STUDIA INFORMATICA 2011
Volume 32 Number 2A (96)

Piotr KOWALSKI, Katarzyna HARĘŻLAK
Politechnika Śląska, Instytut Informatyki

ANALIZA ARCHITEKTUR ZARZĄDZANIA DANYMI W WARUNKACH ROZPROSZENIA GEOGRAFICZNEGO

Streszczenie. W artykule przeprowadzono dyskusję architektonicznych i funkcjonalnych problemów, pojawiających się przy projektowaniu szeroko rozumianych systemów rozproszonych. Zaprezentowano różne, możliwe do zastosowania w tym celu, architektury. Wśród nich znalazły się architektury dwu– i trójwarstwowe, architektura oparta na usługach sieciowych oraz architektura wykorzystująca replikację danych. Dla każdego z rozwiązań omówiono jego wady i zalety.

Słowa kluczowe: architektury systemów rozproszonych, zarządzanie danymi, replikacja danych

THE ANALYSIS OF DISTRIBUTED DATA MANAGEMENT ARCHITECTURS

Summary. The analysis of functional and architectural issues which appear during distributed system designing was discussed in the paper. Various, possible architectures, which can be used in such systems, were also shown. Among them, there can be enumerated two– and three–layered architectures, architectures based on web services and architectures utilizing data replication. Advantages and disadvantages of all of them were presented as well.

Keywords: distributed systems architectures, data management, data replication

1. Wstęp

Informatyzacja w sposób szeroki i ekspansywny pojawia się obecnie w wielu aspektach działalności biznesowej. Bez dobrej jakości systemów informatycznych coraz trudniej wyobrazić sobie poprawne działanie firm w różnych branżach i gałęziach gospodarki oraz różnych obszarach działania, takich jak: produkcja, handel czy usługi, w szczególności prowa-

dzące swą działalność w geograficznie rozproszonych środowiskach. Wyrazistym przykładem może być szeroko rozumiana gałąź sprzedaży towarów. Firmy sprzedające różnego rodzaju produkty, w sposób detaliczny lub hurtowy, za pomocą swoich placówek lub Internetu, chcąc wdrożyć system informatyczny, mają zwykle dwie drogi do wyboru. Pierwszą jest kupno jednego z gotowych systemów, które mają pewien zakres konfigurowalności, jednak bardzo często nie da się ich w pełni dostosować do potrzeb danego przedsiębiorstwa. Drugą drogą jest zaprojektowanie i wykonanie rozwiązania dedykowanego dla danej firmy. Najczęściej będzie ono znacznie droższe od pozostałych, jednak może być w pełni dostosowane do określonych wymagań. W niniejszym artykule przedstawione zostaną funkcjonalne i architektoniczne problemy pojawiające się przy projektowaniu „na miarę" szeroko rozumianych systemów rozproszonych.

2. Elementy oprogramowania systemów rozproszonych

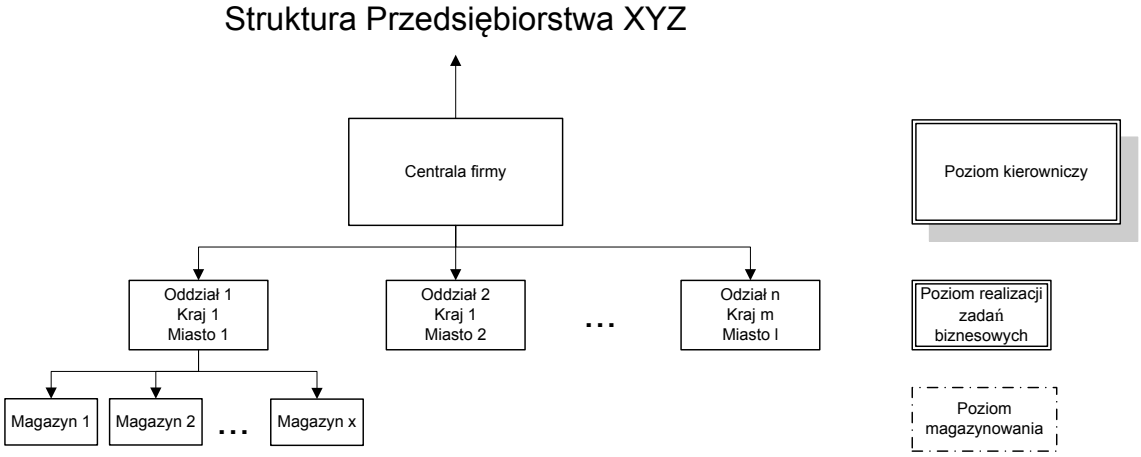
Systemy obsługi przedsiębiorstw o rozproszonej geograficznie strukturze zwykle mogą składać się z wielu modułów, w zależności od wielkości firmy i zakresu zagadnień, jakimi system będzie zarządzał. W tego typu systemach można wyróżnić wiele elementów składowych.

