Sebastian Rafałko, Adam Pelikant
Wydział Informatyki i Zarządzania
Wyższa Szkoła Informatyki w Łodzi

ZASTOSOWANIE SERVICE BROKER DO AUTOMATYCZNEJ SYNCHRONIZACJI DANYCH NA PLATFORMIE MICROSOFT SQL SERWER

Streszczenie – Głównym tematem artykułu jest stworzenie automatycznej komunikacji między dwoma bliźniaczymi bazami danych, dzięki której na obu bazach są wykonywane te same zapytania dotyczące zarówno modyfikacji danych w istniejącej strukturze relacyjnej, jaki i zmiany strukturalne bazy. Takie podejście stanowi rozszerzenie funkcjonalności replikacji, która pozwala jedynie na synchronizacje ściśle określonych danych. Omawiana praca skupia się na technicznej stronie problemu, stworzeniu pełnej struktury usługi Service Broker (kolejki, trasy, serwisy), nawiązaniu konwersacji między bazami, przetwarzaniu komunikatów. Omawiana funkcjonalność została uzyskana na skutek działania zestawu procedur wyzwalanych utworzonych dla schematu bazy danych, które automatycznie tworzą wyzwalacze poziomu pośredniego odpowiedzialne za synchronizacje danych. Przedstawiono również aplikacje zewnętrzną służąca do śledzenia komunikatów generowanych przez wyzwalacze oraz zaprezentowania wykonanych zmian. Jej zadaniem jest również przekierowanie przetwarzania do mniej obciążonej bazy, w celu poprawy wydajności przetwarzania.

1 Wstęp

Jednym z ważnych nowoczesnych podejść do przetwarzania jest idea rozpraszania baz danych. Pociąga ono za sobą konieczność synchronizowania danych pochodzących z różnych składów danych, co ma zapewnić bezpieczeństwo w przypadku awarii lub czasowego braku dostępu do jednego z węzłów. Takie mechanizmy synchronizacji możemy zapewnić stosując znane, wbudowane rozwiązania replikacji [21], [22]. Nie pozwalają one jednak w sposób automatyczny panować nad synchronizowaniem dla nowych elementów składów danych, ani elementów proceduralnych. Z drugiej strony modyfikacje danych pochodzące z procesu równoległego do procesu obsługiwanego przez aplikacje nie są dla tej aplikacji „widoczne" dopóki do nich się nie odwoła. Konieczne byłoby wiec tworzenie agentów cyklicznie

odpytujących serwer o takie zmiany. Wprowadzenie takiego procesu, działającego w tle, powoduje znaczne obciążenie serwera. Obie funkcjonalności mogą zostać zrealizowane za pomocą systemu powiadamiania.

SQL Server Service Broker jest wbudowaną w silnik serwera bazy danych usługą pozwalającą na obsługę wiadomości i kolejek aplikacji. Ułatwia ona tworzenie rozbudowanych aplikacji wykorzystujących ten silnik, do komunikacji między różnymi bazami danych [9], [10]. Pozwala on na łatwe tworzenie niezawodnych aplikacji rozproszonych, dzięki czemu można dystrybuować obciążenie pracą danych między wiele różnych baz danych. Takie rozwiązanie omija konieczność tworzenia złożonych kanałów komunikacji oraz przesyłania wiadomości wewnątrz oprogramowania wyższego rzędu, co prowadzi do zwiększenia wydajności programowanych aplikacji. Dzięki oparciu się na architekturze zorientowanej na usługi i asynchronicznemu przetwarzaniu informacji, narzędzie to znalazło wiele zastosowań, do których należą:

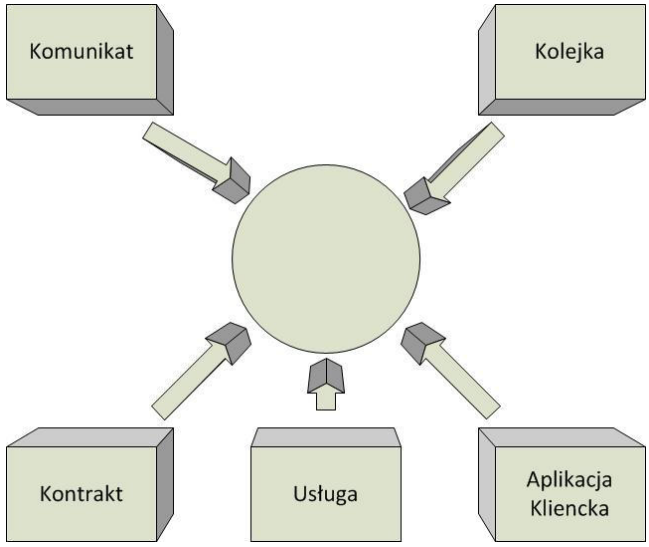
- asynchroniczne wyzwalacze [1], [2];
- niezawodne przetwarzanie zapytań [3], [4];
- serwery rozproszonego przetwarzania klient – aplikacja [5], [7], [10];
- konsolidacja danych dla aplikacji klienckich [6], [9];
- przetwarzanie danych rozproszonych [5], [11];
- tworzenie rozwiązań bazujących na przepływach (zestawienie funkcjonalności Service Broker i Windows Workflow Fundation) [8];
- tworzenie struktur typu publikacyjnych (subskrybenckich) [12].

