ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1994

Seria: INFORMATYKA z. 25 Nr kol. 1236

Stanisław KOZIELSKI

JĘZYKI ZAPYTAŃ RELACYJNYCH BAZ DANYCH A ROZPRASZANIE OBLICZEŃ W SIECI KOMPUTEROWEJ

Streszczenie. W pracy przedyskutowano własności języków zapytań relacyjnych baz danych pod kątem możliwości równoległej realizacji zadań wyszukiwania zapisanych w tych językach. Zwrócono uwagę na algebrę relacji jako poziom zapisu wygodny dla analizy procesu rozpraszania. Rozpatrzono możliwość rozpraszania obliczeń w systemie wyszukiwania wykorzystującym model relacji uniwersalnej.

RELATIONAL QUERY LANGUAGES AND COMPUTATION DISTRIBUTION IN THE COMPUTER NETWORK

Summary. Features of relational query languages are discussed in the paper in the context of possibility of data retrieving tasks parallel realization. Four levels of languages are considered. Relational algebra is indicated as a convenient level for such data retrieving tasks formulation. Possibility of computation distribution in the data retrieving system based on the universal relation model is considered.

Pracę wykonano w ramach grantu KBN nr 8 0269 91 01

DIE ABFRAGENSPRACHEN DER RELATIONALEN DATENBANKEN UND DIE VERTEILUNG DER BERECHNUNGEN IM COMPUTER-NETZWERK

Zusammenfassung. Im Artikel wurden die Eigenschaften der Abfragensprachen der relationalen Datenbanken im Hinblick auf die Möglichkeit der parallelen Ausführung der in diesen Sprachen formulierten Aussuchaufgaben erörtert. Auf die relationale Algebra wurde als aufs bequeme Formulierungsniveau für die Analyse des Verteilungsprozesses hingewiesen.
Die Möglichkeit der Verteilung der Berechnungen wurde für ein Aussuchsystem erörtert, das auf dem Modell der universalen Relation basiert.

1. Wstęp

W artykule rozważa się pewne zagadnienia dotyczących rozproszonej realizacji zadań wyszukiwania danych w sieci komputerowej.

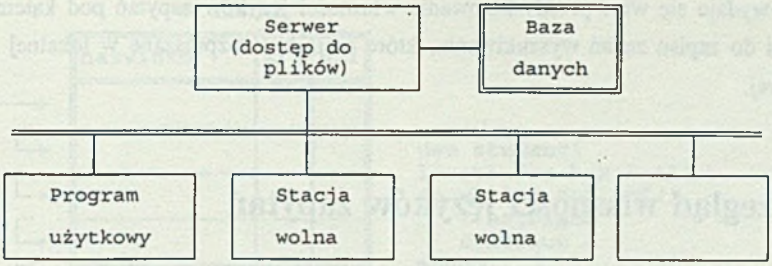
Celem pierwszej części pracy jest wskazanie powiązań między poziomem zapisu zadań wyszukiwania danych a łatwością rozpraszania obliczeń potrzebnych do realizacji wyszukiwania. Szczególną uwagę skupiono na możliwości rozpraszania obliczeń w systemie wyszukiwania opartym na modelu relacji uniwersalnej.

W drugiej części pracy omówiono dwa warianty sterowania współbieżną realizacją obliczeń rozproszonych.

2. Sformułowanie zadania

Rozważmy lokalną sieć komputerową zilustrowaną rys. 1. W sieci tej baza danych została zorganizowana na dysku serwera. Serwer pełni rolę zarządcy plików, tzn. jednym z jego zadań jest organizowanie dostępu do plików. W komputerach - stacjach sieci realizowane są programy użytkowe, przy czym zakładać będziemy, że w danej chwili są w sieci stacje nie obciążone żadnymi zadaniami.

Rozpatrzmy w bardzo ogólnej postaci program wyszukiwania danych (lub jego część) składający się z ciągu operacji $O_1, O_2, O_3, \ldots, O_n$. Zauważmy, że jeśli w sekwencji operacji O_i, O_{i+1} wykonanie operacji O_{i+1} nie zależy od rezultatów i zakończenia operacji O_i, to



Rys. 1. Ilustracja sieci komputerowej umożliwiającej równoległą realizację zadań wyszukiwania

Fig. 1. Outline view of computer network making parallel realization of data retrieving tasks possible

istnieje możliwość równoległej realizacji tych operacji. Jednym ze sposobów takiego wykonania mogłoby być umieszczenie programu realizującego operację O_{i+1} w wolnej w danej chwili stacji sieci. Oczywiście, przejście do wykonania kolejnej operacji O_{i+2} musiałoby być uwarunkowane sprawdzeniem zakończenia operacji O_{i+1}. Gdyby jednak wykonanie O_{i+2} nie zależało od rezultatów O_i, ani O_{i+1}, to również operacja O_{i+2} mogłaby być realizowana równolegle do O_i oraz O_{i+1} w trzeciej stacji sieci.

Rozwiązanie tak postawionego problemu będzie zależało od kilku elementów.

Pierwszym będzie niewątpliwie dostępność mechanizmów programowych umożliwiających przekazywanie komunikatów i parametrów do odległej stacji, zdalne uruchamianie programu w takiej stacji oraz synchronizowanie realizacji programów w kilku stacjach. Jeden ze sposobów rozwiązania tych problemów przedstawiono w pracy [13] (proponowana w tej pracy biblioteka obejmuje trzy procedury: wstępnej inicjalizacji systemu, zdalnego uruchamiania zadania oraz synchronizacji zadań).

Kolejnym problemem może być podatność programów na rozpraszanie. Zauważmy bowiem, że jeśli operacje O_i, O_{i+1} lub O_{i+2} będą operacjami bardzo prostymi, to nakład pracy na rozpraszanie obliczeń przewyższy korzyści z równoległej ich realizacji.

Jeszcze innym zagadnieniem będzie sposób wyodrębniania operacji O_1, O_2, ..., O_n. W systemach o ustalonych zadaniach podział taki może zostać dokonany na etapie projektowania programu. Przedmiotem rozważań niniejszej pracy jest natomiast rozpraszanie obliczeń w systemie wyszukiwania danych. Zadania dla takiego systemu są formułowane na bieżąco, a więc wyodrębnianie rozpraszanych operacji musi następować automatycznie w samym systemie. W takim przypadku niezbędne wydaje się dysponowanie odpowiednio wysokiego poziomu językiem do zapisu programów podlegających rozpraszaniu.

