ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1995

Seria: INFORMATYKA z. 29 Nr kol. 1306

Mirosław CHŁOPEK
Henryk JOSIŃSKI

JĘZYK CLIPPER - KORZYSTANIE Z MECHANIZMÓW ŚLEDZENIA TRANSAKCJI W SYSTEMIE NETWARE 3.11

Streszczenie. W pracy omówiono zarządzanie transakcjami w systemie NetWare 3.11 i przedstawiono zestaw funkcji udostępnianych przez ten system dla śledzenia transakcji w aplikacjach. Rozważono możliwości użycia tych funkcji w programach pisanych w języku Clipper. Omówiono również problem zabezpieczeń przed utratą lub niespójnością danych w aplikacjach nie korzystających z mechanizmów śledzenia transakcji.

THE CLIPPER LANGUAGE - USAGE OF THE TRANSACTION TRACKING MECHANISMS IN THE NETWARE 3.11 SYSTEM

Summary. The paper discusses the transaction management in the NetWare 3.11 System and presents a set of the functions offered by the system for a transaction tracking in applications. The discussion of the possibilities of the usage of these functions in programms written in the Clipper language is included in this paper. The problem of protection against data loss and data inconsistence in the applications which don't use the transaction tracking was also considered.

PROGRAMMIERSPRACHE CLIPPER - BENUTZUNG DER MECHANISMEN DER TRANSAKTIONSBEOBACHTUNG IM SYSTEM NETWARE 3.11

Zusammenfassung. Im Artikel wurde die Transaktionsverwaltung im System NetWare 3.11 besprochen. Der Satz der zwecks der Transaktionsbeobachtung vom System zugänglich gemachten Funktionen wurde dargestellt. Die Möglichkeiten ihrer Verwendung in den in der Clipper-Sprache geschriebenen Programmen hat man erörtert. Auch das Problem der Absicherung vor dem Datenverlust und vor der Dateninkonsistenz in den Applikationen, die die o.g. Mechanismen nicht benutzen, hat man in diesem Artikel diskutiert.

1. Wstęp

Transakcje stanowią jeden z ważnych mechanizmów pozwalających na utrzymanie spójności bazy danych pomimo wystąpienia zdarzeń temu zagrażających. Zaleca się więc, aby w aplikacjach, które będą funkcjonowały w systemie NetWare, wykorzystywać funkcje obsługujące transakcje w tym systemie. Język Clipper, stanowiący nadal popularne narzędzie do tworzenia aplikacji związanych z bazami danych, w wersjach użytkowanych przez autorów (Summer'87, CA 5.2c), nie posiada wbudowanych elementów obsługi transakcji. W pracy omówiono więc zarządzanie transakcjami w systemie NetWare 3.11 i przedstawiono zestaw funkcji udostępnianych przez ten system w celu wykorzystania mechanizmów śledzenia transakcji w aplikacjach. Rozważono również problem zabezpieczeń przed utratą lub niespójnością danych w aplikacjach nie korzystających z tych mechanizmów.

2. Mechanizmy transakcji

Transakcja jest operacją wykonywaną na spójnym systemie danych, która prowadzi do nowego stanu spójnego. Aby wykluczyć możliwość znalezienia się w stanie niespójnym, transakcja musi być operacją niepodzielną. Przetwarzanie baz danych w sieci powinno być realizowane tylko w środowisku, w którym serwer plików zapewnia środki

niepodzielnego przetwarzania transakcji. W przeciwnym razie wyłączenie bądź uszkodzenie stacji roboczej lub serwera w czasie przetwarzania transakcji grozi niespójnością danych.

Zbiór operacji odczytu, przetwarzania i zapisu, składający się na jedną transakcję, musi być jawnie określony przez programistę w zależności od konkretnego zastosowania. Wprowadzono więc:

- instrukcje do oznaczenia początku i końca transakcji,
- instrukcję umożliwiającą wycofanie rozpoczętej transakcji tak, aby nie pozostawiła ona żadnych skutków.

Transakcje są zwykle wykonywane współbieżnie przez wiele stacji roboczych. Przy braku synchronizacji dostępu do zasobów współbieżne wykonywanie zbioru transakcji w systemie rozproszonym rozumiane jako przeplatanie się operacji cząstkowych, należących do różnych transakcji, może dać inny wynik niż ich realizacja szeregowa, co jest niedopuszczalne. Z drugiej strony współbieżne wykonywanie zbioru transakcji jest korzystne ze względu na efektywność przetwarzania. Należy więc przy tym sposobie przetwarzania zapewnić uzyskanie identycznych wyników jak przy realizacji szeregowej. Najczęściej używanym środkiem do osiągnięcia tego celu są *blokady* ograniczające dostęp do zasobów.

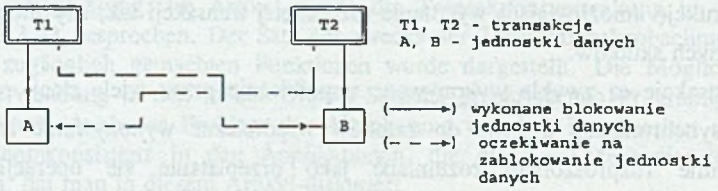
3. Mechanizmy blokady

W trybie pracy wymagającym stosowanie blokad pierwszy dostęp do danych musi zostać poprzedzony wykonaniem operacji zablokowania dostępu do danych innym użytkownikom. Udana operacja blokowania pozwala na dostęp do danych w transakcji, w której wykonano blokowanie. Przy nieudanym blokowaniu (próbie blokowania danych uprzednio zablokowanych przez inną transakcję) transakcja nie może być kontynuowana do czasu zdjęcia blokady. Taka definicja mechanizmów blokady czyni z nich również mechanizmy synchronizacji.

