Małe bazy danych w mikrokomputerze

Mgr Jan RASZEWSKI

Baza danych — plik lub pliki danych (ewentualnie ze zbiorami indeksów) zawierające całość przetwarzanych informacji.

Jednym z głównych zastosowań informatyki, od początku jej istnienia, było gromadzenie i przetwarzanie wielkich ilości informacji. Obecnie trzeba tworzyć bazy danych tak duże, że przeszukiwanie ich przekracza możliwości najpotężniejszych komputerów. Jedyne wyjście polega na tym, żeby pamiętać tylko wybrane, podstawowe informacje. Pozostałe komputer powinien umieć wydedukować (pisał o tym J. Pacholczyk w *Delcie* 1/1986). Tworzenie małych baz danych nigdy nie było zadaniem informatyki. Aż do czasu, kiedy pojawiły się mikrokomputery.

Czy mikrokomputer w ogóle nadaje się do tego celu? To zależy od jego parametrów. A zwłaszcza od rozmiaru pamięci wewnętrznej oraz pojemności pamięci masowej. Im większa pamięć wewnętrzna, tym bardziej zaawansowane może być oprogramowanie. Im większa pamięć masowa, tym bardziej rozbudowana może być sama baza.

Do obsługi niewielkich baz danych najczęściej wykorzystywany jest system dBASE, rozpowszechniony w dwóch wersjach. Wersja II powstała w 1982 roku dla maszyn zawierających co najmniej 64 kB pamięci operacyjnej i jeden napęd dysków elastycznych. W 1984 roku z myślą o komputerach szesnastobitowych powstała wersja III, która wymaga 256 kB oraz dwóch dysków. dBASE III jest jednak tak rozbudowany, że staje się wygodny dopiero przy współpracy z dyskiem stałym typu Winchester. Wymagania te spełnia coraz bardziej popularny, również w Polsce, profesjonalny mikrokomputer IBM PC/XT.

Najistotniejszym problemem jest szybka odpowiedź programu na zadane pytanie, czyli odpowiedni algorytm wyszukiwania informacji w bazie danych. Wyobraźmy sobie, że mamy bazę danych zawierającą dane bibliograficzne na temat artykułów i że chcemy znaleźć artykuł prof. Trąbalskiego. Jak to zrobić najłatwiej? Ano, po kolei: może pierwszy artykuł? Jeśli nie, to może drugi itd. Średnio przeszukać potrzeba około połowy bazy danych. A teraz wyobraźmy sobie, jak byśmy szukali, gdyby baza danych była uporządkowana alfabetycznie. Przede wszystkim nie zaczynalibyśmy od początku, a od środka. Po pierwszym porównaniu wiedzielibyśmy, czy szukany autor znajduje się w pierwszej, czy w drugiej połowie. Po drugim porównaniu mielibyśmy oszacowane jego położenie z dokładnością do ćwiartki itd. Łatwo widać, że przy takim postępowaniu znajdziemy odpowiedź NAJPÓŹNIEJ po $\log_2 n$ porównań, gdzie n to liczba artykułów. Nie jest to duża liczba. Dla tysiąca wynosi 10, dla miliona — 20, dla miliarda (a, jak sądzę, nie napisano jeszcze tylu artykułów) — 30. Metoda ta nazywa się *przeszukiwaniem binarnym*.

A więc mamy cudowną metodę na błyskawiczne wyszukiwanie? Niezupełnie. Pamiętajmy, że zrobiliśmy jedno niebagatelne założenie — że dane są uporządkowane. Niestety, algorytmy sortowania są bardzo złożone (patrz *Delta* 9/1985, artykuł T. Przytyckiej). W praktyce opłaca się sortować tylko takie bazy danych, w których zmiany są wprowadzane bardzo rzadko. Poza tym zwróćmy uwagę, że samo sortowanie nie zawsze rozwiązuje problem. Skoro mamy dane posortowane według nazwisk autorów, to nie są one posortowane według roczników. Nie da się więc już wykorzystać naszego szybkiego algorytmu wyszukiwania do znalezienia artykułów z określonego roku.

Plik danych — zbiór rekordów zawierających opis obiektów tego samego rodzaju.

Są jednak inne metody. Najpowszechniej stosowana to korzystanie z tzw. *plików indeksowanych*. Proces indeksowania polega na tym, że tworzymy dodatkowy plik, zawierający informacje na temat położenia rekordów w pliku danych. Plik zawierający dane nie ulega zmianie. Natomiast para plików: dane + indeksy, jest widziana przez algorytm wyszukiwania tak, jakby dane były posortowane. Dla jednej bazy danych możemy założyć wiele różnych plików z indeksami. W ten sposób możemy udawać, że nasze dane są posortowane zarówno według nazwisk autorów, jak według roczników albo tytułów czasopism.

