ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1997
Seria: INFORMATYKA z. 32 Nr kol. 1356

Mirosław CHŁOPEK, Katarzyna HARĘŻLAK, Henryk JOSIŃSKI
Politechnika Śląska, Instytut Informatyki

IMPLEMENTACJA PODSTAWOWYCH MECHANIZMÓW ZARZĄDZANIA ROZPROSZONĄ BAZĄ DANYCH W EKSPERYMENTALNYM SYSTEMIE WYKORZYSTUJĄCYM MODEL WIRTUALNIE WSPÓŁDZIELONEJ PAMIĘCI

Streszczenie. W pracy przeprowadzono analizę własności systemu zarządzania rozproszoną bazą danych oraz przedstawiono jego implementację w systemie eksperymentalnym. Architektura systemu oparta została na modelu wirtualnie współdzielonej pamięci. Przedstawiono funkcje i zadania realizowane przez moduły koordynatora transakcji rozproszonej, klienta oraz lokalnych serwerów. Jako lokalne systemy bazy danych zostały wykorzystane serwery Ingres oraz Postgres95.

THE IMPLEMENTATION OF THE BASIC DISTRIBUTED DATABASE MANAGEMENT MECHANISMS IN THE EXPERIMENTAL SYSTEM USING THE MODEL OF VIRTUAL SHARED MEMORY

Summary. In the article the authors have done the analysis of the qualities of the distributed database management system. Apart from this, the authors have presented its implementation in the experimental system. The architecture of the system was based on the model of the virtual shared memory. Also, the authors have displayed the functions and tasks performed by the modules of distributed transaction coordinator, client and local servers. As local database systems, Ingres's and Postgres95's servers have been used.

1. Wstęp

Rosnące wymagania użytkowników dotyczące zwiększenia wydajności wykorzystywanych systemów baz danych, ich niezawodności oraz szerokiego dostępu do danych, polegającego na integracji wielu baz niezależnie od modeli danych, na których się opierają, i środowisk ich funkcjonowania, powodują wzrost zainteresowania systemami rozproszonych baz danych. Ważnym aspektem ich użyteczności jest to, że w naturalny sposób odzwierciedlają zdecentralizowaną strukturę wielu organizacji.

Rozproszoną bazę danych stanowi zbiór baz lokalnych, z reguły umieszczonych w wielu węzłach sieci, które mogą się komunikować ze sobą. Umożliwia ona dostęp do odległych baz danych (jeśli opierają się one na różnych modelach danych, mówi się o *heterogenicznej wielobazowej rozproszonej bazie danych*), znosi ograniczenie objętości danych obowiązujące w węźle bazy lokalnej oraz pozwala na równoległe przetwarzanie obliczeń w wielu węzłach. Z punktu widzenia użytkownika widziana jest jako jedna baza danych. Jest to efektem pracy *systemu zarządzania rozproszoną bazą danych* (SZRBD), który *fizyczne jednostki danych* (np. tabele, perspektywy) zapamiętane w węzłach lokalnych baz danych integruje w *globalny schemat rozproszonej bazy danych*. Każda fizyczna jednostka należąca do rozproszonej bazy ma swój odpowiednik w schemacie globalnym w postaci *logicznej jednostki danych*.

Poza niewątpliwymi zaletami, jakie niesie ze sobą możliwość korzystania z rozproszonych baz danych, należy zwrócić uwagę na pewne problemy wiążące się z ich realizacją. Należy do nich duża złożoność operacji odtwarzania stanu zgodnego bazy, kontroli dostępu do danych i ich modyfikacji oraz dodatkowe narzuty związane z zarządzaniem systemem.

2. Cechy i mechanizmy systemu zarządzania rozproszoną bazą danych

Wysokie wymagania stawiane przez użytkowników systemom zarządzania bazami danych obligują ich autorów do tworzenia takich systemów, które nie tylko zagwarantują bezpieczeństwo danych, efektywne przetwarzanie informacji oraz odporność na awarie, ale również zapewnią ich wygodną obsługę. Wymagania te dotyczą także systemów rozproszonych baz danych. Ze względu na możliwość integracji wielu baz danych różnych typów powinny one zostać wzbogacone o pewne dodatkowe cechy i mechanizmy, których istnienia użytkownik nie musi być świadomy. Należą do nich:

Przeźroczystość

Posiadanie tej własności przez SZRBD zwalnia użytkownika z konieczności poznawania schematów i lokalizacji baz danych wchodzących w skład bazy rozproszonej. Powinien on widzieć wszystkie udostępnione mu dane tak, jak gdyby były one scentralizowane w lokalnej bazie danych.

