Zalety i wady architektury rozproszonej wykorzystującej migawki tylko do odczytu

Marek Wojciechowski

Instytut Informatyki

Politechnika Poznańska

ul. Piotrowo 3a, 60-965 Poznań

marek@cs.put.poznan.pl

Streszczenie

Niniejszy artykuł przedstawia zalety i wady architektury rozproszonej wykorzystującej replikację danych opartą na migawkach tylko do odczytu (ang. read-only snapshots) w stosunku do architektury scentralizowanej oraz rozproszonej bez replikacji. Omówiono w nim kryteria, które należy uwzględnić przy wyborze architektury systemu oraz jej wpływ na funkcjonalność, niezawodność, skalowalność, efektywność przetwarzania i poufność danych. Artykuł jest wynikiem rozważań i analiz prowadzonych podczas pracy nad projektem systemu informatycznego dla szpitali.

1. Wstęp

Systemy scentralizowane, w których wszystkie dane znajdują się na jednym serwerze, są zdecydowanie łatwiejsze w projektowaniu, konfiguracji i zarządzaniu od rozproszonych. Nie oznacza to jednak, że scentralizowana architektura bazy danych jest w każdym przypadku właściwa. Systemy rozproszonych baz danych znajdują zastosowanie z dwóch podstawowych powodów. Pierwszy z nich to fakt, że wiele informatyzowanych jednostek (przedsiębiorstw, urzędów, szpitali, itp.) ma ze swej natury charakter rozproszony. W przypadku, gdy firma posiada swoje oddziały w wielu miastach niedopuszczalnym rozwiązaniem byłoby oparcie systemu informatycznego na jednym serwerze bazy danych, ponieważ system taki byłby podatny na awarie łączy komunikacyjnych, a ponadto nawet dostęp do danych o charakterze lokalnym wymagałby odwołań do zdalnego serwera. Z drugiej strony wykorzystywanie przez poszczególne jednostki organizacyjne przedsiębiorstwa w pełni autonomicznych systemów baz danych, między którymi nie ma żadnych powiązań, również nie jest rozwiązaniem idealnym, gdyż w większości przypadków oprócz danych o charakterze lokalnym, występują dane współdzielone przez wszystkie jednostki, bądź udostępniane przez pewne jednostki innym. W takich sytuacjach najlepszym wyjściem jest wykorzystywanie systemów rozproszonych baz danych.

Drugim ważnym powodem, dla którego projektuje się systemy o architekturze rozproszonej jest potrzeba zwiększenia mocy przetwarzania. Obecnie systemy informatyczne przetwarzają ogromne ilości danych, a ponadto w większości przypadków ilość pamiętanych danych zwiększa się w trakcie użytkowania systemu. Dlatego też często nawet w sytuacji gdy możliwe byłoby wykorzystanie jednego serwera, dokonuje się rozproszenia danych na kilku mniej obciążonych serwerach.

Podstawowe problemy związane z projektowaniem i eksploatacją systemów rozproszonych baz danych wynikają z konieczności wykorzystywania sieci komputerowych przy komunikowaniu się serwerów ze sobą. Żadne łącza transmisyjne nie są niezawodne, w związku z czym zawsze istnieje niebezpieczeństwo, że dane zgromadzone na niektórych serwerach będą przez pewien czas niedostępne. Dodatkowo czas odpowiedzi systemu w przypadku odwoływania się do zdalnych danych może być znacznie dłuższy niż podczas korzystania z danych lokalnych. Prawdopodobieństwo wystąpienia tego typu problemów jest tym większe im wolniejsze i mniej stabilne są połączenia sieciowe.

Jak wspomniano wcześniej, rozproszenie danych może mieć na celu rozłożenie obciążenia na kilka serwerów. Jednakże mimo rozproszenia danych, serwery udostępniające dane, z których korzystają wszystkie jednostki organizacyjne mogą w dalszym ciągu być nadmiernie obciążone.

Pewnym rozwiązaniem wspomnianych wyżej problemów związanych z rozproszeniem danych jest replikacja. Utrzymywanie kopii danych na wszystkich serwerach, do których kierowane są żądania dostępu do tych danych, jest sposobem na zwiększenie dostępności danych, przy jednoczesnym skróceniu czasu odpowiedzi.

Oprogramowanie Oracle Server silnie wspiera replikację. Jednakże bogactwo oferowanych mechanizmów zależy od tego, jakie dodatkowe opcje zostaną zakupione wraz z serwerem. Replikacja podstawowa bazująca na migawkach tylko do odczytu dostępna jest w wersji serwera z PL/SQL i opcją rozproszoną (*ang. Oracle Server with the distributed option*). Bardziej zaawansowane mechanizmy oferowane są dopiero w wersji serwera uzupełnionej o opcję zaawansowanej replikacji (*ang. Oracle Server with the distributed and advanced replication options*).