Wykorzystanie funkcjonalności usługi Service Broker zapewnia szereg korzyść takich jak:

- zwiększona wydajność aplikacji i uproszczenie administracji dzięki integracji z bazą danych;
- koordynowanie kolejności przesyłania wiadomości do projektowanej aplikacji;
- elastyczność obciążenia pracą dzięki luźnym sprzężenia aplikacji;
- automatyczna aktywacja, która pozwala aplikacji samodzielnie odpowiadać na wiadomości przybywające do kolejki;
- blokowanie wiadomości pokrewnych, które pozwala na więcej niż jedno odwołanie aplikacji do procesów wiadomości z tej samej kolejki bez wyraźnej synchronizacji.

2 Wprowadzenie do technologii Service Broker Microsoft SQL Server

Service Broker jest usługą powiadamiania dostępną w SQL Serwer 2008. W poprzedniej wersji SQL Serwer 2005 usługa ta była dostępna pod nazwą Notification Services. Każda baza danych posiada własną usługę Service Broker. Dla nowo utworzonych baz danych usługa ta jest domyślnie wyłączona i należy ja włączyć. Usługa Service Broker jest opracowana w architekturze SOA, (ang. Service - Oriented Architecture – rysunek 1) [13], która opiera się na przesyłaniu komunikatów między usługami, w celu realizacji określonych zadań związanych z przetwarzaniem, administracją lub bezpieczeństwem systemu.



Rys. 1. Architektura SOA

Informacja wymieniana między aplikacjami wykorzystującymi tę architekturę bazuje na komunikatach, które mają określony typ. Typ komunikatu decyduje o jego formacie:

- NONE – zawartość komunikatów może być dowolna i nie jest sprawdzana;
- EMPTY – komunikaty tego typu muszą mieć pustą zawartość – NULL;
- WELL_FORMED_XML – zawartość komunikatu musi być poprawnie sformatowanym łańcuchem o formacie pliku XML;

- VAILD_XML WITH SHEMA COLLECTION – zawartość komunikatów musi zawierać dane w formacie XML, które są zgodne z określoną kolekcją schematów XML.

Komunikaty wchodzą w skład konwersacji. Oprócz typu, komunikat posiada unikalny identyfikator GUID konwersacji oraz numer porządkowy, który jest nadawany sekwencyjne w ramach sesji.

Kontrakt określa, jakiego typu komunikaty będą użyte w konwersacji oraz przez jakie strony konwersacji dane komunikaty będą wysyłane. W celu utworzenia komunikatu należy posłużyć się poleceniem CREATE CONTRACT. Możliwe jest określenie kierunku przesyłania komunikatu poprzez polecenie SENT BY:

- ANY – komunikaty tego typu może wysyłać dowolna usługa uczestnicząca w konwersacji;
- INITIATOR – komunikaty tego typu może wysyłać tylko usługa inicjująca konwersację;
- TARGET – komunikaty tego typu może wysyłać tylko usługa, która akceptuje konwersacje rozpoczętą przez inną usługę.

Definicja kontraktu nie narzuca z góry definicji usług, które mogą uczestniczyć w konwersacji. Informacje o istniejących kontraktach dostępne są w tabelach systemowych sys.service_contracts oraz sys.service_contracts_message_usages.

Kolejka służy do przechowywania komunikatów dla dalszego ich przetwarzania przez aplikację. W definicji kolejki mogą znaleźć się następujące opcje:

- ACTIVATION – zestaw informacji zawierających dane niezbędne do aktywacji procedury składowanej uruchamianej w odpowiedzi na przyjście komunikatu do kolejki; na zestaw ten składają się następujące cztery właściwości:
  - STATUS – określa czy aktywacja dla danej kolejki jest używana;
  - PROCEDURE_NAME – nazwa procedury składowej uruchamianej, gdy w kolejce pojawia się nowy komunikat; procedura musi istnieć w momencie wykonywania polecenia CREATE QUEUE;
  - MAX_QUEUE_READERS – określa maksymalną liczbę jednoczesnych uruchomień procedury składowej wskazanej przez wartość PROCEDURE_NAME;
  - ECECUTE_AS – określa kontekst (użytkownika bieżącej bazy danych) wykorzystany do wykonywania procedury składowej wskazanej przez właściwość PROCEDUE_NAME;

- RETENTION – określa, czy komunikaty maja być przechowywane w kolejce do końca konwersacji;
- STATUS – określa, czy kolejka jest otwarta na przychodzące komunikaty.