Uniwersalny i jednorodny dostęp do danych

Cecha ta jest pewnym uzupełnieniem poprzedniej własności. Uniwersalność dostępu do danych pozwala użytkownikowi na posługiwanie się jednym językiem zapytań. Uwzględnienie tej własności wymaga, by rozproszona baza danych była obsługiwana za pomocą jednorodnych interfejsów.

Autonomia lokalnej bazy danych

Włączenie lokalnej bazy danych do bazy rozproszonej zwiększa zasięg dostępności danych, nie powinno mieć jednak wpływu na jej funkcjonowanie jako scentralizowanej bazy danych obsługiwanej przez lokalny serwer. Pełna kontrola dostępu do posiadanych danych leży nadal w gestii administratora bazy lokalnej.

Obsługa różnych typów rozproszeń danych

Przystępując do projektu SZRBD należy wziąć pod uwagę wszystkie możliwe typy rozproszeń danych:

- *replikację* – powielenie danych w wielu węzłach sieci;
- *fragmentację pionową* – podział atrybutów tabeli podstawowej uzyskiwany za pomocą operacji projekcji. Różne fragmenty pionowe tej samej tabeli podstawowej nie muszą być rozłączne; wręcz wskazane jest, by każdy z nich zawierał klucz główny tabeli celem dokonania bezstratnego i nieredundancyjnego odtworzenia tabeli podstawowej;
- *fragmentację poziomą* – podział rekordów tabeli podstawowej zawierający wszystkie jej atrybuty. Uzyskuje się go przez wykonanie na tabeli podstawowej operacji selekcji według warunku nazywanego *dozorem*, który musi być spełniony przez każdy rekord należący do fragmentu;
- różne warianty kombinacji typów wyżej wymienionych.

Globalny katalog danych

Korzystanie z obiektów należących do bazy danych wymaga od systemu zarządzania przechowywania o nich pewnych informacji. Ich zbiór tworzy *katalog danych* (*słownik*, *repozytorium*). W przypadku bazy rozproszonej obejmuje informacje o wszystkich jej składowych bazach lokalnych, stąd nazywany jest *katalogiem globalnym*. Zawartość globalnego katalogu tworzą:

- globalny schemat rozproszonej bazy danych (informacje o tabelach lokalnych baz danych oraz o tabelach istniejących wyłącznie w ramach bazy rozproszonej),

- specyfikacja ograniczeń związanych z użyciem danych (prawa dostępu, więzy integralności),
- specyfikacja elementów przechowywanych wewnątrz bazy danych (perspektyw, reguł, procedur bazy danych),
- informacje wykorzystywane w procesie optymalizacji zapytań rozproszonych.

Najbardziej naturalnym sposobem pamiętania tych informacji jest dla systemu zarządzania bazą danych przechowywanie ich w postaci specjalnych tabel tworzących odrębną bazę danych – scentralizowaną lub rozproszoną. SZRBD dysponując słownikiem o tej postaci zezwala użytkownikowi na korzystanie z informacji związanych z opisem bazy za pośrednictwem języka zapytań.

Kontrola współbieżnego dostępu do danych

Utrzymywanie spójności bazy danych wymaga od SZRBD umiejętności synchronizacji współbieżnego dostępu do danych. Oznacza to, że żaden z użytkowników bazy danych nie powinien odczytywać lub aktualizować rekordów, których zawartość jest właśnie zmieniana przez innego użytkownika. W konsekwencji zezwolenia na taką sytuację może dojść do udostępnienia nieaktualnych danych. Niezbędne jest więc użycie odpowiednich środków kontroli współbieżnego dostępu do danych. Wśród klasycznych metod synchronizacji można wyodrębnić:

- grupę metod pesymistycznych – opartych na blokadzie dostępu do danych,
- grupę metod optymistycznych – polegających na niezależnym wykonaniu transakcji i późniejszej detekcji konfliktu.

Niepożądanym zjawiskiem, mogącym wystąpić przy próbie współbieżnego dostępu do danych, jest zakleszczenie (impas). Sytuacji tej można zapobiegać stosując strategie blokowania nie dopuszczające do niego. Zakładają one przy przewidywaniu jego wystąpienia udział systemu, który w takiej sytuacji opóźnia wykonanie jednej z transakcji. Alternatywnym rozwiązaniem jest wykorzystywanie metod pozwalających zlikwidować zakleszczenie przez wycofanie jednej z transakcji, co pozwala na dokończenie drugiej.

