ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1995
Seria: INFORMATYKA z. 28 Nr kol. 1270

Maciej PIECHÓWKA
Henryk KRAWCZYK

ODTWARZANIE TRANSAKCJI W ROZPROSZONYCH SYSTEMACH BAZ DANYCH *

Streszczenie. Przedstawiono problemy odtwarzania rozproszonych obliczeń w systemach baz danych. Zwrócono uwagę na rozwiązania programowe (undo, redo), dotyczące pojedynczej transakcji. Przeanalizowano strategie w przypadku występowania kilku zarządców transakcji, gdy są one wykonywane równolegle. Zasygnalizowano zmodyfikowaną wersję protokołu akceptacji transakcji.

TRANSACTION RECOVERY IN DISTRIBUTED DATA BASE SYSTEM

Summary. The paper presents recovery problems of distributed processing in data base systems. The fundamental hardware (disk mirroring) and software (undo, redo) solutions corresponding to a single transaction are given. Recovery strategies for complex transactions, executed in parallel and co-ordinated by several transaction manager are analysed. The modified version of three phase commit protocol is suggested.

WIEDERHERSTELLUNG DER TRANSAKTIONEN IN VERTEILTE DATABANKSYSTEMEN

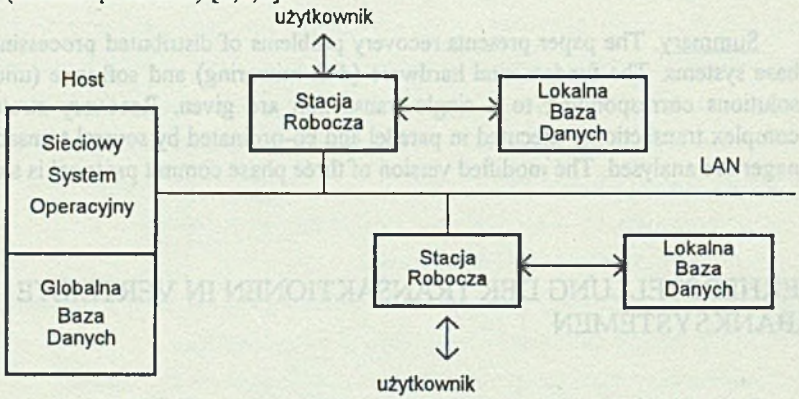
Zusammenfassung. Im Aufsatz werden Probleme der Wiederherstellung verteilten Rechnungen in den Datenbanksystemen dargestellt. Man betrachtet ausführlich Losungen zur einzelnen Transaktion und eine Strategie für parallel realisierte Transaktionen, die einige Verwalters haben. Man zeigte modifizierte Version von Protokoll der Akzeptazion.

* Praca sponsorowana przez grant KBN nr: 8 S503 017 07.

1. Wprowadzenie

System rozproszony składa się z wielu autonomicznych węzłów połączonych za pomocą sieci komunikacyjnej, zdolnych do wykonania wspólnego zadania. Różni się od sieci komputerowej tym, że jest trudniejszy do konstrukcji, ale łatwiejszy do wykorzystania przez użytkownika. Na ogół w systemach takich wykorzystuje się scentralizowane bądź rozproszone bazy danych (patrz rys. 1). Szeroko wykorzystywanym modelem obliczeń w takim systemie jest model klient-serwer [1,2,4], zaś pojęciem podstawowym transakcja, która przekształca system bazy danych z jednego stanu zgodnego w inny stan zgodny. Zgodność może być zachwiana jedynie podczas wykonywania transakcji. Oprócz cechy zgodności transakcja wykazuje własności atomowości (jest niepodzielną jednostką), niezależności (częściowe wykonanie transakcji nie jest widzialne dla innych transakcji) i trwałości (wyniki zakończonej transakcji są na stałe zapamiętane w bazie danych). Odtwarzanie w bazach danych dotyczy zapewnienia przede wszystkim trwałości i atomowości wykonywanych transakcji.

