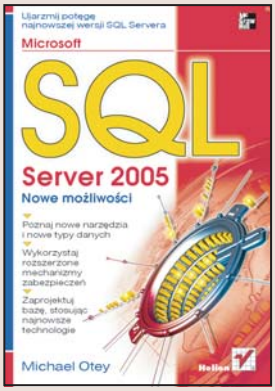
Microsoft SQL Server 2005. Nowe możliwości



Autor: Michael Otey
Tłumaczenie: Piotr Pilch
ISBN: 83-7361-982-8
Tytuł oryginału: Microsoft SQL Server 2005 New Features
Format: B5, stron: 264

Microsoft SQL Server to system zarządzania bazami danych wykorzystywany wszędzie tam, gdzie niezbędna jest wysoka wydajność i bezpieczeństwo. W oparciu o ten system budowane są hurtownie danych, aplikacje operujące na milionach rekordów i przetwarzające ogromne ilości informacji. SQL Server stosowany jest również jako zaplecze bazodanowe dla witryn WWW i aplikacji mobilnych. Każda kolejna wersja Microsoft SQL Server wyposażona jest w nowe narzędzia oraz udoskonalone mechanizmy zarządzania danymi i udostępnia projektantom oraz administratorom baz danych coraz większe możliwości.

„Microsoft SQL server 2005. Nowe możliwości" to książka opisująca najnowszą wersję SQL Server, oznaczoną symbolem 2005. Prezentuje nowe funkcje związane z zarządzaniem bazami danych, projektowaniem tabel i baz oraz generowaniem raportów i analizowaniem danych. Czytając ją, poznasz narzędzie SQL Workbench, możliwości tworzenia procedur składowanych, wyzwalaczy i funkcji za pomocą dowolnego języka programowania platformy .NET oraz zasady korzystania z funkcji SQL Server Reporting Services. Dowiesz się wszystkiego o najważniejszych nowych funkcjach serwera SQL Server 2005 oraz zrozumiesz, czym różni się on od swoich poprzedników.

- Obsługa nowych urządzeń
- Integracja z platformą .NET
- Stosowanie udoskonalonych mechanizmów zabezpieczeń i praw dostępu
- Korzystanie z narzędzi SQL Server Management Studio oraz Business Intelligence Development Studio
- Strojenie bazy za pomocą Database Tuning Advisor
- Archiwizacja i odtwarzanie danych
- Projektowanie baz z wykorzystaniem nowych narzędzi
- Stosowanie języka XML
- Generowanie raportów za pomocą usługi Reporting Services
- Analiza danych i tworzenie hurtowni danych

Jeśli chcesz wykorzystać wszystkie nowe funkcje SQL Servera 2005, koniecznie przeczytaj tę książkę

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Rozdział 1.

Funkcje związane z architekturą bazy danych i mechanizmem przechowywania danych

Opierając się w zasadzie na tej samej architekturze, którą zdefiniowano w serwerze SQL Server 7, Microsoft kontynuował wprowadzanie w mechanizmie serwera ewolucyjnych udoskonaleń. Zgodnie z prawem Moore'a liczba tranzystorów będzie podwajana co 18 miesięcy przy jednoczesnym ciągłym obniżaniu ceny urządzeń. Jednak choć urządzenia tanieją, wzrasta koszt związany z zatrudnieniem specjalistów. W serwerze SQL Server 2005 Microsoft dokonał ulepszeń, które umożliwiają zastosowanie bardzo wydajnych urządzeń najnowszej generacji i skalowanie ich na potrzeby największych przedsiębiorstw. Jednocześnie firma dodała funkcje, które sprawiają, że serwer SQL Server jest prostszy w zarządzaniu. Serwer oferuje też nowe możliwości pozwalające na jego lepsze wykorzystanie w organizacji. W niniejszym rozdziale zaprezentuję niektóre z najważniejszych udoskonaleń dokonanych przez Microsoft w architekturze serwera SQL Server 2005. Chcę w ten sposób ułatwić Ci zorientowanie się, w jaki sposób innowacje mogą zwiększyć wydajność i niezawodność środowiska bazodanowego.

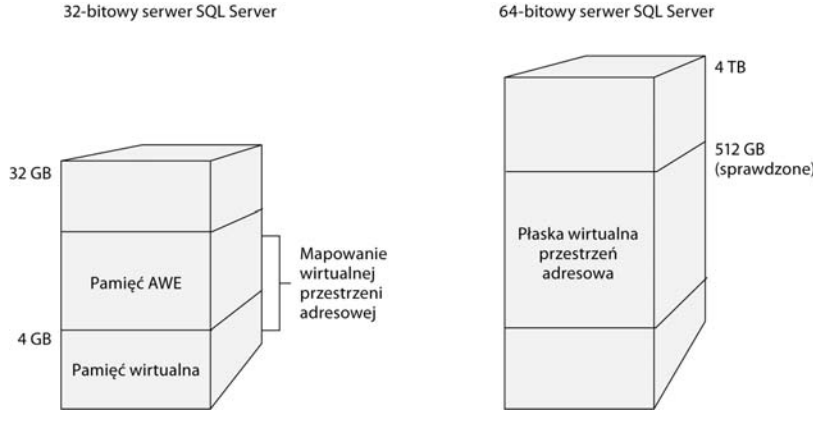
Obsługa nowych urządzeń

Jedną z kluczowych nowych funkcji umożliwiających serwerowi bazodanowemu SQL Server 2005 uzyskanie wydajności porównywalnej z osiągami największych baz danych opartych na systemie UNIX jest udoskonalona obsługa urządzeń. Pamięć zawsze była jednym z najbardziej krytycznych czynników wpływających na wydajność baz danych. Duże uniksowe bazy danych (bez klastrów), które znajdowały się na szczycie

rankingów wydajności ogólnego przetwarzania transakcyjnego TPC-C, osiągały to dzięki korzystaniu z 64-bitowej platformy sprzętowej i systemu operacyjnego. Gdy serwer SQL Server mógł korzystać z porównywalnej 64-bitowej platformy złożonej z procesora Intel Itanium 2 i systemu Windows Server 2003, od razu uplasował się na szczycie rankingów TPC-C. Serwer SQL Server 2005 korzysta z 64-bitowości systemu Windows Server 2003 i dodatkowo obsługuje architekturę NUMA (ang. *Non-Uniform Memory Architecture*). Korzystając z platformy Windows Server 2003, serwer SQL Server 2005 jest zgodny zarówno z 64-bitową architekturą Itanium IA-64, jak i AMD x64.

Macierzysta obsługa 64-bitowości

Gdy serwer SQL Server 2005 zostanie uruchomiony pod systemem Windows Server 2003 zgodnym z 64-bitowymi procesorami Intel Itanium, będzie obsługiwał układy Itanium i Itanium 2. Po uruchomieniu serwera SQL Server 2005 pod systemem Windows Server 2003 for 64-bit Extended Systems, będzie współpracował z 64-bitowymi układami Opteron i Athlon 64 firmy AMD, a także z procesorami Intel Xeon z technologią EMT64 (ang. *Extended Memory 64 Technology*). Wszystkie podstawowe usługi serwera SQL Server 2005 w pełni obsługują zarówno 32-, jak i 64-bitowe platformy sprzętowe. Do usług tych zaliczają się m.in. SQL Server Engine, Analysis Services, SQL Agent i Reporting Services. Prawdziwą korzyścią wynikającą z przeniesienia bazy danych na platformę 64-bitową nie jest większa moc obliczeniowa, a raczej znacznie zwiększona adresowalna pamięć. Na rysunku 1.1 przedstawiono porównanie maksymalnej adresowalnej pamięci 32-bitowego serwera SQL Server i 64-bitowej wersji serwera SQL Server 2005.