Optymalizacja zapytań rozproszonych

Wymagania użytkowników stawiane systemom zarządzania bazami danych dotyczą również szybkości wyszukiwania informacji. Znalezieniem najbardziej efektywnego sposobu realizacji tej operacji zajmuje się specjalizowany moduł systemu nazywany *optymalizatorem zapytań*. Złożoność problemu rośnie w przypadku zadania wyszukiwania skierowanego do rozproszonej bazy danych, bowiem – oprócz czynników rozważanych w procesie optymalizacji zapytania lokalnego (informacje o organizacji fizycznego dostępu do danych, zestaw procedur elementarnych realizujących operacje składowe zapytania oraz histogramy rozkładu wartości dla poszczególnych atrybutów) – należy również wziąć pod uwagę następujące elementy:

- typ rozproszenia danych,
- przepustowość linii komunikacyjnych,
- względną wydajność poszczególnych węzłów.

Źródłem tych dodatkowych informacji jest globalny katalog danych.

Protokół wypełnienia transakcji rozproszonej, odtwarzanie skutków transakcji rozproszonej po awarii

Korzystając z danych rozproszonych pomiędzy węzły sieci, należy mieć na uwadze, że pojedyncza transakcja może wymagać dostępu do kilku lokalnych baz danych należących do bazy rozproszonej. Każde zapytanie wchodzące w skład transakcji może więc podlegać dekompozycji na podzapytania, które zostaną wykonane przez odległe serwery. Wyniki tych podzapytań należy następnie scalić w wynik globalny w serwerze, który dokonał dekompozycji. Transakcję realizowaną w ten sposób określa się mianem *transakcji rozproszonej*. Można w niej wyodrębnić zestawy elementarnych operacji dotyczących tej samej lokalnej bazy danych (*transakcje lokalne*). Schemat na rys. 1 ilustruje przebiegi lokalnej i rozproszonej transakcji. Specyfika wykonania transakcji rozproszonej wymaga prowadzenia przez moduły systemu dwóch typów plików (określanych mianem *plików log*). W *transakcyjnych plikach log* zapisuje się informacje o rozpoczęciu kolejnych etapów transakcji rozproszonej. Natomiast *pliki log bazy danych* zawierają zapisy o operacjach dokonywanych na niej w ramach transakcji.

Przetwarzanie każdej transakcji odbywa się w dwóch etapach: (1) obliczenia i aktualizacje, (2) *wypełnienie*, które oznacza zapis zmian dokonanych przez transakcję do pliku log, fizyczne wprowadzenie aktualizacji do bazy danych i odblokowanie danych dla innych transakcji. W transakcji rozproszonej realizacja etapu (2) w odległych, lokalnych bazach danych związana jest z wymianą informacji poprzez zawodne łącza i węzły. Wypełnienie wymaga więc specjalnego protokołu uzgodnień, który zapewni rozsądną niezawodność i wydajność. Przykładami takich protokołów (*protokołów wypełnienia transakcji*) są: protokół 2PC (*dwufazowy protokół wypełnienia transakcji*, ang. two-phase commit protocol) w swoich odmianach – scentralizowanej, zdecentralizowanej i zlinearyzowanej – oraz protokół 3PC. Schemat na rys. 2 ilustruje ideę scentralizowanego protokołu 2PC. Istotną rolę odgrywa w nim wyróżniony węzeł sieci nazywany *koordynatorem transakcji rozproszonej*. Jego zadaniem jest:

- rozpoczęcie realizacji protokołu 2PC przez wysłanie do każdego wykonawcy transakcji rozproszonej (*uczestnika*) żądania określenia jego gotowości do wypełnienia transakcji lokalnej,
- zebranie nadesłanych przez wykonawców odpowiedzi,
- podjęcie ostatecznej decyzji o wypełnieniu lub wycofaniu transakcji rozproszonej na podstawie zebranych informacji i wysłanie tej decyzji do każdego uczestnika. Decyzja

o wypełnieniu transakcji podejmowana jest tylko w przypadku gotowości wszystkich uczestników do wypełnienia swoich transakcji lokalnych.

