ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1993

Seria: INFORMATYKA z. 25 Nr kol. 1236

Hafedh ZGHIDI

# CHARAKTERYSTYKA WYBRANYCH CECH SYSTEMU RELACYJNEJ BAZY DANYCH PROGRESS

Streszczenie. Niniejsze opracowanie poświęcone jest relacyjnej bazie danych Progress. Pierwsza część artykułu przybliża możliwości funkcjonalne tego systemu i jego nowości w porównaniu z innymi systemami. Opisuje również najistotniejsze cechy systemu a w szczególności mechanizmy bezpieczeństwa i integralności danych, które nie były w dotychczasowych publikacjach wystarczająco naświetlone. Druga część natomiast przedstawia wyniki testów wykonywanych w systemach baz danych: Progress, Progress/SQL i FoxBase celem oceny wydajności tych systemów. Jako kryterium wydajności przyjęto czas realizacji pewnych zadań.

# THE CHARACTERISTIC OF SOME CHOSEN FEATURES OF RELATIONAL DATABASE SYSTEM PROGRESS

Summary. The paper presents the Progress relational database.
In the first part the functional possibilities of this program are described, with particular emphasis on its innovative character in comparison with other programs. Also, the most important features of this system are presented, especially the implements of data safety and integrity, which have not been accounted for sufficiently in other publications. In the second part of this paper, the results of tests carried out in the following database systems are demonstrated: Progress, Progress/SQL and FoxBase, the main objective of the tests being the estimation of the efficiency of these systems. The performance time of given search tasks has been assumed as the measure of efficiency.

## CHARAKTERISTIK AUSGEWÄHLTER MERKMALE DES SYSTEM VON BEZUGDATENBANK PROGRESS

Zusammenfassung. Die voliegende bearbeitung wurde der bezugdatenbank Progress gewidmet. Der erste teil des artikels macht mit den funktionellen möglichkeiten dieses Systems und dessen neuartigkeit in Bezug auf andere systeme bekannt. Er beschreibt auch die wesentlichsten merkmale des Systems insbesondere die mechanizmen der sicherheit und datenintegrität die in den bisher erschienenen veröffentlichungen nicht ausreichend exponiert wurden. Der zweite teil dagegen stellt die Ergebnisse von Testen dar, die in den Datenbanksystemen Progress, Progress/SQL und FoxBase durchgeführt wurden, um die Leistung dieser Systeme abzuschätzen. Als Leistungskriterium wurde die Realisierungszeit von gewählten Aufsucheaufgaben angenommen.

# 1. Wstęp

Od powstania pierwszej wersji Progressu minęło już prawie osiem lat, aktualnie sprzedawana jest szósta jego wersja a na przyszły rok producenci zapowiadają siódmą. Mimo takiej bogaty historii i licznych użytkowników tego systemu na całym świecie i mimo ogromnych możliwości, jaki ten system oferuje użytkownikowi, jest on w Polsce mało znany. Pojawił się on tu dopiero dwa lata temu, a nie wszyscy użytkownicy baz danych znają jego możliwości. W kilku słowach można powiedzieć, że jest to system zarządzania relacyjną bazą danych i język programowania czwartej generacji. Jest to narzędzie wspomagające pracę projektanta i programisty we wszystkich fazach tworzenia aplikacji. Umożliwia tworzenie wygodnych aplikacji w dużo krótszym czasie niż w tradycyjnych systemach, a jego szczególną zaletą jest przenośność aplikacji między różnymi systemami operacyjnymi DOS, OS/2, UNIX, VMS i BTOS/CTOS na różnych platformach sprzętowych i przy różnych protokołach sieciowych (jak TCP/IP, NETBIOS, SPX/IPX, DECNET), zapewniając integralnośc i bezpieczeństwo danych. Celem pierwszej części niniejszej pracy jest przybliżenie możliwości funkcjonalnych tego systemu i jego nowości w porównaniu z innymi systemami, a w szczególności mechanizmów bezpieczeństwa i integralności danych. Te cechy systemu Progress nie były w dotychczasowych publikacjach wystarczająco naświetlone. Druga część poświęcona będzie porównaniu wydajności języków Progress, Progress/SQL i FoxBase w realizacji zadań wyszukiwania w specjalnie przygotowanej bazie.

# 2. Środowisko Progressu

Progress oferuje użytkownikowi zintegrowane środowisko projektowe, zawierające: słownik bazy danych, system zarządania bazą danych, edytor oraz kompilator.

## 2.1. Edytor

Edytor Progressa jest pełnoekranowym edytorem, pozwalającym na wygodną pracę. Jego szczególną zaletą jest zestaw klawiszy funkcyjnych, które umożliwiają różnego rodzaju operacje. Dla przykładu F1: Kompliacja i uruchomienie procedur zawartych w aktualnym ekranie; F2: Wywołania głównego menu pomocy; F3: Włączenie lub wyłączenie klawisza Insert; F4: skok do końca linii itd.

Poza tym liczba linii, jakie edytor oferuje, zależy od konfiguracji systemu operacyjnego i od komputera. Określa się ona wzorem: *Il = (wielkość bufora)/82*, wielkość bufora waha od 2 kB do 63 kB dla mikroprocesorów Intel 8086 i 80286 i do 2 kB do 4 GB dla innych systemów.