Operacje blokowania mogą dotyczyć różnej wielkości *jednostek danych*. Najczęściej spotykanymi wielkościami są plik i rekord. Wybór jednostki zależy od rodzaju operacji, które mają być wykonane na danych. Drugi podział blokad dotyczy sposobu blokowania. Przedstawione wyżej reguły odpowiadają *blokowaniu całkowitemu* (*na wyłączność*) (ang. exclusive-lock), oddającemu transakcji prawa wyłączności w dostępie do danych. Do wykonania odczytu wystarczy wykonanie *blokowania dzielonego* (ang. read-lock, shared-lock), które może być wykonane z sukcesem przez wiele transakcji.

Można spotkać jeszcze inne warianty blokowania, jak na przykład w języku Clipper, gdzie (z pewnym uproszczeniem) wykonanie odczytu nie wymaga założenia żadnej blokady, założenie zwykłej blokady pozwala na zapis, zaś blokowanie na wyłączność stosowane jest do operacji specjalnych.

Blokowanie a impas



Rys. 1. Ilustracja zjawiska blokady wzajemnej
Fig. 1. Illustration of the deadlock

Współbieżna realizacja dwóch lub więcej transakcji operujących na tych samych jednostkach danych i stosujących mechanizmy blokady może doprowadzić do tzw. *impasu* (*blokady wzajemnej*, *zakleszczenia*). Dochodzi do niego w najprostszym przypadku wtedy, gdy dwie transakcje T_1 i T_2 zamierzają operować na dwóch jednostkach danych (A i B) w takiej kolejności, że T_1 blokuje najpierw A, zaś w podobnym czasie T_2 blokuje B, po czym obie transakcje podejmują próby zablokowania pozostałych jednostek, tzn. T_1 - B, zaś T_2 - A (rys. 1). Jeśli jednostki A i B nie zostały przedtem odblokowane, to w systemie wystąpi impas wymagający działania specjalnego. Istnieje możliwość stosunkowo prostego wychodzenia z impasu przy wykorzystaniu mechanizmu wycofywania transakcji.

3.1. Przegląd mechanizmów blokowania w języku Clipper

Mechanizmy blokowania podzielono w języku Clipper na tzw. tryby dostępu i funkcje blokujące.

Rozróżnia się dwa rodzaje trybów dostępu:

- dostęp wyłączny (ang. exclusive),
- dostęp dzielony (ang. shared).

Dostęp do pliku w trybie wyłącznym uzyskuje się używając frazy EXCLUSIVE w instrukcji USE lub też wykonując instrukcję SET EXCLUSIVE ON przed użyciem instrukcji USE. W tym drugim przypadku wszystkie kolejne instrukcje USE otwierają

pliki w trybie wyłącznego dostępu. Tryb wyłączny jest przyjmowany w języku Clipper jako tryb domyślny.

Dostęp do plików w trybie dzielonym uzyskuje się poprzedzając instrukcje ich otwarcia wykonaniem instrukcji SET EXCLUSIVE OFF (w wersji CA 5.2c można również dla poszczególnych plików użyć USE...SHARED lub USE...READONLY).

Jeśli plik danych zostanie otwarty w trybie wyłącznym, to żaden inny program nie może otworzyć tego pliku ani w trybie wyłącznym, ani w trybie dzielonym.

Plik danych otwarty przez jeden program w trybie dzielonym może zostać następnie otwarty (tylko w trybie dzielonym) przez inne programy. Tak więc kilka programów może równocześnie mieć dostęp do jednego pliku danych w trybie dzielonym.

Możliwe jest programowe sprawdzenie poprawności realizacji instrukcji dotyczącej operacji sieciowej (np. USE) za pomocą funkcji NETERR(). W rezultacie wykonania tej funkcji otrzymuje się wartość logiczną .T., jeśli instrukcja poprzedzająca zakończy się niepowodzeniem.

W języku Clipper istnieją dwie funkcje wykonujące operacje blokowania. Funkcja FLOCK() wykonuje operację blokowania dostępu do całego pliku danych, otwartego wcześniej w trybie dzielonym. Inne programy mogą otwierać (w trybie dzielonym) zablokowany plik i dokonywać odczytu zawartości rekordów. Żaden rekord nie może jednak być blokowany ani modyfikowany przez inne programy.

Funkcja RLOCK() wykonuje operację blokowania dostępu do bieżącego rekordu w pliku otwartym w trybie dzielonym. Rekord taki może być następnie odczytywany przez inne programy, nie może jednak zostać powtórnie zablokowany ani zmieniony.

W wyniku wykonania funkcji FLOCK() i RLOCK() otrzymuje się wartość .T., jeśli operacja blokady powiedzie się.