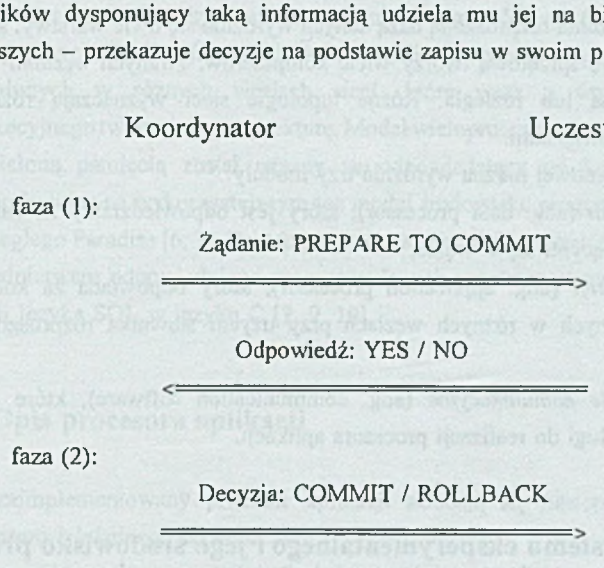
Transakcja lokalna	Transakcja rozproszona
Początek transakcji	Początek transakcji
Obliczenia i aktualizacje	Obliczenia i aktualizacje
Sygnał zakończenia ze strony użytkownika	Sygnał zakończenia ze strony użytkownika
Zapis zmian dokonanych przez transakcję do pliku log bazy danych	Realizacja protokołu wypełnienia transakcji rozproszonej z bieżącą aktualizacją transakcyjnych plików log koordynatora i uczestników
Dokonanie zmian w bazie danych	
Koniec transakcji	Zapisy zmian dokonanych przez transakcje lokalne do plików log bazy danych
	Dokonanie zmian w lokalnych bazach danych
	Koniec transakcji

Rys. 1. Schemat przebiegu transakcji lokalnej i rozproszonej
Fig. 1. Schema of execution of local and distributed transactions

Każdy z uczestników realizuje swoją transakcję lokalną złożoną z otrzymanego od koordynatora zestawu operacji elementarnych. Zakończenie transakcji może nastąpić dopiero po decyzji koordynatora podjętej zgodnie z zasadami protokołu 2PC. W jego pierwszej fazie uczestnik na żądanie koordynatora określa swoją gotowość do wypełnienia transakcji. Druga faza rozpoczyna się oczekiwaniem na ostateczną decyzję koordynatora, której realizacja zamyka protokół 2PC.

Transakcję rozproszoną uważa się za zakończoną po wydaniu ostatecznej decyzji przez koordynatora i odnotowaniu tego faktu w jego transakcyjnym pliku log. Stanu tego nie może zmienić nawet awaria łączy komunikacyjnych lub przerwanie pracy jednego z uczestników przed odebraniem decyzji. Jeśli uczestnik, po ponownym nawiązaniu łączności lub

uruchomieniu, w swoim pliku log nie znajduje zapisu o sposobie zakończenia transakcji rozproszonej, pyta o tę informację. Dla transakcji aktualnie biegnącej koordynator lub jeden z uczestników dysponujący taką informacją udziela mu jej na bieżąco, dla transakcji wcześniejszych – przekazuje decyzje na podstawie zapisu w swoim pliku log.



Rys. 2. Schemat protokołu 2PC
Fig. 2. Schema of the 2PC protocol

Narzędzia do zarządzania rozproszoną bazą danych

Administrator rozproszonej bazy danych powinien mieć możliwość wykonywania następujących operacji:

- tworzenie perspektyw i tabel istniejących wyłącznie w ramach bazy rozproszonej,
- rejestrowanie obiektów lokalnych w bazie rozproszonej,
- nadawanie praw i przywilejów użytkownikom.

Przenoszenie aplikacji między środowiskami

Istotna dla programistów jest możliwość przygotowania aplikacji na innych komputerach niż węzły rozproszonej bazy danych, a następnie przeniesienie aplikacji bez zmian do środowiska docelowego.

Realizacja powyższych cech i mechanizmów zapewnia użytkownikowi wymagane przez niego bezpieczeństwo danych i wygodę pracy, lecz z drugiej strony wiąże się ze zwiększonym nakładem pracy informatyków tworzących system.

3. Warstwa sprzętowa i programowa SZRBD

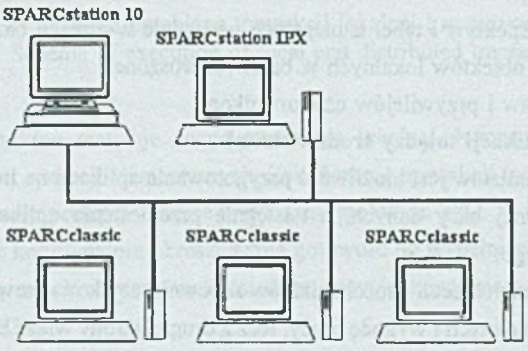
W systemie zarządzania rozproszoną bazą danych wyróżnia się dwie warstwy: sprzętową i programową. Warstwę sprzętową tworzy wiele komputerów, zwanych *węzłami* oraz sieć komunikacyjna lokalna lub rozległa. Różne topologie sieci wyznaczają różne drogi komunikacyjne między węzłami.

W warstwie programowej można wyróżnić trzy moduły:

- *procesor danych* (ang. data processor), który jest odpowiedzialny za zarządzanie danymi znajdującymi się w węźle;
- *procesor aplikacji* (ang. application processor), który odpowiada za koordynację dostępu do danych w różnych węzłach przy użyciu słownika rozproszonej bazy danych;
- *oprogramowanie komunikacyjne* (ang. communication software), które dostarcza podstawowe usługi do realizacji procesora aplikacji.