## 2.2. Kompilator

Kompilator Progressa sprawdza poprawność składni procedur oraz jej zgodność ze schematem bazy danych. W przypadku pojawienia się błędu można go natychmiast poprawić i powtórzyć kompilację. Jeżeli nie ma błędów, to procedury są natychmiast uruchamiane. Działanie programu można w każdej chwili przerwać naciskjąc Ctrl-C. Podczas kompilacji wszystkie zmienne (lokalne, globalne, dzielone), zbiory robocze i rekordy aktywne muszą się zmieścić w buforze o wielkości od 1 do 36 kB dla S.O. DOS, OS/2, XENIX286, BTOS/CTOS, a dla pozostałych systemów operacyjnych wielkości bufora można zwiększać do 4 GB.

## 2.3. Słownik bazy danych (ang. Data Dictionary )

Jest to narzędzie systemowe do interakcyjnego definiowania i modyfikowania schematu bazy danych. Jest to odpowiednik programu Assist w dBase, oferujący szereg dodatkowych funkcji, z których najciekawsze to:

1. Tworzenie indeksów: Na poziomie słownika tworzy się indeksy dla poszczególnych tablic bazy danych. Umożliwia to Progressowi:

a) szybkie znalezienie rekordu,

b) automatyczne porządkowanie rekordów,

c) szybkie znalezienie rekordu powiązanego,

d) sprawdzenie unikalności indeksów (2 rekordy nie mogą mieć tej samej wartości klucza).

Warto podkreślić, że Progress sam zarządza indeksami, sam z nich korzysta bez wiedzy użytkownika.

2. Generowanie raportów: Jest dość cenna cecha podczas pracy zespołowej. Funkcja ta opisuje struktury bazy danych i powiązania między tablicami. Generowane raporty można wyświetlić na ekranie lub wydrukować.

3. Możliwość zabezpieczenia bazy danych przed obcym użytkownikiem, poprzez założenie hasła dostępu do struktury lub do uruchomienia pewnych procedur.

4. Możliwość składowania i odzyskania (ang. dumping i undumping) danych wykorzystywana przy przenoszeniu bazy danych do innych komputerów lub innych systemów operacyjnych.

5. Możliwość korzystania z języka SQL do definiowania struktur danych lub do napisania aplikacji.

6. Możliwość importu i eksportu danych do innych baz danych lub do innego formatu (tekstowy, sylk ...) oraz możliwość dołączania lub odłączenia innych baz danych (ang. connect/disconnect data base).

## 2.4. System zarządzanie bazą danych

System zarządzanie bazą danych (ang. Data Base Manager) zajmuje się kontrolą wykonywania stworzonych procedur. Umożliwia on dostęp do danych, zabezpieczanie danych, realizację transakcji oraz zachowuje więzy integralności i spójności referencyjnej danych.

W przypadku pracy w systemie wielozadaniowym zapewnia blokowanie rekordów aktualnie zajmowanych przez innego użytkownika. Blokowanie to jest realizowane na poziomie konkretnego rekordu co optymalizuje dostęp do zasobów.

Do głównych zadań systemu zarządzania bazą danych należy również ochrona danych przed utratą i przed nieodpowiedzialnym dostępem.

### 2.5. Język programowania

Jest pełnym językiem programowania, tzn. cały kod aplikacji może zostać napisany w Progressie bez konieczności sięgania do języków niższego poziomu. Możliwe jest również dołączenie zewnętrznych procedur napisanych np. w C [6].

Język Progress jest bardzo przyjazny i bliski naturalnego języka angielskiego, co ułatwia jego zapamiętanie i opanowanie. Oferuje bardzo bogaty zbiór instrukcji umożliwiając między innymi iteracje, tworzenie i przekazywanie tablic, definiowanie klawiszy i komunikację z systemem operacyjnym. Zbiór instrukcji jest w kolejnych wersjach Progressa coraz bogatszy i stale rozszerzany. Dostępne są również wersje Progressa komunikujące się z użytkownikiem w różnych językach, między innymi w języku polskim, umożliwiając również stosowanie polskich liter i sortowanie pól tekstowych z polskimi znakami, co odciąża użytkowników od pisania specjalnych procedur do realizacji tych funkcji.

## 3. Zbiory bazy danych

Progressowa baza danych jest pojedyńczym zbiorem o rozszerzeniu .db, zawierającym wszystkie zbiory i indeksy aplikacji. Tymczasem Progress korzysta ze zbioru bazy danych oraz z szeregu dodatkowych zbiorów z nimi związanych:

a. Zbiór obrazu pierwotnego o rozszerzeniu .bi (ang. before image), zapewniający integralność danych i zachowujący pierwotny stan bazy danych na wypadek błędu systemowego lub niekompletnej transakcji.

b. Zbiór blokujący bazy o rozszerzeniu .lk. Zbiór ten nie jest generowany dla wszystkich systemów operacyjnych, ma na celu blokowanie dostępu jednoużytkownikowego podczas pracy wieloużytkownikowej. Po prawidłowym zakończeniu sesji progressowej, zbiór ten jest automatycznie kasowany. W przeciwnym przypadku zbiór zostaje na dysku, co świadczy o błędnym zakończeniu ostatniej sesji.

c. Zbiór kartoteki wieloużytkownikowej o rozszerzeniu .ld generowany tylko w Unixie.

d. Zbiór statusów zameldowań o rozszerzeniu .lg zawierający informacje o dacie stworzenia bazy oraz datę i czas zameldownia i wymeldowania wszystkich użytkowników bazy, a także komunikaty serwera bazy.