Rysunek 1.1. *Porównanie 32- i 64-bitowego adresowania pamięci*

Architektura 32-bitowa ograniczona jest maksymalnie do 4 GB adresowalnej pamięci. W systemie Windows ograniczenie to jest równomiernie podzielone między system operacyjny i aplikacje. Inaczej mówiąc, 2 GB pamięci zarezerwowane jest dla systemu Windows, natomiast taka sama pojemność dla aplikacji. Korzystając z technologii AWE (ang. *Address Windowing Extensions*) dostępnej w 32-bitowej wersji systemu Windows, 32-bitowa wersja serwera SQL Server jest w stanie zaadresować pamięć RAM o maksymalnej pojemności wynoszącej 32 GB. Choć jest to znaczny wzrost, nadal dodatkowy obszar pamięci był przeznaczany na stronicowanie umożliwiające

uzyskanie dostępu do odpowiednich stron pamięci. Wprowadzenie obsługi 64-bitowej architektury niemal zupełnie wyeliminowało ograniczenia pamięci. Uzyskano to przez zwiększenie maksymalnej pojemności adresowalnej pamięci do 32 TB. Aktualnie żaden system produkcyjny nie posiada pamięci RAM, której pojemność byłaby choćby zbliżona do tak dużej pojemności. Maksymalna pojemność pamięci RAM, w przypadku której testowano serwer SQL Server 2005, wynosiła 512 GB. Po pojawieniu się kolejnej wersji dodatku Service Pack dla systemu Windows Server 2003 maksymalna obsługiwana pojemność pamięci ma się zwiększyć do 1 TB.

Obsługa architektury NUMA

Kolejną nową funkcją wpływającą na system operacyjny Windows Server 2003 jest architektura NUMA (ang. *Non-Uniform Memory Architecture*). NUMA jest architekturą systemową wykorzystywaną przez niektórych producentów systemów, takich jak IBM i Unisys. Architektura NUMA zarządza w systemach wieloprocesorowych obciążeniem procesora i pamięci bardziej wydajnie niż zwykła architektura SMP. W standardowych systemach SMP wraz ze wzrostem szybkości i liczby procesorów rywalizacja między nimi o dostęp do pamięci powoduje konkurowanie o magistralę. Doprowadza to do opóźnień w przetwarzaniu i blokuje możliwość skalowania systemu. W efekcie systemy SMP nie skalują się liniowo wraz ze wzrostem liczby procesorów. Architektura NUMA została zaprojektowana w celu rozwiązania problemu występującego w systemach SMP, polegającego na tym, że procesory potrafią uzyskać dostęp do danych zawartych w pamięci RAM szybciej niż pamięć i magistrala systemowa są w stanie je dostarczyć. Architektura NUMA grupuje procesory i pamięć RAM w złożone z kilku procesorów lokalne zestawy. Są one połączone ze sobą za pośrednictwem zewnętrznej magistrali przesyłającej dane wymieniane przez zestawy. Poprzez zastosowanie zestawów rozwiązuje się problem z rywalizacją procesorów o dostęp do pamięci, ograniczając liczbę układów, które ze sobą konkurują. W celu maksymalnego wykorzystania zalet architektury NUMA zarówno system operacyjny, jak i aplikacje muszą być tak zaprojektowane, aby zminimalizować ilość danych przesyłanych między zestawami i zmaksymalizować czas dostępu do pamięci dla poszczególnych zestawów. Jeśli system operacyjny i aplikacje zostały odpowiednio zaprojektowane, architektura NUMA umożliwi prawie liniowe skalowanie, gdy będą dodawane kolejne procesory. W celu uzyskania jak największej bliskości w jednym zestawie wątków i pamięci, z której one korzystają, system Windows Server 2003 i serwer SQL Server 2005 spożytkowują zalety architektury NUMA.

Obsługa hiperwątkowości

Częściowo dzięki wprowadzeniu przez Microsoft w systemie Windows Server 2003 obsługi hiperwątkowości serwer SQL Server 2005 też może skorzystać z tej technologii. Hiperwątkowość jest technologią stosowaną w układach Intela, która dla każdego fizycznego procesora komputera tworzy dwa logiczne procesory. Każdy logiczny procesor jest w stanie w tym samym czasie przetwarzać niezależne wątki. Celem hiperwątkowości jest zaoferowanie lepszego wykorzystywania zasobów przez aplikacje wielowątkowe lub wiele programów uruchomionych na jednym komputerze. Choć

hiperwątkowość stwarza potencjał zwiększenia przepustowości serwera, logiczne procesory rywalizują o zasoby systemowe, takie jak dane znajdujące się w pamięci podręcznej fizycznego procesora. Rywalizowanie o zasoby uniemożliwia procesorowi z hiperwątkowością pracę z wydajnością, taką jak porównywalny system wyposażony w dwa fizyczne układy.

Obsługa hiperwątkowości systemu Windows Server 2003 wykorzystywana jest przez serwer SQL Server 2005 na dwa podstawowe sposoby. W przeciwieństwie do systemu Windows 2000, w przypadku licencji system Windows Server 2003 uwzględnia jedynie liczbę fizycznych procesorów. A zatem pojedynczy układ z hiperwątkowością jest licencjonowany jako jeden procesor, a nie jak dwa, co miałoby miejsce w systemie Windows 2000. To pierwsza korzyść. Kolejna jest taka, że w przypadku komputera z procesorem obsługującym hiperwątkowość system Windows Server 2003 oferuje udoskonalone szeregowanie wątków. Wymienione rozszerzenia zwiększają wydajność wielowątkowych aplikacji, takich jak serwer SQL Server.

Mechanizm serwera SQL Server

W pierwszym podrozdziale tego rozdziału omówiono ogólnie obsługę nowych urządzeń przez serwer SQL Server 2005, natomiast niniejszy zostanie poświęcony kilku najważniejszym udoskonaleniom, jakie Microsoft wprowadził w mechanizmie serwera SQL Server.

Integracja z platformą .NET Framework

Najbardziej istotnym rozszerzeniem mechanizmu serwera SQL Server 2005 jest integracja z platformą Microsoft .NET Framework. Dzięki niej zwiększono możliwości serwera SQL Server 2005 w zakresie tworzenia procedur przechowywanych, funkcji użytkownika, wyzwalaczy, agregatów i typów użytkownika przy użyciu dowolnego języka .NET, takiego jak VB.NET, C# lub J#. Integracja platformy .NET CLR z serwerem SQL Server 2005 jest nie tylko powierzchowna, ponieważ mechanizm serwera SQL Server obsługuje aktywne środowisko CLR. Więcej informacji na temat integracji platformy .NET Framework z serwerem SQL Server 2005 zawarto w rozdziale 4.

Rozszerzona obsługa wielu instancji

Kolejnym ważnym rozszerzeniem wersji Enterprise Edition serwera SQL Server 2005 jest obsługa maksymalnie 50 instancji. Dla porównania serwer SQL Server 2000 obsługiwał maksymalnie 16 instancji. Rozszerzenie to jest szczególnie istotne dla firm zajmujących się hostingiem, które w ramach swoich usług internetowych dzierżawią wiele instancji serwera SQL Server.

