Polskie Towarzystwo Informatyczne
Sekcja Baz Danych

do użytku wewnętrznego

SZKOŁA

"BAZY DANYCH '85"

KARPACZ
3 — 6 grudnia 1985

Od organizatorów

Głównym celem szkoły "Bazy Danych '85" zorganizowanej przez Sekcję Baz Danych Polskiego Towarzystwa Informatycznego jest umożliwienie poszerzenia wiedzy i zdobycia doświadczeń osobom zainteresowanym problematyką baz danych.

Tematy poruszone w referatach koncentrują się na następujących zagadnieniach: reprezentacja wiedzy, systemy ekspertowe, sztuczna inteligencja, modelowanie konceptualne, relacyjne bazy danych, rozproszone bazy danych, komputerowe wspomaganie projektowania baz danych.

Staraliśmy się o to, aby poruszone tematy należały do głównego nurtu badań, jak również o jak najwyższy poziom referatów.

Szkoła została zorganizowana w krótkim czasie, co zmusiło zarówno autorów referatów jak i organizatorów do intensywnego wysiłku. Pragniemy więc podziękować prelegentom za szybkie przygotowanie tekstów, dzięki czemu książka ta jest dostępna dla uczestników już w momencie rozpoczęcia Szkoły. Pragniemy również serdecznie podziękować wszystkim tym osobom, które w jakikolwiek sposób pomogły w organizacji Szkoły.

Marian Furman
Zbigniew Struk
Kazimierz Sycz
Stanisław Wrycza

Spis treści

PRZECHOWYWANIE WIEDZY I "IGNORANCJI" W BAZACH DANYCH

Witold Łukaszewicz

Instytut Informatyki
Uniwersytet Warszawski
00-901 Warszawa, skr. pocz. 1210

Streszczenie

Wiele zastosowań praktycznych wymaga przechowywania i manipulowania wiedzą. Jednym z możliwych sposobów przechowywania wiedzy w bazach danych jest reprezentowanie jej w postaci formuł pewnego systemu logicznego.

Charakterystyczną cechą tradycyjnych systemów logicznych, rozważanych w filozoficznej i matematycznej literaturze, jest możliwość wyprowadzania wniosków wyłącznie w oparciu o fakty już udowodnione, a więc wyłącznie w oparciu o wiedzę.

W codziennym życiu posługujemy się często metodami wnioskowania, w których konkluzje zależą nie tylko od naszej wiedzy, ale również od naszej ignorancji. Typowym przykładem takiego wnioskowania jest reguła stwierdzająca, że osoba jest niewinna, o ile nie można dowieść, że jest inaczej.

Typową cechą konkluzji wyprowadzonych w oparciu o ignorancję, jest możliwość ich unieważnienia przez nową informację. Własność ta nazywana jest niemonotonicznością, ponieważ w tego typu systemach logicznych zbiór twierdzeń nie zwiększa się monotonicznie wraz ze wzrostem zbioru aksjomatów.

Ponieważ modelowanie sposobów wnioskowania specyficznych dla istot ludzkich jest centralnym problemem sztucznej inteligencji, formalizacja metod wnioskowania niemonotonicznego jest kluczowa dla dalszego rozwoju tej dyscypliny. Jest rów-

nież jasne, że bazy wiedzy konstruowane dla systemów sztucznej inteligencji, powinny opierać się na niemonotonicznej strukturze dedukcyjnej, a zatem zawierać nie tylko wiedzę, ale pośrednio, także ignorancję.

Celem niniejszej pracy jest uzasadnienie konieczności stosowania wnioskowania niemonotonicznego przez systemy sztucznej inteligencji, przedstawienie podstawowych problemów związanych z jego formalizacją oraz krótka prezentacja dwóch konkretnych formalizmów wnioskowania niemonotonicznego.

1. Wstęp

Jednym z problemów ogólnej teorii baz danych jest problem przechowywania i manipulowania wiedzą, rozumianą jako zbiór faktów opisujących pewną dziedzinę zastosowań. Bazy danych, których zawartością jest wiedza, nazywane są bazami wiedzy (ang. knowledge bases).

Spośród różnych możliwych sposobów reprezentowania wiedzy w bazach danych, coraz więcej zwolenników ma podejście logiczne, zakładające, że wiedza reprezentowana jest jako zbiór formuł pewnego systemu logicznego, na ogół klasycznej logiki pierwszego rzędu.

Zalety logicznego podejścia do problemu reprezentacji wiedzy zilustrujemy przykładem, stanowiącym jednocześnie pretekst do nieformalnego przypomnienia podstawowych pojęć klasycznej logiki pierwszego rzędu. Rozważmy następującą bazę wiedzy opisującą pewien mikroświat ptaków:

(1) ptak(Agata)

(2) kanarek(Sylwia)

(3) kanarek(Kuba)

(4) struś(Maciek)

(5) lubi(Kuba,Sylwia)

(6) $\forall x$(lubi(Maciek,x))

Powyższa baza wiedzy zapisana jest w języku klasycznej logiki pierwszego rzędu. "Agata", "Sylwia", "Kuba" i "Maciek" to symbole stałych, interpretowane jako nazwy konkretnych obiektów występujących w opisywanym świecie. "ptak", "kanarek" i "struś" to symbole jednoargumentowych relacji predykatów, interpretowane jako nazwy konkretnych jednoargumentowych relacji, określonych na obiektach świata. Jednoargumentowe relacje nazywane są własnościami. Zatem formuła (1) stwierdza, że obiekt o nazwie Agata posiada własność o nazwie ptak, lub po prostu, że Agata jest ptakiem. Formuły (2)-(4) stwierdzają, że Sylwia i Kuba są kanarkami, natomiast Maciek jest strusiem. "lubi" to symbol relacji dwuargumentowej, interpretowany jako nazwa pewnej konkretnej dwuargumentowej relacji określonej na obiektach świata. Formuła(5) stwierdza zatem, że obiekty o nazwach Kuba i Sylwia znajdują się w tej relacji, czyli, że Kuba lubi Sylwię. Należy podkreślić, że nie wynika stąd wcale, że Sylwia lubi Kubę.

Formuły (1)-(5) mają bardzo podobną budowę. Takie formuły nazywane są formułami atomowymi. Stwierdzają one bardzo proste fakty, a mianowicie, że pewne obiekty opisywanego świata znajdują się w pewnych relacjach. Z formuł atomowych bu-

dujemy formuły złożone. Przykładem takiej formuły jest formuła (6). Symbol "$\forall$" to kwantyfikator ogólny, który czytamy "dla każdego". "x" jest symbolem zmiennej. Zmienne interpretujemy jako obiekty świata. Zatem formuła (6) stwierdza, że dla każdego obiektu x, obiekt o nazwie Maciek znajduje się w relacji o nazwie lubi z obiektem x. Mówiąc prościej, że Maciek wszystkich lubi.

Konstrukcja logicznej bazy wiedzy opisującej pewien konkretny świat, sprowadza się do podania zbioru formuł reprezentujących fakty zachodzące w tym świecie. Na ogół są to fakty najbardziej oczywiste, opisujące podstawowe związki zachodzące w tym świecie. Fakty te nazywamy faktami pierwotnymi, a reprezentujące je formuły aksjomatami. Wszystkie inne fakty, a dokładniej reprezentujące je formuły, zwane twierdzeniami, wyprowadza się z aksjomatów na drodze logicznego wnioskowania. Jest to możliwe, ponieważ z każdym systemem logicznym związana jest pewna struktura dedukcyjna, określona w postaci zbioru tzw. reguł wnioskowania. Reguły wnioskowania mają zawsze charakter czysto syntaktyczny, tzn. są pewnymi manipulacjami na formułach traktowanych jako napisy. Każda reguła wnioskowania jest zawsze postaci:

$$\frac{A_1,\ldots,A_n}{B}$$

Powyższy zapis należy czytac: " z formuł $A_1,\ldots, A_n$ można wnioskowac formułę B".

Zbiór twierdzeń, które można wyprowadzic z danego zbioru

aksjomatów, definiuje się jako najmniejszy zbiór spełniający następujące dwa warunki:

(1) Każdy aksjomat jest twierdzeniem.

(2) Jeśli $A_1,\ldots,A_n$ są twierdzeniami oraz $\frac{A_1,\ldots,A_n}{B}$ jest regułą wnioskowania, to B jest twierdzeniem.

