helion.pl
KSIĘGARNIA INTERNETOWA · WYDAWNICTWO

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

KATALOG KSIĄŻEK
KATALOG ONLINE
ZAMÓW DRUKOWANY KATALOG

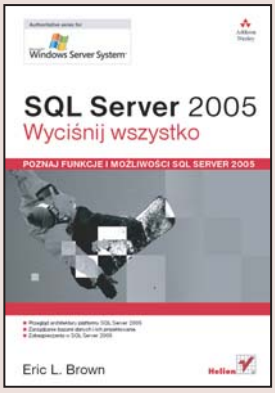
TWÓJ KOSZYK
DODAJ DO KOSZYKA

CENNIK I INFORMACJE
ZAMÓW INFORMACJE O NOWOŚCIACH
ZAMÓW CENNIK

CZYTELNIA
FRAGMENTY KSIĄŻEK ONLINE

SQL Server 2005. Wyciśnij wszystko

Autor: Eric L. Brown
Tłumaczenie: Kosma Nitkiewicz
ISBN: 978-83-246-0549-1
Tytuł oryginału: SQL Server 2005 Distilled (MS Windows Server System)
Format: B5, stron: 320



Poznaj funkcje i możliwości SQL Server 2005

- Przegląd architektury platformy SQL Server 2005
- Zarządzanie bazami danych i ich projektowanie
- Zabezpieczenia w SQL Server 2005

SQL Server 2005 to najnowsza wersja flagowego produktu Microsoftu przeznaczonego do zarządzania danymi i ich analizowania. Zapewnia ona większą wydajność, bezpieczeństwo, a także udostępnia szereg nowych funkcji związanych między innymi z usługami analitycznymi i raportowaniem. Jak działają te funkcje i które z nich są Ci naprawdę potrzebne? Jakiej wersji SQL Server 2005 potrzebujesz? Jakie korzyści dla Twojej firmy przyniesie zastosowanie nowych rozwiązań bazodanowych? Jak wykorzystać możliwości SQL Server 2005, aby usprawnić działanie używanych baz danych?

W udzieleniu odpowiedzi na powyższe pytania pomoże Ci książka „SQL Server 2005. Wyciśnij wszystko"; dzięki niej w przystępny sposób poznasz możliwości tej platformy. Dowiesz się, jak działa SQL Server 2005 oraz jakie są różnice między jej poszczególnymi wersjami. Zrozumiesz funkcjonowanie i znaczenie różnych zabezpieczeń oraz poznasz funkcje i usługi związane z bezpieczeństwem. Zobaczysz, na czym polega zarządzanie firmowymi bazami danych i jak wygląda projektowanie aplikacji bazodanowych. Przeczytasz o generowaniu analiz i raportów oraz wykonywaniu innych zadań za pomocą platformy Business Intelligence.

- Przegląd wersji i możliwości SQL Server 2005
- Zabezpieczenia i kontrola nad danymi
- Administrowanie korporacyjnymi bazami danych
- Projektowanie aplikacji bazodanowych
- Analizy i raporty w Business Intelligence
- Pisanie i diagnozowanie kodu za pomocą Visual Studio
- Przykładowy kod demonstrujący stosowanie nowych funkcji

Ta książka będzie dla Ciebie nieocenioną pomocą przy podejmowaniu decyzji dotyczących technologii bazodanowych.

Wydawnictwo Helion
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl

Spis treści

Wstęp **11**

O autorze **17**

Rozdział 1. Wprowadzenie do platformy SQL Server 2005 ..... **19**

Wersje platformy SQL Server 2005 22
Różnice pomiędzy SQL Server Enterprise a Standard Edition 22
SQL Server 2000 w wersji 64-bitowej 25
Nowości w SQL Server 2005 26
Skalowalność bazy danych — podstawy 30
Baza danych jako serwer aplikacji 32
Zwiększenie dostępności systemów SQL Server 36
Wysoka dostępność na platformie SQL Server 2005 36
Rozszerzenie wysokiej dostępności do wszystkich aplikacji bazodanowych 39
Dystrybucja dzienników kopii zapasowej 41
Ogólna dostępność danych 42
Partycjonowanie tabel — horyzontalne partycjonowanie danych 43
Widoki materializowane 44
Indeksowanie w trybie online 45
Utrzymywanie bardzo dużych baz danych 45
Rozszerzenia funkcjonalności wykonywania kopii zapasowej i odzyskiwania danych 46
Przegląd funkcji Business Intelligence platformy SQL Server 2005 46
Przegląd usług Analysis Services 48
Data Mining 50

Funkcje relacyjne hurtowni danych ... 51
Ekstrakcja, transformacja i ładowanie danych w SQL Server 2005 ... 52
Reporting Services ... 53
Najbardziej wysublimowane funkcje SQL Server 2005 ... 54
Podsumowanie ... 56

Rozdział 2. Co wszyscy powinni wiedzieć o bezpieczeństwie ... 57
Podstawowe funkcje zabezpieczające silnik SQL Server ... 60
Wymuszanie polityki haseł ... 61
Zasada najmniejszych przywilejów ... 62
Kontekst wykonawczy ... 67
Poświadczenia ... 67
Certyfikaty i Secure Socket Layer ... 69
Szyfrowanie ... 71
Koncepcja bezpieczeństwa punktu końcowego ... 77
Bezpieczeństwo usług sieciowych ... 77
Bezpieczeństwo usług Notification Services ... 78
Bezpieczeństwo mirroringu bazy danych ... 78
Bezpieczeństwo usługi SQL Service Broker ... 79
Bezpieczeństwo kodu zarządzanego ... 80
Bezpieczeństwo usługi Analysis Services ... 81
Bezpieczeństwo na poziomie kostki ... 84
Bezpieczeństwo na poziomie wymiaru ... 84
Bezpieczeństwo na poziomie komórki ... 84
Bezpieczeństwo struktury i modelu drążenia danych oraz źródła danych ... 85
Dostęp Analysis Services do procedur składowanych CLR ... 85
Funkcje bezpieczeństwa usługi SQL Server Integration Services ... 86
Bezpieczeństwo na poziomie pakietu ... 86
Role bazy danych w usłudze SSIS ... 87
Zabezpieczanie środowiska operacyjnego ... 87
Podpisywanie pakietów certyfikatami ... 88
Szyfrowanie pakietów SSIS ... 88
Bezpieczeństwo usługi Reporting Services ... 89
Zabezpieczanie serwera ... 90
Usługa SQL Browser Service ... 91
SQL Server Configuration Manager ... 92
Narzędzie SQL Server Surface Area Configuration ... 93
Podsumowanie ... 95

Rozdział 3. Korporacyjne zarządzanie danymi ... 97
SQL Server Management Studio ... 99
Stan połączenia lub rozłączenia ... 101
Object Explorer ... 102
Tworzenie bazy danych ... 103
Tworzenie tabel ... 103

Query Editor ... 104
Niemodalne okna dialogowe ... 104
Dostosowywanie SQL Server Management Studio ... 106
Projekty i rozwiązania związane z SQL Server Management Studio ... 106
Pomoc ... 108
Zarządzanie infrastrukturą SQL Server 2005 ... 108
Migawka bazy danych ... 110
Co nowego w programie instalacyjnym? ... 111
Integracja z programem Dr Watson ... 113
Narzędzia do zarządzania operacyjnego ... 114
SQL Computer Manager ... 114
SQL Server Agent ... 115
Funkcje zdalnego zarządzania platformą SQL Server 2005 ... 116
SQLCMD ... 117
SQL Server Management Objects ... 118
Windows Management Instrumentation ... 118
Monitorowanie SQL Server ... 119
Monitoring reaktywny ... 120
Monitoring proaktywny ... 125
Zarządzanie bardzo dużymi bazami danych ... 130
Partycjonowanie tabel i indeksów ... 130
Rozszerzenia replikacji SQL Server ... 135
Replikacja peer-to-peer ... 137
Oracle Publication ... 138
Uaktualnienie replikacji opartej na sieci WWW ... 138
Wydajność i skalowalność ... 139
Wysoka dostępność dla wszystkich ... 140
Rozwiązania zapewniające wysoką dostępność ... 140
Ogólna dostępność danych ... 149
Izolacja migawki ... 150
Widoki indeksowane ... 152
Common Language Runtime i administrator baz danych ... 153
Jak głęboka jest integracja? ... 153
Business Intelligence i administrator baz danych ... 156
Podsumowanie ... 157

Rozdział 4. Funkcje do projektowania baz danych ... 159
Integracja z Visual Studio ... 160
Integracja z .NET Framework ... 162
SQL Server jako środowisko uruchomieniowe ... 163
Wybór pomiędzy SQL transakcyjnym a kodem zarządzanym ... 164
Typy danych zdefiniowane przez użytkownika ... 165
Funkcje użytkownika ... 166
Funkcje agregujące definiowane przez użytkownika ... 166

SQL Server Manager Provider 167
Zarządzane typy SQL 168
Technologia XML 170
Po co wykorzystywać relacyjne bazy danych do przechowywania danych w formacie XML? 170
Wybór technologii XML 172
Typ danych XML 174
Klauzula FOR XML 176
OPENXML 177
Obsługa XQuery 177
Rozwinięcia DML 178
Native XML Web Services 178
WSDL 180
SQL Server jako platforma serwera aplikacji 181
SQL Service Broker 182
Notification Services 187
SQL Server Mobile Edition 195
Ustawienia językowe w SQL Server 2005 196
Wyszukiwanie pełnotekstowe 198
Zapytania serwerów połączonych 199
Przeszukiwanie różnych kolumn 199
Specyfikacja języka w zapytaniu 199
SQL Server Express 201
Integracja z Visual Studio 202
Czy SQL Server Express wystarczy? 203
Ograniczenia platformy SQL Server Express 205
Instalacja i konfiguracja 206
Protokoły obsługiwane przez SQL Server Express 208
Podsumowanie 209

Rozdział 5. Przegląd funkcji Business Intelligence 211
Przegląd Analysis Services 213
Architektura Analysis Services 214
Ujednolicony model wymiarowy 216
Widoki źródła danych 219
Kreator kostki 220
Business Intelligence Wizard 221
Lokalizacja reprezentacji danych 222
Gdzie przechowuje się UDM? 223
Modele dostępu do danych OLAP 224
BI czasu rzeczywistego i proaktywne buforowanie 224
Analiza przechowywania danych w usłudze Analysis Services 227
Kluczowe wskaźniki wydajności 229
Język MDX (Multidimensional Expression) 230

Bezpieczeństwo usługi Analysis Services ... 231
Funkcje relacyjne dla hurtowni danych ... 231
Drążenie danych w SQL Server 2005 ... 234
SQL Server Integration Service ... 236
Architektura SSIS ... 237
Przepływ danych ... 238
Kontrola przepływu danych — nowe zastosowanie SSIS ... 241
Obsługa błędów w SSIS ... 242
Rozmieszczanie pakietów ... 243
Skalowanie SSIS — wybór pomiędzy 32-bitową a 64-bitową platformą sprzętową ... 243
Kompatybilność wstecz ... 245
SQL Server Reporting Services ... 245
Architektura usługi Reporting Services ... 247
Projektowanie raportów ... 248
Projektowanie modelu raportu ... 249
Kreator raportu ... 249
Definiowanie raportu ... 251
Kontrolki do przeglądania raportu ... 251
Grupa wykonania raportu ... 251
Zarządzanie raportami ... 252
Menedżer konfiguracji Reporting Services ... 253
Wdrażanie mechanizmów raportowania ... 254
Rozważania na temat Reporting Services ... 254
Zarządzanie i rozwijanie aplikacji Business Intelligence ... 255
SQL Server Management Studio ... 255
Business Intelligence Studio Development ... 256
Administrator baz danych Business Intelligence ... 256
Podsumowanie ... 258

Rozdział 6. Kod ... 259
Obsługa wyjątków transakcji ... 260
Model programistyczny dużych obiektów ... 260
Nowe funkcje analityczne ... 261
Warunek TABLESAMPLE ... 263
Nowy operator APPLY ... 263
Operatory PIVOT i UNPIVOT ... 264
Zapytania rekurencyjne ... 266
Rozszerzenia operatora TOP ... 269
Wyzwalacze DDL ... 271
DML Output ... 273
Rozszerzenia obsługi kolejkowania ... 274
Widoki indeksowane ... 275
Kolumny wyliczane w sposób ciągły ... 277
Synonimy ... 278

SQL transakcyjny a kwestia bezpieczeństwa 279
Zastosowanie ról 282
Szyfrowanie i certyfikaty 283
Praca z zestawami 285
Praca z XML 286
Usługi sieciowe XML 286
Widok Dynamic Management 287
Wyłączanie niektórych funkcji bazy danych 289
Partycjonowanie tabeli 290
Podsumowanie 293

Dodatek A Informacje o systemie SQL Server 2005 295

Dodatek B Tabele systemowe i widoki w SQL Server 2005 301

Dodatek C Funkcje wbudowane w SQL Server 305

Skorowidz 307

ROZDZIAŁ 4.

Funkcje do projektowania baz danych

W tym rozdziale omówię zestaw funkcji, z jakimi projektanci baz danych są zazwyczaj zaznajomieni. Administratorzy baz danych mogą skłaniać się do pominięcia tego zagadnienia, ja jednakże tak nie postąpię. Zmiany, które wprowadzono w prezentowanej wersji SQL Server, można po prostu opisać jako ponowne opracowanie struktur służących do projektowania nowych klas aplikacji. W SQL Server 2005 rozszerzalność całościowej platformy baz danych została znacznie zwiększona. Możliwości SQL Server w zakresie obsługi wspólnego środowiska uruchomieniowego (ang. *Common Language Runtime* — CLR) mogą spowodować oszołomienie. Takich elementów jest więcej. Wprowadzenie kodu zarządzanego do SQL Server oznacza większe możliwości integracji dla rozwoju pakietów programów i niestandardowych aplikacji. Zasadniczo platforma SQL Server składa się z trzech warstw.

- Pierwszą warstwę stanowią podstawowe mechanizmy, takie jak SQL Server Engine — SQL Server.exe.
- Drugą warstwą jest szkielet. Stąd interfejsy uzyskują dostęp do poszczególnych komponentów. Ta warstwa jest warstwą prywatną firmy Microsoft.

- W trzeciej warstwie możemy znaleźć zestaw programów dostarczonych przez Microsoft, wiążących interfejs aplikacji z obiektami SQL Server. SQL Server 2005 zapewnia nowy zakres funkcjonalności, włączając w to usługi Analysis Services i Replication, jak również dotychczasowe elementy platformy SQL Server.

Deweloperzy mogą pracować z trzecią warstwą opisywanej architektury bez korzystania z interfejsu użytkownika. Oznacza to możliwość używania takich elementów jak usługa SQL Server Integration Service do programowego przenoszenia danych. Można tworzyć usługi, które będą zarządzały kopiami zapasowymi całego serwera baz danych, nie korzystając przy tym z programu SQL Management Studio. W rzeczywistości najlepszym sposobem na zaprojektowanie aplikacji SQL jest skorzystanie z interfejsu użytkownika w celu utworzenia podstawowego szkieletu, a następnie opisanie skryptami całej aplikacji.

KRÓTKI PRZEGLĄD: FUNKCJE SQL SERVER DLA PROJEKTANTÓW

SQL Server 2005 oferuje projektantom baz danych dużo nowych udogodnień.

- Projektowanie obiektów bazy danych w językach platformy .NET. Można pisać obiekty proceduralne, zawierające funkcje, procedury i wyzwalacze, w takich językach, jak C# i VB.NET.
- Nowy zestaw obiektów kodu zarządzanego dla rozszerzenia i pracy z obiektami SQL Server.
- Projektowanie i uruchamianie programu zostało zintegrowane ze środowiskiem Visual Studio. Pozwala to projektantom na używanie tych samych narzędzi zarówno do tworzenia obiektów baz danych i skryptów, jak i do pisania komponentów i usług .NET.
- Przechowywanie dokumentów w języku XML (ang. *Extensible Markup Language*), tworzenie zapytań XML i zwracanie wyników zapytań w postaci dokumentów XML.
- SQL Service Broker wprowadza możliwość tworzenia aplikacji bazodanowych zorientowanych na usługi (SOA).

Integracja z Visual Studio

Dla administratorów czy projektantów baz danych Visual Studio .NET 2005 stanowi ogromny krok naprzód w pracy z bazami danych. Microsoft zawsze zapewniał lepszą obsługę własnych baz danych — przykładowo, w przypadku SQL Server wygląda to lepiej niż dla IBM DB2. Dzięki równoczesnemu utworzeniu platformy SQL Server 2005, .NET Framework 2.0, ASP.NET 2.0 i na-

rzędzi do projektowania aplikacji, Microsoft skutecznie zsynchronizował linię swoich produktów. Poszczególne wersje .NET Framework mogą funkcjonować równocześnie w tym samym systemie operacyjnym, zatem wprowadzanie nowych produktów powinno przebiegać bezproblemowo i nie wymaga usuwania poprzedniej wersji danego oprogramowania.

W poprzedniej wersji SQL Server trzeba było samodzielnie zaprojektować pakiety usług transformacji danych (ang. *Data Transformation Services* — DTS), kostki Analysis Services i moduły transakcyjnego SQL, używając odpowiednich narzędzi dla każdej technologii. W SQL Server 2005 można wykorzystywać Visual Studio do projektowania wszystkich elementów i instalować całą aplikację jednym kliknięciem myszy. Dzięki Visual Studio z jego wszystkimi narzędziami dla deweloperów uzyskujemy liczne korzyści.

- **Projektowanie zespołowe.** Dzięki kontroli systemu i źródeł projektu można monitorować cały proces budowy aplikacji, począwszy od modułów bazy danych, na interfejsie użytkownika skończywszy. Narzędzie to ułatwia również kontrolę nad poszczególnymi wersjami tworzonej aplikacji.
- **Uruchamianie i wdrażanie.** Visual Studio obsługuje teraz uruchamianie procedur składowanych i kodu zarządzanego w tym samym procesie usuwania błędów. W celu instalacji nowej aplikacji wystarczy skorzystać z funkcji instalacji — wtedy instalacja odbywa się za pomocą jednego kliknięcia.
- **Doświadczenie użytkownika.** Nowe SQL Server Management Studio i Business Intelligence Development Studio zostały zaprojektowane na podstawie interfejsu użytkownika Visual Studio. Funkcjonalność pomocy została zintegrowana w ten sposób, aby projektant mógł skorzystać z SQL Sever i Visual Studio Books Online z poziomu jednej aplikacji.

