ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 199?

Seria: INFORMATYKA z. 25 Nr kol. 1236

Dariusz AUGUSTYN

MECHANIZMY WIELODOSTĘPU W SYSTEMACH FoxBASE+ 2.0 I FoxPro/LAN 2.0

Streszczenie. W artykule zaprezentowane zostały mechanizmy wielodostępu, zaimplementowane w systemach FoxBASE+ i FoxPro 2.0/LAN. Wyjaśnione zostały pojęcia trybu dostępu, zakresu blokad, funkcji blokujących, mechanizmu automatycznego blokowania.

THE MULTI-USER DATABASE MANAGEMENT SYSTEM FoxBASE+ 2.0 AND FoxPro/LAN 2.0

Summary. The multi-user environments of database management systems FoxBASE+ 2.0 and FoxPro/LAN 2.0 were presented in this article. The article explained problems which are related to data access modes, manual locking data functions, automatic data locking, scopes of data locking.

DIE MECHANISMEN DES MEHRBENUTZERZUGRIFFS IN DEN SYSTEMEN FoxBASE+ UND FoxPro 2.0/LAN

Zusammenfassung. Im Artikel wurden die in den Systemen FoxBASE+ und FoxPro 2.0/LAN implementierten Mechanismen des Mehrbenutzerzugriffs dargestellt. Die folgenden Begriffe: der Zugriffsmodus, der Sperrenbereich, die Sperrfunktionen, der Mechanismus der automatischen Sperre wurden erläutert.

1. Wstęp

W związku z rozwojem sieciowych systemów komputerowych współczesne zastosowania nakładają na systemy zarządzania bazami danych oczywistą konieczność obsługi wielodostępu. Wielodostępne bazy danych posiadają mechanizmy pozwalające na bezpieczne zrównoleglenie przetwarzania danych przy zachowaniu spójności danych.

Właściwe wykorzystanie wspomnianych mechanizmów przez programistę umożliwia napisanie "bezpiecznego" programu, tzn. poprawnie działającego w wielodostępnym środowisku.

Celowość wprowadzenia mechanizmów kontroli dostępu do danych można zilustrować przykładami programów, których jednoczesne wykonanie w środowisku sieciowym może doprowadzić do błędnych rezultatów.

Poniższy przykład może posłużyć do demonstracji zdarzenia utraty efektu modyfikacji danych w wyniku współbieżnej realizacji tego samego programu w różnych węzłach sieci.

```
program:
 begin
   read (a,A); A := A - X; write (a,A);
   read (b,B); B := B + X; write (b,B); display (A+B)
 end
```

gdzie:

a,b - jednostki danych tzn. rekord (inaczej wiersz, krotka),
A,B,X - zmienne dla operacji wykonanej na jednostce danych,
T1,T2 - procesy, zadania - transakcje przelewu (wykonania tego samego programu)

T1	T2
read (a,A)	
A := A - X	
	read (a,A)
write (a,A)	
	A := A - X
read (b,B)	
	write (a,A)
B := B + X	
write (b,B)	
	read (b,B)
display(A+B)	
	B = B + X
	write (b,B)
	display(A+B)

W przykładzie nastąpiła utrata zmiany wartości jednostki danych *a*, realizowanej przez proces *T1*. Widoczna jest potrzeba wykorzystania mechanizmu typu blokowanie dostępu, który uniemożliwiłby dostęp do jednostki danych przez inne procesy w czasie operacji dokonywanej na niej.

Przykład programu pozwalającego na synchronizację realizacji procesów poprzez zastosowanie blokowania dostępu do danych.

```
program:
  begin
    lock(a); read (a,A); A := A - X; write (a,A); unlock(a);
    lock(b); read (b,B); B := B + X; write (b,B); display(A+B); unlock(b)
  end
```

lock (x) - rozkaz blokady na wyłączność jednostki danych *x*.
W sytuacji kolizji dostępu rozkaz lock wstrzymuje realizację procesu aż do czasu zwolnienia danych.

unlock(x) - rozkaz zwolnienia blokady jednostki danych *x*.

```
        T1                  |        T2
                            |
lock (a)                    |
                            |   lock (a)
read (a,A)                  |
                            |
A := A - X                  |
                            |
write (a,A)                 |
                            |
unlock (a)                  |
                            |     read (a,A)
lock (b,B)                  |     A := A - X
                            |     write (a,A)
read (b,B)                  |     unlock (a)
                            |     lock (b)
B := B + X                  |
                            |
write (b,B)                 |
                            |
display(A+B)                |
                            |
unlock (b)                  |
                            |     read (b,B)
                            |     B := B + X
                            |     write (b,B)
                            |     display(A+B)
                            |     unlock(b)
```