I tak po utworzeniu nowej bazy danych o nazwie "demo" zbiory, jakie mogą być tworzone (zależnie od systemu operacyjnego), to:

1) zbiór danych : demo.db (zawsze),

2) zbiór obrazu pierwotnego : demo.bi (zawsze),

3) zbiór blokady : demo.lk,

4) zbiór kartoteki wieloużytkownikowej : demo.ld,

5) zbiór status komunikatów : demo.lg (zawsze).

Po prawidłowym zakończeniu pracy, Progress automatycznie kasuje zbiór blokady (.lk) i zbiór kartoteki wieloużytkownikowj (.ld), są one bowiem potrzebne tylko podczas pracy w Progressie.

## 4. Język Progress 4GL

Język czwartej generacji musi mieć następujące własności [5] [6]:.

a) obsługa ekranu,

b) możliwość iteracji,

c) wielodostęp,

d) zapewnienie bezpieczeństwa danych,

e) interfejs do systemu operacyjnego,

f) zapewnienie integralności danych,

g) kontrola poprawności danych,

h) zintegrowane środowisko projektowe,

i) możliwość generowanie raportów.

Progress spełnia powyższe wymagania i dlatego można powiedzić, że jest on językiem czwartej generacji.

## 5. Progress i XWindows

Przy użyciu języka Progress można tworzyć aplikacje pracujące w systemie Windows, umożliwiając tym samym użytkownikom korzystanie z różnych interfejsów oferowanych przez Windows. System z Windows jest oprogramowaniem umożliwiającym prezentowanie danych w formacie w pełni zgodnym z możliwościami Windows (skalowanie i przemieszczanie okien, zmiana atrybutów okien, używanie myszki itd.). Za pomocą tego systemu możliwe jest również uruchomienie kilku aplikacji progressowych w różnych okienkach tego samego ekranu. Jest to sieciowy system okienkowy. Serwer

obrazowania (ang. Display Server) zarządza protokołami komunikacji z użytkownikiem (z klientem) poprzez specjalnego rodzaju ekran rastrowy. Klienci to aplikacje, sprzęgane z systemem XLibrary, który zapewnia komunikację z serwerem, mogącym być na tej samej maszynie (komputerze), co klient lub na odległej maszynie.
Oznacza to, że za pomocą tego systemu można uruchomić aplikacje na odległej maszynie.

## 6. Progress/SQL

Żeby definiować strukturę bazy danych, można używać języka definiowania danych Progress/SQL Data Definition Language (DDL) korzystając z odpowiedniej opcji słownika bazy danych. Komendy języka Progress/SQL są traktowane interaktywnie. W aplikacjach można używać zarówno komendy SQL jak i zwykłych komend Progressa.Możliwe jest również korzystanie z SQL'a do udostępniania i aktualizowania danych w obcych bazach (nie progressowych) za pomocą tzw. gateway database (interfejs do bazy danych), które przetwarzają żądania użytkownika.

## 7. Progress i inne bazy danych

Progress umożliwia komunikację z innymi bazami danych. Jest możliwy import z bazy danych zapisanych w formacie dBase, które mogą być później obsługiwane przez procedury napisane w języku Progress.

Wyżej wymienione gateways (interfejsy do innych baz danych) umożliwiają współpracę z bazami danych Oracle, C-ISAM, Rdb/VMS, Sybase i OS/400. Oznacza to, że użytkownik zarówno w fazie programowania, jak i w fazie korzystania z aplikacji nie widzi żadnych różnic korzystając z bazy danych Progressa czy innej udostępnionej przez interfejs. Ma to szczególną wagę podczas pracy w systemie o heterogenicznym charakterze, gdyż pozwala zachować integralność danych.

## 8. Bezpieczeństwo danych

Progress posiada bardzo rozbudowany system zabezpieczania danych. Zadanie to jest jednym z głównych zadań systemu zarządzania bazą danych. Możliwe jest zabezpiecz-

enie przed obcym dostępem oraz przed utratą w wyniku awarii sprzętu lub awarii systemu.

## 8.1. Zabezpiecznie przed obcym dostępem

Słownik bazy danych umożliwia użytkownikowi identyfikację użytkowników i hasła. W momencie uruchomienia aplikacji Progress żąda podania nazwy użytkownika oraz jego hasła, zapewniając tym samym, że tylko znani użytkownicy mają dostęp do aplikacji. Dodatkowo można ustalić nawet blokadę dostępu na poziomie wybranych procedur. Dzieje się to w ten sposób, że aplikacja może sprawdzić prawa dostępu do uruchomienia procedury przez już zameldowanego użytkownika.

Do osiągnięcia tego celu można stosować jedną z dwóch metod:

1/ należy dla każdej procedury definiować listę użytkowników, którzy mogą ją uruchomić,

2/ należy definiować zbiór czynności (ang. activities file), który ustala dla każdej procedury aplikacji listę użytkowników, którzy mogą ją uruchomić.

## 8.2. Zabezpiecznie przed utratą

Dzieli się ono na zabezpieczenie przed awarią systemu i przed awarią sprzętu.

**8.2.1. Zabezpieczenie przed awarią systemu** Jest możliwe za pomocą automatycznego zachowania tzw. obrazu pierwotnego (ang. before imaging). Przed zmianą bazy danych w wyniku uruchomienia procedury, Progress kopiuje bazę do zbioru o rozszerzeniu .bi. Na wypadek awarii systemu przed zakończeniem transakcji lub przy błędnej transakcji Progress odzyska informacje z zbioru .bi i przywraca bazie jej poprzedni stan. Operacja ta:

- jest automatyczna (użytkownik nie musi jej wykonać),
- chroni przed awarią systemową,
- nie chroni danych przed awarią sprzętu,

**8.2.2. Zabezpieczenie przed awarią sprzętu**

Progress umożliwia ochronę danych przed awarią sprzętu, należy jednak te mechanizmy ustawić i samemu wykonywać.