Informacje o istniejących kolejkach w bazie danych można uzyskać odpytując perspektywę systemową sys.service_queues. Na skutek przypisania kolejek do schematów w bazie danych (kolejki, to jedyne obiekty Service Brokera wymagające takiego przypisania) informacje o kolejkach można również uzyskać odpytując widoki systemowe: sys.objects, sys.all_objects.

Usługa pozwala na wysyłanie komunikatu do właściwej bazy, kolejki zgodnie z określonym kontraktem. Usługa przyjmuje następujące argumenty:

- AUTORIZATION – określa właściciela usługi;
- ON QUEUE – określa kolejkę, która jest przypisana do usługi.

Jedna usługa może obsługiwać wysłanie komunikatów zgodnych z wieloma różnymi kontraktami. Jeżeli dla usługi nie zostanie podany żaden kontrakt, może on jedynie inicjować konwersację z inną usługą. Listę zdefiniowanych usług można uzyskać wykonując zapytanie na widoku systemowym sys.services:

Trasa określa miejsce dostarczenia komunikatu. Usługi mogą znajdować się w różnych bazach danych a nawet instancjach serwera SQL Serwer 2008. Usługa wysyłająca komunikat używa tras do zlokalizowania usługi, która powinna otrzymać komunikat. Odpowiedzi również są przesyłane tymi samymi trasami. Podczas tworzenia trasy można użyć następujących parametrów:

- ADDRESS – adres sieciowy używany przez trasę;
  - LOCAL – oznacza doręczenie komunikatów do usługi znajdującej się na tej samej instancji serwera SQL, co bieżąca baza danych;
  - TRANSPORT – oznacza, że nazwa usługi podana w SERVICE_NAME odgrywa jednocześnie rolę adresu sieciowego;
- AUTHORIZATION – właściciel trasy;
- BROKER_INSTANCE – globalny identyfikator bazy danych GUID, która jest hostem usługi podanej, jako wartość parametru SERVICE_NAME;
- LIFETIME – czas przechowywania trasy w tabeli tras wyrażony w sekundach;
- MIRROR_ADDRESS – adres sieciowy odpowiadający kopii bazy danych powstałej w wyniku podwajania bazy danych, parametr ten

nie może być określony, jeżeli parametrowi ADDRESS przypisze się wartość LOCAL lub TRANSPORT.

- SERVICE_NAME – nazwa zdalnej usługi (należy uwzględnić wielkość liter, ponieważ porównanie odbywa się z rozróżnianiem wielkości liter oraz nie uwzględnia reguł sortowania używanych w aktualnej bazie danych), do której kieruje trasa.

Listę dostępnych tras można uzyskać z widoku systemowego sys.routes.

Konwersacje to cykl wymiany komunikatów między usługami. Gwarantują dotarcie komunikatów do celu, jeżeli użytkownik poprawnie zdefiniował obiekty niezbędne do prowadzenia konwersacji. Rozróżnia się dwa rodzaje konwersacji:

- MONOLOG – jedna usługa wysyła komunikaty, inne je otrzymują i przetwarzają, ale nie wysyłają komunikatów do usługi inicjującej;
- DIALOG – konwersacja między dwiema usługami; jedna usługa rozpoczyna konwersacje, inne zaś odbierają komunikaty i przetwarzają, a następnie wysyłają własne, dopóki konwersacja nie zostanie zakończona. W wersji SQL Serwer 2008 używane są tylko dialogi.

Obsługa dialogów przez Service Broker oferuje:

- gwarancję dostarczenia komunikatów;
- możliwość ustanowienia długiego czasu życia komunikatów;
- gwarancję doręczenia każdego wysłanego komunikatu dokładnie raz;
- gwarancję doręczenia komunikatów w kolejności ich wysłania;
- trwałość dialogów.

Konwersacjom można nadawać priorytety, które służą do:

- identyfikacji ważności konwersacji;
- obsługi scenariuszy biznesowych (klient płacący więcej ma większe przywileje od klienta płacącego mniej);
- OBSŁUGA konwersacji realizujących procesy biznesowe przed konwersacjami technicznymi, nieobsługującymi klientów.