W rozproszonych systemach baz danych również mogą pojawić się błędy. Wyróżniamy przy tym błędy transakcji (związane z pewną nieprawidłowością wewnątrz transakcji, jak np. błędy formatu, danych), błędy węzłów (fizyczne uszkodzenie procesora, błędy w systemie operacyjnym czy pamięci), błędy nośników (błędy obsługi dysku, czy uszkodzenia dysku), błędy sieciowe (uszkodzenie połączeń komunikacyjnych, zakłócenia). W zależności od typu błędów wykorzystywane są różne strategie odtwarzania. W przypadku błędów transakcji następuje powrót do poprzedniego stanu zgodnego, w przypadku błędów węzła jego restart, bądź wymiana, zaś dla błędów nośników celowe jest dublowanie zbiorów na różnych dyskach. Eliminacja błędów komunikacyjnych związana jest z wykorzystaniem protokołów akceptacji (commit protocols) [1,2,4].



Rys. 1. Przykład systemu rozproszonego
Fig. 1. Distributed Data Base Management System Architecture

W pracy przedstawiono formalny opis rozproszonej bazy danych. Zaprezentowano różne strategie odtwarzania pojedynczej transakcji, jak i zbioru równolegle wykonywanych transakcji.

2. Opis funkcjonowania rozproszonej bazy danych

Rozproszoną bazą danych (RBD) nazywać będziemy czwórkę:

RBD = {ST, L, D, LOC}, gdzie:

ST = $\{ST_1,...,ST_n\}$ - zbiór stanowisk komputerowych przechowywania danych,

L = $\{x_1,...,x_n\}$ - zbiór danych logicznych nazywany logiczną bazą danych,

D = $\{x_{11},...,x_{1m_1}, x_{21},...,x_{2m_2},...,x_{n1},...,x_{nm_n}\}$ - zbiór danych fizycznych, nazywany fizyczną bazą danych,

Loc: D --> ST - funkcja określająca lokalizacje danej fizycznej,

$Loc(x_{ik}) \neq Loc(x_{il}), 1 \leq k < l < m_i$.

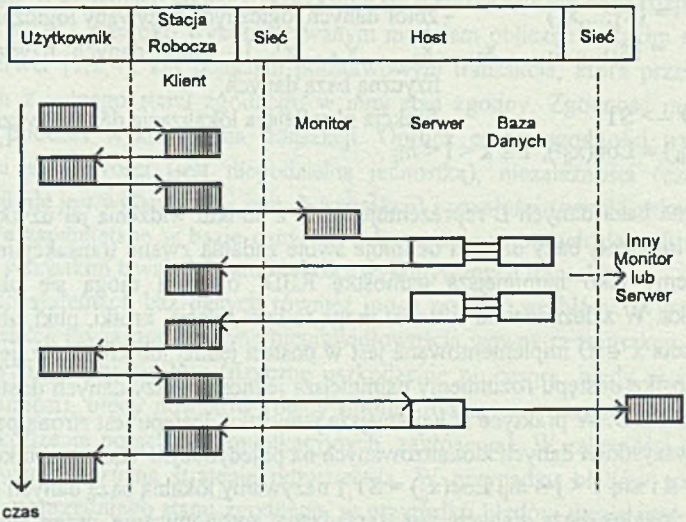
Logiczna baza danych L reprezentuje RBD z punktu widzenia jej użytkownika, który w terminach logicznej bazy danych definiuje swoje zadania zwane transakcjami. Daną logiczną interpretujemy jako najmniejszą jednostkę RBD, o którą mogą się ubiegać transakcje użytkownika. W zależności od systemu mogą to być: relacje, krotki, pliki, zbiory. Pojedyncza dana fizyczna $x \in D$ implementowana jest w postaci jednej lub kilku tzw. jednostek dostępu. Przez jednostkę dostępu rozumiemy najmniejszą jednostkę bazy danych dostępną dla systemu zarządzania RBD. W praktyce ta najmniejsza jednostka dostępu jest stroną pamięci wirtualnej.

Zbiór wszystkich danych zlokalizowanych na pojedynczym stanowisku komputerowym ST_i $\{x_{ij}: 1 \leq i \leq n, 1 < j \leq m_i, Loc(x_{ij}) = ST_i\}$ nazywamy lokalną bazą danych (LBD).