Nowe typy danych

Serwer SQL Server 2005 obsługuje też kilka nowych typów danych. Choć integracja platformy .NET Framework umożliwia korzystanie z typów danych użytkownika, serwer SQL Server 2005 udostępnia również kilka innych nowych własnych typów danych. Są to takie typy, jak `varbinary(max)` i `XML`. Typ danych `varbinary(max)` oferuje nową metodę używania w serwerze SQL Server obiektów LOB. W przeciwieństwie do starszych typów danych `image` i `text`, nowy typ `varbinary(max)` może pełnić funkcję zmiennej. Ponadto typ ten może być traktowany przez kod źródłowy jak mniejsze typy danych pozwalające na stosowanie prostszych i bardziej spójnych wariantów.

Nowy typ danych `XML` oparty na typie `varbinary(max)` umożliwia przechowywanie dokumentów XML w bazie danych. Jednak w przeciwieństwie do typu danych `varbinary(max)`, który w zasadzie jest uniwersalny, nowy typ `XML` stworzono z myślą o danych XML. W związku z tym typ ten obsługuje weryfikację dokumentów XML wykorzystującą schemat. Więcej informacji na temat nowych typów danych oferowanych przez serwer SQL Server 2005 zawarto w rozdziale 7.

Technologie Database Snapshot i Database Mirroring

Umożliwiając serwerowi SQL Server 2005 natychmiastowe udostępnienie rezerwowej bazy danych, technologia Database Mirroring chroni przed skutkami awarii podstawowej bazy danych. Database Mirroring jest technologią zwiększającą dostępność bazy danych, która współpracuje z wszystkimi standardowymi urządzeniami obsługiwanymi przez serwer SQL Server. Dzięki niej pomiędzy podstawowym i duplikującym serwerem nie trzeba stosować żadnego wspólnego urządzenia masowego. Technologia nie narzuca też ograniczeń dotyczących odległości serwerów. Działanie technologii polega na przesyłaniu dzienników transakcji między podstawowym i duplikującym serwerem. Jest to zatem nowe rozwiązanie pozwalające na przekazywanie dzienników w czasie rzeczywistym. Technologia Database Mirroring może być użyta dla jednej lub wielu baz danych.

Technologia Database Snapshot umożliwia wykonanie w określonej chwili migawki bazy danych tylko do odczytu. Technologia najlepiej nadaje się do tworzenia kopii bazy danych na potrzeby raportowania lub kopii zapasowych bazy, których można użyć do odtworzenia do poprzedniego stanu produkcyjnej bazy danych. W celu utworzenia serwera raportującego, korzystającego z danych znajdujących się na duplikującym serwerze, technologie Database Snapshot i Database Mirroring mogą być ze sobą połączone. Dostęp do danych zawartych na serwerze duplikującym nie może zostać uzyskany bezpośrednio, ponieważ zawsze znajduje się on w trybie odzyskiwania. Jednak dla duplikowanej bazy danych można utworzyć migawkę, a następnie na potrzeby raportowania korzystać z widoku bazy danych. Więcej informacji na temat widoków bazy danych i duplikowania zawarto w rozdziale 3.

Obsługa żądań HTTP

Jednym z innych istotnych rozszerzeń mechanizmu serwera SQL Server jest obsługa żądań HTTP. Możliwość przetwarzania przez serwer SQL Server przychodzących żądań HTTP pozwala mu zaoferować wykonywanie instrukcji SQL i wywoływanie procedur przechowywanych za pośrednictwem protokołu SOAP. Oznacza to, że serwer SQL Server 2005 jest w stanie przetwarzać przychodzące żądania HTTP bez dostępu do serwera IIS lub innego serwera HTTP. Będąca nowością obsługa żądań HTTP poszerza możliwości serwera SQL Server o nasłuch HTTP. Uwzględniona jest w tym obsługa punktów końcowych HTTP identyfikujących adres URL, port i żądania, które mają być przetworzone. Serwer SQL Server może też publikować dla punktów końcowych HTTP usługi internetowe przy użyciu języka WSDL (ang. *Web Services Description Language*). Obsługa żądań HTTP przez serwer SQL Server jest zgodna z takimi standardami, jak SOAP 1.0, SOAP 1.2, WSDL 1.1, a także z protokołami uwierzytelniania systemu Windows i serwera SQL Server oraz protokołem SSL. Aby obsługa żądań HTTP była bardziej zgodna z oprogramowaniem warstwy pośredniej, procedury przechowywane mogą zwracać zestawy wynikowe jako obiekt ADO.NET `DataSet`.

Zdarzenia serwerowe i wyzwalacze DDL

Nowe zdarzenia i wyzwalacze DDL serwera SQL Server 2005 umożliwiają kodowanie działań będących odpowiedzią na zmiany zachodzące w systemie. Choć obie te funkcje odgrywają podobną rolę, stosowane są w dość odmienny sposób. Wyzwalacze DDL, podobnie jak standardowe wyzwalacze DML, są synchronicznymi zdarzeniami powodującymi wykonanie procedur przechowywanych. Więcej na temat wyzwalaczy DDL zawarto w rozdziale 4.

Z kolei zdarzenia serwerowe są asynchroniczne. W modelu zdarzenia serwerowego serwer przesyła zdarzenie do usługi SQL Broker Service, po czym odbiorca może je niezależnie pobrać. Samo zdarzenie ma postać danych XML. Nie jest możliwe cofnięcie zdarzenia, które może zostać zignorowane, gdy nie pobierze go odbiorca. Po wystąpieniu zdarzenia serwerowego generowane jest zdarzenie systemowe, które może powiadomić Cię o zdarzeniu serwerowym. Opcjonalnie przez zdarzenie systemowe może zostać wykonany kod. W poniższym przykładzie zaprezentowano składnię kodu definiującego powiadomienie o zdarzeniu.

```
CREATE EVENT NOTIFICATION MyDDLEvents
ON SERVER FOR DDL_STATEMENTS TO SERVICE MyDDL_log
```

W przykładzie tworzone jest nowe zdarzenie, nadawana powiadomieniu o zdarzeniu nazwa `MyDDLEvents` i kojarzone zdarzenie z instrukcją języka DDL. Klauzula `TO SERVICE` określa, że usługa SQL Broker Service o nazwie `MyDLL_log` będzie odbiorcą zdarzeń. Więcej na temat usługi można znaleźć w rozdziale 6.

Rozszerzenia dotyczące plików danych bazy

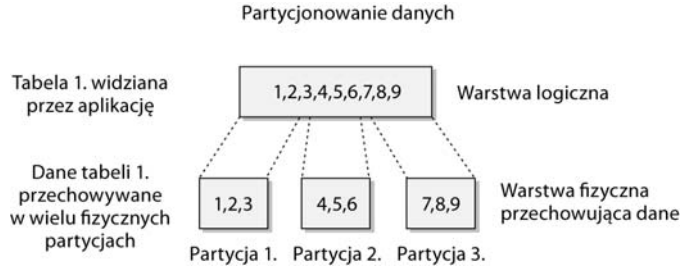
Obecnie serwer SQL Server 2005 umożliwia przy użyciu instrukcji `ALTER DATABASE` zmianę ścieżki plików danych i plików dzienników powiązanych z bazą danych. Serwer SQL Server 2000 oferował możliwość przeniesienia plików bazy danych `tempdb`. Jednak nie było to już możliwe w przypadku innych baz danych. Jak można oczekiwać, serwer SQL Server 2005 pozwala na przenoszenie plików tylko w trybie offline. Poniższy przykład ilustruje nowe elementy składni instrukcji `ALTER DATABASE`.

```
ALTER DATABASE <nazwa_bazy_danych>
  MODIFY FILE(name=<'nazwa_pliku_danych'>, filename=<'nowa_ścieżka'>)
```