Innymi słowy, twierdzenia to aksjomaty oraz formuły wyprowadzalne z nich w wyniku stosowania reguł wnioskowania.

Różne podręczniki logiki podają różne zestawy reguł wnioskowania dla klasycznej logiki pierwszego rzędu. Wadą tych zestawów jest fakt, że każdy z nich wymaga dołączenia odpowiedniego zbioru dodatkowych aksjomatów. Aksjomaty te nie opisują żadnej konkretnej rzeczywistości, ale reprezentują fakty prawdziwe w każdym świecie. Nazywamy je aksjomatami logicznymi. Typowym przykładem formuły, która może być uznana za aksjomat logiczny, jest każda formuła postaci $A \vee \sim A$, stwierdzająca, że pewien fakt zachodzi lub nie zachodzi.

Systemy automatycznego dowodzenia twierdzeń oparte są zwykle na tzw. regule rezolucji. Reguła ta jest formalnie dość skomplikowana. Jedną z jej zalet jest to, że nie wymaga ona dodawania żadnych aksjomatów logicznych.

Mówiąc o automatyzacji dowodzenia twierdzeń w logice klasycznej, należy podkreślić, że jest to zadanie nierozstrzygalne. Nie istnieje bowiem żaden algorytm, który dla danego zbioru aksjomatów X oraz danej formuły A, rozstrzyga, czy A jest twierdzeniem wyprowadzalnym z X. Wszystko co można osiągnąć,

i co charakteryzuje rzeczywiste systemy dowodzenia twierdzeń w logice pierwszego rzędu, to skonstruowanie algorytmu, który dla zadanej formuły wyprowadzalnej z danego zbioru aksjomatów, zatrzymuje się i stwierdza, że formuła jest wyprowadzalna. W przypadku przeciwnym algorytm ten może się nigdy nie zatrzymać.

Fakt, że formuła A jest wyprowadzalna z danego zbioru formuł X zapisujemy $X \vdash A$.

Niech X będzie zbiorem formuł zawartych w pewnej bazie wiedzy. Mówimy, że baza ta jest sprzeczna wtedy i tylko wtedy, gdy dla każdej formuły A zachodzi $X \vdash A$. W przeciwnym przypadku bazę nazywamy niesprzeczną. Mówimy, że baza jest zupełna wtedy i tylko wtedy, gdy dla każdej formuły A zachodzi $X \vdash A$ lub $X \vdash \sim A$. W przeciwnym przypadku mówimy, że baza jest niezupełna.

Wracając do bazy wiedzy opisującej mikroświat ptaków, można pokazać, że jest ona niezupełna, a zatem i niesprzeczna. Posługując się aksjomatami (1) - (6) nie możemy na przykład stwierdzić, ani że Sylwia jest ptakiem, ani że nie jest. Wynika to stąd, że nasza baza nie zawiera informacji mówiącej, że kanarki są ptakami. Gdybyśmy chcieli ten fakt uwzględnić, musielibyśmy dodać nowy aksjomat:

$$\forall x(\text{kanarek}(x) \supset \text{ptak}(x))$$

Analizując bazę wiedzy określoną aksjomatami (1) - (6), stwierdziliśmy, że reprezentuje ona pewien świat ptaków. Hipotezę tę uzasadnia jednak wyłącznie semantyka użytych

nazw. Należy sobie zdawać sprawę, że nazwy języka logiki mogą reprezentować dowolne obiekty i dowolne relacje zachodzące między nimi. Należy się zatem spodziewać, że ta sama reprezentacja może opisywać fakty prawdziwe w wielu różnych światach. Aby się przekonać, że tak jest w istocie, rozważmy świat liczb naturalnych. Jeśli nazwiemy:

liczbę 1 - "Maciek"
liczbę 2 - "Kuba"
liczbę 3 - "Sylwia"
liczbę 4 - "Agata"
własność bycia liczbą parzystą - "ptak"
własność bycia liczbą pierwszą - "kanarek"
własność bycia liczbą mniejszą niż 5 - "struś"
relację $\leq$ - "lubi"

to czytelnik może się przekonac, że wszystkie fakty reprezentowane przez aksjomaty (1)-(6) są prawdziwe w świecie liczb naturalnych. Przykładowo, aksjomat (6) stwierdza, że liczba 1 jest mniejsza bądź równa od dowolnej liczby naturalnej. Oczywiście nikt rozsądny nie użyje tego typu nazw do opisu świata liczb naturalnych. Nie zmienia to jednak faktu, że skoro obiekty i relacje opisywanego świata można nazwać dowolnie, to aksjomaty (1)-(6) należy uznać za formalnie poprawną reprezentację fragmentu wiedzy o liczbach naturalnych.

Fakt, że baza wiedzy może równocześnie reprezentować różne, często bardzo odległe dziedziny, nie prowadzi do żadnych istotnych problemów. Należy zdawać sobie sprawę, że lo-

gika posługuje się czysto syntaktycznym mechanizmem wnioskowania, sprowadzającym się do manipulowania formułami jako napisami. Reguły wnioskowania używane w systemach logicznych nie są przypadkowe. Wybiera się tylko takie reguły, w których prawdziwość przesłanek gwarantuje prawdziwość konkluzji, bez względu na specyficzne cechy reprezentowanego świata. Zauważmy ponadto, że każdy proces logicznego wnioskowania rozpoczyna się od aksjomatów. A zatem bez względu na własności opisywanego świata, prawdziwość aksjomatów zapewnia prawdziwość wyprowadzalnych z nich twierdzeń. Oczywiście, w różnych światach twierdzenia te mają inną interpretację. Tym niemniej, zawsze reprezentują prawdziwe fakty.

Powyższą uwagę zilustrujemy na przykładzie bazy wiedzy zadanej aksjomatami (1)-(6). Można pokazać, że z aksjomatu (6) wynika formuła

lubi(Maciek,Agata)

W świecie ptaków formuła ta reprezentuje fakt, że Maciek lubi Agatę, co natychmiast wynika z faktu, reprezentowanego aksjomatem (6), że Maciek lubi wszystkich. W świecie liczb naturalnych, ta sama formuła opisuje fakt, że liczba naturalna 1 jest niewiększa od liczby 4, co tym razem wynika z faktu, również reprezentowanego aksjomatem (6), że liczba 1 jest najmniejszą liczbą naturalną.

Powyższa dyskusja pozwala sformułować następujące zalety logicznej reprezentacji wiedzy w bazach danych:

(1) Dobrze określona struktura dedukcyjna systemów logicznych umożliwia sprowadzenie wnioskowania w opisywanym świecie do dowodzenia twierdzeń w logice formalnej.

(2) Dobrze określona, jasna i ogólnie akceptowana, przynajmniej w przypadku klasycznej logiki pierwszego rzędu, semantyka pozwalająca traktować zawartośc bazy jako opis pewnej rzeczywistości, a nie jako zbioru abstrakcyjnych napisów.

(3) Proste i jednoznaczne konwencje notacyjne wpływające na czytelność.

(4) Precyzyjnie określone pojęcia takie jak niesprzeczność, czy zupełność, trudne do zdefiniowania w innych metodach reprezentacji wiedzy.

Jak stwierdziliśmy, logiczna baza wiedzy jest zbiorem formuł pewnego systemu logicznego. W literaturze możemy znaleźć bardzo wiele różnych systemów logicznych. Wszystkie one posiadają podobną strukturę dedukcyjną, opartą na założeniu, że każda reguła wnioskowania jest schematem postaci:

"z formuł $A_1,\ldots,A_n$ wnioskuj formułę B".

Wynika stąd zatem, że każdy wniosek, który można w tych systemach wyprowadzić, wynika wyłącznie z aksjomatów i wcześniej udowodnionych twierdzeń. Systemy logiczne o powyższej strukturze dedukcyjnej będziemy nazywać systemami <u>standardowymi</u>. Można powiedzieć zatem, że konkluzje, które wyprowadzamy

w systemach standardowych, oparte są wyłącznie o naszą wiedzę.