Poprzedni przykład programu, w którym zastosowano regułę blokowania zasobu w czasie wykonywania operacji, stanowi ilustrację wpływu blokowania na kolejność realizacji operacji. Pomimo zastosowania blokad program nadal może nie działać poprawnie w odniesieniu do jednej z możliwych funkcji, jaką ma realizować, tzn. wyświetlania sumarycznego stanu kont a i b. Dla różnej szybkości realizacji procesów *T1*, *T2* (wynikającej na przykład z różnych szybkości komputerów w węzłach sieci) możliwe jest pokazanie pozornej utraty spójności. Spójność danych dla omawianego przykładu realizującego tzw. transakcję przelewu wyraża się prawdziwością formuły $a + b = \text{const}$.

```
          T1                    T2
lock (a)
                          lock (a)
read (a,A)
A := A - X
write (a,A)
unlock (a)
                          read (a,A)
                          A := A - X
                          write (a,A)
                          unlock (a)
                          lock (b)
lock (b)                  read (b,B)
                          B := B + X
                          write (b,B)
                          display(A+B)
                          unlock (b)
read (b,B)
B := B + X
write (b,B)
display(A+B)
unlock (b)
```

W celu zapewnienia poprawnego działania należałoby zablokować obydwie jednostki danych przed wykonaniem wszystkich pozostałych operacji.

```
program:
  begin
    lock(a); lock(b) ;
    read (a,A); A := A - X; write (a,A);
    read (b,B); B := B + X; write (b,B);
    display(A+B); unlock(a); unlock(b)
  end
```

Zadanie właściwego użycia mechanizmów kontroli wielodostępu należy do programisty. Programista, mając na uwadze warunek spójności bazy danych, powinien zastosować taki system rozwiązywania kolizji, który umożliwi poprawną pracę programu w środowisku sieciowym i zapewni maksymalną możliwość równoległego przetwarzania danych w innych węzłach sieci.

Systemy dBaseIII+ i FoxBASE+ wyposażone są w proste mechanizmy kontroli dostępu, umożliwiające tworzenie programów pracujacych poprawnie w środowisku wielodostępnym. Bardzo podobne mechanizmy zastosowane zostały w języku CLIPPER. System FoxPro/LAN zawiera pewne rozszerzenie rozwiązań wielodostępu w stosunku do FoxBASE+. Przedmiotem rozważań zawartych w artykule jest prezentacja zagadnienia wielodostępu głównie w kontekście możliwości systemów FoxBASE+ i FoxPro/LAN.

2. Mechanizmy kontroli dostępu w systemie FoxBASE+ 2.0

Dotychczasowe zainteresowanie FoxBASE-m wynikało m.in. z faktu zgodności z dBaseIII+ i posiadania implementacji pracujących pod systemami operacyjnymi DOS i SCO Xenix, SCO Unix. Cecha ta pozwala na łatwe przenoszenie aplikacji i plików baz danych pomiędzy różnymi środowiskami sieciowymi.

W typowych lokalnych sieciach komputerowych, w których końcówką jest komputer z systemem operacyjnym DOS, programy napisane z użyciem FoxBASE przetwarzają dane na końcówkach - węzłach sieci, a serwer sieci pełni jedynie rolę serwera plików. Takie działanie, polegające na transmisji całych plików danych, powoduje zwiększenie liczby żądań i czasu trwania transmisji w całej sieci. Efektywność przetwarzania danych jest mocno uzależniona od szybkości transmisji i oczywiście szybkości samej końcówki.

Mechanizmy kontroli dostępu zastosowane w systemie FoxBASE+ są uproszczone, co wynika m.in. również z utrzymania zgodności z dBaseIII+. dBaseIII+ nie jest systemem projektowanym od podstaw z uwzględnieniem wielodostępu.

Sterowanie wielodostępem w systemie FoxBASE+ polega na:
- sterowaniu trybem dostępu do pliku bazy danych (tryb otwarcia pliku),
- blokowaniu dostępu do rekordu lub pliku bazy danych.

2.1. Tryby dostępu

W systemie FoxBASE+ wprowadzono dwa tryby dostępu do plików:
- wyłączny (exclusive),
- dzielony (shared).

Próba otwarcia pliku bazy danych (wraz z ewentualnymi plikami indeksowymi i notatnikowymi) w trybie wyłącznym może być zrealizowana tylko w sytuacji, gdy plik nie jest otwarty w jakimkolwiek trybie przez inne zadania. Otwarcie pliku w trybie wyłącznym oznacza, że

zadanie może wykonywać wszystkie dozwolone przez język FoxBASE+ operacje na tym pliku. W tym czasie wszystkie próby otwarcia pliku w jakimkolwiek trybie przez inne zadania kończą się niepowodzeniem.

Otwarcie pliku w trybie dzielonym może być zrealizowane, jeśli plik nie jest otwarty w trybie wyłącznym w innym węźle sieci. Poprawne wykonanie operacji na pliku bazy danych uzależnione jest od rodzaju operacji i zastosowanego mechanizmu blokowania.

Istnieje grupa instrukcji (modyfikująca obraz plików) takich, jak: MODIFY STRUCTURE, PACK, REINDEX, ZAP, INSERT, INSERT BLANK, których poprawne działanie wymaga otwarcia pliku bazy danych w trybie wyłącznym.

