ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1992
Seria: INFORMATYKA z. 16 Nr kol. 1140

Jacek PIEC

SYSTEM WYSZUKIWANIA DANYCH Z PLIKÓW dBASE

Streszczenie. Przedstawiony system programowy stanowi próbę dostarczenia użytkownikowi bazy danych mechanizmów umożliwiających wygodne zadawanie nietypowych pytań do bazy danych opartych na rachunku relacji. Skoncentrowano się na uzyskaniu efektywności porównywalnej z systemami dedykowanymi.

Summary. The presented system is an approach to provide a database user with means of convenient quering the database basing on relational calculus. The aim was to obtain the officiency comparable to that of the systems designed for a particular user.

Резюме. В статье представлена система программирования использующая реляционное исчисление, которая дает возможность формулирования нетипичных вопросов к базе данных.

Celem pracy jest prezentacja systemu wyszukiwania danych z plików dBASE. Głównym elementem systemu jest preprocesor języka zapytań opartego na rachunku relacji. Efektem pracy programu preprocesora jest program w języku opartym na algebrze relacji (przyjęto język systemu dBASE), przy użyciu którego realizowane jest zadanie wyszukiwania wyrażone na podstawie rachunku relacji.

Przedstawiono algorytm redukcji grafu reprezentującego zadanie wyszukiwania przedstawione w języku opartym na algebrze relacji. Redukcja ta prowadzi do określenia ciągu operacji algebry relacji służących do otrzymania relacji wynikowej.

Przedstawiono również algorytm realizacji operacji równołączenia przy użyciu strategii typu "sortuj i połącz" oraz omówiono efektywność programu realizującego równołączenie przy użyciu tego algorytmu w porównaniu ze zleceniami służącymi do realizacji łączenia w systemie dBASE.

1. WPROWADZENIE

Główną korzyścią wynikającą z koncepcji formułowania zapytań na podstawie rachunku relacji jest fakt, że w zadaniu wyszukiwania określa się jedynie relacje (zbiory z danymi) niezbędne do wyznaczenia odpowiedzi oraz warunki definiujące poszukiwane dane, nie precyzując sekwencji operacji niezbędnych do otrzymania wyniku. Jest to znaczne ułatwienie dla użytkownika - szczególnie widoczne w przypadku operowania danymi zawartymi w wielu relacjach.

Użytkownicy systemu dBASE korzystają zwykle ze specjalnie przygotowanych programów wyszukiwania, które przewidują możliwość odpowiedzi wyłącznie na typowe zapytania, z góry określone przez danego użytkownika. Prezentowany program jest próbą dostarczenia mechanizmów wyszukiwania o bardziej uniwersalnym charakterze. Jednocześnie sposób realizacji operacji równołączenia za pomocą specjalnie w tym celu stworzonego programu zapewnia efektywność wyszukiwania porównywalną z systemami projektowanymi dla konkretnego użytkownika.

Teoria relacyjnych baz danych precyzyjnie definiuje podstawowe pojęcia, takie jak relacja, jej schemat, krotka, atrybut, dziedzina oraz znaczenie najważniejszych operatorów relacyjnych, tj. selekcji (σ), projekcji (π), iloczynu kartezjańskiego (x) i łączenia.

Tu przypominamy tylko, że relację można poglądowo interpretować jako tablicę, w której kolumny identyfikowane są nazwami atrybutów, wiersze zaś utożsamiane są z krotkami. Tablica taka jest z kolei naturalną ilustracją pliku, w którym nazwy pól odpowiadają nazwom kolumn, zaś rekordy - poszczególnym wierszom. Zbiór nazw atrybutów (kolumn) nazywa się schematem relacji.

Operacja selekcji polega na wyborze tych krotek (wierszy tablicy), które spełniają określony warunek. W wyniku projekcji otrzymujemy relację zawierającą niektóre spośród atrybutów (kolumn). Relacja powstała w wyniku operacji łączenia dwóch relacji zawiera krotki powstałe z takich par krotek relacji będących argumentami łączenia, które spełniają pewien warunek. W przypadku gdy warunek ten wyraża równość pewnych par atrybutów łączonych relacji, mówimy o operacji równołączenia.

2. ALGORYTM WYZNACZANIA RELACJI WYNIKOWEJ

Załóżmy, że poszukiwane przez nas dane zawarte są w relacjach $r_1 \ldots r_m$ i muszą spełniać jakiś warunek F. W przypadku gdyby dla rozwiązania zadania wyszukiwania należało użyć kilku kopii jednej relacji, będziemy te kopie traktować tak, jak kilka odrębnych relacji. Wynik umieścimy w relacji r o schemacie R.

Przyjmijmy też, że zadanie można rozwiązać przy użyciu operatorów selekcji, projekcji i łączenia (tzn. że zadanie należy do klasy select project-join).

Ogólne wyrażenie określające wynik wyszukiwania ma postać [1]

$$r = \Pi_R (\Sigma_F (r_1 x \ldots x r_m))$$

Założymy, że narzucony warunek stanowi koniunkcję (w dalszej części oznaczaną przez znak $\wedge$) warunków prostych określonych przy użyciu nie więcej niż 2 atrybutów. Każdy z atrybutów skojarzony jest w tekście warunku z pewną relacją. Mamy zatem

$$F = \wedge_{i=1}^{y} F_i \wedge \wedge_{j=1}^{z} U_j ,$$

przy czym każdy z warunków F_i określono przy użyciu atrybutów skojarzonych z dokładnie dwiema relacjami, każdy z warunków U_j określony jest przy użyciu atrybutów związanych z dokładnie jedną relacją, zaś y i z oznaczają odpowiednio ilość warunków F_i oraz U_j.