Klasycznym użytkownikom baz danych SQL Server zapewnia wszystkie potrzebne narzędzia poprzez SQL Server Management Studio. Typowy projektant rozwiązań Business Intelligence (BI)[1] może wykonywać swoją pracę na poziomie serwera i korzystać z usług Analysis Services, hurtowni danych, SQL Server Integration Services oraz Reporting Services za pomocą Business Intelligence Development Studio. Można by się zastanawiać, jak będzie wyglądało projektowanie bazy danych SQL Server, Business Intelligence albo aplikacji ASP.NET z wykorzystaniem Visual Studio 2005? Jeżeli wszystkie komponenty SQL Server zostały zainstalowane na komputerze razem z Visual Studio 2005, do dyspozycji

[1] Business Intelligence — oprogramowanie wspomagające procesy podejmowania decyzji poprzez analizę danych zgromadzonych w systemach informatycznych. Źródło: *http://pl.wikipedia.org/wiki/Business_intelligence — przyp. tłum.*

pozostają wszystkie narzędzia, a Visual Studio jest rozszerzone o nową możliwość projektowania komponentów SQL Server. Niestety, nie ma możliwości tworzenia aplikacji w C# za pomocą Business Intelligence Development Studio. Warto teraz zapoznać się z najczęściej omawianą funkcją SQL Server 2005, zapewniającą możliwość współpracy z .NET Framework.

Integracja z .NET Framework

Wydawać by się mogło, że klienci Microsoftu są jednymi z bardziej przedsiębiorczych i kreatywnych ludzi w dziedzinie projektowania aplikacji. Kiedy menedżerowie produktu i menedżerowie projektu z tej firmy rozmawiają z klientami o ich aplikacjach, często wypytują o sposób wykorzystywania technologii SQL Server. Często też słyszą głosy oburzenia klientów na wprowadzanie nowych funkcji projektowania aplikacji do produktów lub języków Microsoftu. Jeżeli baza danych prezentuje sobą pewne możliwości, osoby takie uważają się za ekspertów. Tymczasem warto wiedzieć, że wdrażając SQL Server jako środowisko uruchomieniowe, wchodzimy w nową erę kreatywności.

Dzięki integracji z CLR możemy pisać własne procedury składowane, wybierając do tego celu dowolny język .NET Framework. Języki programowania Microsoft — Visual Basic .NET i C# — umożliwiają tworzenie konstrukcji zorientowanych obiektowo, a także zapewniają strukturalną obsługę wyjątków, tablic, przestrzeni nazewniczych i klas. Dodatkowo .NET Framework dostarcza tysięcy wbudowanych klas i metod, które mogą być łatwo implementowane po stronie serwera. Wiele zadań, które były niewygodne lub trudne do wykonania za pomocą kodu transakcyjnego SQL, można zrealizować za pomocą kodu zarządzanego. Dodatkowo są dostępne dwa nowe typy danych — złożenia i typy definiowane przez użytkownika. Teraz można lepiej wykorzystać swoją wiedzę i umiejętności nabyte podczas pisania kodu pośredniego. W skrócie rzecz ujmując, SQL Server 2005 pozwala na rozwijanie bazy danych, aby mogła łatwiej wykonywać odpowiednie obliczenia i operacje po stronie wewnętrznej.

Taka integracja SQL Server z CLR daje kilka ważnych korzyści.

- Poprawa zabezpieczeń i ochrona dla nietypowych typów danych z wykorzystaniem kodu zarządzanego poprzez CLR.
- Funkcje i funkcje agregujące definiowane przez użytkownika. Są to dwa nowe obiekty baz danych, które rozszerzają możliwości przechowywania i wykonywania zapytań. Są dostępne dzięki współpracy z CLR.

- Bliższa współpraca pomiędzy bazą danych a logiką aplikacji. W bazie danych można umieścić więcej logiki biznesowej, która może być obsługiwana przez rozwinięte komponenty zarządzania bazą danych.

Pisząc kod, który ma bardziej złożoną logikę i jest lepiej dopasowany do zadań obliczeniowych, można oprzeć się na integracji z CRL i korzystać z takiego języka programowania, jak VB .NET czy C#. Dodatkowo Visual Basic .NET i C# umożliwiają wykorzystywanie pewnych właściwości programowania zorientowanego obiektowo, takich jak hermetyzacja, dziedziczenie i polimorfizm. Z łatwością można zorganizować powiązany kod w klasy i przestrzenie nazewnicze. Oznacza to, że pracując z dużą ilością kodu, możemy łatwiej go organizować i utrzymywać. Taka logiczna i fizyczna organizacja kodu w zestawy (ang. *assemblies*) i przestrzenie nazewnicze jest wielką korzyścią. Pozwala lepiej odnajdywać i łączyć różne części kodu w implementację dużej bazy danych.

SQL Server jako środowisko uruchomieniowe

W przeciwieństwie do COM (ang. *Component Object Model*) i rozszerzonych procedur składowania, umieszczenie na serwerze SQL Server 2005 kodu zarządzanego, oddziałującego na bazę danych, nie stwarza zagrożeń. Środowisko .NET Framework uwzględnia kilka istotnych aspektów wykorzystywania kodu zarządzanego.

Kod .NET wykorzystuje zabezpieczenia zapewniane przez kod dostępu. Zastosowano tu politykę bezpieczeństwa opartą na głównym uruchomieniu kodu. Należy pamiętać, że zazwyczaj istotną rolę w tym zakresie pełni konto użytkownika. Przywileje głównego użytkownika wpływają zarówno na sam kod, jak i na miejsce, z którego został załadowany. Po wczytaniu zestawu do pamięci, jego możliwości są automatycznie ograniczane do samego uruchomienia. Proces pamięci i środowisko systemowe, które załadowało zestaw, również może go ograniczać. Używając metafory, zestaw jest zmuszony do zabawy tylko w jednej piaskownicy (ang. *sandbox*). W SQL Server 2005 ta koncepcja została przekształcona do tzw. **AppDomain**. AppDomain izoluje wszystkie zasoby wykorzystywane przez SQL Server, których wymaga kod .NET. Dzieje się tak nawet wtedy, gdy SQL Server i kod .NET znajdują się w tej samej przestrzeni procesowej. AppDomain chroni SQL Server przed niewłaściwym lub złośliwym użyciem poszczególnych elementów zestawu, takich jak procedury składowane.

Podczas instalacji SQL Server 2005 można zauważyć, że nowa architektura zabezpieczeń domyślnie wyłącza zainstalowane funkcje. Co więcej, administrator bazy danych albo administrator systemowy musi jawnie przyznać użytkownikowi prawo do utworzenia zestawu w SQL Server 2005. Tworzenie zestawu w rzeczywistości jest załadowaniem wstępnie skompilowanego zapytania DDL do bazy danych.

Zestaw jest przechowywany w wierszu tabeli bazy danych. Aby sprawdzić, które zestawy zostały załadowane, należy przejrzeć katalog *sys.assemblies*. Po załadowaniu zestawu następuje ustawienie atrybutu polecenia DDL `verbiage` w celu ustalenia stanu bezpieczeństwa tego zestawu. Możliwe są trzy podstawowe stany:

- `SAFE`. Ten tryb dostępu przypisuje zestaw do procesu SQL Server.
- `External_Access`. W ten sposób zestaw może uzyskiwać dostęp do systemu operacyjnego i korzystać z zasobów systemu plików.
- `UNSAFE`. To najmniej bezpieczny tryb. Zezwala zestawowi na szeroki, potencjalnie stwarzający zagrożenia dostęp do instancji bazy danych i obszaru systemu operacyjnego, więc nie jest polecany do użytku.

Wybór pomiędzy SQL transakcyjnym a kodem zarządzanym

Podczas tworzenia procedur składowanych, wyzwalaczy i funkcji użytkownika trzeba podjąć decyzję, czy korzystać z kodu SQL transakcyjnego, czy z jakiegoś języka programowania kompatybilnego z .NET Framework, takiego jak Visual Basic .NET czy C#. Odpowiedź na to pytanie zależy od konkretnej sytuacji. W niektórych przypadkach warto użyć SQL transakcyjnego, w innych — kodu zarządzanego.

Język SQL jest najlepszy w sytuacjach, kiedy kod przede wszystkim uzyskuje dostęp do danych, jednak zastosowana logika proceduralna jest bardzo niewielka albo nawet w ogóle jej nie ma. Języki programowania kompatybilne z .NET są najodpowiedniejsze dla intensywnie wykorzystywanych funkcji procedur, które cechuje złożona logika, lub dla sytuacji, kiedy trzeba skorzystać z zalet biblioteki klas .NET Freamework.

Kod zarządzany jest bardziej efektywny niż SQL transakcyjny przy przetwarzaniu liczb i zarządzaniu skomplikowaną logiką wykonawczą. Zapewnia rozszerzoną obsługę ciągów znakowych i wyrażeń regularnych. Wszystkie operacje, począwszy od rozwiniętych funkcji matematycznych, poprzez operacje na datach, uzyskiwanie dostępu do zasobów systemowych czy plików, zaawansowanych algorytmów sortujących, przetwarzanie obrazów, a na przetwarzaniu danych w formacie XML skończywszy, są prostsze do wykonania za pomocą zarządzanych procedur składowanych, funkcji i funkcji agregujących. Wraz z .NET Framework można rozszerzyć typy i funkcje dostępne w SQL Server.

Typy danych zdefiniowane przez użytkownika

Typy danych definiowane przez użytkownika, które wprowadzono w SQL Server 2000, nie były łatwe w stosowaniu i niewiele było informacji na ten temat. Tworzenie i korzystanie z typów danych użytkownika we wcześniejszych wersjach nie było trudne, ale miało ograniczoną użyteczność. Zostało to zmienione w SQL Server 2005. Nowe typy danych zdefiniowane przez użytkownika (ang. *User-defined data types* — UDT) pozwalają na określenie abstrakcyjnych typów danych w dowolnym języku obsługiwanym przez .NET CLR. Takie typy danych mogą mieć skomplikowaną strukturę, a ich zachowanie jest zdeterminowane poprzez metody, pola i właściwości klasy zdefiniowane w zestawie .NET. UDT są mocno zintegrowane z systemem typów i silnikiem SQL Server i mogą być wykorzystane w każdym kontekście, w jakim wbudowane typy są dozwolone. Serwer pozwala na indeksowanie, replikację i ładowanie dużej ilości UDT.

Typy danych zdefiniowane przez użytkownika można wykorzystywać podczas definiowania kolumn w tabeli, w kodzie funkcji lub argumentach procedur składowanych. Takie dane mogą też być zmiennymi w funkcjach lub procedurach składowanych. Wyrażenia zawierające UDT mogą być stosowane w kontekście zapytania SQL transakcyjnego. Mimo że w tej książce nie będę zagłębiał się w tematykę UDT, administrator baz danych musi wiedzieć, w jaki sposób odczytać kod VB .NET albo C#, aby zrozumieć działanie UDT. UDT muszą być skonstruowane w pewien określony sposób. Projektanci znający obiekty `Struct` mogą tworzyć UDT w łatwiejszy sposób. Podczas tworzenia UDT trzeba skonkretyzować poniższe deklaracje.

- Należy wyspecyfikować `SqlUserDefiniedTypeAttribute`. W tabeli 4.1 przedstawiłem wszystkie potrzebne atrybuty.
- Trzeba określić atrybut `Serializable`.
- Należy zaimplementować interfejs `IsNullable` w klasie lub strukturze, tworząc pustą metodę `public static` (albo `shared` w Microsoft Visual Basic). SQL Server jest domyślnie ustawiony jako `null-aware`. Jest to niezbędne podczas wykonywania kodu, aby dane typu UDT zostały rozpoznane z wartością pustą.
- UDT muszą zawierać metody konwersji `public state` (albo `shared`) `Parse` i `ToString`, które umożliwiają przekształcenie do ciągu znakowego reprezentującego obiekt i analizę jego składni.
- Należy zapewnić metody odczytu i zapisu w klasach, w których wyspecyfikowano format serializacji użytkownika. UDT zdefiniowane w klasie muszą mieć publiczny bezargumentowy konstruktor. Można opcjonalnie utworzyć dodatkowy, przeciążony konstruktor.
- Trzeba wyeksponować elementy danych jako publiczne pola lub właściwości procedury.

Tabela 4.1. *Właściwości atrybutu SqlUserDefinedTypeAttribute*

Właściwość	Opis
`Format`	Określa format serializacji, który może przybierać wartość `Native` lub `UserDefined`, zależnie od typu danych UDT.
`MaxByteSize`	Maksymalny rozmiar instancji w bajtach. Należy zdefiniować `MaxByteSize` z określonym formatem serializacji `UserDefined`.
`IsByteOrdered`	Wartość logiczna (Boolean), która określa, jak SQL Server wykonuje porównania binarne UDT.
`ValidationMethodName`	Metoda wykorzystywana do sprawdzania poprawności instancji tego typu.
`IsFixedLength`	Wartość logiczna, która określa, czy wszystkie instancje tego typu mają taką samą długość.

Funkcje użytkownika

Metody napisane w dowolnym języku programowania .NET mogą zostać zarejestrowane jako funkcje użytkownika. Wprowadzone w SQL Server 2000 funkcje użytkownika są podprogramami, które na wejściu akceptują zerowe lub skalarne wartości albo parametry tabelaryczne oraz zwracają wartości skalarne lub tabelaryczne. Załóżmy, że projektant pracuje nad aplikacją księgową i musi zaimplementować obliczanie podatku dochodowego, płaconego przez poszczególnych pracowników w ciągu pewnego czasu. W tym celu projektant rejestruje metodę, specjalnie napisaną w C#, jako funkcję zdefiniowaną przez użytkownika. Można wtedy łatwo napisać zapytania, które będą wykorzystywać funkcję użytkownika, służącą do wyliczania podatku dochodowego pracowników.

Funkcje agregujące definiowane przez użytkownika

Funkcje agregujące zdefiniowane przez użytkownika (**UDAggs**) można tworzyć za pomocą dowolnego języka .NET. UDAggs pozwalają na zdefiniowanie dowolnych funkcji agregujących. W tym celu wykorzystuje się wbudowane funkcje agregujące, takie jak `SUM` i `MAX`. Można się do nich odwoływać w zapytaniach `SELECT` w taki sam sposób, jak do wbudowanych funkcji agregacji.

UDAggs reprezentują inny sposób włączania skomplikowanej logiki biznesowej, związanej z wykonywaniem zapytań. Proces ten zachodzi wewnątrz serwera, blisko danych. Może być bardziej efektywny niż dostarczanie danych do środkowej warstwy aplikacji i agregowanie ich w tym miejscu. Przy projektowaniu UDAggs trzeba spełnić pewne kryteria projektowe, aby wszystko działało poprawnie. Układ agregacji składa się z mechanizmu zapisującego pośredni stan

agregacji i mechanizmu gromadzącego nowe wartości, który składa się z czterech metod: `Init()`, `Accumulate()`, `Merge()` i `Terminate ()`. Podczas deklarowania UDAgg konieczne jest jawne określenie atrybutu `SqlUserDefinedAggregate`, który informuje SQL Server, że deklarowany typ danych dostosowuje się do struktury UDAgg. Dwie obowiązkowe właściwości atrybutu `SqlUserDefinedAggregate` są wymagane przez format serializacji i są to:

- właściwość `Format` określa rodzaj serializacji — dla każdego formatu serializacji istnieje określony zestaw wymagań (zalecam sprawdzenie tej informacji w SQL Server Books Online);
- `MaxByteSize` jest maksymalnym rozmiarem, wyrażanym w bajtach, potrzebnym do przechowywania danych podczas obliczeń. Maksymalna dopuszczalna wartość wynosi 8 000.

W celu poprawnego utworzenia i wykorzystywania UDAgg trzeba zadeklarować cztery funkcje. W tabeli 4.2 przedstawiłem te funkcje oraz ich zastosowanie.

Tabela 4.2. *Funkcje agregujące definiowane przez użytkownika*

Funkcja	Opis
`Init()`	Pozwala procesorowi zapytań na zainicjalizowanie funkcji agregacji.
`Accumulate()`	Wykorzystywana do agregacji danych — ładuje typ danych wyspecyfikowany w nawiasie.
`Merge()`	Pozwala na połączenie funkcji agregacji ze sobą. Jest to ważne w przypadku obliczeń matematycznych, posiadających różne wejścia.
`Terminate()`	Przerywa proces agregacji i zwraca końcową wartość.

Jeśli chodzi o wykonywanie sumowania, przeliczeń i innych podstawowych funkcji matematycznych, UDAggs są dużo bardziej funkcjonalne niż transakcyjny SQL. Tak więc UDAgg można wykorzystywać w przypadku funkcji statystycznych, geometrycznych i innych funkcji numerycznych.

Wprowadzenie UDAgg wymaga przestrzegania takich samych zasad zarządzania i rozmieszczania, co UDT.

SQL Server Manager Provider

Projektanci, pisząc procedury składowane lub wyzwalacze, mogą korzystać z różnych metod dostępu do danych zapewnianych przez ADO.NET, zoptymalizowanego dla kodu zarządzanego i uruchamianego na SQL Server. Dostawca jest zgodny z modelem programistycznym niemal identycznym ze standardem ADO.NET. Zarządzany dostawca jest stosowany w zestawach, które wykorzystują dane

SQL Server jako dane wejściowe i wyjściowe. Pisząc zarządzaną procedurę składowaną, która uaktualnia dane w tabeli, można zapewnić dostęp do bazy danych za pośrednictwem dostawcy zarządzanego. Podstawowa różnica pomiędzy ADO.NET a dostawcą zarządzanym polega na braku potrzeby podawania parametrów połączenia (ang. *connection string*), ponieważ połączenie z bazą danych jest już ustanowione. Dostęp do zarządzanego dostawcy uzyskuje się, odwołując się do `System.Data.SQLServer`.