Zapyta ktoś: po co wyszukiwanie liniowe, skoro można tak szybko wyszukiwać po indeksach? Wyszukiwanie za pomocą indeksów jest jednak mniej uniwersalne. Można je stosować tylko wtedy, gdy odnosi się do pola danych, po którym wykonano indeksowanie. Jeśli chcemy przeszukiwać dane według bardziej złożonych kryteriów, musimy korzystać ze znacznie wolniejszego, ale ogólnego wyszukiwania liniowego. Dlatego właśnie dBASE oferuje dwie instrukcje wyszukiwania: wolno działającą LOCATE, której parametrem może być wyrażenie opisujące dowolne informacje występujące wśród danych lub związki między nimi, oraz bardzo szybką FIND, której parametrem może być tylko konkretna wartość pola danych, według którego plik był wcześniej indeksowany.

Jeśli na przykład rekord danych zawiera pola *Nazwisko* i *Rok wydania*, to instrukcje

```
FIND Trąbalski
oraz
LOCATE FOR Nazwisko = ,,Trąbalski''
```

obie znajdują kolejny artykuł Trąbalskiego, choć pierwsza jest wielokrotnie szybsza. Ale jeśli chcemy znaleźć kolejny artykuł Trąbalskiego z lat 1970—1976, to musimy się posłużyć instrukcją LOCATE FOR Nazwisko = ,,Trąbalski''. AND. Rok wydania > 1969 .AND. Rok wydania < 1977.

Rozwiązanie zadania M 445. Nie można. Ponumerujmy pola szachownicy liczbami od 1 do 100 w taki sposób, by w k-tym wierszu znalazły się kolejno liczby $10(k-1)+1$, $10(k-1)+2$, ..., $10k$. Dwadzieścia pięć kostek domina ustawionych poziomo pokrywa wtedy dwadzieścia pięć liczb parzystych i dwadzieścia pięć nieparzystych. Kostka pionowa pokrywa parzystą ilość liczb nieparzystych (0 lub 2). Kostki pionowe nie mogą więc pokryć pozostałych dwudziestu pięciu liczb nieparzystych.

Projektując algorytmy wyszukiwania musimy pamiętać, że trzeba pogodzić sprzeczne cele. Podczas wprowadzania i modyfikowania danych trzeba aktualizować pliki z indeksami. A zatem im więcej plików indeksowanych, tym dłużej będzie trwało aktualizowanie informacji.

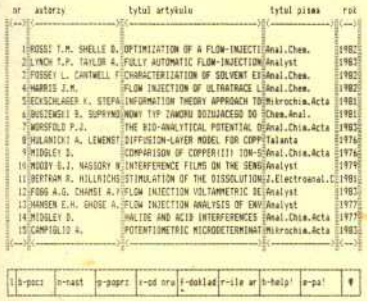
Oprogramowanie bazy danych artykułów, w którym brałem niedawno udział, jest typowym zagadnieniem z dziedziny małych baz danych. Zadanie w pierwszej chwili może się wydać banalne: stworzyć katalog artykułów naukowych dla pracowników jednej z pracowni na Wydziale Chemii UW. Jest ich pięciu, każdy ma 1000 do 2000 artykułów. Niektóre jednak się powtarzają, więc w sumie liczba różnych artykułów nie przekracza 6000. Dla każdego artykułu trzeba pamiętać: nazwiska autorów, tytuł artykułu, tytuł pisma, rok, tom, strony, kilka linijek streszczenia, dla łatwiejszej identyfikacji, kilka słów kluczowych oraz identyfikatory właścicieli artykułów. W sumie dla każdego artykułu niespełna pięćset znaków — a więc około 500 bajtów. Jeśli obsługiwać mamy 6000 artykułów, to musimy przyjąć na zapas, że w ciągu kilku lat może być ich nawet 10000. Po przemnożeniu przez długość opisu artykułu okaże się, że baza danych zajmie 6 MB. Chemikom trochę zrzedła mina, gdy dowiedzieli się, że nie chodzi o kilka dyskietek, a o znaczną część 10 megabajtowego dysku stałego w ich mikrokomputerze IBM PC/XT.

Pole rekordu — opis pojedynczej informacji dotyczącej jednego obiektu, np. nazwisko autora, tytuł artykułu lub tytuł czasopisma.

Rekord — całkowity opis obiektu, np. w omawianym przykładzie opis jednego artykułu.

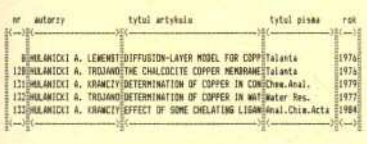
System baz danych (ang. DBMS — Data Base Management System) — baza danych wraz z programami, które zapewniają aktualizowanie oraz wszechstronne wyszukiwanie informacji.