Celowe wydaje się więc przedyskutowanie własności języków zapytań pod kątem ich przydatności do zapisu zadań wyszukiwania, które mają być rozpraszane w lokalnej sieci komputerowej.

3. Przegląd własności języków zapytań

Dla relacyjnego modelu danych opracowano różne języki manipulowania danymi oraz języki zapytań. Tworzą one kilka poziomów abstrakcji. Poniżej przedstawiona zostanie charakterystyka tych języków, ograniczona do zagadnień wyszukiwania danych (języki zapytań). Jej podstawą jest sposób formułowania zadań wyszukiwania danych w przypadkach, kiedy pytania użytkownika dotyczą kilku plików danych. W celu zilustrowania rozważań rozpatrzymy przykładową bazę danych.

Baza danych: Studenci(nazwisko, imie, kierunek, nr_indeksu)
Oceny(nr_indeksu, ident_przedmiotu, ocena)
Przedmioty(ident_przedmiotu, nazwa, wykładowca)

W celu uproszczenia zapisu zastosowano następujące skróty:

Studenci(nazwisko, imie, k_k, nri)
Oceny(nri, idp, ocena)
Przedmioty(idp, nazwa, wykl)

Rozpatrzmy następnie dwa pytania kierowane do tej bazy i sposób ich zapisu przy użyciu różnych języków zapytań.

Przykłady zadań wyszukiwania (pytań do bazy):

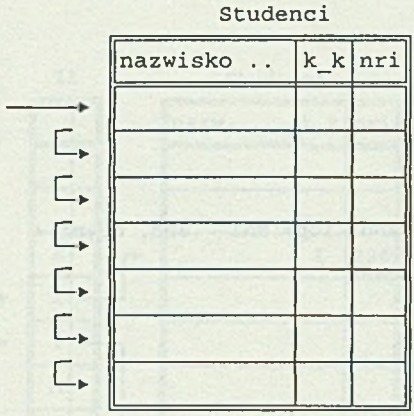
P1) Znajdź nazwiska studentów kierunku Informatyka (I).

P2) Znajdź nazwiska studentów kierunku Informatyka (I), którzy z przedmiotu matematyka (MAT) otrzymali oceny większe od 4.

3.1. Języki operujące na pojedynczych rekordach

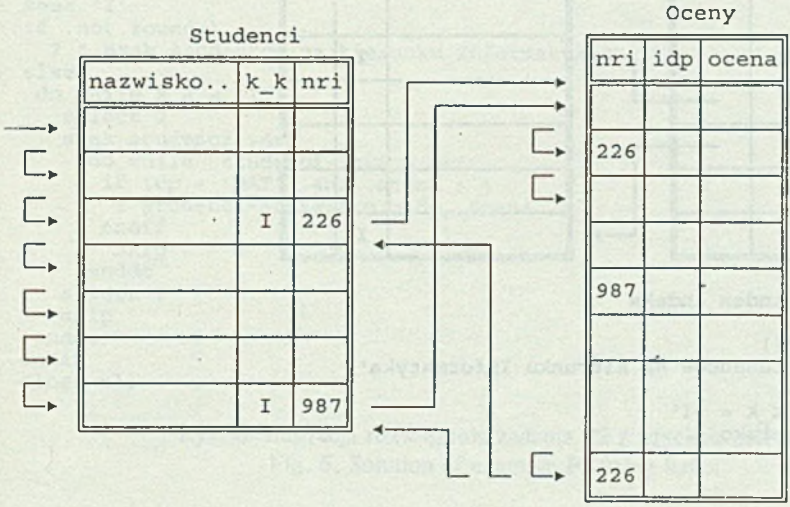
Wykorzystując język tego poziomu (np. dBASE, Clipper, FoxBase, Btrieve) programista musi "nawigować" wśród rekordów pliku, przeglądając je kolejno przy dostępie sekwencyjnym lub wybierając bezpośrednio za pomocą indeksu. Przy operacjach na kilku plikach programista musi indywidualnie kojarzyć rekordy różnych plików dobierając je przez porównanie określonych pól.

Rozwiązania powyższych zadań bez użycia indeksów ilustrują rysunki 2, 3 i 4 (programy zapisano w języku typu dBASE). Użycie indeksów zilustrowano na rysunkach 5 i 6.




```
use studenci
locate for k_k = 'I'
do while .not.eof()
   ? nazwisko
   continue
enddo
close all
```

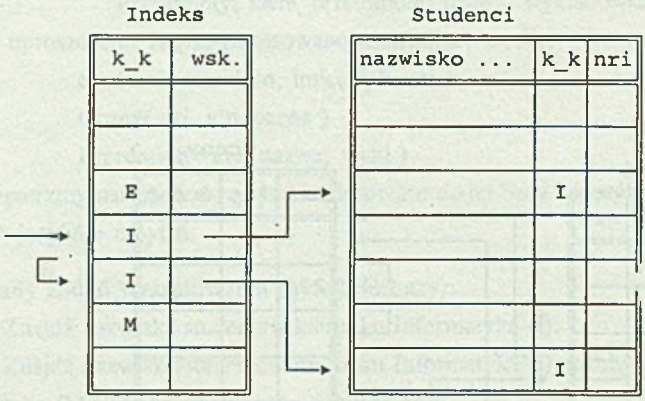
Rys. 2. Ilustracja procesu wyszukiwania w jednym pliku (zadanie P1)
Fig. 2. Data retrieving process in one file (example P1)



Rys. 3. Ilustracja procesu wyszukiwania w dwóch plikach (zadanie P2)
Fig. 3. Data retrieving process in two files (example P2)

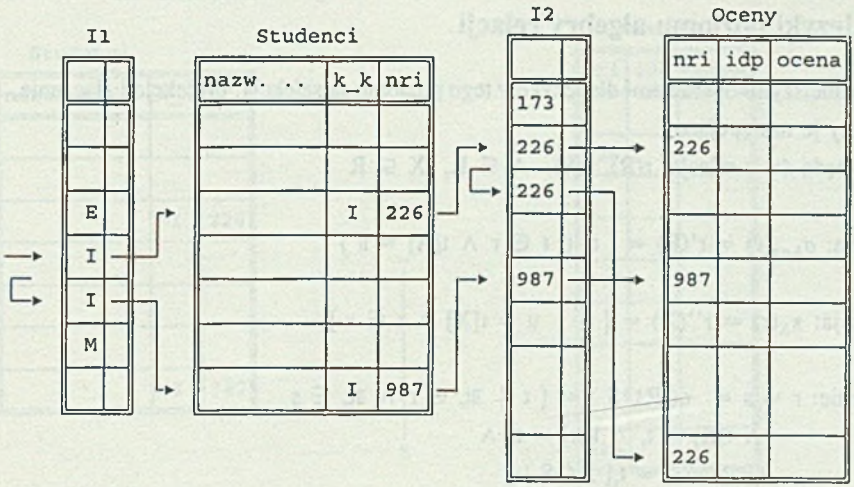
```
select 2
use oceny
select 1
use studenci
locate  for k_k = 'I'
do while .not.eof()
    select 2
    locate  for nri = studenci->nri  .and. idp='MAT' .and. ocena>4
      do while .not.eof()
          ? studenci->nazwisko, idp, ocena
          continue
      enddo
    select 1
    continue
enddo
close all
```