W celu nadania priorytetu konwersacji należy użyć polecenia CREATE BROKER PRIORITY. Polecenie to przyjmuje następujące parametry:

- CONTRACT_NAME – nazwa kontraktu, która pozawala serwerowi SQL odnaleźć te konwersacje, którym ma być nadany priorytet;
- LOCAL_SERVICE_NAME – nazwa usługi lokalnej, której ma być nadany priorytet;
- REMONT_SERVICE_NAME – nazwa usługi zdalnej, której ma być nadany priorytet;

- PRIORITY_LEVEL – poziom priorytetu, dla której wartością domyślna jest określana przez DEFAULT natomiast liczby z zakresu 1-10 określają rzeczywisty priorytet.

Wszystkie konwersacje są grupowane (Conversation Group). Każda konwersacja należy dokładnie do jednej grupy konwersacyjnej, natomiast grupy konwersacyjne mogą być powiązane z wieloma konwersacjami. Grupy konwersacyjne zostały utworzone, aby zapewnić spójność procesów biznesowych obsługiwanych przez wiele konwersacji, które są powiązane z wieloma usługami Service Brokera. Dzięki blokowaniu grupy konwersacji w momencie wysłania lub odbioru komunikatu, Service Broker gwarantuje otrzymanie tylko jednego komunikatu w ramach jednej grupy konwersacyjnej.

3 Wysyłanie i odbieranie komunikatów

Komunikaty są wysyłane i odbierane podczas trwania konwersacji. Rozpoczęcie konwersacji następuje poprzez polecenie BEGIN DIALOG CONVERSATION. Parametry, jakie należy podać przy rozpoczynaniu konwersacji to:

- UNIQUEIDENTIFIER – identyfikator GUID konwersacji;
- FROM SERVICE – nazwa usługi inicjalizującej konwersacje;
- TO SERTVICE – nazwa usługi docelowej dla konwersacji;
- ON CONTRACT – nazwa kontraktu na podstawie, którego ma odbywać się konwersacja;
- WITH ENCRYPTION – ustawienie szyfrowania danych przesyłanych podczas konwersacji.

Polecenie SEND wysyła komunikat przyjmując następujące parametry:

- ON CONVERSATION – identyfikator GUID konwersacji;
- MESSAGE TYPE – typ konwersacji;
- treść komunikatu zgodnie z ustalonym typem.

Odbieranie komunikatów odbywa się poprzez polecenie RECEIVE bezpośrednio z kolejki. Polecenie END CONVERSATION zamyka konwersacje.

Service Broker obsługuje dwa rodzaje aktywacji:

- aktywacja wewnętrzna (Internal Activation) – automatycznie aktywuje procedurę składowaną, która została przypisana do kolejki, która nazywana jest również monitorem kolejki;
- aktywacja zewnętrzna (External Activation) – następuje w momencie wystąpienia określonego zdarzenia, która wykorzystywana jest w celu powiadomienia o zdarzeniach (Event Notification).

Zatrute komunikaty (Poison Message) to takie, których aplikacja nie może przetworzyć. Jeżeli polecenie RECEIVE zawierało transakcję to taki komunikat jest wycofywany za pomocą polecenia ROLLBACK i komunikat wraca do kolejki. Wycofany komunikat może być wybierany ponownie z kolejki. W celu uniknięcia zapętlenia się aplikacji w wyniku wywoływania zatrutych komunikatów, po pięciu nieudanych próbach przetworzenia komunikatu, blokowane są wszystkie kolejki, z których odebrano komunikat od wycofanej transakcji. Użytkownik może ponownie włączyć wszystkie zablokowane kolejki.

Service Broker zapewnia bezpieczną komunikację między usługami nawet, jeśli są one zdefiniowane w równych instancjach serwera. Co więcej, instancje te mogą znajdować się na różnych serwerach i nie mieć między sobą relacji zaufania i nie należeć do jednej sieci. Udostępniane są dwa rodzaje zabezpieczeń:

- zabezpieczenie transportu – zapewnia odizolowanie nieuprawnionej instancji serwera SQL, co w rezultacie kończy się niemożliwością wysyłania komunikatów przez tę instancję;
- zabezpieczenia dialogu – komunikaty są szyfrowane w obrębie dialogu, a uczestnicy identyfikowani.

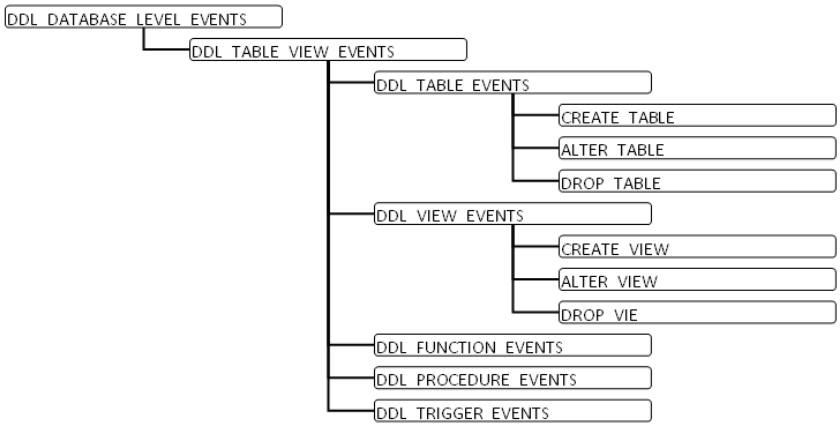
Ponadto Service Broker udostępnia dwa mechanizmy uwierzytelniania, dzięki którym można stworzyć bezpieczny kanał komunikacji pomiędzy dwoma instancjami serwera:

