTE to coś w rodzaju tymczasowej tabeli zawierającej nazwany zbiór wyników, dostępny jedynie dla pojedynczego zapytania lub wyrażenia DML albo do jawnego usunięcia. CTE tworzy się w tym samym wierszu kodu, co wyrażenie SELECT lub wyrażenie DML, które z niego korzystają, podczas gdy budowanie i użycie tabeli tymczasowej to zazwyczaj dwuetapowy proces. W porównaniu z trwałymi tabelami i widokami CTE mają kilka zalet. Oto one.

- CTE są tymczasowe i istnieją jedynie w czasie działania pojedynczego zapytania lub wyrażenia DML. Oznacza to, że nie musisz tworzyć trwałych obiektów bazodanowych, takich jak widoki.
- Do pojedynczego CTE można odwoływać się wiele razy za pomocą nazwy w pojedynczym zapytaniu lub wyrażeniu DML, co czyni Twój kod łatwiejszym do zarządzania. Tabele pomocnicze muszą być przepisywane w całości przy każdym odwołaniu do nich.
- Można korzystać z CTE do grupowania za pomocą kolumn tworzonych ze zbioru skalarного lub za pomocą funkcji niedeterministycznej.
- CTE mogą odwoływać się do siebie, co jest użytecznym mechanizmem rekurencyjnym.
- Zapytania odwołujące się do CTE można wykorzystywać do definiowania kursora.

CTE mogą być bardzo różne — od prostych do bardzo skomplikowanych konstrukcji. Wszystkie CTE rozpoczynają się od słowa kluczowego WITH, po którym umieszczana jest nazwa CTE i lista zwracanych przez zapytanie kolumn. Następnie umieszczane jest słowo kluczowe AS i treść CTE, którą tworzy zapytanie lub wyrażenie DML zakończone średnikiem w przypadku ciągu złożonego z większej liczby wyrażeń. Na listingu 9.1 pokazany jest bardzo prosty przykład służący do demonstracji podstawowej składni.

Listing 9.1. Proste CTE

```

WITH PobierzNazwiskaCTE ( BusinessEntityID, FirstName, MiddleName, LastName )
AS
(
    SELECT BusinessEntityID, FirstName, MiddleName, LastName
    FROM Person.Person
)
SELECT
    BusinessEntityID,
    FirstName,
    MiddleName,
    LastName
FROM PobierzNazwiskaCTE;

```

Na listingu 9.1 jest definiowane CTE o nazwie `PobierzNazwiskaCTE`, które zwraca kolumny `BusinessEntityID`, `FirstName`, `MiddleName` i `LastName`. Treść CTE zawiera proste zapytanie do tabeli `Person.Person` bazy danych *AdventureWorks2014*. Bezpośrednio po CTE znajduje się związane z nim wyrażenie `SELECT`. Wyrażenie `SELECT` odwołuje się do CTE w klauzuli `FROM`.

Przeładowane WITH

Słowo kluczowe `WITH` jest w SQL Server przeładowane, co oznacza, że wykorzystywane jest na kilka różnych sposobów w odmiennych zadaniach w T-SQL. Wykorzystywane jest między innymi przy określaniu dodatkowych opcji w wyrażeniach `DDL CREATE`, przy dodawaniu odpowiedzi do zapytań i wyrażań `DML` oraz przy deklarowaniu przestrzeni nazw XML w klauzuli `WITH XMLNAMESPACES`. Obecnie jest też wykorzystywane jako słowo kluczowe rozpoczynające definicję CTE. Jeśli zatem CTE nie jest pierwszym wyrażeniem w ciągu, poprzednie wyrażenie musi być zakończone średnikiem. Jest to jeden z powodów tego, że zdecydowanie należy korzystać ze średnika kończącego wyrażenie w tworzonym kodzie.

Proste CTE mają kilka ograniczeń przy definiowaniu i deklarowaniu.

- Do CTE muszą być dołączone pojedyncze wyrażenia `INSERT`, `DELETE`, `UPDATE` lub `SELECT`.
- Wszystkie kolumny zwracane przez CTE muszą mieć unikalne nazwy. Jeśli wszystkie kolumny zwracane przez zapytanie w treści CTE mają unikalne nazwy, możesz pominąć listę kolumn w deklaracji CTE.
- CTE może odwoływać się do innego, wcześniej zdefiniowanego CTE w tej samej klauzuli `WITH`, ale nie może odwoływać się do CTE zdefiniowanego później (nazywa się to odwołaniem do przodu).
- W CTE nie można korzystać ze słów kluczowych, klauzul i opcji, takich jak `COMPUTE`, `COMPUTE BY`, `FOR BROWSE`, `INTO` i `OPTION` (podpowiedź do zapytania). Nie możesz też korzystać z `ORDER BY`, jeśli nie użyjesz klauzuli `TOP`.
- W nierekurencyjnym CTE można zdefiniować wiele CTE. Wszystkie definicje muszą być połączone za pomocą jednego z operatorów działających na zbiorach; są to `UNION ALL`, `UNION`, `INTERSECT`, `EXCEPT`.
- Jak wspomniano w ramce „Przeładowane WITH”, jeśli CTE nie jest pierwszym wyrażeniem w ciągu, poprzedzające je wyrażenie musi być zakończone średnikiem.

Pamiętaj o tych ograniczeniach podczas tworzenia CTE.

Wiele zapytań nazwanych

Możesz definiować wiele CTE dla pojedynczego zapytania lub wyrażenia DML, oddzielając definicje CTE za pomocą przecinków. Głównym powodem, by to robić, jest możliwość uproszczenia kodu, co ułatwia jego analizę i zarządzanie. Wyrażenia nazwane oferują sposoby wizualnego rozdzielania kodów na mniejsze bloki funkcjonalne, co upraszcza jego tworzenie i debugowanie. Zapytanie z listingu 9.2 zawiera wiele CTE, przy czym drugie CTE odwołuje się do pierwszego. Wyniki jego działania zostały pokazane na rysunku 9.1.

Listing 9.2. Wiele CTE

```
WITH PobierzNazwiskaCTE ( BusinessEntityID, FirstName, MiddleName, LastName )
AS (
    SELECT BusinessEntityID, FirstName, MiddleName, LastName
    FROM Person.Person
),

PobierzKontaktCTE ( BusinessEntityID, FirstName, MiddleName, LastName,
    Email, HomePhoneNumber )
AS (
    SELECT gn.BusinessEntityID, gn.FirstName
        , gn.MiddleName, gn.LastName
        , ea.EmailAddress, pp.PhoneNumber
    FROM PobierzNazwiskaCTE gn
    LEFT JOIN Person.EmailAddress ea
        ON gn.BusinessEntityID = ea.BusinessEntityID
    LEFT JOIN Person.PersonPhone pp
        ON gn.BusinessEntityID = pp.BusinessEntityID
    AND pp.PhoneNumberTypeID = 2
)

SELECT BusinessEntityID, FirstName
    , MiddleName, LastName
    , Email, HomePhoneNumber
FROM PobierzKontaktCTE;
```

	BusinessEntityID	FirstName	MiddleName	LastName	Email	HomePhoneNumber
133	20216	Alyssa	M	Alexander	alyssa64@adventure-works.com	NULL
134	3057	Amanda	L	Alexander	amanda40@adventure-works.com	NULL
135	7458	Ana	R	Alexander	ana17@adventure-works.com	1 (11) 500 555-0181
136	7678	Angela	NULL	Alexander	angela21@adventure-works.com	NULL
137	7161	Angelica	NULL	Alexander	angelica18@adventure-works.com	214-555-0150
138	3683	Anna	NULL	Alexander	anna44@adventure-works.com	150-555-0156
139	13220	Antonio	NULL	Alexander	antonio19@adventure-works.com	NULL
140	7719	Arianna	NULL	Alexander	arianna17@adventure-works.com	140-555-0167
141	12597	Ashley	J	Alexander	ashley46@adventure-works.com	NULL
142	6266	Austin	NULL	Alexander	austin15@adventure-works.com	1 (11) 500 555-0140
143	6726	Benjamin	A	Alexander	benjamin20@adventure-works.com	240-555-0151
144	6919	Brandon	NULL	Alexander	brandon16@adventure-works.com	780-555-0152
145	19276	Brianna	NULL	Alexander	brianna64@adventure-works.com	242-555-0137
146	7210	Brittany	NULL	Alexander	brittany17@adventure-works.com	804-555-0125
147	5561	Caleb	NULL	Alexander	caleb15@adventure-works.com	670-555-0141
148	6315	Cameron	NULL	Alexander	cameron14@adventure-works.com	399-555-0196
149	6800	Caroline	NULL	Alexander	caroline20@adventure-works.com	NULL
150	13268	Carson	G	Alexander	carson18@adventure-works.com	NULL

Rysunek 9.1. Część wyniku zapytania z wieloma CTE

Zwiększona czytelność CTE

Dzięki wykorzystaniu CTE możesz tworzyć bardziej czytelne zapytania niż przy użyciu zagnieżdżonych podzapytań. Dla celów demonstracyjnych poniższe zapytanie służy do uzyskania takich samych wyników jak zapytanie korzystające z CTE pokazane na listingu 9.2, ale za pomocą zagnieżdżonych podzapytań.

```
SELECT
    gn.BusinessEntityID,
    gn.FirstName,
    gn.MiddleName,
    gn.LastName,
    gn.EmailAddress,
    gn.HomePhoneNumber
FROM
(
    SELECT
        p.BusinessEntityID,
        p.FirstName,
        p.MiddleName,
        p.LastName,
        ea.EmailAddress,
        ea.HomePhoneNumber
    FROM Person.Person p
    LEFT JOIN
    (
        SELECT
            ea.BusinessEntityID,
            ea.EmailAddress,
            pp.HomePhoneNumber
        FROM Person.EmailAddress ea
        LEFT JOIN
        (
            SELECT
                pp.BusinessEntityID,
                pp.PhoneNumber AS HomePhoneNumber,
                pp.PhoneNumberTypeID
            FROM Person.PersonPhone pp
        ) pp
        ON ea.BusinessEntityID = pp.BusinessEntityID
        AND pp.PhoneNumberTypeID = 2
    ) ea
    ON p.BusinessEntityID = ea.BusinessEntityID
) gn
```

Korzystająca z CTE wersja tego zapytania pokazana na listingu 9.2 upraszcza i hermetyzuje kod zapytania oraz jest dużo prostsza do czytania i zrozumienia niż wersja korzystająca z zagnieżdżonych podzapytań, co sprawia, że jest łatwiejsza do debugowania i utrzymania w długim horyzoncie czasowym.

Przykład z listingu 9.2 zawiera dwa CTE o nazwach PobierzNazwiskaCTE i PobierzKontaktCTE. Zapytanie PobierzNazwiskaCTE zapożyczony jest z listingu 9.1; pobiera ono nazwiska z tabeli Person.Person.

```
WITH PobierzNazwiskaCTE ( BusinessEntityID, FirstName, MiddleName, LastName )
AS (
    SELECT BusinessEntityID, FirstName, MiddleName, LastName
    FROM Person.Person
),
```

Drugie CTE, PobierzKontaktCTE, łączy wyniki PobierzNazwiskaCTE z tabelami Person.EmailAddress i Person.PersonPhone.

```
PobierzKontaktCTE ( BusinessEntityID, FirstName, MiddleName, LastName,
                    Email, HomePhoneNumber )
AS (
    SELECT gn.BusinessEntityID, gn.FirstName
        , gn.MiddleName, gn.LastName
        , ea.EmailAddress, pp.PhoneNumber
```

```

FROM PobierzNazwiskaCTE gn
LEFT JOIN Person.EmailAddress ea
    ON gn.BusinessEntityID = ea.BusinessEntityID
LEFT JOIN Person.PersonPhone pp
    ON gn.BusinessEntityID = pp.BusinessEntityID
AND pp.PhoneNumberTypeID = 2
)

```

Zauważ, że słowo kluczowe `WITH` wykorzystywane jest tylko raz na początku całego wyrażenia. Druga deklaracja CTE jest oddzielona od pierwszej za pomocą przecinka i nie wymaga użycia słowa kluczowego `WITH`. W końcu warto zauważyć jak proste i czytelne jest zapytanie `SELECT` powiązane z CTE dzięki temu, że złączenia zostały przeniesione do CTE.

```

SELECT BusinessEntityID, FirstName
    , MiddleName, LastName
    , Email, HomePhoneNumber
FROM PobierzKontaktCTE;

```

-
- **Wskazówka** Do CTE możesz odwołać się z treści innego CTE, z powiązanego zapytania lub wyrażenia DML. Oba typy odwołań CTE pokazane są na listingu 9.2 — `PobierzNazwiskaCTE` wywoływane jest przez `PobierzKontaktCTE`, a `PobierzKontaktCTE` jest wywoływane przez zapytanie powiązane z CTE.
-

Rekurencyjne zapytania nazwane

Rekurencyjne CTE to takie wyrażenie, które wywołuje samo siebie wielokrotnie, by ustalić podzbiór danych aż do momentu uzyskania pełnego wyniku. CTE może odwołać się do samego siebie, co jest użytecznym mechanizmem do odpytywania o dane hierarchiczne. Rekurencyjne CTE są podobne do nierekurencyjnych CTE, ale w treści CTE znajduje się wiele zbiorów zapytań generujących zbiory wyników, w których wiersze są połączone za pomocą operatora zbiorów `UNION ALL`. Przynajmniej jedno z zapytań z treści rekurencyjnego CTE nie może odwoływać się do CTE — zapytanie takie nazywane jest **zapytaniem kotwiczącym** (ang. *anchor query*). Rekurencyjne CTE zawierają też jedno lub więcej rekurencyjnych zapytań odwołujących się do CTE. Rekurencyjne zapytania są połączone ze sobą za pomocą zapytania kotwiczącego (lub zapytań kotwiczących) w treści CTE. Wymagają operatora `UNION ALL` najwyższego poziomu łączącego rekurencyjne i nierekurencyjne zapytania. Wiele zapytań kotwiczących można łączyć za pomocą `UNION ALL`. Rekurencja kończy się, gdy zapytanie nie zwróci żadnych wierszy. Na listingu 9.3 znajduje się proste rekurencyjne CTE pobierające zbiór wyników zawierający liczby od 1 do 10.

Listing 9.3. Proste rekurencyjne CTE

```

WITH Liczby (n)
AS (
SELECT 1 AS n
    UNION ALL
SELECT n + 1
    FROM Liczby
WHERE n < 10 )

SELECT n FROM Liczby;

```

CTE z listingu 9.3 zaczyna się od deklaracji definiującej nazwę CTE i zwracane kolumny.

```
WITH Liczby (n)
```

Treść CTE zawiera pojedyncze zapytanie kotwiczące zwracające pojedynczy wiersz z liczbą 1 w kolumnie *n*.

```
SELECT 1 AS n
```

Zapytanie kotwiczące jest łączone z zapytaniem rekurencyjnym za pomocą operatora zbiorów UNION ALL. Zapytanie rekurencyjne zawiera odwołanie wewnętrzne do CTE Liczby, które dodaje 1 do kolumny *n* przy każdym rekurencyjnym wywołaniu. Klauzula WHERE ogranicza zbiór wyników do pierwszych dziesięciu liczb.

```
SELECT n + 1
FROM Liczby
WHERE n < 10
```

Rekurencyjne CTE mają domyślnie ustawioną maksymalną głębokość rekurencji 100. Oznacza to, że zapytanie rekurencyjne w treści CTE może wywołać samo siebie maksymalnie 100 razy. Możesz użyć opcji MAXRECURSION, by zwiększyć maksymalną wartość dla każdego CTE osobno. Na listingu 9.4 znajduje się CTE z listingu 9.3 zmodyfikowane w taki sposób, by zwracało liczby od 1 do 1000. Zmodyfikowane zapytanie korzysta z opcji MAXRECURSION zwiększającej maksymalny poziom rekurencji. Bez opcji MAXRECURSION to CTE zwróciłoby błąd po 100 poziomach rekurencji.