1. Oprogramowanie dla jednostek realizujących podstawową funkcjonalność biznesową – element systemu zwykle w postaci aplikacji „grubego klienta", służącego do zarządzania zadaniami firmy z poziomu określonej placówki (sklepu, hurtowni). Dodatkowo, tego rodzaju klient może udostępniać funkcjonalność zarządzania pozostałymi elementami systemu na różnym poziomie zarządczym oraz poziomie realizacji działań biznesowych. Programy takie wyposażone są także w obsługę drukarek przeznaczonych do różnych zadań (np. drukarki fiskalne, drukarki dokumentów, drukarki etykiet).
2. Oprogramowanie dla obsługi internetowej – strona internetowa umożliwiająca przeglądanie produktów lub usług, dokonywanie zakupów, zamówień oraz dostęp do innych funkcjonalności w trybie on-line. Taka aplikacja WWW może również pełnić rolę „cienkiego klienta", udostępniającego część funkcjonalności zarządzania zadaniami firmy oraz inne czynności administracyjne.
3. Oprogramowanie dla magazynów – na ten element mogą składać się zarówno aplikacje w postaci „grubych klientów", jak i aplikacje dla urządzeń mobilnych. Często magazyny wyposażone są w urządzenia, takie jak:
   - przenośne terminale PDA; rozwijane są dla nich dedykowane aplikacje mobilne, wykorzystujące funkcjonalność tych urządzeń w zakresie czytników kodów kreskowych, czytników kart magnetycznych oraz modułów Wi-Fi;

- stacje z panelami dotykowymi; najczęściej są to terminale będące zwykłymi komputerami klasy PC (z różnymi opcjami w zakresie pojemności HDD, procesora, pamięci RAM itd.), wyposażone w monitor typu touchscreen, na których działa system operacyjny przeznaczony dla komputerów stacjonarnych oraz dedykowana aplikacja, np. typu Windows Forms.

3. Architektury systemów rozproszonych geograficznie

Wśród rozwiązań dedykowanych prezentowanym systemom proponuje się wiele różnych architektur, spośród których można wybrać jedną, najlepiej odpowiadającą potrzebom konkretnej firmy. Dla ujednoliconej analizy można założyć, że firma, planująca wdrożyć system rozproszony ma określoną strukturę, przedstawioną na rysunku 1.



Rys. 1. Struktura przykładowego przedsiębiorstwa
Fig. 1. The structure of an exemplary company

Struktura przedsiębiorstwa jest hierarchiczna. Każdy poziom hierarchii wyznacza osobny typ lokalizacji. Na szczycie znajduje się centrala firmy, zwykle ulokowana w jednym z dużych miast, ze względu na chęć pozyskania szerokiego spektrum kontaktów biznesowych. Niższy poziom stanowią oddziały znajdujące się w różnych miejscowościach. Przy założeniu, że docelowymi grupami odbiorców usług są zarówno znaczący odbiorcy, reprezentujący inne przedsiębiorstwa, jak i klienci indywidualni, wielkość miast, w których ulokowane są placówki może być silnie zróżnicowana. Dodatkowo, jeżeli firma prowadzi działalność międzynarodową, można założyć, że kolejne oddziały będą w różnych krajach. Oddziały prowadzące podstawową działalność biznesową (np. sprzedaż) realizują usługi na podstawie szeroko pojętego poziomu magazynowego, który w założeniu znajduje się w bliskiej odległości. Magazyny stanowią interfejs komunikacji z ewentualnymi podwykonawcami usług (np. dostawcami towaru).

Można pokazać wiele uzasadnień biznesowych, będących tłem dla wytworzenia i wdrożenia rozproszonego systemu informatycznego dla określonego przedsiębiorstwa. Przykła-

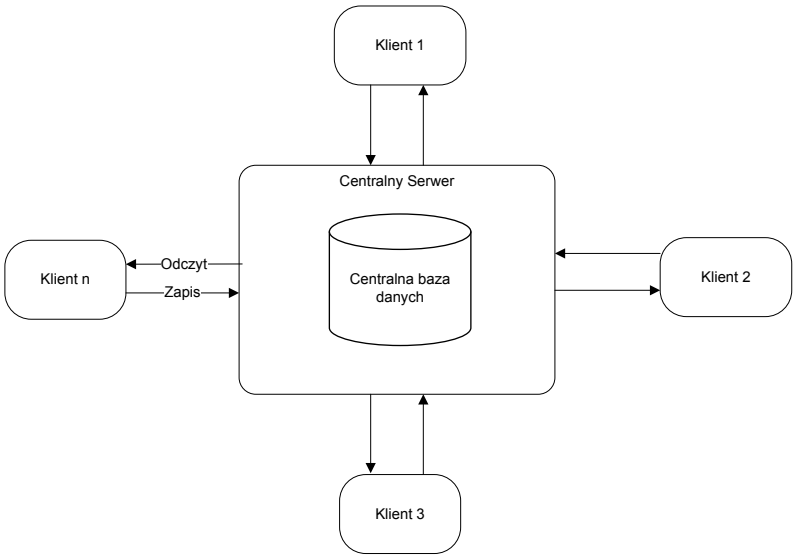
dem może być scenariusz, w którym dana firma ma już działającą sieć w różnych miastach, ale jej placówki są niezależne i nie istnieje żaden elektroniczny system integrujący i koordynujący działanie tych oddziałów. W takim przypadku zaprojektowany system ma za zadanie połączyć ich działanie oraz umożliwić im wymianę danych. Inny przypadek może być związany z chęcią rozszerzenia działalności firmy o usługi udostępniane w Internecie, które będą zintegrowane z działającym lub dopiero rozwijanym lokalnym systemem firmy. Każdy z przypadków będzie wymagał zastosowania odpowiedniej architektury.