Relacja wynikowa równa będzie

$$r = \Pi_R (\Sigma \wedge_{i=1}^{y} F_i \wedge \wedge_{j=1}^{z} U_j (r_1 x \ldots x r_m)) \qquad (I)$$

Pytanie nasze możemy przedstawić w postaci grafu, w którym:

- każdej spośród relacji $r_1 \ldots r_m$ odpowiada wierzchołek,
- każdemu z warunków F_i odpowiada krawędź grafu, która łączy wierzchołki relacji związanych tym warunkiem,

- każdemu z wierzchołków przypisane są również (jeżeli były określone) odpowiednie warunki U_j, które stanowią warunki selekcji dla odpowiadającej im relacji.

Rozpatrzmy przykładową bazę danych pewnej firmy, która prowadzi sprzedaż owoców (rys.1). Baza zawiera cztery relacje:

DOSTAWCY - zawierającą dane o dostawcach owoców,
OWOCE - zawierającą dane o dostarczanych owocach,
ZAKUPY - zawierającą dane o transakcjach zakupu owoców,
SPRZEDAŻ - zawierającą dane o cenach sprzedawanych owoców.

DOSTAWCY

NRD	DOSTAWCA	SIEDZIBA

OWOCE

NRO	OWOC	ODMIANA

ZAKUPY

NRD	NRO	ILOŚĆ	CENA_DOST	DATA_DOST

SPRZEDAŻ

MIASTO	NRO	CENA_ZBYT	DATA_ZBYT

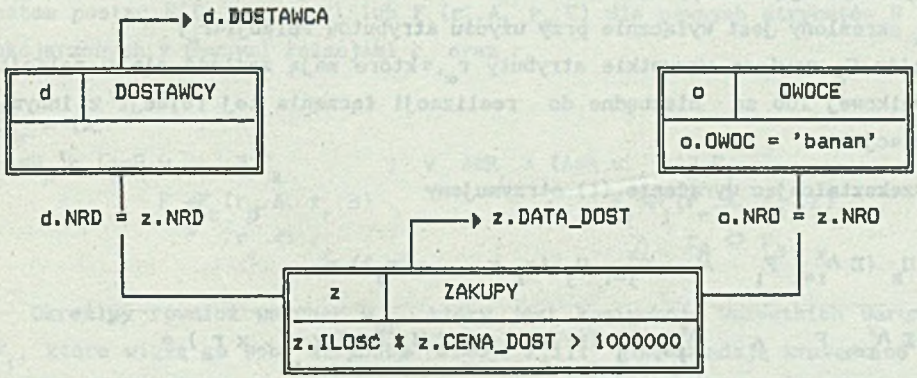
Rys. 1. Struktura przkładowej bazy danych
Fig. 1. Sample database structure

NRD i NRO oznaczają odpowiednio numer dostawcy i numer owocu, zaś znaczenie pozostałych nazw atrybutów jest oczywiste.

Przykład I

Rys. 2 przedstawia graf reprezentujący pytanie o daty i nazwy dostawców, dla dostaw bananów, których wartość przekracza 1 000 000 zł.

Procesowi wyszukiwania odpowiadać będzie przekształcenie grafu (jego redukcja) aż do momentu, gdy będzie on zawierał tylko jeden wierzchołek odpowiadający relacji wynikowej. Optymalizacja tego procesu oparta jest na strategii takiego przekształcenia grafu i wyrażenia (I), by operacje selekcji i projekcji były wykonywane jak najwcześniej i w miarę możliwości jednocześnie, zaś łączenie w kolejności optmalnej ze względu na liczność użytych relacji.



Rys. 2. Graf reprezentujący pytanie z przykładu I
Fig. 2. Query graph for Example I

Litery "o", "d" i "z" oznaczają tutaj lokalne kody relacji. Ich rola zostanie omówiona w dalszej części.

Spróbujmy teraz nieco precyzyjniej opisać operacje wykonywane podczas redukcji grafu wykorzystując operatory algebry relacji SELEKCJA-PROJEKCJA oraz ŁĄCZENIE-PROJEKCJA.

2.1. Operator SELEKCJA-PROJEKCJA

Dla relacji r_α o schemacie R_α, dla której określono warunki selekcji, wyznaczmy warunek U_α, stanowiący koniunkcję warunków selekcji. Wyznaczmy również zbiór C_α zawierający te atrybuty relacji r_α, które mają znaleźć się w relacji wynikowej bądź też były użyte przy formułowaniu warunków F_i, wiążących relację r_α z innymi relacjami. Każdy z tych warunków ma zatem postać

F_1 $(r_\alpha.A, r_r.B)$, dla pewnego atrybutu B skojarzonego z pewną relacją r_r.

$$U_\alpha = \bigwedge_{U_i = U_i(r_\alpha)} U_i$$

$$C_\alpha = \{A : A \in R_\alpha \wedge (A \in R \vee \exists_{F_i = F_i(r_\alpha.A,\ r_r.B)} F_i)\}$$

Wykorzystując fakty, że

- U_α określony jest wyłącznie przy użyciu atrybutów relacji r_α,
- zbiór C_α zawiera wszystkie atrybuty r_α, które mają znaleźć się w relacji wynikowej lub są niezbędne do realizacji łączenia tej relacji z innymi relacjami

i przekształcając wyrażenie (I) otrzymujemy

$$r = \Pi_R (\Sigma \bigwedge_{i=1}^{y} F_i \wedge \bigwedge_{j=1}^{z} U_j (r_1 \times \ldots \times r_m)) =$$

$$\Pi_R (\Sigma \bigwedge_{i=1}^{y} F_i \wedge \bigwedge_{\substack{j=1 \\ U_j <> U_j(r_\alpha)}}^{z} U_j (r_1 \times \ldots \times \Sigma U_\alpha (r_\alpha) \times \ldots \times r_m) =$$

$$\Pi_R (\Sigma \bigwedge_{i=1}^{y} F_i \wedge \bigwedge_{\substack{j=1 \\ U_j <> U_j(r_\alpha)}}^{z} U_j (r_1 \times \ldots \times \Pi c_\alpha (\Sigma u_\alpha (r_\alpha)) \times \ldots r_m)$$