Migawki tylko do odczytu są dostępne już wraz z opcją rozproszoną, ich konfigurowanie nie jest skomplikowane, a odpowiednie ich wykorzystanie pozwala na złagodzenie niedogodności związanych z faktem rozproszenia danych. Jednak w konkretnych zastosowaniach, ze względu na swą naturę, migawki mogą wpływać niekorzystnie na funkcjonalność i własności systemu informatycznego.

Struktura niniejszego artykułu jest następująca. Rozdział drugi zawiera porównanie różnych architektur pokazując ich zalety i wady. Rozdział trzeci poświęcony jest w całości migawkom tylko do odczytu, ich funkcjonalności oraz sugerowanym zastosowaniom. Rozdział czwarty pokazuje konkretne problemy związane z wykorzystaniem migawek. Rozdział piąty zawiera podsumowanie oraz wskazówki, którymi należy kierować się przy wyborze architektury systemu.

2. Architektury systemów baz danych

Podstawowym modelem przetwarzania stosowanym we współczesnych systemach informatycznych jest model klient-serwer (*ang. client-server model*). W modelu tym przetwarzanie z definicji ma charakter rozproszony i jest realizowane na wielu węzłach komunikujących się ze sobą za pośrednictwem sieci komputerowej. W przypadku systemów baz danych wykorzystujących oprogramowanie firmy Oracle, serwerem jest maszyna, na której pracuje system zarządzania bazą danych Oracle Server. Podstawowym zadaniem aplikacji klientów jest komunikacja z użytkownikiem obejmująca przyjmowanie zapytań do bazy danych oraz prezentację wyników. Należy jednak podkreślić, że część przetwarzania odbywa się również na klientach, co jest jedną z zalet modelu klient-serwer, gdyż pozwala odciążyć serwery powierzając im jedynie zarządzanie danymi.

System bazy danych jest określany jako rozproszony, gdy nie tylko przetwarzanie danych odbywa się na wielu węzłach, ale również dane są rozmieszczone na wielu serwerach. W przypadku gdy wszystkie dane znajdują się na jednym serwerze, mamy do czynienia ze scentralizowaną architekturą bazy danych.

Zarówno architektura scentralizowana jak i rozproszona mają swoje zalety. Jednakże, jak wspomniano we wstępie, niekiedy geograficzne rozproszenie informatyzowanej organizacji narzuca wybór architektury rozproszonej. Inaczej wygląda sytuacja w przypadku jednostek, które w całości objęte są jedną stabilną siecią lokalną. Wtedy do wyboru architektury rozproszonej mogą skłonić następujące jej zalety:

- rozłożenie obciążenia – każdy z serwerów jest odpowiedzialny za zarządzanie częścią danych, co prowadzi do ich mniejszego obciążenia, a w związku z tym skrócenia czasu odpowiedzi systemu,
- łatwiejsza skalowalność – oprócz skalowalności pionowej polegającej na wymianie serwera na mocniejszy, możliwa jest skalowalność pozioma sprowadzająca się do dodania do systemu nowego serwera (taka operacja może wiązać się z koniecznością rekonfiguracji systemu, jest jednak na pewno mniej kosztowna w przypadku systemu, który od początku swego istnienia zakładał rozproszenie danych).

Architektura rozproszona ma również pewne wady, które mogą przemawiać na korzyść architektury scentralizowanej. Problemy związane z rozproszeniem danych to:

- większy koszt systemu – rozproszenie danych wymaga większej liczby komputerów zdolnych do pełnienia funkcji serwera,
- trudniejsze odtwarzanie spójnego stanu systemu po awarii – w przypadku awarii jednego z serwerów, po odtworzeniu jego stanu na podstawie kopii zapasowej stan rozproszonej bazy danych może nie być spójny.

Powyższe niedogodności oczywiście nie dyskwalifikują architektury rozproszonej. Jeżeli chodzi o koszt systemu, to w pewnym stopniu fakt rozproszenia danych zmniejsza wymagania sprzętowe stawiane serwerom, w związku z czym w konkretnych przypadkach może okazać się, że zastąpienie jednego mocnego serwera kilkoma słabszymi jest korzystne także ze względów finansowych. Z kolei proces odtwarzania stanu systemu po awarii (*ang. recovery*), chociaż skomplikowany w przypadku rozproszonych baz danych, jest silnie wspomagany

przez Oracle Server np. poprzez wykorzystywanie dzienników powtórzeń (*ang. redo logs*), co może pozwolić na odtworzenie stanu systemu z chwili tuż przed wystąpieniem awarii.

Istotnym kryterium oceny systemu jest jego podatność na awarie. W przypadku architektury scentralizowanej, w której wszystkie dane przedsiębiorstwa znajdują się na jednym serwerze, każda awaria serwera powoduje całkowitą niemożność korzystania z systemu informatycznego we wszystkich jednostkach organizacyjnych przedsiębiorstwa. Natomiast jeżeli system przedsiębiorstwa oparty jest na rozproszonej bazie danych i każda z jednostek organizacyjnych posiada własny serwer zarządzający lokalnymi danymi a jedynie podczas wykonywania niektórych operacji konieczne jest odwołanie do innych serwerów, to awaria dowolnego serwera zatrzyma pracę jedynie w jednej z jednostek, a pozostałe odczują tylko ograniczenie funkcjonalności systemu. Należy jednak zwrócić uwagę, że z punktu widzenia każdej z jednostek szansa wystąpienia awarii pozbawiającej ją dostępu do własnych danych nie jest mniejsza niż w przypadku systemu z jednym serwerem (może być większa ze względu na to, że często zakup kilku serwerów jest alternatywą dla jednego mocniejszego i bardziej niezawodnego serwera), a dodatkowo pojawiają się możliwości częściowej utraty funkcjonalności związane z awariami innych serwerów.