Stosując tę zasadę, można otworzyć połączenie z SQL Server. W tym celu należy napisać następujący kod:

```
SqlConnection oConn = SqlContext.GetConnection();
SqlPipe oPipe = SqlContext.GetPipe();
```

Projektanci zaznajomieni z tworzeniem aplikacji ADO.NET mogą docenić zalety tego samego modelu podczas dodawania logiki do obiektów bazy danych dostępu do danych .NET. Dodatkowo, kod napisany na jednej warstwie jest łatwiejszy do przenoszenia do innej.

Zarządzane typy SQL

Microsoft SQL Server 2005 zapewnia obsługę typów zarządzanych, które odwołują się do dostępnych typów wbudowanych. Te typy danych, w formie parametrów dla funkcji użytkownika i procedur w językach .NET, takich jak VB .NET i C#, można wykorzystywać do wymiany informacji pomiędzy językami zarządzanymi a wbudowanym systemem typów SQL Server. Co więcej, zarządzane typy danych mogą być wykorzystywane do przyłączenia ich do kolumn tabel za pomocą zarządzanego dostawcy ADO.NET. Zarządzane typy są zestawami typów danych, które z kolei są częścią podstawowej biblioteki klas środowiska uruchomieniowego .NET Runtime. W przeciwieństwie do innych typów danych .NET Runtime, które nie zapewniają takiej składni, jak „NULL awareness", zarządzane typy charakteryzują się taką samą składnię i zachowaniem, jak odpowiednie wbudowane typy danych SQL Server. W porównaniu z innymi typami danych .NET, korzystanie z tych typów minimalizuje konieczność konwersji i utraty precyzji. Zarządzane typy danych SQL są zdefiniowane w przestrzeni nazewniczej `System.Data.SqlTypes`

Jedną z technologii, w które Microsoft inwestuje najbardziej, jest XML. W ciągu następnych pięciu lat większość głównych produktów Microsoftu będzie obsługiwało technologię XML. W kolejnym podrozdziale przedstawię funkcjonalność XML w programie SQL Server 2005.

UWAGA: Niektóre wbudowane typy danych SQL Server w zasadzie są przekształconymi typami danych SQL Server. W Books Online można znaleźć artykuł: „Typy danych (silnik bazy danych)". Należy pamiętać, że wykorzystywanie SQL Manager Provider i uzyskiwanie dostępu do bazy poprzez kod zarządzany jest możliwe pod warunkiem, że zastosowany kod jest rozpoznawalny przez SQL Server. Innymi słowy, typy danych, parametry i zwrócone wartości muszą mieć formę odpowiednią dla SQL Server. Poszczególne typy danych wymieniłem w tabeli 4.3, jednak nie wyszczególniłem tam typów danych użytkownika. Planując wykorzystywanie typów danych użytkownika w swojej organizacji, warto przygotować podobną tabelę i dołączyć ją do dokumentacji programu. Rozdział 6. „Kod" zawiera przykłady sposobów uzyskiwania dostępu i zastosowania obiektów CLR.

Tabela 4.3. Mapowanie typów SQL

Typy SQL (zarządzany dostawca)	Typy natywne SQL	Typy CLR
SQLBinary	Binary	Byte[]
	Varbinary	
	TIMESTAMP	
	IMAGE	
SQLInt64	BIGINT	Int64
SqlInt32	INT	Int32
SQLInt16	SMALLINT	Int16
SQLByte	TINYINT	Byte
SQLString	CHAR	String
	VARCHAR	
	NCHAR	
	NVARCHAR	
	TEXT	
	NTEXT	
	SYSNAME	
SqlDateTime	DATETIME	DateTime
	SMALLDATETIME	
SqlDecimal	DECIMAL	Decimal
	NUMERIC	
SqlDouble	FLOAT	Double
SqlSingle	REAL	Float
SqlMoney	MONEY	Decimal
	SMALLMONEY	
SqlGuid	UNIQUEIDENTIFIER	Guid
SqlBoolean	BIT	Boolean

Technologia XML

XML jest technologią, w którą stale się inwestuje i która nieustannie staje się coraz bardziej interesująca. Pierwotnie XML stanowiło rozwiązanie wyraźnie wyartykułowanych problemów. Obecnie XML jest powszechnie wykorzystywane — począwszy od Really Simple Syndication (RSS), aż po skomplikowaną integrację aplikacji, XML jest formatem przekazywania danych. XML jest z natury samoopisującym się językiem i może zawierać moc zbioru, zatem wielu projektantów aplikacji bazodanowych stosuje XML do rozwiązywania problemów wydajności aplikacji. W tym rozdziale przedstawię sposób wykorzystywania funkcji XML na platformie SQL Server.

Po co wykorzystywać relacyjne bazy danych do przechowywania danych w formacie XML?

Przechowywanie danych XML w relacyjnych bazach danych ułatwia zarządzanie danymi i ich wyszukiwanie. Bazy danych w formacie XML nie zapewniają tych korzyści albo nie są do końca rozwinięte. SQL Server zapewnia dużą wydajność zapytań i łatwość zarządzania relacyjnymi danymi, włączając w to możliwość tworzenia zapytań i modyfikowania danych XML. Pozwala to na wykorzystanie zmian dokonanych w poprzednich wersjach programu. Na przykład techniki indeksacji, które w relacyjnych bazach danych są dobrze znane, zostały rozszerzone do indeksowania danych XML. Miało to na celu zoptymalizowanie zapytań za pomocą decyzji podejmowanych po analizie kosztów. Dane XML mogą współpracować z istniejącymi relacyjnymi bazami danych i aplikacjami SQL, więc XML może zostać wprowadzony do systemu, ponieważ konieczność modelowania danych nie powoduje zakłócania pracy istniejącej aplikacji.

Poniżej wyszczególniłem niektóre powody, dla których warto przechowywać dane XML w relacyjnych bazach danych:

- Platforma SQL Server pozwala na wygodne zarządzanie danymi XML z wykorzystaniem wszystkich niezbędnych funkcji, w tym systemu zabezpieczeń.
- Wyszukiwanie lub modyfikacja danych XML może zachodzić w efektywny sposób, z uwzględnieniem mechanizmu transakcji.
- Dostęp do danych XML jest dobrze zorganizowany, co jest bardzo ważne z punktu widzenia działania aplikacji. Na przykład można uzyskiwać dostęp do określonych części dokumentu XML lub wstawiać do dokumentu nową sekcję bez zamieniania całego dokumentu.

- Aplikacje oparte na danych relacyjnych oraz SQL umożliwiają współpracę danych relacyjnych z danymi XML w ramach takiej aplikacji. Zastosowanie XML ułatwi zapewnienie obsługi językowej dla kwerend i modyfikację danych w aplikacji międzydomenowej.
- Łatwo uzyskać poprawne formatowanie danych XML, a zachowanie pełnej zgodności jest możliwe dzięki sprawdzeniu za pomocą XML Schema (XSD).
- SQL Server zapewnia indeksowanie danych XML dla zwiększenia efektywności zapytań.
- SQL Server umożliwia uzyskiwanie dostępu do danych XML za pomocą Simple Object Access Protocol (SOAP), ADO.NET i OLE DB.

SQL Server 2005 pozwala na dokonanie wyboru spośród kilku możliwości: dane XML można przechowywać jako dane typu `Varchar(max)` lub jako dane XML. Format `Varchar(max)` jest dobrym rozwiązaniem, kiedy chodzi tylko o składowanie niskiego poziomu. Jeżeli potrzeby są większe i dane są przechowywane poza systemem plików, zaleca się zastosowanie typu danych XML. Typ danych XML jest wbudowanym typem danych SQL Server ze wszystkimi cechami innych wbudowanych typów. Przyjrzyjmy się bliżej typowi danych XML.

Dane XML są przechowywane we własnej wewnętrznej reprezentacji, która chroni ich wewnętrzną strukturę, taką jak hierarchia zawartości, układ dokumentu, wartości elementu i atrybutów itp. Szczególnie chroniony jest atrybut **`Information Set`** danych XML (więcej informacji o `Information Set` znajduje się na stronie *http://www.w3.org/TR/xml-infoset*). Ograniczeniem typu danych XML jest niemożność zapamiętywania takich informacji, jak białe znaki (ang. *white spaces*), kolejność atrybutów, prefiksy przestrzeni nazewniczej i deklaracja XML. Jeżeli w danej sytuacji właściwości `Information Set` dokumentu XML są niewystarczające, należy użyć typów danych służących do przechowywania dużych obiektów, takich jak `Varchar(max)` albo `Varbinary`, ponieważ zachowują one wszystkie właściwości dokumentu. Kolumny tego typu posiadają wiele znaczących ograniczeń, takich jak brak możliwości użycia indeksów XML dla dokumentów XML.

Dla zdefiniowanych typów danych XML (tj. typów danych XML powiązanych ze schematami XML), walidacja poschematowa (ang. *Post-schema validation Information Set* — PSVI), która dodaje informacje o typie do `Information Set`, jest kodowana w wewnętrznej reprezentacji danych. To znacznie zwiększa możliwości analizy składni (więcej informacji na ten temat znajduje się w specyfikacji World Wide Web Consortium, W3C, na stronie *http://www.w3.org/TR/xmlschema-1* i *http://www.w3.org/TR/xmlschema-2*).

Ogólnie rzecz ujmując, niekiedy obydwa te rozwiązania okazują się przydatne. Na przykład można przechowywać dane XML w kolumnie typu XML i przenosić właściwości do kolumn relacyjnych. Zamiennie można wykorzystać technologię mapowania i przechowywania nierekurencyjnych części w kolumnach o typach różnych od XML i tylko rekurencyjnych części w kolumnach o typach XML.

Wybór technologii XML

Wybór technologii XML (XML natywny, czyli wbudowany, lub podgląd XML) zależy przede wszystkim od poniższych uwarunkowań:

- Dane XML mogą być bardziej odpowiednie dla przechowywania dużych obiektów (na przykład instrukcja użytkowania produktu) lub lepiej się nadawać do przechowywania w kolumnach relacyjnych (takich jak niewielki element przekształcony do postaci XML). Każdy z tych wariantów magazynowania do pewnego stopnia zabezpiecza jakość dokumentu.
- Wybór odpowiedniejszego wariantu przechowywania danych zależy od sposobu i granic wykonywania zapytań na danych XML. Obsługa dobrze zorganizowanych zapytań danych XML, uwzględniających takie mechanizmy, jak przewidywalne szacowanie węzłów XML, zależy od wybranego sposobu magazynowania danych.
- Indeksowanie danych XML może przyspieszyć wykonywanie zapytań XML. Opcje indeksacji różnią się zależnie od opcji przechowywania danych, więc trzeba dokonać odpowiedniego wyboru, aby zoptymalizować działanie aplikacji.
- Niektóre procesy wymagają dobrze zorganizowanej modyfikacji danych XML (na przykład dodawanie nowej sekcji do dokumentu), inne tego nie wymagają (na przykład zawartość strony WWW). Obsługa poleceń DDL może być istotna dla aplikacji.
- Dane XML mogą być opisane za pomocą schematów, które mogą, ale nie muszą, być schematami dokumentów XML. Obsługa schematów związanych z XML zależy od wyboru technologii XML.

Typ danych Native XML

Przechowywanie danych w formacie Native XML jest użyteczne w przypadku, kiedy dokumenty XML posiadają różną strukturę lub jeśli ich schematy są zbyt różnorodne czy skomplikowane, aby je opisać za pomocą struktury relacyjnej. W tabeli 4.4 przedstawiłem powiązania pomiędzy typami danych SQL Server.

Tabela 4.4. *Porównanie kolumn Varchar i XML*

Funkcjonalność	Varchar()	Typ danych XML
Ochrona struktury dokumentu XML	X	Information Set
Nieznany schemat	X	X
Znany schemat		X
Indeks XML		X
Zapytanie wewnątrz XML		X
Ładowanie dużej ilości danych	X	X
Użycie XPath		X

Widoki XML

Aby utworzyć widok XML (ang. *XML View*) trwałych danych, należy zdefiniować powiązania pomiędzy schematami XML a tabelami w bazie danych. Widoki XML mogą służyć do zapełniania tabel podstawowych techniką masowego ładowania danych (ang. *bulk load*), można też odpytywać widok XML, korzystając z XPath 1.0. Taka kwerenda jest tłumaczona na zapytanie SQL do odpowiedniej tabeli. Podobnie są wykonywane uaktualnienia tych tabel. Taka technologia okazuje się użyteczna, kiedy trzeba zapewnić model programowania oparty na XML, który korzysta z widoku XML podczas uzyskiwania dostępu do danych relacyjnych. Istnieją schematy, które można wykorzystać do pracy z danymi XML (takie jak XSD czy XDR — *XML-Data Reduced*), oferowane przez zewnętrznych dostawców. Dodatkowo warto wspomnieć, że widoki XML są przydatne w następujących sytuacjach:

- kolejność danych jest nieistotna lub dane potrzebne do zapytania nie są rekurencyjne albo też jest znana maksymalna liczba rekurencji w głąb;
- istnieje potrzeba wyszukania lub zmodyfikowania danych poprzez widok XML za pomocą XPath 1.0;
- istnieje potrzeba masowego ładowania danych XML i przeniesienia ich do odpowiednich tabel za pomocą widoku XML.

W powyższych sytuacjach dane relacyjne są przechowywane w postaci XML, co ma na celu łatwiejszą wymianę danych, usług sieciowych i danych XML z ustalonym stałym schematem (więcej informacji znajduje się pod adresem *http://msdn.microsoft.com/SQLXML*).

Typ danych XML

Twórcy SQL Server 2005 wprowadzili nowy skalarny typ danych XML, który jest wariantem typu *binary large object* (BLOB). Typ danych XML w SQL Server 2005 spełnia wymogi określone w specyfikacji W3C dla typu danych XML, a nawet wychodzi poza specyfikację (konstrukcje modyfikacji danych XML). XML może modelować skomplikowane dane, nie ograniczając się tylko do skalarnych typów danych obsługiwanych przez SQL Server. Tak więc takie typy danych, jak wbudowane, oparte na ciągach znakowych `Char` lub `Varchar`, nie wystarczą do pełnego i efektywnego wykorzystania wielu cennych właściwości XML. Przykładowo, jeżeli XML jest przechowywany jako ciąg znakowy, można wstawiać lub wybierać cały dokument albo nawet uzyskać z niego ciąg bajtów, ale nie można przeszukiwać zawartości samego dokumentu. Dzięki nowemu typowi danych XML, platforma SQL Server 2005 pozwala na przeszukiwanie części dokumentu XML, sprawdzenie, czy dokładnie odpowiada on schematowi XML, a nawet zmodyfikowanie zawartości dokumentu XML. Pozwala także na integrację standardowych, relacyjnych danych z danymi nieustrukturyzowanymi lub wstępnie ustrukturyzowanymi dokumentami XML, w sposób, jaki nie był możliwy przy użyciu SQL Server 2000. Platforma SQL Server 2005 przechowuje dane XML jako dane typu BLOB w wewnętrznej reprezentacji, która umożliwia efektywną analizę i pewną kompresję.

Kolekcja schematów XML może być powiązana z kolumnami typu XML. Spowoduje to sprawdzenie poprawności dla połączeń, wstawiania, uaktualniania i wpisywania wartości wewnątrz przechowywanych danych XML, jak również optymalizację przechowywania i przetwarzania zapytań. SQL Server 2005 zapewnia także kilka poleceń DDL służących do zarządzania schematami na serwerze. Typ danych XML posiada metody — `query()`, `exist()`, `value()`, `nodes()` oraz `modify()` — które implementują ważny podzbiór specyfikacji XML Query (XQuery).

Niezdefiniowane, zdefiniowane i ograniczone typy danych XML

Typ danych SQL Server XML zaprojektowano zgodnie ze standardem ISO SQL-2003 dla danych XML. Tak więc, może on przechowywać dobrze zorganizowane dokumenty XML 1.0, jak również fragmenty zawartości dokumentów XML z węzłami tekstowymi oraz pomocnicze numery szczytowych elementów w *niezdefiniowanych* kolumnach XML. System sprawdza sposób formatowania danych, nie wymaga powiązania kolumn ze schematami XML i odrzuca dane, jeżeli nie są odpowiednio sformatowane. Dotyczy to także niezdefiniowanych zmiennych i parametrów XML.

Schemat XML opisujący dane XML może zostać powiązany z kolumną XML, dzięki czemu powstanie *zdefiniowana* kolumna XML. Schemat XML, wykorzystany do sprawdzenia poprawności danych, przeprowadza podczas kompilacji zapytania i poleceń DDL dokładniejsze analizy niż niezdefiniowany XML — optymalizuje przechowywanie danych i przetwarzanie zapytań.

Definiowane kolumny XML, parametry i zmienne mogą przechowywać dokumenty XML lub ich zawartość. Trzeba zadeklarować odpowiednią wartość flagi (`DOCUMENT` lub `CONTENT`). Co więcej, należy też dostarczyć kolekcję schematów XML. Trzeba ustawić wartość `DOCUMENT`, jeżeli każde wystąpienie XML ma tylko jeden element szczytowy — w innym przypadku musimy użyć `CONTENT`. Kompilator zapytań stosuje flagę `DOCUMENT` do sprawdzania typu podczas kompilacji kwerendy określającej szczytowe elementy. Dodatkowo do opisania kolumny XML można użyć więzów relacyjnych (kolumna lub wiersz) dla zdefiniowanych bądź niezdefiniowanych kolumn typów danych XML.

Indeksowanie kolumn typu dane XML

Indeksy XML można tworzyć na kolumnach danych typu XML. Wszystkie znaczniki, wartości i ścieżki są indeksowane za pomocą instancji XML w tej kolumnie, która optymalizuje przetwarzanie zapytań. Aby indeksy XML przynosiły korzyści, należy spełnić następujące warunki:

- Zapytania na kolumnach XML powinny być wykonywane na co dzień. Trzeba wziąć pod uwagę koszty utrzymania indeksów XML podczas modyfikacji danych. Można popatrzeć na plan, żeby zrozumieć wpływ indeksów XML na przetwarzanie zapytań.
- Wartości XML powinny być stosunkowo duże, a wydobywane części danych stosunkowo niewielkie. Tworzenie indeksów zapobiega rozdzielaniu wszystkich danych podczas pracy aplikacji i przynosi korzyści w wyszukiwaniu indeksów dla wydajnego przetwarzania zapytań.