Oznaczając $r_\alpha' = \Pi c_\alpha (\Sigma u_\alpha (r_\alpha))$ przekształcamy wyrażenie (I) do następującej postaci

$$r = \Pi_R (\Sigma \bigwedge_{i=1}^{y} F_i \wedge \bigwedge_{\substack{j=1 \\ U_j <> U_j(r_\alpha)}}^{z} U_j (r_1 \times \ldots \times r_\alpha' \times \ldots \times r_m))$$

Dla grafu reprezentującego pytanie operator SELEKCJA-PROJEKCJA (r_α) zdefiniowany jest następująco:

1. Wyznacz $r_\alpha' = \Pi c_\alpha (\Sigma u_\alpha (r_\alpha))$
2. Wierzchołek grafu G, odpowiadający dotąd relacji r_α przyporządkuj relacji r_α'.

2.2. Operator ŁĄCZENIE-PROJEKCJA

Spośród relacji (wierzchołków grafu), dla których warunek selekcji nie jest określony (tzn. nie został określony przez użytkownika bądź też dana relacja jest wynikiem wykonania operacji SELEKCJA-PROJEKCJA lub ŁĄCZENIE-PROJEKCJA) wybierzmy parę relacji r_β i r_r związanych pewnym warunkiem (krawędzią) F_i.

Utwórzmy zbiór $D_{\beta r}$, zawierający wszystkie te atrybuty relacji r_β i r_r, które mają znaleźć się w relacji wynikowej bądź też służą do opisania warunków wiążących relacje r_β i r_r z innymi relacjami. Każdy z tych warunków ma zatem postać $F_k(r_\beta.A, r_r.B)$ lub $F_j(r_r.A, r_\delta.E)$ dla pewnych atrybutów B i E skojarzonych z pewnymi relacjami r_r oraz r_δ

$$D_{\beta r} = \{A: A \in R_\beta \wedge (A \in R \vee \exists_{\substack{F_k=F_k(r_\beta.A,\ r_r.B) \\ \wedge\ r_r <> r_r}} F_k) \vee A \in R_r \wedge (A \in R \vee \exists_{\substack{F_j=F_j(r_r.A,\ r_\delta.E) \\ \wedge\ r_\beta <> r_\delta}} F_j)$$

Określmy również warunek $W_{\beta r}$, który jest koniunkcją wszystkich warunków F_i, które wiążą ze sobą atrybuty relacji r_β i r_r (odpowiadają krawędziom łączącym wierzchołki odpowiadające relacjom r_β i r_r), tzn.

$$W_{\beta r} = \bigwedge_{F_i=F_i(r_\beta, r_r)} F_i$$

Bez szkody dla ogólności rozważań (użyte relacje zawsze możemy przenumerować) możemy przyjąć, że wybrane relacje r_β i r_r to relacje r_1 i r_2.

Wykorzystując fakty, że

- warunek W_{12} określony jest wyłącznie przy użyciu atrybutów relacji r_1 i r_2,
- zbiór D_{12} zawiera wszystkie atrybuty relacji r_1 i r_2 "eksportowane" do relacji wynikowej lub niezbędne do realizacji łączenia relacji r_1 i r_2 z innymi relacjami

i przekształcając wyrażenie (I) otrzymujemy

$$r = \Pi_R (\Sigma \bigwedge_{i=1}^{y} F_i \quad \wedge \quad \bigwedge_{j=1}^{z} U_j (r_1 \times \ldots \times r_m)) =$$

$$\Pi_R (\Sigma_{F_i <> F_i (r_1, r_2)} \bigwedge_{i=1}^{y} F_i \quad \wedge \quad \bigwedge_{j=1}^{z} U_j (\Sigma w_{12} (r_1 \times r_2 \times \ldots \times r_m))) =$$

$$\Pi_R (\Sigma_{F_i <> F_i (r_1, r_2)} \bigwedge_{i=1}^{y} F_i \quad \wedge \quad \bigwedge_{j=1}^{z} U_j (\Sigma w_{12} (r_1 \times r_2) \times \ldots \times r_m)) =$$

$$\Pi_R (\Sigma_{F_i <> F_i (r_1, r_2)} \bigwedge_{i=1}^{y} F_i \quad \wedge \quad \bigwedge_{j=1}^{z} U_j (\Pi_{D12} (\Sigma w_{12} (r_1 \times r_2)) \times \ldots \times r_m)$$

Oznaczając

$$r^{12} = \Pi_{D12} (\Sigma w_{12} (r_1 \times r_2))$$

otrzymujemy

$$r = \Pi_R (\Sigma_{F_i <> F_i (r_1, r_2)} \bigwedge_{i=1}^{y} F_i \quad \wedge \quad \bigwedge_{j=1}^{z} U_j (r^{12} \times r_3 \times \ldots \times r_m))$$

Dla grafu reprezentującego pytanie operator ŁĄCZENIE-PROJEKCJA (r_β, r_r) zdefiniowany jest następująco:

1. Wyznacz $r^{\beta r} = \Pi_{D\beta r} (\Sigma w_{\beta r} (r_\beta \times r_r))$.
2. Usuń z grafu G wierzchołki odpowiadające relacjom r_β i r_r.
3. Usuń krawędzie grafu, które odpowiadają warunkom wiążącym relacje r_β i r_r.
4. Utwórz wierzchołek odpowiadający relacji $r^{\beta r}$.
5. Krawędzie grafu, które łączyły dotąd inne relacje z relacjami r_β i r_r, połącz z relacją $r^{\beta r}$.