Biorąc pod uwagę oczekiwania w stosunku do takich aspektów, jak: wielkość systemu, spektrum funkcjonalności, stopień niezawodności, skalowalności i bezpieczeństwa, można wprowadzić następujący podział architektur:

1) dwuwarstwowa architektura scentralizowana typu klient-serwer,
2) dwuwarstwowa architektura rozproszona z zastosowaniem rozproszonych baz danych,
3) trójwarstwowa architektura z serwerem aplikacji,
4) architektury hybrydowe.

3.1. Dwuwarstwowa architektura scentralizowana

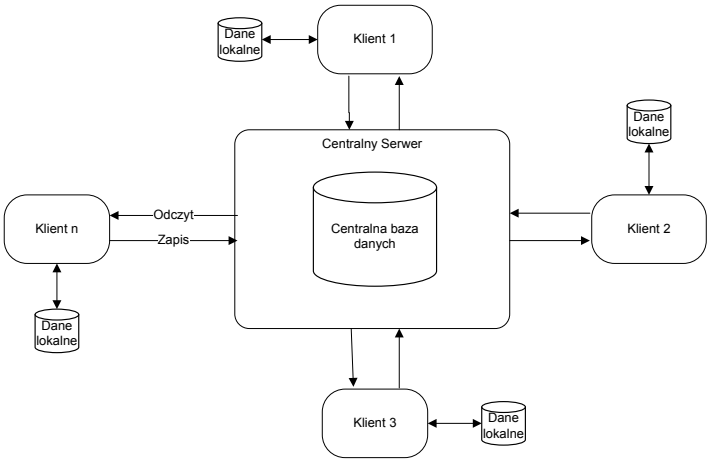
Podstawowym wymaganiem, które postawione jest w rozległym systemie informatycznym, jest możliwość komunikacji i wymiany danych pomiędzy węzłami pracującymi w ramach systemu. Najprostszym architektonicznie podejściem jest uruchomienie silnego centralnego serwera bazy danych, do którego odwołują się rozproszone aplikacje klienckie. Aplikacje klienckie – niezależnie od typu klienta (PC – ang. personal computer, Touchstation – komputer z panelem dotykowym, PDA – ang. personal digital asistant), na którym są uruchomione i typu lokalizacji, w jakiej się znajdują (Oddział, Magazyn) – będą uzyskiwały połączenie bezpośrednio do serwera centralnego. Takie rozwiązanie przedstawiono na rysunku 2.



Rys. 2. Architektura klient - serwer
Fig. 2. The client - server architecture

Jak można się spodziewać, dostęp do centralnego serwera bazy danych może być „wąskim gardłem" systemu. Dodatkowo występuje tutaj wysoka podatność systemu na awarię: awaria centralnego serwera powoduje awarię całego systemu, awarie łącz komunikacyjnych eliminują klientów z udziału w systemie. Ponadto, rozwiązanie to cechuje sie bardzo niską skalowalnością, ograniczoną do rozszerzania mocy obliczeniowej pojedynczego serwera.

Sposobem na wyeliminowanie tych niedogodności jest wprowadzenie na maszynach klienckich lokalnych kopii bazy danych, które będą przechowywać często odczytywane dane. Każdy z klientów będzie posiadał zaimplementowaną logikę, odpowiadającą za pobieranie nowych danych i aktualizację lokalnej bazy danych, co przedstawione jest na rysunku 3.



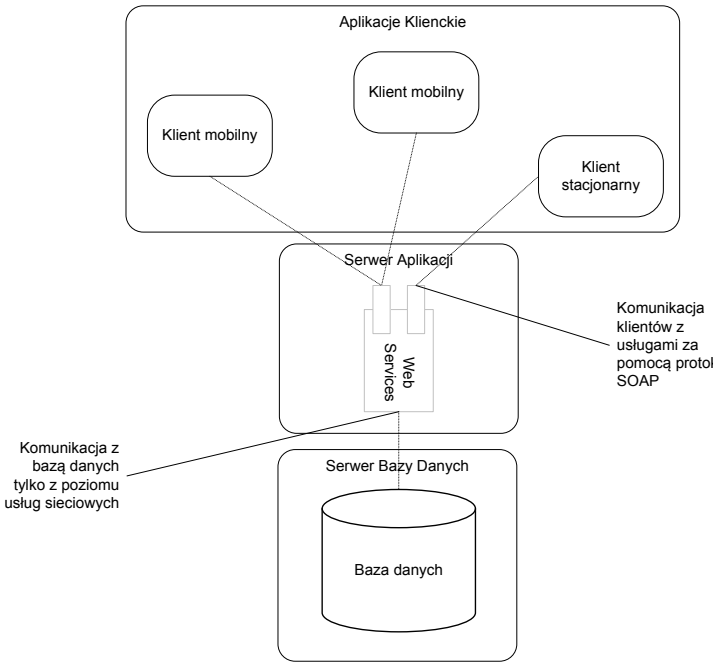
Rys. 3. Zmodyfikowana architektura klient - serwer
Fig. 3. The modified client - server architecture

Wadą tego rozwiązania jest fakt, że logikę odpowiadającą za pobieranie danych należy zaimplementować wprost w aplikacjach klienckich. Implementacja tego typu mechanizmu, bez użycia gotowych rozwiązań, nie jest zadaniem trywialnym i często może okazać się nieopłacalna z punktu widzenia biznesowego. Szczególnie istotną wadą tego mechanizmu jest konieczność zmian kodu aplikacji w przypadku, gdy na etapie implementacji pojawiają się zmiany w schemacie centralnej bazy danych, rzutujące na lokalne repliki. Do zalet rozwiązania można zaliczyć natomiast niski poziom skomplikowania architektury, wpływający na łatwiejsze projektowanie, konfigurację i zarządzanie systemem oraz niezależność od gotowych rozwiązań, zapewniających możliwości synchronizacji danych.