Listing 9.4. Rekurencyjne CTE z opcją MAXRECURSION

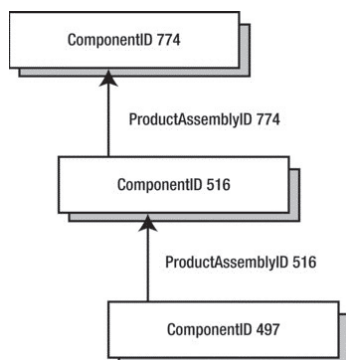
```
WITH Liczby (n)
AS (
SELECT 0 AS n
UNION ALL
SELECT n + 1
FROM Liczby
WHERE n < 1000 )
SELECT n
FROM Liczby OPTION (MAXRECURSION 1000);
```

Wartość MAXRECURSION musi być z zakresu od 0 do 32767. SQL Server wyrzuca wyjątek, jeśli zostanie przekroczony limit MAXRECURSION. Wartość 0 w zmiennej MAXRECURSION oznacza brak ograniczeń poziomów rekurencji dla CTE. Opcji tej należy używać ostrożnie — jeśli nie ograniczysz odpowiednio wyników w klauzuli WHERE zapytania, możesz znaleźć się w nieskończonej pętli.

-
- **Wskazówka** Utworzenie trwałej tabeli z kolejnymi liczbami może być wydajniejsze niż korzystanie z rekurencyjnego CTE do generowania liczb szczególnie wtedy, kiedy często będziesz generował liczby w ten sposób.
-

Rekurencyjne CTE są użyteczne przy odpytywaniu o dane zapisane w postaci rozgałęzionego grafu. Tabela Production.BillOfMaterials z bazy danych AdventureWorks zawiera przykład takich danych. W tabeli tej znajdują się dwie ważne kolumny ComponentID i ProductAssemblyID, które odwzorowują strukturę hierarchiczną. ComponentID to unikalna liczba identyfikująca każdy z komponentów wykorzystywanych przez AdventureWorks przy tworzeniu swoich produktów. ProductAssemblyID oznacza bardziej złożony komponent utworzony z jednego lub większej liczby komponentów z listy. Na rysunku 9.2 pokazano relacje pomiędzy komponentami i bardziej złożonymi produktami w bazie danych AdventureWorks.

Rekurencyjne CTE pokazane na listingu 9.5 pobiera pełną listę części (**BOM**, ang. *bills of materials*) dla wskazanego komponentu. Komponent wykorzystany w przykładzie to srebrny 48-calowy rower Mountain-100 mający w bazie AdventureWorks identyfikator ComponentID równy 774. Część wyników pokazana jest na rysunku 9.3.



Rysunek 9.2. Zależności pomiędzy komponentami i produktami

Listing 9.5. Rekurencyjne CTE zwracające BOM

```

DECLARE @ComponentID int = 774;

WITH BillofMaterialsCTE
(
    BillofMaterialsID,
    ProductAssemblyID,
    ComponentID,
    Quantity,
    Level
)
AS
(
    SELECT
        bom.BillofMaterialsID,
        bom.ProductAssemblyID,
        bom.ComponentID,
        bom.PerAssemblyQty AS Quantity,
        0 AS Level
    FROM Production.BillofMaterials bom
    WHERE bom.ComponentID = @ComponentID

    UNION ALL

    SELECT
        bom.BillofMaterialsID,
        bom.ProductAssemblyID,
        bom.ComponentID,
        bom.PerAssemblyQty,
        Level + 1
    FROM Production.BillofMaterials bom
    INNER JOIN BillofMaterialsCTE bomcte
        ON bom.ProductAssemblyID = bomcte.ComponentID
    WHERE bom.EndDate IS NULL
)

SELECT
    bomcte.ProductAssemblyID,
    p.ProductID,
    p.ProductNumber,
    p.Name,
    p.Color,
    bomcte.Quantity,
    bomcte.Level
FROM BillofMaterialsCTE bomcte
INNER JOIN Production.Product p
    ON bomcte.ComponentID = p.ProductID
ORDER BY bomcte.Level;
  
```

	ProductAssemblyID	ProductID	ProductNumber	Name	Color	Quantity	Level
1	NULL	774	BK-M82S-48	Mountain-100 Silver, 48	Silver	1.00	0
2	774	516	SA-M687	HL Mountain Seat Assembly	NULL	1.00	1
3	774	741	FR-M94S-52	HL Mountain Frame - Silver, 48	Silver	1.00	1
4	774	807	HS-3479	HL Headset	NULL	1.00	1
5	774	810	HB-M918	HL Mountain Handlebars	NULL	1.00	1
6	774	817	FW-M928	HL Mountain Front Wheel	Black	1.00	1
7	774	825	RW-M928	HL Mountain Rear Wheel	Black	1.00	1
8	774	894	RD-2308	Rear Derailleur	Silver	1.00	1
9	774	907	RB-9231	Rear Brakes	Silver	1.00	1
10	774	937	PD-M562	HL Mountain Pedal	Silver/Black	1.00	1
11	774	945	FD-2342	Front Derailleur	Silver	1.00	1
12	774	948	FB-9873	Front Brakes	Silver	1.00	1
13	774	951	CS-9183	HL Crankset	Black	1.00	1
14	774	952	CH-0234	Chain	Silver	1.00	1
15	774	996	BB-9108	HL Bottom Bracket	NULL	1.00	1
16	996	3	BE-2349	BB Ball Bearing	NULL	10.00	2
17	996	526	SH-9312	HL Shell	NULL	1.00	2
18	951	319	CA-7457	HL Crankarm	Black	2.00	2

Rysunek 9.3. Część wyników rekurencyjnego CTE generującego BOM

Tak jak wcześniejsze przykłady CTE, listing 9.3 rozpoczyna się od nazwy CTE i deklaracji listy kolumn.

```
WITH BillOfMaterialsCTE
(
    BillOfMaterialsID,
    ProductAssemblyID,
    ComponentID,
    Quantity,
    Level
)
```

Zapytanie kotwiczące po prostu pobiera z tabeli wiersz, dla którego ComponentID jest równy wskazanemu identyfikatorowi. Jest to komponent najwyższego poziomu z BOM, w tym przypadku 774. Zauważ, że CTE może odwoływać się do zmiennych T-SQL tak, jak @ComponentID w przykładzie.

```
SELECT
    bom.BillOfMaterialsID,
    bom.ProductAssemblyID,
    bom.ComponentID,
    bom.PerAssemblyQty AS Quantity,
    0 AS Level
FROM Production.BillOfMaterials bom
WHERE bom.ComponentID = @ComponentID
```

Rekurencyjne zapytania pobierają kolejne poziomy BOM z CTE, gdzie ProductAssemblyID każdego wiersza ma taką samą wartość jak ComponentID wierszy wyższego poziomu. Chodzi o to, że rekurencyjne zapytanie CTE pobiera wiersze niższych poziomów z hierarchii, korzystając z zależności hierarchicznych pokazanych wcześniej na rysunku 9.2.

```
SELECT
    bom.BillOfMaterialsID,
    bom.ProductAssemblyID,
    bom.ComponentID,
    bom.PerAssemblyQty,
    Level + 1
FROM Production.BillOfMaterials bom
INNER JOIN BillOfMaterialsCTE bomcte
    ON bom.ProductAssemblyID = bomcte.ComponentID
WHERE bom.EndDate IS NULL
```

Zapytanie nazwane ma dołączone zapytanie SELECT łączące wyniki jego działania z tabelą `Production.Product`, dzięki czemu pobierane są szczegółowe informacje na temat produktu, takie jak nazwa i kolor komponentu.

```
SELECT
    bomcte.ProductAssemblyID,
    p.ProductID,
    p.ProductNumber,
    p.Name,
    p.Color,
    bomcte.Quantity,
    bomcte.Level
FROM BillOfMaterialsCTE bomcte
INNER JOIN Production.Product p
    ON bomcte.ComponentID = p.ProductID
```

Ograniczenia prostych CTE opisywane wcześniej w tym rozdziale dotyczą też rekurencyjnych CTE. W rekurencyjnych CTE pojawiają się też dodatkowe ograniczenia.

- Rekurencyjne CTE muszą mieć w treści przynajmniej jedno zapytanie kotwiczące i przynajmniej jedno zapytanie rekurencyjne. Wszystkie zapytania kotwiczące muszą znajdować się przed zapytaniami rekurencyjnymi.
- Wszystkie zapytania kotwiczące muszą być połączone za pomocą operatorów działających na zbiorach UNION, UNION ALL, INTERSECT i EXCEPT. Przy korzystaniu z wielu zapytań kotwiczących i zapytań rekurencyjnych pierwsze zapytanie rekurencyjne i ostatnie zapytanie kotwiczące muszą być połączone za pomocą operatora UNION ALL. Dodatkowo wszystkie zapytania rekurencyjne muszą być połączone przy użyciu UNION ALL.
- Typy danych wszystkich kolumn we wszystkich zapytaniach kotwiczących oraz zapytaniach rekurencyjnych muszą być takie same.
- Klauzula FROM zapytania rekurencyjnego powinna odwoływać się do nazwy CTE tylko raz.
- Zapytania rekurencyjne nie mogą zawierać następujących operatorów i słów kluczowych: GROUP BY, HAVING, LEFT JOIN, RIGHT JOIN, OUTER JOIN i SELECT DISTINCT. Zapytania rekurencyjne nie mogą też zawierać funkcji agregujących (takich jak SUM i MAX), funkcji okna, podzapytań czy wskazówek w rekurencyjnym wywołaniu CTE.

Funkcje okna

SQL Server 2014 zawiera funkcje okna, które pozwalają dzielić wyniki zapytań i dodawać numerowanie, rankingi i funkcje agregujące do każdej części danych (partycji). Kluczowa dla funkcji okna jest klauzula OVER, która pozwala definiować partycje i w niektórych przypadkach kolejność wierszy w partycji. W tym podrozdziale omówione zostaną funkcje okna w SQL Server 2014 wraz z funkcjami rozszerzającymi klauzulę OVER, które umożliwiają numerowanie, porządkowanie i agregowanie.

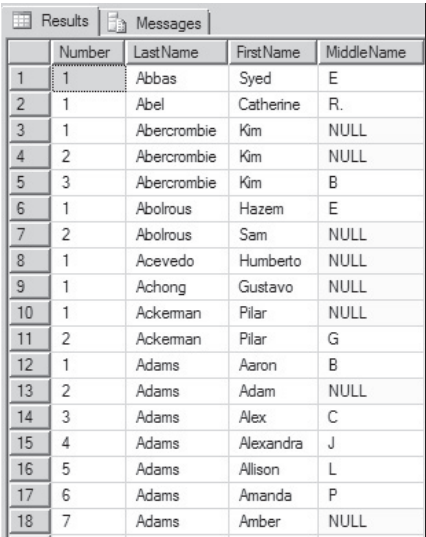
Funkcja ROW_NUMBER

Funkcja ROW_NUMBER przyjmuje klauzulę OVER z klauzulą ORDER BY i opcjonalną klauzulą PARTITION BY. Kod z listingu 9.6 pobiera nazwiska z tabeli `Person.Person`. Klauzula OVER jest wykorzystana do podzielenia wierszy ze względu na zawartość kolumny `LastName` i porządkuje wiersze każdej partycji ze względu na wartości kolumn `LastName`, `FirstName` i `MiddleName`. Funkcja ROW_NUMBER wykorzystana jest do przypisania każdemu wierszowi liczby.

Listing 9.6. ROW_NUMBER z partycjonowaniem

```
SELECT
  ROW_NUMBER() OVER
  (
    PARTITION BY
      LastName
    ORDER BY
      LastName,
      FirstName,
      MiddleName
  ) AS Number,
  LastName,
  FirstName,
  MiddleName
FROM Person.Person;
```

Partycja utworzona na listingu 9.6 działa jak okno przesuwające się przez zestaw wyników (stąd nazwa funkcja okna). Klauzula ORDER BY porządkuje wiersze każdej partycji ze względu na wartości zapisane w kolumnach LastName, FirstName i MiddleName. SQL Server korzysta z funkcji ROW_NUMBER w każdej partycji. Powoduje to, że funkcja ROW_NUMBER numeruje wszystkie wiersze wyniku, zaczynając numerowanie od 1 dla każdej kolejnej wartości LastName, co jest widoczne na rysunku 9.4.



	Number	LastName	FirstName	MiddleName
1	1	Abbas	Syed	E
2	1	Abel	Catherine	R.
3	1	Abercrombie	Kim	NULL
4	2	Abercrombie	Kim	NULL
5	3	Abercrombie	Kim	B
6	1	Abolrous	Hazem	E
7	2	Abolrous	Sam	NULL
8	1	Acevedo	Humberto	NULL
9	1	Achong	Gustavo	NULL
10	1	Ackeman	Pilar	NULL
11	2	Ackeman	Pilar	G
12	1	Adams	Aaron	B
13	2	Adams	Adam	NULL
14	3	Adams	Alex	C
15	4	Adams	Alexandra	J
16	5	Adams	Allison	L
17	6	Adams	Amanda	P
18	7	Adams	Amber	NULL

Rysunek 9.4. Użycie ROW_NUMBER do numerowania wierszy w partycji

■ **Uwaga** Przy korzystaniu z klauzuli PARTITION BY należy umieścić ją przed ORDER BY w klauzuli OVER.

Z funkcji ROW_NUMBER można też korzystać bez klauzuli PARTITION BY i wtedy cały zwracany zbiór danych jest traktowany jak jedna partycja. Traktowanie wszystkich wyników jak pojedynczej partycji może być użyteczne w niektórych przypadkach, ale najczęściej stosuje się partycjonowanie.

Stronicowanie zapytania za pomocą OFFSET/FETCH

SQL Server zawiera różne mechanizmy umożliwiające stronicowanie zwracanych danych.

Tradycyjnym sposobem stronicowania jest wykorzystanie operatora TOP pozwalającego wybrać *n* pierwszych wierszy zwróconych przez zapytanie. SQL Server 2005 wprowadził funkcję ROW_NUMBER; za jej pomocą można osiągnąć podobny efekt w odrobinę inny sposób.

SQL Server 2012 wprowadził do wyrażenia SELECT nowe słowa kluczowe służące do obsługi stronicowania zapytania.

Słowo kluczowe OFFSET umożliwia dużo łatwiejsze stronicowanie. Pozwala ono dosłownie określić numer wiersza, od którego chcesz uzyskać wiersze danych. Następnie możesz użyć FETCH, by pobrać określoną liczbę wierszy wyniku. Jeśli połączysz OFFSET i FETCH z klauzulą ORDER BY, możesz zwrócić dowolnie wybraną część danych wyniku, przesuwając się przez dane wedle potrzeby.

Na listingu 9.7 pokazano stronicowanie wykonane za pomocą OFFSET i FETCH. Procedura składowana korzysta z klauzul OFFSET i FETCH do pobrania wierszy z tabeli Person.Person z bazy danych AdventureWorks na podstawie wartości parametrów przekazanych przy wywołaniu procedury. Procedura określa, w jaki sposób stronicowanie ma być wykonywane za pomocą parametrów wejściowych @RowsPerPage i @StartPageNum. Parametr @RowsPerPage określa, jak wiele wierszy powinno znajdować się na stronie wyników. OFFSET określa liczbę wierszy, jakie należy opuścić, licząc od początku zbioru danych zwracanego przez zapytanie. FETCH określa liczbę wierszy, jaką należy zwrócić na każdej stronie wyników zapytania.

Listing 9.7. Przykład OFFSET/FETCH

```
CREATE PROCEDURE Person.PobierzKontakty
    @StartPageNum int,
    @RowsPerPage int
AS
SELECT
    LastName,
    FirstName,
    MiddleName
FROM Person.Person
ORDER BY
    LastName,
    FirstName,
    MiddleName
OFFSET (@StartPageNum - 1) * @RowsPerPage ROWS
FETCH NEXT @RowsPerPage ROWS ONLY;
GO
```

Przykładowe wywołanie procedury wykorzystującej klauzulę OFFSET/FETCH

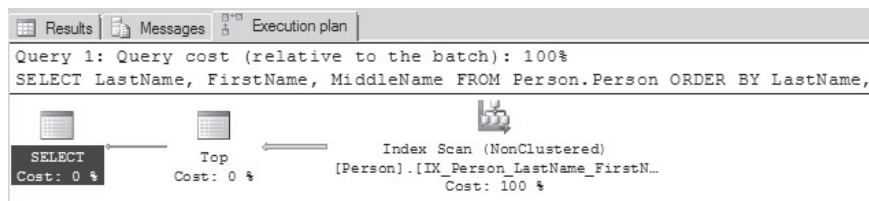
```
EXEC Person.PobierzKontakty 16,10
```

przekazuje wartość 10 jako parametr @RowsPerPage i wartość 16 jako parametr @StartPageNum oraz zwraca 10 wierszy strony 16, co widać na rysunku 9.5. Słowo kluczowe OFFSET w wyrażeniu SELECT omija wiersze przed stroną określoną wartościami parametrów @StartPageNum i @RowsPerPage — w tym przykładzie omija 150 wierszy i zaczyna zwracać wyniki od 151. wiersza. Słowo kluczowe FETCH zwraca tyle wierszy, ile określono w zmiennej @RowsPerPage (10). Plan zapytania jest pokazany na rysunku 9.6.