Partycjonowanie danych

Kolejnym nowym rozszerzeniem, które pojawiło się w serwerze SQL Server 2005, jest możliwość wykonywania partycjonowania danych. *Partycjonowanie danych* pozwala na rozbicie pojedynczego obiektu bazodanowego, takiego jak tabela lub indeks, na wiele części. Nowa funkcja partycjonowania danych ułatwia zarządzanie bardzo dużymi tabelami i indeksami. Partycjonowanie jest niezauważalne dla aplikacji, które „widzą" jedynie sam obiekt bazodanowy i nie dysponują informacją, że przechowywany jest w wielu częściach przez serwer SQL Server. Partycje mogą być tworzone i usuwane bez wpływu na dostępność obiektu bazodanowego. Zasadniczo partycjonowanie umożliwia podzielenie danych przechowywanego obiektu na wiele obiektów podrzędnych przy jednoczesnym udostępnianiu aplikacji jednolitego widoku obiektu i wszystkich jego partycji. Na rysunku 1.2 zaprezentowano ogólny schemat partycjonowania.

Rysunek 1.2. *Partycjonowanie danych*



Serwer SQL Server 2005 oferuje partycjonowanie danych w przypadku tabel, indeksów i widoków indeksowanych. Podstawową jednostką stosowaną podczas partycjonowania jest wiersz. Partycje mogą być tworzone na podstawie wartości znajdujących się w polach wiersza. Określa się to mianem *partycjonowania poziomego*. Przykładowo, tabela może być partycjonowana według daty, ze zdefiniowaną inną partycją dla każdego roku. Partycjonowanie według daty pozwala na wykonanie przetwarzania typu „przesuwne okna daty", w przypadku którego możliwe jest usunięcia partycji zawierającej dane z zeszłego roku bez wpływu na dostęp do danych znajdujących się w partycji powiązanej z bieżącym rokiem.

Partycjonowanie danych zapewnia kilka korzyści istotnych w przypadku bardzo dużych baz danych. Pozwalając na archiwizowanie tylko wybranych partycji, partycjonowanie może ułatwić zarządzanie danymi. Przykładowo, w przypadku dużej tabeli partycjonowanej według daty można zarchiwizować tylko dane z bieżącego roku, pomijając dane z partycji przechowującej dane z poprzedniego roku. Kolejną korzyścią jest to, że w systemach wieloprocesorowych procesor można przeznaczyć do obsługi tylko jednej wybranej partycji, dzięki czemu zwiększy się przepustowość.

W celu zastosowania partycjonowania danych należy wykonać dwa podstawowe kroki. Pierwszy polega na dokładnym określeniu, jak wybrany obiekt ma być partycjonowany. Drugi sprowadza się do wyznaczenia dla każdej partycji fizycznej lokalizacji, w której będzie przechowywana. Różne partycje mogą zostać powiązane z jednym lub wieloma plikami grupy.

W poniższym przykładzie pokazano składnię instrukcji tworzących funkcję i schemat partycji zakresowej, która dokona partycjonowania tabeli.

```
CREATE PARTITION FUNCTION MyPF
 (int) AS RANGE LEFT FOR VALUES (50, 100)
GO
CREATE PARTITION SCHEME MyPS
  AS PARTITION MyPF TO (FileGroup1)
GO
CREATE TABLE MyTable (col1 int, col2 varchar(50))
  ON MyPS(col1)
GO
```

W pierwszym wierszu tworzona jest funkcja partycji o nazwie `MyPF`. Łańcuch `(int)` oznacza, że partycjonowanie będzie wykonywane dla kolumny, którą zdefiniowaniu przy użyciu typu danych `int`. Słowo kluczowe `RANGE` określa, że zostanie użyta partycja zakresowa. Słowo kluczowe `LEFT` decyduje o tym, która partycja otrzyma wartości graniczne. Użycie tego słowa sprawia, że każdy wiersz posiadający wartość zgodną z wartością graniczną partycji zostanie od razu przesunięty na lewą stronę partycji. Klauzula `VALUES` służy do definiowania punktów granicznych partycji. Trzeba podkreślić, że wyżej podane wartości są punktami granicznymi, a nie samym partycjami. Powoduje to w efekcie utworzenia trzech partycji. Pierwsza partycja będzie zawierała ujemne wartości i dodatnie aż do liczby 50, druga — wartości od 51 do 100, natomiast trzecia — wszystkie wartości od 101 w górę.

W drugim wierszu tworzony jest schemat partycji o nazwie `MyPS`. Klauzula `AS PARTITION` służy do określenia funkcji partycji, która zostanie użyta przez schemat. W przykładzie zastosowano funkcję partycji `MyPF`. Klauzula `TO` identyfikuje grupę lub grupy plików przechowujące partycje. W przykładzie zastosowano jedną grupę plików o nazwie `FileGroup1`.

W kolejnym wierszu schemat partycji jest przypisywany do tabeli, która będzie partycjonowana. W przykładzie pokazana jest rozszerzona składnia instrukcji `CREATE TABLE`, pozwalająca na partycjonowanie tabeli. Pierwsza część składni instrukcji nie uległa zmianie. Identyfikuje nazwę tabeli (w tym przypadku `MyTable`) i jej kolumny. Przykładowa prosta tabela złożona jest z dwóch kolumn o nazwach `col1` i `col2`. Nowe słowo kluczowe `ON` służy do określenia schematu partycji, który zostanie użyty. W przykładzie zastosowano wcześniej zdefiniowany schemat partycji `MyPS`. W nawiasach okrą-

głych podana jest nazwa kolumny z danymi odgrywającymi rolę klucza partycji. W tym przypadku jest to kolumna `col1` typu danych `int`, który musi być zgodny z typem określonym dla funkcji partycji.

Istnieje kilka ograniczeń dotyczących typów kolumn, które mogą pełnić funkcję klucza partycji. Ograniczenia te są bardzo podobne do restrykcji nałożonych na kolumny używane w indeksie. Nie można korzystać z kolumn takich typów danych, jak `text`, `ntext` i `image`.

To samo dotyczy kolumn typu danych `timestamp`. Używane mogą być jedynie własne typy danych języka T-SQL. Jako klucza partycji nie wolno zastosować typu użytkownika. Można jednak wykorzystać w tym celu nowy typ danych `varchar(max)`. Istnieje też ograniczenie liczby partycji przypadających na tabelę; może ich być maksymalnie 1000. Wszystkie partycje muszą znajdować się na jednym węźle.

Rozszerzenia indeksu

Serwer SQL Server 2005 oferuje wiele rozszerzeń dotyczących indeksów. Pierwsze powoduje, że przebudowa indeksu zgrupowanego nie wiąże się już z koniecznością wykonania takiej samej operacji w przypadku wszystkich indeksów niezgrupowanych. Gdy w serwerze SQL Server 2000 przebudowywano zgrupowany indeks, to samo należało zrobić dla wszystkich powiązanych z nim indeksów niezgrupowanych. W przypadku serwera SQL Server 2005 indeksy niezgrupowane pozostają nienaruszone podczas przebudowy zgrupowanego indeksu.