Jedną z cech tego podejścia jest również brak podziału aplikacji na moduły. Wadę tę można wyeliminować, stosując jedną z technologii implementacji aplikacji rozproszonych, jaką jest technologia Web Services. Architektura oparta na tej technologii zostanie zaprezentowana w kolejnym podrozdziale.

3.2. Architektura z użyciem Usług Sieciowych

Technologia Usług Sieciowych (ang. Web Services) jest jedną z wielu technologii umożliwiających komunikację oprogramowania poprzez sieć komputerową w sposób podobny do wymiany informacji pomiędzy procesami, działającymi w ramach jednego komputera. Technologia oparta jest na otwartych, wykorzystujących język XML, standardach, regulowanych przez dwie organizacje: W3C i OASIS [1]. Dzięki temu możliwa jest współpraca aplikacji zaimplementowanych w różnych językach i środowiskach (np. Phyton, Java, C#), dedykowanych różnym platformom systemowym (np. Windows, Linux). Struktura Usług Sieciowych opisywana jest za pomocą języka WSDL, a komunikacja z usługami odbywa się poprzez wymianę wiadomości (ang. messages), opisanych w języku SOAP [1]. Wymiana ta odbywa się najczęściej przy użyciu komunikacji HTTP, dzięki czemu eliminuje się utrudnienia komunikacji, wynikające z istnienia w sieci komputerowej aplikacji typu firewall.



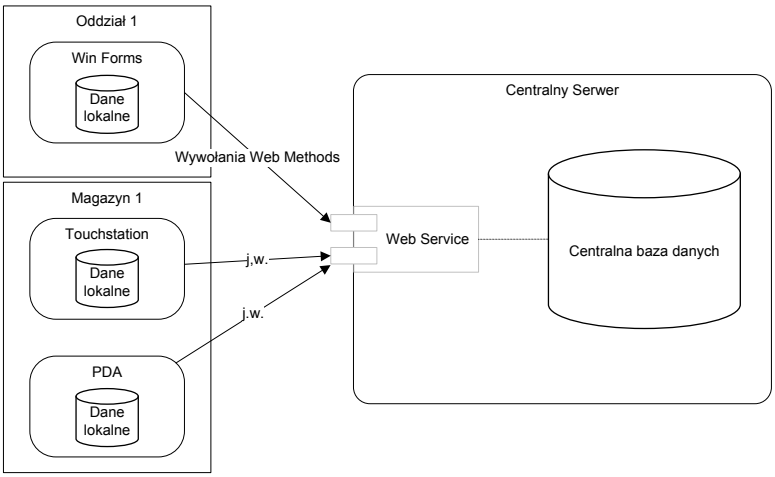
Rys. 4. Architektura trójwarstwowa
Fig. 4. The tree–layered architecture

Usługi sieciowe, poprzez swoją elastyczność, zyskały obecnie dominujący udział w architekturach aplikacji biznesowych. Z ich pomocą tworzone są systemy oparte na idei architektury zorientowanej na usługi (ang. Service Oriented Architecture) [2]. Wprowadzone zostało pojęcie architektury trójwarstwowej, zawierającej dodatkowy moduł usług sieciowych (rys. 4). W module tym znajduje się logika biznesowa systemu, która jest wykorzystywana przez różne aplikacje klienckie, poprzez zdalne wywołanie tzw. metod sieciowych (ang. Web Methods) [2]. Służą one do wymiany danych za pomocą obiektów serializowanych do języka XML. Przy takim podejściu aplikacje klienckie, niezależnie od ich typu (klient stacjonarny, mobilny, „gru-

by klient", „cienki klient"), zawierają jedynie interfejsy użytkownika i cienkie warstwy, odpowiadające za komunikację z obiektami Web Services.

Usługi sieciowe działają w formie aplikacji na dedykowanym dla nich serwerze aplikacji. Dla środowiska .NET [3] serwerem aplikacji jest serwer IIS [4], w środowisku Java dostępne są różne serwery aplikacji, np. IBM Websphere [5]. Odwołania do centralnej bazy danych mają miejsce jedynie z poziomu obiektów Web Services. Wprowadza to duże uporządkowanie w systemie i sprawia, że aplikacje klienckie korzystają jedynie z udostępnionego im interfejsu usług sieciowych.

W ogólności utworzone i opublikowane usługi sieciowe mogą być udostępnione w rejestrze wyszukiwania usług poprzez mechanizm UDDI (ang. Universal Description, Discovery and Integration) [1]. Przy zastosowaniu mechanizmu Web Services, w przypadku rozważanego systemu, można zaproponować strukturę zaprezentowaną na rysunku 5.