Zapytanie z listingu 9.7 jest dużo bardziej czytelnym i eleganckim rozwiązaniem stronicowania wyników niż wykorzystanie klauzuli TOP czy funkcji ROW_NUMBER z zapytaniami nazwanymi. Jedynym wyjątkiem będzie sytuacja, gdy korzystasz z OFFSET/FETCH i zechcesz uzyskać ROW_NUMBER; w takim przypadku będziesz musiał dodać ROW_NUMBER do swojego zapytania. Dlatego klauzula OFFSET/FETCH to dużo bardziej przejrzysty sposób implementowania stronicowania *ad hoc*.

	LastName	FirstName	MiddleName
1	Alexander	Cassidy	NULL
2	Alexander	Chloe	NULL
3	Alexander	Christian	C
4	Alexander	Connor	G
5	Alexander	Dakota	NULL
6	Alexander	Dalton	NULL
7	Alexander	David	NULL
8	Alexander	Destiny	NULL
9	Alexander	Devin	NULL
10	Alexander	Dylan	NULL

Rysunek 9.5. Użycie `OFFSET` i `FETCH` do implementacji stronicowania po stronie klienta



Rysunek 9.6. Plan zapytania dla implementacji stronicowania po stronie klienta za pomocą `OFFSET` i `FETCH`

Korzystając z `OFFSET` i `FETCH`, pamiętaj o pewnych ograniczeniach.

- `OFFSET` i `FETCH` muszą być stosowane z klauzulą `ORDER BY`.
- `FETCH` nie można używać bez `OFFSET`, ale `OFFSET` można użyć bez `FETCH`.
- Liczba wierszy określona w klauzuli `OFFSET` musi być większa lub równa 0.
- Liczba wierszy określona w klauzuli `FETCH` musi być większa lub równa 1.
- W zapytaniach, w których używasz `OFFSET` i `FETCH`, nie możesz korzystać z operatora `TOP`.
- Wartości `OFFSET` i `FETCH` muszą być stałe lub muszą być parametrami o wartości całkowitej.
- `OFFSET` i `FETCH` nie są obsługiwane z klauzulą `OVER`.
- `OFFSET` i `FETCH` nie są obsługiwane dla widoków indeksowanych i widoków z `WITH CHECK OPTION`.

Generalnie w SQL Server 2012 i nowszych połączenie `OFFSET` i `FETCH` jest najbardziej przejrzystym podejściem do stronicowania wyników zapytania.

Funkcje `RANK` i `DENSE_RANK`

Funkcje `RANK` i `DENSE_RANK` to funkcje numerujące dostępne w SQL Server. Obie te funkcje przypisują wartość liczbową do każdego wiersza w partycji, ale różnią się sposobem numerowania wierszy. Oto przykład.

- Jeśli masz trzy wartości 7, 7 i 9, funkcja `RANK` przypisze im numery 1, 1 i 3. Dzieje się tak dlatego, że dwie 7 są przypisane do pierwszego miejsca, podczas gdy 9 jest trzecia na liście. `RANK` nie uwzględnia tego połączenia przy obliczaniu numeru dla wartości 9.
- Funkcja `DENSE_RANK` w takiej sytuacji przypisze numery 1, 1 i 2. Dzieje się tak, ponieważ `DENSE_RANK` łączy obie 7 na pozycji pierwszej i umieszcza 9 na drugiej pozycji.

Nie ma dobrego lub złego sposobu numerowania danych w przypadku braku wymagań biznesowych. SQL Server udostępnia dwa sposoby i możesz wybrać ten, który lepiej pasuje do wymagań biznesowych.

Załóżmy, że chcesz ustalić najlepsze dni sprzedaży firmy *AdventureWorks* w roku kalendarzowym 2012. Może to być opisane pytaniem: „W jakie dni mieliśmy najlepszą sprzedaż w 2012 roku?”. Funkcja RANK pozwala łatwo uzyskać takie informacje, co widać na listingu 9.8. Część wyników można zobaczyć na rysunku 9.7.

Listing 9.8. *Ranking dziennej sprzedaży AdventureWorks*

```
WITH TotalSalesBySalesDate
(
    DailySales,
    OrderDate
)
AS
(
    SELECT
        SUM(soh.SubTotal) AS DailySales,
        soh.OrderDate
    FROM Sales.SalesOrderHeader soh
    WHERE soh.OrderDate >= '20120101'
        AND soh.OrderDate < '20130101'
    GROUP BY soh.OrderDate
)
SELECT
    RANK() OVER
    (
        ORDER BY
            DailySales DESC
    ) AS Ranking,
    DailySales,
    OrderDate
FROM TotalSalesBySalesDate
ORDER BY Ranking;
```

	Ranking	DailySales	OrderDate
1	1	3617451,9232	2012-06-30 00:00:00.000
2	2	3064632,131	2012-09-30 00:00:00.000
3	3	2888213,6087	2012-07-31 00:00:00.000
4	4	2408187,2542	2012-05-30 00:00:00.000
5	5	2402984,3967	2012-12-31 00:00:00.000
6	6	2306297,6081	2012-03-30 00:00:00.000
7	7	2205251,3127	2012-10-30 00:00:00.000
8	8	1919379,3786	2012-01-01 00:00:00.000
9	9	1815297,7111	2012-08-30 00:00:00.000
10	10	1473667,5818	2012-01-29 00:00:00.000
11	11	1329986,583	2012-11-30 00:00:00.000
12	12	1016116,847	2012-04-30 00:00:00.000
13	13	895032,9488	2012-02-29 00:00:00.000
14	14	46135,95	2012-03-14 00:00:00.000
15	15	44273,4146	2012-05-22 00:00:00.000
16	16	42354,40	2012-02-21 00:00:00.000
17	17	40352,6064	2012-05-18 00:00:00.000
18	18	40174,3264	2012-03-29 00:00:00.000

Rysunek 9.7. Ranking wartości dziennej sprzedaży *AdventureWorks*

Listing 9.8 zawiera CTE zwracające dwie kolumny, `DailySales` i `OrderDate`. Kolumna `DailySales` to suma sprzedaży pogrupowana według wartości `OrderDate`. Wyniki są ograniczane przez klauzulę `WHERE` do 2012 roku.

```
WITH TotalSalesBySalesDate
(
    DailySales,
    OrderDate
)
AS
(
    SELECT
        SUM(soh.SubTotal) AS DailySales,
        soh.OrderDate
    FROM Sales.SalesOrderHeader soh
    WHERE soh.OrderDate >= '20120101'
        AND soh.OrderDate < '20130101'
    GROUP BY soh.OrderDate
)
```

Funkcja `RANK` jest wykorzystywana z klauzulą `OVER`, by numerować wiersze zwracane przez CTE w kolejności malejącej (od największych do najmniejszych) wartości kolumny `DailySales`.

```
SELECT
    RANK() OVER
    (
        ORDER BY
            DailySales DESC
    ) AS Ranking,
    DailySales,
    OrderDate
FROM TotalSalesBySalesDate
ORDER BY Ranking;
```

Tak jak funkcja `ROW_NUMBER`, funkcja `RANK` może przyjmować klauzulę `PARTITION BY` w klauzuli `OVER`. Kod z listingu 9.9 rozszerza poprzedni przykład i wykorzystuje klauzulę `PARTITION BY` do utworzenia rankingu dziennej sprzedaży dla każdego miesiąca. Tego typu zapytanie może odpowiedzieć na pytanie biznesowe: „W jakie dni każdego miesiąca 2012 roku sprzedaż firmy *AdventureWorks* była największa?”. Część wyników można zobaczyć na rysunku 9.8.

Listing 9.9. Ustalenie rankingów dziennej sprzedaży w podziale na miesiące

```
WITH TotalSalesBySalesDatePartitioned
(
    DailySales,
    OrderMonth,
    OrderDate
)
AS
(
    SELECT
        SUM(soh.SubTotal) AS DailySales,
        DATENAME(MONTH, soh.OrderDate) AS OrderMonth,
        soh.OrderDate
    FROM Sales.SalesOrderHeader soh
    WHERE soh.OrderDate >= '20120101'
        AND soh.OrderDate < '20130101'
    GROUP BY soh.OrderDate
)
SELECT
    RANK() OVER
    (
        PARTITION BY
            OrderMonth
```



```

ORDER BY
  DailySales DESC
) AS Ranking,
DailySales,
OrderMonth,
OrderDate
FROM TotalSalesBySalesDatePartitioned
ORDER BY DATEPART(mm,OrderDate),
  Ranking;

```

	Ranking	DailySales	OrderMonth	OrderDate
1	1	1919379,3786	January	2012-01-01 00:00:00.000
2	2	1473667,5818	January	2012-01-29 00:00:00.000
3	3	40060,0682	January	2012-01-11 00:00:00.000
4	4	39856,7882	January	2012-01-26 00:00:00.000
5	5	35401,14	January	2012-01-10 00:00:00.000
6	5	35401,14	January	2012-01-22 00:00:00.000
7	7	29325,2582	January	2012-01-24 00:00:00.000
8	8	24844,61	January	2012-01-03 00:00:00.000
9	9	24666,33	January	2012-01-13 00:00:00.000
10	10	24087,7328	January	2012-01-06 00:00:00.000
11	11	23363,6346	January	2012-01-31 00:00:00.000
12	12	22867,8164	January	2012-01-14 00:00:00.000
13	13	21990,4382	January	2012-01-30 00:00:00.000
14	14	21469,62	January	2012-01-15 00:00:00.000
15	15	21088,06	January	2012-01-07 00:00:00.000
16	16	20884,78	January	2012-01-05 00:00:00.000
17	17	18590,4482	January	2012-01-02 00:00:00.000
18	17	18590,4482	January	2012-01-19 00:00:00.000

Rysunek 9.8. Część wyników rankingów dziennej sprzedaży w podziale na miesiące

Zapytanie z listingu 9.9, tak jak przykład z listingu 9.8, zaczyna się od CTE obliczającego dzienną sprzedaż dla wszystkich dni roku. Główna różnica między tym CTE i poprzednim przykładem jest taka, że kod z listingu 9.9 zwraca dodatkową kolumnę OrderMonth. Oto treść tego CTE.

```

WITH TotalSalesBySalesDatePartitioned
(
  DailySales,
  OrderMonth,
  OrderDate
)
AS
(
  SELECT
    SUM(soh.SubTotal) AS DailySales,
    DATENAME(MONTH, soh.OrderDate) AS OrderMonth,
    soh.OrderDate
  FROM Sales.SalesOrderHeader soh
  WHERE soh.OrderDate >= '20120101'
    AND soh.OrderDate < '20130101'
  GROUP BY soh.OrderDate
)

```

Zapytanie SELECT powiązane z CTE korzysta z funkcji RANK, by przypisać pozycje rankingowe wynikom. Klauzula PARTITION BY została wykorzystana do podzielenia wyników według OrderMonth, dzięki czemu pozycje rankingowe zaczynają się od 1 dla każdego miesiąca. Oto przykład.

```

SELECT
  RANK() OVER
  (
    PARTITION BY
      OrderMonth
    ORDER BY
      DailySales DESC
  ) AS Ranking,
  DailySales,
  OrderMonth,
  OrderDate
FROM TotalSalesBySalesDatePartitioned
ORDER BY DATEPART(mm, OrderDate),
  Ranking;

```

Gdy funkcja RANK spotyka dwie równe wartości DailySales w tej samej partycji, przypisuje taką samą pozycję w rankingu obu wartościom i pomija kolejną liczbę. Na rysunku 9.9 można zobaczyć, że wartość DailySales dla dwóch dni w październiku 2012 wynosiła 7479,3221 dolarów, co sprawiło, że funkcja RANK przypisała obu dniom w rankingu wartość 25. Następnie pominęła wartość 26 i przypisała kolejnemu wierszowi w rankingu wartość 27.

	Ranking	DailySales	OrderMonth	OrderDate
296	22	8761.1932	October	2012-10-16 00:00:00.000
297	23	8745.4303	October	2012-10-01 00:00:00.000
298	24	7712.5325	October	2012-10-04 00:00:00.000
299	25	7479.3221	October	2012-10-21 00:00:00.000
300	25	7479.3221	October	2012-10-08 00:00:00.000
301	27	7302.5178	October	2012-10-18 00:00:00.000
302	28	7036.8471	October	2012-10-06 00:00:00.000
303	28	7036.8471	October	2012-10-22 00:00:00.000
304	30	5297.7596	October	2012-10-14 00:00:00.000
305	31	2443.35	October	2012-10-03 00:00:00.000

Rysunek 9.9. Funkcja RANK pomija kolejną liczbę w przypadku dwóch identycznych wartości

Funkcja DENSE_RANK, tak jak funkcja RANK, przypisuje identycznym wartościom taką samą pozycję w rankingu, ale z jedną ważną różnicą — nie pomija kolejnej pozycji rankingu. Na listingu 9.10 znajduje się kod z listingu 9.9 zmodyfikowany w taki sposób, by korzystać z funkcji RANK i DENSE_RANK. Jak możesz zobaczyć na rysunku 9.10, DENSE_RANK też przypisuje taką samą wartość Ranking do obu identycznych wartości sprzedaży, ale w odróżnieniu od funkcji RANK nie pomija kolejnej wartości Ranking.

Listing 9.10. Użycie DENSE_RANK do uporządkowania najlepszych wyników sprzedaży w miesiącu

```

WITH TotalSalesBySalesDatePartitioned
(
  DailySales,
  OrderMonth,
  OrderDate
)
AS
(
  SELECT
    SUM(soh.SubTotal) AS DailySales,
    DATENAME(MONTH, soh.OrderDate) AS OrderMonth,
    soh.OrderDate
  FROM Sales.SalesOrderHeader soh
  WHERE soh.OrderDate >= '20120101'
    AND soh.OrderDate < '20130101'
  GROUP BY soh.OrderDate

```

```

)
SELECT
  RANK() OVER
  (
    PARTITION BY
      OrderMonth
    ORDER BY
      DailySales DESC
  ) AS Ranking,
DENSE_RANK() OVER
(
  PARTITION BY
    OrderMonth
  ORDER BY
    DailySales DESC
) AS Dense_Ranking,
DailySales,
OrderMonth,
OrderDate
FROM TotalSalesBySalesDatePartitioned
ORDER BY DATEPART(mm,OrderDate),
  Ranking;

```

	Ranking	Dense_Ranking	DailySales	OrderMonth	OrderDate
1	1	1	1919379,3786	January	2012-01-01 00:00:00.000
2	2	2	1473667,5818	January	2012-01-29 00:00:00.000
3	3	3	40060,0682	January	2012-01-11 00:00:00.000
4	4	4	39856,7882	January	2012-01-26 00:00:00.000
5	5	5	35401,14	January	2012-01-10 00:00:00.000
6	5	5	35401,14	January	2012-01-22 00:00:00.000
7	7	6	29325,2582	January	2012-01-24 00:00:00.000
8	8	7	24844,61	January	2012-01-03 00:00:00.000
9	9	8	24666,33	January	2012-01-13 00:00:00.000
10	10	9	24087,7328	January	2012-01-06 00:00:00.000
11	11	10	23363,6346	January	2012-01-31 00:00:00.000
12	12	11	22867,8164	January	2012-01-14 00:00:00.000
13	13	12	21990,4382	January	2012-01-30 00:00:00.000
14	14	13	21469,62	January	2012-01-15 00:00:00.000
15	15	14	21088,06	January	2012-01-07 00:00:00.000
16	16	15	20884,78	January	2012-01-05 00:00:00.000

Rysunek 9.10. DENSE_RANK nie pomija kolejnej pozycji rankingu po powieleniu

Funkcja NTILE

NTILE to inna funkcja do tworzenia rankingów, która odpowiada na odrobinę inne zapotrzebowanie. Funkcja ta dzieli wyniki na przybliżone n -tyle. N -tylem może być kwartył (podział na kawałki o wielkości 1/4 lub 25%), kwintyl (podział na kawałki o wielkości 1/5 lub 20%) czy percentyl (podział na kawałki o wielkości 1/100 lub 1%) albo każda inna część całości, jaką możesz sobie wyobrazić. NTILE dzieli zbiór wyników na przybliżone n -tyle, ponieważ liczba zwracanych wierszy nie musi być dokładnie podzielna przez określoną liczbę grup. Tabela z 27 wierszami na przykład nie da się dokładnie podzielić na kwartyle czy kwintyle. Jeśli odpytujesz za pomocą funkcji NTILE i liczba wierszy nie da się dokładnie podzielić na określoną liczbę grup, NTILE tworzy grupy o dwóch rozmiarach. Większe grupy mają o jeden wiersz więcej niż mniejsze grupy i są umieszczane na początku. Gdy przykładowo 27 wierszy jest podzielonych na kwintyle (1/5), dwie pierwsze grupy mają po 6 wierszy, a ostatnie trzy grupy mają po 5 wierszy.