Kolejne nowe rozszerzenie pozwala na dołączanie do indeksu kolumn pozbawionych klucza. Dzięki temu rozszerzeniu więcej kwerend może być obsłużonych przez indeks, co przyczynia się do zwiększenia wydajności przetwarzania. Jednocześnie minimalizuje się liczbę przypadków, w których mechanizm serwera SQL Server zmuszony jest do użycia tabeli w celu zakończenia przetwarzania kwerendy. Mechanizm serwera może spełnić wymagania kwerendy, używając jedynie danych oferowanych przez zaangażowany indeks. Jednym z wygodniejszych aspektów nowego rozszerzenia jest to, że dołączane kolumny, które nie tworzą klucza, nie są uwzględniane w maksymalnym rozmiarze indeksu. Dzięki temu nadal wynosi on 900 bajtów.

Następnym nowym rozszerzeniem indeksu dodanym przez Microsoft do serwera SQL Server 2005 jest możliwość wyłączania indeksu. Operacja taka powoduje, że indeks przestaje być utrzymywany przez mechanizm serwera SQL Server, a także uniemożliwia skorzystanie z indeksu. Po wyłączeniu indeksu serwer SQL Server zwalnia zajmowaną przez niego przestrzeń dyskową, ale zatrzymuje metadane indeksu. Zanim wyłączony indeks będzie mógł być ponownie uaktywniony, musi zostać odbudowany przy użyciu instrukcji `ALTER INDEX`.

Operacje na indeksie w trybie online

W poprzednich wersjach serwera SQL Server nie można było uzyskać dostępu do indeksu, gdy był akurat przebudowywany. Zanim można było ponownie uaktualnić tabelę, należało poczekać do zakończenia operacji przebudowywania indeksu. Dzięki

nowym operacjom przetwarzającym indeks w trybie online w przypadku serwera SQL Server 2005 aplikacje mogą uzyskiwać dostęp do indeksu, a także wykonywać dla tabeli operacje aktualizacji, wstawiania i usuwania, gdy trwa przebudowywanie indeksu. Więcej informacji na temat operacji wykonywanych przez serwer SQL Server 2005 na indeksie w trybie online zawarto w rozdziale 3.

Rozszerzenia dotyczące katalogu systemowego i metadanych

W serwerze SQL Server 2000 i jego starszych wersjach katalog systemowy i metadane jako część każdej bazy danych przechowywane były w bazie danych `master`. W serwerze SQL Server 2005 uległo to zmianie i obecnie metadane znajdują się w bazie danych `resource`. Mają one postać obiektu `sys`. Serwer SQL Server 2005 nie umożliwia już uzyskania bezpośredniego dostępu do tabel systemowych. Dzięki tej zmianie poprawiono poziom zabezpieczeń. Konsolidując systemowe metadane, przyspieszono wykonywanie aktualizacji. Metadane katalogu są zabezpieczane przy użyciu filtrów poziomu wiersza. Więcej informacji na temat zabezpieczeń poziomu wiersza zapewnianych przez serwer SQL Server 2005 można znaleźć w podrozdziale „Zabezpieczenia", znajdującym się w dalszej części rozdziału.

Jeśli nie używano nieudokumentowanych tabel systemowych, czego nieustannie Microsoft każdemu odradzał, nowe metadane będą całkowicie zgodne wstecz. Systemowe metadane w serwerze SQL Server 2005 udostępniane są przy użyciu zestawu widoków katalogu. Widoki katalogu, a także widoki ANSI INFORMATION_SCHEMA, funkcje właściwości i funkcje wbudowane eliminują potrzebę korzystania z tabeli systemowych w taki sposób, jak w przypadku serwera SQL Server 2000. Serwer SQL Server 2005 oferuje ponad 250 nowych widoków katalogu, które mogą być przeglądane za pośrednictwem schematu `sys` każdej bazy danych użytkownika. W celu odszukania nowych widoków systemowych należy użyć narzędzia Microsoft SQL Server Manager Studio, a następnie uruchomić program Object Browser i przejść do węzła *Databases|<baza_danych>|Views|System Views*. Można też otworzyć okno nowej kwerendy i wprowadzić następujące zapytanie:

```
SELECT * FROM sys.system_views
```

Funkcja MARS (Multiple Active Results Sets)

Poprzednie wersje serwera SQL Server były ograniczone do jednego aktywnego zestawu wynikowego przypadającego na połączenie. Obecnie serwer SQL Server 2005 jest w stanie obsługiwać dla pojedynczego połączenia wiele aktywnych zestawów wynikowych. Nowa funkcja MARS umożliwia nawiązanie połączenia z bazą danych, wykonanie kwerendy i zmodyfikowanie jej wyników, a następnie rozpoczęcie kolejnego zapytania i przetworzenie jego wyników. Aplikacje mogą się swobodnie przełączać między wieloma aktywnymi zestawami wynikowymi. W rozdziale 4. wyjaśniono na przykładach, jak korzystać z nowej funkcji MARS.

Ładowanie dużej ilości danych

W przypadku wczytywania dużych ilości danych serwer SQL Server 2005 oferuje kilka znakomitych udoskonaleń i większą wydajność. Obecnie operacja wczytywania oparta jest na pliku formatu XML, który zapewnia wszystkie funkcjonalności formatu pliku używanego w poprzednich wersjach narzędzia BCP (ang. *Bulk Copy Program*) i nie tylko. Ponadto format XML sprawia, że plik formatu obsługiwanego przez narzędzie BCP jest prostszy do odczytania i zrozumienia. Z myślą o zachowaniu zgodności wstecz z istniejącymi aplikacjami plik starego formatu może być nadal używany.

Procesor ładowania dużej ilości danych realizowany przez serwer SQL Server 2005 obsługuje obecnie rejestrowanie złych wierszy. Dzięki temu proces może być kontynuowany nawet wtedy, gdy natrafi na niepoprawne wiersze lub dane. Nieprawidłowo sformatowane wiersze zapisywane są w pliku błędów wraz z opisem okoliczności wystąpienia błędu. Wiersze naruszające ograniczenia przekazywane są do tabeli błędów wraz z opisem sytuacji, w której wystąpił błąd.

Wyszukiwanie pełnotekstowe

W serwerze SQL Server 2005 rozszerzono również obsługę funkcji wyszukiwania pełnotekstowego. Starsze wersje serwera SQL Server do utworzenia katalogów wyszukiwania pełnotekstowego wymagały użycia procedur przechowywanych. W serwerze SQL Server 2005 pojawiło się kilka nowych instrukcji DDL umożliwiających korzystanie z funkcji wyszukiwania pełnotekstowego. Przykładowo, dwie nowe instrukcje DDL to `CREATE FULLTEXT CATALOG` i `CREATE FULLTEXT INDEX`.

Do innych rozszerzeń wyszukiwania pełnotekstowego, które dodano do serwera SQL Server 2005, należy zaliczyć możliwość archiwizowania i odzyskiwania katalogów i indeksów funkcji wyszukiwania wraz z danymi zawartymi w bazie danych. Katalogi i indeksy wyszukiwania pełnotekstowego mogą być podłączane do odpowiednich baz danych i od nich odłączane. Kolejnym interesującym rozszerzeniem serwera SQL Server związanym z wyszukiwaniem jest możliwość użycia tezaurusa w celu znalezienia synonimów szukanych słów.

Rozszerzenia procesora zapytań języka T-SQL

W serwerze SQL Server 2005 pojawiło się kilka rozszerzeń procesora zapytań, takich jak CTE (ang. *Common Tables Expressions*), rozbudowana klauzula `TOP` i instrukcja `WAITFOR`, a także nowa klauzula `OUTPUT` instrukcji DML. W rozdziale 4. zamieszczono przykłady zastosowania tych rozszerzeń procesora zapytań języka T-SQL.