Rys. 4. Program wyszukiwania dla zadania P2
Fig. 4. Program of data retrieving for the example P2




```
use studenci index indeks
seek 'I'
if .not.found()
    ? 'Brak studentow na kierunku Informatyka'
else
    do while k_k = 'I'
        ? nazwisko
        skip
    enddo
endif
close all
```

Rys. 5. Ilustracja rozwiązania zadania P1 z użyciem indeksu
Fig. 5. Solution of example P1 using index




```
use studenci
index on k_k to i1
use oceny
index on nri to i2
select 1
use studenci index i1
select 2
use oceny index i2
select 1
seek 'I'
if .not.found()
  ? " Brak studentow na kierunku Informatyka"
else
 do while k_k = 'I'
   select 2
   seek studenci->nri
     do while  studenci->nri = nri
       if idp = 'MAT' .and. ocena > 4
        ? studenci->nazwisko, idp, ocena
       endif
        skip
     enddo
   select 1
   skip
 enddo
endif
close all
```

Rys. 6. Ilustracja rozwiązania zadania P2 z użyciem indeksu
Fig. 6. Solution of example P2 using index

3.2. Języki poziomu algebry relacji

Najważniejszymi operacjami dla języków tego poziomu są selekcja, projekcja i złączenie. Definiujemy je następująco:

Niech będą dane relacje: r(R), s(S), $A \in R$, $X \subseteq R$

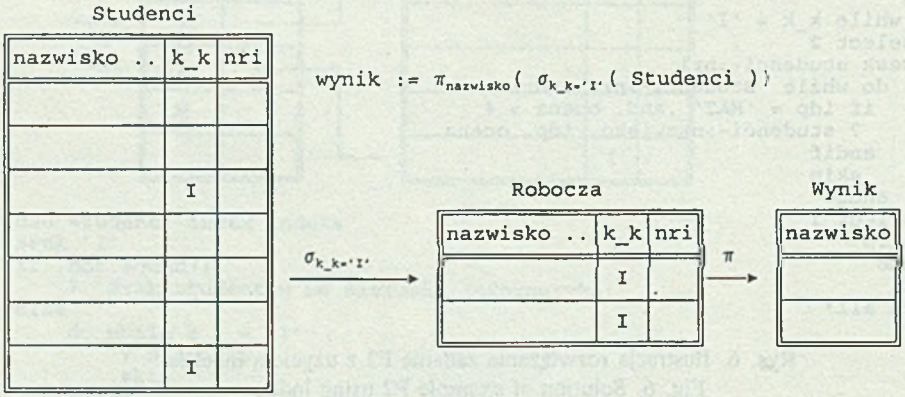
Selekcja: $\sigma_{A=a}(r) = r'(R) = \{ t \mid t \in r \wedge t[A] = a \}$

Projekcja: $\pi_X(r) = r''(X) = \{ u \mid u = t[X] \wedge t \in r \}$

Złączenie: $r \bowtie s = q(R \cup S) = \{ t \mid \exists t_r \in r \wedge \exists t_s \in s$
$\wedge\ t[R] = t_r \wedge t[S] = t_s \wedge$
$t_r[R \cap S] = t_s[R \cap S] \}$

Instrukcje języka tego poziomu (np. instrukcje COPY i JOIN systemów dBASE, Clipper, FoxBase) umożliwiają operowanie na całych plikach (relacjach), dając w efekcie nowe pliki. Kluczową dla operacji na kilku plikach jest instrukcja złączenia pozwalająca na automatyczne skojarzenie wszystkich odpowiadających sobie rekordów dwóch plików.

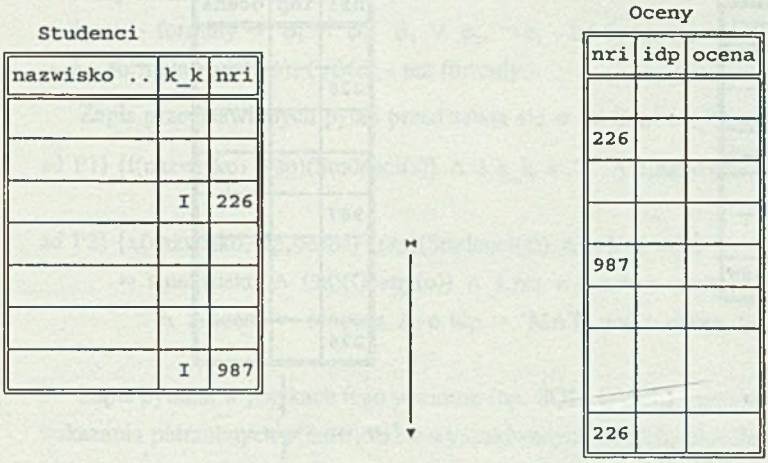
Rozwiązania rozważanych uprzednio przykładów prezentują rysunki 7 i 8. Inną możliwość rozwiązania przykładu P2 prezentuje rys. 9.




```
use studenci
copy to wynik fields nazwisko for k_k = 'I'
```

Rys. 7. Ilustracja rozwiązania zadania P1 za pomocą algebry relacji
Fig. 7. Solution of example P1 using relational algebra

$$\text{wynik} := \pi_{\text{nazwisko,idp,ocena}}(\sigma_{\text{k_k='I'} \wedge \text{idp='MAT'} \wedge \text{ocena>4}}(\text{Studenci} \bowtie \text{Oceny}))$$



$$\sigma_{\text{k_k='I'} \wedge \text{idp='MAT'} \wedge \text{ocena>4}} \circ \pi_{\text{nazwisko,idp,ocena}}$$

nazwisko	idp	ocena

```
select 2
use oceny alias o
select 1
use studenci
join with o to wynik fields nazwisko, o->idp, o->ocena;
   for nri=o->nri .and. k_k='I' .and. idp='MAT' .and. ocena>4
```