Odblokowanie dostępu do pliku lub rekordu może zostać wykonane jednym z następujących sposobów:

- wykonanie instrukcji UNLOCK (instrukcja ta dotyczy pliku w aktualnie wybranym obszarze roboczym),
- wykonanie instrukcji UNLOCK ALL (odblokowanie następuje we wszystkich aktywnych obszarach roboczych),
- zamknięcie pliku,
- zakończenie programu,
- zablokowanie w programie innego rekordu w danym pliku powoduje odblokowanie rekordu poprzednio zablokowanego przez ten program w tym pliku.

Z ostatniego punktu wynika, że niemożliwe jest zablokowanie w jednym pliku przez ten sam program więcej niż jednego rekordu za pomocą RLOCK().

Ogólne zasady dotyczące blokowania można sformułować następująco:

- blokowanie jest wymagane w każdym przypadku, kiedy ma być wykonany zapis do pliku danych,
- blokowanie jest opcjonalne we wszystkich innych przypadkach.

Podjęcie próby zapisu do pliku otwartego w trybie dzielonym bez uprzedniego zablokowania pliku lub rekordu powoduje błąd systemowy.

4. Język Clipper a ochrona danych w NetWare 3.11

W trakcie realizacji programu użytkowego może dojść do zdarzeń, które będą prowadzić do jego błędnego zakończenia. Zdarzenia te można podzielić następująco:

a) błędy programowe,
b) awarie w stacjach roboczych,
c) awarie w serwerze plików.

Błędy programowe są błędami zawartymi w aplikacji. Do tej grupy można zaliczyć na przykład:

- użycie zmiennej nie zainicjalizowanej,
- próbę zapisu rekordu nie zablokowanego (w przypadku pracy w trybie dzielonym).

Zdarzenia typu b) i c) mogą powodować utratę efektów modyfikacji danych dokonanych w aplikacji (operacje wstawiania, aktualizacji, usuwania). Ewentualność ta wynika z faktu wykorzystywania przez język Clipper bufora umieszczonego w pamięci operacyjnej, pośredniczącego w zapisie przetworzonych danych na dysk. Jeśli awaria nastąpi przed lub w trakcie zapisu zawartości bufora na dysk, dochodzi do utraty danych przechowywanych w buforze.

Do zdarzeń określanych jako awarie w stacjach roboczych można zaliczyć między innymi:

- zanik zasilania,
- reset komputera.

Analogiczne zdarzenia mogą zajść również w serwerze plików.

W zależności od umieszczenia plików bazy (dysk lokalny lub dysk serwera plików) istnieje różnica w wielkości utraconych danych.

4.1. Praca z bazą danych umieszczoną na dysku lokalnym

W zależności od rozmiaru pojedynczego rekordu bufor danych może pomieścić od jednego do kilkuset rekordów. Każdorazowe zapełnienie bufora powoduje zapis jego zawartości do plików dyskowych. Wystąpienie zdarzenia należącego do grupy b) lub c) przed zapisem danych znajdujących się w buforze na dysk prowadzi do utraty efektów modyfikacji danych. Jedyną metodą zabezpieczającą przed tą stratą jest użycie instrukcji COMMIT po każdej instrukcji modyfikującej dane. COMMIT powoduje zapisanie na dysku zawartości wszystkich aktualnie otwartych obszarów roboczych. Trzeba jednak w tym miejscu podkreślić, że częsta realizacja tej instrukcji może prowadzić do wyraźnego wydłużenia czasu realizacji programu. Należałoby więc sugerować używanie instrukcji COMMIT nie po każdorazowej modyfikacji danych, ale po modyfikacji większego zestawu danych, najlepiej spójnego logicznie.

4.2. Praca z bazą danych umieszczoną na dysku serwera plików

W zależności od trybu otwarcia pliku bazy danych występują różnice w sposobie wykorzystania bufora danych:

- tryb wyłączny - sposób wykorzystania bufora danych jest analogiczny do pracy z plikami bazy umieszczonymi na dysku lokalnym,
- tryb dzielony - zapis efektów modyfikacji rekordu na dysk następuje nie tylko po całkowitym zapełnieniu bufora, lecz także po zmianie położenia wskaźnika bieżącego rekordu lub po zwolnieniu blokady ze zmodyfikowanego rekordu, co jednak może nastąpić z pewnym opóźnieniem związanym z funkcjonowaniem sieciowego systemu operacyjnego.

Użycie w programie instrukcji COMMIT zabezpiecza przed utratą zmodyfikowanych danych.

4.3. Wykorzystanie mechanizmów transakcyjnych NetWare 3.11

Podczas pracy z bazą danych na serwerze można wykorzystać mechanizmy transakcyjne TTS (Transaction Tracking System) systemu operacyjnego NetWare. Pakiet TTS stanowi integralną część systemu NetWare, tym niemniej można doprowadzić do zablokowania mechanizmów transakcyjnych:

- za pomocą komendy **DISABLE TTS** wydanej z konsoli operatora,
- wewnątrz programu FCONSOLE,

- wywołując w aplikacji odpowiednią funkcję opisaną w podrozdziale 4.3.2 (dwie ostatnie wymienione możliwości dotyczą użytkownika posiadającego odpowiednie prawa),
- w przypadku zapełnienia wolumenu SYS lub niedostatecznej wielkości pamięci serwera bazy danych.