Zabezpieczenia

Odkąd Microsoft zainaugurował program Trustworthy Computing Initiative, zabezpieczenia stały się dla firmy ważnym elementem strategii. Celem tego programu jest sprawienie, aby wszystkie produkty Microsoftu były bezpieczniejsze i bardziej niezawodne. Jak można było oczekiwać, serwer SQL Server 2005 uwzględniony w programie stał się przedmiotem kilku bardzo istotnych rozszerzeń zabezpieczeń. Celem Microsoftu jest uczynienie produktu bardziej bezpiecznym i niezawodnym. Wymogi te brane są pod uwagę na każdym etapie, począwszy od projektowania, a skończywszy na wdrażaniu. Projektując rozszerzenia zabezpieczeń serwera SQL Server 2005, Microsoft postępował zgodnie z kilkoma podstawowymi zasadami. Przede wszystkim firmie zależało na tym, aby serwer był bezpieczny od razu po zainstalowaniu. Bezpieczne środowisko miało być rezultatem odpowiedniego skonfigurowania domyślnych ustawień instalacyjnych. Choć użytkownikom pozostawiono możliwość wyboru mniej bezpiecznych ustawień, taką decyzję należy podejmować z rozwagą. W dalszej kolejności Microsoft opracowując swoje systemy przestrzegał zasady najmniejszych przywilejów. System jest tak projektowany, aby użytkownik dysponował tylko przywilejami niezbędnymi do zrealizowania określonego zadania, i nic ponadto. Microsoftowi zależało wreszcie na zmniejszeniu liczby obszarów narażonych na atak. W tym celu umożliwił zainstalowanie tylko niezbędnych komponentów.

Na wszystkie nowe funkcje zabezpieczeń serwera SQL Server 2005 duży wpływ miały kwestie, które Microsoft odkrył na początku 2002 r. podczas wzmożonych działań mających na celu poprawienie zabezpieczeń, a następnie uwzględnił przy projektowaniu i wdrażaniu serwera SQL Server 2005. Do niektórych podstawowych rozszerzeń zabezpieczeń serwera SQL Server 2005, które omówiono w tym podrozdziale, należy zaliczyć: wyizolowanie użytkowników ze schematów, nowy kontekst wykonywania procedury przechowywanej, bardziej szczegółowa kontrola uprawnień, wymuszanie stosowania zasady haseł, modyfikacje zabezpieczeń poziomu wiersza i udoskonalenie zabezpieczeń katalogu.

Wyizolowanie użytkowników ze schematów

Najbardziej istotną zmianą dotyczącą zabezpieczeń, która pojawiła się w serwerze SQL Server 2005, jest wyizolowanie użytkowników ze schematów. Mianem użytkownika, a ściślej pryncypała określa się dowolną jednostkę, w powiązaniu z którą zabezpieczane są obiekty bazy danych. Pryncypałem może być użytkownik systemu Windows lub serwera SQL Server, rola lub rola aplikacji. W przypadku serwera SQL Server 2000 obiekty bazy danych bezpośrednio będące w posiadaniu użytkowników i sami użytkownicy znajdowali się w tabeli systemowej `sys_users`. W serwerze SQL Server 2005 uległo to zmianie. Obecnie obiekty bazy danych są w posiadaniu schematów. Użytkownicy nie mogą już bezpośrednio być posiadaczami obiektów bazy danych, są natomiast posiadaczami schematów. W serwerze SQL Server 2005 użytkowników i innych pryncypałów zabezpieczeń można znaleźć w nowym widoku `sys.database_principals`. Lista schematów serwera SQL Server 2005 jest udostępniana w nowym widoku `sys.schemas`.

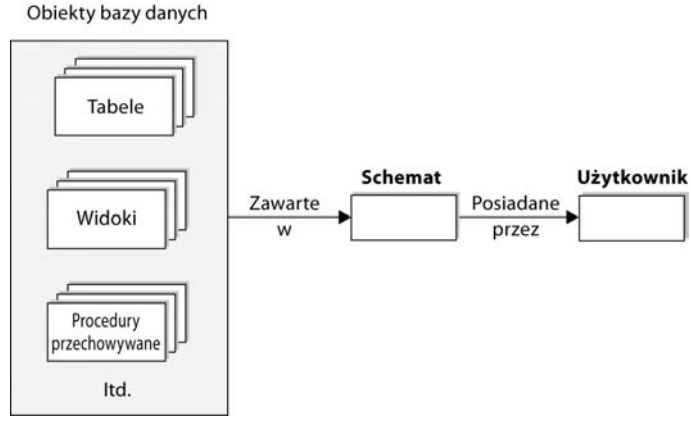
Schemat jest kontenerem obiektów. Identyfikowany jest przez trzeci składnik czteroczęściowej składni nazwy obiektów stosowanej przez serwer SQL Server. W poniższym przykładzie zaprezentowano składnię nazewniczą używaną przez serwer SQL Server 2005, w której każda część jest coraz dokładniejsza.

```
nazwa_serwera.nazwa_bazy_danych.nazwa_schematu.nazwa_obiektu
```

W przypadku wszystkich poprzednich wersji serwera SQL Server nazwa schematu i właściciela w zasadzie oznaczały to samo. W serwerze SQL Server 2005 dokonano oddzielenia właściciela od schematu. Gdy serwer SQL Server 2000 i jego starsze wersje identyfikowały nazwy obiektów, najpierw szukały łańcucha *nazwa_bazy_danych.nazwa_użytkownika.nazwa_obiektu*. Jeśli się to nie powiodło, serwer szukał łańcucha *nazwa_bazy_danych*.dbo.*nazwa_obiektu*.

Głównym powodem oddzielenia w serwerze SQL Server 2005 użytkowników od schematów jest rozwiązanie problemu polegającego na konieczności zmiany właściciela dla wielu obiektów bazy danych, gdy określony użytkownik (inaczej stary właściciel obiektów) zakończy pracę w firmie. Zmiana właściciela obiektu bazy danych powodowała też modyfikację jego nazwy. Jeśli na przykład właściciel tabeli `Table1` w bazie danych `MyDB` zmieni nazwę z użytkownik `UserA` na użytkownik `UserB`, kwalifikowana nazwa tabeli `Table1` zmieni się z `MyDB.UserA.Table1` na `MyDB.UserB.Table1`. Aby zapobiec temu problemowi, wiele organizacji zaadaptowało standard, w którym posiadaczem wszystkich obiektów bazy danych jest schemat `dbo`. Jednak w samym serwerze nie było nic, co by wymusiło stosowanie tego standardu.

Sposób praktycznej realizacji przez serwer SQL Server 2005 zagadnienia schematu bazy danych wprowadza poziom abstrakcji w łańcuchu praw własności do obiektu bazodanowego. Łańcuch taki pokazano na rysunku 1.3.



Rysunek 1.3. *Łańcuch praw własności do obiektu bazodanowego serwera SQL Server 2005*

W przypadku serwera SQL Server 2005 obiekty bazodanowe zawarte są w schematach, będących w posiadaniu użytkowników. Nowy poziom abstrakcji sprawia, że problem ze zmianą praw własności do obiektów bazy danych staje się znacznie mniejszy. Usunięcie w serwerze SQL Server 2005 użytkownika będącego właścicielem obiektów bazodanowych oznacza, że administrator musi teraz jedynie zmienić właściciela schematu, a nie wszystkich poszczególnych obiektów bazodanowych. Dzięki temu

znacznie spada liczba obiektów, które administrator bazy danych musi zmodyfikować w celu zmiany ich właściciela. Aby zmienić właściciela wszystkich obiektów bazy danych serwera SQL Server 2005, należy po prostu ustalić innego właściciela schematu, a następnie usunąć starego użytkownika. Zmiana właściciela obiektu bazodanowego nie powoduje modyfikacji jego nazwy, ponieważ zmianie nie ulega nazwa schematu, a jedynie jej właściciel.