Aby chociaż w pewnym stopniu uodpornić system na niebezpieczeństwo utraty połączenia ze zdalnymi serwerami wykorzystuje się replikację danych. Replikacja polega na utrzymywaniu wielu kopii tych samych danych (np. wszędzie tam gdzie są one wykorzystywane). Podstawowe zadanie jakie stoi przed systemem zarządzania bazą danych oferującym replikację to utrzymywanie spójnego stanu replik. W zależności od sposobu propagowania zmian między replikami, replikacja może być synchroniczna lub asynchroniczna. W przypadku replikacji synchronicznej zmiany dokonane na innych stanowiskach są natychmiast odzwierciedlane lokalnie (modyfikacja wszystkich replik odbywa się w ramach jednej transakcji). Replikacja asynchroniczna gwarantuje, że zmiana jednej z replik po pewnym czasie będzie propagowana do pozostałych replik, ale nie musi to nastąpić natychmiast (transakcja modyfikująca jedną z replik może się zakończyć przed uspójnieniem stanu wszystkich replik).

Architektura rozproszona z replikacją danych ma następujące zalety w stosunku do architektury rozproszonej bez replikacji:

- odwołania do zdalnych danych mogą być zastąpione odwołaniami do ich lokalnych replik, co może spowodować skrócenie czasu odpowiedzi (szczególnie w przypadku systemów, w których serwery komunikują się ze sobą poprzez sieci rozległe), a także rozłożenie obciążenia serwerów przy dostępie do współdzielonych w systemie danych,
- gdy niektóre stanowiska są niedostępne np. z powodu awarii sieci, możliwe jest w dalszym ciągu operowanie na kopiach danych.

Stosowanie replikacji ma jednak pewne wady. Replikacja synchroniczna może uniemożliwiać modyfikacje replikowanych danych w sytuacji gdy którakolwiek z replik jest niedostępna. Z kolei replikacja asynchroniczna wiąże się z zezwoleniem na okresową niespójność replik. W związku z powyższym replikacja synchroniczna znajduje zastosowanie w systemach, w których wymagana jest absolutna spójność replik, a połączenia sieciowe są niezawodne. Natomiast wykorzystanie replikacji asynchronicznej jest celowe w przypadkach, gdzie istotna jest przede wszystkim dostępność danych, a nie jest wymagana absolutna spójność replik.

Oracle Server umożliwia implementację replikacji synchronicznej za pomocą procedur składowanych i wyzwalaczy. Wsparcie dla replikacji asynchronicznej zależy w sposób istotny od opcji serwera. *Oracle Server with the distributed option* oferuje jedynie migawki tylko do odczytu (*ang. read-only snapshots*). Bardziej zaawansowane mechanizmy (replikacja symetryczna, migawki modyfikowalne) dostępne są dopiero w *Oracle Server with the distributed and advanced replication options*. Opcja zaawansowanej replikacji jest kosztowna, a migawki tylko do odczytu są dostępne już wtedy, gdy możliwe jest rozproszenie bazy danych, pozwalając na korzystanie ze wspomnianych wcześniej zalet replikacji. Wszystko to zachęca do ich stosowania w systemach rozproszonych baz danych, lecz fakt, że nie zezwalają one na modyfikacje danych, a także sam charakter replikacji asynchronicznej, może w niektórych sytuacjach istotnie wpływać na funkcjonalność systemu.

3. Migawki tylko do odczytu

Migawka tylko do odczytu jest definiowana przez rozproszone zapytanie odwołujące się do jednej lub więcej tabel źródłowych, perspektyw lub innych migawek. W zależności od postaci zapytania, migawka może być pełną kopią tabeli, kopią fragmentu tabeli lub udostępniać dane z wielu tabel. Dane zawarte w migawce są wynikiem wydania zapytania definiującego migawkę w pewnym momencie w przeszłości. W związku z tym, że dane źródłowe poddane replikacji za pomocą migawek tylko do odczytu mogą się zmieniać w czasie, aby migawka odzwierciedlała w miarę aktualny stan tabel źródłowych, konieczne jest jej odświeżanie. W dowolnej chwili można zażądać odświeżenia migawki lub grupy migawek poprzez wywołanie odpowiedniej procedury

składowanej. Ponadto, migawki mogą być odświeżane automatycznie przez system w zadanych odstępach czasowych.