Wyrażenie (I) po wykonaniu operacji ŁĄCZENIE-PROJEKCJA (r_β, r_r) przyjmie postać

$$r = \Pi_R \left(\Sigma \bigwedge_{i=1}^{y} F_i \wedge \bigwedge_{j=1}^{z} U_j \right.$$

$$F_i <> F_i(r_\beta, r_r)$$

$$\left. (r^{*\beta r} \times r_1 \times \ldots \times r_{\beta-1} \times r_{\beta+1} \times \ldots \times r_{r-1} \times r_{r+1} \times \ldots \times r_m)\right))$$

2.3. Algorytm wyznaczania relacji wynikowej

Zdefiniowane operatory algebry relacji zostały zaimplementowane w języku dBASE w następujący sposób:

SELEKCJA-PROJEKCJA (r) = Π_C (Σ_U (r))

- przy użyciu zleceń COPY lub DISPLAY, w zależności od tego; czy wynik operacji ma zostać zapamiętany w zbiorze, czy też wyświetlony na ekranie,

ŁĄCZENIE-PROJEKCJA (r_beta, r_gamma) = Π_D (Σ_W (r_beta x r_gamma))

- przy użyciu zlecenia JOIN lub w przypadku równołączenia, przy użyciu opracowanego w tym celu programu EQJOIN.

Jak pokazano przy definiowaniu tych operacji, powodują one uzyskanie grafu reprezentującego pytanie równoważne wyjściowemu, umożliwiając równocześnie jego uproszczenie. Stąd cykliczne powtarzanie operacji ŁĄCZENIE-PROJEKCJA poprzedzonej ewentualnymi operacjami SELEKCJA-PROJEKCJA prowadzi do uzyskania grafu zawierającego dokładnie jeden wierzchołek odpowiadający relacji wynikowej. Strategię tę można przedstawić w postaci następującego algorytmu:

początek

jeżeli graf zawiera dokładnie jeden wierzchołek, **to**

SELEKCJA-PROJEKCJA (r_1);

w przeciwnym razie

dopóki graf zawiera więcej niż 1 wierzchołek **wykonuj**

początek

Spośród par wierzchołków połączonych pewną krawędzią wybierz parę β i r;

jeżeli dla relacji r_β określono warunek selekcji **to**

SELEKCJA-PROJEKCJA (r_β);

jeżeli dla relacji r_r określono warunek selekcji **to**

SELEKCJA-PROJEKCJA (r_r);
ŁĄCZENIE-PROJEKCJA (r_p, r_r);
koniec

komentarz jedyny wierzchołek grafu reprezentuje relację wynikową
koniec.

Przy wyborze pary relacji do wykonania operacji ŁĄCZENIE-PROJEKCJA powinno obowiązywać kryterium minimalnej liczności relacji powstałej w wyniku tej operacji. Odpowiada to idei przetwarzania na wstępie "małych" relacji, co zmniejsza prawdopodobieństwo szybkiego wzrostu rozmiarów relacji roboczych tworzonych w wyniku łączeń. W prezentowanym programie preferowane są relacje z określonym warunkiem selekcji bądź biorące udział w operacji równołączenia.

Każdemu krokowi redukcji grafu towarzyszy generacja fragmentu programu wynikowego w języku dBASE. Następnie sterowanie przekazywane jest do systemu dBASE, który wykonuje wygenerowany program (COMMANDS.$$$), co prowadzi do umieszczenia w relacji wynikowej (bądź na ekranie monitora) danych spełniających warunki wyszukiwania. Przykład grafu, jego redukcji oraz wygenerowany program przedstawiony zostanie w dalszej części.

3. SFORMUŁOWANIE ZADANIA WYSZUKIWANIA

Formułowanie pytania przebiega konwersacyjnie według następujących reguł:

- W pierwszym etapie określa się potrzebne relacje (pliki). W celu umożliwienia formułowania pytań wymagających wielokrotnego użycia tej samej relacji oraz dla wygody przy zapisie warunków wyszukiwania definiuje się dodatkowo kody wykorzystywanych relacji.
- Następnie określa się nazwę relacji (pliku), do której wprowadzone zostaną wyszukane wartości danych i jej strukturę używając kodów relacji i nazw atrybutów. Możliwe jest również wyświetlenie wyniku wyszukiwania bezpośrednio na ekranie monitora (bez zapamiętywania w pliku).
- Kolejnym krokiem jest określenie warunków wyszukiwania. Przyjęta forma dialogu wymusza koniunkcję warunków prostych. Warunek prosty może wiązać nazwę atrybutu i stałą, bądź też dwie nazwy atrybutów (powiązanie dwóch

relacji). Reguły syntaktyczne warunku prostego dopuszczają dowolne wyrażenie logiczne zbudowane zgodnie z regułami języka dBASE, przy czym

- nazwy atrybutów należy poprzedzić lokalnym kodem relacji zakończonym znakiem '.' ,
- w przypadku gdy graf reprezentujący pytanie nie jest spójny, tzn. użytkownik nie wprowadził wystarczającej ilości warunków, aby powiązać ze sobą wszystkie używane relacje, sygnalizowany jest błąd.