Otwarcie pliku bazy danych poprzez wywołanie instrukcji USE powoduje przyjęcie domyślnego trybu dostępu. Bezpośrednio po zainicjowaniu systemu jest nim tryb wyłączny (chyba, że wystąpiła komenda SET EXCLUSIVE OFF w zbiorze konfiguracyjnym). Tryb wyłączny można również ustawić wydając komendy:

```
SET EXCLUSIVE ON
USE <nazwa relacji>
lub
USE <nazwa relacji> EXCLUSIVE
```

Zmianę aktualnego trybu na dzielony tryb otwarcia pliku bazy danych dla później wykonywanych operacji można uzyskać poprzez wykonanie komendy:

```
SET EXCLUSIVE OFF
```

Próba otwarcia pliku, gdy jest on już otwarty na wyłączność w innym węźle sieci, powoduje przekazanie sterowania do procedury obsługi błędu (systemowej lub własnej, utworzonej przez programistę) z ustawieniem kodu błędu - 108. Błędowi o kodzie 108 odpowiada systemowy komunikat "File is in use by another". Kod błędu, odpowiedni komunikat systemowy, nazwa programu, w którym wystąpił błąd, osiągalne są poprzez wywołania funkcji ERROR(), MESSAGE(), SYS(16). Nazwę procedury obsługi błędu oraz parametry wywołania określa instrukcja ON ERROR.

Przykład prostej obsługi programowej błędu otwarcia pliku bazy danych:

```
* Program TEST1.PRG:
SET PROCEDURE TO OBS_BLED
ON ERROR DO OBS_BLED WITH ERROR(),MESSAGE(),SYS(16)
!
USE placa EXCLUSIVE
!
RETURN

* Procedura obslugi bledow OBS_BLED.PRG:
PARAMETERS nr_bledu, komunikat, program
!
IF nr_bledu = 108
  * 108 : Plik jest w użyciu przez innego uzytkownika
  IF program == "U:TEST1.PRG"
    @ 9,5  SAY " Komunikat systemowy : " + komunikat
    @ 10,5 SAY " Blokowany dostep do pliku. "
    @ 11,5 SAY " Ponowienie proby dostepu za chwile."
    licznik = 0
    DO WHILE licznik < 5000
      licznik = licznik + 1
    ENDDO
    @ 9,5  CLEAR
    @ 10,5 CLEAR
    @ 11,5 CLEAR
    RETRY
  ENDIF
ENDIF
* Programowa obsługa innych bledow.
!
```

Instrukcja RETRY w procedurze OBS_BLED.PRG powoduje ponowne wznowienie wykonywania instrukcji, podczas realizacji której ostatnio wystąpił błąd. Użycie RETURN spowodowałoby wznowienie wykonania programu TEST1.PRG od instrukcji następnej po rozkazie generującym błąd.

2.2. Blokowanie dostępu

Mechanizmy blokowania mogą być sensownie użyte dla plików bazy danych otwieranych w trybie dzielonym. Sprawdzenie trybu dostępu i stanu blokowania dla każdego pliku

otwartego w danym obszarze roboczym umożliwia instrukcja DISPLAY STATUS. Blokowane mogą być całe pliki albo pojedyncze rekordy. Obowiązuje zasada tzw. ziarnistości blokowania:

- plik nie może być blokowany, jeżeli co najmniej jeden jego rekord lub cały plik jest już blokowany przez inne zadanie,
- rekord nie może być blokowany, jeżeli inne zadanie blokuje ten rekord lub cały plik.

2.3. Otwarcie pliku w trybie wyłącznym

Otwarcie pliku w trybie wyłącznym stanowi "blokadę" wszystkich ewentualnych operacji na pliku w innych węzłach sieci (w tym operacji odczytu danych), ze względu na brak możliwości otwarcia pliku w tych węzłach sieci.

2.4. Jawne blokowanie

Jawne blokowanie pojedynczego rekordu lub pliku przeprowadza się za pomocą wywołań funkcji realizujących blokadę jednostki danych i zwracających wartość logiczną określającą powodzenie próby. W danym obszarze roboczym może obowiązywać blokada całego pliku bądź jednego rekordu. Nałożona blokada obowiązuje aż do jej jawnego odwołania.

Instrukcje realizujące odwołanie blokady dostępu do danych:

UNLOCK - instrukcja odblokowania danych (rekordu lub pliku) w aktualnym obszarze roboczym. Instrukcja USE, wywołana bez parametrów, powoduje miedzy innymi odblokowanie otwartego pliku w aktualnym obszarze roboczym.

UNLOCK ALL - instrukcja odblokowania wszystkich rekordów i plików aktualnego zadania. Instrukcje takie jak QUIT, CLOSE DATABASES, CLOSE ALL między innymi również powodują odblokowanie wszystkich danych.

Dodatkowym, niejawnym sposobem anulowania blokady jest próba nałożenia innej blokady w tym samym obszarze roboczym. Przykładem może być próba zablokowania całego pliku przy utrzymanej blokadzie jednego z rekordów tego pliku albo próba zablokowania innego rekordu w danym pliku. Niestety, system FoxBASE działa według schematu: najpierw odblokowanie jednostki danych, potem próba (niekoniecznie pomyślna) blokady nowej jednostki danych.

2.5. Jawne blokowanie pliku

Jawne blokowanie pliku w trybie dzielonym przeprowadza się poprzez wywołanie funkcji FLOCK() dla pliku w aktualnym obszarze roboczym. Na ogół blokada pliku stosowana jest przed realizacją operacji dotyczącej modyfikacji całości pliku lub grupy rekordów.