Jak można oczekiwać, nowy obiekt schematu zmienia sposób, w jaki serwer SQL Server tłumaczy nazwę obiektu bazodanowego. Obecnie z każdym użytkownikiem powiązany jest domyślny schemat. Serwer SQL Server 2005 szukając niekwalifikowanej nazwy obiektu użyje najpierw domyślnego schematu użytkownika. Jeśli się to nie uda, serwer SQL Server poszuka obiektu korzystając ze schematu `dbo`. Jeśli na przykład użytkownik `UserA` posiada domyślny schemat `MySchema1` i wykona kwerendę szukającą tabeli `Table1`, serwer spróbuje najpierw przetłumaczyć nazwę przy użyciu schematu `MySchema1.Table1`, a następnie korzystając z `dbo.Table1`.

Podobnie jak bazy danych serwera SQL Server 2000 mogły zawierać wiele użytkowników i ról, również bazy serwera SQL Server 2005 mogą dysponować wieloma schematami. Każdy schemat posiada własnego pryncypała, którym zwykle jest użytkownik lub rola. Na potrzeby rozwiązywania nazw każdemu użytkownikowi przypisywany jest domyślny schemat. Same obiekty bazodanowe zawarte są w schemacie. W celu utworzenia w schemacie nowych obiektów bazy danych trzeba dysponować uprawnieniem `CREATE` dla obiektu bazodanowego i uprawnieniem `ALTER` lub `CONTROL` dla schematu. Łańcuch praw własności nadal jest oparty na właścicielach, a nie na schematach.

W serwerze SQL Server 2005 dokonano kilku zmian w instrukcjach DDL, które mają umożliwić korzystanie z rozdzielonych użytkowników i schematów. Należy do nich zaliczyć instrukcje `CREATE`, `DROP` i `ALTER`, powiązane z obiektami `USER`, `ROLE` i `SCHEMA`. Poniższy kod ukazuje, w jaki sposób schemat jest tworzony i przypisywany.

```
/* Utworzenie identyfikatora logowania */
CREATE LOGIN UserA WITH PASSWORD = 'ABC123#$'
GO
/* Utworzenie użytkownika dla identyfikatora – istnienie schematu nie jest konieczne */
CREATE USER UserA FOR LOGIN UserA
  WITH DEFAULT_SCHEMA = MySchema
GO
/* Utworzenie schematu i przypisywanie mu właściciela */
CREATE SCHEMA MySchema AUTHORIZATION UserA
GO
/* Utworzenie tabeli w nowym schemacie */
CREATE TABLE MySchema.Table1 (col1 char (20))
GO
```

W pierwszym wierszu powyższego kodu tworzony jest nowy identyfikator logowania `UserA` i definiowane jego hasło. W kolejnym wierszu tworzony jest nowy użytkownik `UserA` powiązany z identyfikatorem logowania i jest mu przypisywany domyślny schemat `MySchema`. Sam schemat nie musi istnieć w momencie, gdy jego nazwa podawana jest w instrukcji `CREATE USER`. Jeśli podczas tworzenia nowego użytkownika nie określi się domyślnego schematu, będzie nim schemat `dbo`. W następnym wierszu przy użyciu instrukcji `CREATE SCHEMA` tworzony jest nowy schemat o nazwie `MySchema`. Klau-

zula AUTHORIZATION ustanawia użytkownika UserA właścicielem schematu. Na końcu w schemacie MySchema definiowana jest tabela o nazwie Table1. Właścicielem schematu MySchema i jego obiektów, takich jak tabela Table1, jest użytkownik UserA.

Kontekst wykonywania procedury przechowywanej

Choć Microsoft nazywa jedną z nowych funkcji serwera SQL Server kontekstem wykonywania procedury przechowywanej, tak naprawdę dotyczy ona raczej modułów, nie zaś tylko procedur przechowywanych. Modułem może być procedura przechowywana, funkcja lub zestaw. Zdefiniowanie kontekstu wykonywania dla modułu spowoduje, że dla wszystkich zawartych w nim instrukcji sprawdzane będą uprawnienia powiązane z kontekstem. Inaczej mówiąc, po ustawieniu kontekstu wykonywania dla określonego modułu wszystkie jego instrukcje zostaną przetworzone z uprawnieniami wybranego użytkownika, a nie tego, który uaktywnił moduł. Choć ta nowa funkcja pozwala uzyskać podobne korzyści jak w przypadku łańcucha praw własności serwera SQL Server 2000, jest bardziej elastyczna i dotyczą jej ograniczenia, które występowały wcześniej. Przykładowo, w przeciwieństwie do łańcucha praw własności serwera SQL Server 2000, który uniemożliwiał modyfikowanie kontekstu wykonywania dla dynamicznych zapytań SQL, moduł kontekstu wykonywania serwera SQL Server 2005 można stosować w równym stopniu w przypadku dynamicznych i statycznych zapytań SQL. Aby to lepiej zrozumieć, należy przyjrzeć się rysunkowi 1.4 ukazującemu łańcuch praw własności serwera SQL Server 2000.

Rysunek 1.4. *Łańcuch praw własności serwera SQL Server 2000*

Aby użytkownik UserA mógł wykonać procedurę dbo.Proc1, musi dysponować uprawnieniem wykonywania dla tego obiektu. Jednak gdy procedura dbo.Proc1 próbuje uzyskać dostęp do tabeli dbo.Table1, nie są sprawdzane żadne uprawnienia, ponieważ schemat dbo jest właścicielem obu obiektów. Jest to przykład nienaruszanego łańcucha praw własności. W kolejnym przypadku, aby użytkownik UserA mógł wykonać procedurę UserB.Proc2, musi dysponować uprawnieniem wykonywania dla tego obiektu. Gdy procedura UserB.Proc2 spróbuje uzyskać dostęp do tabeli UserC.Table2, będzie musiało zostać sprawdzone, czy użytkownik UserA dysponuje uprawnieniem wybierania. Ponieważ w tym przypadku procedura UserB.Proc2 i tabela UserC.Table2 mają różnych właścicieli, łańcuch praw własności zostanie przerwany.

W przypadku kontekstu wykonywania serwera SQL Server 2005 powyższy przypadek staje się prostszy (rysunek 1.5). Gdy użytkownik UserA próbuje wykonać procedurę

Rysunek 1.5. *Kontekst wykonywania w serwerze SQL Server 2005*

`UserB.Proc2`, serwer SQL Server upewnia się, czy dysponuje odpowiednim uprawnieniem. Jeśli obiekt `UserB.Proc1` tworzony jest w kontekście wykonywania określającym, że ta procedura przechowywana będzie przetwarzana z uprawnieniami użytkownika `UserZ`, po podjętej przez nią próbie uzyskania dostępu do tabeli `UserC.Table1` serwer SQL Server sprawdzi uprawnienie wybierania tylko dla użytkownika podanego w kontekście wykonywania, którym w tym przypadku jest użytkownik `UserZ`. Od użytkownika `UserA`, który faktycznie wywołuje procedurę, nie są wymagane żadne uprawnienia wybierania.