Progress umożliwia tworzenie zbioru obrazu wtórnego (ang. after image) o rozszerzeniu .ai, po zakończeniu transakcji. Zbiór ten powinien być zapisany na innym dysku niż tym, na którym są zbiory bazy danych. Zawiera on stan bazy po wykonaniu tran-

sakcji lub podprogramu. Wykonanie tej czynności jest jednak opcjonalne, co oznacza, że użytkownik musi ją ustawić. Warto też pamiętać, że zbiór ten musi być tworzony na fizycznie innym dysku niż ten, na którym są podstawowe zbiory bazy.

Do tych zabezpieczeń należy również sporządzanie kopii bazy danych (ang. backup). Progress umożliwia nam również sporządzenie kopii przyrostkowych, przy czym odpowiedni dobór współczynnika redundancji gwarantuje nam zachowanie danych nawet przy utracie jednego elementu ciągu kopii przyrostkowych [6].

Następną cenną możliwością jest sporządzanie kopii bazy danych podczas pracy systemu (ang. backup on-line). Taki sposób kopiowania jest korzystny w systemach pracujących bez przerwy. Mając kopię bazy danych i zbiór obrazu wtórnego można odzyskać dane na wypadek awarii sprzętu (dysku). Operacja ta jest:

- nie automatyczna (użytkownik musi ją wykonać),
- wymaga reguralnego tworzenie kopii bazy danych,
- chroni przed utratą danych na wypadek awarii sprzętu, chyba że uszkodzenie dotyczy dysku zawierającego zbiory obrazu pierwotnego i wtórnego równocześnie.

Taki system zabezpieczenia gwarantuje wysoki poziom ochrony danych, jakiego inne systemy nie posiadają.

## 9. Programowanie w sieci

Progress umożliwia pracę w różnych typach sieci. Obejmuje to zarówno sieci jednolite, jak i różnorodne. Jest przystosowany do protokołów transmisji typu TCP/IP, NetBios, SPX/IPX, DECnet. W sieciach z systemem operacyjnym DOS (DOS LAN) z protokołami IBM NetBios lub SPX/IPX należy korzystać z wersji. Dos'a 3.2 lub późniejszej, Progress korzysta z protokołów transmisji do zarządzania przesyłem danych pomiędzy serwerem a stacjami roboczymi: Serwer może być również skonfigurowany tak, żeby pełnił rolę serwera (udostępnienie i kontrola danych), jak i stacji roboczej. Serwer to minimum komputer typu PC/AT z 640 kB RAM i dostatecznie dużym dyskiem.

Sieci z protokołem TCP/IP posiadają tzw gniazda (ang. sockets), których Progress korzysta do przesyłania danych. Sockets to narzędzie unixowe. W przypadku pracy innym systemem opercyjnym (jak Dos, Xenix) za pośrednictwem protokołu TCP/IP, należy instalować odpowiednie karty sieciowe i odpowiednie oprogramowanie do obsługi tego protokołu.

Praca w sieci z Progressem nie jest uzależniona ani od protokołu przesyłu ani od platformy sprzętowej. Oznacza to, że Progress może pracować w heteregonicznach sieciach, że sieć ta może być w każdej chwili rozbudowana, a używane komputery (zarówno serwer, jak i stacje robocze) mogą być zmienione na większe i szybsze bez konieczności zmiany kodu aplikacji.

Progressowa baza danych może być również rozproszona na kilku serwerach, z zapewnieniem integralności i spójności danych. Jest to możliwe dzięki zastosowaniu dwufazowego protokołu potwierdzeń (ang. two phase commit), który zapewnia wykonanie transakcji dopiero po nadejściu potwierdzeń od wszystkich serwerów biorących udział w tej operacji [6].

Dodatkową wygodą przy programowaniu w sieci jest zwolnienie programisty od konieczności "ręcznego" blokowania zasobów. Zadanie to wykonuje samodzielnie system zarządzania bazą danych. Blokowanie, jak już wspominałem wcześniej, wykonuje sie na poziomie rekordu, a nie całego pliku, co jest dodatkową zaletą, gdyż wyklucza blokowanie pozostałych rekordów dla innych użytkowników.

## 10. Integralność danych

Progress posiada wysoce niezawodny system zapewniający integralność i spójność danych. System zarządzania bazą danych sprawdza poprawność danych, a w przypadku niezgodności nie dopuszcza do zapisu operacji lub transakcji. Dodatkowo w słowniku bazy danych można definiować warunki przyjęcia, kasowania lub zmiany zawartości całego rekordu lub pojedynczego pola rekordu oraz rekordów logicznie ze sobą powiązanych. Po definiowaniu typu zmiennych, Progress automatycznie kontroluje ich poprawność. Na przykład po zdefiniowaniu zmienej x typu całkowitego bez znaku i przepisaniu jej wartości ujemnej, Progress sygnalizuje błąd. Podobnie jest dla innych typów danych, jak data, godzina itd. Jest to najniższy poziom integralności danych.