Pierwszym indeksem kolumny XML jest podstawowy indeks XML (ang. *primary*). Za jego pomocą można utworzyć drugorzędny indeks XML dla kolumny XML, żeby przyspieszyć działanie zwykłej klasy zapytań. Istnieją trzy typy drugorzędowych indeksów XML.

Indeks podstawowy indeksuje wszystkie znaczniki, wartości i ścieżki wewnątrz wystąpienia XML w kolumnie XML. Podstawowa tabela musi mieć zgrupowany indeks dla klucza podstawowego tabeli. Klucz podstawowy jest wykorzystywany do korelowania indeksów wiersza z tym właśnie wierszem w tabeli podstawowej. Pełne wystąpienia XML są uzyskiwane z kolumn XML. Wykonywane zapytania wykorzystują podstawowe indeksy XML, zwracając skalarne wartości lub poddrzewa XML.

Po utworzeniu podstawowego indeksu XML, można utworzyć drugorzędny indeks XML, żeby przyspieszyć wykonywanie różnych klas zapytań wewnątrz procesu pracy. Drugorzędne indeksy XML — indeksy typu ścieżki, właściwości i wartości — przynoszą korzyści, odpowiednio, zapytaniom opartym na ścieżce, scenariuszom zarządzania właściwościami i zapytaniom opartym na wartości:

- Indeks typu ścieżki (PATH) buduje drzewo B+ (ścieżka, wartość) pary dla każdego węzła XML w porządku dokumentu dla każdego wystąpienia XML w kolumnie.
- Indeks typu właściwości (PROPERTY) tworzy drzewo B+ zgrupowane (PK, ścieżka, wartość) pary wewnątrz każdego wystąpienia XML, gdzie PK jest kluczem podstawowym tabeli podstawowej.
- Indeks typu wartości (VALUE) tworzy drzewo B+ (wartość, ścieżka) pary dla każdego węzła porządku dokumentu poprzez wszystkie wystąpienia XML w kolumnie.

Jeżeli plan pracy zawiera ścieżki trudne do zdefiniowania w kolumnach XML, dodatkowy indeks typu ścieżki (PATH) najprawdopodobniej przyspieszy działanie środowiska pracy. Najczęściej stosowanym przypadkiem jest użycie metody `Exist()` dla kolumny XML w klauzuli `WHERE` zapytania SQL.

Jeżeli środowisko pracy stosuje różne wartości z indywidualnych wystąpień XML, korzystając z wyrażenia ścieżki, pomocne może okazać się zgrupowanie ścieżek wewnątrz każdego wystąpienia XML w indeksie typu właściwości (PROPERTY). Taki scenariusz pojawia się w przypadku takich struktur danych, gdzie właściwości obiektu są załadowane i wartość indeksu podstawowego jest znana. Jeżeli praca polega na tworzeniu zapytań o wartości wewnątrz wystąpienia XML bez znajomości tego elementu lub nazw atrybutu, które zawierają te wartości, można rozważyć utworzenie indeksu `VALUE`.

Klauzula FOR XML

Twórcy SQL Server 2005 wprowadzili również kilka rozszerzeń do klauzuli `FOR XML`. Klauzula `FOR XML`, właściwa dla SQL Server 2000, nie określała sposobu użytkowania wyników XML na serwerze. Oczywiście, nie można przechowywać wyników w formacie XML w tabeli bez zwrócenia ich najpierw do klienta lub przypisania do zmiennej. Twórcy platformy SQL Server 2005 rozszerzyli klauzulę `FOR XML`, zapewniając obsługę danych typu XML, co pozwala na wykorzystanie danych bezpośrednio na serwerze. W tym celu należy dodać dyrektywę `TYPE` w klauzuli `FOR XML`. Na przykład wynik polecenia `SELECT...FOR XML TYPE` wygeneruje dane XML, które mogą zostać przypisane lokalnie do zmiennej XML lub mogą zostać wykorzystane w późniejszym poleceniu `INSERT` do

wypełnienia kolumny danych XML. Tryb PATH określa ścieżkę w drzewie XML, gdzie powinna pojawić się wartość kolumny. Opcja TYPE i opcja PATH, włączone do klauzuli FOR XML, upraszczają generowanie skomplikowanych danych XML i są wygodniejsze w użyciu niż zapytania FOR XML EXPLICIT. FOR XML działa także z kolumnami typu XML w SQL Server 2005. Głównymi udogodnieniami, ułatwiającymi wykorzystywanie zapytań FOR...XML, są:

- możliwość wyboru przestrzeni nazw dla wstawionego schematu XSD;
- możliwość tworzenia danych XML opartych na centralnym elemencie (ang. *element-centric*) z wykorzystaniem czystego XML;
- możliwość tworzenia głównego elementu dla fragmentów XML w celu ich przekształcenia w dokument XML;
- udoskonalenia w obsłudze tzw. białych spacji (ang. *white spaces*) oraz możliwość zagnieżdżania przy korzystaniu z XML auto. Można także zagnieżdżać zapytania FOR XML, pozwalając na zwiększenie elastyczności przygotowania dokumentów XML.

Nie będę dokładnie opisywał tego nowego mechanizmu. Zainteresowanym Czytelnikom polecam lekturę *A First Look At Microsoft SQL Server 2005 for Developers* (Bob Beauchemin i in., wydawnictwo Addison Weasley Professional). Jest to znakomita pozycja omawiająca możliwości XML w ramach SQL Server 2005.

OPENXML

SQL Server 2000 zasadniczo obsługiwał polecenie FOR XML, umożliwiające tworzenie dokumentów XML, oraz polecenie OPENXML do ich otwierania. Innymi słowy, korzystając z FOR XML, można przekształcić dane relacyjne do postaci XML, a korzystając z OPENXML, można przekształcić XML w dane relacyjne, dzięki czemu można utworzyć połączenia SQL i wykonywać zapytania. SQL Server 2005 zwiększa funkcjonalność polecenia OPENXML. Jako dodatek do typów danych XML dodano obsługę kilku nowych typów danych, takich jak na przykład UDT. Można je stosować w klauzulach OPENXML WITH, można także umieszczać wystąpienia typów danych XML w sp_preparedocument.

Obsługa XQuery

XML Query Language lub XQuery jest standaryzowanym językiem, zoptymalizowanym do wyszukiwania wszystkich danych typu XML. XQuery pozwala na uruchamianie kwerend wyszukujących zmienne i kolumny typu XML za pomocą później przypisanych metod. W3C nadzoruje rozwój XQuery, podobnie jak wiele

innych standardów XML. XQuery wywodzi się z języka zapytań Quilt, który był oparty na różnych innych językach zapytań, takich jak XML Path Language (XPath) wersja 1.0, XML Query Language (XQL) i SQL. Podzbiorem XQuery jest XPath 2.0. Osoba, która posiada doświadczenie w używaniu XPath 1.0, może zapoznać się z właściwościami XQuery, bez konieczności nauki nowego języka zapytań, tym niemniej jednak pojawiły się znaczące rozszerzenia w stosunku do XPath 1.0, takie jak kontrola typów, specjalne funkcje i lepsza obsługa iteracji, możliwość sortowania rezultatów oraz zmiany w konstrukcji.

SQL Server 2005 bezpośrednio pozwala na korzystanie z XQuery, dzięki któremu obiekty XML mogą operować w warstwie danych. Platforma zapewnia obsługę statycznie zdefiniowanego podzbioru XQuery Working Draft z 15 listopada 2003 r.

Rozwinięcia DML

Obecnie specyfikacja XQuery zawiera składnię i semantykę zapytań, ale nie uwzględnia modyfikowania dokumentów XML. Polecenia manipulacji danymi (DML) XML stanowią rozszerzenie funkcji XQuery i służą do modyfikacji danych. SQL Server 2005 dodaje trzy słowa kluczowe: `insert`, `update` i `delete`. Każde z tych słów jest wykorzystywane wewnątrz metody `modify()` typu danych XML.

Native XML Web Services

W poprzednich wersjach SQL Server można było rozszerzyć możliwości platformy o przetwarzanie żądań SOAP, korzystając z aplikacji środkowej warstwy nazwanej SQLXML. SQLXML został wprowadzony jako Web Services Toolkit w 2001 roku. Zawierał proste środki do automatycznego wykorzystania poleceń Web Service Description Language (WSDL) poprzez użycie kreatora usług sieciowych. Aby wykorzystywać SQLXML Web Service, koniecznym jest posiadanie serwera z zainstalowanym IIS 5.0 (ang. *Internet Information Services*). IIS funkcjonował jako kanał dla usług sieciowych, łączący klienta i serwer. Funkcjonalnośc SQLXML pozwalała na przekształcanie żądań SOAP do zapytań transakcyjnego SQL z wykorzystaniem OLE DB. W SQL Server 2005 baza danych sama w sobie jest dostawcą usług sieciowych. Platforma SQL Server 2005 może teraz przetwarzać żądania SOAP bez pośrednictwa komponentu środkowej warstwy. Obecność zainstalowanego IIS również jest niekonieczna, choć zwiększa stopień bezpieczeństwa, ułatwiając jednocześnie korzystanie z tej funkcjonalności. Przypomnę jeszcze, że podczas instalacji SQL Server 2005 funkcjonalność usług sieciowych jest uwzględniana tylko wtedy, kiedy system operacyjny należy do grupy serwerów Windows 2003. SQL Server obsługuje sterownik *http.sys* jądra systemu.

Konfigurując SQL Server dla usług sieciowych, aby zapewnić możliwość natywnego nasłuchu żądań HTTP, należy pamiętać, że SOAP wymaga utworzenia punktu końcowego HTTP i zdefiniowania metod, które ten punkt obsłużą. Podczas tworzenia punktu końcowego HTTP trzeba również określić niepowtarzalny adres URL (ang. *Unified Resource Locator*), niezbędny podczas nasłuchiwania przychodzących żądań HTTP. Na przykład po utworzeniu punktu końcowego z adresem URL *http://servername/sql*, żądania SOAP, przesyłane do *http://servername/sql* będą obsługiwane przez *http.sys*. *http.sys* przekazuje żądania SOAP do instancji SQL Server, który obsługuje punkt końcowy związany z URL. Od tego momentu wszystkie żądania są obsługiwane przez warstwę obsługi SOAP wewnątrz platformy SQL Server.

Instancja SQL Server może mieć różne punkty końcowe. Każdy z nich może obsługiwać dowolną liczbę procedur składowanych (zaimplementowanych za pomocą transakcyjnego SQL lub CLR) jako WebMethod punktu końcowego i może być wywoływany przez SOAP poprzez zdalne wywoływanie procedur (RPC). Metoda sieciowa może mieć inną nazwę niż właśnie obsługiwana procedura składowana. Nazwa metody sieciowej jest widziana przez użytkownika jako nazwa operacyjna w WSDL.

Punkt końcowy HTTP może zostać utworzony i administrowany za pośrednictwem poleceń DDL transakcyjnego SQL. Utworzenie punktu końcowego HTTP jest pierwszym krokiem w umożliwieniu dostępu do SQL Server 2005 za pomocą HTTP/SOAP. Każdy punkt końcowy ma nazwę i zestaw opcji, których kombinacja definiuje zachowanie tego punktu końcowego. Wszystkie punkty końcowe są przechowywane w bazie danych *master*, w widoku katalogu *master.sys.http_endpoints*. Punkt końcowy nie posiada metod SOAP dopóty, dopóki administrator ich nie zdefiniuje.

Punkty końcowe HTTP obsługują następujące standardy uwierzytelniania: Basic, Digest, Integrated (Windows NT LAN Manager, czyli NTLM, Kerberos) i Anonymous — SQL Auth. Najpierw autoryzacja następuje na poziomie protokołu HTTP. Jeśli zakończy się powodzeniem, identyfikator bezpieczeństwa użytkownika (SID) służy do uwierzytelniania na platformie SQL Server. Dzieje się tak we wszystkich przypadkach oprócz *ANONYMOUS*. Wybór *ANONYMOUS* w punkcie końcowym oznacza, że jest możliwy dostęp anonimowy na poziomie protokołu HTTP. Klient ciągle musi dostarczyć dane SQLAuth, żeby móc uzyskać dostęp do bazy danych. Wybór opcji *ANONYMOUS* oznacza, że jest wymagany dodatkowy nagłówek HTTP, żeby zaszyfrować nazwę użytkownika i hasło. Nagłówek ten musi się nazywać *MS-SQL Auth* i jego wartość musi zawierać poprawne dane użytkownika i hasło, zakodowane metodą base64. Administratorzy mogą także ustawiać restrykcje dotyczące adresu IP, ograniczając dostęp do punktu końcowego wyspecyfikowanym adresom IP lub zakresowi adresów IP. Pojęciowo „punkt końcowy" oznacza „aplikację". Wszystkie metody, które implementuje

aplikacja, są mapowane na punkt końcowy, więc zabezpieczenia zastosowane do punktu końcowego chronią dostęp do aplikacji. Punkty końcowe są bezpieczne z założenia. Poniżej przedstawiam kilka rad, dzięki którym ich ochrona może być łatwiejsza.

- Należy domyślnie wyłączyć tworzone punkty końcowe. Żadne punkty końcowe ani metody sieciowe nie powinny być domyślnie mapowane, należy je jawnie utworzyć i wyspecyfikować.
- Należy kontrolować zabezpieczenia obiektów. Mapowane procedury składowane mogą być uruchamiane tylko wtedy, kiedy użytkownik posiada uprawnienia do korzystania z połączenia z punktem końcowym i dodatkowo uprawnienia do procedury składowanej.
- Należy wyłączyć obsługę anonimowego dostępu do punktu końcowego. Wszystkie żądania, włączając w to żądania dla WSDL, są autoryzowane. Aby skorzystać z któregokolwiek żądania, klient musi się autentykować.

Po utworzeniu punktu końcowego tylko członkowie roli `sysadmin` i właściciel punktu końcowego mogą połączyć się z tym punktem. Jeśli inni użytkownicy mają otrzymać dostęp do tego punktu, trzeba im jawnie przyznać odpowiednie uprawnienia. W tym celu można posłużyć się następującym poleceniem:

```
GRANT CONNECT ON HTTP ENDPOINT::hello_world_endpoint
TO [DOMAIN\USER]
```

Klienci używający platformy innej niż systemy Microsoft, mogą połączyć się z SQL Server, korzystając z metody uwierzytelniania BASIC albo SQL Auth, jednakże użycie BASIC albo SQL Auth wymaga zabezpieczenia kanału połączenia, zatem takie połączenie jest możliwe tylko za pomocą portów obsługujących SSL.

WSDL

WSDL jest dokumentem napisanym w XML, zawierającym specyfikację usług sieciowych. Opisuje położenie danej usługi i operacje (lub metody) obsługiwane przez daną usługę. WSDL dostarcza klientom informacje niezbędne do korzystania z usług sieciowych. Takie narzędzia, jak Visual Studio 2005 i JBuilder wykorzystują WDSL do generowania kodu pośredniego, który aplikacja klienta może użyć do komunikacji z usługą sieciową. Jeżeli punkt końcowy obsługuje WSDL, to przetwarza go w chwili, kiedy otrzymuje do niego żądanie. Serwer odpytuje metadane skojarzone z punktem końcowym i dynamicznie generuje WSDL. Wygenerowany WDSL oferuje wiele informacji dotyczących parametrów procedury składowanej. Serwer może wygenerować dwa różne rodzaje dokumentów WSDL — prosty lub złożony — zależnie od tego, czy korzysta z podstawowych, czy ze złożonych typów XSD. Typy XSD służą do opisu parametrów w komunikacie żądania lub odpowiedzi. Domyślnie wykorzystuje się typy złożone.

SQL Server jako platforma serwera aplikacji

Jednym z krytycznie ważnych obszarów, jakiemu Microsoft poświęcił dużą uwagę podczas tworzenia SQL Server 2005, jest pełnienie przez tę platformę funkcji serwera aplikacji. Microsoft nazwał to Service Oriented Database Application (SODA), co można przetłumaczyć jako *zorientowana na usługi aplikacja bazodanowa*. Użytkownicy wykorzystują bazy danych w coraz bardziej kreatywny sposób, więc systemy zarządzania relacyjnymi bazami danych (RDBMS), które zachowują się jak serwer aplikacji bazodanowych, są bardzo sensownym rozwiązaniem. W rzeczywistości twierdzenie, że SQL Server jest dostarczany z usługami takimi, jak Reporting Services, Notification Services, SQL Service Broker i usługi sieciowe XML, oznacza, że warto mieć dane w bazie danych, a logikę biznesową umieścić blisko danych.

W tabeli 4.5 przedstawiłem możliwości implementacyjne poszczególnych usług platformy SQL Server, jako punkt wyjścia przyjmując architekturę aplikacji. Z punktu widzenia architektury, warstwa danych może stanowić jedno repozytorium dla wszystkich danych. W takim scenariuszu SQL Server jest klasycznym systemem zarządzania relacyjnymi bazami danych (RDBMS). Odmienna sytuacja wystąpi, jeśli aplikacja ma wykorzystywać usługi sieciowe XML. Pytanie zatem brzmi: w jaki sposób zaprojektować aplikację i warstwy danych z użyciem tych samych głównych technologii? Microsoft zastosował wszystkie praktyki projektowe podczas budowania aplikacji, począwszy od programowania zorientowanego obiektowo (OOP), aż po architekturę zorientowaną na usługi (SOA). Jeśli więc Czytelnik przeanalizuje możliwości platformy SQL Server z różnych punktów widzenia, praktyk projektowania, wdrażania rozwiązań architektonicznych, z pewnością zauważy, że jest to bardzo wszechstronny i elastyczny produkt.