Warunkami koniecznymi funkcjonowania mechanizmu transakcyjnego są:

- aktywny system TTS,
- włączony atrybut transakcyjności pliku bazy danych i plików z nim skojarzonych (zawierających pola notatnikowe (MEMO) oraz plików indeksowych).

Uaktywnienie wyłączonego systemu TTS następuje:

- za pomocą zlecenia **ENABLE TTS** wydanego z konsoli operatora,
- wewnątrz programu FCONSOLE,
- przez wywołanie w aplikacji odpowiedniej funkcji opisanej w podrozdziale 4.3.2 (dwie ostatnie wymienione możliwości dotyczą użytkownika posiadającego odpowiednie prawa).

Włączenie atrybutu transakcyjności pliku jest efektem zlecenia **FLAG** *nazwa_pliku* **+T**, wydanego na stacji roboczej lub wywołania odpowiedniej funkcji opisanej w podrozdziale 4.3.2. Włączenie i wyłączenie atrybutu transakcyjności może nastąpić tylko dla pliku zamkniętego. Plik z włączonym atrybutem transakcyjności nie może zostać usunięty (a więc również przemianowany i przemieszczony).

Pojęcie transakcji dla systemu TTS obejmuje ciąg zapisów danych na dysk (w szczególnym przypadku transakcję może stanowić pojedyncza operacja zapisu). Należy zwrócić uwagę, że na stacji roboczej w danym czasie może być otwarta tylko jedna transakcja (przez aplikację napisaną w języku Clipper), która swoim zasięgiem obejmuje wszystkie pliki z atrybutem transakcyjności, do których zapisywane są dane. Otwarcie transakcji powoduje wykonanie kopii danych w obszarze specjalnie do tego przeznaczonym, natomiast modyfikacje są przeprowadzane na danych oryginalnych. Po zakończeniu transakcji kopia jest usuwana. Jeśli w systemie wystąpiła awaria (transakcja nie została zakończona), to kopia może zostać wykorzystana do odtworzenia stanu pliku z chwili rozpoczęcia transakcji. Sposób funkcjonowania systemu TTS zależy od rodzaju zdarzenia, które spowodowało taką sytuację (zdarzenia typu: awaria stacji roboczej lub awaria serwera plików).

W przypadku awarii stacji roboczej wszelkie niedokończone transakcje są wycofywane automatycznie. Ponadto, nadając systemowej fladze **TTS Abort Dump Flag** wartość ON, można spowodować zapis szczegółowych informacji o wycofanej transakcji do pliku TTS$LOG.ERR, znajdującego się w kartotece głównej wolumenu SYS.

W przypadku awarii serwera plików, podczas jego ponownego startu, system TTS pozwala na wycofanie niedokończonej transakcji wykorzystując zapisy o jej przebiegu. W zależności od stanu flagi systemowej **Auto TTS Backout Flag** system będzie wycofywać transakcję bezwarunkowo (ON) lub zażąda potwierdzenia od użytkownika na konsoli operatora (OFF - wartość domyślna). Zapis szczegółowych informacji o niedokończonych transakcjach do pliku TTS$LOG.ERR nie jest tym razem uzależniony od stanu flagi systemowej **TTS Abort Dump Flag**. Jeśli flaga **Auto TTS Backout Flag** jest wyłączona (OFF), zapis tych informacji zależy od odpowiedzi operatora na pytanie systemu zadane w trakcie jego instalacji, natomiast w przeciwnym przypadku żadne informacje szczegółowe nie zostaną zapisane. Należy zwrócić uwagę na fakt, że decyzja użytkownika o rezygnacji z wycofania niedokończonej transakcji może spowodować pojawienie się w pliku danych wartości przypadkowych, jeśli awaria nastąpiła w trakcie zapisu danych na dysk.

Podczas pracy z systemem TTS rozróżnia się transakcje *jawne* (ang. explicit) i *niejawne* (ang. implicit). W trybie wyłącznym użytkownik może jedynie posługiwać się transakcjami jawnymi. Jeśli z nich w aplikacji nie korzysta, to praca z plikami danych jest analogiczna do pracy na dysku lokalnym. Natomiast w trybie dzielonym dostępne są oba typy transakcji.

4.3.1. Transakcje niejawne

Początek transakcji niejawnej określony jest przez założenie blokady dostępu do rekordu lub pliku. Natomiast transakcja uważana jest za zakończoną, jeśli nastąpiło zdjęcie założonej blokady, na przykład za pomocą jednej z następujących instrukcji:

- UNLOCK,
- USE,
- CLOSE.

W przypadku jednoczesnego otwarcia większej liczby plików danych transakcja będzie uważana za zakończoną, jeśli wyżej wymienione instrukcje zostaną wykonane we wszystkich używanych obszarach roboczych. Należy zaznaczyć, że wykonanie instrukcji COMMIT na pliku z włączonym atrybutem transakcyjności nie powoduje trwałego zapisu dokonanych modyfikacji (nie kończy transakcji), tzn. w razie wystąpienia awarii efekty modyfikacji będą podlegały procesowi wycofywania.

4.3.2. Transakcje jawne

Do mechanizmów systemu TTS, służących do określania transakcji jawnych, należy zestaw funkcji pozwalających na:

- określenie dostępności systemu TTS,

- określenie początku i końca transakcji,
- wycofanie otwartej transakcji.