FLOCK() - funkcja wykonująca próbę zablokowania pliku i zwracająca wartość logiczną, określającą rezultat tej próby.

Przykład użycia funkcji blokującej przy pracy w interaktywnym środowisku FoxBASE:

```
.USE placa
.? FLOCK()
.T.
.BROWSE
.UNLOCK
```

Przykład operacji modyfikacji wszystkich rekordów pliku:

```
SET EXCLUSIVE OFF
USE placa
IF FLOCK()
  REPLACE ALL kwota WITH kwota + 1000
  UNLOCK
ELSE
  ? " Dostep do pliku zablokowany ! "
ENDIF
```

2.6. Jawne blokowanie rekordu

Jawne blokowanie rekordu w trybie dzielonym przeprowadza się poprzez wywołanie funkcji RLOCK() dla pliku w aktualnym obszarze roboczym. Ewentualna blokada zostaje nałożona na bieżący rekord w aktualnym obszarze roboczym.

RLOCK(),LOCK() - funkcje o identycznym działaniu, wykonujące próbę zablokowania rekordu i zwracające wartość logiczną, określającą rezultat tej próby.

Przykład 1:

```
* dzielony tryb dostepu
USE placa
LOCATE FOR prac_numer = 10
IF RLOCK()
  @ 10,5 GET kwota
  READ
ELSE
  @ 20,5 SAY " Rekord niedostepny ! "
ENDIF
```

Przykład 2:

```
SET ESCAPE ON
SET EXCLUSIVE OFF
USE placa
LOCATE FOR prac_numer = 10
* Przerwanie przez Esc
DO WHILE .NOT. RLOCK()
ENDDO
REPLACE kwota WITH kwota + 10000
UNLOCK
```

System blokad w FoxBASE+ przeznaczony jest głównie dla operacji modyfikacji danych. Operacja odczytu wykonana zostanie nawet w przypadku nałożonej blokady w innym węźle. Chcąc upewnić się, że odczytywane dane nie są aktualnie modyfikowane gdzie indziej, można posłużyć się funkcjami RLOCK() czy FLOCK().

2.7. Automatyczne blokowanie

W systemie FoxBASE, dla instrukcji modyfikujących bazę danych, wprowadzono dodatkowy mechanizm kontroli dostępu, tzw. automatyzm blokowania jednostek danych. Automatyzm blokowania, w dzielonym trybie otwarcia pliku bazy danych, polega na niejawnym blokowaniu jednostki danych przez operację. Zakres blokady wynika z zakresu działania realizowanej operacji. Przed wykonaniem właściwej operacji następuje próba nałożenia odpowiedniej blokady. Po wykonaniu operacji następuje anulowanie założonej blokady (tzn. nie ma potrzeby jawnego odwoływania blokady przez instrukcje typu UNLOCK). W przypadku nieudanej próby nałożenia blokady następuje przejście do procedury obsługi błędów (z możliwością uzyskania właściwego kodu błędu).

Korzystanie z automatycznego blokowania, w przypadku prostych programów, pozwala na rozłączne umieszczanie treści programu i kodu związanego z kontrolą dostępu.

2.8. Automatyczne blokowanie pliku

Do grupy instrukcji dokonujących automatycznej próby blokowania pliku należą między innymi: APPEND, APPEND FROM, BROWSE, INDEX, JOIN
oraz REPLACE, RECALL , DELETE z niepustą specyfikacją zakresu,
różną od NEXT 1,RECORD <numer rekordu>.

W przypadku kolizji generowany jest błąd o numerze 108 ze standardowym komunikatem "File is in use by another".

W poniższych przykładach błąd kolizji dostępu obsługiwany jest przez odpowiednią procedurę. W obydwu przykładach, w przypadku niewystąpienia kolizji przy realizacji instrukcji REPLACE ALL, blokada utrzymywana jest tylko na czas realizacji tej instrukcji. Jeżeli kolizja wystąpiła i blokada została nałożona później, w wyniku realizacji procedury obsługi błędu, to zakres jej obowiązywania w obu przykładach będzie różny. W przykładzie drugim instrukcja UNLOCK jest wywołana, by anulować blokadę, którą mogła założyć funkcja RLOCK() w sytuacji po kolizji dostępu.

Przykład 1:

```
* Tekst programu TEST.PRG :
SET EXCLUSIVE OFF
ON ERROR DO BLEDY WITH ERROR(),MESSAGE()
*
USE placa
REPLACE ALL kwota WITH kwota + 20000
*
RETURN


* Program BLEDY.PRG:
PARAMETERS nr_bledu,komunikat
*
IF nr_bledu = 108
  * 108 : Plik w uzyciu przez innego uzytkownika
  @15,5 SAY " Komunikat systemowy: " +komunikat
  @16,5 SAY " Nieudana proba automatycznej blokady pliku."
  odp = 'T'
  @20,5 SAY " Ponowienie proby ? (T/N) " GET odp
  READ
  CLEAR
  IF UPPER ( odp ) == 'T'
    RETRY
  ELSE
    * Zakonczenie programu
```

```
   CANCEL
  ENDIF
ENDIF
```