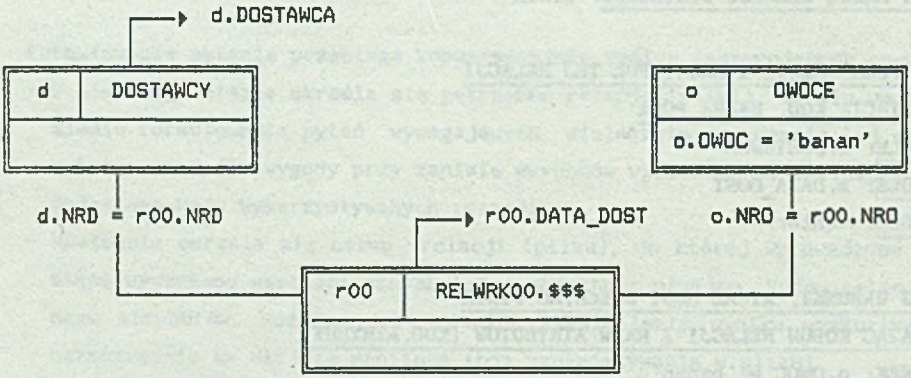
A oto jak wygląda formułowanie pytania z przykładu (I) (podkreślenia wyróżniają tekst wypisywany na ekranie przez program)

```
PODAJ NAZWY RELACJI I LOKALNE KODY DLA TYCH NAZW
RELACJA: DOSTAWCY
    KOD: d
RELACJA: OWOCE
    KOD: o
RELACJA: ZAKUPY
    KOD: z
RELACJA: <enter>

PODAJ NAZWĘ RELACJI WYNIKOWEJ: WYNIK

ORAZ POCHODZENIE I NAZWY PÓL TEJ RELACJI
W POSTACI: KOD. NAZWA POLA
   POLE: d.DOSTAWCA
   POLE: z.DATA_DOST
   POLE: <enter>

PODAJ WARUNKI, KTÓRE MUSI SPEŁNIAĆ WYNIK
UŻYWAJĄC KODÓW RELACJI I NAZW ATRYBUTÓW (KOD.ATRYBUT)
WARUNEK: o.OWOC = 'banan'
ORAZ: z.ILOŚĆ * z.CENA_DOST > 1000000
ORAZ: d.NRD = z.NRD
ORAZ: z.NRD = o.NRD
ORAZ: <enter>
```

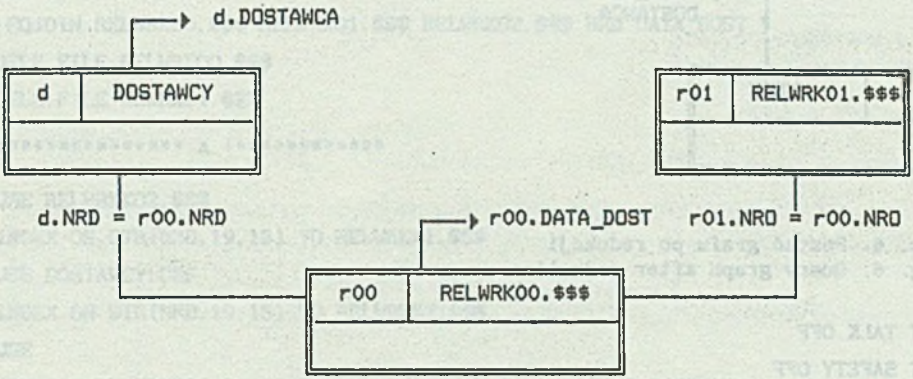
```
WYKONAĆ ?    T.Y.<enter> - TAK        N.F. - NIE
? T
OK
```

Poniżej przedstawiono postać grafu reprezentującego pytanie po wykonaniu kolejnych kroków redukcji grafu oraz treść programu COMMANDS.$$$ realizującego wyszukiwanie. W treści tego programu wydzielono fragmenty odpowiadające kolejnym etapom redukcji

```
(1)

RELWRKOO.$$$  <-SELEKCJA-PROJEKCJA (ZAKUPY)
  C = { z.DATA_DOST, z.NRD, z.NRD }
  U = "z.ILOŚĆ * z.CENA_DOST > 1000000"

(2)

RELWRKO1.$$$   <-SELEKCJA-PROJEKCJA ( OWOCE )
  C = {o.NRD}
  U = "o.OWOC = 'banan'"
```



Rys. 3. Postać grafu po pierwszym etapie redukcji
Fig. 3. Query graph after first reduction stage



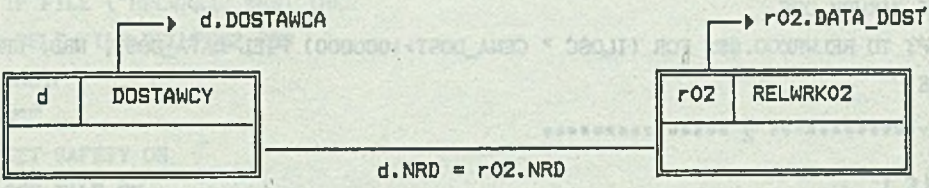
Rys. 4. Postać grafu po drugim etapie redukcji
Fig. 4. Query graph after second reduction stage

(3)

RELWRK02.$$$ <-ŁĄCZENIE-PROJEKCJA (r01, r02)

D = { r00.DATA_DOST, r00.NRD }

W = "r01.NRD = r00.NRD"



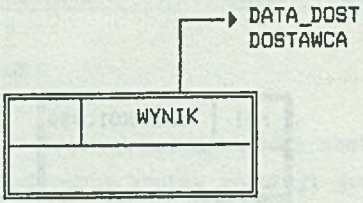
Rys. 5. Postać grafu po trzecim etapie redukcji
Fig. 5. Query graph after third reduction stage

(4)

WYNIK <-ŁĄCZENIE-PROJEKCJA (r01, r02)

D = { r01.DOSTAWCA, r02.DATA_DOST }

W = "r01.NRD = r02.NRD"