Tak jak w przypadku funkcji ROW_NUMBER, do klauzuli OVER możesz dołączyć zarówno PARTITION BY, jak i ORDER BY. Funkcja NTILE wymaga dodatkowego parametru, który określa, na jak wiele grup trzeba podzielić wyniki zapytania.

Funkcja NTILE przydaje się przy odpowiadaniu na pytania biznesowe w stylu: „Którzy sprzedawcy zaliczają się do 25% najlepszych w lipcu 2013?” oraz „Jakie były wyniki ich sprzedaży?”. Kod z listingu 9.11 korzysta z NTILE, by podzielić sprzedawców AdventureWorks na cztery grupy, z których każda obejmuje 25% pracowników. Klauzula ORDER BY sprawia, że wiersze są przypisywane do grupy na podstawie wyników sprzedaży. Wyniki zostały pokazane na rysunku 9.11.

Listing 9.11. Użycie NTILE do grupowania i utworzenia rankingu sprzedawców

```
WITH SalesTotalBySalesPerson
(
    SalesPersonID, SalesTotal
)
AS
(
    SELECT
        soh.SalesPersonID,
        SUM(soh.SubTotal) AS SalesTotal
    FROM Sales.SalesOrderHeader soh
    WHERE DATEPART(YEAR, soh.OrderDate) = 2013
        AND DATEPART(MONTH, soh.OrderDate) = 2
    GROUP BY soh.SalesPersonID
)
SELECT
    NTILE(4) OVER( ORDER BY st.SalesTotal DESC) AS Tile,
    p.LastName,
    p.FirstName,
    p.MiddleName,
    st.SalesPersonID,
    st.SalesTotal
FROM SalesTotalBySalesPerson st
INNER JOIN Person.Person p ON st.SalesPersonID = p.BusinessEntityID;
```

	Tile	LastName	FirstName	MiddleName	SalesPersonID	SalesTotal
1	1	Pak	Jae	B	289	430730,8928
2	1	Carson	Jillian	NULL	277	400651,4944
3	1	Blythe	Michael	G	275	314936,4504
4	1	Reiter	Tsvi	Michael	279	172527,4835
5	2	Campbell	David	R	283	155124,1524
6	2	Mitchell	Linda	C	276	88379,2611
7	2	Ito	Shu	K	281	87934,131
8	2	Varkey Chudukatil	Ranjit	R	290	56935,0391
9	3	Ansman-Wolfe	Pamela	O	280	49152,4316
10	3	Saraiva	José	Edvaldo	282	47113,0052
11	3	Jiang	Stephen	Y	274	43254,2036
12	4	Mensa-Annan	Tete	A	284	9479,9522
13	4	Vargas	Garrett	R	278	8091,5083
14	4	Alberts	Amy	E	287	968,4284

Rysunek 9.11. Sprzedawcy AdventureWorks pogrupowani i uszeregowani za pomocą NTILE

Na początku kodu znajduje się proste wyrażenie nazwane, zwracające wartość SalesPersonID i sumę wartości SubTotal dla zamówień z tabeli Sales.SalesOrderHeader. CTE ogranicza wyniki do wyników sprzedaży z lutego 2013 roku. Oto treść tego CTE.

```

WITH SalesTotalBySalesPerson
(
    SalesPersonID, SalesTotal
)
AS
(
    SELECT
        soh.SalesPersonID,
        SUM(soh.SubTotal) AS SalesTotal
    FROM Sales.SalesOrderHeader soh
    WHERE DATEPART(YEAR, soh.OrderDate) = 2013
        AND DATEPART(MONTH, soh.OrderDate) = 2
    GROUP BY soh.SalesPersonID
)

```

Zapytanie SELECT dołączone do tego CTE korzysta z NTILE(4) do grupowania sprzedawców *AdventureWorks* w czterech grupach zawierających około 25%. Klauzula OVER wskazuje, że grupy powinny być tworzone na podstawie SalesTotal w kolejności malejącej. Całe zapytanie SELECT wygląda tak.

```

SELECT
    NTILE(4) OVER( ORDER BY st.SalesTotal DESC) AS Tile,
    p.LastName,
    p.FirstName,
    p.MiddleName,
    st.SalesPersonID,
    st.SalesTotal
FROM SalesTotalBySalesPerson st
INNER JOIN Person.Person p ON st.SalesPersonID = p.BusinessEntityID;

```

Funkcje agregacyjne, analityczne i klauzula OVER

Jak już wcześniej napisano, funkcje numerujące i tworzące rankingi (ROW_NUMBER, RANK i tak dalej) współpracują z klauzulą OVER przy definiowaniu kolejności i partycjonowania przekazanych do nich wierszy za pomocą klauzul ORDER BY i PARTITION BY. Klauzula OVER dostarcza też funkcjonalności okna do funkcji agregacyjnych, takich jak SUM, COUNT i zdefiniowane przez użytkownika agregacje SQL CLR.

Funkcje okna mogą pomóc przy odpowiadaniu na typowe pytania biznesowe obejmujące tworzenie sum częściowych i średnich kroczących. Możesz na przykład użyć klauzuli OVER na tabeli Purchasing.PurchaseOrderDetail z bazy danych *AdventureWorks*, aby uzyskać sumę wartości zamówionych produktów w postaci sumy kroczącej. Możesz dodatkowo ograniczyć zakres zwracanych danych, na których zechcesz użyć funkcji agregującej, partycjonując zbiór wyników według PurchaseOrderId, co w efekcie wygeneruje podsumę dla każdego zamówienia. Przykładowe zapytanie pokazane zostało na listingu 9.12. Część jego wyników można zobaczyć na rysunku 9.12.

Listing 9.12. Użycie klauzuli OVER z SUM

```

SELECT
    PurchaseOrderID,
    ProductID,
    OrderQty,
    UnitPrice,
    LineTotal,
    SUM(LineTotal)
    OVER (PARTITION BY PurchaseOrderID ORDER BY ProductId
        RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)
    AS CumulativeOrderQty
FROM Purchasing.PurchaseOrderDetail;

```

	PurchaseOrderID	ProductID	OrderQty	UnitPrice	LineTotal	CumulativeOrderQty
1	1	1	4	50,26	201,04	201,04
2	2	359	3	45,12	135,36	135,36
3	2	360	3	45,5805	136,7415	272,1015
4	3	530	550	16,086	8847,30	8847,30
5	4	4	3	57,0255	171,0765	171,0765
6	5	512	550	37,086	20397,30	20397,30
7	6	513	550	26,5965	14628,075	14628,075
8	7	317	550	27,0585	14882,175	14882,175
9	7	318	550	33,579	18468,45	33350,625
10	7	319	550	46,0635	25334,925	58685,55
11	8	403	3	47,4705	142,4115	142,4115
12	8	404	3	45,3705	136,1115	278,523
13	8	405	3	49,644	148,932	427,455
14	8	406	3	45,3705	136,1115	563,5665
15	8	407	3	43,2705	129,8115	693,378
16	9	422	3	47,523	142,569	142,569
17	9	423	3	45,423	136,269	278,838
18	9	424	3	49,6965	149,0895	427,9275

Rysunek 9.12. Część wyników z zapytania generującego sumę kroczącą

Na listingu 9.12 znalazła się nowa klauzula.

RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW

Nazywana jest **klauzulą ramki** (ang. *framing clause*) i w tym przypadku określa, że każda suma zawiera wszystkie wartości od pierwszego wiersza partycji do bieżącego wiersza. Klauzula taka ma sens tylko wtedy, kiedy wiersze są uporządkowane, a aby to uzyskać, korzystamy z klauzuli ORDER BY ProductId. Klauzula ramki w połączeniu z klauzulą ORDER BY generuje sumę kroczącą widoczną na rysunku 9.12.

-
- **Wskazówka** Istnieją też inne klauzule ramki. Zapisana na listingu 9.12 klauzula RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW jest wartością domyślną przyjmowaną w sytuacji, gdy klauzula ramki nie zostanie jawnie określona. Pamiętaj, że często piszący zapytania są zdumieni niespodziewanymi wynikami, ponieważ nie wiedzą, iż używana jest domyślna klauzula ramki.
-

Przeanalizujmy przykład, by zobaczyć, jak domyślna klauzula ramki może wpłynąć na wyniki zapytania. Powiedzmy, że zechcesz obliczyć i zwrócić sumę sprzedaży w podziale według PurchaseOrder dla każdego wiersza. W zależności od sposobu zdefiniowania ramki, możesz uzyskać bardzo różne wyniki, ponieważ suma może oznaczać całkowitą sumę lub sumę kroczącą. Zmodyfikujmy zapytanie z listingu 9.12, określmy klauzulę ramki RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING obok domyślnej klauzuli ramki i sprawdźmy wyniki. Zmodyfikowane zapytanie pokazane jest na listingu 9.13, a wyniki jego działania widać na rysunku 9.13.

Listing 9.13. Sumowanie z różnymi ramkami

```
SELECT
    PurchaseOrderID,
    ProductID,
    OrderQty,
    UnitPrice,
    LineTotal,
```

```

SUM(LineTotal)
  OVER (PARTITION BY PurchaseOrderID ORDER BY ProductId)
AS TotalSalesDefaultFraming,
SUM(LineTotal)
  OVER (PARTITION BY PurchaseOrderID ORDER BY ProductId
        RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING)
AS TotalSalesDefinedFraming
FROM Purchasing.PurchaseOrderDetail
ORDER BY PurchaseOrderID;

```

	PurchaseOrderID	ProductID	OrderQty	UnitPrice	LineTotal	TotalSalesDefaultFraming	TotalSalesDefinedFraming
1	1	1	4	50,26	201,04	201,04	201,04
2	2	359	3	45,12	135,36	135,36	272,1015
3	2	360	3	45,5805	136,7415	272,1015	272,1015
4	3	530	550	16,086	8847,30	8847,30	8847,30
5	4	4	3	57,0255	171,0765	171,0765	171,0765
6	5	512	550	37,086	20397,30	20397,30	20397,30
7	6	513	550	26,5965	14628,075	14628,075	14628,075
8	7	317	550	27,0585	14882,175	14882,175	58685,55
9	7	318	550	33,579	18468,45	33350,625	58685,55
10	7	319	550	46,0635	25334,925	58685,55	58685,55
11	8	403	3	47,4705	142,4115	142,4115	693,378
12	8	404	3	45,3705	136,1115	278,523	693,378
13	8	405	3	49,644	148,932	427,455	693,378
14	8	406	3	45,3705	136,1115	563,5665	693,378
15	8	407	3	43,2705	129,8115	693,378	693,378
16	9	422	3	47,523	142,569	142,569	694,1655
17	9	423	3	45,423	136,269	278,838	694,1655
18	9	424	3	49,6965	149,0895	427,9275	694,1655

Rysunek 9.13. Część wyników zapytania z różnymi ramkami

Na rysunku 9.13 możesz zobaczyć, że suma sprzedaży w ostatnich dwóch kolumnach znacząco się różni. W kolumnie 6., `TotalSalesDefaultFraming`, znajduje się suma krocząca wyników sprzedaży — ponieważ nie została określona ramka dla tej kolumny, użyta została domyślna ramka `RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW`, co oznacza, że funkcja agregująca jest obliczana tylko do bieżącego wiersza. Jednak w przypadku kolumny 7., `TotalSalesDefinedFraming`, określona jest klauzula ramki `RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING`, co oznacza, że ramka jest rozciągana na wszystkie wiersze partycji i dlatego suma jest obliczana ze wszystkich pozycji o tej samej wartości `PurchaseOrderID`. Celem jest obliczenie i zwrócenie sumy sprzedaży dla zamówienia w każdym wierszu, ale brak doprecyzowania rodzaju ramki powoduje zwrócenie sumy kroczącej. Ten przykład pokazuje, jak ważne jest określenie odpowiedniej ramki, aby uzyskać pożądaný wynik.

Przyjrzyjmy się teraz innemu przykładowi. Na listingu 9.14 znajduje się zmodyfikowany kod z listingu 9.13. Dalej używamy w nim klauzuli `OVER` na tabeli `Purchasing.PurchaseOrderDetail` z bazy danych `AdventureWorks`, ale tym razem chcemy pobrać dwudniową średnią wartości zamówionych produktów. Wyniki są posortowane według `DueDate`. Zauważ inną klauzulę ramki w tym zapytaniu; jest to `ROWS BETWEEN 1 PRECEDING AND CURRENT ROW`. Wiersze są sortowane według dat. Dla każdego wiersza dla dwudniowej średniej pobierany jest bieżący wiersz oraz wiersz opisujący poprzedni dzień. Część wyników tego zapytania można zobaczyć na rysunku 9.14.

Listing 9.14. Użycie klauzuli *OVER* do zdefiniowania ramki zwracającej dwudniową średnią kroczącą

```

SELECT
    PurchaseOrderID,
    ProductID,
    DueDate,
    LineTotal,
    Avg(LineTotal)
      OVER (ORDER BY DueDate
            ROWS BETWEEN 1 PRECEDING AND CURRENT ROW) AS [2DayAvg]
FROM Purchasing.PurchaseOrderDetail
ORDER BY DueDate;

```

	PurchaseOrderID	ProductID	DueDate	LineTotal	2DayAvg
1	1	1	2011-04-30 00:00:00.000	201.04	201.04
2	2	359	2011-04-30 00:00:00.000	135.36	168.20
3	2	360	2011-04-30 00:00:00.000	136.7415	136.0507
4	3	530	2011-04-30 00:00:00.000	8847.30	4492.0207
5	4	4	2011-04-30 00:00:00.000	171.0765	4509.1882
6	5	512	2011-05-14 00:00:00.000	20397.30	10284.1882
7	6	513	2011-05-14 00:00:00.000	14628.075	17512.6875
8	7	317	2011-05-14 00:00:00.000	14882.175	14755.125
9	7	318	2011-05-14 00:00:00.000	18468.45	16675.3125
10	7	319	2011-05-14 00:00:00.000	25334.925	21901.6875
11	8	403	2011-05-14 00:00:00.000	142.4115	12738.6682
12	8	404	2011-05-14 00:00:00.000	136.1115	139.2615
13	8	405	2011-05-14 00:00:00.000	148.932	142.5217
14	8	406	2011-05-14 00:00:00.000	136.1115	142.5217
15	8	407	2011-05-14 00:00:00.000	129.8115	132.9615
16	9	422	2011-12-28 00:00:00.000	142.569	136.1902
17	9	423	2011-12-28 00:00:00.000	136.269	139.419
18	9	424	2011-12-28 00:00:00.000	149.0895	142.6792

Rysunek 9.14. Część wyników zapytania zwracającego dwudniową średnią kroczącą

Przyjrzyjmy się jeszcze jednemu scenariuszowi, w którym obliczamy sumę kroczącą sprzedaży z grupowaniem po *ProductID*, by dostarczyć zarządowi informacje o tym, które produkty szybko się sprzedają. Na listingu 9.15 znajduje się zmodyfikowane zapytanie z listingu 9.14 ze zdefiniowanymi dodatkowymi oknami partycjonującymi wyniki według *ProductID*. Możesz zobaczyć, jak zwiększa się rozmiar ramki podczas wykonywania obliczeń w ramce. Po zmianie *ProductID* ramka jest tworzona od nowa i obliczenia również zaczynane są od nowa. Na rysunku 9.15 pokazana jest część wyników.