Rys. 8. Ilustracja rozwiązania zadania P2 za pomocą algebry relacji
Fig. 8. Solution of example P2 using relational algebra

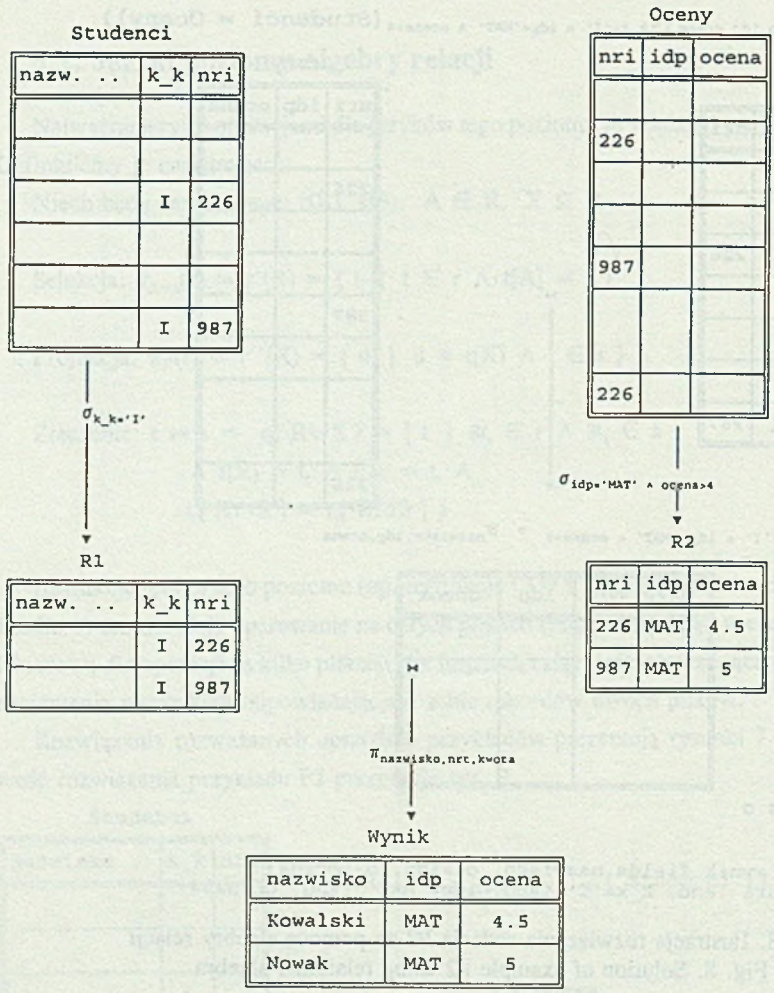
3.3. Języki poziomu rachunku relacji

Podstawy tych języków wywodzą się z rachunku zdań. Ogólną postać wyrażeń w tych językach zapisać można jako: $\{ t \mid \phi(t) \}$ i zapis ten oznacza zbiór krotek t spełniających formułę ϕ.

Formuły i poprzedzające je atomy definiujemy następująco:

Atomy: r(R), s(S) - relacje

- r(t), s(u) - t, u są zmiennymi krotkowymi w relacjach r i s,
- $t.A\ \theta\ u.B$ - $A \in R$, $B \in S$, θ ma postać =, >, <, >=, <=,
- $t.A\ \theta\ a$ - a - stała.




```
select 2
use oceny
copy to rob1 for idp = 'MAT' .and. ocena > 4
use rob1 alias r1
select 1
use studenci
copy to rob2 for k_k = 'I'
use rob2
join with r1 to wynik fields nazwisko, r1->idp, r1->ocena for nri=r1->nri
```

Rys. 9. Inne rozwiązanie zadania P2 za pomocą algebry relacji
Fig. 9. Another solution of example P2 using relational algebra

Formuły:

- każdy atom jest formułą,
- ϕ_1, ϕ_2 - formuły $\Rightarrow$ $\phi_1 \wedge \phi_2$, $\phi_1 \vee \phi_2$, $\neg\phi_1$ - też formuły,
- ϕ - formuła $\Rightarrow$ $(\exists t)(\phi)$, $(\forall t)(\phi)$ - też formuły.

Zapis przedstawionych pytań przedstawia się w tej konwencji następująco:

ad P1) {t(nazwisko) | (∃p)(Studenci(s)) ∧ s.k_k = 'I' ∧ t.nazwisko = s.nazwisko}

ad P2) {x(nazwisko,idp,ocena) | (∃p)(Studenci(s)) ∧ s.k_k = 'I' ∧ x.nazwisko =
= s.nazwisko ∧ (∃d)(Oceny(o)) ∧ s.nri = o.nri ∧ x.idp = o.idp ∧
∧ x.ocena = o.ocena ∧ o.idp = 'MAT' and o.ocena > 4}

Zapis pytania w językach tego poziomu (np. SQL, QUEL) sprowadza się (zazwyczaj) do wskazania potrzebnych plików, nazw wyszukiwanych danych, określenia wymaganych cech tych danych oraz wskazania powiązań między wykorzystywanymi plikami. Są to języki nieproceduralne w stosunku do języków algebry relacji, nie wymagają bowiem określania rodzaju ani kolejności operacji wykonywanych na plikach danych.

Zdefiniowane uprzednio zadania zapiszemy przy użyciu tych języków następująco:

Język QUEL

```
ad P1)   range of s is Studenci
         retrieve (s.nazwisko) where s.k_k='I';

ad P2)   range of s is Studenci
         range of o is Oceny
         retrieve (s.nazwisko, o.idp, o.ocena) where s.k_k='I' and o.idp='MAT' and
         o.ocena>4 and s.nri=o.nri;
```

Język SQL

```
ad P1)   select nazwisko
         from Studenci
         where k_k='I';

ad P2)   select nazwisko, idp, ocena
         from Studenci s, Oceny o
         where s.k_k='I' and o.idp='MAT' and o.ocena>4 and s.nri=o.nri;
```

lub

```
select nazwisko
from Studenci s
where k_k='I' and nri in
    (select nri
    from Oceny
    where idp='MAT' and ocena>4);
```