Tabela 4.5. *Serwery aplikacji w SQL Server 2005*

Komponent	Warstwa danych	Druga warstwa
Notification Services	Źródłowa baza danych	Silnik powiadomień
Reporting Services	Źródłowa baza danych	Katalog raportów
SQL Service Broker	Umieszczony w SQL Server	brak
Usługi sieciowe XML	Umieszczony w SQL Server	Środkowa warstwa przez SQLXML
Replikacja	Wydawca w SQL Server	Subskrybent, prawdopodobnie mobilny

SQL Server może być uważany za komponent SOA z obsługą usług sieciowych XML. Mimo że pojęcia *usługi sieciowe XML* i *SOA* nie są synonimami, większość ludzi poprawnie bądź niepoprawnie całościowo wiąże SOA z usługami sieciowymi. Patrząc od strony architektury komponentów, można dostrzec nowe właściwości SQL Server 2005, gdyż wykorzystywanie bazodanowych komponentów .NET

pozwala na utworzenie kompletnego systemu. Można to uznać za klasyczne podejście obiektowe (OOP), zresztą systemy RDBMS wykazują wiele podobieństw do systemów zorientowanych obiektowo. Na przykład klasa w modelu obiektowym przypomina relację w relacyjnej bazie danych. Rekord jest podobny do obiektu instancji, a kolumna — do atrybutu. Punktem wyjścia tych podobieństw jest obszar metod. W przypadku RDBMS procedury składowane opisują proces biznesowy jako element działającego systemu. W podejściu obiektowym (OOP) metoda jest funkcjonalną częścią klasy. Oczywiście SQL Server 2005 obsługuje funkcjonalność większą od zwykłego przechowywania czy pozyskiwania danych.

Planując wykorzystanie platformy SQL Server 2005 w systemie informatycznym firmy, należy uwzględnić właściwości tego produktu, zwłaszcza te związane z architekturą, wydajnością i skalowalnością. Szczególnie istotny jest sposób zaimplementowania komponentów i usług. Kiedy na przykład trzeba zbudować skalowalny serwer aplikacji, osobne instalowanie komponentów będzie możliwe przede wszystkim w przypadku Notification Services i SQL Server Reporting Services. Mechanizm replikacji i SQL Server Broker wymagają zainstalowania podstawowej części serwera, gdyż są funkcjonalnie związane z bazą danych, natomiast usługi Notification Services i Reporting Services wymagają SQL Server jako miejsce przechowywania metadanych, jednak metadane te mogą być pobierane z innego serwera.

Ktoś, kto dokładnie przeanalizuje szczegóły struktury platformy SQL Server, może stwierdzić, że wiele funkcji służy również do implementowania innych funkcji. Na przykład Reporting Services i Analysis Services korzystają z usług sieciowych XML w celu uzyskiwania dostępu do danych, przechowywania metadanych i dostarczania modelu programistycznego. Usługi Notification Services bezpośrednio nie wykorzystują usług sieciowych ani nie przypisują orientacji na ich podstawie, ale stosują XML jako format tworzenia metadanych. SQL Service Broker (SSB) jest w rzeczywistości specjalną blokadą, zaimplementowaną jako część silnika SQL Server Engine przy projektowaniu aplikacji SSB — końcowy użytkownik jest informowany o wykorzystaniu usług sieciowych XML podczas projektowania aplikacji. SQL Server Broker stanowi znakomity przykład sposobu, w jaki Microsoft implementuje architekturę modularną w serwerze bazodanowym, który jednocześnie jest środowiskiem uruchomieniowym i implementacją języka.

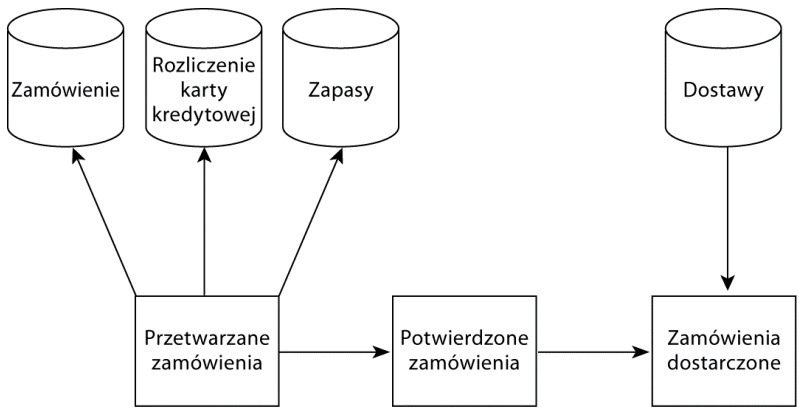
SQL Service Broker

SQL Service Broker (SSB) jest nowym elementem, pozwalającym wewnętrznym lub zewnętrznym procesom na wysyłanie lub otrzymywanie asynchronicznych komunikatów. SQL Service Broker jest wbudowany w silnik SQL Server i pozwala

na przesyłanie komunikatów w ramach jednego serwera, ale także pomiędzy serwerami. Przesyłanie wiadomości jest realizowane za pośrednictwem rozszerzeń poleceń DML transakcyjnego SQL.

Po co przesyłać komunikaty?

Kolejkowanie jest bardzo ciekawym mechanizmem, który umożliwia elastyczne rozkładanie obciążenia pomiędzy poszczególne elementy systemu. Zastosowanie mechanizmu kolejkowania może pozwolić na uzyskanie większej skalowalności i wydajności systemu. Poniżej znajduje się prosty przykład. Na rysunku 4.1 przedstawiłem schemat budowy aplikacji do handlu elektronicznego.



Rysunek 4.1. *Przykładowy scenariusz działania usługi Service Broker*

Z analizy rysunku 4.1 wynika, że niektóre procesy powinny zachodzić równocześnie: po wprowadzeniu do bazy danych informacji o zamówieniu, następuje proces autoryzacji karty kredytowej. Równocześnie należałoby sprawdzić stan zapasów, jednak czynności te zostaną wykonane asynchronicznie. Generowanie listy zamówienia i określenie szczegółów transportu towaru może nastąpić asynchronicznie, zwłaszcza że informacje dotyczące transportu niekiedy wymagają korekty zamówienia. Możliwość wykorzystania mechanizmu przesyłania komunikatów spowoduje zwiększenie równoległości.

Przede wszystkim jednak trzeba zrozumieć pięć kluczowych koncepcji SQL Server Broker.

- **Kolejkowanie.** Kolejkowanie realizowane przez SQL Service Broker polega na tworzeniu luźnego połączenia pomiędzy nadawcą a odbiorcą komunikatu. Nadawca umieszcza komunikat w kolejce i kontynuuje pracę z aplikacją. SQL Service Broker zapewnia transmisję wiadomości do miejsca przeznaczenia.

- **Dialogi.** SQL Service Broker korzysta z mechanizmu dialogów. Dialog jest dwukierunkowym strumieniem komunikatów pomiędzy dwiema bazami danych lub kolejkami. Wszystkie komunikaty w dialogu są uporządkowane i zawsze dostarczane w kolejności wysłania. Sposób uporządkowania komunikatów zależy od transakcji, wątków wejściowych i wyjściowych, od awarii i restartów systemu. Niektóre komunikaty systemowe gwarantują kolejność wiadomości w pojedynczej transakcji, a nie wielu transakcjach. W tym względzie SQL Service Broker działa w sposób jednoznaczny: zapewnia, że każdy komunikat posiada identyfikator, jednoznacznie określający dialog z nim związany. Ten identyfikator zasadniczo stanowi blokadę transakcji, dopóki proces dla każdego dialogu nie zostanie zakończony.
- **Grupy konwersacji**. Grupa konwersacji jest grupą wszystkich dialogów związanych z pewnym zadaniem. W przykładzie aplikacji do handlu elektronicznego, wszystkie dialogi biorące udział w realizacji pojedynczego zamówienia są grupowane w pojedynczą instancję. Grupy konwersacji są w rzeczywistości jednoznacznym identyfikatorem, umieszczonym w każdym komunikacie związanym z danym zamówieniem. Zwykle grupy konwersacji stosuje się do obsługi stanu związanego z określonym procesem, na przykład jeśli taki proces wymaga więcej czasu dla przetwarzania komunikatu. Taka sytuacja może się przytrafić w przypadku klienta składającego zamówienie, gdy dostarczenie i przetwarzanie autoryzacji karty kredytowej opóźnia się. Wtedy właśnie grupy konwersacji i stan zamówienia jest przechowywany w bazie danych do momentu wysłania kolejnego komunikatu i zmiany stanu związanego z zamówieniem.
- **Aktywacja.** Funkcje SSB pozwalają na określenie procedury składowanej do obsługi dostarczenia komunikatu dla pewnych usług. Po otrzymaniu komunikatu SSB sprawdza, czy procedura składowana funkcjonuje i czy może przetworzyć komunikat. Jeśli potrzebna procedura nie jest aktywna, SSB uruchamia ją. Procedura składowana pozostaje aktywna do momentu opróżnienia kolejki. Jeżeli komunikaty przychodzą zbyt szybko, aby pojedyncza instancja procedury mogła je obsłużyć, SSB uruchamia dodatkowe instancje, dopóki żądania nie są przetwarzane płynnie. Opisywany mechanizm można skonfigurować. Warto podkreślić, że gwarantuje on dostępność odpowiedniej ilości zasobów.
- **Przesyłanie komunikatów.** Komunikaty są przesyłane przez dwa protokoły, podobne do protokołu TCP/IP. Pierwszy z nich, protokół Binary Adjacent Broker (BAB) jest wysoko wydajnym interfejsem protokołu TCP/IP. BAB obsługuje podstawowe przesyłanie komunikatów. Drugim protokołem jest Dialog, łączący dwa punkty końcowe dostarczania wiadomości. Obsługuje porządkowanie i potwierdzanie wysłania komunikatu. Obsługuje także komunikację w przypadku błędów dostarczenia i jest odpowiedzialny za bezpieczeństwo komunikatów.

Po co są potrzebne komunikaty w bazie danych?

Istnieje wiele istotnych powodów, dla których warto korzystać z mechanizmu komunikatów na poziomie bazy danych. Najważniejszymi cechami charakteryzującymi aplikacje wysyłające komunikaty są:

- możliwość porządkowania komunikatów;
- zgodność transakcyjna;
- złożoność;
- aplikacje te nie biorą udziału w mechanizmach zarządzania bazą danych.

SQL Service Broker obsługuje transakcyjne przesyłanie komunikatów (ang. *transaction messaging*). Termin „transakcyjne" ma to samo znaczenie, co w przypadku transakcji realizowanych przez typowe bazy danych. Transakcja jest zbiorem działań, traktowanym jak jedno działanie. Kiedy transakcji nie udaje się wykonać do końca, wysłane i odebrane komunikaty są odwoływane i nie przynoszą skutku, dopóki transakcja nie zostanie potwierdzona.

Transakcyjne przesyłanie komunikatów ułatwia programowanie aplikacji przesyłających wiadomości. Nawet jeśli zostanie wydane polecenie wysłania czy odbioru komunikatu, wykonanie polecenia nastąpi dopiero po potwierdzeniu transakcji. W przypadku wycofania transakcji komunikaty nie są wysyłane, natomiast odebrane wiadomości wędrują do kolejki w celu dalszego przetworzenia.

Jak już wcześniej zaznaczyłem, SQL Service Broker może rozwiązać kilka trudnych problemów, z jakimi borykają się aplikacje wysyłające komunikaty. SSB zapewnia, że porządkowanie komunikatów jest przezroczyste. Komunikaty są odbierane zgodnie z kolejnością wysłania, bez zaburzania sekwencji, a ich koordynowanie zachodzi dzięki dialogom i identyfikatorom grup konwersacji. Grupy konwersacji ułatwiają zachowanie porządku i przypisywanie konkretnych komunikatów poszczególnym żądaniom. W przypadku typowych aplikacjach komunikacyjnych zapewnienie wielowątkowego działania jest trudnym zagadnieniem. Zastosowanie SSB pozwala na zablokowanie przesyłania komunikatów do danego wątku tak długo, dopóki transakcja nie zostanie zakończona. W ten sposób można rozwiązać problem naruszania integralności referencyjnej.

SSB automatycznie obsługuje obciążenie związane z przetwarzaniem komunikatów oczekujących w kolejce, tak więc istnieje możliwość dopasowania liczby procesów przetwarzających informacje. Z punktu widzenia administratora danych, SSB posiada wszystkie zalety aplikacji do zarządzania bazami danych. SSB jest również istotny dla ustawienia mechanizmu służącego do sporządzania kopii zapasowych, dla wykorzystywania rozwiązań wysokiej dostępności i magazynowania dyskowego. SSB posiada też zalety procedur składowanych, XML i usług

sieciowych. Może otrzymywać komunikaty z systemów znajdujących się poza bazą danych, takich jak handlowy punkt dostępu, który obsługuje karty kredytowe za pośrednictwem wywołań HTTPS.

Bezpieczeństwo SSB

Przy wszystkich swoich zaletach, SQL Service Broker wymaga szczególnego poziomu zabezpieczeń. Model SQL Service Broker jest oparty na modelu komunikacyjnym. SQL Service Broker charakteryzuje się zaletami nowego mechanizmu certyfikatów, stanowiącego element SQL Server 2005. SQL Service Broker jest chroniony na dwa sposoby.

- **Bezpieczny dialog** zapewnia szyfrowanie poszczególnych komunikatów w dialogu i weryfikację tożsamości jego uczestników. Bezpieczny dialog obsługuje także zdalną autoryzację i sprawdzanie integralności komunikatu. Bezpieczny dialog ustala autentykowaną i szyfrowaną komunikację pomiędzy dwiema usługami.
- **Bezpieczny transport** zapewnia ochronę przed wysyłaniem komunikatów SQL Service Broker pochodzących z nieautoryzowanych baz danych do bazy danych w lokalnej instancji. Bezpieczny transport umożliwia ustalenie autentykowanego sieciowego połączenia pomiędzy dwiema bazami danych.

SQL Service Broker korzysta z certyfikatów do uwierzytelniania komunikacji pomiędzy lokalnym a zdalnym serwerem. Przypomnijmy sobie wcześniejszą dyskusję o założeniach autoryzacji i uwierzytelniania: drugi ID, znajdujący się w czteroczęściowej nazwie, może zawierać certyfikat. Certyfikat zapewnia mechanizm zaufania, który mapuje użytkownika lub nazwę logowania do schematu, zawierającego funkcjonalność instancji SQL Service Broker. Zaletą certyfikatu jest to, że nie tworzy żadnego łańcucha powiązań własności.

SQL Service Broker nie stanowi panaceum na problemy wydajnościowe związane ze skalowalnością aplikacji. Jest jednak przydatny w przypadku aplikacji z problemami skalowalności, wynikającymi z wystąpienia tzw. wąskiego gardła w procesach zaliczanych do logiki biznesowej. Przykładem takiej sytuacji jest oczekiwanie na odpowiedź systemu. Innym przykładem jest system słabo powiązany, który wymaga odpowiedzi od zewnętrznych nadawców komunikatów, takich jak usługa sieciowa obsługująca wysyłkę lub moduł handlowy obsługujący karty kredytowe. W takich przypadkach SQL Service Broker może być dogodnym rozwiązaniem.

Notification Services

Twórcy SQL Server 2005 wprowadzili znaczne ulepszenia do usługi Notification Services (NS).

Instancja usługi aplikacji Notification Services jest dostępna poprzez SQL Computer Management. Ułatwia to wyłączanie usługi Notification Services w przypadku problemów z aplikacją. Pliki konfiguracyjne usługi Notification Services w dalszym ciągu można tworzyć za pomocą Visual Studio, jednak teraz cały proces można przeprowadzić za pomocą programu SQL Server Management Studio. Aby zbudować aplikację opartą na usłudze NS, trzeba przeanalizować podstawowe elementy tej usługi. Platforma Notification Services składa się z dwóch komponentów:

- Prostego, ale potężnego programistycznego środowiska uruchomieniowego dla aplikacji powiadamiających. Środowisko programistyczne jest oparte na XML i języku SQL. Funkcjonuje wewnątrz bazy danych SQL Server.
- Niezawodny, wysokiej jakości, skalowalny serwer, który uruchamia aplikacje powiadamiające. Serwer jest zbudowany na platformie .NET Framework i SQL Server. Serwer działa w systemach operacyjnych Windows Server 2000 i 2003 jako usługa. Usługa Notification Services została zaprojektowana jako rozszerzenie, pozostające w ścisłym związku z istniejącymi aplikacjami.

Platforma SQL Server Notification Services składa się z kilku komponentów:

- **NSService.exe.** Instancja Notification Services uruchomiona jako usługa Windows. Ta usługa jest związana z SQL Computer Manager i zachowuje się jak silnik dla aplikacji Notification Services.
- **NSControl.exe.** Ten program generuje bazę SQL Server i inne obiekty wykorzystywane przez NSService i aplikację subskrypcji. Aplikacja przechowuje metadane o instancjach NS, które są zapisywane w rejestrze. Żeby włączyć lub wyłączyć pewne części aplikacji NS, można skorzystać z NSControl,
- **Microsoft.SQLServer.NotificationServies.dll-NSService.exe.** Ładuje zestaw, który zawiera kod dla SQLNS, wbudowanych dostawców, a także komponenty wywoływane do przetwarzania danych w instancjach aplikacji NS.

Z usługą Notification Services są związane następujące pojęcia:

- **Zdarzenia.** W odniesieniu do aplikacji powiadamiających, zdarzenie definiujemy jako komunikat odebrany przez proces nasłuchu. Komunikaty zdarzeń są przechowywane w tabeli bazy danych aplikacji powiadamiającej.

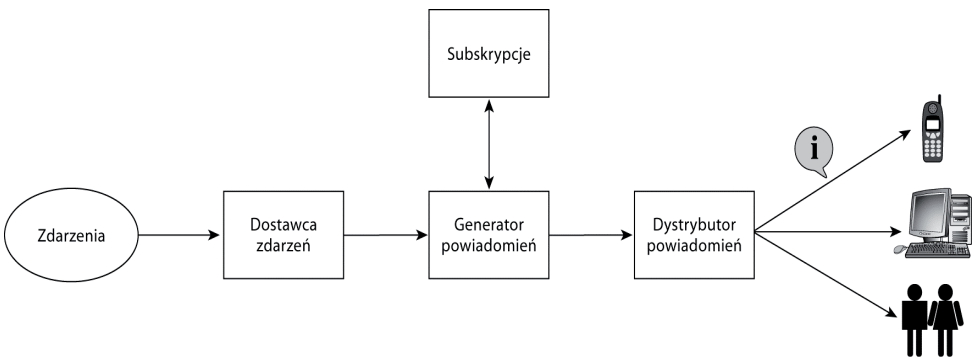
- **Subskrypcja.** Subskrypcje określają sposób, w jaki użytkownicy korzystają z aplikacji NS, wskazując, co ich interesuje.
- **Powiadomienia.** Komunikaty wysyłane do użytkownika. Zawierają zasubskrybowane przez użytkownika informacje o zdarzeniach.