Listing 9.15. Definiowanie ramek wewnątrz klauzuli *OVER* do obliczania sumy kroczącej

```

SELECT
    PurchaseOrderID,
    ProductID,
    OrderQty,
    UnitPrice,
    LineTotal,
    SUM(LineTotal)
      OVER (PARTITION BY ProductId ORDER BY DueDate
            RANGE BETWEEN UNBOUNDED PRECEDING AND
            CURRENT ROW) AS CumulativeTotal,
    ROW_NUMBER()
      OVER (PARTITION BY ProductId ORDER BY DueDate ) AS No
FROM Purchasing.PurchaseOrderDetail
ORDER BY ProductId, DueDate;

```


	PurchaseOrderID	ProductID	OrderQty	UnitPrice	LineTotal	CumulativeTotal	No
44	3378	1	3	50,2635	150,7905	6685,0315	44
45	3457	1	3	50,2635	150,7905	6835,822	45
46	3536	1	3	50,2635	150,7905	6986,6125	46
47	3615	1	3	50,2635	150,7905	7137,403	47
48	3694	1	3	50,2635	150,7905	7288,1935	48
49	3773	1	3	50,2635	150,7905	7438,984	49
50	3852	1	3	50,2635	150,7905	7589,7745	50
51	3931	1	3	50,2635	150,7905	7740,565	51
52	79	2	3	41,916	125,748	125,748	1
53	158	2	3	41,916	125,748	251,496	2
54	237	2	3	41,916	125,748	377,244	3
55	316	2	3	41,916	125,748	502,992	4
56	395	2	3	41,916	125,748	628,74	5
57	425	2	3	41,916	125,748	754,488	6
58	504	2	3	41,916	125,748	880,236	7
59	587	2	3	41,916	125,748	1005,984	8
60	674	2	3	41,916	125,748	1131,732	9
61	757	2	3	41,916	125,748	1257,48	10

Rysunek 9.15. Część wyników zawierających sumę kroczącą liczoną dla kolejnych ProductID

Możesz też zobaczyć w zapytaniu z listingu 9.15, że nie jesteś ograniczony do korzystania z jednej funkcji agregującej w wyrażeniu SELECT. Możesz użyć wielu funkcji agregujących w tym samym zapytaniu.

Ramki można definiować, używając wierszy (ROWS) lub zakresów (RANGE) z dolnym i górnym ograniczeniem. Jeśli zdefiniujesz tylko dolne ograniczenie, górnym ograniczeniem będzie bieżący wiersz. Jeśli zdefiniujesz ramkę, korzystając z ROWS, możesz określić granicę za pomocą liczby lub wyrażenia skalarne go zwracającego liczbę całkowitą. Jeśli nie zdefiniujesz granicy ramki, przyjmowana jest wartość domyślna RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW.

Przykłady funkcji analitycznych

W SQL Server 2012 wprowadzono kilka pomocnych funkcji analitycznych. Niektóre bardziej użyteczne z nich zostały opisane w kolejnych punktach. Niektóre są bardziej statystyczne, inne są przydatne przy tworzeniu raportów, w których musisz odwoływać się do wartości z wielu zwracanych wierszy.

CUME_DIST i PERCENT_RANK

CUME_DIST i PERCENT_RANK to dwie funkcje analityczne, które zostały wprowadzone w SQL Server 2012. Przyjmijmy, że chcesz ustalić, jakie wyniki mają najlepsi, średni i najslabsi sprzedawcy *AdventureWorks*, i porównać je. Szczególnie zainteresowany jesteś danymi niejakiej Jillian Carson, na której nazwisko natknąłeś się przy wstępnym przeglądaniu danych. W języku biznesowym może być to wyrażone pytaniem: „Jakie są wyniki Jillian Carson w porównaniu z innymi sprzedawcami?”. CUME_DIST pozwala w łatwy sposób uzyskać taką informację, co widać na listingu 9.16. Wyniki zapytania pokazane zostały na rysunku 9.16.

Listing 9.16. Użycie funkcji CUME_DIST

```

SELECT
    round(SUM(TotalDue),1) AS Sales,
    LastName,
    FirstName,
    SalesPersonId,
    CUME_DIST() OVER (ORDER BY round(SUM(TotalDue),1)) as CUME_DIST
FROM
    Sales.SalesOrderHeader soh
    JOIN Sales.vSalesPerson sp
        ON soh.SalesPersonID = sp.BusinessEntityID
GROUP BY SalesPersonID,LastName,FirstName;

```

	Sales	LastName	FirstName	SalesPersonId	CUME_DIST
1	195528,80	Abbas	Syed	285	0,0588235294117647
2	826417,50	Alberts	Amy	287	0,117647058823529
3	1235934,40	Jiang	Stephen	274	0,176470588235294
4	1606441,40	Tsofilas	Lynn	286	0,235294117647059
5	2062393,10	Valdez	Rachel	288	0,294117647058824
6	2608116,40	Mensa-Annan	Tete	284	0,352941176470588
7	3748246,10	Ansman-Wolfe	Pamela	280	0,411764705882353
8	4069422,20	Vargas	Garrett	278	0,470588235294118
9	4207894,60	Campbell	David	283	0,529411764705882
10	5087977,20	Varkey Chudukatil	Ranjit	290	0,588235294117647
11	6683536,70	Saraiva	José	282	0,647058823529412
12	7259567,90	Ito	Shu	281	0,705882352941177
13	8086073,70	Reiter	Tsvi	279	0,764705882352941
14	9585124,90	Pak	Jae	289	0,823529411764706
15	10475367,10	Blythe	Michael	275	0,882352941176471
16	11342385,90	Carson	Jillian	277	0,941176470588235
17	11695019,10	Mitchell	Linda	276	1

Rysunek 9.16. Wyniki obliczeń CUME_DIST

Zapytanie z listingu 9.16 zaokrągliło wartość `TotalDue` i zwraca wartość `Sales` w postaci bardziej czytelnej. Ponieważ `CUME_DIST` zwraca pozycję wiersza, wartości w kolumnie są zwracane w postaci ułamka dziesiętnego. Procentowe wyniki można uzyskać, mnożąc tę wartość przez 100. Wyniki z rysunku 9.16 pokazują, że 94,11% wszystkich sprzedawców ma sprzedaż gorszą od sprzedaży Jillian Carson lub jej równą, o czym mówi wartość dystrybucyjności 0,9411.

Jeśli odrobinę zmodyfikujesz pytanie do postaci: „W jakim percentyli znajduje się sprzedaż Jillian Carson?”, odpowiedź możesz uzyskać za pomocą `PERCENT_RANK`. Na listingu 9.17 znajduje się zmodyfikowana wersja zapytania z listingu 9.16, zawierająca wywołanie `PERCENT_RANK`. Część wyników można zobaczyć na rysunku 9.17.

Listing 9.17. Użycie funkcji PERCENT_RANK

```

SELECT
    round(SUM(TotalDue),1) AS Sales,
    LastName,
    FirstName,
    SalesPersonId,
    CUME_DIST() OVER (ORDER BY round(SUM(TotalDue),1)) as CUME_DIST,
    PERCENT_RANK() OVER (ORDER BY round(SUM(TotalDue),1)) as PERCENT_RANK
FROM Sales.SalesOrderHeader soh
    JOIN Sales.vSalesPerson sp
        ON soh.SalesPersonID = sp.BusinessEntityID
GROUP BY SalesPersonID,LastName,FirstName;

```

	Sales	LastName	FirstName	SalesPersonId	CUME_DIST	PERCENT_RANK
1	195528,80	Abbas	Syed	285	0,0588235294117647	0
2	826417,50	Alberts	Amy	287	0,117647058823529	0,0625
3	1235934,40	Jiang	Stephen	274	0,176470588235294	0,125
4	1606441,40	Tsoflias	Lynn	286	0,235294117647059	0,1875
5	2062393,10	Valdez	Rachel	288	0,294117647058824	0,25
6	2608116,40	Mensa-Annan	Tete	284	0,352941176470588	0,3125
7	3748246,10	Ansman-Wolfe	Pamela	280	0,411764705882353	0,375
8	4069422,20	Vargas	Garrett	278	0,470588235294118	0,4375
9	4207894,60	Campbell	David	283	0,529411764705882	0,5
10	5087977,20	Varkey Chudukatil	Ranjit	290	0,588235294117647	0,5625
11	6683536,70	Saraiva	José	282	0,647058823529412	0,625
12	7259567,90	Ito	Shu	281	0,705882352941177	0,6875
13	8086073,70	Reiter	Tsvi	279	0,764705882352941	0,75
14	9585124,90	Pak	Jae	289	0,823529411764706	0,8125
15	10475367,10	Blythe	Michael	275	0,882352941176471	0,875
16	11342385,90	Carson	Jillian	277	0,941176470588235	0,9375
17	11695019,10	Mitchell	Linda	276	1	1

Rysunek 9.17. Wyniki obliczeń CUME_DIST i PERCENT_RANK dla sprzedawców

Funkcja PERCENT_RANK zwraca pozycję wśród wszystkich sprzedawców w *AdventureWorks*. Jak możesz zobaczyć, pojawia się tu 17 unikalnych wartości: pierwszą jest 0, a ostatnią 1. Inne wiersze mają wartości mniejsze od 1. W tym przykładzie Jillian Carson ma percentyl 93,75% wśród sprzedawców *AdventureWorks*, co opisuje wartość PERCENT_RANK równa 0,9375.

■ **Uwaga** Możesz użyć klauzuli PARTITION BY na funkcjach CUME_DIST i PERCENT_RANK, aby zdefiniować okno, na którym będziesz przeprowadzał te obliczenia.

PERCENTILE_CONT i PERCENTILE_DISC

PERCENTILE_CONT i PERCENTILE_DISC to nowe funkcje, które są odwrotnościami funkcji CUME_DIST i PERCENT_RANK. Przyjmijmy, że zechcesz ustalić 40 percentyl sprzedaży dla wszystkich kont *AdventureWorks*. Można to wyrazić w języku biznesowym pytaniem: „Jaki jest 40 percentyl całej sprzedaży dla wszystkich kont?”. PERCENTILE_CONT i PERCENTILE_DISC wymagają klauzuli WITHIN GROUP określającej kolejność i kolumny do obliczeń. PERCENTILE_CONT interpoluje wszystkie wartości okna i na podstawie tego oblicza wynik. PERCENTILE_DISC zwraca istniejącą wartość. Zarówno PERCENTILE_CONT, jak i PERCENTILE_DISC wymagają podania percentyla jako argumentu o wartości z zakresu od 0,0 do 1,0. Przykład z listingu 9.18 oblicza sumę sprzedaży dla 40 percentyla w podziale na konta. W przykładzie wykorzystane są PERCENTILE_CONT i PERCENTILE_DISC z wartością 0,4 jako percentyl do obliczenia oznaczającą 40 percentyl. Wyniki zapytania pokazane są na rysunku 9.18.

Listing 9.18. Użycie PERCENTILE_CONT i PERCENTILE_DISC

```
SELECT
    round(SUM(TotalDue),1) AS Sales,
    LastName,
    FirstName,
    SalesPersonId,
    AccountNumber,
    PERCENTILE_CONT(0.4) WITHIN GROUP (ORDER BY round(SUM(TotalDue),1))
    OVER(PARTITION BY AccountNumber ) AS PERCENTILE CONT,
    PERCENTILE_DISC(0.4) WITHIN GROUP (ORDER BY round(SUM(TotalDue),1))
```

```

OVER(PARTITION BY AccountNumber ) AS PERCENTILE_DISC
FROM
Sales.SalesOrderHeader soh
JOIN Sales.vSalesPerson sp
ON soh.SalesPersonID = sp.BusinessEntityID
GROUP BY AccountNumber,SalesPersonID,LastName,FirstName

```

	Sales	LastName	FirstName	SalesPersonID	AccountNumber	PERCENTILE_CONT	PERCENTILE_DISC
1	95924,00	Ansman-Wolfe	Pamela	280	10-4020-000001	95924	95924,00
2	28310,00	Campbell	David	283	10-4020-000002	28310	28310,00
3	176830,40	Carson	Jillian	277	10-4020-000003	198391,28	176830,40
4	230732,60	Blythe	Michael	275	10-4020-000003	198391,28	176830,40
5	222309,60	Carson	Jillian	277	10-4020-000004	308720,28	222309,60
6	438336,30	Blythe	Michael	275	10-4020-000004	308720,28	222309,60
7	97031,20	Ito	Shu	281	10-4020-000005	97031,2	97031,20
8	3023,30	Mitchell	Linda	276	10-4020-000006	3023,3	3023,30
9	3302,50	Jiang	Stephen	274	10-4020-000007	4166,78	3302,50
10	5463,20	Mitchell	Linda	276	10-4020-000007	4166,78	3302,50
11	25064,60	Reiter	Tsvi	279	10-4020-000008	25064,6	25064,60
12	14250,80	Reiter	Tsvi	279	10-4020-000009	14250,8	14250,80
13	500181,70	Pak	Jae	289	10-4020-000010	500181,7	500181,70
14	54905,20	Saraiva	José	282	10-4020-000011	54905,2	54905,20
15	281414,80	Vargas	Garrett	278	10-4020-000012	281414,8	281414,80
16	8393,50	Valdez	Rachel	288	10-4020-000014	8393,5	8393,50
17	163865,50	Tsofilas	Lynn	286	10-4020-000015	163865,5	163865,50
18	56147,10	Alberts	Amy	287	10-4020-000016	224876,46	56147,10

Rysunek 9.18. Wyniki z funkcji `PERCENTILE_CONT` i `PERCENTILE_DISC`

Na rysunku 9.18 możesz zobaczyć, że wartości `PERCENTILE_CONT` i `PERCENTILE_DISC` różnią się w zależności od numeru konta. Dla konta o numerze 10-4020-000003, niezależnie od sprzedawcy, `PERCENTILE_CONT` ma wartość 198391,28, która jest wartością interpolowaną i może nie występować w zbiorze danych. `PERCENTILE_DISC` ma wartość 176830,40, co jest równe wartości istniejącej w zbiorze danych. Dla konta 10-4020-000004 `PERCENTILE_CONT` ma wartość 308720,28, a `PERCENTILE_DISC` ma wartość 222309,60.

LAG i LEAD

LAG i LEAD to nowe funkcje przesunięcia, które umożliwiają wykonywanie obliczeń dla określonego wiersza znajdującego się przed bieżącym wierszem lub po nim. Funkcje te umożliwiają dostęp do więcej niż jednego wiersza bez konieczności tworzenia złączenia typu *self-join*. Funkcja LAG udostępnia wiersze poprzedzające bieżący wiersz, a LEAD daje dostęp do wierszy znajdujących się za bieżącym wierszem.

LAG pomaga odpowiedzieć na pytania biznesowe: „Jakie są bieżące i historyczne koszty produkcji dla wszystkich produktów obecnie produkowanych?”. Na listingu 9.19, pokazany jest przykład zapytania liczącego bieżące koszty produkcji i wcześniejsze koszty produkcji dla wszystkich aktywnych produktów za pomocą funkcji LAG. Część wyników znajduje się na rysunku 9.19.