Każda lokalna baza danych jest zarządzana autonomicznie przez system zarządzania lokalną bazą danych SZLBD. Zawiera ona zgodnie z modelem rozproszonego przetwarzania transakcyjnego DTP (Distributed Transaction Processing -DTP X/OPEN, ISO 10026 [4]) następujące elementy: program aplikacyjny (AP) definiujący transakcję oraz wykorzystywane przez nią zasoby systemowe, zarządcę zasobów (RM) oraz zarządcę transakcji (TM) odpowiedzialnego za realizację transakcji, potwierdzający jej poprawne zakończenie i odtwarzający środowisko w przypadku awarii. Użytkownik oddziałuje z RBD poprzez realizacje transakcji. Transakcja może składać się z czterech typów operacji: rozpoczęcia transakcji (Begin transaction), odczytu (Read), zapisu (Write), akceptacji (Commit) i anulowania, unieważnienia (Abort). Operacja odczytu zwraca wartość odczytywanego obiektu bazy danych, zaś operacja zapisu modyfikuje wartość danego obiektu. Operacja akceptacji informuje system bazy danych, że transakcja zakończyła się i że wszystkie dokonane przez nią modyfikacje powinny być trwale odzwierciedlone w bazie danych. Operacja unieważnienia informuje system bazy danych o nienormalnym zakończeniu transakcji i o konieczności przywrócenia stanu obiektów przez nią modyfikowanych do stanu poprzedniego. Transakcja może mieć tylko jedną akcję akceptacji albo odrzucenia.

Funkcjonowanie systemu rozproszonej bazy danych bazuje na mechanizmach zdalnego wywoływania TRPC (ang. transactional remote procedure call, [3]) oraz najczęściej na modelu architektury klient-serwer. Zatem wiele aplikacji może być rozproszonych jak klienci i serwery w sieci komputerowej. Na rys. 2 przedstawiono przepływ wiadomości między klientami i serwerami dla typowej transakcji złożonej. Użytkownik komunikuje się przez odpowiedni interfejs. Klient czyta informacje od użytkownika i komunikuje się z serwerem za

pomocą TRPC. Najpierw monitor wywołuje serwery, a potem serwery współdziałają z bazą danych i klientem. Co więcej, serwer może odwoływać się do innych serwerów i do innych odległych baz danych. Klient może również wywołać kilka serwerów. Takie współdziałanie klient-serwer może być przedstawione jako pojedyncza transakcja lub jako sekwencja transakcji.



Rys. 2. Przykład realizacji transakcji w ujęciu klient-serwer
Fig. 2. An example of the transaction execution in client-server architecture