Scenariusze wdrożeń usługi Notification Services

Istnieją trzy standardowe scenariusze wdrażania usługi Notification Service. Wybór konkretnego sposobu postępowania zależy od wielkości i poziomu usługi aplikacji powiadamiania. W niewielkim, opartym na intranecie środowisku platforma SQL Server i usługa Notification Services mogą zostać wdrożone na jednym serwerze. Jeśli zaś chodzi o sprzęt, trzeba zapewnić co najmniej dwa procesory. Mimo to zastosowanie 64-bitowych procesorów prawdopodobnie jest nieopłacalne, ponieważ parametry procesora i pamięci RAM w obecnie dostępnych modelach serwerów są najzupełniej wystarczające dla małych wdrożeń.

W przypadku średniej wielkości wdrożeń, gdzie aplikacja powiadamiająca będzie użytkowana poprzez internet, konieczne są co najmniej dwie warstwy. I ponownie, po przeprowadzeniu analizy obciążenia można dojść do wniosku, że umieszczenie silnika powiadamiania i aplikacji zarządzania subskrypcjami powiadomień na tej samej warstwie jest sensownym rozwiązaniem. Pozostawienie bazy danych na niższej warstwie pozwoli na łatwiejsze zarządzanie tą bazą. Druga warstwa jest warstwą przechowywania danych, zatem powinna być zarządzana przez administratora Windows.

Na rysunku 4.2 przedstawiono podstawowe procesy aplikacji Notification Services.



Rysunek 4.2. *Przepływ informacji w aplikacji opartej na Notification Services*

Aplikacja służąca do zarządzania subskrypcjami jest budowana jako część rozwijającej się usługi powiadamiania. Aplikacja ta nie jest wewnętrznym komponentem. Ma to niebagatelne znaczenie, ponieważ tworzy interfejs użytkownika, taki jak strona WWW, gdzie użytkownicy mogą subskrybować bądź odwoływać subskrypcję:

- **Dostawcy zdarzeń** rejestrują zdarzenia i dostarczają dane do usługi Notification Services. Aplikacja może wykorzystywać standardowego dostawcę zdarzeń, pochodzącego z usługi Notification Services. Równie dobrze projektant może utworzyć własnego dostawcę.
- **Generator powiadamiania** przetwarza subskrypcje i tworzy notyfikacje oparte na bieżącym zestawie zdarzeń. Projektant aplikacji definiuje operacje generatora powiadamiania, dostarczając parametry konfiguracyjne i zasady subskrypcji.
- **Dystrybutor powiadamiania** formatuje powiadomienia i wysyła je do subskrybenta poprzez usługę dostarczania. Projektant aplikacji definiuje przekształcenie czystych danych na postać sformatowanego powiadomienia, definiując funkcję powiadomienia i argumenty tej funkcji.

Usługa SQL Server Notification Services zapewnia deklaratywny model XML Expressing Notification Applications. Projektant, budując aplikację powiadamiania, tworzy plik definicji aplikacji (ang. *application definition file* — ADF) oraz plik konfiguracyjny, żeby dokładnie opisać aplikację. ADF opisuje, które zdarzenia są gromadzone, w jaki sposób zarządzane i analizowane są subskrypcje oraz w jaki sposób wynikowe powiadomienia są formatowane i dostarczane. Po zdefiniowaniu aplikacji powiadamiania system Notification Services sprawdza poprawność ADF i plików konfiguracyjnych, tworzy bazę danych SQL Server do przechowywania danych aplikacji i konfiguruje usługę powiadamiania systemu Windows.

Zarządzanie subskrypcjami. Interfejs API do zarządzania subskrybentami

Usługa SQL Server Notification Services zawiera programistyczny interfejs aplikacji (API) `Subscriber Management` służący do zarządzania informacjami o subskrybencie. Za pomocą klas `Subscriber` i `SubscriberEnumeration` można dodawać, uaktualniać i usuwać te informacje. Omawiane klasy są dostępne w przestrzeni nazewniczej `Microsoft.SqlServer.NotificationServices`.

- **Klasa `Subscriber`** wykorzystuje dane subskrybenta, które są zazwyczaj w niej umieszczane za pomocą interfejsu użytkownika aplikacji zarządzającej subskrypcjami. Równocześnie obiekt `Subscriber` tworzy lub uaktualnia pojedynczy rekord `Subscriber`.

- **Klasa `SubscriberEnumeration`** dostarcza odniesienie do zbioru subskrybentów w instancji Notification Services. Aby wybrać właściwy rekord `Subscriber`, należy podać ID subskrybenta od indeksera obiektu `SubscriberEnumeration`.

Zarządzanie subskrypcjami. Interfejs API do zarządzania urządzeniem subskrybenta

Usługa SQL Server Notification Services zawiera programistyczny interfejs aplikacji `SubscriberDevice`, przeznaczony do zarządzania urządzeniami subskrybenta. Projektanci mogą dodawać, uaktualniać i usuwać urządzenia, na których subskrybenci otrzymują notyfikacje, korzystając z klas `SubscriberDevice` i `SubscriberDeviceEnumeration`. Te klasy są dostępne w przestrzeni nazw `Microsoft.SqlServer.NotificationServices`.

- **Klasa `SubscriberDevice`**, które wykorzystuje dane urządzenia subskrybenta, które zazwyczaj są dostarczane do tej klasy poprzez interfejs użytkownika aplikacji zarządzającej subskrypcjami.
- **Klasa `SubscriberDeviceEnumeration`**, która dostarcza odniesienie do zbioru urządzeń subskrybenta w instancji Notification Services. Aby wybrać właściwy rekord urządzenia subskrybenta, należy podać nazwę urządzenia do indeksera obiektu `SubscriberDeviceEnumeration`.

Zdarzenia: FileSystemWatcherProvider

Usługa SQL Server Notification Services jest dostarczana ze standardowymi dostawcami zdarzeń, co ma ułatwić sprawne zaprojektowanie i wdrożenie aplikacji opartych na tej usłudze. Jednym z tych dostawców jest *File System Watcher Provider*, zaimplementowany jako część biblioteki dołączanej dynamicznie (DLL) o nazwie *Microsoft.SqlServer.NotificationServices.dll*. Dostawca ten korzysta z klasy `FileSystemWatcher` zdefiniowanej w środowisku .NET Framework, służącej do monitorowania dodatkowych plików w określonym folderze.

Po dodaniu pliku zdarzenia XML do katalogu dostawca jest wywoływany do załadowania pliku zdarzenia XML do pamięci, a następnie korzysta z dostarczonego obiektu `EventLoader`, żeby zapisać informacje o zdarzeniu do tabeli zdarzeń.

Zdarzenia: dostawca zdarzeń SQL Server

Dostawca zdarzeń (SQL Server Event Provider) wykorzystuje procedury składowane do gromadzenia danych o nowych zdarzeniach i dostarcza dane bezpośrednio do tabeli zdarzeń.

Zdarzenia: niestandardowi dostawcy zdarzeń, EventLoaderClass

Usługi SQL Server Notification Services wykorzystują dostawców zdarzeń do gromadzenia danych i przekazywania ich do systemu Notification Services. Dostawca zdarzeń jest zaimplementowany jako biblioteka DLL, która monitoruje źródło zdarzeń i zbiera informacje. Standardowi dostawcy zdarzeń są dołączani do usługi Notification Services, co ma ułatwić szybkie zaprojektowanie i wdrożenie aplikacji powiadamiającej. W przypadku, gdy aplikacja wymaga gromadzenia danych ze źródeł nieposiadających standardowych dostawców zdarzeń zaprojektowanych do monitorowania, trzeba samodzielnie utworzyć dostawcę zdarzeń, który sprosta wymaganiom określonej aplikacji.

Tacy zaprojektowani do konkretnych celów dostawcy zdarzeń mogą korzystać z interfejsu programistycznego aplikacji EventLoader do pobierania dokumentów XML lub strumieni danych jako informacji wejściowych. Metoda `LoadXml` służy wtedy do zapisywania zdarzeń związanych z dokumentem lub strumieniem do tabeli zdarzeń. Projektanci powinni korzystać z interfejsu EventLoader, pracując z wejściowymi danymi źródłowymi XML. Zdarzenia pochodzące z różnych źródeł mogą się znajdować w pliku XML lub strumieniu, pozwalając na wsadowe ich umieszczanie w tabeli zdarzeń.

Zdarzenia: procedury składowane SQL Server

Usługi SQL Server Notification Services pozwalają na to, aby podczas pracy ze źródłem zdarzeń SQL Server korzystać ze standardowych procedur składowanych rejestrujących zdarzenia. Można też wykorzystać zalety logiki biznesowej zdefiniowanej w innej procedurze składowanej SQL Server. Poniżej wymieniłem standardowe procedury składowane rejestrujące zdarzenia.

- **`NSEventBeginBatch`*`EventClassName`*** — zapewnia identyfikator przetwarzania zdarzenia, który powinien być stosowany dla tworzonych zdarzeń.
- **`NSEventWrite`*`EventClassName`*** — wstawia pojedynczy rekord zdarzenia do tabeli zdarzeń.
- **`NSEventFlushBatch`*`EventClassName`*** — potwierdza lub odrzuca grupę zdarzeń.
- **`NSEventSubmitBatch`*`EventClassName`*** — zapewnia izolację funkcjonalną wcześniej opisanych procedur składowanych. Tworzy nową grupę zdarzeń, przypisuje jedno lub więcej zdarzeń do nowej grupy zdarzeń, zapisuje rekordy do tabeli zdarzeń i uruchamia grupę zdarzeń.

Generowanie powiadomień: zarządzanie stanem aplikacji powiadamiania (kroniki)

Usługa SQL Server Notification Services zawsze przetwarza zdarzenia jako część grupy zdarzeń. Grupa jest wykorzystywana przez proces generowania powiadomienia tylko jeden raz, po czym grupa zdarzeń jest traktowana jako wygasła i jest później usuwana z systemu. W niektórych przypadkach dane zdarzeń mogą być przechowywane dla zaplanowanych subskrypcji lub innych wymagań aplikacji. Do stałego utrzymywania informacji o zdarzeniach w systemie Notification Services może służyć tabela kronik zdarzeń. Przykładowo, projektant może korzystać z tabeli kroniki zdarzeń w celu zapamiętania danych o zdarzeniu dla zaplanowanych subskrypcji, uaktualniając tabelę danymi bieżącego zdarzenia zawsze podczas przetwarzania grupy zdarzeń. Można także wykorzystać tabelę kroniki zdarzeń do przechowywania informacji o zakresach lub historii zdarzeń, co ma zapobiec duplikowaniu powiadomień.

Generowanie powiadomień: obsługa odtwarzania po awarii

Istotnym pojęciem, związanym z funkcjonowaniem usług Notification Services, jest *quantum*. Słowo to oznacza odstęp czasu, który określa częstotliwość wywoływania funkcji systemu. W przypadku usług SQL Server Notification Services praca, która powinna zostać wykonana w ciągu quantum, może zostać pominięta z powodu awarii lub przeciążenia systemu. Dzięki `ChronicleQuantumLimit` można ustalić optymalną równowagę pomiędzy szybkością aplikacji a poprawnością danych, cały czas uwzględniając specyfikę aplikacji.

Limity quantum mniejszych kronik wymagają od generatora mniejszej pracy w przypadku awarii, dzięki czemu odtworzenie po awarii jest szybsze.

Generowanie powiadomień: jednorazowe i planowane generowanie powiadomień

Usługa SQL Server Notification Services obsługuje dwie odmienne metody generowania powiadomień:

- **Subskrypcja zdarzeń** wywoływana natychmiast po nadejściu nowych zdarzeń;
- **Zaplanowana subskrypcja** oparta na planie zdefiniowanym przez subskrybenta.

Generowanie powiadomień: klasa TimeZone

Usługa SQL Server Notification Services pozwala na zaprojektowanie aplikacji służących do powiadamiania zależnych od strefy czasowej. Można wykorzystać klasę `TimeZone` do utworzenia aplikacji zarządzającej subskrypcjami, która obsłuży różne strefy czasowe. Użytkownicy aplikacji mogą wprowadzać swój lokalny czas podczas planowania subskrypcji.

Formatowanie powiadomień: formatowanie zawartości za pomocą XSLT

Usługa SQL Server Notification Services umożliwia standardowe formatowanie za pomocą **Extensible Stylesheets Language Transformation** (**XSLT**), co jest pomocne przy szybkim projektowaniu i wdrażaniu aplikacji opartych na usługach Notification Services. Narzędzie do formatowania zawartości pozwala na sprecyzowanie XLST do sformatowania czystych danych powiadomienia. Dzięki takiemu przekształceniu i wprowadzeniu wszelkich niezbędnych zmian w formacie dane są odpowiednio przygotowane do wyświetlenia na urządzeniu odbierającym. Narzędzie do formatowania utworzono w wersjach dla różnych języków (państw, regionów) oraz dla różnych typów urządzeń. W celu zapewnienia obsługi kombinacji danego języka i konkretnego typu urządzenia, obsługiwanego przez aplikację powiadamiającą, należy utworzyć oddzielny plik XSLT.

Formatowanie powiadomień: skrót powiadomienia

Usługa SQL Server Notification Services pozwala na zgrupowanie kilku indywidualnych powiadomień otrzymanych od tego samego subskrybenta w większą całość. Na przykład subskrybent może być zarejestrowany do odbioru kilku powiadomień dziennie. Zamiast oddzielnego wysyłania tych notyfikacji, system może je zgrupować, sumując ich zawartość. Takie sumowanie nazywa się dostarczaniem skrótowym.

Formatowanie powiadomień: multiemisja powiadomień

Usługa SQL Server Notification Services zawiera potężną funkcję grupowego dostarczania, która pozwala aplikacji powiadamiania na jednorazowe formatowanie danych i przesłanie ich do wielu subskrybentów. Na przykład wszyscy użytkownicy, którzy zasubskrybowali prognozę pogody dla Redmond w Washington, otrzymują takie same dane pogodowe w swoich komunikatach. W tych przypadkach formatowanie danych przeprowadza się po prostu raz i tak przygotowane dane mogą być wysłane do każdego subskrybenta.

Usługa SQL Server Notification Services dostarcza kilka standardowych protokołów, ułatwiających projektantom zdefiniowanie kanałów dostarczania dla instancji usług powiadamiania. Dokładniejsze informacje na ten temat zostaną przedstawione nieco dalej.

Dostarczanie powiadomień: protokół dostarczania plików

Projektanci mogą wykorzystywać protokół dostarczania plików do zapisywania danych powiadomienia w plikach tekstowych ASCII. Ten protokół jest zazwyczaj stosowany do uruchamiania i testowania aplikacji. Może być też użyty w aplikacjach dostarczających dane powiadomienia do systemowego katalogu plików, który potem jest przetwarzany przez File System Watcher Provider.

Dostarczanie powiadomień: protokół SMTP

Projektanci mogą wykorzystywać protokół SMTP do tworzenia i ustalania tras komunikatów dostarczanych przez MS Exchange lub inny system pocztowy oparty na protokole SMTP.

Dostarczanie powiadomień: rozszerzenia HTTP protokołu dostarczania

Rozszerzenia HTTP protokołu dostarczania pozwalają projektantom na tworzenie dodatków, które definiują pola nagłówków HTTP bez konieczności definiowania nowych kanałów dostarczania. Rozszerzenia HTTP protokołu dostarczania umożliwiają uwzględnianie mechanizmu powiadomień w usługach sieciowych lub aplikacjach WWW.

Dostarczanie powiadomień: SQL Service Broker

Protokół dostarczania SQL Service Broker pozwala projektantom na umieszczanie powiadomień w niezawodnej kolejce SQL Service Broker w celu dalszego przetwarzania albo określenia dalszej trasy.

Dostarczanie powiadomień: własne protokoły dostarczania

Jeżeli standardowe protokoły dostarczania nie odpowiadają specyfice aplikacji, projektanci mogą utworzyć własne protokoły dostarczania, korzystając z języków obsługiwanych przez .NET Framework.

SQL Server Mobile Edition

W tym podrozdziale zaprezentowałem opis podstaw architektury platformy SQL Server Mobile Edition oraz pewne rozważania na temat zarządzania i użytkowania serwera.

Platforma SQL Server Mobile Edition jest zintegrowana z Visual Studio. Warto wymienić kilka cech projektowych tej wersji produktu.

- Dzięki Server Explorer można rejestrować i wyliczać wersje Mobile Edition bazy danych. Można nawet przeglądać dane i zarządzać schematami, korzystając z Server Explorer.
- Umożliwiono korzystanie z techniki *przeciągnij-i-upuść*. Można na przykład przeciągać tabele i kolumny do okna roboczego, żeby utworzyć kontrolki związane z danymi.
- Mechanizm SmartTask pozwala na tworzenie głównych i szczegółowych widoków do wyświetlania danych z tabeli.
- Udostępniono obiekty biznesowe i usługi sieciowe wykorzystywane przez SQL Mobile Edition.
- Projektowanie end-to-end. Zaprojektowanie i wdrożenie aplikacji SQL Mobile Edition jest możliwe z wykorzystaniem pojedynczego projektu Visual Studio.
- Platforma SQL Mobile Edition wykorzystuje .NET Compact Framework dla protokołów dostępu do danych i interfejsów API.