W aplikacji napisanej w wersji Summer'87 języka należy przygotować podprogramy wywołujące przedstawione poniżej funkcje za pomocą przerwania int $21 systemu DOS. Natomiast wersję CA 5.2c uzupełnia zewnętrzna biblioteka o nazwie CA-Clipper Tools Extension Library for DOS Ver. 3.0, udostępniająca m.in. odpowiedniki tych funkcji.

Opis funkcji służących do współpracy z systemem TTS:

1. Extended File Attributes

   int $21

   Wejście:

   AH ⇐ $B6

   AL ⇐ { $00 - pobranie rozszerzonych atrybutów pliku
   $01 - ustawienie rozszerzonych atrybutów pliku zapisanych w CL; bit 4. (w numeracji 0÷7) reprezentuje atrybut transakcyjności

   DS:DX⇐ wskazanie nazwy pliku (max. 255 bajtów w formacie ASCIZ)

   Wyjście:

   przy bicie przeniesienia równym 0 rejestr CL zawiera aktualny zestaw rozszerzonych atrybutów pliku

   bit przeniesienia równy 1 oznacza wystąpienie błędu

   CA-Clipper Tools

   NNETEXTATT()

   Funkcja umożliwia ustawienie lub wyzerowanie atrybutu transakcyjności.

2. Disable Transaction Tracking

   int $21

   Wejście:

   AH ⇐ $E3

   DS:SI ⇐ adres bufora żądania posiadającego następującą strukturę:

Przesunięcie	Rozmiar	Zawartość
$00	WORD	$0001 (długość ciągu danych)
$02	BYTE	$CF (funkcja "Disable TTS")

   ES:DI ⇐ adres bufora odpowiedzi posiadającego następującą strukturę:

Przesunięcie	Rozmiar	Zawartość
$00	WORD	$0000 (pusty bufor odpowiedzi)

Wyjście:

AL = { $00 - sukces
$C6 - użytkownik nie posiada praw operatora konsoli

CA-Clipper Tools

NNETDISTTS()

Funkcja blokuje aktywny system TTS.

3. Enable Transaction Tracking

int $21

Wejście:

AH ⇐ $E3

DS:SI ⇐ adres bufora żądania posiadającego następującą strukturę:

Przesunięcie	Rozmiar	Zawartość
$00	WORD	$0001 (długość ciągu danych)
$02	BYTE	$D0 (funkcja "Enable TTS")

ES:DI ⇐ adres bufora odpowiedzi posiadającego następującą strukturę:

Przesunięcie	Rozmiar	Zawartość
$00	WORD	$0000 (pusty bufor odpowiedzi)

Wyjście:

AL = { $00 - sukces
$C6 - użytkownik nie posiada praw operatora konsoli

CA-Clipper Tools

NNETENTTS()

Funkcja uaktywnia zablokowany system TTS.

4. TTS Is Available

int $21

Wejście:

AH ⇐ SC7

AL ⇐ $02

Wyjście:

AL = { $00 - niedostępny TTS
$01 - dostępny TTS
$FD - zablokowany TTS

CA-Clipper Tools

NNETISTTS()

Funkcja sprawdza dostępność i aktywność systemu TTS.

5. TTS Begin Transaction

int $21

Wejście:

AH ⇐ $C7

AL ⇐ $00

Wyjście:

przy bicie przeniesienia równym 0:

AL = $00 - sukces

przy bicie przeniesienia równym 1:

AL = { $96 - brak pamięci
$FE - początek transakcji jawnej
$FF - kontynuacja aktywnej transakcji

CA-Clipper Tools

NNETTTSBEG()

Funkcja informuje o rozpoczęciu przez stację roboczą nowej transakcji. System TTS przekazuje do programu wartość $FE w przypadku, kiedy w chwili wywołania funkcji istnieje blokada rekordów. Jeśli wcześniej nie wywołano systemowej funkcji końca transakcji, to zablokowane rekordy zostaną objęte przez rozpoczynaną właśnie transakcję. Z tego względu wskazane jest, by wywołanie funkcji końca lub przerwania transakcji poprzedzone było zdjęciem wszystkich blokad.

6. TTS Abort Transaction

int $21

Wejście:

AH ⇐ $C7

AL ⇐ $03

Wyjście:

przy bicie przeniesienia równym 0:

AL = $00 - sukces

przy bicie przeniesienia równym 1:

AL = { $FD - zablokowany TTS
$FE - funkcja wykonana, rekord(y) wciąż zablokowane
$FF - nie ma jawnej transakcji

CA-Clipper Tools

NNETTTSAB()

Funkcja przerywa bieżącą transakcję jawną (rozpoczętą wywołaniem funkcji TTS Begin Transaction). Stan plików będzie dokładnie taki, jak w momencie wywołania funkcji początku transakcji.

7. TTS End Transaction

int $21

Wejście:

AH ⇐ $C7

AL ⇐ $01

Wyjście:

CX:DX zawiera numer transakcji

przy bicie przeniesienia równym 0:

AL = { $00 - sukces
$FD - zablokowany TTS
$FE - funkcja wykonana, rekord(y) wciąż zablokowane

przy bicie przeniesienia równym 1:

AL = $FF - nie ma jawnej transakcji.