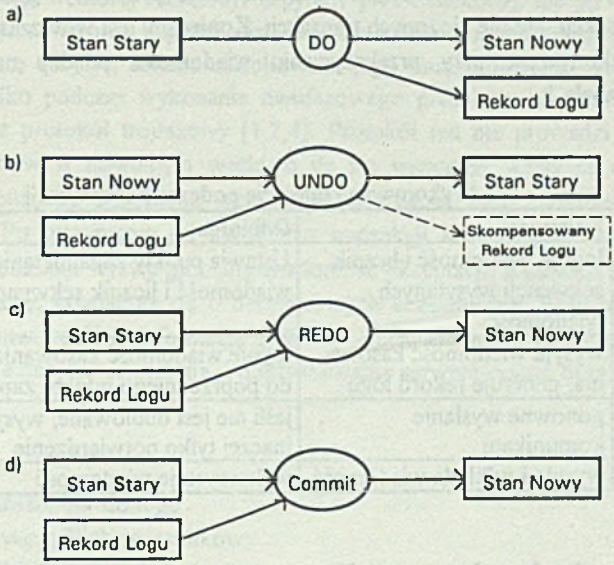
3. Podstawowe protokoły odtworzeniowe

Protokoły odtworzeniowe dzielimy na lokalne i rozproszone [1,3]. Lokalne algorytmy odtwarzania zależą od sposobu realizacji przez zarządcę danych LBD następujących operacji transakcji: begin, read, write, commit, abort, restart. Zgodnie z przyjmowaną architekturą baz danych, głównie ze względu na efektywność systemu, obiekt bazy danych może mieć dwie reprezentacje: ulotną - w pamięci operacyjnej oraz trwałą - na dysku, których wartości chwilowe mogą być różne. Własność integralności obiektu określa, że trwała reprezentacja obiektu w warunkach ustalonych powinna mieć wartość nadaną jej przez ostatnio zaakceptowaną transakcję. Algorytm odtwarzania zależy od strategii zarządcy danych LBD, określającej kiedy w stosunku do momentu wystąpienia operacji akceptacji, unieważnienia transkacji lub wystąpienia uszkodzenia, zostają wyrównywane wartości obu reprezentacji obiektu. Rozróżnia się trzy podstawowe techniki odtwarzania: logu (dziennika), wersji (cieni ang.shadows) oraz plików różnicowych [3].

W technice dziennika wykorzystuje się mechanizmy przedstawione na rys. 3. Zauważmy, że ich wykonanie związane jest z zapisaniem odpowiedniej informacji do logu (dziennika).

Dziennik musi zawierać wszystkie informacje wymagane dla transformacji "zastanego" po uszkodzeniu stanu bazy danych w (najbardziej) ostatni spójny stan bazy danych poprzez:

- usunięcie "brudnych" danych (modyfikacji zrealizowanych przez niezakończone transakcje) z pamiętanej bazy danych,
- uzupełnienie bazy danych o wartości zmodyfikowne przez zakończone (akceptowane) transakcje, które nie były jeszcze w niej zawarte w momencie uszkodzenia.



Rys. 3. Operacje definiujące podstawowe protokoły odtwarzania lokalnego
Fig. 3. The basic operations of the local recovery protocol

Podstawą idei techniki cieni (ang. shadows) jest zasada nie uaktualniania w miejscu, lecz utrzymywania podczas aktywności transakcji dwu kopii obiektu: kopii modyfikowanej i kopii obiektu sprzed rozpoczęcia transakcji. Ta ostatnia z kopii nazywana jest cieniem. Po akceptacji transakcji kopia cienia jest zastępowana kopią uaktualnioną. Zasadniczą zaletą koncepcji cieni jest brak dodatkowych operacji I/O związanych z pamiętaniem w pamięci zewnętrznej "obrazu_przed" modyfikowanego obiektu. Powstają jednak problemy z bardziej skomplikowanym schematem alokacji stron [4]. Algorytmy odtwarzania w technice cieni mogą być klasyfikowane zgodnie z mechanizmem stosowanym przy atomowej modyfikacji tablicy stron:

1. tradycyjny schemat tworzy cienie zmian bazy danych oraz zapisuje w logu zmiany w tablicy stron, jest więc hybrydowym mechanizmem dziennika i cieni,
2. stosuje się cieniowanie tak tablicy stron, jak i bazy danych. Zaletą tego wariantu jest prawie natychmiastowe odtworzenie stanu sprzed uszkodzenia systemowego.

Ocena mechanizmów odtwarzania lokalnego uwzględnia zwykle dwa kryteria: nakład podczas normalnego przetwarzania oraz szybkość odtwarzania po uszkodzeniu. Wybór algorytmu odtwarzania zależy także od charakterystyk i wymagań środowiska aplikacji.

W praktyce zakłada się, że podsystem odtwarzania musi zapewnić zniesienie efektów wycofanej transakcji (UNDO), odtworzenie stanu bazy danych po błędzie węzła (całkowite UNDO częściowe REDO), bądź odtworzenie po błędzie nośnika (całkowite REDO). W przypadku systemów rozproszonych przedstawione protokoły gwarantują atomowość prostych transakcji, ale nie złożonych transakcji. Konieczny jest wówczas protokół akceptacji. Operacje wykonywane przy przekazywaniu wiadomości między nadawcą a odbiorcą przedstawia tabela 1.