4. Elementy systemu eksperymentalnego i jego środowisko pracy

Ideą autorów opracowania było stworzenie eksperymentalnego systemu zarządzania rozproszoną bazą danych w oparciu o dostępne narzędzia sprzętowo-programowe. Przyjęto założenie, że system zostanie uruchomiony w lokalnej sieci Ethernet łączącej grupę stacji roboczych Sun (rys. 3). Stacje te posiadają różną architekturę i różnią się mocą obliczeniową.



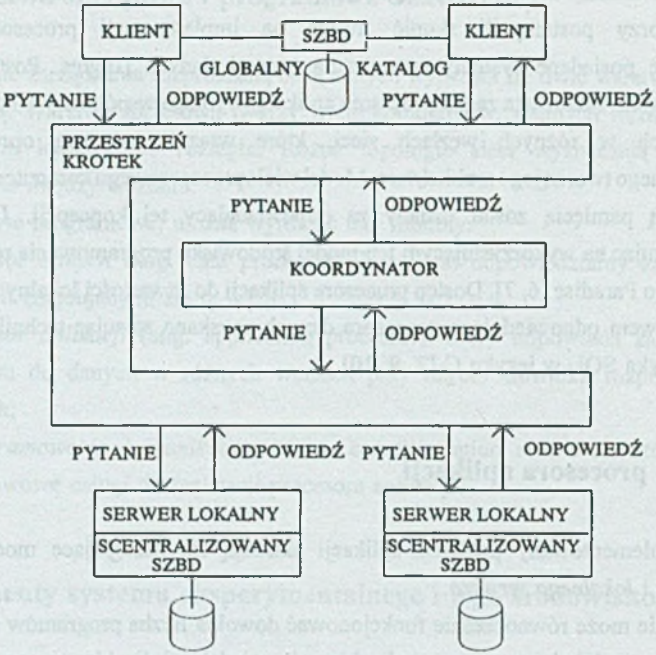
Rys. 3. Konfiguracja sieci lokalnej
Fig. 3. The configuration of the local area network

W efekcie analizy funkcji poszczególnych modułów tworzących warstwę programową SZRBD autorzy postanowili skupić uwagę na implementacji procesora aplikacji, wykorzystując posiadane systemy zarządzania bazami danych (Ingres, Postgres95) jako procesory danych. Realizacja zadań procesora aplikacji wymaga współpracy kilku programów uruchomionych w różnych węzłach sieci, które wraz z warstwą oprogramowania komunikacyjnego tworzą jego architekturę. Model wieloprocesorowego komputera z wirtualnie współdzieloną pamięcią został uznany za odpowiadający tej koncepcji. Implementacji dokonano bazując na wykorzystującym ten model środowisku programowania rozproszonego i równoległego Paradise [6, 7]. Dostęp procesora aplikacji do zawartości lokalnych baz danych za pośrednictwem odpowiedniego procesora danych uzyskano stosując technikę zanurzania instrukcji języka SQL w języku C [7, 9, 10].

5. Opis procesora aplikacji

Na zaimplementowany procesor aplikacji składają się następujące moduły: *klienta*, *koordynatora* i *lokalnego serwera*.

W systemie może równocześnie funkcjonować dowolna liczba programów klienta, jeden koordynator oraz lokalne serwery w liczbie odpowiadającej liczbie procesorów danych. Poszczególne moduły uruchamiane niezależnie na różnych komputerach komunikują się za pośrednictwem jednostek nazywanych *krotkami*, które stanowią sekwencje pól, poprzez wirtualnie współdzieloną pamięć – *przestrzeń krotek* (rys. 4).



Rys. 4. Struktura systemu
Fig. 4. The structure of the system

5.1. Moduł klienta

Program klienta jest uruchamiany na lokalnym komputerze użytkownika. Stanowi interfejs między użytkownikiem a modułem koordynatora. Do jego zadań zalicza się:

- pobranie od użytkownika ciągu zapytań w języku SQL tworzącego transakcję operującą na jednostkach logicznych,
- dokonanie analizy składniowej i semantycznej zapytań,
- odczyt informacji ze słownika danych,
- dekompozycję pojedynczego zapytania na podzapytania operujące na jednostkach fizycznych,
- wysłanie operacji do wykonania do koordynatora,
- odebranie wyników wykonania operacji od koordynatora,
- prezentację wyników wykonanych operacji.