Niech X będzie zbiorem formuł pewnego systemu logicznego. Zbiór wszystkich twierdzeń wyprowadzalnych ze zbioru X oznaczamy przez Th(X). Formalnie:

$$Th(X) = \{A: \; X \vdash A\}.$$

Specyficzną cechą wszystkich standardowych systemów logicznych jest ich monotoniczność. Własność ta stwierdza, że każde twierdzenie wyprowadzalne z danego zbioru aksjomatów pozostaje twierdzeniem, jeśli do zbioru aksjomatów dodamy nową formułę. Innymi słowy, zbiór twierdzeń zwiększa się monotonicznie wraz ze wzrostem aksjomatów:

$$\text{jeśli } X \subseteq Y \text{ , to } Th(X) \subseteq Th(Y).$$

Istnieją zastosowania, szczególnie w dziedzinie sztucznej inteligencji, wymagające baz wiedzy opartych o niestandardowe systemy logiczne. Specyficzną cechą tych systemów jest ich niemonotoniczność, tzn. możliwość wyprowadzenia twierdzenia, które zostaje unieważnione w wyniku dodania nowego aksjomatu. Bierze się to z faktu, że konkluzje wyprowadzalne w niestandardowych systemach logicznych zależą nie tylko od wiedzy, ale również od ignorancji. Można zatem powiedzieć, że bazy wiedzy oparte o niestandardowe systemy logiczne zawierają pośrednio także ignorancję.

W niniejszej pracy uzasadnimy konieczność stosowania wnioskowania niemonotonicznego przez systemy sztucznej inteligencji, przedstawimy podstawowe problemy związane z jego

formalizacją, oraz przedstawimy dwa konkretne formalizmy niemonotoniczne.

2. Problemy wnioskowania niemonotonicznego

Sztuczna inteligencja zajmuje się konstruowaniem systemów symulujących inteligentne zachowanie się istot ludzkich. Jest zatem oczywiste, że formalizacja mechanizmów wnioskowania specyficznych dla ludzi jest centralnym problemem tej dziedziny. Wnioskowanie tego typu nazywamy wnioskowaniem naturalnym (ang. common-sense reasoning).

Specyficzną cechą wnioskowania naturalnego jest konieczność uwzględniania ignorancji w procesie wyprowadzania konkluzji. Zilustrujemy to następującym przykładem (Winograd, 1980).

Załóżmy, że planuję podróż samochodem, który zostawiłem wczoraj na parkingu niewidocznym z okien mojego mieszkania. O ile jestem człowiekiem racjonalnym, to pierwszą czynnością jest udanie się na parking. Oznacza to, że spodziewam się znaleźć tam samochód. Z drugiej strony, ponieważ parking jest niewidoczny z okien mojego mieszkania, konkluzja, że samochód znajduje się na parkingu, nie może wynikać z mojej wiedzy. W rzeczywistości potrafię sobie wyobrazic wiele różnych sytuacji, w których samochodu na parkingu nie ma. Na przykład ktoś mógł go ukraść. Ale nigdy nie będę w stanie wykonać jakiejkolwiek czynności, jeśli będę chciał koniecznie uwzględnić wszystkie możliwe okoliczności uniemożliwiające jej wy-

konanie. Zamiast zatem siedzieć i analizować wszystkie możliwe scenariusze, akceptuję regułę wnioskowania:

"jeśli nie wiesz, że jest inaczej, załóż, że samochód jest tam, gdzie go zostawiłeś"

i udaję się na parking.

Należy podkreślić, że ponieważ każda nowa informacja zmniejsza naszą ignorancję, każda konkluzja wyprowadzona w oparciu o regułę uwzględniającą ignorancję, może zostać unieważniona w wyniku dopływu nowych faktów. Chcąc rozpocząć podróż zakładam, że samochód jest na parkingu. Ale informacja, że samochód został skradziony, natychmiast obala powyższe założenie.

W codziennym życiu ciągle akceptujemy konkluzje, które mogą być unieważnione w wyniku dopływu nowych informacji. Umiejętność wyprowadzania tego typu wniosków, które będziemy nazywali hipotezami (ang. belief), sprawia, że wnioskowanie naturalne jest niemonotoniczne, tzn. zbiór konkluzji nie zwiększa się monotonicznie wraz ze wzrostem zbioru przesłanek.

Aby zilustrować pewne problemy związane z niemonotonicznością, rozważmy regułę:

"jeśli pewna osoba jest dzieckiem, to jeśli nie wiesz, że jest inaczej, żałóż, że osoba ta ma rodziców".

Wiarygodność powyższej reguły opiera się na obserwacji, że

dzieci na ogół mają rodziców. Jest to oczywiście reguła niemonotoniczna. Jeśli cała nasza wiedza o Janku sprowadza się do faktu, że jest on dzieckiem, to reguła ta pozwala nam założyć, że Janek ma rodziców. Założenie to musimy jednak natychmiast odrzucić, jeśli dowiemy się, że Janek jest sierotą.

Oto dwie przybliżone reprezentacje powyższej reguły w standardowej logice pierwszego rzędu:

(1) $\forall x(\text{dziecko}(x) \supset \text{ma-rodziców}(x))$

(2) $\forall x(\text{dziecko}(x) \wedge \sim\text{sierota}(x) \supset \text{ma-rodziców}(x))$

Pierwsza z powyższych reprezentacji jest za silna. Opisuje ona fałszywy fakt, że wszystkie dzieci mają rodziców. Umieszczenie (1) w bazie wiedzy jest ryzykowne. Może bowiem łatwo doprowadzić do sprzeczności bazy.
Druga z powyższych reprezentacji jest za słaba. Posługując się nią, nie możemy założyć o typowym dziecku, że ma ono rodziców. Założenie to wymaga dodatkowej informacji, że rozważane dziecko nie jest sierotą.

To czego naprawdę potrzebujemy, to coś pośredniego pomiędzy reprezentacjami (1) i (2). Potrzebny jest nam pewien mechanizm umożliwiający wnioskowanie, że typowe dziecko ma rodziców, ale równocześnie blokujący ten wniosek, jeśli prowadzi on do sprzeczności.

Konstrukcja systemu niemonotonicznego zakłada rozwiązanie dwóch różnych problemów. Można je nazwac problemem formalizacji i problemem weryfikacji hipotez. Pierwszy z nich można

sformułować następująco: określić sposób reprezentacji faktów opisywanego świata oraz zdefiniować zbiór hipotez wyprowadzalnych z tej reprezentacji. Problem weryfikacji hipotez jest to problem reorganizacji dotychczas wyprowadzonych hipotez w przypadku, kiedy niektóre z nich zostają obalone w wyniku uzupełnienia bazy wiedzy o nowy fakt. Wbrew pozorom jest to bardzo skomplikowany problem. Trudność polega na tym, że nie wystarczy usunięcie tych tylko hipotez, które są sprzeczne z nowo wprowadzoną informacją. Należy zdawać sobie sprawę, że każda hipoteza może byc użyta przy wyprowadzaniu innych. Usuwając zatem pewną hipotezę, należy jednocześnie usunąć wszystkie jej konsekwencje. Problem weryfikacji hipotez nie będzie omawiany w niniejszej pracy. Zainteresowanego czytelnika odsyłamy do (Doyle, 1979), (McAllester, 1980), (Doyle, 1982), (Martins, Shapiro, 1984).

Wiele wczesnych systemów konstruowanych na użytek sztucznej inteligencji posiadało wbudowane mechanizmy niemonotoniczne (wiele przykładów można znaleźc w pracy (Winograd, 1980)). Okazało się jednak, że te ad hoc budowane narzędzia, działały poprawnie w najprostszych tylko przypadkach. Ponieważ nie posiadały żadnych podstaw teoretycznych, ich działanie w bardziej skomplikowanych sytuacjach było całkowicie niejasne. Stało się oczywiste, że problem niemonotoniczności wymaga bardziej formalnego podejścia.

Ostatnie lata przyniosły cały szereg prac poświęconych formalizacji różnych mechanizmów niemonotonicznych. Ogromna większośc tych propozycji sformułowano na użytek mode-

lowania wnioskowania naturalnego. Niektóre dotyczą innych zastosowań.

3. Hipoteza domkniętości świata

Hipoteza domkniętości świata (ang. closed world assumption) jest jednym z pierwszych mechanizmów niemonotonicznych. Istnieje kilka wersji tego formalizmu, różniących się szczegółami technicznymi. Wersja, którą przedstawimy, została wprowadzona w pracy (Reiter, 1978). Inna wersja, znana pod nazwą negacja jako porażka (ang. negation as failure), używana jest w programowaniu w logice (zob. (Clark, 1978)).