Dla pojedynczych migawek lub ich grup specyfikuje się następujące parametry automatycznego odświeżania:

- interwał czasowy – mówiący jak często migawka ma być automatycznie odświeżana (jeśli nie jest podany, to migawka nie jest odświeżana automatycznie i wymaga „ręcznego" odświeżania za pomocą wywołania procedury składowanej)
- tryb odświeżania (FAST, COMPLETE lub FORCE).

Tryb FAST dostępny jest tylko dla tzw. migawek prostych. Migawka prosta bazuje na jednej tabeli i zapytanie ją definiujące nie zawiera: agregatów, operatora DISTINCT, klauzul GROUP BY lub CONNECT BY, podzapytań, operacji połączenia i operacji na zbiorach. Odświeżanie w trybie FAST jest odświeżaniem przyrostowym polegającym na uwzględnieniu w replice zmian, których dokonano w replikowanej tabeli od chwili poprzedniego odświeżenia migawki. Aby możliwe było wykorzystanie tego trybu, na tabeli źródłowej musi być założony tzw. dziennik migawki, który zawiera informacje o usuniętych, zmodyfikowanych i dodanych krotkach. Replika, która nie spełnia wspomnianych warunków, jest repliką złożoną i wymaga odświeżania w trybie COMPLETE polegającym na ponownym utworzeniu migawki poprzez wykonanie definiującego ją zapytania. Tryb COMPLETE jest z reguły bardziej czasochłonny niż FAST (szczególnie wtedy gdy niewielka część krotek tabeli źródłowej ulega zmianie w okresie między kolejnymi odświeżeniami migawki) i dlatego, mimo że jest oczywiście możliwy również dla migawek prostych, jest na ogół wykorzystywany tylko przy odświeżaniu migawek złożonych, gdy nie ma dla niego alternatywy. Tryb odświeżania FORCE sprowadza się do wykonania odświeżenia w trybie FAST gdy jest ono możliwe, a w przeciwnym wypadku COMPLETE.

Podstawową wadą migawek tylko do odczytu jest to, że z definicji nie pozwalają na modyfikowanie danych. W związku z tym, wspomagają one implementację jedynie podstawowego modelu replikacji, w którym jedna z kopii jest „kopią matką" i tylko ona może być modyfikowana, natomiast pozostałe repliki pozwalają jedynie na odczyt danych.

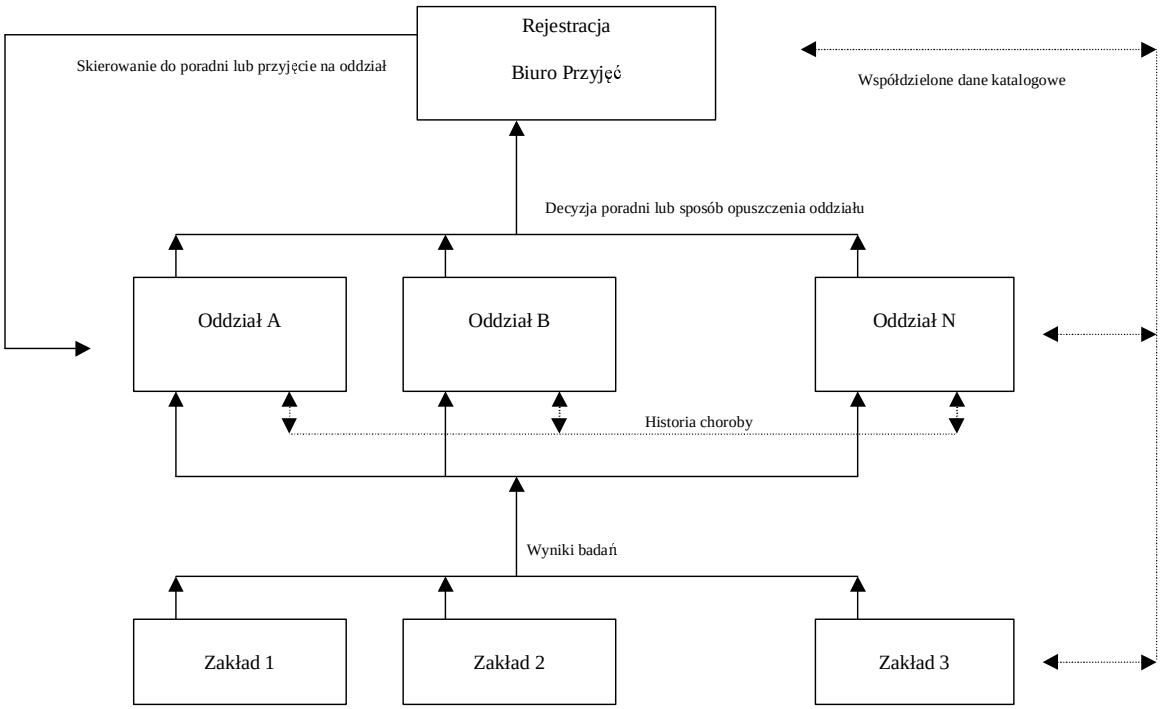
Model taki jest najbardziej odpowiedni w sytuacjach, gdy dane są modyfikowane tylko w jednej z jednostek organizacyjnych przedsiębiorstwa, a czytane w pozostałych. Mimo swojej prostoty i istotnych ograniczeń funkcjonalnych podstawowy model replikacji wykazuje wyższość nad architekturą rozproszoną bez replikacji szczególnie w przypadku znacznego geograficznego rozproszenia jednostek przedsiębiorstwa, gdyż korzystanie z replik przy operacjach odczytu zamiast odwoływania się do zdalnych danych pozwala skrócić czas odpowiedzi, a także gwarantuje dostęp do potrzebnych danych na wypadek awarii połączeń sieciowych. Replikacja asynchroniczna bazująca na migawkach tylko do odczytu jest efektywnym mechanizmem szczególnie wtedy gdy modyfikacje replikowanych danych są rzadkie, bądź gdy nie jest wymagana natychmiastowa propagacja zmian, ponieważ w takich sytuacjach możliwe jest stosunkowo rzadkie odświeżanie migawek w chwilach gdy obciążenie systemu jest niewielkie, np. w nocy.