Progress umożliwia definiowanie kryteriów poprawności danych za pomocą dwóch metod:

1) na poziomie procedury korzystając z polecenia VALIDATE,

2) na poziomie słownika bazy danych.

Kryteria te dzielą się na 3 rodzaje:

a) sprawdzanie zakresu danych,

b) sprawdzanie przynależności wartości danych do zbioru kilku możliwych wartości,
c) sprawdzanie istnienia powiązanego rekordu.

Należy jednak przypomnieć, że Progress sprawdza poprawność danych tylko w chwili ich wprowadzania lub zmiany.

## 10.1. Sprawdzanie zakresu danych

Na poziomie słownika bazy danych można definiować zakres danych. Należy w polu valexp napisać odpowiedny warunek (np: cena > 100 and cena < 1000). W przypadku błędu (tzn. przy wprowadzeniu danych nie spełniających warunku), chcąc wyjaśnić użytkownikowi zasady wprowadzania danych, w polu valmsg można napisać odpowiedni komunikat (np: Cena nie może być mniejsza od 100 i większa od 1000).

## 10.2. Sprawdzanie przynależności danych do zbioru kilku możliwych wartości

Wiedząc, że wartość jakiegoś pola może być jedna z kilku znanych nam z góry wartości, możemy tworzyć listę tych wartości i sprawdzić poprawność wprowadzonych danych za pomocą funkcji LOOKUP. Funkcja ta zwraca nam pozycję wprowadzanej przez użytkownika wartości. Jeśli funkcja zwraca 0 oznacza to, że użytkownik wprowadził złe dane. Taki sposób sprawdzania dokonuje się przez napisanie w polu valexp odpowiedniego warunku. Dla przykłdu: LOOKUP (kolor: "czerwony, czarny, niebieki") <> 0.

## 10.3. Sprawdzanie istnienia powiązanego rekordu

W relacyjnych bazach danych wprowadzenie nowego rekordu do pliku często wiąże się z koniecznością istnienia odpowiedniego rekordu w innym pliku. Tak samo jest z operacją kasowania rekordu, która wymaga uprzedniego kasowania innego rekordu w innym pliku. Progress umożliwia sprawdzanie spełnienia takich warunków. Należy w polu valexp napisać: Can-Find (wskaźnik rekordu w drugim pliku). Oznacza to, że warunkiem stworzenia nowego rekordu jest istnienie innego rekordu w innym pliku, a w przypadku kasowania należy przy definiowaniu nowego pliku określić warunki kasowa-

nia rekordu. W celu wyjaśnienie zasady działania tych mechanizmów korzystamy z przykładowej bazy danych. Baza ta składa się z natępujących plików danych:

Pracownik

| Nazwa pola | Typ | Długość | Komentarz |
|---|---|---|---|
| Nrp | Integer | 3 | Numer pracownika |
| Nrz | Integer | 3 | Numer zespołu |
| Nazwisko | Character | 25 | Nazwisko pracownika |

Dochody

| Nazwa pola | Typ | Długość | Komentarz |
|---|---|---|---|
| Nrp | Integer | 3 | Numer pracownika |
| Nrt | Integer | 3 | Numer tematu |
| Kwota | Integer | 9 | Kwota zarobiona za realizację tematu |

Temat

| Nazwa pola | Typ | Długość | Komentarz |
|---|---|---|---|
| Nrt | Integer | 3 | Numer tematu |
| Nazwat | Character | 25 | Nazwa tematu |
| Nazwkiert | Character | 25 | Nazwisko kierownika tematu |

Zespol

| Nazwa pola | Typ | Długość | Komentarz |
|---|---|---|---|
| Nrz | Integer | 3 | Numer zespołu |
| Nazwaz | Character | 25 | Nazwa zespołu |
| Nazwkierz | Character | 25 | Nazwisko kierownika zespołu |

Rekordy plików Zespól, Pracowni i Temat opisują istniejące zespoły, ich pracowników oraz tematy projektów realizowanych przez pracowników. Każdy rekord pliku

Dochody określa sumaryczną kwotę zarobioną przez pracownika (Nrp) w ramach tematu (Nrt).

Przy definiowaniu pliku Pracowni, chcąc zapobiegać błędnemu kasowaniu rekordu, należy upewnić się, że w pliku dochody nie istnieje żaden rekord, którego pole nrp = pole nrp w pliku Pracownik (nie można kasować danego pracownika przed kasowaniem jego dochodów). W tym celu należy w polu valexp napisać następujący warunek: NOT (CAN-FIND (dochody OF pracownik)). Odpowiedni komunikat wyjaśniający takie postępowanie można wprowadzić w polu valmsg: "Dochody tego pracownika jeszcze nie są skasowane z pliku Dochody". Należy przy tym podkreślić, że korzystając z klauzuli CAN-FIND, należy te dwa pliki (Pracownik i Dochody) indeksować według tego samego klucza Nrp i dodatkowo jeden z tych indeksów musi być unikalny (unique).

Drugim aspektem zapewnienia integralności danych jest możliwość kontroli przy wprowadzaniu nowego rekordu do bazy. Przykładowo, chcąc wprowadzić dochody pewnego pracownika za realizację tematu o numerze 10, muszę być pewien, że taki numer tematu już istnieje w pliku Temat. W tym celu w polu valexp przy definiowaniu pole Nrt należy wprowadzić taki warunek: CAN-FIND (temat OF dochody). W polu valmsg można wprowadzić komunikat wyjaśniający:

"Numer tematu musi już istnieć w pliku Temat. Spróbuj jeszcze raz lub F4 rezygnacja".