3.4. Język poziomu relacji uniwersalnej

Dla pewnej bazy danych $r = \{ r_1, r_2, ..., r_n \}$ o schemacie $R = \{R_1, R_2, \ldots, R_n\}$ relacja uniwersalna jest hipotetyczną relacją u utworzoną nad zbiorem atrybutów $U = R_1 \cup R_2 \cup ... \cup R_n$ z zawartości relacji $r_1, r_2, \ldots, r_n$. Istnieje kilka metod definiowania hipotetycznej zawartości relacji uniwersalnej. Proponują one między innymi wykorzystanie tzw. egzemplarza charakterystycznego [9] lub opis relacji uniwersalnej za pomocą tzw. obiektów [3, 11, 12]. Ze względu na cele pracy nie będziemy prezentowali teoretycznych podstaw relacji uniwersalnej odwołując się do cytowanej literatury. Interesującym nas aspektem będą natomiast własności języka zapytań definiowanego dla tej relacji. Głównym celem tworzenia modelu relacji uniwersalnej jest zwolnienie użytkownika z koniecznej znajomości struktury bazy, tzn. schematów relacji (plików) i powiązań między nimi. Przy tym podejściu cala baza danych traktowana jest jako jeden plik (relacja) obejmujący wszystkie pola (atrybuty) danych. Użytkownik nie musi więc znać struktury poszczególnych plików ani powiązań między nimi - są one ustalane automatycznie według identycznych nazw atrybutów z poszczególnych plików.

Formalne podstawy języka dla modelu relacji uniwersalnej stwarza rachunek relacji. Składnia tego języka może więc być wyprowadzona ze składni takich języków, jak QUEL czy SQL. Należy przy tym zauważyć, że definicja samej relacji uniwersalnej wprowadza domyślnie do każdego pytania listę wszystkich tworzących tę relację plików oraz listę warunków łączących, określających powiązania między plikami zawierającymi identyczne atrybuty. Sformułowanie pytania ogranicza się wtedy do wskazania nazw wyszukiwanych danych (atrybutów) oraz określenia cech, jakie powinny spełniać.

Zapis rozważanych uprzednio przykładów dla relacji uniwersalnej

U(nazwisko, imie, k_k, nri, idp, ocena, nazwa, wykl)

będzie miał następującą postać:

ad P1) retrieve (nazwisko) where k_k = 'I';

ad P2) retrieve (nazwisko, idp, ocena) where k_k = 'I' and idp = 'MAT' and ocena > 4;

4. Analiza możliwości równoległej realizacji zadań wyszukiwania dla różnych poziomów języków zapytań

W rozdziale tym rozważymy celowość i możliwość rozpraszania obliczeń dla zadań wyszukiwania zapisanych w różnych językach zapytań przy założeniu, że obliczenia te będą realizowane równolegle w sieci komputerowej.

4.1. Języki operujące na pojedynczych rekordach

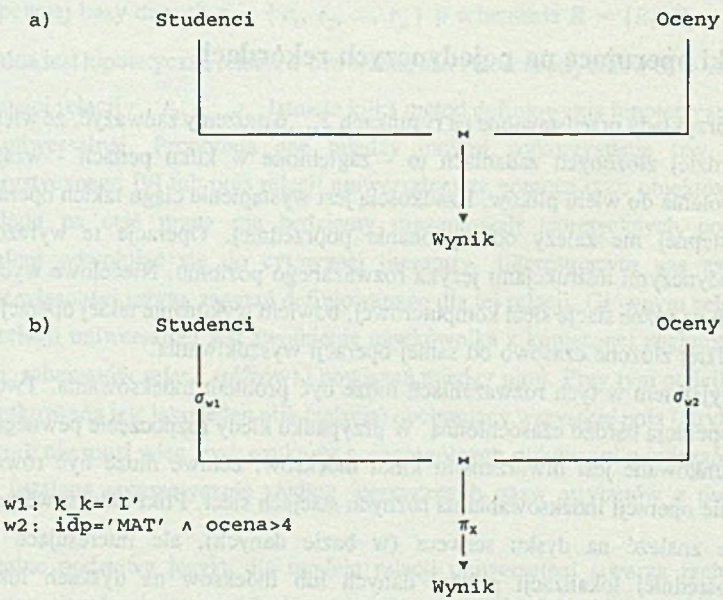
Analizując przykłady przedstawione na rysunkach 2,..,6 możemy zauważyć, że większość operacji w bardziej złożonych zadaniach to - zagłębione w kilku pętlach - wzajemnie powiązane odwołania do wielu plików. Rzadkością jest wystąpienie ciągu takich operacji, że wykonanie następnej nie zależy od wykonania poprzedniej. Operacje te wyrażone są zazwyczaj pojedynczymi instrukcjami języka rozważanego poziomu. Niecelowe wydaje się rozpraszanie ich na różne stacje sieci komputerowej, bowiem wykonanie takiej operacji może się okazać bardziej złożone czasowo od samej operacji wyszukiwania.

Istotnym wyjątkiem w tych rozważaniach może być problem indeksowania. Tworzenie indeksów jest operacją bardzo czasochłonną. W przypadku kiedy rozpoczęcie pewnego etapu obliczeń uwarunkowane jest utworzeniem kilku indeksów, celowe może być równoległe przeprowadzenie operacji indeksowania na różnych stacjach sieci. Pliki indeksowe powinny się ostatecznie znaleźć na dysku serwera (w bazie danych), ale interesujące byłoby rozważenie pośredniej lokalizacji plików danych lub indeksów na dyskach lokalnych zaangażowanych stacji.

4.2. Języki algebry relacji

Argumentami operatorów tych języków są całe relacje (pliki danych), zaś wynikiem operacji są nowe relacje (pliki). Złożoność tych operacji jest O(n) dla selekcji i projekcji oraz O(n × m) dla złączenia, gdzie n i m są liczbami rekordów w relacjach biorących udział wtych operacjach. Przy takiej złożoności uzasadnione jest więc równoległe wykonywanie obliczeń, o ile pozwoli na to brak wzajemnych uwarunkowań kolejnych operacji. Analizę takich uwarunkowań można przeprowadzić tworząc drzewo obliczeń dla wyrażenia algebry relacji będącego zapisem rozważanego zadania wyszukiwania danych.