5.2. Moduł koordynatora

Koordynator zarządza wykonaniem operacji zleconych przez użytkownika w środowisku rozproszonym. Jest scentralizowany i może być uruchomiony na dowolnym komputerze w sieci. Realizuje następujące funkcje:

- kierowanie odebranych od klientów operacji do wykonania do odpowiednich lokalnych serwerów,
- odbieranie od lokalnych serwerów wyników wykonania operacji i odsyłanie ich do klientów,
- zarządzanie transakcjami rozproszonymi (rozpoczynanie, monitorowanie, wypełnianie bądź wycofywanie),
- kontrola współbieżnego dostępu do danych poprzez wprowadzenie dodatkowego mechanizmu blokowania (*blokad logicznych*) niezależnego od blokad stosowanych przez procesory danych.

5.3. Moduł lokalnego serwera

Wymiana informacji pomiędzy modułem koordynatora a procesorami danych realizowana jest przez lokalne serwery. Specyfika realizacji funkcji lokalnego serwera związana jest z właściwościami systemu stanowiącego procesor danych. Należą do nich:

- pobieranie operacji od koordynatora,
- ewentualna translacja zapytań w języku SQL na język właściwy dla procesora danych lokalnej bazy danych (np. QUEL),
- wykonywanie pobranych operacji,
- odsyłanie wyników lub kodu błędu operacji do koordynatora,
- zarządzanie lokalnymi blokadami, o ile tej funkcji nie wykonuje sam procesor danych,
- zarządzanie lokalnymi transakcjami (rozpoczynanie, monitorowanie, wypełnianie bądź wycofywanie).

5.4. Ogólny algorytm wymiany informacji elementami procesora aplikacji

Podstawowy cykl pracy procesora aplikacji, po zadaniu przez użytkownika ciągu instrukcji stanowiących transakcję, można podzielić na następujące fazy (częściowo nakładające się na siebie):

(1) *Klient*:
dekompozycja zapytania na ciąg podzapytań operujących na jednostkach fizycznych;

(2) *Klient*:
wysłanie krotek z podzapytaniami do koordynatora;

(3) *Koordynator*:
założenie blokad logicznych;

(4) *Koordynator*:
wysłanie krotek z podzapytaniami do lokalnych serwerów;

(5) *Lokalny serwer*:
wykonanie operacji na lokalnych bazach danych za pomocą mechanizmów procesora danych;

(6) *Lokalny serwer*:
wysłanie krotek z danymi oraz potwierdzenia wykonania podzapytania do koordynatora;

(7) *Koordynator*, *lokalne serwery*:
realizacja protokołu wypełnienia transakcji rozproszonej;

(8) *Koordynator*:
wysłanie krotek z danymi oraz statusu wykonania transakcji do klienta;

(9) *Koordynator*:
zdjęcie blokad logicznych;

(10) *Klient*:
prezentacja wyników.

6. Implementacja wybranych cech oraz mechanizmów systemu zarządzania rozproszoną bazą danych

W rozdziale 2 opracowania skupiono uwagę na najistotniejszych cechach, które powinien posiadać SZRBD. Sposób ich realizacji zależny jest od implementacji konkretnego systemu. W przypadku systemu eksperymentalnego wpływ na proces jego tworzenia miało zastosowanie środowiska programowania równoległego i rozproszonego Paradise. Wykorzystanie współdzielonej pamięci do komunikacji pomiędzy poszczególnymi elementami systemu znalazło odbicie w sposobie realizacji transakcji rozproszonej i implementacji protokołu jej wypełnienia. Możliwość zastosowania techniki zanurzania instrukcji języka SQL w języku C zadecydowała o wyborze metody prowadzenia globalnego katalogu danych. W poniżej omówionych mechanizmach, które zostały wykorzystane w projekcie, uwzględnionio wpływ specyfiki użytych narzędzi.

Przeźroczystość

Użytkownik operuje schematem globalnym bazy posługując się nazwami tabel logicznych (odpowiednikami perspektyw ukrywających przed użytkownikiem rzeczywisty sposób rozproszenia danych). Stało się to możliwe dzięki przygotowaniu analizatora składniowo-semantycznego, który zadane przez użytkownika zapytanie dekomponuje, przy użyciu globalnego katalogu danych, na podzapytania przeznaczone dla poszczególnych lokalnych serwerów.

Uniwersalny i jednorodny dostęp do danych

Projektując interfejs między użytkownikiem a bazą danych wybrano język SQL ze względu na fakt, że stał się on powszechnie używanym standardem. W języku tym użytkownik formułuje zapytania, przetwarzane następnie przez analizator składniowo-semantyczny. Ewentualna translacja podzapytań SQL na język procesora danych jest zadaniem lokalnego serwera. Na obecnym etapie prac nad systemem umożliwiono wykonywanie uproszczonych wersji następujących instrukcji: SELECT, INSERT, UPDATE oraz DELETE.