Niech A będzie formułą atomową klasycznej logiki pierwszego rzędu. Mówimy, że A jest zamknięta, wtedy i tylko wtedy, gdy A nie zawiera zmiennych. W przeciwnym przypadku mówimy, że A jest otwarta. Przykładem zamkniętej formuły atomowej jest formuła "kanarek(Sylwia)". Przykładem otwartej formuły atomowej jest formuła "kanarek(x)". Zamknięta formuła atomowa reprezentuje konkretny fakt zachodzący w opisywanym świecie. Zamknięta formuła atomowa reprezentuje zbiór takich faktów. Zbiór ten zawiera wszystkie fakty, które można otrzymać, wstawiając w miejsca zmiennych konkretne obiekty.

Hipoteza domkniętości świata jest to dodanie do klasycznej logiki pierwszego rzędu następującej reguły wnios-

kowania:

"jeśli, w oparciu o posiadaną informację, nie można dowieść, że zamknięta formuła atomowa jest prawdziwa, należy założyć, że prawdziwa jest jej negacja."

Zastosowanie hipotezy domkniętości świata zilustrujemy przykładem. Rozważmy następującą bazę wiedzy:

(1) pracuje(Jan)

(2) nauczyciel(Piotr)

(3) $\forall x(\text{pracuje}(x) \supset \text{zdrowy}(x))$

Rozważmy zamkniętą formułę atomową "zdrowy(Piotr)". Przy użyciu klasycznej logiki pierwszego rzędu, formuła ta nie może być wyprowadzona z aksjomatów (1)-(3). Stosując zatem do niej hipotezę domkniętości świata, natychmiast wnioskujemy formułę "$\sim$zdrowy Piotr ". Zauważmy, że hipotezy domkniętości świata nie możemy zastosowac dla formuły "zdrowy(Jan)". Wynika to z faktu, że formułę tę można wyprowadzić z aksjomatów (1)-(3).

Hipoteza domkniętości świata jest za silna, aby uznać ją za właściwe narzędzie do modelowania wnioskowania naturalnego. Widac to wyraźnie na powyższym przykładzie. Wnioskowanie, że osoba jest chora, tylko dlatego, że nie można dowieśc, że jest inaczej, jest nie do przyjęcia.

Hipotezę domkniętości świata stosuje się w bazach wiedzy, opisujacych dziedziny, o których nasza wiedza jest zupełna. Możemy wówczas pozwolić sobie na usunięcie z bazy wszystkich

informacji negatywnych, tzn. negacji zamkniętych formuł atomowych, które i tak będzie można wyprowadzic w oparciu o hipotezę domkniętości świata. Ponieważ w przypadku, kiedy nasza wiedza o świecie jest zupełna, informacja negatywna stanowi na ogół ogromny procent całości informacji, uzyskujemy znaczne korzyści pamięciowe. Zauważmy ponadto, że zupełność naszej wiedzy gwarantuje, że hipoteza domkniętości świata nie będzie prowadzić do nieintuicyjnych wniosków. Wszystko bowiem co będziemy mogli przy jej użyciu wywnioskować, to usunięte informacje negatywne.

Niekontrolowane użycie hipotezy domkniętości świata może prowadzić do sprzeczności. Rozważmy następującą bazę wiedzy:

(1) student(Jan)

(2) uczy-się-angielskiego(Jan) $\vee$ uczy-się-niemieckiego(Jan)

Zauważmy, że aksjomaty (1)-(2) nie pozwalają wyprowadzić ani formuły "uczy-się-angielskiego(Jan)", ani też formuły "uczy-się-niemieckiego(Jan)". Stosując zatem hipotezę domkniętości świata, możemy wywnioskować:

(3) $\sim$ uczy-się-angielskiego(Jan)

(4) $\sim$ uczy-się-niemieckiego(Jan)

co natychmiast prowadzi do sprzeczności z (2)

4. Logika defaultów

Logika defaultów (ang. default logic) została wprowadzona w pracy (Reiter, 1980). Służy ona do modelowania jednej z form

wnioskowania naturalnego, zwanej wnioskowanie przez domniemanie (ang. default reasoning). Wnioskowanie przez domniemanie jest to wyprowadzanie wiarygodnych, choć nie absolutnie pewnych, hipotez z niepełnej informacji, przy braku dowodu, że jest inaczej. Charakterystyczną cechą wnioskowania przez domniemanie jest to, że wyprowadzona konkluzja zawsze opisuje sytuację statystyczną lub prototypową.

Typowym przykładem wnioskowania przez domniemanie jest reguła o dzieciach i rodzicach. Jeśli Janek jest dzieckiem, to przy braku informacji, że jest inaczej, wnioskujemy, że ma on rodziców. Hipoteza ta oparta jest na obserwacji, że dzieci na ogół mają rodziców, większość dzieci ma rodziców, lub że typowe dziecko ma rodziców.

W logice defaultów reguła o dzieciach i rodzicach reprezentowana jest przez następującą regułę wnioskowania zwaną defaultem (ang. default):

$$(1) \qquad \frac{\text{dziecko}(x) : M\ \text{ma-rodziców}(x)}{\text{ma-rodziców}(x)}$$

Regułę (1) interpretujemy następująco: dla każdego obiektu x, jeśli x jest dzieckiem i założenie, że x ma rodziców jest niesprzeczne z posiadaną informacją, to należy wnioskować, że x ma rodziców.

Jeśli nasza wiedza o Janku sprowadza się do informacji, że Janek jest dzieckiem, to założenie, że Janek ma rodziców jest niesprzeczne z naszą wiedzą. Stosując zatem regułę (1), możemy wywnioskować, że Janek ma rodziców. Z drugiej strony,

jeśli wiemy, że Janek jest sierotą, to założenie, że Janek ma rodziców jest sprzeczne z naszą wiedzą i reguła (1) nie może być zastosowana.

Niech A będzie formułą logiki pierwszego rzędu zawierającą zmienną x. Mówimy, że x jest zmienna wolna w A, jeśli x nie znajduje się w obrębie działania kwantyfikatora $\forall x$ lub $\exists x$. Przykładowo formuła $P(x) \vee \exists y Q(y,z)$ zawiera dwie zmienne wolne: x oraz z. Dla podkreślenia, że $x_1,\ldots,x_n$ są zmiennymi wolnymi występującymi w formule A, formułę tę będziemy czasami oznaczac $A(x_1,\ldots,x_n)$.

W logice defaultów wiedza o świecie reprezentowana jest przez teorię defaultowa (ang. default theory), tzn. parę składającą się ze zbioru aksjomatów, zapisanych w klasycznej logice pierwszego rzędu, i zbioru defaultów. Aksjomaty reprezentują niepodważalne fakty zachodzące w opisywanym świecie. Defaulty rozszerzają tę wiedzę o wiarygodne, choć niezupełnie pewne hipotezy.

Oto przykład prostej teorii defaultowej składającej się z dwóch aksjomatów i dwóch defaultów:

(2) ptak(Sylwia)

(3) dziecko(Janek)

(4) $$\frac{\text{ptak}(x) : M\ \text{lata}(x)}{\text{lata}(x)}$$

(5) $$\frac{\text{dziecko}(x) : M\ \text{ma-rodziców}(x)}{\text{ma-rodziców}(x)}$$

Należy podkreślić, że w logice defaultów, jak i w innych formalizmach niemonotonicznych, niepodważalne fakty i obalalne hipotezy traktowane są jednakowo. Noszą też wspólną nazwę hipotez.

Formalnie, default jest dowolnym wyrażeniem postaci

$$\frac{A(x_1,\ldots,x_n) : M(B\ x_1,\ldots,x_n)}{C(x_1,\ldots,x_n)} ,$$

gdzie A, B i C są formułami klasycznej logiki pierwszego rzędu zawierającymi zmienne wolne $x_1,\ldots,x_n$. A nazywamy warunkiem wstępnym (ang. prerequisite), B uzasadnieniem (ang. justification), a C wnioskiem (ang. consequent) defaultu. Powyższy default posiada następującą interpretację:

"dla dowolnych obiektów $x_1,\ldots,x_n$, jeśli $A(x_1,\ldots,x_n)$ jest hipotezą i jeśli $B(x_1,\ldots,x_n)$ jest niesprzeczne ze zbiorem hipotez, to $C(x_1,\ldots,x_n)$ należy traktować jako hipotezę."