Rozpatrzmy w tym celu ponownie dwa warianty zapisu zadania P2 za pomocą operatorów algebry relacji, zilustrowane rysunkami 8 i 9. Dla tych wyrażeń algebry relacji możemy zbudować dwa drzewa obliczeń przedstawione kolejno na rys. 10a i 10b. W strukturze pierwszego drzewa całość obliczeń jest skoncentrowana w jednym węźle, co nie daje podstaw do ich rozpraszania. Natomiast z analizy postaci drugiego drzewa (uzyskanego na drodze optymalizacji wyrażenia algebry relacji [11]) wynika, że obie selekcje σ_{w1} i σ_{w2} nie są wzajemnie uwarunkowane i mogą być realizowane w tym samym czasie po rozproszeniu odpowiadających im fragmentów obliczeń na różne stacje sieci komputerowej. Realizacja kolejnej operacji w rozważanym drzewie wymaga natomiast zakończenia obu selekcji.



Rys. 10. Dwa warianty drzewa obliczeń dla rozwiązań zadania P2
Fig. 10. Two variants of computation tree for solution of example P2

Bardziej interesującym z punktu widzenia rozpraszania obliczeń jest rozpatrzenie bardziej złożonych zadań wyszukiwania. Rozważmy w tym celu nieco rozszerzoną bazę danych:

Studenci(nazwisko, imie, srednia, k_k, nri),
Oceny(nri, idp, ocena),
Przedmioty(idp, nazwa, wykl),
Stypendia (nri, rodzaj, rok, miesiac, kwota).

Rozpatrzmy zadanie: Znaleźć nazwiska i imiona studentów, którzy uzyskali średnią ocen większą od 3.3, ocenę z egzaminu z programowania komputerów większą lub równą 3.5 i w miesiącu październiku 1993 r. otrzymali stypendium socjalne (S).

Zauważmy, że zapis tego zadania na poziomie algebry relacji będzie miał postać (jest to jedna z kilku możliwych form):

wynik := π_X (σ_{w1}(studenci) $\bowtie$ σ_{w2}(stypendia) $\bowtie$ σ_{w3}(przedmioty) $\bowtie$ σ_{w4}(oceny_egz))

gdzie X : {nazwisko, imie}

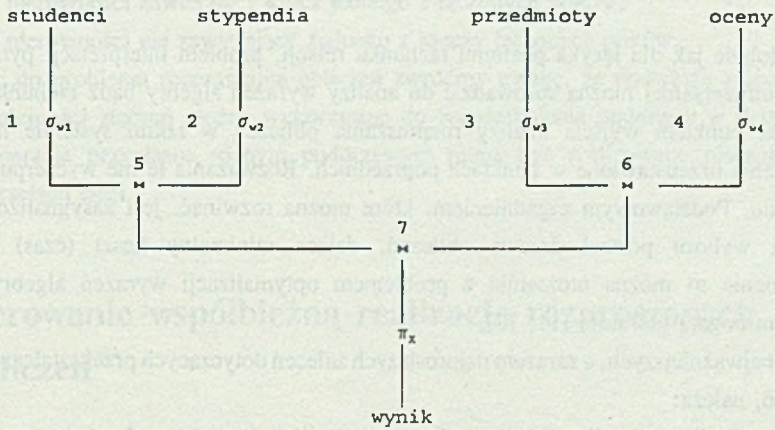
w_1 : (srednia > 3.3)

w_2 : (rodzaj = 'S') $\wedge$ (miesiac = '10') $\wedge$ (rok = '1993')

w_3 : nazwa = 'PROGRAMOWANIE KOMPUTEROW'

w_4 : (ocena >= 3.5)

Drzewo obliczeń odpowiadające powyższemu wyrażeniu przedstawiono na rys. 11.



Rys. 11. Drzewo obliczeń dla złożonego zadania wyszukiwania
Fig. 11. Computation tree for complex data retrieving problem

Rysunek ten pozwala zaproponować następującą formę rozproszenia i realizacji obliczeń:

- równoległa realizacja selekcji 1, 2, 3, 4 na czterech różnych stacjach sieci,
- równoległa realizacja złączeń 5 i 6 na dwóch z poprzednich stacji,
- końcowa operacja złączenia i projekcji 7 na docelowej stacji.

Powyższe rozważania można zakończyć wnioskiem, że analiza zadania wyszukiwania (zapisanego za pomocą algebry relacji) na odpowiednio wczesnym etapie pozwala określić możliwość rozproszenia obliczeń i wykonania ich równolegle.

Zamykając rozważania dotyczące algebry relacji zwróćmy uwagę na fakt, że realizacja operacji złączenia w językach typu dBASE (którymi posłużono się do ilustracji w rozdziale 3.2) jest wysoce nieefektywna czasowo. Znane są jednak bardzo efektywne realizacje tej

operacji w dużych systemach zarządzania bazami danych (np. INGRES), a także np. w eksperymentalnym systemie wyszukiwania danych [5, 8].

4.3. Języki rachunku relacji

W tym przypadku całe zadanie wyszukiwania może być zazwyczaj zapisane jako jedna instrukcja języka zapytań. Na poziomie samego języka (np. SQL) brak więc podstaw do rozpraszania obliczeń. Zagadnienie to może być natomiast rozważane na etapie translacji (ewentualnie interpretacji) zapisu pytania. Jednym ze stosowanych rozwiązań jest bowiem kompilacja do zapisu w języku algebry relacji. W takim przypadku pomocne mogą być rozważania z poprzedniego punktu.

4.4. System relacji uniwersalnej

Podobnie jak dla języka poziomu rachunku relacji, problem interpretacji pytań języka relacji uniwersalnej można sprowadzić do analizy wyrażeń algebry bądź rachunku relacji. Stąd też punktem wyjścia analizy rozpraszania obliczeń w takim systemie mogą być rozważania przedstawione w punktach poprzednich. Rozważania te nie wyczerpują całości problemu. Podstawowym zagadnieniem, które można rozwinąć, jest zasygnalizowana już kwestia wyboru postaci drzewa obliczeń, dająca minimalny koszt (czas) obliczeń. Zagadnienie to można utożsamić z problemem optymalizacji wyrażeń algebry relacji, mającym bogatą literaturę [11, 12].

Do najważniejszych, a zarazem najprostszych zaleceń dotyczących przekształcania drzewa obliczeń, należą:

- przesuwanie operacji selekcji w drzewie możliwie wysoko, tak aby we wszystkich możliwych przypadkach wyprzedzić operacje złączeń,
- analogiczne przesuwanie projekcji oraz związanie ich z operacjami selekcji lub złączeń w celu wspólnego wykonania.

Problemem równie ważnym, ale nie mającym tak prostego rozwiązania, jest zagadnienie kolejności złączeń wielu plików.