4. Problemy związane z wykorzystywaniem migawek na przykładzie systemu informatycznego szpitala

Podczas prac prowadzonych w Instytucie Informatyki nad systemem informatycznym szpitala Eskulap/2000 [3] rozważane były różne architektury bazy danych. Mimo, że można przyjąć iż w większości przypadków jednostki organizacyjne informatyzowanego szpitala będą połączone stabilną siecią lokalną, wybór padł na architekturę rozproszoną ze względu na ogromne ilości danych przetwarzane w szpitalu oraz nieustanny ich przyrost, który w przypadku architektury scentralizowanej mógłby prowadzić do nadmiernego obciążenia serwera. Natomiast rozproszona architektura bazy danych nie tylko pozwala rozłożyć obciążenie systemu na kilka serwerów, ale także cechuje się bogatszymi możliwościami skalowania systemu.

Szpital składa się z wielu jednostek organizacyjnych, które wykonują specyficzne zadania, a w związku z tym są również odpowiedzialne za generowanie różnych danych. Ze względu na duże ilości danych przetwarzanych w systemie informatycznym szpitala oraz poufność niektórych informacji architektura, w której każda z jednostek organizacyjnych posiada swój własny serwer bazy danych, wydaje się być odpowiednia. Problemem jest oczywiście fakt, że dane wprowadzane do systemu w jednej z jednostek organizacyjnych szpitala mogą być wykorzystywane również w innych jednostkach. Idealnym rozwiązaniem jest sytuacja, w której dla zrealizowania wszystkich operacji aplikacje użytkowników systemu z danej jednostki muszą odwoływać się jedynie do lokalnego serwera bazy danych. Aby było to możliwe niezbędna jest replikacja niektórych danych. Ze względu na znaczną różnicę w cenie wersji serwera *Oracle Server with the distributed option* w stosunku do *Oracle Server with the distributed and advanced replication options* oprogramowanie Eskulap/2000 zakłada istnienie jedynie opcji rozproszonej, która dla wsparcia replikacji oferuje jedynie migawki tylko do odczytu. Zastosowanie replikacji asynchronicznej opartej na migawkach tylko do odczytu ze względu

na ich naturę może jednak w niektórych konkretnych przypadkach w istotny sposób niekorzystnie zmieniać funkcjonalność systemu wprowadzając zagrożenia nie występujące w architekturze rozproszonej bez replikacji. Ponadto, może się okazać, że obciążenie systemu związane z odświeżaniem migawek będzie większe niż w przypadku bezpośredniego odwoływania się do zdalnych relacji. Implementacja replikacji danych za pomocą migawek komplikuje się dodatkowo ze względu na fakt, że często okazuje się iż pewne dane mogą być modyfikowane w kilku jednostkach organizacyjnych. Poniższy rysunek przedstawia uproszczony schemat udostępniania pewnych informacji przez jednostki organizacyjne innym jednostkom. Uwzględnione zostały na nim dane, których udostępnianie komplikuje się w przypadku rozproszonej architektury systemu. Linie przerywane symbolizują możliwość modyfikacji współdzielonych danych w wielu jednostkach, a linie ciągłe udostępnianie danych innym jednostkom tylko do odczytu.



Rysunek 1. Schemat przepływu informacji między jednostkami szpitala

Aby lepiej przyjrzeć się zasygnalizowanym problemom, rozważmy następujące zagadnienia związane z funkcjonalnością systemu informatycznego szpitala.