Oczywiście, pliki te muszą być indeksowane według pola Nrt, a jeden z indeksów musi być unikalny.

Trzecim aspektem integralności danych jest kontrola przy wprowadzaniu wartości pola rekordu. Przykładowo wiedząc, że numery zespołów należą do przydziału 1..99 można w polu valexp przy definiowaniu pole Nrz pliku Zespol wprwadzić następny warunek w polu valexp: nrz > 0 AND nrz < 100. Komunikat wyjaśniający może zatem być, w polu valmsg: "Numer zespolu musi być > 0 i < 100". Zapobiega to wprowadzeniu do pola nrz wartości zerowych, ujemnych lub większych od 99.

Może się zdarzyć, że wyrażenie warunkowe jest za długie i nie zmieści się w polu valexp słownika bazy danych. Możliwe jest wtedy zapisanie tego wyrażenia w zbiorze 'include', a w polu valexp należy wskazać nazwę tego zbioru w nawiasach klamrowych (np. valexp: {warunki.v}).

Kryteria poprawności danych mogą być również definiowane na poziomie procedury. Do tego celu korzystać należy z klauzuli VALIDATE oraz z funkcji IF...THEN...ELSE. Tym sposobem można definiować skomplikowane warunki poprawności.

Warunki poprawności danych są w Progressie sprawdzane tylko przy aktualizacji pól (zmienna, pole rekordu, rekordu powiązanego) lub przy kasowaniu rekordu, nie dotyczą natomiast rekordów już w bazie zapisanych. W przypadku zmiany warunków poprawności (integralności) danych w słowniku bazy danych, należy ponownie kompilować procedury z nich korzystające.

W niektórych przypadkach, użytkownik jest zmuszony do pominięcia warunku poprawności danych zdefiniowanych w słowniku. Progress umożliwia mu ignorowanie tych warunków poleceniem NO-VALIDATE. (np. UPDATE nrz with NO-VALIDATE) => (nrz = 110).

Wnioski. Zalety Progressa w stosunku do innych baz danych można streścić w następujących punktach:

1. Bardzo rozbudowany słownik bazy danych, który udostępnia dodatkowe funkcje oprócz tradycyjnego definiowania pól rekordów.
2. Silne instrukcje, które umożliwiają pisanie krótkich, ale skomplikowanych aplikacji w bardzo krótkim czasie.
3. Dobrze rozwiązany system zabezpieczania danych.
4. Zapewnienie integralności danych na różnych poziomach i różnymi sposobami.
5. Transakcyjność systemu polegająca na dzieleniu programu na fragmenty (bloki), które wykonywane są do końca lub wcale. System zarządzania bazą danych automatycznie wykonuje ten podział, również programista może go wykonać instrukcją: DO TRANSACTION: ... END.
6. Automatyczny dobór indeksów, zwalniający od tej czynności programistę.
7. Efektywne narzędzia (FAST TRACK) do szybkiego tworzenia struktury bazy danych, formatek ekranowych i tabulogramów.
8. Automatyczne blokowanie rekordu podczas pracy w sieci.
9. Możliwość pracy w różnych systemach operacyjnych, na różnych platformach sprzętowych i różnymi protokołami transmisji.
10. Przenośność aplikacji na inne systemy bez konieczności zmiany kodu z zapewnieniem integralności danych.
11. Możliwość pracy z różnymi bazami danych, w tym możliwość dostępu do danych za pomocą tzw. interfejsu do baz danych (ang. gateway).
12. Wielojęzyczność systemu.

Biorąc pod uwagę wszystkie przedstawione cechy można twierdzić, że Progress jest bardzo dobrym narzędziem do zarządzania relacyjną bazą danych, zapewniającym użytkownikowi wygodę podczas pracy i doskonały system zabezpieczenia i kontroli integralności danych. Kolejne wersje Progressa są coraz doskonalsze i bardziej rozbudowane o dodatkowe instrukcje i narzędzia systemowe, co zapewnia coraz większą liczbę zwolenników i użytkowników.

# 11. Badanie efektywności

W tej części opracowania przedstawimy i umówimy wyniki testów wydajności Progressa, Progress/SQL i FoxBase. Testy te zostały przeprowadzone na komputerze typu IBM/PC/AT/386 DX z zegarem 40Mhz i 1MB RAM pracującym pod systemem operacyjnym Dos.

Do tego celu została stworzona baza danych składająca się z trzech plików o następującej strukturze danych:

studenci

| Nazwa Pola | Typ | Długość |
|---|---|---|
| Nazwisko | Character | 25 |
| Imie | Character | 25 |
| Album | Character | 5 |
| Stypendium | Character | 6 |
| Inne pola | | 184 |

egzaminy

| Nazwa Pola | Typ | Długość |
|---|---|---|
| Album | Character | 5 |
| Przedmiot | Character | 5 |
| Ocena | Character | 3 |
| Data | Character | 8 |
| Inne pola | | 15 |

stypendia

| Nazwa Pola | Typ | Długość |
|---|---|---|
| Album | Character | 5 |
| Rok | Character | 5 |
| Miesiac | Character | 2 |
| Stypendium | Character | 1 |
| Potracenie | Character | 8 |
| Stawka | Character | 8 |
| Wyrownanie | Character | 8 |
| Zaliczka | Character | 8 |
| Podatek | Character | 8 |
| Razem | Character | 8 |

Celem testów jest porównanie wydajności (szybkości działania) programów wykonanych w językach Progress, Progress/SQL i FoxBase, wpływ wielkości zbiorów danych (liczba rekordów w pliku) i wpływ operacji indeksowania na przyspieszenie otrzymania odpowiedzi na zadane pytania. W tym celu zostały sformułowane dwa pytania, które dalej będziemy nazwać zadanie I i zadanie II.