Default zawierający zmienne wolne nazywamy otwartym. W przeciwnym przypadku zamkniętym. Default o dzieciach i rodzicach jest przykładem defaultu otwartego. Przykładem defaultu zamkniętego jest default:

$$\frac{ptak(Sylwia) : M\ lata(Sylwia)}{lata(Sylwia)}$$

Należy zwrócić uwagę, że o ile defaulty zamknięte są konkretnymi regułami wnioskowania, o tyle defaulty otwarte

reprezentują pewne schematy wnioskowania, które mogą być użyte dla różnych konkretnych obiektów.

Należy rozróżniać pomiędzy defaultem otwartym i jego instancjami. Instancją defaultu otwartego D jest dowolny default zamknięty, powstały z D w wyniku zastąpienia wszystkich zmiennych wolnych nazwami konkretnych obiektów. Na przykład default

$$\frac{\text{ptak(Sylwia)} : \text{M lata(Sylwia)}}{\text{lata(Sylwia)}}$$

jest instancją defaultu

$$\frac{\text{ptak(x)} : \text{M lata(x)}}{\text{lata(x)}}$$

Teorię defaultową zawierającą przynajmniej jeden default otwarty nazywamy teorią otwartą. W przeciwnym przypadku mówimy, że teoria jest zamknięta.

Zbiór wszystkich hipotez wyprowadzalnych z danej teorii defaultowej nazywamy rozszerzeniem (ang. extension) tej teorii. Pojęcie rozszerzenia definiuje się bezpośrednio tylko dla zamkniętych teorii defaultowych. Jeśli A jest otwartą teorią defaultową, to jej rozszerzenie definiuje się jako rozszerzenie zamkniętej teorii A', otrzymanej z A w wyniku pewnej transformacji. Transformacja ta sprowadza się do zastąpienia każdego defaultu otwartego występującego w A, zbiorem jego wszystkich instancji.

Powyższą transformację zilustrujemy przykładem. Rozważmy następującą teorię otwartą:

(6) ptak(Sylwia)

(7) słoń(Klaudiusz)

(8) $$\frac{\text{słoń}(x) : M\ \text{szary}(x)}{\text{szary}(x)}$$

Ponieważ teoria ta zawiera dwa obiekty, możemy utworzyć dwie instancje defaultu (8). A zatem teoria zamknięta odpowiadająca powyższej teorii wygląda następująco:

(9) ptak(Sylwia)

(10) słoń(Klaudiusz)

(11) $$\frac{\text{słoń}(\text{Sylwia}) : M\ \text{szary}(\text{Sylwia})}{\text{szary}(\text{Sylwia})}$$

(12) $$\frac{\text{słoń}(\text{Klaudiusz}) : M\ \text{szary}(\text{Klaudiusz})}{\text{szary}(\text{Klaudiusz})}$$

Należy zwrócić uwagę, że warunek wstępny defaultu (11) nie jest hipotezą. A zatem default ten nie będzie mógł być uaktywniony. Tym niemniej jest on intuicyjnie akceptowalny. Gdyby Sylwia była słoniem, to fakt, że Sylwia jest szara, byłby zupełnie wiarygodną hipotezą.

Przechodzimy do podania definicji rozszerzenia dla teorii defaultowych. Ze względu na powyższą transformację, możemy ograniczyć się do teorii zamkniętych. Mówiąc niedokładnie,

rozszerzenie teorii defaultowej powinno zawierać wszystkie hipotezy wyprowadzalne z niej bądź to przy użyciu logiki klasycznej, bądź to w wyniku stosowania defaultów. Sugeruje to następującą definicję.

Niech A =(W,D) będzie zamkniętą teorią defaultową, składającą się ze zbioru aksjomatów W i zbioru defaultów D. Zbiór formuł E jest rozszerzeniem teorii A, wtedy i tylko wtedy, gdy E jest minimalnym zbiorem spełniającym następujące warunki:

(R1) $W \subseteq E$

(R2) $E = Th(E)$

(R3) Dla każdego defaultu $(A: MB / C) \in D$, jeśli $A \in E$ oraz $\sim B \notin E$, to $C \in E$.

Warunek (R1) stwierdza, że każdy aksjomat jest hipotezą. Warunek (R2) stwierdza, że zbiór hipotez jest zamknięty ze względu na strukturę dedukcyjną klasycznej logiki pierwszego rzędu. Warunek (R3) stwierdza, że hipotezą jest konsekwencja każdego, możliwego do zastosowania defaultu.

Powyższa definicja ma charakter czysto ilustracyjny. Jest to uproszczenie definicji oryginalnej, która posługuje się pewnym operatorem algebraicznym i wykorzystuje pojęcie punktu stałego tego operatora (zob. (Reiter, 1980)).

Należy podkreślić, że istnieją teorie defaultowe posiadające więcej niż jedno rozszerzenie. Oto przykład takiej teorii:

(13) republikanin(Smith)

(14) kwakier(Smith)

(15) $$\frac{\text{republikanin(Smith)} : M \sim \text{pacyfista(Smith)}}{\sim \text{pacyfista Smith}}$$

(16) $$\frac{\text{kwakier(Smith)} : M\ \text{pacyfista(Smith)}}{\text{pacyfista(Smith)}}$$

Zauważmy, że posiadana wiedza pozwala nam zastosować zarówno default (15) jak i (16). Z drugiej strony, użycie któregokolwiek z nich blokuje zastosowanie pozostałego. Dlatego powyższa teoria posiada dwa rozszerzenia. Jedno z nich zawiera hipotezę, że Smith jest pacyfistą, drugie hipotezę przeciwną.

W przypadku kiedy teoria defaultowa posiada więcej niż jedno rozszerzenie, każde z nich traktowane jest jako alternatywny zbiór hipotez zachodzących w opisywanym świecie.

Istotną wadą formalizmu Reitera jest to, że istnieją teorie defaultowe nie posiadające rozszerzenia. Najprostszym, choć sztucznym przykładem, jest teoria składająca się wyłącznie z pojedyńczego defaultu

$$\frac{: M \sim \text{pada-deszcz}}{\text{pada-deszcz}}$$

Brak warunku wstępnego oznacza, że jest on nieistotny dla stosowalności defaultu. Przykład intuicyjnie naturalnej teorii defaultowej, która nie posiada rozszerzenia, znaleźć można w pracy (Łukaszewicz, 1984). W pracy tej podana jest

również alternatywna formalizacja logiki defaultów. Istotną zaletą proponowanego podejścia jest fakt, że każda teoria defaultowa posiada przynajmniej jedno rozszerzenie.

Pewne problemy związane z reprezentacją wiedzy przy użyciu logiki defaultów dyskutowane są w (Reiter, Criscuolo, 1981) i (Etherington, Reiter, 1983).

Semantyka logiki defaultów omawiana jest w pracy (Łukaszewicz, 1985).

Zastosowaniu logiki defaultów do przetwarzania języka naturalnego poświęcone są prace (Mercer, Reiter, 1982), (Dunin-Kęplicz, 1984), (Joshi, Webber, Weischedel, 1984).

5. Inne formalizmy niemonotoniczne

Hipoteza domkniętości świata i logiki defaultów nie wyczerpują wszystkich możliwych podejśc do problemu formalizacji wnioskowania niemonotonicznego.

Jednym z ciekawszych formalizmów niemonotonicznych jest logika niemonotoniczna (ang. nonmonotonic logic). Pierwszy system logiki niemonotonicznej podany został w pracy (McDermott, Doyle, 1980). W logice niemonotonicznej defaulty reprezentowane są bezpośrednio w języku. Aby to było możliwe, język logiki niemonotonicznej zawiera specjalny operator modalny M interpretowany "niesprzeczne jest, że". W logice niemonotonicznej regułę o dzieciach i rodzicach można zapisac w postaci następującego aksjomatu:

$\forall x(\text{dziecko}(x) \wedge M\ \text{ma-rodziców}(x) \supset \text{ma-rodziców}(x))$

Pewne ogólne zasady związane z konstruowaniem logiki niemonotonicznej podane zostały w pracy (Łukaszewicz, 1983). Konkretne systemy logiki niemonotonicznej znaleźć można w pracach (McDermott, 1982), (Moore, 1983), (Łukaszewicz, 1984a), (Gabbay, 1982), (Marek, 1984), (Akama, 1985).

Bardzo silnym mechanizmem niemonotonicznym jest otaczanie (ang. circumscription). Mechanizm ten wprowadzony został w pracy (McCarthy, 1980) i rozszerzony w (McCarthy, 1984). Otaczanie zakłada reprezentację świata w języku logiki klasycznej. Niemonotoniczną strukturę dedukcyjną uzyskuje się na drodze czysto syntaktycznej, w wyniku dołączenia pewnej formuły klasycznej logiki drugiego rzędu.