Rys. 5. Architektura oparta na usługach sieciowych
Fig. 5. The architecture based on web services

Centralny serwer składa się z dwóch serwerów: serwera bazy danych i serwera aplikacji z udostępnionymi usługami sieciowymi. Aplikacje klienckie w oddziałach i podlegającym im magazynach mają do dyspozycji dedykowane interfejsy mechanizmu Web Services. Za ich pomocą dokonywany jest przepływ danych w kierunku centralnego serwera i umieszczonej tam bazy danych (zapis). Dodatkowo, każda z aplikacji klienckich ma lokalną bazę danych służącą do przechowywania często odczytywanych danych, związanych z określonym oddziałem lub magazynem. Aktualizacja danych lokalnych będzie miała miejsce przy użyciu specjalnie zaimplementowanych metod na poziomie usług sieciowych. Interfejs, służący synchronizacji, będzie więc znajdował się w jednym miejscu i będzie wspólny dla aplikacji klienckich danego typu. Doświadczenia z wykorzystania protokołu SOAP wskazują, że komunikacja HTTP (przy jego użyciu) może być bardzo opłacalna pod względem wydajnościowym. Ma to kluczowe znaczenie przy podejmowaniu decyzji o implementacji własnego mechanizmu aktualizacji lokalnych replik. W przypadku rozwiązania opartego na serwerze

aplikacji IIS firmy Microsoft, dla tego protokołu, udostępniona jest opcja kompresji wiadomości HTTP. Na platformie .NET kompresowane są zarówno komunikaty typu *Request*, jak i *Response*.

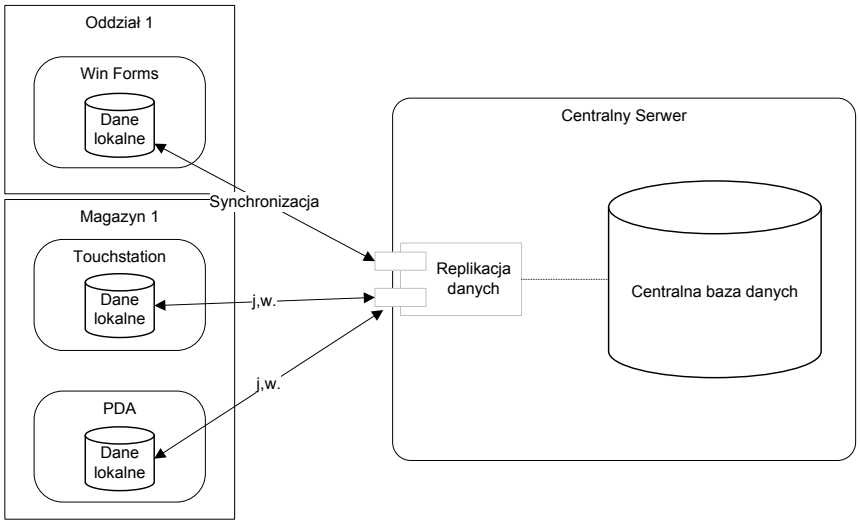
Opisywana architektura, wspólnie z architekturami przedstawianymi wcześniej, wskazuje na problem operowania danymi z poziomu różnych elementów systemu i fakt, że dane mogą być podzielone na logiczne porcje, odpowiadające określonym lokalizacjom. Ułatwienie dostępu z poziomu węzłów znajdujących się na niższych poziomach hierarchii, do powiązanych z nimi danych, zostało osiągnięte poprzez wprowadzenie lokalnych baz danych, zawierających repliki często odczytywanych rekordów. Rozwiązanie takie zawiera jednak pewne, kluczowe wady. Istotnym elementem jest konieczność aktualizacji lokalnych replik, która spoczywa na logice aplikacji. Dodatkowo, przy takim podejściu na danym poziomie hierarchii brak jest pełnej autonomiczności węzłów. Wszelkie czynności realizowane z poziomu oddziału firmy, które wymagają przepływu danych w kierunku serwera centralnego i zapisania ich w centralnej bazie danych, muszą mieć dostępne połączenie z tym serwerem. W przypadku braku połączenia, do czasu usunięcia awarii łącza, dany oddział nie jest w stanie wykonywać istotnych zadań. Ponadto, w architekturze tej brakuje pełnego odzwierciedlenia struktury firmy w lokalizacji składowania poszczególnych danych. Wady te są wyeliminowane w architekturze za pomocą replikacji, opisywanej w kolejnym podrozdziale.

3.3. Architektura z użyciem replikacji danych

Rozwiązaniem proponowanym w tym podrozdziale jest zastosowanie replikacji danych. Replikacja danych będzie służyć do przemieszczania danych oraz obiektów bazodanowych w zdecentralizowanej strukturze firmy, w taki sposób, że dane w poszczególnych węzłach będą względem siebie aktualne i spójne. Zestaw mechanizmów, który kryje się pod wyrażeniem „replikacji danych” zapewnia, więc synchronizację wszystkich baz danych w rozproszonej strukturze. Synchronizacja ta jest wykonywana na poziomie systemów zarządzania bazami danych. Producenci takich systemów oferują gotowe i całościowe rozwiązania, które mogą być wykorzystane do budowy omawianego systemu, co eliminuje konieczność implementacji własnej logiki.