Przykład 2:

```
* Tekst programu PODWYZKA.PRG :
SET EXCLUSIVE OFF
SET PROCEDURE TO PODWYZKA
ON ERROR DO OBSLUGA WITH ERROR(),MESSAGE()
 :
USE placa
REPLACE ALL kwota WITH kwota + 200000
UNLOCK
 :
SET PROCEDURE TO
ON ERROR
RETURN

PROCEDURE OBSLUGA
PARAMETERS nr_bledu,komunikat
 :
IF nr_bledu = 108
  * 108 : Plik w uzyciu przez innego uzytkownika
  @ 20,5 SAY komunikat
  ilosc = 0
  DO WHILE .NOT.FLOCK() .AND. ilosc < 1000
    ilosc = ilosc + 1
  ENDDO
  IF FLOCK()
    @ 20,5 CLEAR
    RETRY
  ELSE
    @ 20,5 SAY " Nieudana proba blokowania !"
    CANCEL
  ENDIF
ENDIF
```

2.9. Automatyczne blokowanie rekordu

Do grupy instrukcji dokonujących automatycznej próby blokowania rekordu należą między innymi:
DELETE, DELETE RECORD <numer rekordu>, DELETE NEXT 1,
RECALL, RECALL RECORD <numer rekordu>, RECALL NEXT 1,
REPLACE NEXT 1, REPLACE <numer rekordu>. W przypadku kolizji generowany jest błąd o numerze 109 i standardowym komunikacie "Record is in use by another".

W poniższym przykładzie błąd kolizji dostępu obsługiwany jest przez odpowiednią procedurę.

Przykład:

```
* Tekst programu TEST2.PRG:
SET EXCLUSIVE OFF
ON ERROR DO OBS_BLD WITH ERROR(),MESSAGE()
 :
USE placa
LOCATE FOR prac_numer = 10
DELETE
 :
ON ERROR
RETURN

* Tekst podprogramu OBS_BLD.PRG :
PARAMETERS nr_bledu,komunikat
 :
IF nr_bledu = 109  * 109 : Rekord jest w uzyciu przez innego uzytkownika
  :
  @ 18, 5  SAY komunikat
  @ 19, 5  SAY "Proby uzyskania dostepu."
  ilosc = 0
  DO WHILE .NOT. RLOCK() .AND. ilosc < 5000
    ilosc = ilosc + 1
  ENDDO
  IF RLOCK()
    @ 18,5 CLEAR
    @ 19,5 CLEAR
    RETRY
  ELSE
    @ 10,5 SAY " Nieudana proba dostepu do rekordu."
    @ 19,5 CLEAR
  ENDIF
ENDIF
```

2.10. Instrukcje wymagające blokowania rekordu

Instrukcje operujące na rekordzie bieżącym takie, jak:
REPLACE <pole> WITH <wyrażenie>, @ .. GET <pole> READ
wymagają jawnego blokowania danych przed ich użyciem (poprzez użycie funkcji RLOCK, FLOCK lub poprzez otwarcie pliku w trybie wyłącznym). Użycie wspomnianych instrukcji (w stosunku do plików otwartych w trybie dzielonym), bez poprzedzenia ich wywołaniami funkcji blokujących, powoduje przekazanie sterowania do procedury obsługi błędu. Numer 130 jest kodem błędu odpowiadającym tej sytuacji. "Record is not locked" jest komunikatem systemowym odpowiadającym tej sytuacji.

W poniższym przykładzie procedura obsługi błędów jest zawsze wywoływana w trakcie realizacji instrukcji REPLACE, niezależnie od wystąpienia ewentualnych kolizji.

Przykład:

```
SET EXCLUSIVE OFF
ON ERROR DO BLAD
USE placa
:
LOCATE FOR prac_numer = 10
REPLACE kwota WITH kwota + 10000
:
ON ERROR
RETURN

* Tekst programu BLAD.PRG :
IF ERROR() = 130
  * 130 : Rekord nie jest blokowany
  @ 20,5 SAY MESSAGE()
  ilosc = 0
  DO WHILE .NOT. RLOCK() .AND. ilosc < 5000
    ilosc = ilosc + 1
  ENDDO
  IF RLOCK()
    @ 20,5 CLEAR
    RETRY
```

```
  ELSE
    @ 20,5 SAY " Nieudana proba blokady rekordu."
    CANCEL
  ENDIF
ELSE
  :
ENDIF
```

2.11. Interaktywne instrukcje modyfikacji wymagające jawnego blokowania

Instrukcje EDIT, CHANGE, realizujące operacje modyfikacji danych w trybie interaktywnym, wymagają blokowania rekordu przed jego modyfikacją. Wywołanie tych instrukcji powoduje odczytanie zawartości bieżącego rekordu, bez automatycznej próby nałożenia blokady. Takie działanie (bez blokowania przy wywołaniu) instrukcji interaktywnych wynika z idei uniknięcia długiego, przypadkowego, niepożądanego blokowania danych wyświetlanych na ekranie.

Modyfikacja wyświetlanych pól rekordu może być wykonana po blokowaniu tego rekordu, które może nastąpić w wyniku:

- wcześniejszego użycia funkcji RLOCK(), FLOCK(),
- naciśnięcia przez użytkownika klawiszy Ctrl-O.