Listing 9.19. Użycie funkcji LAG

```

WITH ProductCostHistory AS
(SELECT
    ProductID,
    LAG(StandardCost)
      OVER (PARTITION BY ProductID ORDER BY ProductID) AS PreviousProductCost,

```

```

StandardCost AS CurrentProductCost,
Startdate,Enddate
FROM Production.ProductCostHistory
)
SELECT
ProductID,
PreviousProductCost,
CurrentProductCost,
StartDate,
EndDate
FROM ProductCostHistory
WHERE Enddate IS NULL

```

	ProductID	PreviousProductCost	CurrentProductCost	StartDate	EndDate
1	707	13,8782	13,0863	2013-05-30 00:00:00.000	NULL
2	708	13,8782	13,0863	2013-05-30 00:00:00.000	NULL
3	711	13,8782	13,0863	2013-05-30 00:00:00.000	NULL
4	712	5,2297	6,9223	2013-05-30 00:00:00.000	NULL
5	713	29,0807	38,4923	2013-05-30 00:00:00.000	NULL
6	714	29,0807	38,4923	2013-05-30 00:00:00.000	NULL
7	715	29,0807	38,4923	2013-05-30 00:00:00.000	NULL
8	716	29,0807	38,4923	2013-05-30 00:00:00.000	NULL
9	717	722,2568	868,6342	2013-05-30 00:00:00.000	NULL
10	718	722,2568	868,6342	2013-05-30 00:00:00.000	NULL
11	719	722,2568	868,6342	2013-05-30 00:00:00.000	NULL
12	720	722,2568	868,6342	2013-05-30 00:00:00.000	NULL
13	721	722,2568	868,6342	2013-05-30 00:00:00.000	NULL
14	722	170,1428	204,6251	2013-05-30 00:00:00.000	NULL
15	723	170,1428	204,6251	2013-05-30 00:00:00.000	NULL
16	724	170,1428	204,6251	2013-05-30 00:00:00.000	NULL
17	736	170,1428	204,6251	2013-05-30 00:00:00.000	NULL
18	737	170,1428	204,6251	2013-05-30 00:00:00.000	NULL

Rysunek 9.19. Wyniki pozwalające na porównanie historycznych kosztów produkcji za pomocą funkcji LAG

W tym przykładzie w listingu 9.19 skorzystano z funkcji LAG w CTE do obliczenia różnicy pomiędzy bieżącymi kosztami produkcji i wcześniejszymi poprzez partycjonowanie zbioru danych względem ProductID.

```

SELECT
ProductID,
LAG(StandardCost)
OVER (PARTITION BY ProductID ORDER BY ProductID) AS PreviousProductCost,
StandardCost AS CurrentProductCost,
Startdate,Enddate
FROM Production.ProductCostHistory

```

Wyrażenie SELECT związane z CTE zwraca wiersze zawierające najnowsze koszty produkcji ze zbioru danych przy użyciu wartości NULL w kolumnie EndDate za pomocą poniższego wywołania.

```

SELECT
ProductID,
PreviousProductCost,
CurrentProductCost,
StartDate,
EndDate
FROM ProductCostHistory
WHERE Enddate IS NULL

```

Funkcja LEAD, która jest przeciwieństwem LAG, pomaga odpowiadać na pytania biznesowe, na przykład takie: „Jak miesięczne wyniki sprzedaży wypadają w porównaniu ze sprzedażą w kolejnym miesiącu dla wszystkich sprzedawców *AdventureWorks* w roku 2012?”. Listing 9.20 zawiera przykładowe zapytanie, które zawiera sprzedaż z kolejnego miesiąca pobraną za pomocą funkcji LEAD dla porównania ze sprzedażą w bieżącym miesiącu dla danych z roku 2012. Częściowe wyniki są pokazane na rysunku 9.20.

Listing 9.20. Użycie funkcji LEAD

```
SELECT
    LastName,
    SalesPersonID,
    Sum(SubTotal) CurrentMonthSales,
    DateName(Month,OrderDate) Month,
    DateName(Year,OrderDate) Year,
    LEAD(Sum(SubTotal),1)
    OVER (ORDER BY SalesPersonID, OrderDate) TotalSalesNextMonth
FROM Sales.SalesOrderHeader soh
JOIN Sales.vSalesPerson sp
ON soh.SalesPersonID = sp.BusinessEntityID
WHERE DateName(Year,OrderDate) = 2012
GROUP BY FirstName, LastName, SalesPersonID,OrderDate
ORDER BY SalesPersonID,OrderDate;
```

	LastName	SalesPersonID	CurrentMonthSales	Month	Year	TotalSalesNextMonth
132	Alberts	287	12648,1326	September	2012	29649,0509
133	Alberts	287	29649,0509	December	2012	631704,2589
134	Pak	289	631704,2589	May	2012	331640,849
135	Pak	289	331640,849	June	2012	474009,9807
136	Pak	289	474009,9807	July	2012	393554,9573
137	Pak	289	393554,9573	August	2012	283398,3465
138	Pak	289	283398,3465	September	2012	408527,5444
139	Pak	289	408527,5444	October	2012	257637,1408
140	Pak	289	257637,1408	November	2012	233804,9696
141	Pak	289	233804,9696	December	2012	97496,2916
142	Varkey ...	290	97496,2916	May	2012	262749,5275
143	Varkey ...	290	262749,5275	June	2012	163687,8125
144	Varkey ...	290	163687,8125	July	2012	55789,7317
145	Varkey ...	290	55789,7317	August	2012	180614,7458
146	Varkey ...	290	180614,7458	September	2012	87995,2931
147	Varkey ...	290	87995,2931	October	2012	34516,2132
148	Varkey ...	290	34516,2132	November	2012	113442,2926
149	Varkey ...	290	113442,2926	December	2012	NULL

Rysunek 9.20. Wyniki porównania sprzedaży handlowców w 2012 roku za pomocą funkcji LEAD

Na rysunku 9.20 możesz zobaczyć, że ostatni wiersz zwraca NULL dla sprzedaży w kolejnym miesiącu, ponieważ nie ma wartości LEAD dla ostatniego wiersza.

FIRST_VALUE i LAST_VALUE

FIRST_VALUE i LAST_VALUE to funkcje przesunięcia zwracające pierwszą i ostatnią wartość w oknie zdefiniowanym za pomocą klauzuli OVER. Funkcja FIRST_VALUE zwraca pierwszą wartość z okna, a LAST_VALUE zwraca ostatnią wartość z okna.

Funkcje te pomagają odpowiedzieć na pytania: „Jaka jest kwota pierwszej i ostatniej transakcji w miesiącu dla danego sprzedawcy?”. Na listingu 9.21 znajduje się przykładowe zapytanie, a na rysunku 9.21 pokazano część wyników zwracanych przez to zapytanie.

Listing 9.21. Użycie *FIRST_VALUE* i *LAST_VALUE*

```

SELECT DISTINCT
  LastName,
  SalesPersonID,
  datename(year,OrderDate) OrderYear,
  datename(month, OrderDate) OrderMonth,
  FIRST_VALUE(SubTotal)
    OVER (PARTITION BY SalesPersonID, OrderDate ORDER BY SalesPersonID )
  FirstSalesAmount,
  LAST_VALUE(SubTotal)
    OVER (PARTITION BY SalesPersonID, OrderDate ORDER BY SalesPersonID)
  LastSalesAmount,
  OrderDate
FROM Sales.SalesOrderHeader soh
  JOIN Sales.vSalesPerson sp
    ON soh.SalesPersonID = sp.BusinessEntityID
ORDER BY OrderDate;

```

	LastName	SalesPersonID	OrderYear	OrderMonth	FirstSalesAmount	LastSalesAmount	OrderDate
1	Blythe	275	2011	May	6122,082	20541,4072	2011-05-31 00:00:00.000
2	Mitchell	276	2011	May	419,4589	5056,4896	2011-05-31 00:00:00.000
3	Carson	277	2011	May	6107,082	874,794	2011-05-31 00:00:00.000
4	Vargas	278	2011	May	7793,1108	1316,0575	2011-05-31 00:00:00.000
5	Reiter	279	2011	May	20565,6206	419,4589	2011-05-31 00:00:00.000
6	Ansman-Wolfe	280	2011	May	24432,6088	24432,6088	2011-05-31 00:00:00.000
7	Ito	281	2011	May	9799,9243	38510,8973	2011-05-31 00:00:00.000
8	Saraiva	282	2011	May	32726,4786	2624,382	2011-05-31 00:00:00.000
9	Campbell	283	2011	May	14352,7713	3463,2998	2011-05-31 00:00:00.000
10	Jiang	274	2011	July	20544,7015	20544,7015	2011-07-01 00:00:00.000
11	Blythe	275	2011	July	32940,1556	4049,988	2011-07-01 00:00:00.000
12	Mitchell	276	2011	July	3911,5991	68,40	2011-07-01 00:00:00.000
13	Carson	277	2011	July	1718,8983	32971,323	2011-07-01 00:00:00.000
14	Vargas	278	2011	July	919,5201	50948,9161	2011-07-01 00:00:00.000
15	Reiter	279	2011	July	8580,0739	52803,4477	2011-07-01 00:00:00.000
16	Ansman-Wolfe	280	2011	July	10993,3942	32459,9278	2011-07-01 00:00:00.000

Rysunek 9.21. Wyniki pokazujące wartość pierwszej i ostatniej transakcji

Przykład ten zwraca wielkość pierwszej i ostatniej transakcji dla każdej osoby w podziale na miesiące i lata. Na rysunku 9.21 możesz zobaczyć, że w niektórych przypadkach wartości *FirstSalesAmount* i *LastSalesAmount* są takie same, co oznacza, że w danym miesiącu odbyła się tylko jedna transakcja. W miesiącach z więcej niż jedną transakcją wymienione są wartości pierwszej i ostatniej transakcji.

Podsumowanie

Wyrażenia nazwane to użyteczne narzędzie SQL Server istniejące w dwóch odmianach, rekurencyjnej i nierekurencyjnej. Nierekurencyjne CTE pozwalają na pisanie bardziej przejrzystego kodu T-SQL łatwiejszego w tworzeniu, debugowaniu i zarządzaniu od złożonych zapytań korzystających z tabel pośrednich. Rekurencyjne CTE upraszczają pisanie zapytań do pobierania danych hierarchicznych i umożliwiają proste generowanie przydatnych w wielu zastosowaniach zbiorów danych, zawierających kolejne liczby całkowite.

SQL Server obsługuje funkcje okna, a klauzula `OVER` upraszcza korzystanie z funkcji agregujących z uwzględnieniem okien i porządkowania. SQL Server obsługuje kilka poniższych funkcji okna.

- `ROW_NUMBER` numeruje zwracane wiersze, rozpoczynając od 1.
- `RANK` i `DENSE_RANK` tworzą ranking zwracanych wyników, dopisując przy numerowaniu taką samą liczbę do wierszy zawierających pola o takiej samej wartości.
- `NTILE` dzieli zbiór wyników na określoną przez użytkownika liczbę grup.
- `CUME_DIST`, `PERCENTILE_CONT`, `PERCENT_RANK` i `PERCENTILE_DISC` to funkcje analityczne T-SQL, umożliwiające między innymi obliczanie dystrybucji.
- `LAG` i `LEAD` umożliwiają odwoływanie się do sąsiednich wierszy oddalonych o określoną liczbę wierszy.
- `FIRST_VALUE` i `LAST_VALUE` zwracają pierwszy i ostatni wiersz okna zdefiniowanego przez klauzulę partycjonującą dane.

Można też wykorzystać klauzulę `OVER` do utworzenia okna przy korzystaniu z wbudowanych funkcji agregujących oraz funkcji agregujących SQL CLR utworzonych przez użytkownika.

Zarówno wyrażenia nazwane, jak i funkcje okna są bardzo użyteczne i rozszerzają składnię T-SQL, pozwalając na pisanie kodu potężniejszego niż kiedykolwiek za pomocą dużo prostszych konstrukcji.

Ćwiczenia

1. Czy jeśli CTE nie jest pierwszym wyrażeniem w ciągu, poprzedzające je wyrażenie musi być zakończone średnikiem?
2. Co jest wymagane w rekurencyjnym CTE:
 - a. słowo kluczowe `WITH`,
 - b. zapytanie kotwiczące,
 - c. słowo kluczowe `EXPRESSION`,
 - d. zapytanie rekurencyjne.
3. Opcja `MAXRECURSION` przyjmuje wartości pomiędzy 0 i
4. Które z poniższych funkcji okna są obsługiwane przez SQL Server:
 - a. `ROW_NUMBER`,
 - b. `RANK`,
 - c. `DENSE_RANK`,
 - d. `NTILE`,
 - e. wszystkie powyższe.
5. Czy można użyć `ORDER BY` w klauzuli `OVER` wykorzystanej z funkcjami agregującymi?
6. Czy jeśli `PARTITION BY` i `ORDER BY` są wykorzystane w klauzuli `OVER`, klauzula `PARTITION BY` musi pojawić się pierwsza?
7. Nazwy wszystkich kolumn zwracanych przez CTE muszą być
8. Domyślną klauzulą ramki jest
9. Czy jeśli nie jest określona klauzula `ORDER BY` dla funkcji niewymagającej stosowania klauzuli `OVER`, ramka okna obejmuje całą partycję?

Skorowidz

A

- ADO.NET, 417
- ADO.NET 4.5, 502, 521
- ADO.NET Entity Data Model, 512
- adres e-mail, 492
- AES, Advanced Encryption Standard, 202
- agregat UDA, 480
- aktualizacja widoku, 190
- aktualnie wykonywany SQL, 403
- aktywne zbiory wyników, 438
- algorytm Soundex, 96
- algorytmy
 - do szyfrowania kluczy, 220
 - szyfrowania symetrycznego, 210
- alokacja
 - danych, 545
 - obiektów użytkownika, 409
 - przestrzeni, 544
- analiza filtrowania, 317
- API, application programming interface, 30, 127
- aplikacje sieciowe, 513
- architektura
 - FTS, 297, 298
 - SOA, 510
- arkusz stylów XSLT, 348
- atomowość, 26
 - wykonywania procedury, 124
- atrybuty
 - CLR UDT, 487
 - XPath, 355

B

- baza danych, 116
 - AdventureWorks, 61
 - klucze główne, 201
 - systemowa, 25
 - master, 25
 - model, 25
 - msdb, 25
 - resource, 25
 - tempdb, 25
 - użytkownika, 25
- BCP, Bulk Copy Program, 58, 431
- bezpieczeństwo SQL CLR, 350
- białe znaki, 31
- blok
 - BEGIN ... END, 66
 - BEGIN TRY...END TRY, 526, 527
 - try...catch, 493
- blokowane zapytania, 407
- błędy, 526
 - poziomy istotności, 527
- BOL, Books Online, 60
- BOM, bills of materials, 228
- breakpoint, 40, 45

C

- certifikat serwera, 220
- certyfikaty, 203
- CLR, Common Language Runtime, 125, 459
 - funkcje użytkownika, 465
 - pakiety, 461
 - procedury składowane, 472

CLR, Common Language Runtime
 TVF, 471
 typy użytkownika, 484
 UDA, 476
 UDF, 469
 wyzwalacze, 491
 Code Snippet Manager, 42
 CORBA, 510
 CRUD, create, read, update, delete, 30
 CTE, Common Table Expression, 72, 95, 223
 czytelność kodu, 226
 rekurencyjne, 227
 CWA, closed-world assumption, 65
 czas
 oczekiwania, 572
 wojskowy, 261
 czytanie planów zapytań, 566

D

dane
 FILESTREAM, 285
 typu MultiPolygon, 279
 data i czas, 258
 DataTip, 45
 DCL, Data Control Language, 25, 191
 DCOM, 510
 DDL, Data Definition Language, 24
 debugger
 T-SQL w Visual Studio, 533
 Debugger Watch, 40
 zintegrowany z SSMS, 532
 debugowanie, 43, 530–534
 definiowanie
 ramek, 244
 źródła danych, 512
 DER, Distinguished Encoding Rules, 204
 Designer, 440
 deszyfrowanie, 208
 diagram przepływu, 34, 36
 DMF, Dynamic Management Functions, 143, 398
 DMK, 201
 DML, Data Manipulation Language, 24, 104
 DMV, Dynamic Management View, 121, 143, 398
 dodatek LocalDB, 498, 522
 dodawanie
 elementu, 463
 fragmentu kodu, 43
 grupy plików, 156
 kontenera optymalizowanego, 157

liczb zespolonych, 490
 usługi danych, 515
 dokumentacja BOL, 60
 DOM, Document Object Model, 323
 domena danych, 27
 dopasowywanie wyrazów, 317
 dostęp
 do danych SqlConnection, 419
 do fragmentów kodów, 41
 FILESTREAM, 283
 DRI, cascading declarative referential integrity, 28
 drzewo zapytania, 447
 DST, Daylight Saving Time, 265
 dynamiczna tabela przestawna, 77
 dynamiczne
 generowanie XML, 371
 konstruktory, 372
 tworzenie elementów, 372
 widoki zarządcze, 317, 393, 397
 zapytanie SQL, 78, 537
 dynamiczny SQL, 536, 537
 dysk SSD, 158
 dzielenie liczb całkowitych, 380
 dziennik transakcji, 26

E

EKM, Extensible Key Management, 199, 218
 eksplorator obiektów, 40, 50
 elementy wyrażenia, 23
 Enterprise Manager, 39
 Entity Data Model Designer, 514
 Entity Framework, 448
 ETL, Extract Transform Load, 58, 394
 Extended Events, 57, 573