Schemat przedstawiony na rysunku 6 prezentuje wymianę danych pomiędzy centralnym serwerem, oddziałami i magazynami za pomocą replikacji. Każdy z węzłów, niezależnie od poziomu hierarchii, jest autonomiczny. Odczytuje on i zapisuje dane tylko względem swojej lokalnej bazy danych. Lokalna baza danych zawiera wyłącznie partycję danych, związanych z daną lokalizacją. Widać wyraźnie zysk takiego rozwiązania, które eliminuje konieczność odwołań do odległych lokalizacji (w przypadku odczytu i zapisu dla pierwszego wariantu klient–serwer lub w przypadku zapisu dla drugiego wariantu klient-serwer oraz systemu opartego na usługach sieciowych). Dzieje się tak dlatego, że aplikacja kliencka może korzy-

stać z lokalnej repliki, która posiada zsynchronizowane dane. Zapis danych do lokalnej bazy danych jest odzwierciedlany w pozostałych lokalizacjach w momencie synchronizacji. Synchronizacja taka może odbywać się w sposób synchroniczny (ang. *synchronious replication*, *eager replication*) lub asynchroniczny (ang. *asynchronious replication*, *lazy replication*). Pierwszy typ propagacji zmian bazuje na wykonywaniu transakcji rozproszonych [7, 8]. Oznacza to, że zmiany dokonywane w danym węźle zostaną przeniesione w czasie rzeczywistym (a właściwie w „czasie transakcyjnym") na pozostałe węzły w systemie. Drugi typ propagacji zmian zakłada dokonywanie modyfikacji tylko w węźle, na którym operacja została wykonana i uruchomienie synchronizacji na żądanie lub w sposób okresowy.



Rys. 6. Architektura z wykorzystaniem replikacji
Fig. 6. Architecture utilizing data replication

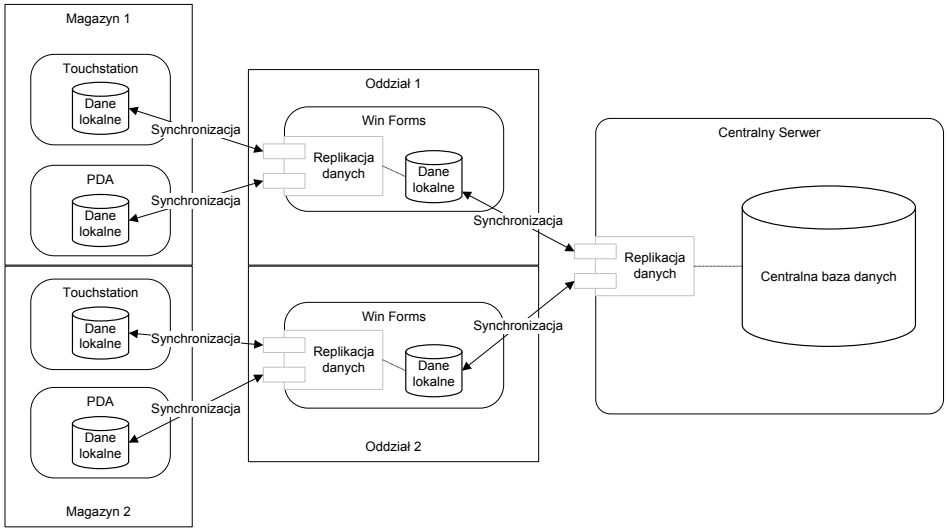
Z punktu widzenia omawianego systemu, tryb synchroniczny może znaleźć zastosowanie jedynie w przypadku połączenia pomiędzy centralnym serwerem a oddziałami. Może to być uzasadnione w przypadku, gdy oczekiwana jest natychmiastowa spójność danych na serwerze względem operacji wykonywanych przez pracowników. Dla części systemu, którą stanowi poziom magazynów, bardziej naturalnym sposobem synchronizacji jest synchronizacja asynchroniczna. Wynika to z faktu, że wykorzystanie klientów mobilnych lub paneli dotykowych będzie odbywać się przy braku połączenia sieciowego z serwerem. Urządzenia te mogą być odłączone, a praca z nimi będzie opierała się jedynie na danych zgromadzonych w lokalnych bazach danych. W takim przypadku, należy umożliwić użytkownikom modyfikację danych, których propagacja na inne węzły nastąpi w momencie, kiedy połączenie z siecią zostanie przywrócone.

Topologia replikacji w różnych systemach zarządzania bazami danych zakłada model *wydawca – subskrybent*, w którym wydawcę stanowi komputer na szczycie hierarchii, a subskrybentów stanowią komputery podłączone do wydawcy i synchronizujące z nim dane. Zmiany możliwe są również na komputerach subskrybentów i są one propagowane do bazy

danych wydawcy podczas synchronizacji. Widać wyraźnie, że w tym miejscu mogą pojawić się różne konflikty, wynikające z współbieżnych zmian tych samych danych na różnych komputerach subskrybentów, co stawi podstawowy problem w replikacji danych [6].

Struktura przedstawiona na rysunku 1 zakłada, że system jest podzielony na jednego wydawcę oraz wielu subskrybentów. Wydawcą jest centralny serwer, a subskrybenci reprezentowani są przez pozostałe komputery w systemie, na poziomach oddziałów oraz magazynów. Nie jest to jedyna możliwa topologia, która może zostać zaproponowana. Na rysunku 7 zamieszczona została architektura, w której subskrybentami centralnego serwera są jedynie komputery na poziomie oddziałów firmy. Komputery te są jednocześnie wydawcami dla subskrybentów na poziomie magazynów. Jest to model tzw. powtórnej publikacji (ang. republishing), która pozwala na lepsze odzwierciedlenie geograficznego rozproszenia węzłów w strukturze systemu.