Użytkownicy korzystający z SQL Mobile Edition z pewnością docenią możliwości tej platformy w zakresie zarządzania bazami danych. We wcześniejszych wersjach produktu administrowanie danymi było nieco trudniejsze, gdyż związek pomiędzy mobilną bazą danych a SQL Server nie był zbyt ścisły. SQL Server 2005 jest dużo bardziej zintegrowany. Teraz można zarządzać bazą danych SQL Server Mobile Edition na swoim osobistym komputerze lub na urządzeniu przenośnym, korzystając z programu SQL Server Management Studio. Aplikacja ta posiada wiele zalet. Poniżej przedstawiam najbardziej interesujące właściwości programu SQL Server Management Studio.

- Graficzny plan zapytań, który pozwala projektantom na łatwe analizowanie sposobu wykonania zapytań, modyfikowanie ich parametrów i zapoznawanie się z rezultatami dostrajania wydajności. Można przeglądać plany projektowanych i wykonanych zapytań.
- Nowy kreator subskrypcji, dzięki któremu tworzenie i późniejsza synchronizacja bazy danych subskrypcji nie wymaga dużego wysiłku.

- Integracja z usługą SQL Server Integration Services, co pozwala projektantom na importowanie i eksportowanie danych do i z różnych źródeł danych, włączając w to Microsoft Access, Oracle i IBM DB2.
- Możliwość wykorzystywania plików utworzonych za pomocą programu kopiującego Bulk (BCP). W ten sposób można zwiększyć skalowalność wstępnej synchronizacji i wydajności dla dynamicznie filtrowanych publikacji. W tym celu należy skorzystać z migawek, które zawierają plik, wstępnie utworzony za pomocą programu BCP.

W celu pełniejszego zaspokojenia potrzeb związanych z zarządzaniem bazami danych, Mobile Edition wykorzystuje nowe funkcje replikacji. W ten sposób poprawia się skalowalność baz danych Mobile Edition, szczególnie podczas ważnych faz inicjalizacji i synchronizacji danych. SQL Server 2005 zapewnia lepszą kontrolę dla połączonej replikacji. Poniżej wymieniono dwie nowe techniki.

- **Obsługa tylko pobranych (ang. *download-only*) artykułów (tabel).** Ta funkcja zmniejsza ilość metadanych transferowanych podczas synchronizacji i skraca czas przetwarzania danych na platformie SQL Server podczas wykonywania kolejnych synchronizacji.
- **Obsługa partycjonowanych (filtrowanych) artykułów.** Wiele aplikacji mobilnych (takich jak niektóre aplikacje GPS służące do nawigacji) filtruje dane w taki sposób, aby ich nie udostępniać innym urządzeniom. Dzięki mechanizmowi partycjonowania można znacznie zredukować czas synchronizacji i zwiększyć skalowalność aplikacji. Ta funkcja jest szczególnie przydatna dla dużych wdrożeń mobilnych urządzeń.

UWAGA: SQL Server Mobile Edition obsługuje systemy operacyjne Pocket PC 2002/3/4 i SmartPhone.

Ustawienia językowe w SQL Server 2005

Szacuje się, że jedna czwarta populacji ludzkiej posługuje się językami, których zapis wymaga użycia złożonego alfabetu. Można tu wymienić takie języki jak tajski, arabski, hebrajski, wietnamski, urdu, hinduski, tamilski, jidysz czy birmański. SQL Server 2005 obsługuje skomplikowane alfabety i dwukierunkowość pisania tekstu.

Alfabet określa się jako złożony, kiedy jest konieczna dodatkowa obsługa kodowania i wyświetlania znaków (w odniesieniu do wymogów stawianych przez alfabet łaciński). Oto kilka cech złożonego alfabetu:

- **Dwukierunkowość tekstu.** Ma miejsce wówczas, kiedy tekst pisany od prawej strony do lewej, tak jak jest w języku arabskim lub hebrajskim, jest łączony z tekstem, który się pisze od lewej do prawej, tak jak w przypadku języków europejskich. W przypadku wykorzystywania takich możliwości należy rozważyć takie kwestie jak kolejność czytania, wyrównanie tekstu i pozycja kursora.
- **Kształtowanie kontekstowe.** Niezbędne w przypadku alfabetu arabskiego i hinduskiego, gdzie kształt znaku zależy od kontekstu (położenia) wewnątrz słowa. Powoduje to inny ryt.
- **Łączenie znaków.** Ma miejsce wtedy, kiedy łączy się dwa lub więcej znaków, żeby zdefiniować nowy ryt, tak jak ma to miejsce w alfabetach arabskim, tajskim i hinduskim.
- **Dzielenie słów i justowanie.** Alfabet tajski nie uwzględnia odstępów pomiędzy słowami. Aby wprowadzić dzielenie słów, jest potrzebny wewnętrzny słownik. Podobnie jest w przypadku alfabetów arabskich. Odstępy pomiędzy znakami, wprowadzone w celu wyjustowania tekstu, spowodują zniszczenie kształtu znaku.
- **Kombinacje niedozwolonych znaków.** Tego należy unikać w przypadku alfabetu tajskiego, gdyż spowoduje to pojawienie się nieznanego rytu.
- **Kierunek czytania.** Wizualny porządek, w jakim są wyświetlane znaki i słowa oraz w jakim są odczytywane grupy słów.

Poprawione funkcje językowe pozwalają silnikowi SQL Server i narzędziom zarządzającym na poprawną obsługę wprowadzania, przechowywania i wyświetlania znaków złożonego alfabetu, jednak nie uwzględnia to lokalizacji ani mirroringu. Wcześniejsze wersje SQL Server obsługiwały przechowywanie złożonych i dwukierunkowych alfabetów, ale interfejs użytkownika nie obsługiwał wprowadzania i wyświetlania takich znaków.

SQL Server 2005 obsługuje GB18030 — standard kodowania znaków używanych w Chińskiej Republice Ludowej. Wszystkie systemy komputerowe wykorzystywane w ChRL muszą obsługiwać GB18030. GB18030 nie tylko definiuje zakres znaków, ale także standaryzuje punkty kodowania znaków. Zawiera w sobie Unicode's Unihan Extension A i definiuje kody odstępu dla wszystkich używanych i nieużywanych punktów kodowania Unicode poziom 0 (BMP) oraz 16 dodatkowych poziomów. Efektywnie tworzy relację typu *jeden-do-jednego* pomiędzy częściami GB18030 i całkowitą przestrzenią kodowania Unicode. GB18030 definiuje i wykorzystuje czterobajtowy mechanizm kodowania. GB18030 jest nowym standardem, wcześniejsza wersja SQL Server nie obsługiwała go.

Twórcy Microsoft SQL Server 2005 wprowadzili obsługę dla znaków zastępczych. Zadaniem Unicode jest obsługa pojedynczego punktu kodowego każdego znaku w każdym języku, gdy do jego wyświetlenia nie wystarcza 65 536 znaków. Języki takie jak chiński wymagają dużej liczby znaków do zakodowania rzadko występujących liter. Tak więc obsługa w standardzie Unicode została zwiększona dla znaków zastępczych, żeby móc obsłużyć dodatkowe 1 048 576 znaków.

Strefą zastępczą w systemie Unicode jest zakres od U+D800 do U+DFFF, zawierający 1024 niskie wartości zastępcze i 1024 wysokie wartości zastępcze. Kombinacja wysokich i niskich wartości zastępczych pozwala na zakodowanie ponad miliona możliwych znaków. Taki kod składa się z dwóch znaków zastępczych: wysokiego i niskiego. Sprawdzanie znaków zastępczych sprowadza się do porównania zakresów z uwzględnieniem skomplikowanych zasad, gdzie jest wymagane sprawdzenie dwubajtowego systemu znaków (ang. *double-byte character system* — DBCS).

Organizacje, które aktywnie działają na terenie wschodniej Azji, takie jak rząd, media, ubezpieczenia i służba zdrowia, wymagają zapewnienia obsługi symboli zastępczych, dzięki czemu można przechowywać i przetwarzać dane w takich językach jak chiński.

Wyszukiwanie pełnotekstowe

Niezależnie od tego, czy chodziło o wyszukiwanie danych w internecie, czy o nowy model przeszukiwania dokumentów, Microsoft zawsze pokładał wielkie nadzieje w wyszukiwaniu pełnotekstowym (ang. *Full-Text Search*). Twórcy platformy SQL Server 2005 rozwiązali wiele problemów z zarządzaniem i infrastrukturą, dzięki czemu wyszukiwanie pełnotekstowe w SQL Server 2005 stało się techniką godną uwagi. Przede wszystkim należy tu wspomnieć o uaktualnieniu MSSearch do wersji 3.0, dzięki czemu osiągnięto następujące korzyści:

- znacznie zwiększono wydajność pełnotekstowego indeksu bazy;
- jedna instancja MSSearch 3.0 przypada na każdą instancję SQL Server;
- MSSearch 3.0 jest uruchamiany w ramach tego samego konta usługi, co SQL Server.

Podczas wyszukiwania pełnotekstowego Microsoft SQL Server wykorzystuje jedną instancję MSSearch dla każdej instancji SQL Server. Każda instancja MSSearch uruchamia się w ramach tego samego konta usługi SQL Server. Jeśli administrator zmieni konto usługi SQL Server dla wybranej instancji, związane z nią konto usługi MSSearch także się zaktualizuje. We wcześniejszych wer-

sjach SQL Server jedna instancja MSSearch, uruchomiona jako LocalSystem, była współużytkowana przez wszystkie instancje i aplikacje na serwerze. W tym podrozdziale przedstawię nowe cechy wyszukiwania pełnotekstowego, obsługiwanego przez SQL Server 2005.

Zapytania serwerów połączonych

Istnieje możliwość wprowadzania kwerend wyszukiwania pełnotekstowego odwołujących się do zdalnego serwera. W ten sposób można na platformie SQL Server wykonywać polecenia związane z odwoływaniem się do źródeł danych na różnych serwerach, obsługujących OLE DB. Wcześniejsze wersje SQL Server pozwalały na tworzenie standardowych zapytań do połączonych serwerów, ale nie obsługiwały wyszukiwania pełnotekstowego dla połączonych serwerów.

Po zdefiniowaniu połączonego serwera można wykonywać na nim zdalne zapytania. W tym celu należy skorzystać z indeksów pełnotekstowych i użyć czteroczęściowej pełnej kwalifikowanej nazwy serwera (ang. *fully qualified four-part name*).

Przeszukiwanie różnych kolumn

Przeszukiwanie pełnotekstowe można wykonywać dla różnorodnych kolumn. Wcześniejsze wersje SQL Server ograniczały odwołania do pojedynczej kolumny albo do wszystkich w tabeli. SQL Server 2005 usuwa to ograniczenie, umożliwiając wybór dowolnej liczby kolumn w celu określenia kolumn do wyszukiwania pełnotekstowego.

Listę przeszukiwanych kolumn określa się za pomocą argumentu `column_list` predykatu pełnotekstowego. Kolumny wewnątrz pojedynczej listy kolumn predykatu muszą należeć do tej samej tabeli i nie można korzystać ze znaku * na liście kolumn.

Specyfikacja języka w zapytaniu

Twórcy Microsoft SQL Server 2005 wprowadzili funkcję pozwalającą na dodawanie do zapytania pełnotekstowego parametru, który określa lokalne ID (LCID). W ten sposób można wskazać, że dla określonej lokalizacji należy wykorzystać takie zasoby językowe, jak dzielenie słów i ich odmiana, co pozwala na indeksowanie i przechowywanie w pojedynczej kolumnie danych zapisanych w różnych językach. W ten sposób jest możliwe przetwarzanie zapytań w języku innym niż domyślny dla kolumny.

Mówiąc o zapytaniach pełnotekstowych, należy wymienić cztery słowa kluczowe: `CONTAINS`, `CONTAINSTABLE`, `FREETEXT` i `FREETEXTTABLE`. W poprzednich wersjach platformy te cztery predykaty nie akceptowały parametru LCID, określającego dzielenie słów i ich odmianę w zapytaniach. W SQL Server 2005 składnia tych poleceń została rozszerzona do specyfiki LCID.

UWAGA: W poprzednich wersjach SQL Server można było określić język zapytania pełnotekstowego tylko na poziomie kolumny, z wykorzystaniem procedur składowanych. Warto rozważyć sytuację przedsiębiorstwa, które utrzymuje bibliotekę dokumentów przechowywanych w bazie danych SQL Server. Są to dokumenty Microsoft Word, zawierające informacje w różnych językach, na przykład w niemieckim, japońskim, amerykańskim angielskim i szwedzkim. Jeśli projektant ma utworzyć zapytanie `CONTAINSTABLE`, przeszukujące posiadane dokumenty zapisane w języku niemieckim, w wyniku którego nastąpi wyświetlenie wyszukanych dokumentów uporządkowanych w najlepszy dla tego języka sposób, to może określić niemiecki LCID w napisanym zapytaniu.

Zapytania a widoki

Platforma SQL Server 2005 pozwala na wyświetlanie wyników pełnotekstowych zapytań indeksujących w widokach. W przypadku wcześniejszych wersji SQL Server można było jedynie włączyć zapytania do tabel w indeksach pełnotekstowych i wykorzystać je w dalszych operacjach wyszukiwania. Platforma SQL Server 2005 może wykonywać zapytania na widokach indeksów pełnotekstowych, opartych na jednej lub większej liczbie tabel, przeprowadzając tę operację w taki sam sposób, jak w przypadku przeszukiwania tabel.

Indeksowanie danych typu XML

Wyszukiwanie pełnotekstowe w SQL Server 2005 obsługuje tworzenie pełnotekstowych indeksów i pełnotekstowych zapytań do danych typu XML. Zapytania są tak dokładne, jak wartości w kolumnach. Predykat pełnotekstowy, zastosowany dla indeksowanej pełnotekstowej kolumny XML, zwraca wiersze zawierające poszukiwany ciąg znakowy w dowolnym miejscu kolumny. Istnieją następujące możliwości wyboru sposobów indeksowania w indeksie pełnotekstowym:

- według nazwy znacznika i jego pola (treści);
- według tylko wartości atrybutu znacznika i jego pola (treści);
- według tylko pola (treści) znacznika[2].

[2] Przyjęto następujący schemat znacznika XML: `<nazwa_znacznika atrybut="wartość_atrybutu">pole_znacznika</nazwa_znacznika>` — *przyp. red.*

SQL Server Express

W tym podrozdziale przedstawię najnowsze i prawdopodobnie najciekawsze narzędzie do projektowania, dostępne na platformie SQL Server — SQL Server Express. Warto to omówienie poprzedzić krótką analizą starszych narzędzi tego typu i porównać platformę SQL Server Express z Microsoft Desktop Engine (MSDE) i Microsoft Access. Opiszę także, w jaki sposób należy zarządzać i projektować aplikacje bazodanowe za pomocą platformy SQL Server Express.

Każdego dnia można znaleźć doniesienia o tym, jak Microsoft udaremnia różnym mniejszym przedsiębiorstwom konkurowanie ze sobą lub spowodował ich upadek poprzez swoje miażdżące posunięcia marketingowe. Można się również spotkać ze stwierdzeniami, że Microsoft nie odpowiada konkurencji w sensowny sposób albo że odpowiedzi te są spóźnione. Być może te twierdzenia są prawdziwe w wielu dziedzinach oprogramowania, ale baz danych nie dotyczą. Microsoft cały czas śledził postępy MySQL. W ciągu ostatnich dwóch lat MySQL gwałtownie zwiększył swój udział na rynku relacyjnych baz danych (RDBMS), mimo że w tym czasie główni gracze uważali, że to już koniec walki. Uważano, że Oracle, Microsoft i IBM są jedynymi znaczącymi producentami baz danych wykorzystywanych przez przedsiębiorstwa. Okazało się jednak, że MySQL stał się na rynku prawdziwą potęgą. Jest to darmowe oprogramowanie, działa w systemach Windows i Linux, ponadto jest proste w użytkowaniu. Wykonuje zadania, których oczekuje się od prostych baz danych — przechowuje dane. Ponadto atutem MySQL na rynku projektowania baz danych jest proste licencjonowanie. Dodatkową ważną kwestią jest to, że twórcy aplikacji powinni na bieżąco rozwijać swoje produkty, albo przez dodawanie nowych funkcji, albo nawet przez wdrażanie nowych technologii. Dla takich osób MySQL jest doskonałym rozwiązaniem.

Microsoft był w trakcie przygotowania platformy SQL Server 2005, kiedy otrzymał bardzo bolesną lekcję. W lutym 2001 roku robak SQL Slammer skutecznie zaatakował ogromną liczbę serwerów internetowych, czym zmusił Microsoft do dokładnego przeanalizowania przyczyny problemów — MSDE. MSDE był dostępny wraz z poprzednimi wersjami SQL Server. Z punktu widzenia marketingu i rozwoju produktu, MSDE stał się dość kłopotliwym elementem. W ten sytuacji Microsoft postąpił w standardowy sposób — opublikował raport o szkodliwym kodzie i dostarczył stosowną poprawkę, jednak problem nie polegał na braku poprawki. Problemem było to, że administratorzy systemów informatycznych nie wiedzieli, który produkt na których komputerach korzystał z MSDE w sposób niejawny.

W tym interesującym momencie MSDE stał się bardzo niepopularny. MySQL powoli zaczął przejmować część rynku licencji SQL Server Standard, więc Microsoft musiał zareagować. W ten sposób pojawił się SQL Server Express. Seria Express produktów Microsoft została zaprojektowana z myślą o filozofii *open-source*, zapewnieniu rozszerzalności itp., jednak z zastrzeżeniem, że to są wciąż produkty Microsoft i ich kod nie jest do końca otwarty. Żeby dodać kolejną zmienną do tego intrygującego równania, społeczność *open-source* krytykowała początkujących projektantów, choć mniejsza liczba projektantów tworzących aplikacje w technologii Microsoft nie oznaczała mniejszej sprzedaży aplikacji Microsoft.

Oto kilka interesujących cech platformy SQL Server Express, o których warto pamiętać:

- SQL Server Express posiada wiele cech SQL Server Standard Edition;
- jego licencja nie wymaga uiszczania opłat w przypadku dalszej dystrybucji;
- można pobrać narzędzie do zarządzania, ułatwiające przeprowadzanie podstawowych czynności administracyjnych;
- domyślnie jest zabezpieczony.