CA-Clipper Tools

NNETTTSEND()

Funkcja kończy bieżącą transakcję.

8. TTS Transaction Status

int $21

Wejście:

AH ⇐ $C7

AL ⇐ $04

CX:DX⇐ numer transakcji (odczytany przez funkcję końca transakcji)

Wyjście:

$$AL = \begin{cases} \$00 - \text{sukces} \\ \$FF - \text{transakcja jeszcze nie zakończona} \end{cases}$$

CA-Clipper Tools

NNETTTSSTA()

Funkcja podaje status zakończenia transakcji.

Funkcje pakietu CA-Clipper Tools zwracają wartości logiczne za wyjątkiem NNETTTSEND(), która podaje numer transakcji lub wartość -1 w przypadku błędu.

Przykład 1. Szkielet typowej transakcji w języku CA-Clipper 5.2c z wykorzystaniem CA-Clipper Tools

```
BEGIN SEQUENCE
   NNETTTSBEG()
   *
   * przetwarzanie danych
   *
   IF DISKSPACE() < 50000
      * przykładowa sytuacja uniemożliwiająca dokończenie transakcji
      BREAK  && przeniesienie sterowania do frazy RECOVER
   ENDIF
   *
   * przetwarzanie danych
   *
   NrTrans := NNETTTSEND()
   * testowanie pełnego zakończenia transakcji
   IF NrTrans != -1
      WHILE .NOT. NNETTTSSTA( NrTrans )
      ENDDO
   ENDIF
RECOVER
   NNETTTSAB()
END
```

Po wywołaniu funkcji końca transakcji system utrwala na dysku wszystkie dokonane zmiany. Ze względu na użycie buforów utrwalanie wyników transakcji może zakończyć się z pewnym opóźnieniem w stosunku do wywołania samej funkcji. Awaria serwera plików w tym czasie spowodowałaby likwidację skutków całej transakcji. Wskazane jest wobec tego użycie funkcji podającej status zakończenia transakcji w celu upewnienia się, czy transakcja została zakończona.

Mechanizmy systemu TTS znajdują zastosowanie nie tylko w przypadku wystąpienia awarii - możliwość programowego wycofania (TTS Abort Transaction) transakcji otwartej

za pomocą funkcji TTS Begin Transaction przydatna jest na przykład w sytuacji, w której wystąpiła wzajemna blokada danych. Działanie TTS Abort Transaction obejmie te dane, które zostały zmodyfikowane od chwili rozpoczęcia transakcji.

W przypadku niezakończenia transakcji jawnej nawet normalne zakończenie programu spowoduje wycofanie takiej transakcji. System zachowa się analogicznie, jeśli przyczyną zakończenia aplikacji będzie błąd programowy, który wystąpił przed wywołaniem funkcji TTS End Transaction.

Od chwili rozpoczęcia transakcji jawnej każdy zapis do pliku z atrybutem transakcyjności powoduje automatyczne założenie fizycznej blokady na odpowiednie obszary pliku. Wszystkie występujące w aplikacji zwolnienia blokad tych obszarów lub zamknięcia plików są odraczane aż do zamknięcia lub wycofania transakcji. Jeśli więc kilka stacji roboczych korzysta równocześnie z tego samego pliku danych i aplikacja na jednej z tych stacji rozpoczęła transakcję jawną, to odczyt rekordu z pliku w aplikacjach wykonywanych na pozostałych stacjach powinien zostać poprzedzony próbą zablokowania rekordu, aby uniknąć wystąpienia sytuacji błędnej.