2.12. Instrukcje odczytu danych, niezależne od systemu blokad

Instrukcje odczytu danych, takie jak: DISPLAY ALL, SUM, TOTAL, COUNT omijają system blokad, tzn. mogą działać niezależnie od ewentualnych modyfikacji, przeprowadzanych równocześnie w innych węzłach sieci. Ze względu na poprawność odczytywanych danych, celowe może okazać się blokowanie pliku przed wykonaniem tych instrukcji.

2.13. Instrukcje wymagające wyłącznego trybu otwarcia pliku

Instrukcje zmieniające obraz pliku bazy danych takie, jak: MODIFY STRUCTURE, PACK, REINDEX, ZAP, INSERT, INSERT BLANK wywołane w stosunku do pliku otwartego w trybie dzielonym generują błąd o numerze 110 i odpowiednim komunikatem systemowym "Exclusive open of file is required.".

3. Mechanizmy kontroli dostępu w systemie FoxPro/LAN 2.0

System FoxPro stanowi znaczne poszerzenie systemu FoxBASE. Do nowych cech systemu należą między innymi:

- bardzo wzbogacony w stosunku do systemu FoxBASE język podstawowy,
- dodatkowy, uproszczony język zapytań typu SQL,
- nowe (w stosunku do FoxBASE+), efektywne, oszczędne pod względem rozmiaru pliki bazy danych i pliki indeksowe,
- mechanizm efektywnego, domyślnego wykorzystywania indeksów w operacjach wyszukiwania danych (technika Rushmore),
- wbudowane mechanizmy zarządzania pamięcią,
- interaktywne generatory menu, formatek, raportów,
- program zarządzania elementami aplikacji z generatorem kodu wynikowego aplikacji,
- program wspomagania tworzenia dokumentacji,
- nowoczesny, wygodny interfejs, sterowany przy użyciu myszy.

Zainteresowanie FoxPro wynika również z faktu pojawienia się systemu w polskiej wersji językowej i zapowiedzi systemu FoxPro 2.5 for Windows, FoxPro for SCO Unix.

Większość uwag dotyczących zagadnienia wielodostępu w FoxBASE+ odnosi się również do FoxPro/LAN. Nowe możliwości obsługi wielodostępu wprowadzone do systemu FoxPro są tematem następnych podrozdziałów.

3.1. Możliwość blokowania grupy rekordów w danym obszarze roboczym

Domyślnie, FoxPro dla zachowania zgodności z FoxBASE-m, pozwala na zablokowanie tylko jednego rekordu (RLOCK) lub całego pliku bazy danych (FLOCK).

Instrukcja:

SET MULTILOCK ON | OFF

steruje przełączaniem trybu pracy systemu blokowania. W trybie MULTILOCK możliwe jest zablokowanie dowolnej grupy wybranych rekordów.

W trybie blokowania wielu rekordów kolejne wywołanie funkcji RLOCK() w ramach realizacji programu oprócz ewentualnej blokady założonej na rekord bieżący, nie powoduje anulowania blokady założonej wcześniej na inny rekord (tak jak w systemie FoxBASE). Numery aktualnie blokowanych rekordów można wyświetlić wykonując np. instrukcję DISPLAY STATUS.

Przy zmianie trybu blokowania rekordów przez SET MULTILOCK ON | OFF następuje anulowanie wszystkich blokad (jak UNLOCK ALL).

W trybie pracy blokady wielu rekordów obowiązuje rozszerzona składnia funkcji RLOCK (LOCK):

RLOCK ([<par1>,<par2>]),

RLOCK ([par2])

gdzie:

<par1> stanowi łańcuch znakowy, zawierający listę numerów blokowanych rekordów,

<par2> stanowi numer obszaru lub nazwę pliku w danym obszarze roboczym (tzw. alias).

Przykład:

```
.SET EXCLUSIVE OFF
.SET MULTILOCK ON
.USE placa
.? RLOCK ( "1,3,5" , "placa")
```

W przykładzie funkcja RLOCK zwraca wartość .T. w sytuacji, gdy uda się jednocześnie nałożyć blokady na wszystkie trzy rekordy. W sytuacji braku możliwości zablokowania któregoś z rekordów, blokada nie jest zakładana na żaden z pozostałych rekordów.

3.2. Sterowanie liczbą prób nałożenia blokady

Instrukcja:

SET REPROCESS TO <liczba> [SECONDS], gdzie <liczba> ∈ <-2,32000>,

SET REPROCESS TO AUTOMATIC

pozwala na ustalenie sposobu działania programu w sytuacji nieudanej próby blokowania rekordu lub pliku. Ustawienie SET REPROCESS TO dotyczy liczby kolejnych prób blokowania po nieudanym blokowaniu jawnym (RLOCK,FLOCK) albo automatycznym.

Dodanie frazy SECONDS oznacza, że <liczba> dotyczy czasu trwania ewentualnych, ponownych prób blokowania.