Integracja z Visual Studio

Modele biznesowe zawsze były proste: trzeba tylko zadowolić projektantów, a reszta będzie działać poprawnie. W ramach tej filozofii Microsoft dostarczył nową edycję Visual Studio przeznaczoną dla początkujących i jest ona darmowa. Microsoft Visual Basic Express i Microsoft Visual Web Developer są dwoma podstawowymi narzędziami projektowymi dla SQL Server Express.

Microsoft reklamuje SQL Server 2005 i Visual Studio 2005 jako najlepsze technologie projektowania aplikacji bazodanowych, zarówno tych opartych na WWW, jak i działających w systemach Windows (Czytelnik mógłby zauważyć, że jestem maniakiem bazodanowym, ponieważ słowo „aplikacja" kojarzę z „bazą danych" — tak jakby nie było innych rodzajów aplikacji!). Nawet Xbox korzysta z różnych baz danych. Łatwo zauważyć, że Microsoft zaprojektował kilka technologii pod nazwą Express. Oczywistą prawdą jest, że kiedy jakaś technologia *open-source* osiąga poziom dostępności i łatwości użycia, staje się liczącą się technologią. Do produktów Express zaliczają się aplikacje z linii Visual Studio i SQL Server.

Oto pięć produktów Express służących do projektowania aplikacji:

- **Visual Basic.NET Express** dla aplikacji Windows Forms;
- **Visual Web Developer** dla aplikacji ASP.NET;

- **Visual C# Express** dla aplikacji Windows Forms;
- **Visual C++ Express** dla aplikacji Windows Forms;
- **Visual J# Express** dla aplikacji Windows Forms.

SQL Server Express również należy do tej rodziny. Podczas pobierania dowolnego produktu Visual Studio Express można również pobrać SQL Server Express.

Podczas instalowania SQL Server Express przy instalacji jednego z produktu Visual Studio Express, SQL Server Express instaluje się z domyślną nazwą instancji (SQLEXPRESS). Jeśli SQL Server Express został już zainstalowany, nie jest instalowana inna jego wersja.

Czy SQL Server Express wystarczy?

Jakie cechy środowiska programistycznego są potrzebne początkującym projektantom? SQL Server Express udostępnia wszystkie kluczowe technologie (XML, CLR, SQL transakcyjny), podobnie jak SQL Server Standard Edition. Czy wobec tego SQL Server Express może być porównywany do MSDE 2000, który będzie obsługiwany jeszcze tylko przez krótki czas? MSDE był rozprowadzany wraz z aplikacjami przez kilka lat. Wraz z nadejściem SQL Server Express zmieniono kilka elementów współpracy z MSDE.

- **Projektowanie i wdrażanie.** SQL Server Expres jest instalowany za pomocą programu instalacyjnego opartego na MSI, który pozwala skryptom WMI na kontrolowanie procesu instalacji i konfiguracji. MSDE można było instalować różnymi metodami, co powodowało, że instalacja i zarządzanie były niespójne.
- **SQL Server Express jest domyślnie** wyłączony, trzeba więc samodzielnie zadbać o jego zabezpieczenia. MSDE z Service Pack 3a jest odłączony od sieci. Czynności zabezpieczające można przeprowadzić poprzez Centrum Zabezpieczeń w Windows XP Service Pack 2 i Windows 2003 Service Pack 1.

Bardzo ważne jest, aby użytkownicy zablokowali możliwość aktualizacji aplikacji korzystających z MSDE 2000. Niektóre podstawowe różnice pomiędzy dwoma wersjami produktów przedstawiłem w tabeli 4.6. Microsoft nadal będzie rozwijał MSDE.

Zastosowanie platformy SQL Server 2005 w wersji Express należy rozważyć w przypadku większej bazy danych, aby zapewnić sobie zwiększenie wygody zarządzania.

Tabela 4.6. *Porównanie SQL Server Express i MSDE*

SQL Server Express 2005	MSDE 2000
Obsługa Application XCopy	Opcja niedostępna
Brak SSIS (zmiana nazwy z DTS)	Obecne środowisko DTS
Proste wdrożenie; brak Microsoft Data Access Components (MDAC)	MDAC jest częścią instalacji
tylko MSI	MSI i MSM
Replikacja klienta transakcyjna, połączeniowa i migawkowa	Obsługa połączenia lub publikacji migawki stanowi dodatek do subskrypcji replikacji
Dobry interfejs użytkownika programu instalacyjnego	Podstawowy interfejs instalatora
Brak SQL Agent	SQL Agent obecny
Obsługa Windows 2000 SP4, Windows XP SP1 i Windows 2003	Obsługa Windows 98, Windows Me, Windows 2000, Windows XP, Windows NT 4 i Windows 2003
Obsługa CLR	Brak obsługi CLR
Narzędzia GUI	Brak narzędzi GUI
Limit bazy danych: 4GB	Limit bazy danych: 2 GB
Jeden procesor, 1GB RAM	Dwa procesory, 2 GB RAM
Brak kontroli przepustowości (ang. *throttle*)	Wymuszanie przepustowości zgodnie z obciążeniem

W rzeczywistości wielu użytkowników ocenia aplikacje zaprojektowane na podstawie MSDE jako nisko kosztowe, nisko transakcyjne i nisko budżetowe. W przypadku takich domowych aplikacji istnieje tylko niewielka motywacja, żeby migrować je do SQL Server Express. Jak przedstawię w dalszej części tego rozdziału, platforma SQL Server Express ma stanowić bazę danych dla początkujących, jednak najczęściej użytkownicy stawiają pierwsze kroki w dziedzinie baz danych, korzystając z Microsoft Access. Jak można więc porównać SQL Server Express z Access?

Microsoft SQL Server Express został wyposażony w bazodanowy silnik Jet, który jednak nie sprawdza się przy wykonywaniu bardziej zaawansowanych operacji, takich jak na przykład uaktualnienie do SQL Server Standard Edition. Baza danych oparta na silniku Jet zapewnia jednakże silną integrację z aplikacjami pakietu Office, czyli z Microsoft Access. Ponadto jest dostarczana z wieloma łatwymi do użycia aplikacjami i może być łatwo wbudowywana w bardziej zaawansowane aplikacje. To są najważniejsze przyczyny sukcesu Accessa. Jeżeli jest potrzebna baza danych klientów lub magazynu, wystarczy wykorzystać przykładową bazę danych Access. Z drugiej strony, jeżeli ktoś uważa, że jego baza danych któregoś dnia będzie mogła się stać bazą SQL Server Standard albo jeśli zechce stać się znakomitym projektantem aplikacji bazodanowych, musi zastanowić się nad nauką obsługi SQL Server Express. Osoby, które starają się utrzymać koszty działu IT pod kontrolą, docenią, że SQL Server Express jest bezpłatną aplikacją i że łatwo poddaje się migracji i zarządzaniu dzięki temu, że został oparty na technologii SQL.

Ograniczenia platformy SQL Server Express

W tym podrozdziale krótko przedstawię niektóre ograniczenia platformy SQL Server Express.

Obsługa replikacji

Platforma SQL Server Express umożliwia udział w replikacji jako subskrybent, co może mieć na celu utworzenie połączenia, migawki i replikacji transakcyjnej. SQL Server Express może korzystać ze wszystkich nowych funkcji replikacji, tak jak to opisałem w rozdziale 3. „Korporacyjne zarządzanie danymi". Dodatkowo SQL Server może przeprowadzać synchronizację na żądanie lub zaplanowaną. W tym celu należy się posłużyć narzędziem Windows Synch Manager.

SQL Service Broker

Aby aplikacja na platformie SQL Server Express mogła korzystać z usługi SQL Service Broker, trzeba zapewnić połączenie z innymi edycjami SQL Server. Chociaż SSB jest zaimplementowany jako seria rozszerzeń SQL transakcyjnego, SQL Server Express nie może być równorzędnym partnerem w dialogu z SSB. Kiedy instancja Express otrzymuje komunikat od innej instancji Express i komunikat nie zostaje przetworzony, zostaje on porzucony. Innymi słowy, SQL Server Express nie może być źródłem komunikatu. Komunikat musi zostać przekazany aplikacji SSB opartej na SQL Server Standard Edition lub Enterprise Edition. Celem tego ograniczenia jest ochrona licencji innych edycji SQL Server.

Ustawienia językowe

Platforma SQL Server Express jest dostarczana w 12 językach, określonych przez inne edycje SQL Server. SQL Server Express jest także zgodny z innymi edycjami pod względem obsługi ustawień językowych.

Obsługa technologii 64-bitowej

SQL Server Express zapewnia obsługę technologii Windows on Windows (WOW) na komputerach 64-bitowych. Oznacza to, że można uruchomić 32-bitowe wersje programu na 64-bitowej maszynie. W rzeczywistości 64-bitowa wersja SQL Server Express nie istnieje.

Instalacja i konfiguracja

Platforma SQL Server Express jest jednym z mniej skomplikowanych programów bazodanowych pod względem instalacji. SQL Server Express korzysta z tego samego, opartego na MSI, programu instalacyjnego Windows, co inne platformy z linii SQL Server. Obecne jest drzewo instalacji oraz Setup Consistency Checker. Prawdziwą przewagą nad MSDE daje interfejs użytkownika podczas instalacji — MSDE nie miało go w ogóle. Interfejs użytkownika SQL Server Express ma znajomy wygląd. Platforma SQL Server Express może być też zainstalowana bez użycia interfejsu użytkownika, z wykorzystaniem linii poleceń.

Przed zainstalowaniem SQL Server Express należy rozważyć kilka istotnych kwestii. Przede wszystkim trzeba pamiętać, że SQL Server Express wymaga systemu operacyjnego z.NET Framework 2.0. W przeciwnym przypadku SQL Server Express nie zostanie zainstalowany. SQL Server nie jest dostarczany razem z .NET Framework 2.0. Z drugiej strony, SQL Server Express można pobrać i zainstalować wraz z dowolną edycją Visual Studio Express. Najważniejszą zależnością z .NET Framework jest integracja z CLR. SQL Server wykorzystuje również zestawy zainstalowane przez .NET Framework do tworzenia interfejsu użytkownika podczas instalacji. .NET Framework nie powinien być problemem dla projektantów, ponieważ Microsoft bardzo rozpowszechnia to środowisko — nawet Windows Update zawiera .NET Framework. Pisząc więc swoje aplikacje, trzeba pamiętać o uwzględnieniu kodu sprawdzającego, czy .NET Framework 2.0 został zainstalowany i czy jest to właściwa wersja.

Po rozpoczęciu instalacji trzeba wybrać nazwę dla instancji programu. Instancję można traktować jak kompletną kopię programu. SQL Server Express obsługuje do 15 instancji na pojedynczej maszynie. SQL Server obsługuje ich już do 50 na jednym komputerze. Microsoft poszedł na pewne ustępstwa, oznajmiając projektantom, że aby ułatwić zarządzanie aplikacją, SQL Server Express powinien być instalowany w domyślnej instancji, nazwanej SQLEXPRESS. Takie postępowanie ma sens, ponieważ wszystkie bazy danych znajdują się wewnątrz jednej instancji i nie kolidują ze sobą. Wykorzystanie domyślnej instancji chroni przed innymi potencjalnymi problemami z zarządzaniem, ponieważ potrzeba mniej pamięci RAM, mniej miejsca na dysku i w ogóle powstaje mniej komplikacji, szczególnie w przypadku początkujących projektantów. Inną korzyścią jest to, że domyślna instancja pozwala na zarządzanie wszystkimi metadanymi w jednej nadrzędnej systemowej bazie danych i trzeba zapamiętać tylko jedno hasło administratora.

W przypadku wyboru instalacji nazwanej wersji SQL Server Express wszystkie składniki zostaną zainstalowane w osobnej instancji. W pewnych sytuacjach takie rozwiązanie jest korzystne, jednak zwykle najbardziej sensowne jest instalowanie serwera pod domyślną nazwą. Jeśli wybrano nazwaną instancję, nazwa domyślna jest dostępna dla instalacji innych aplikacji.

Tzw. cicha instalacja SQL Server Express również jest możliwa i przydatna w przypadkach, gdy przewiduje się instalowanie SQL Server Express na maszynach użytkowników bez sprawdzania wcześniej zainstalowanego oprogramowania. Microsoft sugeruje dodanie do aplikacji kodu weryfikującego istnienie kopii SQL Server Express przed próbą instalacji nowej aplikacji i bazy danych. Jeżeli zostanie wybrana instalacja poza istniejącą instancją, należy podać nazwę dla nowej instancji. Podczas cichej instalacji (pojęcie to w rzeczywistości oznacza instalację z poziomu linii poleceń) podajemy nazwę dla odpowiedniej instancji. Projektanci tworzący pakietowe aplikacje, które instalują SQL Server Express, mogą zapisać dane konfiguracji w pliku. Plik ten powinien być jednak usunięty lub zabezpieczony po zakończeniu instalacji. W celu skonfigurowania procesu instalacji można też skorzystać z Server Management Objects (SMO) lub SQLCMD.

Właściwa instalacja wersji Express, wykonana przy użyciu interfejsu użytkownika, jest bardzo podobna do instalacji innych edycji SQL Server. Drzewo instalacji poszczególnych funkcji jest takie samo. Następnie w obydwóch przypadkach uruchamia się Setup Consistency Checker. Dodatkowo można uruchomić instalację, nie posiadając Windows Installer 3.0. Można go pobrać z witryny Windows Download. Został on dołączony także do ostatnich pakietów serwisowych (*service pack*).

Po zakończeniu instalacji SQL Server Express dane są bezpieczne. SQL Server Express został oparty na nowym modelu bezpieczeństwa aplikacji Microsoft i dlatego domyślnie jest wyłączony. Jedną z pierwszych czynności jest podjęcie decyzji o ustawieniach bezpieczeństwa SQL Server Express. Domyślnie pamięć dzielona jest włączona, ale usługa SQL Server Express Server Service (niech ktoś spróbuje wypowiedzieć to trzy razy szybciej…) pozostaje wyłączona w trakcie działania programu instalacyjnego. Korzystając z nowego narzędzia **Server Surface Area Configuration** (**SSAC**), można odpowiednio ustawić poziom zabezpieczeń i usługi. Narzędzie SSAC umożliwia kontrolę bezpieczeństwa i funkcjonalności oferowanych przez SQL Server Express. SQL Server posiada dwie główne grupy *Services and Connections* i *Features*. W panelu *Features* można znaleźć kilka interesujących narzędzi kontrolujących działanie takich funkcji jak na przykład CLR. Należy pamiętać, że są one domyślnie wyłączone, przed projektowaniem aplikacji należy je więc włączyć. Domeną działania narzędzia SSAC jest lokalny komputer, więc każdy komputer posiada własną kopię, która jest instalowana domyślnie.

Protokoły obsługiwane przez SQL Server Express

Platforma SQL Server Express może zostać wykorzystana jako lokalna aplikacja bazodanowa albo w klasycznym schemacie klient-serwer. Każdy z tych scenariuszy wymaga różnych parametrów połączenia i ustawień bezpieczeństwa. Kiedy SQL Server Express wykorzystuje połączenie sieciowe, trzeba włączyć nowe narzędzie SQL Browser. SQL Browser wskazuje aplikacji położenie poszczególnych baz danych w sieci. Jeżeli uruchomiono SQL Browser, należy zwiększyć poziom zabezpieczeń, włączając w to porty niewykorzystane domyślnie przez SQL Server. Należy zmieniać ustawienia aplikacji przy każdej zmianie portu, ale ponieważ pewne przypisania portów wymagają interwencji administratora, po ich przypisaniu najprawdopodobniej będą często przez nich zmieniane. Dostęp do wszystkich wymienionych protokołów jest zarządzany poprzez SQL Computer Manager. Każdy protokół może być włączony lub wyłączony poprzez kliknięcie prawym przyciskiem myszy.

- **Pamięć dzielona.** Jest to najprostszy w użytku protokół, który nie posiada ustawień konfigurowalnych. W związku z tym, że klienty, korzystając z tego protokołu, mogą połączyć się z instancją SQL Server uruchomioną na tym samym komputerze, omawianego protokołu nie można wykorzystywać w aplikacjach działających w środowisku sieciowym.
- **TCP/IP.** Standardowy protokół, powszechnie stosowany w internecie. Zawiera mechanizmy wybierania trasy dla ruchu sieciowego i oferuje zaawansowane funkcje zabezpieczeń. Jest to najbardziej popularny protokół w codziennym zastosowaniu biznesowym. Konfiguracja komputera do korzystania z TCP/IP może być skomplikowana, ale większość komputerów jest już poprawnie skonfigurowana.
- **SQL Native Client.** Zarządzany dostawca bazodanowy, stanowi cześć środowiska .NET Framework. Oferuje najlepszą i najbardziej bezpieczną metodę dostępu do bazy danych SQL Server Express.
- **Potoki nazwane.** Protokół dostępu klienta zawierający TCP/IP, który z kolei jest standardem protokołu dostępu do informacji za pośrednictwem sieci, włączając w to internet!

Jak widać, dla wielu organizacji SQL Server Express nie jest wystarczającym rozwiązaniem, nawet w przypadku małych wdrożeń aplikacji. Jest to aplikacja bazodanowa dla początkujących. Zaletą SQL Server Express jest możliwość redystrybucji i proste uaktualnienie do innych wersji SQL Server.

Podsumowanie

W niniejszym rozdziale przekrojowo opisałem wszystkie nowe funkcje, które są dostępne dla projektantów skalowalnych aplikacji tworzonych w technologii Microsoft i zgodnych z SQL Server 2005. Czytelnik mógł się przekonać, że SQL Server nie jest zwykłym RDBMS, ale raczej platformą aplikacji bazodanowych, służącą do tworzenia szerokiej gamy aplikacji, opartych na różnych modelach architektury. Nowe funkcje zmuszą administratorów do nauki zarządzania aplikacjami i zmagania z .NET w warstwie danych. Z punktu widzenia zarządzania IT, nowe funkcje oznaczają oszczędność kosztów w czasie wdrożeń i elastyczność w biznesie.

Jak Czytelnik mógł zauważyć, kilka nowych technologii składa się na integrację CLR, XML, nowych typów danych i nowej wersji SQL Server Express. Wszystkie te cechy zwiększają możliwości programistyczne.