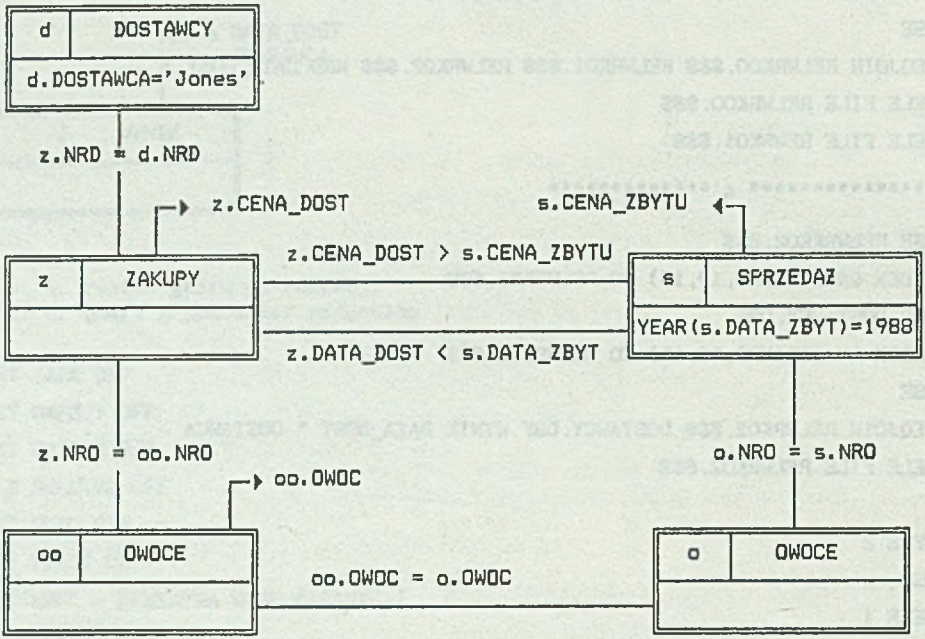
Rys. 6. Postać grafu po redukcji
Fig. 6. Query graph after reduction

```
SET TALK OFF
SET SAFETY OFF
SET UNIQUE OFF
SET DELETED OFF
SET ECHO OFF
SET EXACT ON
? ["ESC" - PRZERYWA WYSZUKIWANIE ]
? [CZEKAJ]
*************** 1 ***************
SELE 1
USE ZAKUPY.DBF
COPY TO RELWRKOO.$$$ FOR (ILOŚĆ * CENA_DOST>1000000) FIEL DATA_DOST, NRD, NRO
USE
*************** 2 ***************
SELE 1
USE OWOCE.DBF
COPY TO RELWRKO1.$$$ FOR (OWOC = 'banan') FIEL NRO
USE
*************** 3 ***************
USE RELWRKOO.$$$
INDEX ON STR (NRO,19,15) TO RELWRK%1.$$$
USE RELWRKO1.$$$
```

```
INDEX ON STR (NRD, 19,15) TO RELWRK%2.$$$
USE
!EQJOIN RELWRKOO.$$$ RELWRKO1.$$$ RELWRKO2.$$$ NRD DATA_DOST *
DELE FILE RELWRKOO.$$$
DELE FILE REWRKO1.$$$
*************** 4 *************
USE RELWRKKO2.$$$
INDEX ON STR(NRD,19,15) TO RELWRK%1.$$$
USE DOSTAWCY.DBF
INDEX ON STR(NRD,19,15) TO RELWRK%2.$$$
USE
!EQJOIN RELWRKO2.$$$ DOSTAWCY.DBF WYNIK DATA_DOST * DOSTAWCA
DELE FILE RELWRKO2.$$$

SELE 2
USE
SELE 1
IF FILE ("RELWRK%1.$$$") THEN
 DELE FILE RELWRK%1.$$$
ENDIF
IF FILE ("RELWRK%2.$$$) THEN
 DELE FILE RELWRK%2.$$$
ENDIF
USE
SET SAFETY ON
SET TALK ON
RETURN
```

Przykład II

Sprawdźmy, czy zdarzyło się w pewnej firmie sprzedawać w 1988 roku owoce po cenie niższej niż cena, jaką firma ta płaciła uprzednio dostawcy o nazwisku Jones za owoce tego samego gatunku (niekoniecznie tej samej jakości).



Rys. 7. Graf reprezentujący pytanie z przykładu II
Fig. 7. Query graph for Example II

Prezentowany przykład uzasadnia wprowadzenie kodów relacji, które identyfikują atrybuty o tej samej nazwie, a pochodzące z różnych relacji i użyte w różnym znaczeniu, jak np. oo.OWOC oznaczający owoc kupowany oraz o.OWOC - owoc sprzedawany.

W wyjątkowych przypadkach skomplikowanych pytań może się zdarzyć, że w nowo utworzonej relacji roboczej na jednym z etapów wyszukiwania musiałyby się znaleźć dwa atrybuty o tej samej nazwie, ale oznaczające w danym pytaniu pojęciowo dwie różne rzeczy (np. w powyższym przykładzie o.nro i oo.nro). Ponieważ naruszałoby to oczywiste ograniczenia systemu dBASE, sygnalizowany jest błąd. Zwykle prawidłową odpowiedź na pytanie można uzyskać zmieniając kolejność wprowadzanych warunków wyszukiwania.

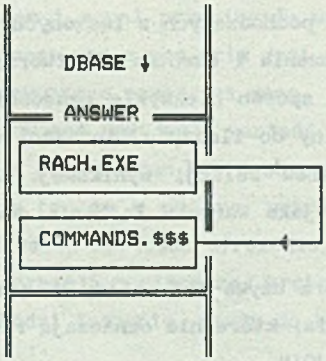
W powstającej, nowej wersji opisywanego systemu mankament ten zostanie usunięty poprzez automatczne przemianowanie atrybutów w kolejnych etapach wyszukiwania.