4.1 Modyfikacje replikowanych danych katalogowych

W wielu aplikacjach wykorzystywane są dane o charakterze katalogów. Przykładem takich danych są informacje o podziale administracyjnym kraju obejmujące listy województw, powiatów i gmin. Oczywiste jest, że wszystkie jednostki szpitala powinny dysponować taką samą listą województw czy gmin. Lista taka charakteryzuje się niewielką zmiennością w czasie, może się zmienić w wyniku reformy administracyjnej kraju, ale mogą również być konieczne drobne poprawki po wykryciu np. błędów literowych. W tej sytuacji nie ma dużej różnicy czy każda z jednostek będzie mieć w swojej bazie danych niezależne tabele zawierające informacje o gminach, województwach, itp., czy też na jednym stanowisku będą znajdować się tabele źródłowe a na pozostałych repliki odświeżane rzadko lub tylko na żądanie by niepotrzebnie nie obciążać systemu. Rozwiązanie oparte na replikacji wydaje się lepsze, gdyż wymusza modyfikacje na wyróżnionej kopii i automatycznie gwarantuje spójność pozostałych.

W systemie wykorzystywane są również katalogi o innym charakterze, które są uaktualniane w trakcie normalnej pracy z systemem. Przykładem takich danych jest lista jednostek obcych, z których pacjenci są kierowani do szpitala lub do których szpital kieruje swoich pacjentów na badania czy konsultacje. Każda z jednostek szpitala powinna dysponować taką samą listą jednostek obcych, aby możliwe było opracowywanie globalnych zestawień statystycznych np. o jednostkach, do których najczęściej kieruje się pacjentów na badania

itp. Replikacja asynchroniczna za pomocą migawek tylko do odczytu umożliwia wszystkim jednostkom szybki dostęp do danych katalogowych, ale modyfikacje mogą być wykonane tylko na „kopii matce". Brak dostępu do stanowiska, na którym znajduje się kopia źródłowa np. w wyniku awarii serwera uniemożliwi uzupełnienie katalogu. Replikacja nie rozwiązuje więc w tym przypadku całkowicie problemów związanych z rozproszeniem danych, ale zdecydowanie jest wskazana, gdyż gwarantuje możliwość odczytu katalogów, jeśli tylko dostępny jest serwer lokalny. Prawdopodobieństwo ograniczenia funkcjonalności na wypadek awarii jest oczywiście większe niż w wypadku scentralizowanej bazy danych, ale można je zminimalizować umieszczając dane katalogowe na serwerze jednostki, która najczęściej je modyfikuje.

4.2 Udostępnianie historii choroby i modyfikacje danych związanych z chorobami pacjentów

Gdy pacjent zostaje skierowany na oddział, zbierane są o nim w trakcie wywiadu informacje o przebytych i współistniejących chorobach, chorobach w rodzinie pacjenta, uczuleniach itp. Są to dane wpisywane do tzw. historii choroby, która jest następnie uzupełniana o aktualne rozpoznania, wyniki badań oraz przebieg leczenia. Ze względu na olbrzymie ilości danych celowe jest zorganizowanie systemu w taki sposób, żeby każdy oddział dysponował lokalnym serwerem bazy danych, na którym składowane są dane dotyczące pacjentów, którzy byli lub aktualnie są leczeni na danym oddziale. Gdy pacjent w przyszłości pojawi się ponownie na tym samym oddziale, lekarze będą mieli oczywiście dostęp do jego historii choroby, będą mogli uaktualnić dane o pacjencie (uczulenia, choroby, itp.). Nie powinna być możliwa modyfikacja dotychczasowej historii choroby, powinna ona jedynie być uzupełniana o nowe informacje.

Sytuacja komplikuje się znacznie, gdy pacjent leczony kiedyś w szpitalu pojawia się ponownie, ale na innym oddziale, bądź też gdy w trakcie leczenia zostaje podjęta decyzja o przeniesieniu pacjenta na inny oddział. Nowe informacje związane z historią choroby pacjenta powinny być składowane na serwerze oddziału, na którym aktualnie pacjent przebywa. Natomiast niezbędne jest również udostępnienie historii choroby, która powstała na oddziałach, na których pacjent przebywał wcześniej. Udostępnianie tych informacji za pomocą rozproszonych zapytań odczytujących dane o danym pacjencie z serwerów wszystkich oddziałów jest niebezpieczne, gdyż zakończy się niepowodzeniem w przypadku awarii któregokolwiek z serwerów oddziałowych, a dodatkowo może okazać się nieefektywne, ponieważ wymaga wykonywania rozproszonych transakcji. Replikowanie danych dotyczących pacjentów leczonych aktualnie na oddziale z innych oddziałów wydaje się kuszącym rozwiązaniem. Nawet ograniczając się do migawek tylko do odczytu, taką replikację można zrealizować na kilka sposobów. Pierwszy z nich to wykorzystanie jednej migawki bazującej na unii danych z odpowiadających sobie tabel ze wszystkich oddziałów. Wadą takiego rozwiązania jest fakt, że odświeżenie migawki nie powiedzie się w przypadku, gdy chociaż jeden z serwerów oddziałowych będzie z jakiegoś powodu niedostępny. Ulepszeniem takiego wariantu jest konfiguracja, w której dla każdego z pozostałych oddziałów tworzony jest w lokalnej bazie danych zestaw migawek odwołujących się do danych tylko z danego oddziału, a dane zbiorcze są udostępniane poprzez perspektywy bazujące na uniach odpowiadających sobie migawek. Wtedy na wypadek awarii, gdy jeden z serwerów oddziałowych będzie niedostępny, każdy z pozostałych oddziałów może mieć dostęp do aktualnych danych ze wszystkich oddziałów poza tym niedostępnym (oczywiście dostępne są dane z chwili ostatniego odświeżenia migawek replikujących jego obiekty). Ceną za bezpieczniejsze odświeżanie migawek w tej konfiguracji jest dłuższy czas wykonywania zapytań odwołujących się do historii choroby z innych oddziałów ze względu na wykorzystanie złożonych perspektyw.