Tabela 1

Wykonywane operacje podczas komunikacji

Operacje	Nadawca	Odbiorca
DO	loguje wiadomość i licznik sekwencji wysyłanych wiadomości	Ustawia punkty zapamiętania stanu, loguje wiadomość i licznik sekwencji potwierdzeń
UNDO	wysyła wiadomość kasowania, generuje rekord logu	loguje wiadomość kasowania, dokonuje powrotu do poprzedniego punktu zapamiętania stanu
REDO	ponowne wysłanie komunikatu	jeśli nie jest dublowane, wysyła operację DO, inaczej tylko potwierdzenie
COMMIT	wysyła i odkłada wiadomość	wykorzystuje wiadomość

4. Protokoły akceptacji

Transakcje dotyczące kilku podsystemów systemu rozproszonego angażują wielu zarządców transakcji, przy czym każdy z zarządców może także uszkadzać się i restartować niezależnie. W takich przypadkach, w celu zapewnienia atomowości wykonywanej transakcji, wykorzystuje się dwu- lub trójfazowy protokół akceptacji (commit protocols). Jeśli jeden z zarządców uszkadza się podczas wykonywania tego protokołu, wówczas konieczne są dodatkowe funkcje jego restartu i ponownego dołączenia się do tej grupy. Protokół akceptacji jest realizowany przez zarządcę transakcji występującego w roli koordynatora (nadzorcy) i sprowadza się do zapytania wszystkich uczestników wykonujących tę transakcję, czy jest ona zgodna i zakończona (faza 1). Jeśli wszyscy uczestnicy są zgodni, że tak, wówczas transakcja jest akceptowana i koordynator zapamiętuje ten fakt w logu, informując jednocześnie wszystkich uczestników o zaistniałej zgodności (faza 2). Od tej chwili każdy zarządca transakcji może zwolnić wszelkie blokady i przejść do wykonania innych transakcji. Jeśli uczestnicy nie są zgodni co do zakończenia transakcji, bądź wystąpił błąd, protokół akceptacji przewiduje wykonanie nawrotu. W tym przypadku zarządca transakcji czyta log dotyczący tej transakcji i dla każdego rekordu wykonuje operację undo. Gdy operacja undo kończy się sukcesem, zarządca transakcji informuje o zaniechaniu transakcji. Typowy program transakcji wykorzystującej taki protokół może być przedstawiony następująco:

```
Begin_ work( );              / rozpoczęcie transakcji
read_window( );              / wykorzystujemy okna jako zasoby
do_some_SQL( );              / wykorzystanie bazy danych
write_windows( );            / wykorzystanie okna do komunikacji zwrotnej
if (ok) Commit_Work ( )      / zaakceptowanie transakcji
else Rollback_Work ( )       / powrót do poprzedniego punktu zapamiętania stanu
```

Rozpatrzmy sytuację, w której server był zapytany przez nadzorcę, ale po wymaganym oczekiwaniu nie otrzymał wyniku uzgodnienia. Jeśli nie może skomunikować się z innymi węzłami, które znają tę decyzję, zostaje zablokowane jego funkcjonowanie. Takie sytuacje występują raczej rzadko podczas wykonania dwufazowego protokołu akceptacji, niemniej proponuje się również protokół trójfazowy [1,2,4]. Protokół ten nie prowadzi do blokady przy wystąpieniu błędów w dowolnym węźle, o ile nie wszystkie węzły są uszkodzone. Proponowane udoskonalenie polega na wprowadzeniu trzeciej fazy, ulokowanej między poprzednimi fazami. Po otrzymaniu od zarządców transakcji informacji o wykonaniu się transakcji nadzorca transakcji wysyła globalną wiadomość akceptacji. Realizatorzy transakcji otrzymując tę informację, dowiadują się o decyzji innych uczestników. Wówczas zarządca transakcji powinien potwierdzić tę informację globalną w celu eliminacji blokady. W trzeciej fazie serwer podejmuje decyzję ostateczną. Funkcjonowanie serwera opisać można w sposób następujący:

```
begin_serwer
 step1 "vote"
   write "begin transaction" do logu
   send "vote" do wszystkich uczestników
   do until wszystkie "vote" będą otrzymane
      wait
      on time out go to step2b
   end_do

 step2a_pre-commit
   if wszystkie "vote" są zaakceptowane
     then begin
        write wiadomość "pre-commit" do logu
        send "pre commit" do wszystkich uczestników

step 2b global abort
 / przynajmniej jeden z uczestnikówow sygnalizuje "abort" lub został przekroczony czas
 oczekiwania przez zarządcę */
  else begin
    write wiadomość "global abort" do logu
    send "global abort" do wszystkich uczestników
    go to step 4
      end
 end_if
```

```
step 3 global commit
  do until wszystkie potwierdzenia pre-commit są odebrane
  wait
  end_do
  write "global commit" do login
  send "global commit" do wszystkich uczestników

step4 termination
  do until potwierdzenia zostały odebrane od uczestników
    wait
  end_do
  write "end transaction" do logu
  finish
end
```