Jeżeli <liczba> jest większa od zera, to system przy nieudanej próbie założenia blokady ponawia <liczbę> razy próbę, po czym funkcja blokująca (RLOCK,FLOCK) zwróci wartość .F. albo po instrukcji automatycznie blokującej, sterowanie przejmie procedura obsługi błędów (systemowa albo definiowana przez użytkownika). Przy ustawionym SET STATUS ON podczas prób wyświetlany jest komunikat "Waiting for lock...".

Jeżeli <liczba> wynosi -1, to w sytuacji nieudanej próby blokowania system nieprzerywalnie powtarza kolejne próby blokady (ignorując klawisz Esc).

Jeżeli <liczba> wynosi 0 (domyślne ustawienie systemu) i nie zdefiniowano własnej procedury obsługi błędu, to w sytuacji nieudanej blokady system w sposób ciągły ponawia kolejne próby. Na ekranie wyświetlany jest komunikat "Attempting to lock... Press ESC to Cancel." W sytuacji, gdy próby blokady wynikały z wywołania instrukcji blokującej automatycznie, naciśnięcie Esc powoduje wyświetlenie systemowego komunikatu "File is in use by another." lub "Record is in use by another." i przerwanie wykonania programu. W sytuacji gdy próby blokady wynikały z wywołania funkcji blokującej typu FLOCK lub RLOCK, naciśnięcie Esc powoduje oddanie sterowania do programu wywołującego funkcję, która zwraca wtedy wartość .F..

Jeżeli <liczba> wynosi 0 i zdefiniowano własną procedurę obsługi błędu, to w sytuacji nieudanej blokady system nie ponawia automatycznie prób nałożenia blokady. Jeśli nieudana blokada wynikała z wywołania instrukcji automatycznie blokującej, sterowanie od razu oddawane jest do zdefinowanej procedury obsługi błędu. Jeśli nieudana blokada wynikała z wywołania FLOCK albo RLOCK, to funkcja ta od razu zwraca wartość .F..

Po ustawieniu SET REPROCESS TO AUTOMATIC (SET REPROCESS TO -2) system w sytuacji kolizji, zawsze przechodzi do ponownych prób blokowania, niezależnie od zdefinowania własnej procedury obsługi błędu. Po naciśnięciu przez użytkownika klawisza Esc, funkcja blokująca zwraca wartość .F. albo (w przypadku instrukcji blokującej automatycznie) sterowanie przechodzi do własnej lub systemowej procedury obsługi błędu.

3.3. Blokowanie danych w instrukcjach BROWSE, CHANGE, EDIT

W systemie FoxPro nowa, rozbudowana instrukcja BROWSE (ze względu na mnogość opcji) pozwala na napisanie programu realizującego złożoną obsługę ekranu, bardzo niskim nakładem czasu.

W systemie FoxBASE instrukcja BROWSE należała do grupy rozkazów automatycznie blokujących plik. W FoxPro rozkaz BROWSE może zablokować pojedynczy rekord. Bieżący rekord nie jest automatycznie blokowany bezpośrednio po wywołaniu BROWSE. Dopiero rozpoczęcie modyfikacji pola w danym wierszu lub naciśnięcie klawiszy Ctrl-O powoduje próbę nałożenia blokady na bieżący rekord.

Przy poruszaniu się kursorem po wierszach wyświetlanych przez BROWSE nie następują próby ich blokowania. (Wykorzystując frazy WHEN, ERROR można jednak tak uruchomić BROWSE, by po przejściu kursora następowała próba blokady nowego, bieżącego rekordu poprzez wywołanie RLOCK()).

Wyjście z BROWSE albo zmiana wiersza powoduje zdjęcie ewentualnej blokady założonej przez BROWSE na aktualny rekord.

Jeżeli system jest w trybie MULTILOCK i nastąpiło wcześniejsze jawne zablokowanie kilku rekordów, to realizacja BROWSE nie spowoduje anulowania tych blokad. Mechanizm ewentualnego blokowania i odblokowywania pojedynczego rekordu w BROWSE jest niezależny od wcześniej zrealizowanych, jawnych blokad. Jawne blokady rekordów mogą być anulowane tylko jawnie, przez wywołania instrukcji typu UNLOCK, po wyjściu z BROWSE.

Oszczędne blokowanie (ewentualnia blokada pojedynczego rekordu) z ewentualną opcją automatycznego wyjścia z BROWSE po odpowiednio długim czasie braku reakcji ze strony użytkownika (fraza TIMEOUT), pozwala na równoczesny dostęp do rekordów danego pliku w innych węzłach sieci.

3.4. Odświeżanie wyświetlanych danych w instrukcjach BROWSE, CHANGE, EDIT

Typowe wywołanie instrukcji BROWSE powoduje wyświetlenie grupy rekordów w formie wierszy pewnej tabeli. Przy domyślnym ustawieniu (SET REFRESH TO 0) rozkaz BROWSE nie odświeża zawartości tabelki w sytuacji, gdy w innym węźle sieci nastąpiła modyfikacja któregoś z wyświetlanych rekordów. Jeżeli użytkownik dokona zmiany aktualnego wiersza, tzn. nastąpi przesunięcie pionowe w tabeli, to obraz nowego, aktualnego wiersza zostanie odświeżony na podstawie zawartości bazy danych.