- uwierzytelnianie oparte na systemie Windows – wykorzystywane, gdy komunikujące się instancje należą do jednej domeny lub dwóch wzajemnie zaufanych domen, bazuje ono na protokołach NTLM i Kerberos;
- uwierzytelnianie oparte na certyfikatach – bazuje na wymianie publicznych certyfikatów między instancjami, działa szybciej niż uwierzytelnianie oparte na systemie Windows.

4 Automatyczne tworzenie wyzwalaczy dla powiadomień użytkownika

Aby zapewnić synchronizację danych można skorzystać z wbudowanego mechanizmu replikacji, jednak w takim przypadku dodanie nowej tabeli powoduje konieczność ponownego definiowania replikowanych obiektów. Konkurencyjnym rozwiązaniem jest utworzenie dla każdej z tabel schematu relacyjnego procedur wyzwalanych dla każdego z typów zapytań modyfikujących dane (INSERT, DELETE, UPDATE). Również w tym przypadku konieczne jest dopisywanie ich do każdego nowo tworzonego lub modyfikowanego obiektu (tabeli, perspektywy). W przypadku usuwania procedura wyzwalana jest

usuwana automatycznie, ale fakt o takiej modyfikacji schematu również powinien generować odpowiednią akcje w synchronizowanej bazie. Skutecznym rozwiązaniem wydaje się być automatyczne tworzenie takich wyzwalaczy w ciele triggerów poziomu wyższego tzn., takich które są reakcją na modyfikacje schematu [1]. Fragment hierarchicznej struktury procedur wyzwalanych z poziomu zdarzeń występujących na bazie danych, ograniczony do zdarzeń związanych głownie z tabelami i perspektywami przedstawia rysunek 2.



Rys. 2. Fragment struktury hierarchicznej wyzwalaczy dla bazy danych (schematu)

W celu wygenerowania powiadomień użytkownika możemy zastosować wyzwalacze z różnych poziomów hierarchii [14], [15], [16]. Przykładem może być wyzwalacz utworzony dla całej bazy (schemat) i wyzwalany w momencie utworzenia nowej tabeli, którego idee przedstawia skrypt:

```
CREATE  TRIGGER AfterCreateTable
ON DATABASE
FOR CREATE_TABLE
AS
-- skrypt wykonywany po aktywacji wyzwalacza
```

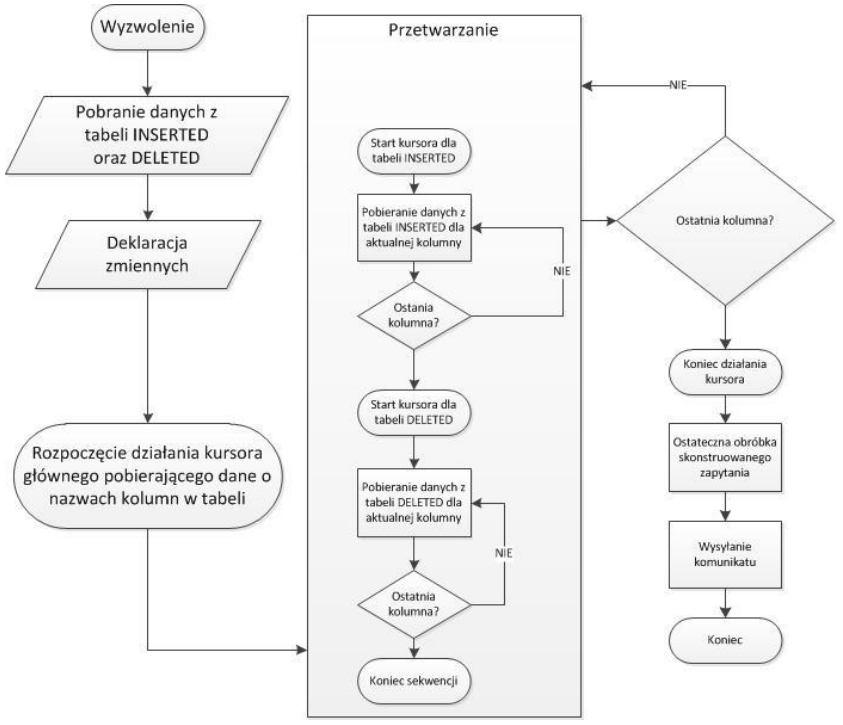
Możliwe jest wygenerowanie wyzwalaczy dla wszystkich trzech zdarzeń związanych z tabelami, albo przesuwając się w górę hierarchii korzystając ze zdarzenia nadrzędnego zawierające wszystkie z poziomu potomnego [21]. Skorzystanie z najwyższego poziomu pozwala na generowanie komunikatów związanych z każdą modyfikacją schematu, nie tylko związaną z tabelami, ale również między innymi z modyfikowaniem elementów proceduralnych, w tym również procedur