Autonomia lokalnej bazy danych

Na obecnym etapie projektu lokalne bazy danych wchodzące w skład bazy rozproszonej obsługiwane są przez dwa procesory danych: Ingres oraz Postgres95. Do zapewnienia autonomii lokalnych baz danych wykorzystano fakt, że realizują tę funkcję mechanizmy obu systemów zarządzania.

Obsługa różnych typów rozproszeń danych

Eksperymentalny system obsługuje następujące rodzaje rozproszeń:

- replikację,
- fragmentację poziomą,
- fragmentację pionową.

Obsługa ta polega na odpowiedniej dekompozycji zapytania realizowanej przez analizator przy użyciu globalnego katalogu danych na podzapytania odnoszące się do właściwych fragmentów bazy rozproszonej. Zwracane przez lokalne serwery wyniki podzapytań ulegają scaleniu w postać oczekiwaną przez użytkownika, zgodną z globalnym schematem danych.

Globalny katalog danych

Słownik danych bazy rozproszonej utworzono jako scentralizowaną bazę danych systemu Ingres. Zapisano w nim informacje o schematach tabel logicznych i tabel fizycznych oraz zależności pomiędzy atrybutami logicznymi i fizycznymi w postaci następujących informacji:

- typ rozproszenia,
- liczba kopii w przypadku replikacji, atrybut wspólny dla fragmentacji pionowej lub dozór dla fragmentacji poziomej
- lokalizacja (określenie węzła),
- przynależność do bazy lokalnej.

Kontrola współbieżnego dostępu do danych

W systemie eksperymentalnym przyjęto dwustopniowy sposób blokowania (blokady logiczne koordynatora oraz blokady procesorów danych). Dzięki temu koordynator może odrzucić transakcję, której wykonanie w danej chwili jest niemożliwe, nie angażując w ten sposób procesorów danych. Blokady logiczne przydzielane są etapami przed każdym kolejnym podzapytaniem. Jednostką granulacji stosowaną przez koordynator jest tabela.

Optymalizacja zapytań rozproszonych

Zapytania sformułowane za pomocą zaimplementowanych wersji instrukcji języka SQL wymagają jedynie dekompozycji, a otrzymane w ten sposób podzapytania podlegają optymalizacji lokalnej dokonywanej przez moduły optymalizatora w poszczególnych procesorach danych.

Protokół wypełnienia transakcji rozproszonej, odtwarzanie skutków transakcji rozproszonej po awarii

Realizacja protokołu wypełnienia transakcji w systemie eksperymentalnym oparta została na przedstawionym w rozdziale 2 protokole 2PC. Komunikacja pomiędzy koordynatorem a lokalnymi serwerami odbywa się poprzez przestrzeń krotek dostępną dla wszystkich modułów procesora aplikacji. Analiza protokołu 2PC w kontekście zastosowanych narzędzi pozwoliła na wprowadzenie modyfikacji polegającej na zredukowaniu liczby komunikatów PREPARE TO COMMIT oraz COMMIT / ROLLBACK (rys. 2). Zamiast rozsyłania komunikatów do każdego z lokalnych serwerów koordynator wprowadza do przestrzeni tylko jedną krotkę z danym komunikatem. Operacja nieniszczącego odczytu krotki zapewnia jej dostępność dla każdego uczestnika protokołu. Krotka z decyzją COMMIT / ROLLBACK pozostaje w przestrzeni tak długo, dopóki nie zostanie odczytana przez wszystkie lokalne serwery. Eliminuje się w ten sposób stan, w którym jeden z uczestników, ze względu na brak łączności z koordynatorem, nie otrzymuje decyzji dotyczącej sposobu zakończenia bieżącej transakcji. Brak decyzji koordynatora równoważny jest z jego awarią. Lokalne serwery podejmują wtedy decyzję o uruchomieniu nowego koordynatora, który na podstawie transakcyjnego pliku log prowadzonego przez swojego poprzednika jest w stanie zarządzać dalszą realizacją transakcji. W pliku log koordynatora znajdują się następujące zapisy:

- początek transakcji,
- początek protokołu wypełnienia,
- decyzja koordynatora,
- koniec transakcji.

Również każdy z lokalnych serwerów prowadzi swój transakcyjny plik log, w którym umieszcza następujące zapisy:

- początek transakcji,

- parametry połączenia z procesorem danych (nazwy lokalnych baz danych, identyfikatory poszczególnych sesji),
- odpowiedź lokalnego serwera,
- decyzję koordynatora,
- koniec transakcji.