Jednym z kryteriów wyboru może być liczba rekordów w plikach podlegających złączeniu. Jako ogólną wytyczną można w takim przypadku przyjąć, że łączenie należy rozpocząć od pliku o najmniejszej liczbie rekordów. Uzasadnieniem takiego działania jest spostrzeżenie, że w większości przypadków operacja złączania zmniejsza liczbę rekordów większego z plików. Jeśli bowiem łączenie następuje według atrybutu kluczowego jednego z plików, to liczba rekordów pliku wynikowego nie będzie przekraczała liczności większego z plików (wynika to z definicji złączenia). Rozwijając tę wskazówkę należy uwzględnić te

pliki, na których będą wykonane operacje selekcji przed łączeniem. Jeśli dodatkowo wprowadzimy do tych rozważań rozróżnienie artybutów kluczowych, to pierwszeństwo należy dać plikom, na których wykonywana jest operacja selekcji zdefiniowana na atrybutach kluczowych. Jako ostatnie należy natomiast wykonywać złączenia plików, dla których warunek złączeń nie obejmuje atrybutów kluczowych żadnego z plików bądź też warunek ten jest typu nierównościowego.

Porządkując te rozważania można zaproponować następującą kolejność wyboru plików do operacji złączenia:

- pliki wyróżniające się mniejszą liczbą rekordów,
- pliki, na których wykonywana jest operacja selekcji wg atrybutu kluczowego.

Dodatkowo należy uwzględnić następującą kolejność warunków złączeń:

- warunek równości zawierający klucz jednego z łączonych plików,
- warunek nierówności zawierający klucz jednego z łączonych plików,
- warunek nierówności nie zawierający żadnego z kluczy łączonych plików.

Wracając do problemu rozpraszania obliczeń zwróćmy uwagę, że powyższe zalecenia dotyczące kolejności złączeń można wykorzystać do wyodrębnienia poddrzew w drzewie obliczeń. Operacje przypisane różnym poddrzewom mogą być realizowane równolegle w różnych stacjach sieci.

5. Sterowanie współbieżną realizacją rozproszonych obliczeń

5.1. Klasyczny model sterowania

Przystępując do dyskusji problemów realizacji obliczeń rozproszonych załóżmy, że każda stacja udostępniona do takich obliczeń dysponuje programem zdolnym do analizy przesyłanych parametrów i uruchomienia jednej z procedur bibliotecznych realizujących operacje algebry relacji. Przekazywane parametry dotyczą rodzaju operacji i nazw plików wykorzystywanych do jej realizacji. Mogą one też ewentualnie dotyczyć lokalizacji plików, załóżmy jednak dla uproszczenia, że wszystkie pliki mieszczą się na dysku serwera sieci. Rozpraszanie obliczeń będzie więc polegało na zdalnym uruchomieniu w kilku stacjach sieci wybranych procedur z rozmieszczonych tam uprzednio bibliotek.

Proces generacji programu wyszukiwania dla zadania (pytania) sformułowanego w języku zapytań wysokiego poziomu (np. relacja uniwersalna) w ogólnym zarysie można przedstawić następująco

1. Przekształcanie pytania do ogólnego zapisu na poziomie algebry relacji (bez uwzględniania rozproszenia).
2. Optymalizacja na poziomie algebry relacji z uwzględnieniem rozproszenia (równoległej realizacji w sieci).
3. Wyróżnienie w drzewie obliczeń podstawowej ścieżki realizowanej jako program główny.
4. Generacja programu głównego z punktami wywołań procedur przekazujących parametry do obliczeń rozproszonych oraz procedur synchronizacji.

Bliższa analiza takiego podejścia pozwala wskazać na pewne - mogące się czasami ujawniać - trudności:

- Optymalizacja wyrażeń relacyjnych z uwzględnieniem rozproszenia (a dokładniej określenie kolejności złączeń) musi być prowadzona z pełną informacją o liczbie (i parametrach) wolnych stacji sieci.
- Wygenerowany program wynikowy przypisuje konkretne operacje konkretnym stacjom sieci.
- Niezrealizowanie operacji we wskazanej stacji (wskutek np. wyłączenia stacji czy też zajęcia jej przez nieświadomego użytkownika) praktycznie uniemożliwia dokończenie programu. Należałoby w takim przypadku powtórzyć generację programu zmieniając (być może automatycznie) wykaz aktualnie wolnych stacji sieci.

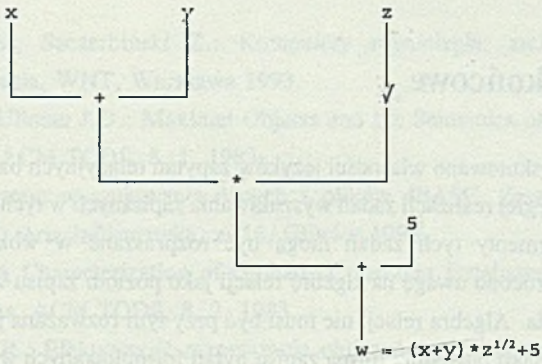
Przedstawione problemy mogą znaleźć - przynajmniej częściowe - rozwiązanie w przedstawionym dalej innym modelu sterowania realizacją obliczeń.

5.2. Sterowanie współbieżną realizacją obliczeń przy wykorzystaniu modelu przepływowego

Punktem wyjścia do rozważenia takiego modelu jest analogia między drzewem obliczeń tworzonym dla zadań wyszukiwania a grafem przepływu danych.

Graf przepływu danych [2, 6] zawiera węzły operacyjne i decyzyjne, węzły łączące i rozgałęziające oraz krawędzie, po których przenoszone są tzw. znaczniki danych. Najistotniejsze w rozważanej analogii węzły operacyjne mogą realizować różne rodzaje operacji arytmetycznych i innych (rys. 12). Dla pierwszych modeli komputerów przepływowych były to operacje elementarne, odpowiadające rozkazom maszynowym, tzn. dodawanie, odejmo-

wanie, mnożenie itd. Komputery dla bardziej specjalizowanych zastosowań zawierają węzły operacyjne realizujące np. szybką transformatę Fouriera, splot, filtrację sygnałów itp.



Rys. 12. Przykładowy graf przepływu danych
Fig. 12. Example of dataflow graph

Znaczniki danych zawierają w najprostszym przypadku wartości danych oraz adresy docelowe, pod które mają być przesyłane.

Kluczowym założeniem dla koncepcji komputera przepływowego jest uzależnienie wykonania operacji w węźle od obecności znaczników danych na wszystkich krawędziach wejściowych do danego węzła.