We wszystkich powyższych wariantach występują dwa istotne problemy. Pierwszy wiąże się z trybem odświeżania migawek, a drugi wynika z faktu, że niekiedy konieczne może być zezwolenie na modyfikacje danych pamiętanych na serwerach innych oddziałów. Efektywnym trybem odświeżania migawek jest tryb FAST, szczególnie w przypadku rzadko modyfikowanych danych. Niestety jest on możliwy tylko dla migawek prostych. Natomiast aby migawki z historią choroby zawierały dane tylko tych pacjentów, którzy w danej chwili są leczeni na oddziale, muszą one w swej definicji zawierać podzapytanie zwracające identyfikatory pacjentów, dla których istnieją aktualne pobyty na oddziale. Migawki takie musiałyby być odświeżane bardziej kosztownym trybem COMPLETE. Odświeżanie w trybie FAST wymagałoby rezygnacji z filtrowania danych i replikacji prawie wszystkich danych każdego oddziału na pozostałych oddziałach, co w znacznym stopniu podważyłoby sens replikacji, której celem było rozłożenie obciążenia w systemie. Wagę opisywanego problemu na szczęście zmniejsza fakt, że odświeżenie migawek konieczne byłoby tylko w chwili przyjęcia na oddział pacjenta, który był kiedyś leczony na innym oddziale.

Drugi ważny problem występuje gdy replikowane dane mogą być modyfikowane przez wiele oddziałów. Przykładem są informacje o chorobach pacjentów. Naturalnym rozwiązaniem jest pamiętanie ich w tabeli o strukturze: identyfikator pacjenta, identyfikator choroby, rok zachorowania, opcjonalny rok zakończenia choroby. Każdy oddział posiadający lokalną tabelę z chorobami pacjentów musiałby posiadać repliki takich tabel z innych oddziałów. Problem pojawia się gdy zachorowanie pacjenta na jakąś chorobę zaewidencjonuje jeden oddział, a datę zakończenia choroby wprowadza drugi. Wariant z modyfikacją tabeli

źródłowej, który sprawdził się w przypadku danych katalogowych, w tym wypadku się komplikuje, gdyż trzeba rozstrzygnąć w której z baz danych tabela źródłowa się znajduje. Rozwiązaniem powyższych problemów może być utrzymywanie danych pacjenta tylko w jednej kopii – na oddziale, na którym pacjent przebywa, a po zakończeniu pobytu przenoszenie ich do centralnego archiwum. Przy kolejnym przyjęciu pacjenta na dowolny z oddziałów, dane o nim byłyby sprowadzane na dany oddział. Rozwiązanie takie również nie jest niestety idealne, ponieważ zdarza się, że dane pacjenta są w danej chwili potrzebne na dwóch oddziałach jednocześnie np. gdy pacjent w ramach pobytu na jednym oddziale jest kierowany na konsultacje lub zabieg na drugi oddział. W związku z tym, być może najlepszym rozwiązaniem jest rezygnacja z rozpraszania tych danych, które potencjalnie mogą dotyczyć wielu oddziałów.

4.3 Udostępnianie wyników badań pacjentów innym jednostkom

Danymi, które również są wprowadzane w pewnych jednostkach i udostępniane innym są wyniki badań. Pacjenci przebywający na oddziałach kierowani są na badania np. do laboratorium, zakładu radiologii, itp. W przypadku architektury rozproszonej, gdzie jednostki wykonujące badania wprowadzają wyniki badań do swoich lokalnych baz danych, pojawia się problem udostępniania oddziałom wyników ich pacjentów. Wykorzystanie migawek tylko do odczytu prowadzi do dylematu nad trybem ich odświeżania. Problem jest tu bardziej istotny niż w przypadku udostępniania historii choroby, gdyż migawki z wynikami badań powinny być odświeżane możliwie często (jeśli czas potrzebny na przesłanie wyników badań drogą elektroniczną byłby dłuższy niż „papierową", sensowność używania systemu informatycznego zostałaby w znacznym stopniu podważona). Rezygnacja z replikacji prowadziłaby do konieczności wykonywania rozproszonych zapytań przy każdym żądaniu wyświetlenia wyników badań pacjenta. Być może jednak takie zapytania byłyby generowane przez użytkowników nie częściej niż przez system w celu odświeżania migawek, a na pewno wymagałyby transmisji mniejszej ilości danych.