Główną wadą otaczania jest fakt, że jego niekontrolowane użycie może prowadzić do nieakceptowalnych hipotez. Co więcej, dość mało na razie wiadomo, jak formalizmu tego używać.

Pewne ogólne własności otaczania znaleźc można w pracach (Doyle, 1984), (Lifschitz, 1984), (Lifschitz, 1985), (Etherington, Mercer, Reiter, 1984).

Innym podejściem do formalizacji wnioskowania niemonotonicznego jest rozszerzenie logiki standardowej o pewne mechanizmy metalogiczne. Podejście to zostało zaproponowane w pracy (Weyhrauch, 1980). Zobacz także (Bowen, Kowalski, 1982) i (Attardi, Simi, 1984).

Formalizacja wnioskowania przez domniemanie przy użyciu trójwartościowej logiki monotonicznej zaproponowana została w pracy (Nutter, 1980).

Pełna bibliografia dotycząca wnioskowania niemonotonicznego, do roku 1984, znajduje się w pracy (Perlis, 1984).

Porównanie różnych podejść do wnioskowania niemonotonicznego, z punktu widzenia ich adekwatności do formalizacji wnioskowania naturalnego, znaleźć można w pracy (Łukaszewicz, 1985a).

Bibliografia

Akama, S. (1985) Formalization of Nonmonotonic Logic from the Standpoint of Constructive Logic. Fujitsu Ltd., Japonia, maszynopis.

Attardi, G., Simi, M. (1984) Metalanguage and Reasoning Across Viewpoints. Proc. 6-tej konfer. ECAI, Pisa-1984, str. 315-324.

Bowen, K. A., Kowalski, R. (1982) Amalgamating Language and Metalanguage in Logic Programming. W: Clark, K. L., Tarnlund, S. A. (wyd.), Logic Programming, A.P.I.C. Studies in Data Processing No 5, Academic Press.

Clark, K. L. (1978) Negation as Failure. W: Gallaire, H., Minker, J. (wyd.), Logic and Databases, Plenum Press, New York, str. 293-322.

Doyle, J. (1979) A Truth Maintenance System. Artificial Intelligence, No 12, str. 231-272.

Doyle, J. (1982) Some Theories of Reasoned Assumptions: An Essay in Rational Psychology. Carnegie-Mellon University, Raport.

Doyle, J. (1984) Circumscription and Implicit Definability. AAAI Workshop on Non-Monotonic Reasoning, New York-1984, str. 57-69.

Dunin-Kęplicz, B. (1984) Default Reasoning in Anaphora Resolution. Proc. 6-tej konf. ECAI, Pisa-1984. str. 157-166.

Etherington, D., Reiter, R. (1983) On Inheritance Hierarchies with Exceptions. Proc. konf. AAAI-83, str. 104-108.

Etherington, D., Mercer, R., Reiter, R. (1984) On the Adequacy of Predicate Circumscription for Closed-World Reasoning. AAAI Workshop on Non-Monotonic Reasoning, New York-1984, str. 70-81.

Gabbay, D. (1982) Intuitionic Basis for Non-Monotonic Logic. Proc. 6-tej konf. CAD, Lecture Notes in Computer Science, Vol. 139, Springer, str. 260-273.

Joshi, A., Webber, B., Weischedel, R. (1984) Default Reasoning in Interaction. AAAI Workshop on Non-Monotonic Reasoning, New York-1984, str. 144-150.

Lifschitz, V. (1984) Some Results on Circumscription. AAAI Workshop on Non-Monotonic Reasoning, New York-1984, str. 151-164.

Lifschitz, V. (1985) Computing Circumscription. Proc. 9-tej konf. IJCAI, Los Angeles, 1985, str. 121-127.

Łukaszewicz, W. (1983) General Approach to Nonmonotonic Logics. Proc. 8-ej konf. IJCAI, Karlsruhe, 1983, str. 352-354.

Łukaszewicz, W. (1984) Considerations on Default Logic. AAAI Workshop on Non-Monotonic Reasoning, New York-1984, str. 165-193.

Łukaszewicz, W. (1984a) Nonmonotonic Logic for Default Theories. Proc. 6-tej konf. ECAI, Pisa-1984, str. 305-314.

Łukaszewicz, W. (1985) Two Results on Default Logic. Proc. 9-tej konf. IJCAI, Los Angeles-85, str. 459-461.

Łukaszewicz, W. (1985a) Formalization of Knowledge and Ignorance. Ukaże się w: The Journal for the Integrated Study of Artificial Intelligence Cognitive Science and Applied Epistemology.

Marek, W. (1984) A Natural Semantics for Modal Logic over Databases and Model-Theoretic Forcing I. AAAI Workshop on Non-Monotonic Reasoning, New York-1984, str. 194-240.

Martins, J., Shapiro, S. (1984) A Model for Beliefs Revision. AAAI Workshop on Non-Monotonic Reasoning, New York-1984, str. 241-294.

McAllester, D. (1980) An Outlook on Truth Maintenance. Raport 551, MIT AI LAB.

McCarthy, J. (1980) Circumscription - A Form of Non-Monotonic Reasoning. Artificial Intelligence, No 13, str. 27-39.

McCarthy, J. (1984) Applications of Circumscription to Formalizing Common Sense Knowledge. AAAI Workshop on Non-Monotonic Reasoning, New York-1984, str. 295-324.

McDermott, D. (1982) Non-Monotonic Logic II: Non-Monotonic Modal Theories. Journal of the ACM, No 29/1, str. 33-57.

McDermott, D., Doyle, J. (1980) Non-Monotonic Logic I. Artificial Intelligence, No 13, str. 41-72.

Mercer, R., Reiter, R. (1982) The Representation of Presuppositions using defaults. University of British Columbia, Raport 82-1.

Moore, R. (1983) Semantical Considerations on Non-Monotonic Logic. Proc. 8-ej konf. IJCAI, Karlsruhe, 1983, str. 272-279. Także w: Artificial Intelligence, No 25/1, str. 75-94.

Nutter, J. (1983) Default Reasoning Using Monotonic Logic: A Modest Proposal. Proc. AAAI-83, str. 297-300.

Perlis, D. (1984) Bibliography of Literature on Non-Monotonic Reasoning. AAAI Workshop on Non-Monotonic Reasoning, New York-1984, str. 396-401.

Rēiter, R. (1978) On Closed World Databases. W: Gallaire, H., Minker, J. wyd. , Logic and Databases, Plenum, str. 55-76.

Reiter, R. (1980) A Logic for Default Reasoning. Artificial Intelligence, No 13, str. 81-132.

Reiter, R., Criscuolo, G. (1981) On Interacting Defaults. Proc. 7-ej konf. IJCAI, Vancouver, 1981, str. 270-276.

Weyhrauch, R. (1980) Prolegomena to a Theory of Mechanized Formal Reasoning. Artificial Intelligence, No 13, str. 133-170.

Winograd, T. (1980) Extended Inference Modes in Reasoning by Computer Systems. Artificial Intelligence, No 13, str. 5-26.

AKTUALNE TRENDY W MODELOWANIU KONCEPTUALNYM

Stanisław Wrycza
Uniwersytet Gdański

Opracowanie poświęcone jest bieżącym tendencjom w konstruowaniu schematów konceptualnych, stanowiących stabilne, niezależne od stosowanej technologii informatycznej, sformalizowane opisy dziedziny przedmiotowej. Po przedstawieniu założeń wprowadzających, scharakteryzowano podejścia do modelowania konceptualnego a następnie dokonano porównań szeregu narzędzi definiowania i użytkowania schematów konceptualnych. W ostatniej części zarysowano proces modelowania konceptualnego. Rozważania te mają charakter ogólny, wprowadzający w zakresie poszczególnych trendów. Przedstawione w opracowaniu wnioski i przykłady zastosowań są wynikiem badań przeprowadzonych w organizacjach gospodarczych.