wyzwalanych. W takim przypadku konieczne jest zabezpieczenie przed powstaniem nieskończonej pętli na skutek modyfikowania wyzwalacza generującego powiadomienia. Takie modyfikacje powinny zostać zablokowane. W zrealizowanym algorytmie synchronizacji zdecydowani się tylko na śledzenie zmian składów danych, stąd triggery zostały utworzone tylko dla zdarzeń związanych z tabelami. Ciała wyzwalaczy DDL zostały podzielne na trzy części odpowiadające za:

- tworzenie wyzwalacza dla zdarzenia INSERT;
- tworzenie wyzwalacza dla zdarzenia DELETE;
- tworzenie wyzwalacza dla zdarzenia UPDATE.

Ze względu na obszerność oraz powtarzalność kodu zawartego przy tworzeniu wyzwalaczy nie zostanie on przytoczony. Wszystkie trzy wyzwalacze dla powiadomień użytkownika opierają się na zagnieżdżonych kursorach. Główny kursor zdeklarowany jest dla listy kolumn wybranej tabeli:

```
SELECT COLUMN_NAME FROM
INFORMATION_SCHEMA.Columns WHERE TABLE_NAME =
@tab
```



Rys. 3. Działanie wyzwalacza DML generującego powiadomienia do synchronizowanej bazy

Następnie wewnątrz tego kursora został zagnieżdżony kolejny kursor operujący na danych zawartych w tabeli tymczasowej #temp, która została wypełniona przy pomocy zapytania (w przypadku zdarzenia UPDATE wewnątrz zostają zagnieżdżone dwa kursory) dla danych zawartych w tabelach wewnętrznych wyzwalaczy – INSERTED oraz DELETED.

```
SELECT * INTO ##temp FROM  INSERTED/DELETED
```

Przy pomocy zagnieżdżonych kursorów budowane jest identyczne zapytanie jak zapytanie wprowadzone przez użytkownika. Dodatkowo zapytanie to zostaje wplecione do skryptu pomiędzy kodem wyłączającym wyzwalacz odpowiedzialny za przechwytywanie danych, a włączającym ten sam wyzwalacz. Taka sekwencja kodu zapobiega powstaniu nieskończonej pętli wywołań procedur wyzwalanych z poziomu aplikacji, a w konsekwencji powstaniu nieskończonej pętli powiadomień. Schemat działania generowania powiadomień w procedurach wyzwalanych przedstawia rysunek 3. Kiedy zapytanie zostanie już utworzone, wyzwalacz rozpoczyna konwersację z serwisem odbierającym bazy docelowej – synchronizowanej.

5 Aplikacja zarządzająca i monitorująca synchronizację

Program nadzorujący działanie procesu synchronizacji baz danych został napisany w języku Microsoft Visual C# 2010 [17], w którym zostały wykorzystane podstawowe biblioteki tego środowiska. Interfejs użytkownika został stworzony z zastosowaniem narzędzi Windows Presentation Foundation (WPF) [18], wykorzystujących język XAML do definiowania warstwy prezentacyjnej. Pozwala to na łatwy dostęp do zaawansowanych kontrolek platformy. Okno główne aplikacji (rysunek 4) składa się z trzech głównych elementów:

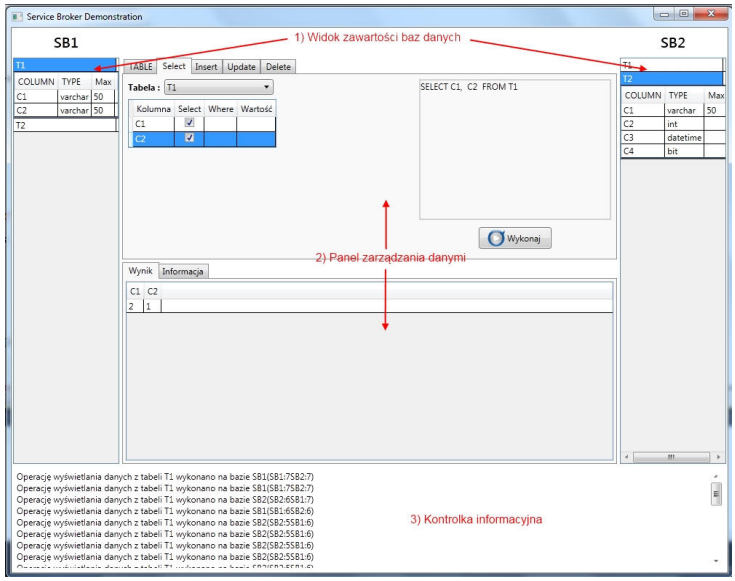
- widoku zawartości baz danych;
- panelu zarządzania danymi;
- kontrolki informacyjnej, pokazującej realizowane operacje.