4. STRUKTURA SYSTEMU WYSZUKIWANIA

Do uruchomienia systemu niezbędna jest obecność następujących plików:

- ANSWER.PRG (plik zawierający program w języku dBASE kontrolujący przepływ sterowania w systemie),
- RACH.EXE (plik zawierający program konwersacji z użytkownikiem i optymalizacji procesu wyszukiwania),
- EQJOIN.EXE (plik zawierający program realizujący 'szybkie łączenie' relacji),
- pliki systemu dBASE,
- pliki z danymi.

Przyjęto następującą koncepcję przepływu sterowania w systemie wyszukiwania danych (rys.8.).



Rys. 8. Przepływ sterowania w systemie wyyszukiwania danych
Fig. 8. Control flowchart for query answering system

ANSWER - plik zleceń systemu dBASE przekazujący sterowanie do programu procesora RACH, a następnie do tworzonego przez niego pliku zleceń COMMANDS.$$$.

RACH - program dialogu z użytkownikiem i optymalizacji procesu wyszukiwania, generujący plik zleceń COMMANDS.$$$

COMMANDS.$$$ - plik zleceń systemu dBASE realizujący wyszukiwanie danych

5. REALIZACJA ŁĄCZENIA

Na koniec poświęcimy nieco uwagi realizacji operacji łączenia jako operacji decydującej o czasie wyszukiwania danych zawartych w wielu relacjach.

W systemach projektowanych dla konkretnego użytkownika operacja ta realizowana jest zwykle poprzez tworzenie pętli programowych, w których dostęp do kolejnych krotek zorganizowany jest poprzez indeksowanie. Jest to sposób efektywny, ale kłopotliwy dla programisty i co ważniejsze, ściśle uzależniony od struktury relacji oraz postaci typowych zapytań w bazie danych, na której operuje dany użytkownik.

dBASE pozwala również na implementację łączenia przy użyciu zlecenia JOIN. Zlecenie to dla każdej pary krotek pochodzących z łączonych relacji, sprawdza, czy spełniają one warunek łączenia i ewentualnie tworzy się z nich nową krotkę relacji wynikowej. Jest to sposób niezwykle czasochłonny, czas przetwarzania jest bowiem proporcjonalny do iloczynu liczności użytych relacji i praktycznie nie zależy od rozmiarów relacji wynikowej. Jego zaletą jest jednak prostota oraz dopuszczenie jako warunku łączenia dowolnego wyrażenia logicznego.

Prezentowany program preprocesora używa tej metody w wyjątkowych przypadkach, gdy określono warunki łączenia, które nie oznaczają równołączenia. Zwykle jednak używany jest program EQJOIN.

Użyta w nim metoda (oparta na strategii typu "sortuj i łącz") polega na wykorzystaniu uporządkowania relacji za pomocą zbiorów indeksowych, dzięki czemu krotki o równych wartościach wyrażenia indeksującego można wyszukać przy jednokrotnym przeglądnięciu tych zbiorów. Umożliwia to realizację

równołączenia w czasie zależnym od sumy liczności łączonych relacji oraz od liczności relacji wynikowej. Czas równołączenia realizowanego tą metodą jest zwykle wielokrotnie krótszy od czasu wykonania równoważnego mu zlecenia JOIN, zaś efektywność całego wyszukiwania może być nawet lepsza od systemów projektowanych dla konkretnego użytkownika.

Poniżej przedstawiamy algorytm wykorzystujący zbiory indeksowe dla realizacji operacji równołączenia.

Założymy, że dla łączonych relacji utworzono zbiory indeksowe. Równość wyrażeń, według których indeksowano relacje, stanowi warunek łączenia. Każdy zbiór indeksowy zawiera dla danej relacji ciąg par <numer krotki> <wartość indeksu> uporządkowanych według rosnących wartości wyrażenia, według którego indeksowaliśmy daną relację.

Zmienna pos_prim wskazuje na pewną parę w zbiorze indeksowym relacji pierwszej, zaś zmienna pos_seco to samo dla relacji drugiej.

Procedura appe_join powoduje dopisanie do reakcji wynikowej nowej krotki. Jest ona tworzona z takich krotek relacji pierwszej i relacji drugiej, których numery wskazują pos_prim i pos_seco. Warunkiem dołączenia nowej krotki do relacji wynikowej jest równość wartości indeksów wskazywanych przez pos_prim i pos_seco.

Funkcje greater i equal przybierają watość true w przypadku, gdy wartość indeksu wskazywanego przez pos_prim jest odpowiednio większa lub równa wartości indeksu wskazywanego przez pos_seco.

Funkcja not_any_end przyjmuje wartość false w przypadku, gdy pos_prim lub pos_seco wskazują poza końcem któregoś z plików indeksowych.

Ze względu na to, że w ogólnym przypadku w relacji może wystąpić kilka krotek o takiej samej wartości wyrażenia indeksującego, istnieje konieczność przechowania informacji o pierwszej krotce z relacji drugiej o danym kluczu, dla którego zachodzi łączenie. Służy do tego zmienna prev_pos_seco.

```
program indexjoin;
begin
pos_seco:= 1;
pos_prim:= 1;
while not_any_end do
```

```
    begin
    while not equal and not_any_end do
       if greater then pos_seco:= pos_seco + 1
                  else pos_prim:= pos_prim + 1;
    if not_any_end then
      begin
      prev_pos_seco:= pos_seco;
      while not_any_end and equal do
        begin
        appe_join (pos_prim, pos_seco);
        pos_seco:= pos_seco + 1
        end;
      pos_seco:= prev_pos_seco;
      pos_prim:= pos_prim + 1
      end
    end;
end.
```

5.1. Program EQJOIN

Parametry: <relacja1> <relacja2> <wynik> <atrybuty1> * <atrybuty2>

<relacja1> i <relacja2>	to nazwy relacji, które zamierzamy łączyć,
<wynik>	oznacza nazwę relacji wynikowej,
<atrybuty1> i <atrybuty2>	to ciąg nazw atrybutów, które mają się znaleźć w relacji wynikowej i pochodzą odpowiednio z <relacji1> i <relacji2> - ciąg ten może być pusty.

znak * jest obowiązkowy i służy do oddzielenia grup atrybutów.