Instrukcję BROWSE można przełączyć w tryb automatycznego odświeżania poprzez wywołanie:

SET REFRESH TO <liczba>, gdzie <liczba> $\in$ <1,3600>.

Ustawienie trybu automatycznego odświeżania powoduje, że okresowo co <liczbę> sekund następuje aktualizacja obrazu wszystkich wyświetlanych wierszy na podstawie bazy danych. Oznacza to, że w chwilach odświeżania na ekranie pojawiają się zaktualizowane dane nawet wtedy, gdy aktualizowany wiersz nie jest wierszem bieżącym tabeli.

Własność odświeżania wyświetlanych danych posiadają jedynie instrukcje BROWSE, CHANGE, EDIT i nie można wprost tworzyć własnych, samoodświeżających się formatek.

Ustawienie zbyt małego czasu odświeżania może powodować widoczne zmiejszenie szybkości działania systemu.

3.5. Brak grupy instrukcji wymagających w trybie dzielonym wcześniejszego, jawnego blokowania rekordu

Instrukcje modyfikacji bieżącego rekordu np. REPLACE <pole> WITH <wyrażenie>, w systemie FoxPro zostały zaliczone do grupy rozkazów automatycznie blokujących rekord. Zrezygnowano z grupy instrukcji wymagających wcześniejszego jawnego blokowania rekordu, znanej z systemów dBaseIII+ czy FoxBASE+.

3.6. Grupa instrukcji odczytu danych, automatycznie blokujących plik

W systemie FoxBASE instrukcje typu SUM, COUNT, TOTAL, LIST, wykonujące operacje odczytu całych plików danych działają niezależnie od systemu blokad. Aby uniknąć niespójności odczytywanych danych, ze względu na ewentualne, równoczesne modyfikacje w innych węzłach sieci, przed wykonaniem którejś z wymienionych instrukcji, w systemie FoxBase należy wcześniej jawnie zablokować plik funkcją FLOCK.

W systemie FoxPro po ustawieniu SET LOCK ON instrukcje takie, jak: COUNT, CALCULATE, SUM, TOTAL, AVERAGE, SORT, COPY TO, LIST, REPORT dokonują automatycznej blokady pliku bazy danych. W celu zapewnienia zgodności z FoxBASE-m domyślnym ustawieniem jest SET LOCK OFF.

4. Podsumowanie

Mechanizmy obsługi wielodostępu w systemach FoxBASE/FoxPro nie należą obecnie do najbardziej zaawansowanych. Brak jest np. blokady dla odczytu, implementacji systemu użytkowników bazy i ich uprawnień.

Istotny jest również brak wbudowanych mechanizmów przetwarzania transakcyjnego. W uproszczeniu przetwarzanie transakcyjne polega na zapewnieniu przez system śledzenia wszystkich modyfikacji danych dokonywanych przez program. Faktyczna realizacja tych modyfikacji w bazie danych następuje dopiero po jawnym wystąpieniu rozkazu potwierdzającego. Jeżeli z jakichś przyczyn program nie zostanie zrealizowany do końca (czyli nie będzie rozkazu potwierdzenia zmian wprowadzonych przez program), to system automatycznie anuluje wcześniejsze modyfikacje. Taką przykładową przyczyną może być awaria komputera. W uproszczeniu transakcjami są na ogół fragmenty programów, dla których przeprowadzone muszą zostać wszystkie przewidziane modyfikacje w bazie danych ze względu na spójność danych. Mechanizm transakcji powinien być integralną częścią systemu zarządzania bazą danych. Występowanie takiego bezpiecznego mechanizmu przetwarzania transakcyjnego jest

warunkiem niezbędnym do utrzymania poprawności bazy danych w dużych, profesjonalnych zastosowaniach.

Pomimo wymienionych braków, system FoxPro użyty dla niewielkich zastosowań, ze względu na wiele innych zalet wart jest polecenia.

LITERATURA

[1] Jeffrey D. Ullman: Systemy baz danych, WNT, Warszawa 1988.
[2] FoxBASE+ User Manual
[3] FoxPro Developer's Guide
[4] FoxPro Commands & Functions

Recenzent: Dr hab. inż Stanisław Wołek

Wpłynęło do redakcji 8 grudnia 1993r.

Abstract

This article presents features of Multi-User environments of database management systems FoxBASE+ 2.0 and FoxPro/LAN 2.0.

The article explains a need of a locking data access system in a Multi-User environment.

Information about database file open modes (share and exclusive) are provided in the article. The article presents features of manual locking access functions (flock for database file locking, rlock for one database record locking) and unlocking command.

The article describes groups of commands of FoxBase+ system which automatically attempt to lock a record or file (in a shared open mode) before executing any other operation. The article explains system functioning when a collision error is detected. An user routine can

handle such network collision error. After error processing the user routine can return control to a calling program and re-execute last command. Groups of commands which do not place lock on a database file or required exclusive open a database file are considered too.

The article describe new features of FoxPro/LAN 2.0 related to Multi-User environment. Possibility of record multi-locking in a one work area and controlling the number or time of automatic attempting to lock are presented. New feature of browse command related to refreshing database display and scope of record locking are considered.