F

FILESTREAM, 156, 281
 flagi śledzenia, 531
 FOR XML
 AUTO, 329
 EXPLICIT, 331
 PATH, 332, 353
 RAW, 327
 format
 daty XQuery, 377
 GML, 277
 tabeli krawędziowej, 325
 WKT, 277

fragmenty kodu
 dostosowane, 41
 otaczające, 41
 rozszerzane, 41
 FTS, full-text search, 297, 307
 funkcja
 APP_NAME, 181
 CHOOSE, 80
 COALESCE, 81
 COLUMNS_UPDATED, 184
 CONTAINSTABLE, 312
 CONVERT, 262
 CUME, 246
 CUME_DIST, 245
 data, 357
 DecryptByCert, 206
 DENSE_RANK, 234, 238
 DoImport, 433
 EncryptByPassPhrase, 218
 EVENTDATA, 192
 FIRST_VALUE, 250
 FnCommaSplit, 115
 FORMAT, 262
 FREETEXTTABLE, 311, 313
 GetPathLocator, 293
 HOST_NAME, 181
 IDENTITY, 257
 LAG, 248
 LAST_VALUE, 250
 LEAD, 248, 250
 LoadSourceFile, 434
 Main(), 433
 NEWID, 266
 NTILE, 239, 240
 NULLIF, 81
 OPENXML, 321
 PERCENT_RANK, 245
 PERCENTILE_CONT, 247
 PERCENTILE_DISC, 247
 RAISERROR, 186
 RANK, 234
 ROW_NUMBER, 231
 semantickeyphrasetable, 320
 SignByCert, 207
 sql:column, 382
 SUSER_SNAME, 181
 SWITCHOFFSET, 265
 sys.dm_db_indexphysical_stats, 85
 sys.dm_fts_parser, 317

TODATETIMEOFFSET, 264
 TRY_CAST, 528
 TRY_CONVERT, 528
 TRY_PARSE, 528
 UPDATE, 184, 188
 XACT_STATE, 527
 funkcje
 agregacyjne, 241
 agregujące użytkownika, UDA, 476
 analityczne, 241, 245
 bloku CATCH, 527
 daty i czasu, 262
 filetable, 290
 niedeterministyczne, 115
 okna, 223, 231
 przesunięcia, 250
 skalarne, 91
 użytkownika, UDF, 30, 91, 465
 ograniczenia, 115
 skalarne UDF, 91
 wielowyrażeniowe TVF, 104
 zwracające tabelę, 104
 w XQuery, 380
 zwracające tabelę
 wbudowane, 112
 wielowyrażeniowe, 104

G

generator wierszy, 323
 generowanie planu zapytania, 568
 GML, Geography Markup Language, 274, 277
 graficzne plany wykonania zapytań, 40, 48
 grupa plików FILESTREAM, 287
 grupowanie elementów, 356, 357
 grupy
 plików, 157, 283, 543
 zdarzeń, 191
 GUI, 40
 GUID, Globally Unique Identifier, 256

H

hierarchia
 instancji typów przestrzennych, 275
 szyfrowania, 199, 200

I

IIS, 20
 iloczyn kartezjański XQuery, 386
 implementacja stronicowania, 234
 import danych, 58
 indeks, 411, 559
 columnstore, 29
 klastrowy, 29
 nieklastrowy, 29, 149
 nieklastrowy pamięciowy, 30
 nieklastrowy typu hash, 30
 pamięciowy, 29
 pokrywający, 564
 przestrzenny, 29
 XML, 29
 drugiego rzędu, 343
 główny, 343
 indeksy
 filtrowane, 565
 klastrowe, 560, 561
 nieklastrowe, 562
 pełnotekstowe, 29
 tabel, 398
 tabel OLTP, 164
 typu hash, 165
 zakresu, 167
 informacje
 o konfiguracji serwera, 410
 o połączeniu, 403
 o sesji, 402
 o woluminach, 410
 o zrzutach SQL Server, 411
 inicjalizacja zmiennych, 37
 instalacja LocalDB, 498
 instancja, 25
 LocalDB, 499
 instrukcja RETURN, 124
 IntelliSense, 40
 interfejs, 127
 ITU, International Telecommunication Union, 260
 izolacja, 26

J

JDBC, 508
 język SEQUEL, 21
 języki
 deklaratywne, 22
 imperatywne, 22
 JSON, JavaScript Object Notation, 510

K

kanał RSS, 469, 471
 kaskadowe zapewnienie integralności odwołań,
 DRI, 28
 kasowanie procedur składowanych, 124
 kategoria
 DMF, 398
 DMV, 398
 klasa
 SqlBulkCopy, 431
 SqlTriggerContext, 493
 System.Data.SqlTypes, 418
 klasy natywnego klienta SQL, 419
 klauzula, 23
 .WRITE, 255, 256
 check, 28
 CREATION_DISPOSITION, 211
 DECRYPTION BY PASSWORD, 208
 ENCRYPTION BY, 211
 EXECUTABLE FILE, 204
 FILE, 204
 FOR XML AUTO, 329
 FOR XML EXPLICIT, 331
 FOR XML PATH, 332, 353
 FOR XML RAW, 327
 FROM ASSEMBLY, 204
 IDENTITYVALUE, 211
 join, 446
 OFFSET/FETCH, 233
 order by, 387
 orderby, 445
 OVER, 241, 244
 PARTITION BY, 232
 PROVIDER_KEY_NAME, 211
 ramki, framing clause, 242
 REMOVE PRIVATE KEY, 208
 RETURNS, 108
 TRUE, 311
 USING XML INDEX, 346
 where, 444
 WITH, 325
 WITH PERMISSION_SET, 350
 WITH PRIVATE KEY, 204, 208
 WITH XMLNAMESPACES, 360
 klient SQL .NET, 419
 klucz
 ciągu opisującego połączenie, 421
 do szyfrowania bazy danych, 220

klucze
 asymetryczne, 207, 208, 210
 główne, 201
 główne usługi, 200
 obce, 27, 28
 symetryczne, 210
 tymczasowe, 212

kod
 CLR, 125
 proceduralny, 63, 96

kodowanie
 ciągów znaków, 98
 fonetyczne NYSIS, 97
 NYSIS, 97, 102, 103

kolejność wykonania, 24

kolizja funkcji skrótu, 217

kolorowanie skryptów, 40

kolumny
 bez nazw, 356
 rzadkie, sparse columns, 555

komentarze XQuery, 373

kompresja
 danych, 550
 stron, 552
 wierszy, 550

konfiguracja FILESTREAM, 283

konsola DBCC, 86

konstruktory, 383

kontekst działania, 123

konwencje nazewnictwa, 32

konwersja
 BOM, 269
 typów System.Data.SqlTypes, 418

kopia zapasowa
 certyfikatu, 204
 klucza asymetrycznego, 210

koszt
 I/O dla zapytania, 548
 wykonania zapytania, 344, 345

kreator
 ADO.NET Entity Data Model, 512
 indeksów pełnotekstowych, 304
 katalogu pełnotekstowego, 299
 LINQ to SQL, 440

kursor, 82, 87, 401

L

liczby zespolone, 485, 491

LINQ, Language Integrated Query, 439

LINQ to SQL, 439

Linux, 503

lista produktów, 111
 pobieranie, 132
 zwracanie, 130

LOB, Large Object, 281

LocalDB, 498, 522

logi transakcyjne, 26

logika trójwartościowa, 24, 63

lokalizacja, 364

Ł

łączenie
 ciągów znaków, 256
 wierszy, 111

M

mapowanie
 schematu bazy danych, 440
 skrótów klawiaturowych, 44

mapy alokacji, 545

MDI, multiple-document interface, 45

mechanizm
 Extended Events, 57
 FTS, 297
 IntelliSense, 40
 LocalDB, 497

menu
 Debug, 532
 kontekstowe Full-Text Index, 301

metadane, 121, 181
 indeksowe, 398
 tabel i kolumn, 394

metoda
 ExecuteNonQuery, 428
 ExecuteReader, 427
 ExecuteScalar, 428
 ExecuteXmlReader, 429
 exist, 339, 352
 Fill, 424
 GetEnvironmentVars, 473
 Main, 442
 modify, 341, 352
 nodes, 340, 352
 query, 338, 352, 363
 SendResultsRow, 475
 ToString, 488
 Transaction.Current.Rollback, 493
 value, 339, 352

metody
 hierarchiid, 273
 typu danych xml, 337
 typu hierarchiid, 274
 model
 danych, 450
 uwierzytelniania, 500
 moduły SQL CLR, 30
 modyfikowanie
 danych, 178, 455
 procedur składowanych, 124
 produktów, 187
 MSFTESQL, 307
 MSXML, 322

N

najdroższe zapytania, 406
 narzędzia, 39
 diagnostyczne, 57
 do debugowania, 530
 narzędzie
 BCP, 58
 Enterprise Manager, 39
 ETL, 58
 Extended Events, 57
 Query Editor, 39
 SQL Profiler, 54
 SQLCMD, 40, 52
 SSDT, 54
 natywne kompilowanie procedur, 122, 169
 nazywanie obiektów, 32, 333
 niestrukturyzowany xml, 334
 niewykorzystywane indeksy, 411
 NYSIIS, 97

O

obiekt
 builder, 501
 SqlCommand, 427, 438
 SqlContext.Pipe, 475
 SqlDataReader, 438
 obiekty
 bazy danych, 449
 pamięciowe, 153, 155
 Object Explorer, 40
 obliczanie
 mediany, 480, 483
 zasięgu statystycznego, 479

obsługa
 błędów, 494, 523, 527
 niepoprawna, 524
 poprawiona, 524
 TRY...CATCH, 526
 DMV, 121
 FILESTREAM, 281, 282
 FileTable, 286
 przestrzeni dyskowej, 545
 Unicode, 253
 UTF-16, 389
 XML, 321
 obszar, extent, 577
 ODBC, Open Database Connectivity, 142, 503
 odbiornik WCF Data Service, 518
 odkrywanie metadanych, 121
 odłączone zbiory danych, 423
 odpytywanie, 24
 elementów, 452
 o alokację obiektów, 408
 o blokowane zapytania, 407
 o najdroższe zapytania, 406
 o pozwolenia, 395
 tabel, 422
 TVF, 472
 uprawnień, 396
 zbioru kolumn rzadkich, 558
 odroczone wykonanie zapytania, 447
 ODS, Open Data Services, 459
 odszyfrowywanie, 206
 ograniczenia
 check, 27
 domen danych, 28
 funkcji UDF, 115
 szyfrowania asymetrycznego, 205
 tabel pamięciowych, 163
 okno
 Debugger Watch, 40
 Locals, 43
 Output, 43
 Quick Watch, 40, 43
 Watch, 43
 OLEDB, 510
 OLTP, 153
 opcja
 CALLED ON NULL INPUT, 93
 ELEMENTS XSINIL, 359
 ENCRYPTION, 93
 EXECUTE AS, 93
 SCHEMABINDING, 93

opcje

do zarządzania projektami, 48

edycji, 45

kursorów

DYNAMIC, 88

FAST_FORWARD, 88

FORWARDONLY, 88

INSENSITIVE, 88

KEYSET, 88

OPTIMISTIC, 88

READ ONLY, 88

SCROLL, 88

SCROLLLOCKS, 88

TYPEWARNING, 88

UPDATE OF, 88

parametrów flag OPENXML, 324

śledzenia zmian, 302

trybu FOR XML RAW, 328

tworzenia indeksu XML, 346

UDF, 93

wiersza poleceń SQLCMD, 610–613

zapytań, 517

zarządzania kodem źródłowym, 50

operacje

asynchroniczne, 503

DML, 178

pamięciowe, dodawanie, 157

grupy plików, 156

kontenera optymalizowanego, 157

tabeli optymalizowanej, 159

operator

?., 74

equals, 446

PIVOT, 75

UNPIVOT, 75

operatory

porównań ogólnych, 376

porównujące wartości, 374

zapytań LINQ, 444

optymalizacja

wydajności FTS, 307

zapytań, 566

orientacja, 278

osie, 369

OWA, open-world assumption, 65

oznaczenia osi, 369

P

pakiet SAFE, 470

pakiety CLR, 461

pamięciowe

hurtownie danych, 153

OLTP, 153

widoki systemowe, 405

pamięć

masowa, 543

podręczna, 143

parametry

tabelaryczne, 140

UDF, 92

parametryzacja dynamicznych zapytań, 541

parametryzowane zapytanie SQL, 425

partycje, 549

plan

wykonania zapytania, 48

zapytania

rzeczywisty, 567

szacowany, 567

plik

sqljdbc4.jar, 509

tsenu.xml, 314

z danymi FILESTREAM, 155

pliki

.dbml, 440

.ldf, 26, 543

.mdf, 543

.ndf, 543

.so, 506

bazy danych, 502

pobieranie

kanału RSS, 469

listy produktów, 132

metadanych, 395

statystyk SP, 151

podpisanie komunikatu certyfikatem, 207, 210

podstawienia znaków, 97

podzapytanie

InventoryDetails, 110

Order_Details, 110

polecenia SQLCMD, 615, 616

polecenie

apt-cache, 505

build_dm.sh, 506

ldd, 507

ln, 507

połączenie, 403
 SqlConnection, 426
 SqlConnection, 421
 ze źródłem danych, 419
 pomoc kontekstowa, 46
 poprawianie wydajności, 543
 porównywanie
 dat, 258
 węzłów, 378
 powiązania pomiędzy elementami, 451
 poziomy istotności błędów, 525
 półkula, 278
 prawo Moore'a, 154
 predykat, 24
 CONTAINS, 308
 FREETEXT, 306, 313
 XQuery, 374
 problem pakowania, 107
 procedura
 dbo.RebuildIndexes, 83
 sp_executesql, 540
 procedury składowane, SP, 30, 34, 92, 119, 317, 472
 natywnie kompilowane, 122, 169
 parametry tabelaryczne, 140
 rekurencja, 132
 tymczasowe, 142
 zarządzanie, 124
 proces sqlserver, 297
 programowanie
 asynchroniczne, 502
 defensywne, 35
 klienta .NET, 417
 projekt bazy danych, 463
 protokół TDS, 503
 przechodzenie przez zbiór wyników, 422
 przechwytywanie
 parametrów, 145, 147
 zdarzeń, 55
 przeglądanie BOM, 272
 przekształcenia XSL, 347
 przepływ
 danych, 59
 sterowania, 65
 przestrzenie nazw XQuery, 369
 przestrzeń nazw
 System.Collections, 471
 System.Data, 417, 464
 przestrzeń tempdb, 407
 przesunięcia, 265

przeszukiwane wyrażenie CASE, 73
 przetwarzanie
 liczb zespolonych, 489
 listy, 113, 114
 przeźroczyste szyfrowanie danych, 219
 przywracanie DMK, 202
 punkt
 kontrolny, 40
 wejścia, 34
 wyjścia, 34

Q

Query Editor, 39
 Quick Watch, 40

R

ranking, 235–237
 reguły kodowania fonetycznego, 97
 rekompilacja, 143, 148
 rekurencja
 w procedurach składowanych, 132
 w skalarnych UDF, 93
 wyzwalaczy, 183
 rekurencyjna skalarna UDF, 94
 rekurencyjne CTE, 227
 reprezentacja danych hierarchicznych, 267
 REST, Representational State Transfer, 511
 ROA, Resource Oriented Architecture, 511
 rozszerzenie bufora SSD, 153
 RPC, Remote Procedure Call, 510
 rzutowanie, 383

S

schemat obiektów, 123
 schematy
 baz danych, 26
 skrótów klawiszowych, 43
 sekwencja, 361
 semantyka statystyczna, 318
 sesja, 402
 XEvents, 574, 575
 skalarna funkcja użytkownika, 91
 skalarne UDF, 91
 skróty, 217
 klawiszowe, 40, 43
 skrypt sesji XEvents, 577