Widok zawartości baz danych przedstawia listę tabel obu baz danych. Wybierając pozycje z listy możemy uzyskać informacje o budowie tabeli, takie jak: nazwa kolumny, typ danych, opcjonalnie maksymalna zdeklarowany rozmiar typu danych.

W celu poprawy wydajności przetwarzania założono, że inicjacja synchronizacji będzie zawsze rozpoczynała się w mniej obciążonej spośród synchronizowanych baz. Sprawdzanie obciążenia bazy następuje przed wykonaniem jakiegokolwiek zapytania SQL przez

program. Dane o obciążeniu obu baz są pobierane z tabeli systemowej sys.sysprocesses przy pomocy zapytania:

```
SELECT DB_NAME(dbid) AS DBName, COUNT(dbid) AS
NumberOfConnections
FROM sys.sysprocesses
WHERE dbid > 0 AND
(DB_NAME(dbid) = '" + Settings.Default.DB1 + "'
OR DB_NAME(dbid) = '" + Settings.Default.DB2 + "'
) GROUP BY dbid
```



Rys. 4. Okno główne programu zarządzającego i monitorującego synchronizację

Otrzymane wyniki są analizowane i na ich podstawie metoda zwraca nazwę mniej obciążonej bazy. Następnie następuje inicjacja konwersacji jak to zostało opisane wcześniej, a następnie wykonywane są zawarte w komunikatach zapytania realizujące zadanie synchronizacji. Z punktu widzenia użytkownika aplikacja zachowuje się jak menadżer klastra składającego z dwóch węzłów, będących dla siebie zwierciadlanym odbiciem.

6 Wnioski

Powstała aplikacja bazodanowa pozwalająca na samoistną, obustronną replikację danych i struktury bazy między dwoma bazami

spełniająca wszystkie założenia projektowe. Została oprogramowana aplikacja pozwalająca zademonstrować działanie systemu. Opisany projekt spełnia wszystkie postawione przed nim założenia:

- replikacja struktury bazy danych – wykorzystanie powiadamianie o zdarzeniach (Event Notyfication) w celu obustronnego odwzorowywania baz danych (odwzorowanie nie tylko struktury tabel, ale również automatyczne powielanie tworzonych lub edytowanych funkcji, procedur czy tez triggerów);
- replikacja danych – stworzenie wyzwalaczy DML przechwytujących informacje o zmianach w danych dla istniejącej tabeli w bazie danych;
- implementacja Service Broker – stworzenie kompletnej struktury Service Brokera nawiązującej dialogi między sobą (wraz ze stworzenie ras, kontraktów i typów wiadomości);
- automatyczne tworzenie wyzwalaczy DML dla nowo stworzonych tabel – implementacja wyzwalaczy DDL w celu tworzenia wyzwalaczy DML dla nowo stworzonych tabel;
- aplikacja demonstrująca działanie – stworzono aplikację, która demonstruje zachodzące zmiany w replikowanych bazach danych wraz z możliwością wyboru parametrów logowania do baz danych oraz synchronizowanych baz.

Powstały system można dalej rozwijać pod kątem tworzenia rozbudowanych aplikacji rozproszonych pozwalającym nie tylko skutecznie informować o zaistniałych zmianach, ale również tworząc bezpieczny kanał komunikacyjny do przesyłania danych bez potrzeby bezpośredniego wydobywania ich z bazy danych. Ważnym elementem rozwojowym jest generowanie powiadomień w celu warunkowego wyzwolenia procesu ETL [22] (extract, transform, load) zasilającego narzędzia przetwarzania analitycznego – hurtownie danych.

Literatura

[1] Kowalski T., Wiślicki J., Kuliberda K., Adamus R., Meina M., Integracja spadkowych danych relacyjnych do gridu obiektowego typu „data grid”, *Automatyka* 2009, tom 13, zeszyt 3.