1. Wprowadzenie

Modelowanie konceptualne w ostatnich kilku latach odgrywa coraz bardziej znaczącą rolę w dziedzinie baz danych DAT-81 i systemów informacyjnych i to zarówno w rozwoju teorii, zastosowań jak i prac standaryzacyjnych. Najogólniej modelowanie konceptualne to proces tworzenia modelu czy schematu konceptualnego. Z kolei model konceptualny to sformalizowany opis statyki i dynamiki bazy danych reprezentującej wybraną dziedzinę przedmiotową. W związku z różnymi metodami formalizacji opisu, w opracowaniu niniejszym wprowadza się, podobnie jak w niektórych cytowanych publikacjach, rozróżnienie pomiędzy modelem a schematem konceptualnym. Ten ostatni stanowi zbiór niesprzecznych zdań wyrażających niezbędne twierdzenia /definicje/ o dziedzinie przedmiotowej. Narzędziem definiowania schematu jest język opisu schematu konceptualnego.

Dotychczasowy rozwój teorii i praktyki modelowania konceptualnego pozwala na określenie wymagań, które winny spełniać narzędzia /ang: facilities/ tworzenia modeli i schematów jak metodyki, metody i techniki, języki opisu, języki zapytań, systemy wspomaganego komputerem modelowania.
Dla spełnienia swej roli narzędzia te winny posiadać następujące założenia i cechy [JAR-84, s.7]:

a/ Podstawowe pojęcia dogodne dla opisu statycznych i dynamicznych aspektów dziedziny przedmiotowej

b/ Język precyzyjnego komunikowania schematu konceptualnego

systemowi komputerowemu

c/ Język opisu schematu konceptualnego zrozumiały dla użytkownika

d/ Możliwość łatwych zmian schematu konceptualnego, odzwierdiedlających zmiany w dziedzinie przedmiotowej lub

e/ Niesprzeczność schematów zewnętrznych poszczególnych użytkowników z odpowiednim schematem konceptualnym

f/ Niezależność schematu konceptualnego od zmian schematu wewnętrznego.

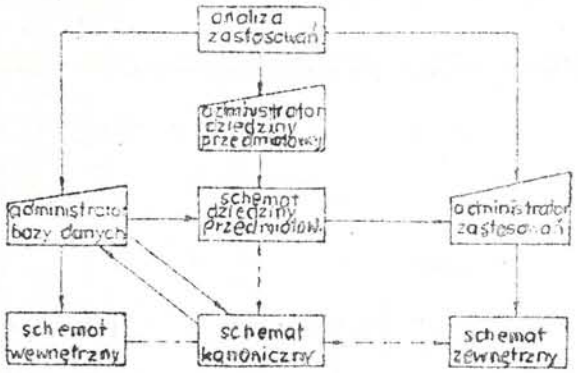
Powyższe wymagania wpłynęły na treść i układ niniejszego opracowania.

Zainicjowane raportem ANSI/X3/SPARC [TSI-78] prace badawcze zaowocowały licznymi propozycjami modelowania konceptualnego o charakterze metodycznym, teoretycznym czy implementacyjnym. Pojawiło się duże zapotrzebowanie aktualnych i potencjalnych użytkowników baz danych i szerzej - systemów informatycznych na różnorodne narzędzia, konstruowania modeli i schematów a w następstwie na teorię modelowania konceptualnego.

Doskonalona jest struktura SZBD w stosunku do pierwotnej wersji TSI-78 . W tym zakresie istotną propozycję stanowi rozszerzony czteroschematowy model architektury baz danych De, Haseman'a i So [DE-81] /schemat 1/. Dokonano w nim podziału schematu konceptualnego na schemat dziedziny przedmiotowej /ang.enterprise schema/ i schemat kanoniczny /ang. canonical schema/.

Schemat dziedziny przedmiotowej jest "czystym" opisem wycinka rzeczywistości, niezależnym od uwarunkowań pamięci i efektywności, konceptualnym opisem zawartości bazy danych. Schemat kanoniczny

Schemat 1



Czteropoziomowa architektura SZBD
Źródło: [DE-81, s.120]

stanowi realizację schematu dziedziny przedmiotowej w kategoriach struktur danych. Rozróżnienie wiąże się z oddzieleniem dwu istotnych zadań schematu konceptualnego - sematycznego opisu wiążącego bazę danych z dziedziną podmiotową oraz opisu składniowego działającego jako sprzęg między użytkownikiem /schemat zewnętrzny/ a pamięcią /schemat wewnętrzny/. Architektura SZBD zaprezentowana na schemacie 1, choć nie uzyskała dotąd szerszego uznania, w znacznie większym zakresie uwzględnia rozwój teorii modelowania konceptualnego a zwłaszcza języków opisu schematu konceptualnego.

2. Podejścia do modelowania konceptualnego

2.1. Klasyfikacja podejść

Opracowano szereg systematyk modeli konceptualnych o różniących się zestawach autorów i listach kryteriów oceny. Dokonali

tego m.in. Biller i Neuhold [BIL-77] ,Kerschberg, Klug i Tsichritzis [KER-77] , Olivé [OLI-83] oraz Kung [KUN-83]. Istota tych analiz porównawczych sprowadza się do opracowania tablic krzyżowych, w których podstawowymi parametrami są: kategorie pojęciowe, poziomy uogólnienia w procesie modelowania, podstawy matematyczne,realizacja pożądanych cech i ról modeli oraz schematów konceptualnych w poszczególnych ujęciach.

Jak wynika z dokonanych porównań większość podejść ukierunkowana jest na modelowanie statyki dziedziny przedmiotowej przy pomocy trzech kategorii : obiektu /typu obiektu/, atrybutu i relacji /typu relacji/.[1/] Najnowszym kierunkiem w dziedzinie konstruowania modeli konceptualnych są abstrakcje danych / lub abstrakcje baz danych/. Większość podejść pomija lub traktuje marginalnie zagadnienia opisu dynamiki dziedziny przedmiotowej

---

1/ W literaturze przedmiotu przyjmuje się często, że pojęcia statyki - obiekt, relacja i atrybut to pojęcia pierwotne. W niniejszym opracowaniu zrezygnowano z szerszej dyskusji terminologicznej, zawartej m.in. w [NIE-76] , przyjmując,iż:
- obiekt to jednoznacznie identyfikowalny składnik badanej dziedziny przedmiotowej
- relacja - związek między dwu lub więcej obiektami
- atrybut - charakterystyka obiektu lub relacji.

Dynamikę dziedziny przedmiotowej opisuje się przy pomocy funkcji, zdarzeń i operacji, gdzie:
- funkcja oznacza czynność, wykonywanie, zakres funkcjonowania jaki przypada w udziale danej organizacji jako całości albo jakiejść jej części [PSZ-78],
- zdarzenie to bodziec oddziałujący na dziedzinę przedmiotową i inicjujący jedną lub więcej funkcji i /lub operacji/,
- operacja jest odpowiednikiem elementarnej czynności tworzenia, skreślania, modyfikacji, wyszukiwania wykonywanych na obiektach relacjach i atrybutach schematu konceptualnego.

oraz specyfikacji więzów integralności. [1] Różnice pomiędzy poszczególnymi podejściami nie zawsze są znaczące - niektóre modele używają różnej terminologii dla opisu tych samych pojęć; w innych modelach ta sama terminologia jest używana dla różnych pojęć.

Sytuacja ta skłoniła ISO do podjęcia prac o charakterze standaryzacyjnym. Międzynarodowa grupa badawcza ISO TC97/SC5/WG3 wyodrębniła trzy typy podejść do konstruowania schematu konceptualnego [GRI-82] :

- Obiekt - Atrybut - Relacja /OAR/
- Binarnych i N - arnych Relacji Elementarnych /B/
- Rachunek Predykatów /RP/

Każde z tych pojęć zostało poddane szczegółowej analizie we wspomnianym opracowaniu. Zaproponowano dla każdego z nich projekt standardu języka opisu schematu konceptualnego. Wydaje się, iż w świetle najnowszych tendencji należy wprowadzić kolejny rodzaj modelu konceptualnego:

- Abstrakcje Danych /AD/.

Model ten wywodzi się z poprzednich podejść, zwłaszcza OAR i

---

[1] Zachowanie integralności bazy danych oznacza dążenie do przechowywania jedynie poprawnych danych. Więzy integralności [LEN-83, s.530] to reguły dotyczące poszczególnych składników schematu konceptualnego, odzwierciedlające logiczne związki i ograniczenia między elementami dziedziny przedmiotowej. Więzy integralności można wyodrębnić ze względu na różne kryteria:
- pojęcia, których dotyczą: obiektów, reakcji, atrybutów i wartości
- czasu oddziaływania : odroczone i natychmiastowe
- dynamiki: statyczna i przejścia
W niektórych ocenach [NIJ-80], 80% zawartości schematu konceptualnego winien zajmować opis więzów integralności. Należy zaznaczyć, że jedynie część reguł integralności jest wyrażona dosłownie, natomiast część jest zintegrowana z opisem innych składników schematu konceptualnego.