Przykład 2. Implementacja funkcji systemu TTS z wykorzystaniem przerwania int $21

```
PUBLIC ExtAtt
PUBLIC DisTTS
PUBLIC EnTTS
PUBLIC IsTTS
PUBLIC TTSBeg
PUBLIC TTSAb
PUBLIC TTSEnd

EXTRN __ret  : FAR
EXTRN __retni : FAR
EXTRN __parni : FAR
EXTRN __parc  : FAR

DGROUP GROUP DataSg

DataSg SEGMENT PUBLIC 'DATA'

Buf1    DW 0001        ; bufor żądania dla funkcji "Disable TTS"
        DB 0CFh
Buf2    DW 0001        ; bufor żądania dla funkcji "Enable TTS"
        DB 0D0h
Buf3    DW 0000        ; wspólny bufor odpowiedzi

DataSg ENDS
```

```
CodeSg SEGMENT 'CODE'
       ASSUME CS:CodeSg, DS:DGROUP, ES:DGROUP
;-------------------------------------------------------------------------
;  ExtAtt( nazwa pliku, operacja ) - "Extended File Attributes"
;       operacja: 0 - zeruj atrybut transakcyjności, 1 - ustaw atrybut
;  funkcja zwraca 0 w przypadku poprawnego wykonania operacji
;-------------------------------------------------------------------------
ExtAtt PROC FAR
       PUSH BP              ; zachowanie zawartości rejestrów
       MOV BP, SP
       PUSH DS
       PUSH ES
       PUSH SI
       PUSH DI

       MOV AX, 1            ; parametr 1. - nazwa pliku
       PUSH AX
       CALL __parc          ; pobranie parametru (DX:AX - wskazanie nazwy pliku)
       ADD SP, 2            ; odtworzenie wskaźnika stosu
       PUSH DX              ; przechowanie wskazania nazwy pliku
       PUSH AX

       MOV AX, 2            ; parametr 2. - rodzaj operacji
       PUSH AX
       CALL __parni         ; pobranie parametru (AX - operacja)
       ADD SP, 2
       MOV BX, AX
       POP DX
       POP DS               ; DS:DX - wskazanie nazwy pliku

       MOV AH, 0B6h
       MOV AL, 00h
       INT 21h              ; pobranie rozszerzonych atrybutów pliku
       JC Koniec

       CMP BX, 1
       JE Ustaw
       AND CL, 11101111b    ; wyzerowanie bitu 4. reprezentującego atrybut transakcyjności
       JMP Zapisz
Ustaw: OR CL, 00010000b     ; ustawienie bitu 4. reprezentującego atrybut transakcyjności

Zapisz: MOV AH, 0B6h
       MOV AL, 01h
       INT 21h              ; zapis rozszerzonych atrybutów pliku
       JC Koniec
       MOV AX, 0            ; AX = 0 - funkcja wykonana poprawnie

Koniec: POP DI              ; odtworzenie zawartości rejestrów
       POP SI
       POP ES
       POP DS
       POP BP
```

```
            PUSH AX             ; AX - wartość zwracana przez funkcję ExtAtt
            CALL __retni
            ADD SP, 2
            RET
    ExtAtt  ENDP
;-------------------------------------------------------------------------------
; DisTTS() - "Disable Transaction Tracking"
; wartości zwracane przez funkcję:
;         0 - sukces
;       198 - użytkownik nie posiada praw operatora konsoli
;-------------------------------------------------------------------------------
    DisTTS  PROC FAR
            PUSH BP
            MOV BP, SP
            PUSH DS
            PUSH ES
            PUSH SI
            PUSH DI

            MOV AH, 0E3h
            LEA SI, Buf1
            LEA DI, Buf3
            INT 21h             ; próba zablokowania TTS

            POP DI
            POP SI
            POP ES
            POP DS
            POP BP

            MOV AH, 00h         ; zerowanie nieznaczącej wartości
            PUSH AX
            CALL __retni
            ADD SP, 2
            RET
    DisTTS  ENDP
;-------------------------------------------------------------------------------
; EnTTS() - "Enable Transaction Tracking"
; wartości zwracane przez funkcję:
;         0 - sukces
;       198 - użytkownik nie posiada praw operatora konsoli
;-------------------------------------------------------------------------------
    EnTTS   PROC FAR
            PUSH BP
            MOV BP, SP
            PUSH DS
            PUSH ES
            PUSH SI
            PUSH DI

            MOV AH, 0E3h
            LEA SI, Buf2
            LEA DI, Buf3
            INT 21h             ; próba uaktywnienia TTS
```

```
        POP DI
        POP SI
        POP ES
        POP DS
        POP BP

        MOV AH, 00h
        PUSH AX
        CALL __retni
        ADD SP, 2
        RET
EnTTS   ENDP
;-------------------------------------------------------------------------------
; IsTTS() - "TTS Is Available"
; wartości zwracane przez funkcję:
;       0 - niedostępny TTS
;       1 - dostępny TTS
;     253 - zablokowany TTS
;-------------------------------------------------------------------------------
IsTTS   PROC FAR
        PUSH BP
        MOV BP, SP
        PUSH DS
        PUSH ES
        PUSH SI
        PUSH DI

        MOV AH, 0C7h
        MOV AL, 02h
        INT 21h                 ; zbadanie dostępności i aktywności TTS

        POP DI
        POP SI
        POP ES
        POP DS
        POP BP

        MOV AH, 00h
        PUSH AX
        CALL __retni
        ADD SP, 2
        RET
IsTTS   ENDP
;-------------------------------------------------------------------------------
; TTSBeg() - "TTS Begin Transaction"
; wartości zwracane przez funkcję:
;       0     - sukces
;      -1     - kontynuacja aktywnej transakcji
;      -2     - początek transakcji jawnej
;    -106     - brak pamięci
;-------------------------------------------------------------------------------
TTSBeg  PROC FAR
        PUSH BP
        MOV BP, SP
        PUSH DS
```

```
            PUSH ES
            PUSH SI
            PUSH DI

            MOV AH, 0C7h
            MOV AL, 00h
            INT 21h             ; próba rozpoczęcia nowej transakcji
            JNC PrznB
            MOV AH, 0FFh        ; AH = $FF dla oznaczenia, że wystąpiło przeniesienie
            JMP OminB
    PrznB:  MOV AH, 00h

    OminB:  POP DI
            POP SI
            POP ES
            POP DS
            POP BP

            PUSH AX
            CALL __retni
            ADD SP, 2
            RET
    TTSBeg  ENDP
;-------------------------------------------------------------------------------
; TTSAb() - "TTS Abort Transaction"
; wartości zwracane przez funkcję:
;       0   - sukces
;       -1  - nie ma jawnej transakcji
;       -2  - funkcja wykonana, rekord(y) wciąż zablokowane
;       -3  - zablokowany TTS
;-------------------------------------------------------------------------------
    TTSAb   PROC FAR
            PUSH BP
            MOV BP, SP
            PUSH DS
            PUSH ES
            PUSH SI
            PUSH DI

            MOV AH, 0C7h
            MOV AL, 03h
            INT 21h             ; próba przerwania bieżącej transakcji
            JNC PrznA
            MOV AH, 0FFh
            JMP OminA
    PrznA:  MOV AH, 00h

    OminA:  POP DI
            POP SI
            POP ES
            POP DS
            POP BP
```

```
            PUSH AX
            CALL __retni
            ADD SP, 2
            RET
    TTSAb   ENDP
;-------------------------------------------------------------------------
; TTSEnd() - "TTS End Transaction"
; wartości zwracane przez funkcję:
;         0 - sukces
;       253 - zablokowany TTS
;       254 - funkcja wykonana, rekord(y) wciąż zablokowane
;        -1 - nie ma jawnej transakcji
;-------------------------------------------------------------------------
    TTSEnd  PROC FAR
            PUSH BP
            MOV BP, SP
            PUSH DS
            PUSH ES
            PUSH SI
            PUSH DI

            MOV AH, 0C7h
            MOV AL, 01h
            INT 21h             ; próba zakończenia bieżącej transakcji
            JNC PrznE
            MOV AH, 0FFh
            JMP OminE

    PrznE:  MOV AH, 00h
            CMP AL, 0
            JNE Omin_E

; pomyślne wykonanie funkcji TTS End Transaction, testowanie pełnego zakończenia transakcji
            MOV AH, 0C7h
    Petla:  MOV AL, 04h
            INT 21h             ; odczyt statusu transakcji - "TTS Transaction Status"
            CMP AL, 0
            JNE Petla
            MOV AX, 0           ; AX = 0 - pomyślne pełne zakończenie transakcji

    OminE:  POP DI
            POP SI
            POP ES
            POP DS
            POP BP

            PUSH AX
            CALL __retni
            ADD SP, 2
            RET
    TTSEnd  ENDP

    CodeSg  ENDS
            END
```