[2] Kuliberda K., Adamus R., Wislicki J., Kaczmarski K., Kowalski T., Subieta K., A generic proposal for a transparent integration of distributed data by an autonomous layer in a virtual repository. *Multiagent and Grid Systems* 3(4): 393-410 (2007)

[3] Kuliberda K., Kowalski T., Draus C., Adamus R., Wiślicki J., Uogulnione podejście do aktualizacji indeksów w obiektowej bazie danych, *Automatyka* 2009, tom 13, zeszyt 3.

[4] Kowalski T., Kuliberda K., Wislicki J., Adamus R., Generic Approach to Automatic Index Updating in OODBMS. *ICEIS* (1) 2009: 255-260

[5] Chu X., Venugopal S., Buyya R., Grid Resource Broker for Scheduling Component-based Applications on Distributed Resources, Cyberinfrastructure Technologies and Applications, J. Cao (ed), ISBN: 978-1-60692-063-3, Nova Science Publishers, Hauppauge NY, USA, 2009.

[6] Mota-Herranz L., Celma-Gimenez M., Automatic Generation of Trigger Rules for Integrity Enforcement in Relational Databases with View Definition

[7] Valatkaite I., Vasilecas O., Automatic enforcement of business rules as adbms triggers from conceptual graphs model, *Informacinės Technologijos Ir Valdymas*, 2004, Nr.2(31)

[8] Fraternali P., Paraboschi S., Tanca L.,Automatic rule generation for constraint enforcement in active databases. *Proceedings of 4th International Workshop on Foundations of Models and Languages for Data and Objects*, Volkse (Germany), 1992

[9] Chen H., Mohapatra P., Using service brokers for accessing backend servers for web applications, *Journal of Network and Computer Applications* 28 (2005) 57–74

[10] Venugopal S., Buyya R., Winton L., A Grid Service Broker for Scheduling e-Science Applications on Global Data Grids, Cyberinfrastructure Technologies and Applications, J. Cao (ed), ISBN: 978-1-60692-063-3, Nova Science Publishers, Hauppauge NY, USA, 2009

[11] Venugopal S., Nadiminti K., Gibbins H., Buyya R., Designing a Resource Broker for Heterogeneous Grids, Software: *Practice and Experience*, Volume 38, Issue 8, Pages: 793-825, ISSN: 0038-0644, Wiley Press, New York, USA, July 10, 2008.

[12] Lin H., Abawajy J., Buyya R., Economy-Based Data Replication Broker, *Proceedings of the 2nd IEEE International Conference on E-Science and Grid Computing* (E-Science 2006, IEEE CS Press, Los Alamitos, CA, USA), Dec. 4- 6, 2006, Amsterdam, Netherlands.

[13] Nakajima S., A SOAP-based Infrastructure for Service Broker, *Proceedings of XML Technologies and Software Engineering* (XSE2001),Toronto, Canada (co-located with ICSE2001), 15 May 2001

[14] Burst A. J., Forte S., Programowanie Microsoft SQL Server 2005. Microsoft Press, 2006

[15] Medrela D., Potaśiński P., Szeliga M., Widera D., Serwer SQL 2008 Administracja i programowanie, Helion 2009.

[16] Dudek W., Bazy danych SQL Teoria i praktyka, Helion 2006

[17] Troelsen A., Język C# 2008 i platforma .NET 3,5, Helion 2009

[18] Powers L., Snell M., Microsoft Visual Studio 2008 Księga Eksperta, Helion 2009

[19] Turek P., Pelikant A., Replikacje w Oracle, *Zeszyty Naukowe Wyższej Szkoły Informatyki w Łodzi*, Vol. 9, Nr 1, 2010 ss. 95-119.

[20] Pelikant A., Bazy danych w zastosowaniach praktycznych – Monografia WSInf., 2007

[21] Pelikant A., Systemy gromadzenia danych, w Informatyka gospodarcza, red. Janusz Zawiła-Niedźwiecki, Wydawnictwo C.H.Beck 2010, ISBN: 978-83-255-2163-9

[22] Pelikant A., Hurtownie danych. Od przetwarzania analitycznego do raportowania Wydawnictwo Helion 2011, ISBN: 978-83-246-2977-0.

USAGE OF SERVICE BROKER TO AUTOMATIC SYNCHRONIZE OF DATA IN MICROSOFT SQL SERVER

Summary – The main topic of the article is an automatic communication between two twin databases, so that both databases are performed the same question concerning both the modification of data in the existing structure and of relational structure changes. This approach extends the functionality of replication, which allows only for the synchronization of specific data. This work focuses on the technical side of the issue, the creation of the full structure of the Service Broker service (queue, endpoints, services), a conversation between databases, processing messages. This functionality has been obtained by the action of a set of procedures, triggers are created for the database schema, which automatically create triggers the intermediate responsible for synchronization of data. Are also external applications used to track messages that are generated by the triggers and present made changes. Its task is also a redirect to a less biased processing base, in order to improve the efficiency of processing.