Zastosowanie mechanizmu powtórnej publikacji we wskazanym przypadku jest opłacalne, ponieważ odległość geograficzna wydawcy głównego od skupionej grupy subskrybentów jest znacząca. Dane są synchronizowane z jednym z subskrybentów, który staje się jednocześnie wydawcą dla pozostałych subskrybentów lokalnych. Takie rozwiązanie znacznie ogranicza koszt użycia łącza pomiędzy odległymi lokalizacjami. Dodatkowym zyskiem jest rozłożenie obciążenia na wielu wydawców, eliminujące konieczność użycia bardzo silnego serwera centralnego. Wraz z zejściem na niższe poziomy hierarchii systemu, moc obliczeniowa komputerów może być mniejsza, ponieważ objętość partycji danych maleje.

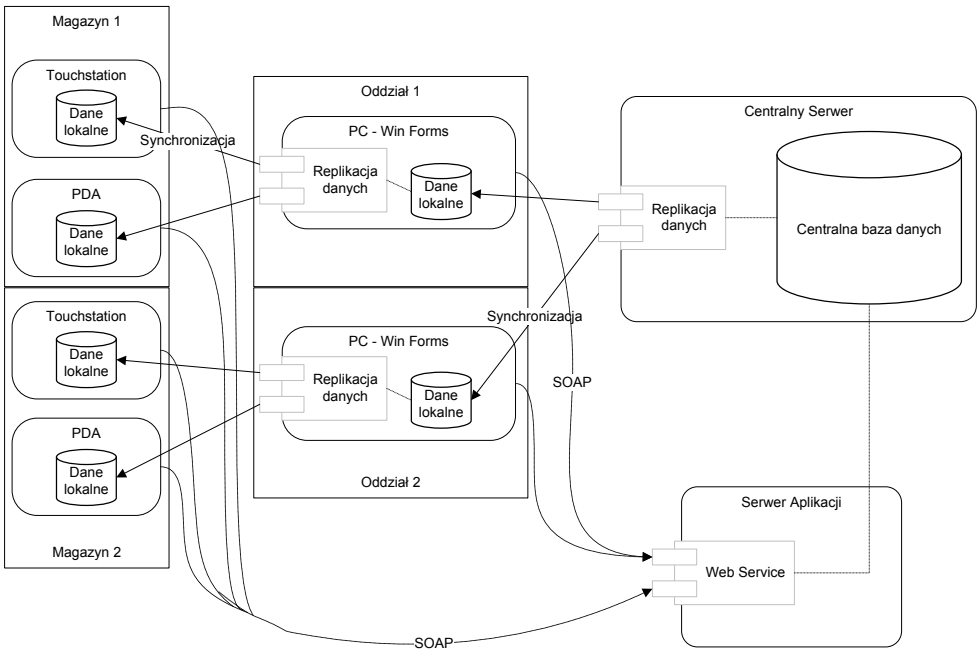


Rys. 7. Architektura z wykorzystaniem replikacji - wersja republishing
Fig. 7. Architecture utilizing data replication in republishing mode

3.4. Architektura hybrydowa

W niektórych przypadkach opłacalne staje się połączenie zalet mechanizmów replikacji danych i usług sieciowych w jednym systemie. Można wyobrazić sobie sytuację, w której

wymaganiem w projektowaniu systemu jest metodologia SOA opierająca się na Web Services. Cała logika wykonywanych przez użytkowników operacji na poziomach oddziałów oraz magazynów realizowana jest poprzez interfejs WWW (rys. 8). Zaletami takiego podejścia jest podział aplikacji na moduły oraz możliwość wykorzystania heterogenicznych klientów i platform systemowych. Aplikacje klienckie potrzebują jednak pewnych informacji o obiektach znajdujących się w centralnej bazie danych. Dostarczenie tych danych można zrealizować na podstawie replikacji generującej przepływ tylko w dolnym kierunku hierarchicznej struktury systemu. Takie rozwiązanie, dodatkowo oparte na partycjonowaniu danych, nie generuje problemu konfliktów podczas procesu synchronizacji. Implementacja logiki biznesowej jest znacznie łatwiejsza, niż ma to miejsce w przypadku replikacji dwustronnej. Sprowadza się ona do implementacji odpowiednich metod na poziomie Web Services, podczas gdy wyłączne użycie mechanizmów replikacji wymaga implementacji logiki, zwykle na poziomie procedur składowanych, która jest często bardziej skomplikowanym zadaniem.



Rys. 8. Architektura hybrydowa
Fig. 8. Hybrid architecture

4. Podsumowanie

W niniejszym artykule opisane zostały różne podejścia do projektowania systemów rozległych. Pierwsze podejście oparte na centralnej bazie danych i aplikacjach klienckich charakteryzuje się największa prostotą w projektowaniu, implementacji i administracji w porównaniu do reszty rozwiązań. Nie jest to jednak odpowiednie rozwiązanie dla systemów rozproszonych geograficznie, dla których zdecydowanie lepszym podejściem jest architektura roz-

proszona, prezentowana w pozostałych rozwiązaniach. Dzieje się tak z kilku powodów. Duże firmy, których obszar działań biznesowych cechuje większa skala, np. międzynarodowa, posiadają zwykle wiele placówek rozłożonych w bardzo odległych od siebie geograficznie miejscach. Można stwierdzić, że zastosowanie (w takim przypadku) scentralizowanego serwera bazy danych generuje duże narzuty komunikacji względem wydajności oraz zbyt duże ryzyko awarii (np. łącz komunikacyjnych lub samego serwera). Kolejnym argumentem przemawiającym za architekturą rozproszoną jest próba zwiększenia wydajności całego systemu, polegającej na balansie obciążenia, które będzie generowane w miejscu centralnego serwera bazy danych. Przy użyciu takiego wariantu zastosowanych będzie wiele serwerów, które parametrami będą mogły ustępować silnemu serwerowi z rozwiązania scentralizowanego, a mimo to pojawi się wzrost wydajności. W takim scenariuszu niejednokrotnie może się okazać, że rozwiązanie to jest tańsze od rozwiązania scentralizowanego, ponieważ koszt zakupu jednego silnego serwera przewyższyłby koszt zakupu wielu, znacznie słabszych komputerów.