W podobny sposób opisuje się zachowanie danego uczestnika wykonywania transakcji złożonej. Co więcej, uczestnik może założyć, że serwer jest błędny. Wówczas istnieje możliwość wyboru nowego zarządcy. Przedstawiona strategia jest zbliżona do rozproszonej strategii diagnostycznej, w której dokonuje się wyboru węzła decydującego o wynikach testowania [6].

Istnieją zastosowania związane na ogół z systemami zarządzania, w których cztery podstawowe własności tradycyjnych transakcji złożonych nie mogą być spełnione [5]. W takich przypadkach protokół akceptacji dotyczy tylko podzbioru transakcji stanowiących zadaną transakcję złożoną. Wówczas istotne staje się odtwarzanie wyprzedzające (forward) polegające na wykonaniu alternatywnego zestawu transakcji prostych. Takie wariantowe transakcje zwiększają funkcjonalność systemu, a także umożliwiają uwzględnienie autonomiczności obiektów w rozproszonym środowisku heterogenicznym. Wymaga to jednak oddzielnego opracowania.

LITERATURA

[1] Bell D., Grimson J.: Distributed Data Base Systems. Addison-Wesley Publishing Company, 1992.

[2] Bernstein P., Hadzilacos V., Goodman N.: Concurrency Control and Recovery in Database Systems. Addison-Wesley Publishing Company, 1987.

[3] Date C.J.: An Introduction to Database Systems. Addison-Wesley Publishing Company, 1985.

[4] Gray J., Reuter A.: Transaction Processing: Concept and Techniques. The Morgan Kaufman Series in Data Management System, 1993.

[5] Elmagarmid A. (Ed.), Advanced Transaction Models for New Applications. The Morgan-Kaufmann Series in Data Management System, 1992.

[6] Krawczyk H.: Analiza i synteza samodiagnozowalnych systemów komputerowych. Prace Naukowe Politechniki Gdańskiej, Nr 64, 1987.

[7] Lee P.A., Anderson T.: Fault-Tolerance. Principles and Practice. Springer-Verlag, 1990.

Recenzent: Dr hab. inż. Stanisław Kozielski
Prof. Politechniki Śląskiej

Wpłynęło do Redakcji 1 grudnia 1994 r.

Abstract

The paper presents recovery problems of distributed processing in data base systems. We consider a typical configuration (see Fig.1) consisted of a file server with a global database and several workstations with local databases. All these nodes connected by typical communication links create a local area network. We describe formally such a system as quadruple: RBD = {ST, L, D, LOC} where ST is a set of nodes (stations), L represents logical data named data base, D corresponds to the physical data named physical data base, and LOC determines the memory location of physical data. Using client-server model an example of transaction processing in the distributed system is shown in Fig.2. The fundamental recovery protocol for simple transactions are defined (see Fig.3) and main functions of senders and receivers are given in Table 1. Then the recovery problem of complex transactions is considered. The complex transaction models are described in terms of dependencies between their components. Transaction dependencies can be viewed as constraints on the execution structure of the complex transaction components and the correctness criteria that such transactions must satisfy. We present two and three-phase commit protocols. All their phases are described and discussed from sender - receiver points of view. Some modifications are also suggested. Moreover, there is a need for various extensions to the classical model to improve functionality and performance. Then the recovery can be often reduced to forward recovery by using n-version and compensating transaction structures.