Zakłada się również, że istnieją zbiory indeksowe RELWRK%1.$$$ oraz RELWRK%2.$$$, które powstały w wyniku indeksowania <relacji1> i <relacji2> wg dowolnego wyrażenia typu znakowego.

Przykładowo, jeśli chcemy połączyć relacje ZAKUPY oraz OWOCE (przy założeniu, że numery owoców NRO są pięciocyfrowe) musimy:

1) utworzyć odpowiednie zbiory indeksowe za pomocą zleceń

```
USE ZAKUPY
INDEX ON STR(NRD,5,0) TO RELWRK%1.$$$
USE OWOCE
INDEX ON STR(NRD,5,0) TO RELWRK%2.$$$
```

2) użyć program EQJOIN

```
EQJOIN ZAKUPY.DBF OWOCE.DBF WYNIK.DBF DATA NRD * OWOC ODMIANA
```

Założyliśmy tutaj, że w relacji wynikowej WYNIK mają znaleźć się atrybuty DATA i NRD z relacji ZAKUPY oraz OWOC i ODMIANA z relacji OWOCE.

Odpowiedni program w języku dBASE mógłby, przy założeniu, że utworzono strukturę relacji WYN i NRO jest kluczem w relacji owoce, wyglądać następująco:

```
SELECT 3
USE WYN
SELECT 2
USE OWOCE INDEX OWOCE
SELECT 1
USE ZAKUPY
SET RELATION TO NRO INTO OWOCE
DO WHILE .NOT. EOF ()
    SELECT WYN
    APPEND BLAN
    REPLACE NRD     WITH ZAKUPY->NRD
    REPLACE DATA    WITH ZAKUPY->DATA
    REPLACE OWOC    WITH OWOCE->OWOC
    REPLACE ODMIANA WITH OWOCE->ODMIANA
    SELECT ZAKUPY
    SKIP
ENDDO
```

lub po prostu

```
SELECT 2
USE OWOCE ALIAS OW
SELECT 1
USE ZAKUPY
JOIN WITH OW TO WYNIK FOR NRO = OW->NRO FIELD DATA,NRD,OW->OWOC,OW->ODMIANA
```

Dla porównania podajemy przykładowe czasy realizacji łączenia (dla komputera klasy IBM AT - 12 Mhz) w trzech wariantach zakładając, że relacja OWOCE zawiera 200 krotek (rekordów), relacja ZAKUPY zawiera 800 krotek i każdy z owoców kupowany był średnio w 4 dostawach.

EQJOIN + tworzenie zbiorów indeksowych	- 15 s
program dBASE'a (bez czasu tworzenia zbioru indeksowego OWOCE.NDX	- 90 s
zlecenie JOIN	- 310 s

6. UWAGI KOŃCOWE

Opisany system wyszukiwania może być oparty także na systemie FoxBASE. W tym celu stworzona została odpowiednia wersja programu EQJOIN.

Aktualnie prowadzone są prace nad nową wersją systemu, która przewiduje między innymi:

- wygodniejszy sposób formułowania pytania oparty o system menu,
- usunięcie ograniczenia ilości atrbutów użytch przy formułowaniu warunków prostych,
- rozwiązanie problemu możliwości wystąpienia konfliktu nazw atrybutów, jak np. w przykładzie II z punktu 3,
- możliwość zapisywania tekstu zadania wyszukiwania w pliku oraz edycji tego tekstu.

Doświadczenia zebrane w trakcie użytkowania eksperymentalnej wersji systemu podczas zajęć dydaktycznych prowadzonych w Instytucie Informatyki Politechniki Śląskiej wskazują na jego szczególną przydatność dla użytkowników formułujących niestandardowe zadania wyszukiwania dla danych zgromadzonych przy użyciu systemu dBASE.

LITERATURA

1. J.D.Ullman: Systemy baz danych, WNT, Warszawa 1988.
2. C.J.Date: Wprowadzenie do baz danych, WNT, Warszawa 1981.
3. J.Martin: Organizacja baz danych, PWN, Warszawa 1983.

Recenzent: prof.dr hab.inż. Wojciech Cellary

Wpłynęło do redakcji: 3 kwietnia 1990 r.

QUERY ANSWERING SYSTEM FOR dBASE III DATA FILES

Abstract

The presented software system is an attempt to provide a database user with convenient tools of querying based on relational calculus.

The system may be operated either independently or as the extension to the existing query systems, in particular in the domain of non - standard queries. Due to its general character, it my be used with any existing database created under dBASE (FoxBASE, with no modifications required).

The main element of the system is the relational calculus - based query-language preprocessor (the language of dBASE / Fox Base systems has been accepted), which outputs the data confirming to the query expressed in terms of relational calculus. In the process of translation an algorithm of reduction of the graph representing the query is applied, the query being described using the relational algebra based language. After the reduction, a sequence of operations of the relational algebra is obtained which in turn make up the final relation.

The major idea of the presented work is to achieve the afficiency comparable to that of the systems designed for particular user. The algorithm of "sort and merge" strategy is presented, and the efficiency of the program using this strategy is compared with that of the dBASE commands applied to perform similar operations.