Każda z przedstawionych w artykule architektur, pomimo istnienia opisanych zalet i wad, może być z powodzeniem wykorzystywana w omawianym typie systemów. Czynnikiem, który decyduje czy dane rozwiązanie sprosta wymaganiom są oczekiwania względem wielkości systemu i jego perspektyw do ekspansji. Podstawowym wnioskiem wynikającym z powyższych rozważań jest fakt, że zdolności do rozwoju możliwości systemu dają kolejne, bardziej złożone architektury rozproszone z rozproszonymi bazami danych. Dzieje się tak, dlatego że skalowalność systemów scentralizowanych i ich zdolność do powiększania poziomu bezpieczeństwa ogranicza się do jednego punktu w systemie. Wszystkie mechanizmy równoważenia obciążenia oraz zapewniania wysokiej dostępności będą w takim przypadku realizowane w jednym, geograficznie położonym miejscu, w odniesieniu do jednego węzła systemu. Na drugim biegunie, w stosunku do tego rozwiązania, pozostają architektury wykorzystujące rozproszone bazy danych. Pozwalają one uzyskiwać wysoką skalowalność poprzez rozpartycjonowanie danych w węzłach. Działanie to jest niezależne od wspomnianych, możliwych mechanizmów równoważenia obciążenia oraz zapewniania wysokiej dostępności, które dalej mają zastosowanie, ale w tym przypadku dotyczą każdego z rozproszonych węzłów. W projektowaniu dużych systemów kluczem do sukcesu pozostaje idea rozproszonych baz danych zapewniająca skalowalność podążającą za naturalnym, hierarchicznym rozproszeniem w strukturach firm, dla których systemy te są tworzone.

Praca naukowa finansowana ze środków na naukę w latach 2010-2012 jako projekt rozwojowy nr O R00 0113 12

BIBLIOGRAFIA

1. Governor J., Hinchcliffe D., Nickull D.: Web 2.0 Architectures: O'Reilly Media / Adobe Dev Library, 2009.
2. Service-oriented architecture (SOA) definition. Web Services and Service-Oriented Architectures. [Online] Barry & Associates, Inc. [Cited: 04 25, 2010.] http://www.service-architecture.com/web-services/articles/service-oriented_architecture_soa_definition.html.
3. Templeman J., Vitter D.: Visual Studio .NET: .NET Framework. Czarna księga. HELION, Gliwice 2003.
4. Schaefer, Kenneth, et al. Professional IIS 7. s.l. : Wrox, 2008.
5. WebSphere software. IBM. [Online] IBM. [Cited: 04 2010, 25.] http://www-01.ibm.com/software/websphere/.
6. Ceri S., Pelagatti G.: Distributed Databases. Principles and Systems. McGraw–Hill Book Company, 1984.
7. Ceri S., Houtsma M. A.W., Keller A. M., Samarati P.: A Classification of Update Methods for Replicated Databases. Computer Science Technical Report STAN-CS-91-1392, October 1991.
8. Bersnstein P. A., Hadzilacos V., Goodman N.: Concurrency Control and Recovery in Database Systems. Addison–Wesley, 1987.

Recenzenci: Dr inż. Dariusz Mrozek
Prof. dr hab. inż. Stanisław Wrycza

Wpłynęło do Redakcji 16 stycznia 2011 r.

Abstract

The analysis of functional and architectural issues which appear during distributed system designing was discussed in the paper. Various, possible architectures, which can be used in such systems, were also shown. Among them, there can be enumerated two– and three–layered architectures, architectures based on web services and architectures utilizing data replication. Advantages and disadvantages of all of them were presented as well.

The simplest architectural approach is to use of one the database server and a number of distributed client applications. However, in this solution, a database server can be a bottle–neck of the whole system and makes it more susceptible to a failure by server or network connections breakdown. The solution for eliminating these inconveniences is to equip every

client computer with local database storing often read data. A drawback of this idea is the necessity of extending a client application functionality with data loading logic.

Another presented architecture uses web service mechanisms for data flow. In this way, every client's application has its own database while the synchronization logic is accessible in one place. Next discussed methods for data management in distributed environment concerned data replication. Usage of such solutions in combination with independent method of synchronization guarantees autonomy of each system node, but in some case may cause inconsistency of data in given interval of time.

Studies concerning solutions mentioned above have shown that all of them can be used, with success, in systems managing data in distributed environment. Determinants deciding if given architecture meets requirements are size of a designed system and planes regarding its expansion.

Adresy

Piotr KOWALSKI: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, p.kowalski.poland@gmail.com.
Katarzyna HARĘŻLAK: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, katarzyna.harezlak@polsl.pl.