Poniżej pokazano, w jaki sposób zmienić kontekst wykonywania dla procedury przechowywanej o nazwie `MyProc1`.

```
ALTER PROC MySchema.Proc1 WITH EXECUTE AS USER UserB
```

Powyższa instrukcja zawiera nową klauzulę `WITH EXECUTE`. W tym przypadku procedura przechowywana `Proc1` zawarta w schemacie `MySchema` jest tak ustawiana, aby była wykonana w kontekście użytkownika `UserB`. Z kontekstem wykonywania trzeba powiązać użytkownika. Nie może to być nazwa roli. Zmiany dokonywane w kontekście wykonywania przechowywane są w nowym widoku `sys.sql_modules`.

Bardziej szczegółowa kontrola uprawnień

Serwer SQL Server 2005 oferuje również bardziej szczegółową kontrolę uprawnień. W przypadku tej wersji serwera SQL Server Microsoft zastosował więcej uprawnień na różnych poziomach, takich jak serwer, baza danych, schemat, obiekt i pryncypał. Przy poszerzaniu uprawnień w serwerze SQL Server 2005 kierowano się zasadą najmniejszych przywilejów, która umożliwia administratorowi bazy danych dokładne kontrolowanie tego, w jaki sposób przydzielane są uprawnienia. Nowe uprawnienia nadal korzystają z istniejących ról serwera SQL Server. Wszystkie starsze role są wciąż obecne i bez problemów mogą współistnieć z nowymi uprawnieniami. Jedną z sytuacji, z myślą o których nowe uprawnienia zostały zdefiniowane, jest inspekcja. Aby w serwerze SQL Server 2000 przeprowadzić inspekcję, należało być członkiem grupy *sysadmins*. Jednak członkostwo w tej grupie daje wiele dodatkowych znacznie większych możliwości. Niektóre z nowych uprawnień dostępnych w serwerze SQL Server 2005 umożliwiają wykonywanie funkcji inspekcji bez wymagania dodania użytkownika do grupy *sysadmins*.

W nowej wersji serwera SQL Server nadal obowiązują te same podstawowe metody przetwarzania uprawnień przy użyciu instrukcji `GRANT`, `DENY` i `REVOKE`, co w poprzednich wersjach. Sposób, w jaki serwer SQL Server 2005 stosuje uprawnienia, różni się jedynie tym, że te same uprawnienia mogą być przypisane na wielu poziomach. Jeśli na przykład uprawnienie zostanie przydzielone na poziomie bazy danych, będzie ono dotyczyło wszystkich obiektów w niej zawartych. Jeśli uprawnienie zostanie powiązane z poziomem schematu, będzie obowiązywało tylko dla tych obiektów, które się

w nim znajdują. Z wyjątkiem uprawnień odbieranych za pomocą instrukcji DENY zawsze używane są uprawnienia powiązane z wyższym poziomem. Jednak na dowolnym poziomie pierwszeństwo ma instrukcja DENY. W tabeli 1.1 zawarto niektóre z najważniejszych nowych uprawnień serwera SQL Server 2005. Uprawnienia serwerowe dostępne są w widoku sys.server_permissions, natomiast uprawnienia bazodanowe — w widoku sys.database_permissions.

Tabela 1.1. *Niektóre nowe uprawnienia, które pojawiły się w serwerze SQL Server 2005*

Uprawnienie	Opis
ALTER	Umożliwia zmianę właściwości obiektu, a także wykonywania instrukcji CREATE, DROP i ALTER.
ALTER ANY '*X*'	Umożliwia modyfikację dowolnego obiektu typu *X*. Jeśli na przykład w miejsce *X* wstawiono by TABLE, uprawnienie pozwoliłoby zmodyfikować dowolną tabelę bazy danych.
ALTER TRACE	Umożliwia przeprowadzenie inspekcji i uruchomienie narzędzie Profiler.
CONTROL	Udziela uprawnień równorzędnych posiadanym przez właściciela-pryncypała.
SELECT	Umożliwia uzyskanie dostępu do obiektu. Obecnie uprawnienie dotyczy poziomów schematu i bazy danych zamiast wyłącznie poziomu obiektów.
EXECUTE	Umożliwia wykonanie procedury, zestawu lub funkcji. Obecnie uprawnienie dotyczy poziomów schematu i bazy danych zamiast wyłącznie poziomu obiektów.
IMPERSONATE	Umożliwia identyfikatorowi logowania lub użytkownikowi wcielenie się w innego użytkownika.
TAKE OWNERSHIP	Umożliwia przejęcie praw własności do obiektu.
VIEW DEFINITION	Umożliwia przeglądnięcie metadanych obiektu.

Wymuszanie stosowania zasady haseł

Kolejnym istotnym nowym rozszerzeniem zabezpieczeń serwera SQL Server 2005 jest obsługa zasad haseł. Nowa funkcja wymuszająca stosowanie zasad haseł przestrzega lokalnych zasad haseł systemu Windows i umożliwia użycie spójnych w skali całej organizacji zasad zabezpieczeń obowiązujących dla serwerów z systemem Windows i serwerów bazodanowych SQL Server. Obecnie serwer SQL Server 2005 jest w stanie wymuszać przestrzeganie zasad dotyczących złożoności haseł, daty ich ważności i blokowania kont. Jak można się domyślić, zasada złożoności haseł wymusza stosowanie haseł o określonym stopniu złożoności. Zasada daty ważności haseł gwarantuje, że hasła utracą ważność i będą musiały być cyklicznie ponownie definiowane. Zasada blokowania kont umożliwia zablokowanie konta, gdy określoną ilość razy zostanie wprowadzone nieprawidłowe hasło. Wszystkie te nowe zasady haseł są obsługiwane przez system Windows Server 2003. W przypadku systemu Windows 2000 Server obsługiwana jest jedynie zasada złożoności haseł. Zgodnie z programem Microsoftu mającym na celu poprawienie zabezpieczeń, wszystkie zasady są domyślnie uaktywniane podczas instalacji. Jednakże mogą też być przekonfigurowane dla poszczególnych zalogowanych użytkowników. Nowe zasady haseł są przechowywane przez serwer SQL Server 2005 w widoku katalogu sys.sql_logins.

Zabezpieczenia katalogu

Ostatnim rozszerzeniem serwera SQL Server 2005 związanym z zabezpieczeniami, które omówiono w tym podrozdziale, są udoskonalone zabezpieczenia katalogu. Tabele systemowe, które były używane przez serwer SQL Server 2000 w poszczególnych bazach danych i bazie `master`, obecnie w serwerze SQL Server 2005 są udostępniane za pośrednictwem widoków katalogu. Serwerowe metadane zawarte w tych widokach domyślnie są zabezpieczone. Widokom przypisane są minimalne publiczne uprawnienia. W przypadku widoków katalogu serwera SQL Server 2005 zastosowano zabezpieczenia poziomu wiersza, ograniczając użytkownikowi dostęp do danych znajdujących się w widokach tylko do tych obiektów, których jest właścicielem lub tych, do których posiada uprawnienia. Oczywiście wyjątkiem jest konto *są*, które nadal dysponuje dostępem do wszystkich obiektów serwera.

W celu użycia użytkownika lub roli, która uzyska dostęp do metadanych, administrator bazy danych musi skorzystać z nowego uprawnienia `VIEW DEFINITION`. Uprawnienie to może posłużyć do umożliwienia przeglądu metadanych obiektu użytkownikowi, który nie jest jego właścicielem ani też nie dysponuje uprawnieniami udzielającymi mu do niego dostępu. Uprawnienie `VIEW DEFINITION` może zostać zastosowane na poziomie serwera, bazy danych, schematu lub obiektu.