Zauważmy, że przedstawione drzewo obliczeń może być rozważane jako graf przepływu danych. Węzły takiego grafu będą zawierały operatory algebry relacji. Znaczniki w tym ujęciu będą zawierały całe pliki danych (ewentualnie ich identyfikatory). Warunkiem wykonania operacji będzie dostępność odpowiednich plików. Jej rezultatem będzie nowy plik danych. Oczywiście, należy w tym miejscu wyjaśnić możliwości praktycznego zrealizowania jednostek wykonujących tego rodzaju operacje. Aczkolwiek nie jest wykluczona sprzętowa realizacja takiego komputera, to zupełnie wystarczające dla naszych rozważań będzie stwierdzenie, że wygodnym narzędziem dla takich badań może być symulator komputera przepływowego [1, 4], wykorzystujący sieć stacji roboczych. W tej koncepcji poszczególne stacje będą realizowały zlecone im operacje algebry relacji.

Rozważmy teraz (uprzedzając opis samego symulatora) ewentualne korzyści wynikające z przyjęcia takiego modelu sterowania współbieżną realizacją obliczeń:

1. Opis zadania obliczeniowego w konwencji grafu przepływu danych nie wymaga żadnych dodatkowych działań w celu wskazania fragmentów programu realizowanych równolegle.
2. Po wyborze gotowych do realizacji operacji są one kierowane do modułu wykonania dysponującego dowolnym (niepustym) zbiorem jednostek przetwarzających, czyli stacji

sieci. Tak więc analiza i optymalizacja wyrażeń algebry relacji nie jest uwarunkowana znajomością aktualnej dyspozycyjności stacji sieci.

6. Uwagi końcowe

W pracy przedyskutowano własności języków zapytań relacyjnych baz danych pod kątem możliwości równoległej realizacji zadań wyszukiwania zapisanych w tych językach. Założono przy tym, że fragmenty tych zadań mogą być rozpraszane w wolnych stacjach sieci komputerowej. Zwrócono uwagę na algebrę relacji jako poziom zapisu wygodny dla analizy procesu rozpraszania. Algebra relacji nie musi być przy tym rozważana jako docelowy język zapytań, ale raczej jako pośrednia forma zapisu pytań formułowanych w językach wyższego poziomu, np. z wykorzystaniem modelu relacji uniwersalnej.

Dalsze prace nad realizacją przedstawionych koncepcji wymagają między innymi odpowiedzi na pytanie, czy prowadzenie równoległych obliczeń w sieci komputerowej dla tak specyficznych zadań, jakie tworzy wyszukiwanie danych, nie spowoduje przeciążenia sieci. Częściową odpowiedź na to pytanie, dającą podstawy do optymizmu i zachętę do dalszych badań, przynosi praca [10].

W pracy zasygnalizowano też różne możliwości sterowania rozpraszaniem obliczeń, zwracając uwagę na celowość wykorzystania w tym zakresie metody sterowania przepływem danych.

LITERATURA

[1] Bieniok B.: Symulator komputera sterowanego przepływem danych. Praca dyplomowa. Instytut Informatyki. Politechnika Śląska, Gliwice 1993.

[2] Hwang K., Briggs F. A.: Computer architecture and parallel processing. McGraw-Hill, New York 1984.

[3] Korth H.F., Kuper G.M., Feigenbaum J., van Gelder A., Ullman J.D.: System/U: A Database System Based on the Universal Relation Assumption. ACM TODS, 9, 3, 1984.

[4] Kozielski S., Bieniok B.: Zastosowanie sieci komputerowej do symulacji komputera przepływowego realizującego zadania wyszukiwania danych, Zeszyty Naukowe Politechniki Śl., seria Informatyka, z. 24, Gliwice 1993.

[5] Kozielski S., Piec J., Grzywocz J.: System wyszukiwania danych oparty na modelu relacji uniwersalnej, Archiwum Informatyki Teoretycznej i Stosowanej, 1993 (w druku).

[6] Kozielski S., Szczerbiński Z.: Komputery równoległe: architektura, elementy programowania, WNT, Warszawa 1993.

[7] Maier D., Ullman J.D.: Maximal Objects and the Semantics of Universal Relation Databases. ACM TODS, 8, 1, 1983.

[8] Piec J.: System wyszukiwania danych z plików dBASE. Zeszyty Naukowe Politechniki Śl., seria Informatyka, z. 16, Gliwice 1993.

[9] Sagiv Y.: A Characterization of Globally Consistent Databases and Their Correct Access Paths. ACM TODS, 8, 2, 1983.

[10] Tutajewicz R.: Efektywność rozpraszania obliczeń w sieci komputerowej dla zadań wyszukiwania danych, Zeszyty Naukowe Politechniki Śl., seria Informatyka, z. 25, Gliwice 1994.

[11] Ullman J.D.: Principles of Database Systems. Computer Science Press, Rockville, 1983. Polskie wydanie: Systemy baz danych. WNT, Warszawa 1988.

[12] Ullman J.D.: Database and Knowledge-Base Systems. Computer Science Press, Rockville, 1989.

[13] Wilk A.: Wykorzystanie sieci NetWare firmy Novell do tworzenia oprogramowania współbieżnego na przykładzie implementacji równoległego algorytmu rozwiązania uproszczonego problemu Hilberta, Gliwice 1990 (opracowanie niepublikowane).

Recenzent: Dr inż. Andrzej Wilk

Wpłynęło do Redakcji 19 listopada 1993 r.

Abstract

The main objective is to indicate links between a level of the data retrieving task and a possibility of computing distribution necessary to realize the retrieving. Four levels of languages are considered.

Languages operating on single records. Using a language of this level the programmist should navigate among file records. Operating on several files the programmist must individualy associate records of different files matching them by comparison of particular fields. Such operations usually can't be distributed for several computers.

Languages of relational algebra level. The most important operations of the languages of this level are selection, projection and join. Complexity of those operations justifies parallel computing if sequential operations are independent.

Languages of relational calculus level. The fundamentals of these languages are based on the classical predicate calculus. Interpretation of formulated queries leads to query transform into the form of relational algebra or operations on single records, so computing distribution conditions are similary to the ones defined for previous two classes.

The language of universal relation level. In this approach the whole database is treated as one relation covering all the attributes. The query notation is simplified in comparison to the languages of relational calculus since there is no need to indicate neither the needed files nor linking conditions. The files and the links between them are chosen automatically in the system of query interpretation. Moreover the interpretation is similar to the one in the relational calculus as well as computing distribution.