- słowo kluczowe
 - AS, 92
 - AUTO, 329
 - BEGIN, 65
 - CREATE FUNCTION, 108
 - DEFAULT, 92
 - END, 65
 - FETCH, 233
 - for, 384
 - let, 388
 - LOCAL, 88
 - OFFSET, 233
 - return, 384
 - RETURNS, 92
 - where, 387
 - WITH, 92
- SMK, service master key, 199
- SOA, Service Oriented Architecture, 497, 510
- SOAP, 511
- sól, salt, 216
- SP, stored procedures, 30, 119
- spójność, 26
- sprawdzanie
 - adresu e-mail, 492
 - przestrzeni, 551
 - punktów kontrolnych, 40
- SQL, Structured Query Language, 21
 - CLR, 30, 459
 - OS, 460
 - Profiler, 54
 - przechwytywanie zdarzeń, 55
 - Server 2014 Books Online, 60
 - Server Data Tools, 54
 - Server Integration Services, 58
 - Server Management Studio, 39
- SQLCMD, 52
 - opcje wiersza poleceń, 609
 - polecenia, 614
 - tryb interaktywny, 53
 - zmienne skryptowe, 613
- SQL-DMO, 393
- SRID, spatial reference identifier, 278
- SSD, Scalable Shared Database, 410
- SSDT, SQL Server Data Tools, 54
 - okno nowego projektu, 55
- SSIS, SQL Server Integration Services, 39, 58, 394, 431
- SSMS, SQL Server Management Studio, 39, 289
 - eksplorator obiektów, 50
 - graficzne plany wykonania zapytań, 48
 - opcje edycji, 45
 - pomoc kontekstowa, 46
 - zarządzanie projektami, 48
- stan bazy danych, 116
- standard IEEE 754, 257
- statystyki
 - oczekiwania, 413
 - operacji IO, 166
 - procedur składowanych, 143
- sterownik JDBC, 508
- sterta, 559
- stoplista, 313, 316
- stos warstw WCF, 511
- stosowanie wyzwalaczy DML, 175
- strefy czasowe, 261, 265
- strona Default.aspx, 518
- stronicowanie zapytania, 233
- struktura
 - B-tree, 560
 - filetable, 288
 - indeksu nieklastrowego, 562
 - Median, 481
- style, 31
- sumy bieżące, 129
- surogaty, 391
- system
 - diagnostyczny, 573
 - kontroli wersji, 40
- systemowa baza danych, 25
- szacowanie oszczędności, 554
- szyfrowanie, 199, 208
 - AES, 211
 - asymetryczne, 203, 205
 - bez kluczy, 217
 - DES, 211
 - DES-X, 211
 - DMK, 202
 - kluczem asymetrycznym, 209
 - przezroczyste danych, 219
 - RC4, 211
 - tekstu certyfikatem, 205
 - UDF, 93

Ś

- ścieżka lokalizacji, 364
- śledzenie
 - zdarzeń, 56
 - zmian
 - Automatycznie, 302
 - Nie śledź zmian, 302
 - Ręcznie, 302

T

- tabela, 27
 - deleted, 177
 - inserted, 177
- tabele
 - do logowania, 192
 - dyskowe, 164
 - korzystające z FILESTREAM, 284
 - krawędziowe, 325
 - liczb, 101
 - pamięciowe, 164
 - indeksy, 164
 - ograniczenia, 163
 - OLTP, 164
 - tworzenie, 159
 - przestawne, 75
 - dynamiczne, 76
 - z CASE, 75
 - z PIVOT, 76
 - rzadkie, 556
 - wąskie, 545
 - wirtualne, 177
 - z szerokimi wierszami, 546
- TCL, Transactional Control Language, 25
- TDE, transparent data encryption, 199
- TDS, Tabular Data Stream, 503
- technologie pamięciowe, 153
- testowanie
 - odroczonego wykonania zapytań, 448
 - surogatów UTF-16, 390
 - węzłów, 358, 360, 366
 - węzłów XPath, 357
 - wyzwalacza, 182, 187
 - wyzwalacza DDL, 194
- tezaurus, 313, 314
- transakcja, 26
- trendy sprzętowe, 154
- trwałość, 26
- tryb
 - Debug, 535
 - interaktywny zapytania, 53
 - SQLCMD, 40
- T-SQL, 21
- TUC, Temps Universel Coordonné, 261
- TVF, inline table-valued function, 91, 104
 - wbudowane, 112
 - wielowyróżnieniowe, 104
- TVF CLR, 472
- tworzenie
 - aplikacji sieciowej, 513
 - atributów XML, 355
 - certyfikatu serwera, 220
 - CLR UDF, 468
 - DMK, 205
 - filetable, 287
 - funkcji agregującej, 476
 - grup plików FILESTREAM, 283
 - indeksu pełnotekstowego, 298, 300, 305, 314
 - indeksu XML, 346
 - interfejsu, 127
 - katalogów pełnotekstowych, 298
 - klucza asymetrycznego, 208, 219
 - klucza do szyfrowania bazy danych, 220
 - klucza głównego, 201
 - klucza symetrycznego, 211
 - komunikatu, 207
 - kopii zapasowej, 202
 - kopii zapasowej certyfikatu, 204
 - listy, 106, 358
 - natywnie kompilowanej procedury
 - składowanej, 170
 - odbiornika WCF Data Service, 518
 - procedury składowanej, 122, 125
 - skrótów, 217
 - stoplisty, 316
 - tabel rzadkich, 556
 - tabeli do logowania, 192
 - tabeli liczb, 105
 - tabeli pamięciowej, 159, 160
 - tabeli tymczasowej, 541
 - typu tabelarycznego, 140
 - usługi danych, 514
 - użytkowników, 395
 - wąskiej tabeli, 545
 - WCF Data Service, 512
 - wyzwalacza CLR, 494
 - zaawansowanych UDA, 479
- tymczasowe
 - klucze symetryczne, 212
 - SP, 142
- typ danych
 - date, 258
 - datetime, 258
 - datetime2, 258
 - datetimeoffset, 258
 - hierarchiid, 267
 - smalldatetime, 258

time, 258
 uniqueidentifier, 265
 xml, 334, 361
 typy
 celów XEvents, 576
 danych
 dla daty i czasu, 258
 max, 254
 podstawowe, 253
 przestrzenne, 274
 tabelaryczne, 140
 XQuery, 589
 LOB, 254
 protokołu, 516
 użytkownika, 484
 zdarzeń DDL, 191

U

Ubuntu Server, 504
 UCS, Universal Character Set, 389
 UDA, User Defined Aggregates, 476
 UDF, user-defined functions, 30, 91, 465
 funkcje niedeterministyczne, 116
 kod proceduralny, 96
 ograniczenia funkcji, 115
 opcje, 93
 parametry, 92
 rekurencja, 93
 udostępnianie danych wyzwalaczom, 183
 Unicode, 253
 uprawnienia, 470
 dla elementów i operacji, 516
 uprawnienie EXTERNAL_ACCESS, 471
 URI, uniform resource identifier, 511
 usługa, 200
 danych, 497
 danych WCF, 510, 515, 516
 filtra, 297
 MSFTESQL, 307
 SSIS, 58
 ustrukturyzowany xml, 335
 usuwanie
 problemów z wydajnością, 570
 procedury składowanej, 125
 wyzwalacza DDL, 194
 UTC, Universal Time Coordination, 261
 UTF-16, 389
 uwierzytelnianie, 216

użycie

COALESCE, 81
 CUME_DIST, 246
 DENSE_RANK, 238
 ExecuteScalar, 428
 FETCH, 234
 FileTableRootPath, 291
 FIRST_VALUE, 251
 FOR XML, 347
 FnCommaSplit, 115
 GetFileNamespacePath, 291
 GetPathLocator, 293
 GML, 277
 hierarchyid, 269, 292
 klauzuli OVER, 241, 244
 kolumny parent_path_locator, 292
 kontekstu obiektu, 452
 LAG, 248
 LAST_VALUE, 251
 LEAD, 250
 listy rozdzielanej przecinkami, 115
 metody
 exist, 340
 modify, 341
 nodes, 340
 query, 338
 value, 339
 NTILE, 240
 NYSIIS, 101
 ObjectSet EF, 454
 OFFSET, 234
 opcji ENCRYPTION, 93
 osi, 370
 PERCENT_RANK, 246
 PERCENTILE_CONT, 247
 PERCENTILE_DISC, 247
 ROW_NUMBER, 232
 SET DATEFORMAT, 263
 sys.dm_fts_parser, 317
 tabeli tymczasowej, 223
 TRY_CONVERT, 529
 WITH XMLNAMESPACES, 360
 wyrażenia THROW, 530
 wyrażen CASE, 75
 zapytania nazwanego, 292
 zmiennych skryptowych, 53

V

Visual Studio 2010, 20

W

- warstwy abstrakcji Entity Framework, 452
- wartości numeryczne, 256
- wartość NULL, 24, 63, 73, 359, 489
- wbudowane
 - funkcje zwracające tabelę, 112
 - TVF, 112
- WCF, Windows Communication Foundation, 497, 510
- WCF Data Services, 512
- wejście, 33
- weryfikacja
 - adresu e-mail, 469
 - działania bazy danych, 318
- wewnętrzny dokument XML, 324
- węzeł Specifications, 368
- węzły niższego poziomu, 273
- widok, 28, 190
 - HumanResources.vEmployee, 31
- widoki
 - indeksowane, 28
 - INFORMATION_SCHEMA, 413–416
 - katalogowe, 317, 393
 - systemowe, 405
- wiele zapytań nazwanych, 225
- wielowyrażeniowe
 - funkcje zwracające tabelę, 104
 - TVF, 104
- wieloznaczniki, 356
- Wieża Hanoi, 133–139
- wirtualne tabele, 177
- WKT, Well-Known Text, 276
- właściwości
 - projektu, 462
 - systemu bazodanowego, 26
 - tabeli, 161
- włączanie
 - kompresji, 551
 - semantyki statystycznej, 319
 - wyzwalaczy, 174
- WSE, Web Services Enhancement, 511
- wstawianie
 - fragmentu kodu, 43
 - podkatalogu, 289
- wstrzykiwanie SQL, 427, 537
- wybór unikalnego indeksu, 301
- wydajność, 427, 543, 570
 - zapytania, 571
- wyjątek, 474, 526
- wyjście, 33
- wyłączanie wyzwalaczy, 174
- wrażenia, 23
 - arytmetyczne, 379
 - FLWOR, 384
 - nazwane, CTE, 223
 - warunkowe, 379
- wyrażenie
 - ALTER ASYMMETRIC, 208
 - ALTER INDEX, 398
 - ALTER PROCEDURE, 124
 - BACKUP CERTIFICATE, 204
 - BREAK, 68
 - CASE, 72, 73
 - przeszukiwane, 73
 - CHOOSE, 80
 - CONTINUE, 68
 - CREATE ASSEMBLY, 461, 465, 470
 - CREATE ASYMMETRIC KEY, 207
 - CREATE CERTIFICATE, 203
 - CREATE FULLTEXT INDEX, 305
 - CREATE MASTER KEY, 202
 - CREATE SYMMETRIC KEY, 211
 - CREATE TABLE, 194
 - CREATE TRIGGER, 196
 - DBCC DBREINDEX, 86
 - DECLARE, 66
 - DROP ASYMMETRIC KEY, 208
 - EXECUTE, 536
 - FLWOR, 344, 388
 - GOTO, 69
 - IF ... ELSE, 66
 - IIF, 79
 - ISNULL, 124
 - LANGUAGE, 124
 - PRINT, 530
 - RAISERROR, 525
 - regularne, 466
 - RESTORE SERVICE MASTER KEY, 201
 - RETURN, 71, 92
 - ROLLBACK TRANSACTION, 186
 - SELECT *, 36
 - THROW, 529
 - WAITFOR, 70
 - WHILE, 68
- wyszukiwanie
 - pełnotekstowe, FTS, 297
 - zakładek, 563
 - zakresu, 167, 169

wywołanie
 strony, 517
 usługi danych, 520
 wyzwacz, 28, 173, 491
 logowania, 195, 196
 standaryzujący wymiary, 185
 zapisujący dane, 181
 wyzwacze, triggers, 27
 DDL, 191
 DML, 173, 197
 logowania, 195
 na filetable, 294
 na widokach, 188
 rekurencyjne, 183
 zagnieżdżone, 183
 wzorzec wyrażenia regularnego, 468

X

XLST, 347
 XML, 321
 indeksy, 343
 niestrukturyzowany, 334
 ustrukturyzowany, 335
 XML DML, 353
 XPath, 353, 359
 XQuery, 361
 dynamiczne generowanie XML, 371
 dzielenie liczb całkowitych, 380
 elementy, 363
 format daty, 377
 funkcje, 380
 iloczyn kartezjański, 385
 komentarze, 373
 konstruktory, 383
 obsługa UTF-16, 389
 operatory porównań ogólnych, 376
 operatory porównujące wartości, 374
 porównania węzłów, 378
 predykaty, 374
 przestrzenie nazw, 367
 rzutowanie, 383
 sekwencje, 361
 testy węzłów, 366
 typy danych, 373, 589
 wyrażenia, 361
 wyrażenia arytmetyczne, 379
 wyrażenia FLWOR, 384
 wyrażenia warunkowe, 379
 XSL, 347

Z, Ż

zabezpieczenie
 bazy danych, 220
 HSM, 219
 zadania oczekujące, 572
 zapisywanie modyfikacji danych, 178
 zapytania
 niezwracające wyników, 428
 parametryzowane, 424
 pełnotekstowe, 305
 skalarne, 428
 XML, 428
 z LINQ to SQL, 442
 zapytanie
 CONTAINS, 308–310
 CONTAINS z FORMSOF, 308, 309
 CONTAINSTABLE, 313
 DML, 456
 FOR XML, 348
 FOR XML AUTO, 330
 FOR XML EXPLICIT, 331
 FOR XML PATH, 333
 FOR XML RAW, 328
 FREETEXT, 306, 307, 315, 316
 OPENXML, 322
 zarządzanie
 kluczami rozszerzone, 218
 kodem źródłowym, 50
 procedurami składowanymi, 124
 projektami, 48
 zasięg
 dynamicznego SQL, 541
 statystyczny, 476
 zasoby serwera, 409
 zastosowanie dysków SSD, 158
 zbiory
 danych odłączone, 423
 kolumn rzadkich, 557
 wyników, 120, 436, 438
 zdarzenia DDL, 191
 zintegrowane CLR, 459
 złączenia nietozsamościowe, 446
 złączenie wewnętrzne, 566
 zmienne, 37
 nawigacyjne, 453
 skryptowe, 53, 613
 skryptowe SQLCMD, 614
 znaki, 253
 zwracanie listy, 130
 źródło danych, 512

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ 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 WYDAWNICZA

 **Helion SA**

T-SQL dla zaawansowanych. Przewodnik programisty

W wielu środowiskach IT serwer bazodanowy jest szczególnie ważnym elementem infrastruktury. Microsoft SQL Server to jedno z częściej spotykanych rozwiązań. Aby w pełni wykorzystać jego możliwości, trzeba dogłębnie poznać język Transact-SQL — potężne narzędzie, które wyewoluowało z języka zapytań SQL i wciąż jest rozwijane przez firmę Microsoft.

Niniejsza książka, przeznaczona dla programistów T-SQL, jest uaktualnionym wydaniem świetnego przewodnika po SQL Server. Omówiono w niej zaawansowane mechanizmy dostępne w T-SQL, w tym nowości, takie jak mechanizmy pamięciowe będące częścią SQL Server 2014. Istotne kwestie są przystępnie wyjaśnione za pomocą praktycznych przykładów i obszernych fragmentów kodu źródłowego. Autorzy położyli nacisk na demonstrację opcji T-SQL, ich możliwe zastosowania, a sam podręcznik zorganizowali w sposób umożliwiający szybkie wyszukanie potrzebnych informacji.

Najważniejsze zagadnienia omówione w książce:

- podstawy T-SQL i przegląd dobrych praktyk programowania w tym języku
- narzędzia dołączone do SQL Server 2014 (w tym SSMS, SQLCMD, SSDT oraz SQL Profiler)
- procedury składowane wykonywane po stronie serwera oraz tabele OLTP
- obsługa XML, XQuery oraz XPath w SQL Server 2014
- szyfrowanie baz danych, usługi ADO.NET oraz usługi IIS
- obsługa błędów, testowanie i zabezpieczenie kodu SQL

Miguel Cebollero — od ponad 16 lat zajmuje się SQL Server i innymi systemami bazodanowymi, włączając w to projektowanie, programowanie i administrację. Często wypowiada się na różnego rodzaju konferencjach związanych z bazami danych.

Jay Natarajan — od ponad 15 lat projektuje i implementuje rozwiązania oparte na SQL Server. Ma na koncie wiele skomplikowanych rozwiązań dla dużych klientów. Dołączyła do Microsoft Consulting Services w 2008 roku.

Michael Coles — ma za sobą 10-letnie doświadczenie w przy projektowaniu i administrowaniu bazami danych SQL Server. Jest autorem licznych artykułów na temat SQL Server, szczególnie o specjalnych zastosowaniach T-SQL.

Helion

44767 numer katalogowy

księgarnia internetowa

<http://helion.pl>

zamówienia telefoniczne



0 801 339900



0 601 339900

Informatyka w najlepszym wydaniu

Sprawdź najnowsze promocje:

- <http://helion.pl/promocje>
- Książki najchętniej czytane:
- <http://helion.pl/bestsellery>
- Zamów informacje o nowościach:
- <http://helion.pl/novosci>

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
<http://helion.pl>

Apress

ISBN 978-83-283-2247-9



9 788328 322479

cena: 99,00 zł

śledź po WIĘCEJ



KOD KORZYŚCI