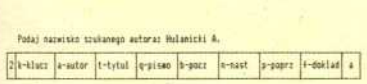
Pierwszą rzeczą, jaką należy zrobić przystępując do pracy nad bazą danych, jest określenie jej struktury. Trzeba więc ustalić, co będzie opisywał pojedynczy rekord (w tym przypadku — jeden artykuł), oraz jakie będą pola rekordu. Następna czynność to ustalenie operacji, jakie będą wykonywane. Każdy system baz danych musi zapewniać realizację podstawowych operacji, takich jak wprowadzanie, modyfikowanie, usuwanie i, oczywiście, wyszukiwanie danych. Jak już widzieliśmy, najistotniejszy jest problem wyszukiwania informacji i pod jego kątem należy projektować pozostałe elementy systemu. W naszym systemie szybkie wyszukiwanie odbywa się według nazwisk autorów oraz słów kluczowych. Nieco wolniejsze — według koniunkcji autorów i słów kluczowych. Natomiast z metody liniowej korzystają najbardziej złożone sposoby wyszukiwania, np. według konkretnego słowa występującego w tytule artykułu.



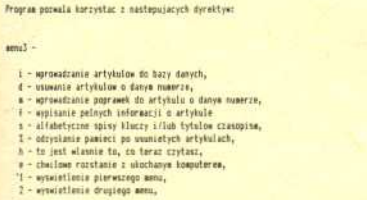
Rys. 1. Postać ekranu po wykonaniu dyrektywy „b", wyświetlającej skrótowy opis pierwszych piętnastu artykułów.

Osobnym zagadnieniem jest zorganizowanie sprawnej i wygodnej komunikacji z użytkownikiem systemu. Już w tak prostym, jak omawiany tutaj, systemie użytkownik dostał do dyspozycji około dwudziestu dyrektyw. Nie należy wymagać, żeby uczył się ich nazw albo skrótów na pamięć. Najkorzystniejszą metodą jest zaprojektowanie tzw. systemu sterowanego przez menu. Użytkownik powinien mieć zawsze na ekranie wyświetlone podstawowe możliwości systemu, wraz ze sposobem ich realizacji. W systemie, o którym mówimy, użytkownik ma do dyspozycji trzy różne menu. W momencie uruchomienia systemu zgłasza się pierwsze, które pozwala na wyświetlanie skrótowych informacji o piętnastu wybranych artykułach naraz lub pełnych informacji o wybranym artykule (rys. 1). Drugie menu służy do wyszukiwania informacji i wyświetlania na różne sposoby informacji znalezionych (rys. 2). Trzecie menu służy do modyfikowania zawartości bazy danych — a więc wprowadzania, poprawiania i usuwania opisów artykułów. Z każdego menu dostępna jest dyrektywa HELP, która informuje o działaniu poszczególnych dyrektyw (rys. 3).





Rys. 2. Za pomocą dyrektywy „a" zostały wyświetlone skrótowe informacje o wszystkich artykułach napisanych przez prof. A. Hulanickiego.



Rys. 3. W każdym menu można uzyskać dokładne informacje o tym, w jaki sposób korzystać z systemu. Wystarczy nacisnąć literę *h* — jak HELP, czyli: pomocy!

Jak widać, nawet tak pozornie proste zadanie wymaga dość złożonego oprogramowania. Napisanie go zajęło dwóm osobom kilka tygodni, a trwałoby to bez porównania dłużej, gdybyśmy nie korzystali z pomocy mechanizmów dostarczanych przez dBASE. System składa się z 37 programów zajmujących łącznie około 40 kB tekstu.

dBASE III to jednak coś więcej niż tylko mechanizmy tworzenia i przeszukiwania baz danych. Jest to tzw. pakiet zintegrowany. Idea takiego pakietu polega na dostarczeniu w jednym programie kompletu narzędzi potrzebnych do pisania i uruchamiania programów. Mamy więc bardzo wygodny (tzw. *full-screen*, operujący na całym ekranie) procesor tekstów, pozwalający na pisanie i modyfikowanie programów, które mogą być natychmiast wykonane przez interpreter. A jednocześnie mamy do dyspozycji dyrektywy systemu operacyjnego, które pozwalają np. na kopiowanie plików albo ich usuwanie. Język programowania dBASE III jest łatwy do opanowania i dość wygodny w użyciu, choć może razić jego „gadatliwość". Na szczególną uwagę zasługuje prostota operacji wykonywanych na ekranie — a zwłaszcza czytania i pisania w dowolnym punkcie. Język programowania zawiera ponadto niesłychanie silny mechanizm makrodefinicji, dzięki któremu można dynamicznie tworzyć zmienne, przechowywać nazwy zmiennych na innych zmiennych albo dynamicznie, w trakcie działania programu, tworzyć tekst wyrażenia logicznego, według którego następnie będziemy wyszukiwać informacje.

Warto dodać, że system dBASE III należy do najdroższej sprzedawanych pakietów. Jest też zabezpieczony przed „piractwem": próba pracy z oprogramowaniem skopiowanym kwitowana jest komunikatem „nieautoryzowana kopia" i uszkodzeniem kopii oraz okolicznych danych.