Sposób realizacji plików log bazy danych jest uwarunkowany specyfiką użytych procesorów danych. Jeśli jest to możliwe, wykorzystuje się wewnętrzny log procesora danych. W przeciwnym przypadku implementuje się własny plik.

Zastosowanie protokołu 2PC wydaje się zbyteczne w przypadku, kiedy transakcja rozproszona złożona jest wyłącznie z operacji odczytu danych lub jeśli zawarta w niej operacja modyfikacji dotyczy zasobów zarządzanych przez jeden procesor danych. W tych sytuacjach koordynator systemu eksperymentalnego ogranicza komunikację z lokalnym serwerem do pierwszej fazy protokołu 2PC (rys. 2), traktując odpowiedź lokalnego serwera jako decyzję dotyczącą sposobu zakończenia transakcji, zapisywaną następnie w transakcyjnym pliku log.

7. Podsumowanie

Wzrastające zainteresowanie rozproszonymi bazami danych powoduje poszukiwanie nowych rozwiązań w dziedzinie ich zarządzania. Dlatego w ramach prac badawczych przeprowadzono analizę własności, które powinien posiadać system zarządzania rozproszoną bazą danych. Zaowocowała ona opracowaniem projektu i implementacją eksperymentalnego systemu w środowisku programowania równoległego i rozproszonego Paradise, opartego na modelu wirtualnie współdzielonej pamięci. W systemie tym, uruchomionym w lokalnej sieci stacji roboczych Sun, wykorzystano mechanizmy opisane w literaturze, dokonując ich adaptacji oraz modyfikacji pod kątem jak najpełniejszego wykorzystania możliwości użytych środków. W opracowaniu przedstawiono aktualny stan prac, które są kontynuowane zgodnie z przyjętym założeniem, by system stanowił platformę służącą do porównywania znanych rozwiązań, jak również poszukiwania nowych. Zakres tematyki badawczej obejmuje:

- rozszerzenie zestawu zapytań zadawanych do rozproszonej bazy danych,
- rozbudowę systemu o inne procesory danych,
- optymalizację zapytań rozproszonych,
- różne koncepcje globalnego katalogu danych,
- rozproszenie funkcji koordynatora.

Ponadto system wykorzystywany jest w procesie dydaktycznym.

LITERATURA

[1] Ceri S., Pelagatti G.: Distributed Databases. Principles and Systems. McGraw Hill, 1986.

[2] Elmasri R., Navathe B.: Fundamentals of Database Systems. Benjamin Cumings, 1989.

[3] Bernstein P. at al.: Concurrency Control and Recovery in Database Systems. Addison-Wesley, 1987.

[4] Subieta K.: INGRES. System Zarządzania Relacyjną Bazą Danych. Akademicka Oficyna Wydawnicza PLJ, Warszawa, 1994.

[5] Petkovič D.: INGRES. Das relationale Datenbanksystem mit Knowledge-Base und Object-Base. Addison-Wesley, 1992.

[6] Praca zbiorowa pod redakcją S. Kozielskiego: Systemy umożliwiające realizację algorytmów równoległych w sieciach komputerowych. Skrypt Politechniki Śl. nr 1975, Gliwice, 1996.

[7] Bajerski P., Josiński H., Stąpor K.: Zastosowanie systemu Paradise do tworzenia systemu zarządzania rozproszoną bazą danych. Zeszyty Naukowe Politechniki Śl., s. Informatyka, z. 30, Gliwice 1996.

[8] Staniszkis W.: Projektowanie rozproszonych systemów informatycznych. Informatyka nr 1, 1995.

[9] ASK/INGRES Technical Communications: INGRES/ESQL Companion Guide for C.

[10] ASK/INGRES Technical Communications: INGRES/SQL Reference Manual.

[11] ASK/INGRES Technical Communications: INGRES Database Administrator's Guide.

Recenzent: Dr hab.inż. Adam Mrózek
Prof.nadzw. Politechniki Śląskiej

Wpłynęło do Redakcji 25 listopada 1996 r.

Abstract

In the article the authors have analyzed the qualities of the distributed database management system and have displayed its implementation in the experimental system working on the group of the Sun workstations connected by the Ethernet local area network (figure 3). The architecture of the system was based on the model of the virtual shared memory with the usage of the Paradise system. The functions as well as the tasks performed

by application processor have been presented. The application processor consists of the following modules: the distributed transaction coordinator, client and local servers. The global structure of the application processor has been shown on the figure 4. The access to the content of the local databases has been achieved exploiting the technique of embedding SQL commands in C language. For the local databases, the centralized Ingres and Postgres95 systems have been applied as the data processors. The chapter 2 contains the analysis of the distributed database management system's function taking into account the user's needs.