Za wykorzystaniem replikacji przemawia jedynie kwestia dostępności danych na wypadek awarii. Zdarza się jednak, że kwestie dostępności pewnych danych są regulowane specjalnymi przepisami i czas udostępniania tych danych musi być ściśle kontrolowany. Replikacja tego typu danych ogranicza kontrolę dostępu do nich, szczególnie na wypadek awarii połączeń sieciowych opóźniającej odświeżenie migawek.

4.4 Propagowanie ważnych decyzji oraz informacji o ich ewentualnym anulowaniu

Wiele decyzji podejmowanych w różnych jednostkach szpitala implikuje pewne akcje w innych jednostkach. Przykładowo decyzja izby przyjęć o skierowaniu pacjenta na dany oddział upoważnia ten oddział do przyjęcia pacjenta. Przenoszenie informacji o tego typu decyzjach za pomocą migawek tylko do odczytu, pozwala na zatwierdzenie i wprowadzenie do systemu decyzji dotyczącej jakiejś jednostki, nawet gdy jej baza danych jest niedostępna, gdyż jest gwarancja, że po usunięciu awarii migawki się odświeżą i informacja dotrze do adresata. Korzystanie z danych innych jednostek za pomocą migawek ukrywa przed użytkownikami awarie połączeń sieciowych. Tego typu awarie nie spowodują utraty dostępu do danych, a jedynie nie pozwolą na odświeżenie migawek, przez co nie przepłyną nowe informacje. Jeśli nie przepłynie drogą elektroniczną np. decyzja o przyjęciu pacjenta na oddział, to uszkodzenie połączenia zostanie wykryte w chwili, gdy pacjent fizycznie zgłosi się na oddział. Są jednak sytuacje, w których nieświadomość użytkowników, że nie mają dostępu do najświeższych danych może mieć katastrofalne skutki.

Żaden z użytkowników systemu nie jest nieomylny i zawsze istnieje możliwość wprowadzenia do systemu pewnych danych przez pomyłkę. Zezwolenie na usuwanie omyłkowo wprowadzonych do bazy danych informacji może być niewskazane. Na przykład wprowadzony przez przypadek błędny wynik badania może spowodować podjęcie przez lekarza niewłaściwej decyzji. Zezwolenie na jego usunięcie zafałszowałoby obraz sytuacji, w której decyzja została podjęta. Właściwszym rozwiązaniem jest umożliwienie jedynie anulowania niepoprawnych wpisów. W przypadku gdy informacja o anulowaniu jakiegoś wpisu udostępniana jest innym jednostkom poprzez migawki tylko do odczytu, istnieje niebezpieczeństwo, że informacja nie będzie propagowana z powodu awarii połączeń sieciowych, a użytkownik dokonujący anulowania nawet nie zostanie poinformowany o braku komunikacji. Dlatego wydaje się, że kluczowe decyzje w systemie powinny być przenoszone w ramach rozproszonych transakcji, aby szybko można było wykryć awarię połączeń.

5. Podsumowanie

W artykule przedstawiono zalety i wady architektury rozproszonej wykorzystującej replikację danych opartą na migawkach tylko do odczytu, odwołując się do konkretnych przykładów z projektowanego w Instytucie Informatyki Politechniki Poznańskiej systemu informatycznego szpitala. Rozproszenie danych pozwala na rozłożenie obciążenia w systemie, a także ułatwia jego skalowanie. Może się ono jednak wiązać ze znacznym

skomplikowaniem implementacji pewnych mechanizmów, a także zwiększyć podatność systemu na awarie. Aby niskim kosztem uodpornić system na awarie połączeń sieciowych oraz poprawić efektywność wykonywania zapytań, wskazane jest replikowanie niektórych danych za pomocą migawek tylko do odczytu. Przy wyborze danych, które mają być replikowane należy zdawać sobie sprawę, że tego typu replikacja może silnie wpływać na funkcjonalność systemu. Najlepsze efekty daje replikacja danych, które są często czytane, a stosunkowo rzadko modyfikowane, przy czym najlepiej aby modyfikacje były dokonywane tylko w jednej z jednostek informatyzowanej organizacji. W przypadku danych modyfikowanych w wielu jednostkach, być może należy nawet zrezygnować z ich rozproszenia lub zastosować bardziej kosztowną replikację symetryczną. Replikacja za pomocą migawek tylko do odczytu, ze względu na swój asynchroniczny charakter, jest również niewskazana przy przesyłaniu między jednostkami informacji, które powinny natychmiast dotrzeć do adresata, a ewentualna niemożność ich przesłania nie powinna być ukrywana przed użytkownikami.

Literatura

[1] Oracle7 Server Distributed Systems Manual, Vol. 1, Oracle7 Documentation.
[2] Oracle7 Server Distributed Systems Manual, Vol. 2, Oracle7 Documentation.
[3] T. Biały, System informatyczny Eskulap/2000, PLOUG'98, listopad 1998.
[4] S. Ceri, G. Pelagatti, Distributed Databases Principles and Systems, McGraw–Hill Book Company, 1984.
[5] T. Koszlajda, Synchroniczna a asynchroniczna replikacja danych w systemach rozproszonych baz danych, POLMAN'98, kwiecień 1998.