**Zadanie I.** Dla studentów otrzymujących stypendium typu "S" wyszukać nazwiska tych studentów oraz wszystkie kwoty stypendiów wraz z miesiącem i rokiem wypłaty.

**Zadanie II.** Wyszukać nazwiska studentów, którzy egzamin z przedmiotu "RPIS" zaliczyli na ocenę >= 4,5.

Test pierwszy został przeprowadzony na plikach o wielkości:

| Nazwa pliku | Liczba rekordów |
|---|---|
| Studenci | 10158 |
| Egzaminy | 6514 |
| Stypendia | 3759 |

Wyszukane zostały odpowiednio dla zadania I i zadania II 360 i 34 rekordy.

Test drugi został przeprowadzony na plikach o wielkości:

| Nazwa pliku | Liczba rekordów |
|---|---|
| Studenci | 5000 |
| Egzaminy | 3000 |
| Stypendia | 2000 |

Wyszukane zostały odpowiednio dla zadania I i zadania II 132 i 34 rekordy.

Powyższe testy zostały powtórzone na tych samych plikach po wykonaniu indeksowania według odpowiedniego klucza. Przez wynik testu będziemy dalej rozumieli czas wykonywania zadań: zadanie I i zadanie II w systemie Progress, Progress/SQL i FoxBase. A oto wyniki badań:

Tabela T1

Wyniki testu pierwszego (baza nie indeksowana)

| | Progress | Prog/SQL | FoxBase |
|---|---|---|---|
| Zadanie I | 46 min 36 s | 1 min 9 s | 5 min 16 s |
| Zadanie II | 11 min 31 s | 37 s | 1 min 34 s |

Tabela T2

Wyniki testu pierwszego (baza indeksowana)

| | Progress | Prog/SQL | FoxBase |
|---|---|---|---|
| Zadanie I | 1 min 47 s | 6 min 25 s | 12 s |
| Zadanie II | 11 min 25 s | 2 min 37 s | 3 s |

Tabela T3

Wyniki testu drugiego (baza nie indeksowana)

| | Progress | Prog/SQL | FoxBase |
|---|---|---|---|
| Zadanie I | 10 min 42 s | 28 s | 1 min 21 s |
| Zadanie II | 5 min 53 s | 19 s | 39 s |

Tabela T4

Wyniki testu drugiego (baza indeksowana)

| | Progress | Prog/SQL | FoxBase |
|---|---|---|---|
| Zadanie I | 51 s | 2 min 30 s | 6 s |
| Zadanie II | 30 s | 1 min 10 s | 2 s |

Analizę wyników przeprowadzimy najpierw dla zadania I:
Czasy otrzymane po wykonaniu testu pierwszego (baza większa) pokazują, że Progress jest bardzo wolny, gdyż realizowanie zadania I trwa ponad 46 min. FoxBase jest prawie 9 razy szybszy: 5 min 16 s. SQL jest prawie 40 razy wydajniejszy od Progressa i prawie 5 razy od FoxBase.

Dla operacji na mniejszej bazie (o prawie 50% mniejsza) czas wykonywania zadania I w Progressie zmniejszył sie prawie 4 razy (z 46 min 36 s na 10 min 42 s), ale stosunek do FoxBase został jednak taki sam (9 razy wolniej). Stosunek czasów realizacji SQL do Progressa jest mniejszy i wynosi mniej więcej 23, a stosunek SQL do FoxBase zmalał z 5 do 3.

Wyniki te pokazują, że przy wzroście liczby rekordów wykorzystanie tylko rozwiązań bazujących na iteracji po wszystkich rekordach staje się zupełnie nieefektywne (patrz teksty źródłowe programów napisanych w Progressie i FoxBasie, natomiast w SQL korzystano z klauzuli SELECT). Zmniejszając wielkość bazy stosunek Progress do FoxBase nie zmienił się (ponieważ oba programy napisane w Progressie i FoxBasie korzystają z mechanizmów iteracji), ale stosunek SQL do nich zmalał.

Operując na bazach indeksowanych należy zwrócić uwagę na wzrost wydajności Progressa aż o 26 razy w przydadku większej bazy i 12 razy w przypadku bazy mniejszej.
Stosunek Progressa do FoxBase w tym przypadku jest również taki sam (FoxBase jest 9 razy szybszy).

Zaskakujący jest natomiast fakt, że indeksowanie pogorszyło efektywności języka Progress/SQL. We wszystkich testach otrzymano gorsze wyniki po indeksowaniu bazy. Zarówno dla zadania I jak i dla zadania II wyniki te były prawie o 5 razy gorsze, a stosunek SQL do Progressa i FoxBase pogorszył się kilkakrotnie.

Powyższa analiza dotyczy zadania I. Porównując te wyniki dla obu zadań (zadanie I i zadanie II) możemy stwierdzić, że dla zadania II różnice pomiędzy Progress, SQL i FoxBase są inne. Spowodowane jest to tym, że zadania te operują na różnych plikach, o

innej strukturze danych i o różnej liczbie rekordów, ale charakter tych różnic zostaje taki sam.