W celu wykorzystania tak zaimplementowanych funkcji systemu TTS w programach pisanych w języku Clipper należy:

- skompilować kod źródłowy funkcji TTS (np. programem TASM),
- uzyskany plik .OBJ dołączyć do listy innych plików tego typu, umieszczonej w wierszu wywołania linkera (np. programu RTLINK) dla wygenerowania kodu wynikowego programu w języku Clipper.

Przykład 3. Szkielet typowej transakcji w języku CA-Clipper 5.2c z wykorzystaniem funkcji zaimplementowanych jak w przykładzie 2.

```
BEGIN SEQUENCE
   TTSBeg()
   *
   * przetwarzanie danych
   *
   IF DISKSPACE() < 50000
      * przykładowa sytuacja uniemożliwiająca dokończenie transakcji
      BREAK  && przeniesienie sterowania do frazy RECOVER
   ENDIF
   *
   * przetwarzanie danych
   *
   TTSEnd()  && pełne zakończenie transakcji
RECOVER
   TTSAb()
END
```

5. Podsumowanie

Uwzględniając dominację sieciowego systemu operacyjnego NetWare autorzy zwracają uwagę na istnienie mechanizmów pozwalających na obsługę transakcji w tym systemie i sugerują ich wykorzystanie w aplikacjach operujących na relacyjnej bazie danych, rozważając pod tym kątem możliwości języka Clipper. Omawiają również blokady jako środek synchronizacji dostępu do zasobów przy współbieżnie realizowanym zbiorze transakcji. Ten sposób przetwarzania w porównaniu z realizacją szeregową tego samego zbioru charakteryzuje się lepszą efektywnością. Ponadto zastosowanie blokad pozwala na uzyskanie wyników identycznych z rezultatami przetwarzania szeregowego.

Warto również dodać, że specyfika definicji transakcji w danym Systemie Zarządzania Relacyjną Bazą Danych może pozwolić na rozszerzenie funkcjonalności zestawu operacji dotyczących transakcji przez wprowadzenie pojęcia podtransakcji lub też umożliwienie wycofania transakcji do określonego punktu.

LITERATURA

[1] Praca zbiorowa pod redakcją Kozielskiego S.: Bazy danych w lokalnych sieciach komputerowych oraz systemach wielozadaniowych. Skrypt Politechniki Śląskiej nr 1869, Gliwice 1994.

[2] Wolisz A.: Podstawy lokalnych sieci komputerowych. WNT, Warszawa 1992.

[3] Spence R.: Clipper 5.2 przewodnik programisty. PLJ, Warszawa 1992.

[4] NOVELL NetWare Version 3.11 Concepts (dokumentacja firmowa).

[5] Dokumentacja języka Clipper Summer'87 firmy Nantucket.

[6] CA-Clipper Tools. Reference Guide. Version 3.0.

Recenzent: Dr inż. Maciej H. Bargielski

Wpłynęło do Redakcji 28 czerwca 1995 r.

Abstract

On account of the NetWare operation system domination the authors discuss the transaction management in the NetWare 3.11 System and present a set of functions offered by the system for a transaction tracking in the applications working with relational database. The discussion of the possibilities of the usage of these functions in programms written in the Clipper language is included in this paper. The problem of protection against data loss and data inconsistence in the applications which don't use the transaction tracking was considered. The locks as a tool for the synchronization of the access to the resources from the parallel executed transactions are also presented in this paper.