Ogólnie można stwierdzić że FoxBase jest szybszy od Progressa. Jest to zrozumiałe, ponieważ Progress jest o wiele bardziej złożonym narzędziem do operowania na relacyjnych bazach danych. Mechanizmy jakie on posiada (sprawdzenie integralności i spójności danych, bezpieczeństwo danych itd.) powodują wydłużenie czasu działania. Tymczasem FoxBase jest o wiele prostszym narzędziem programowym i to jest źródłem jego szybkości. Niemniej jednak wyniki są w pewnym stopniu zaskakujące ze względu na wielką różnicę czasów działania. Progress/SQL jest najwydajniejszy, ale tylko w przypadku bazy nieindeksowanej, w przeciwnym przypadku jest on najwolniejszy. Może to być związane z zasadami indeksowania Progressa, które jednak nie są ujawniane użytkownikowi.

Na zakończenie chciałbym zwrócić uwagę na moc instrukcji Progressa, które pozwalają programistom na napisanie większych aplikacji dużo wygodniej i szybciej niż umożliwia to FoxBase. Program napisany w Progressie jest o wiele krótszy i czytelniejszy od tego samego napisanego w FoxBase. Jest to bardzo cenną cechą, która odróżnia Progress od pozostałych narzędzi baz danych. W tym celu dołączam teksty źródłowe programów rozwiązujących zadania I.

*Treść źródłowa rozwiązania zadania I w języku Progress:*

```
OUTPUT TO prg1.txt.
DISPLAY STRING (TIME, "HH:MM:SS").
FOR EACH studenci WHERE stypendium = 'S':
  FOR EACH stypendia WHERE album = studenci.album:
    DISPLAY studenci.nazwisko stypendia.miesac stypendia.rok stypendia.razem.
  END.
END.
DISPLAY STRING (TIME, "HH:MM:SS").
OUTPUT CLOSE.
```

*Treść źródłowa rozwiązania zadania I w języku Progress/SQL:*

```
OUTPUT TO sql1.txt.
DISPLAY STRING (TIME, "HH:MM:SS").
  SELECT album, miesiac, rok, razem from stypendia s
    WHERE album IN
```

```
    (SELECT album FROM studenci t
        WHERE t.stypendium = 'S').
DISPLAY STRING (TIME, "HH:MM:SS").
OUTPUT CLOSE.
```

*Treść źródłowa rozwiązania zadania I w języku FoxBase:*

```
SET CONSOLE OFF
SET DEVICE TO PRINT
SET PRINT ON
SET PRINTER TO fprg1.txt
?TIME()
SELECT 1
USE studenci
SELECT 2
USE stypendia
SELECT 1
DO WHILE .NOT. EOF()
 IF stypendium = 'S    '
  nr = album
  naz = nazwisko
  SELECT 2
  LOCATE FOR album = nr
  DO WHILE .NOT. EOF()
   IF FOUND()
    @PROW(), 1 SAY naz
    @PROW(), PCOL() SAY miesiac
    @PROW(), PCOL() SAY rok
    @PROW(), PCOL() SAY razem
   ENDIF
   SKIP 1
   CONTINUE
  ENDDO
 ENDIF
 SELECT 1
 SKIP 1
```

```
ENDDO
?TIME()
SET CONSOLE ON
SET PRINT OFF
SET DEVICE TO SCREEN
```

## LITERATURA

[1] Progress, Programming Hand Book, Progress Software Corporation, 1991.
[2] Progress, Language Tutorial, Progress Software Corporation, 1991.
[3] Progress, System Administration I, II, Progress Software Corporation, 1991.
[4] FoxBase, FoxBase+, InterSofland sp zoo Warszawa 1990.
[5] James Martin, Fourth-Generation Languages, 1985.
[6] Anna Ostaszewska, Czas na Progress, PCkurier nr 2, 1993.

Recenzent: Dr hab.inż. Stanisław Wolek

Wpłynęło do redakcji 3 sierpnia 1993 r.

## Abstract

Progress 4GL is a system of managing relational databases and the fourth generation language. It is a tool assisting the work of a designer or a programmer in all phases of the generation of applications.In comparison with other traditional systems, it facilitates the generation of convenient applications in a considerably shorter time, and one of its characteristic features is the ability of transferring applications among different operational systems as DOS, OS/2, UNIX, VMS and BTOS/CTOS, on different equipment and with the use of different network protocols (such as TPC/IP., NETBIOS, SPX/IPX, DECNET) providing data safety and integrity. The object of the first part of the paper is to demonstrate the functional possibilities of this system and its new character in comparison with other systems, especially the implements of data safety and integrity, because, so far these features of the program have not been adequately accounted for. Particular attention has been given to the implements of data integrity, which were explained by the example of simple databases consisting of three files: Employee, Income, Research subject.
Detailed description of the files in included in sub-chapter 9.3.
In the second part of the paper, the comparison of the efficiency of the following

languages has been drawn: Progress, Progress/SQL, FoxBase, with regard to their ability of searching data in the specifically prepared base. To evaluate the efficiency of these programs, the time of the performance of tasks was measured. Also, the relations between the program efficiency and the file ordering and capacity were investigated.

The database on which the tasks were performed consists of three files: Students, Exams, Grants/Scholarships. Detailed description of the database and the tasks contents are included in chapter 10.

The results of tests are presented in four tables. Tables $T_1$, $T_2$ contain the results of the first test for the index and non-index database respectively. Tables $T_3$, $T_4$ contain the results of the second test for the non-index and index database.