Binarnego. Jednak podczas gdy trzy pierwsze podejścia umożliwiają w zasadzie modelowanie statyki Abstrakcje Danych uwzględniają możliwości specyfikacji funkcji i zdarzeń. Poniżej przedstawiono uwagi na temat poszczególnych rodzajów modeli.

2.2. Podejście Obiekt - Atrybut - Relacja

Podstawowe konstrukcje modelowania, wyprowadzone z opracowań Chena [CHE-77] i Bachmana [BAC-69] to obiekt, atrybut i relacja /OAR/. Ich sens zawiera się w definicjach przytoczonych uprzednio. Istnieją różne warianty tego podejścia zaproponowane przez innych autorów. Poza podstawowymi konstrukcjami podejścia OAR, używa się zawsze określonej notacji graficznej jako narzędzia tworzenia diagramów modeli konceptualnych a następnie środka interakcji między użytkownikiem a administratorem dziedziny przedmiotowej czy bazy danych. Dozwolone są tu n-arne typy relacji, jak również przyporządkowanie atrybutów relacjom. Istotnym w omawianym sposobie modelowania jest rozróżnienie pomiędzy obiektem, wartością i relacją a typem obiektu, atrybutem i typem relacji. W języku opisu schematu konceptualnego typu OAR zaproponowanym przez ISO TC97/SC5/WG3 pojęcia te przybierają postać nazw typów obiektów, nazw atrybutów i nazw typów relacji. Główne trzy części tego języka to identyfikacja schematu konceptualnego /nazwa/ oraz opisy: typów obiektów i typów relacji. Opis typów obiektów obejmuje określenie nazwy typu obiektu, jego identyfikatora oraz nazw atrybutów. Podobnie część typów

relacji specyfikuje nazwy typów relacji, ilość związanych typów obiektów, ich nazwy, funkcjonalną zależność między nimi, maksymalną i minimalną liczebność typów obiektów w typie relacji, identyfikator oraz atrybuty typu relacji. Schemat 2 stanowi diagram typu OAR opisany poniżej w adekwatnym języku opisu schematu konceptualnego.

Istniejące metody modelowania bazy danych przy wykorzystaniu podejścia OAR można ogólnie podzielić na dwie grupy. Pierwsza z nich to modelowanie bezpośrednie /modelowanie ad hoc/, w którym twierdzenia o dziedzinie przedmiotowej pozwalają na identyfikację typów obiektów i typów relacji. Druga metoda /modelowanie strukturalne/ polega na specyfikacji atrybutów dziedziny przedmiotowej a następnie konstruowaniu poszczególnych typów obiektów i typów relacji poprzez analizę atrybutów. W przypadku ostatnim istnieją możliwości automatycznego wykonywania tych czynności [OAR-80] dzięki stosowaniu odpowiednich algorytmów. Decyzja o wyodrębnieniu atrybutu czy typu obiektu podejmowana jest w trakcie iteracyjnego procesu modelowania. W odniesieniu do n-arnych relacji stosowana jest dekompozycja umożliwiająca tworzenie relacji niższego stopnia.

Schemat 2

Diagram schematu konceptualnego
Źródło: Opracowanie własne

Opis diagramu /schematu 2/ w języku OAR przedstawia się następująco:

```
CONCEPTUAL SCHEMA DOSTAWA-TOWARU
ENTITY-TYPE ZAPAS
IDENTIFIER SYMBOL-TOWARU
DESCRIPTION SYMBOL-TOWARU
            ZAPAS-MAGAZYNOWY
            KOSZTY-ZAPASÓW
            ZAPAS-POTENCJALNY
            BIEŻĄCA-SPRZEDAŻ
            BIEŻĄCE-DOSTAWY
            BIEŻĄCE-ZAMÓWIENIA
            KOSZTY-ZAPASÓW
            NORMA-ZAPASU
ENTITY-TYPE DOSTAWCA
IDENTIFIER SYMBOL-DOSTAWCY
DESCRIPTION SYMBOL-DOSTAWCY
            NAZWA-DOSTAWCY
            ADRES-DOSTAWCY
            ILOŚĆ-DOSTARCZONYCH-ART.
            WARTOŚĆ-ZAMÓWIENIA
ENTITY TYPE MAGAZYN
IDENTIFIER NR-MAGAZYNU
DESCRIPTION NR-MAGAZYNU
            NAZWA-MAGAZYNU
            POWIERZCHNIA-SKŁADOWA
            ZATRUDNIENIE
RELATIONSHIP-TYPE DOSTAWA
DIMENSION      3
COLLECTION     ZAPAS
               DOSTAWCA
               MAGAZYN
CARDINALITY    ZAPAS        1,1
               DOSTAWCA     1,n
               MAGAZYN      1,n
FUNCTIONAL DETERMINANCY
               MAGAZYN ON ZAPAS
IDENTIFIER NR-DOSTAWY
DESCRIPTION NR-DOSTAWY
            SYMBOL-TOWARU
            ILOŚĆ-TOWARU
            DATA-DOSTAWY
```

2.3. Podejście Binarne

Początki modelowania binarnego wiążą się przede wszystkim z pracami Bracchi [BRA-76], Nijsena [NIJ-76] i Falkenberga [FAL-76]. Cechy podejścia czynią go właściwym narzędziem użytkownika SZBD na poziomie schematu konceptualnego. Wspólną cechą różnych sposobów binarnego modulowania konceptualnego jest odzwierciedlenie dziedziny przedmiotowej poprzez ogół występujących w nim binarnych relacji tj. zdań, których tylko dwa składniki odgrywają rolę. Z punktu widzenia analityka dziedzina przedmiotowa jest w każdym momencie czasu opisywana przez zbiór obiektów i binarnych relacji między nimi. Znaczące jest tu odróżnienie obiektów od nazw obiektów. Ostatecznie podstawowymi konstrukcjami binarnego modelowania są obiekty, nazwy obiektów i binarne relacje. Do tych podstawowych konstrukcji odnoszą się pojęcia typu i wystąpienia. Rozróżnienie rzeczy od ich nazw powoduje identyfikację w opisie konceptualnym obiektów niesłownikowych /ang. NOLOTS: non - lexical objects/ oraz obiektów słownikowych /ang. LOTS: lexical objects/. Tak więc obiekty niesłownikowe odpowiadają obiektom w podanym uprzednio znaczeniu, natomiast obiekty słownikowe nazwom obiektów. Identyfikacja obiektów /NOLOTs/ następuje przez nazwy obiektów /LOTs/. Zarówno słownikowe jak i niesłownikowe typy obiektów mogą być dzielone na podtypy.

Odpowiednio do wyróżnionych typów obiektów istnieją trzy typy relacji binarnych:

a/ typy pojęć /ang. types/ stanowiące [illegible]
powiązania między obiektami niesłownikowymi,

b/ typy powiązań /ang. bridge types/ wiążące obiekty słownikowe z określonymi przez nie obiektami niesłownikowymi,

c/ typy wyrażeń /ang. phrase types/ opisujące relacje tylko między obiektami słownikowymi.

Schemat konceptualny opisywany binarnie zawiera również statyczne i dynamiczne ograniczenia, będące zestawem niezbędnych twierdzeń o obiektach i relacjach. Ograniczenia jak również relacje i obiekty rejestrowane są w schemacie konceptualnym za pomocą odpowiednich operacji wprowadzenia, skreślenia, łączenia i rozłączania obiektów do i z podtypów, powiązania obiektów, wykonywania ciągu operatorów, listowania. Składnia binarnego języka schematu konceptualnego zawiera wszystkie omówione konstrukcje, a więc opis obiektów słownikowych i niesłownikowych oraz relacji pojęć, powiązań i wyrażeń. Schemat ∑ stanowi diagram binarnego modelu konceptualnego. Opis diagramu przy użyciu binarnego języka opisu schematu konceptualnego przedstawia się następująco:

```
CONCEPTUAL SCHEMA called GOSPODARKA TOWAROWA
NOLOT called {'DOSTAWCA' 'DOSTAWA-TOWARU' 'TOWAR' 'MAGAZYN'
              'IMPORTER' 'PRODUCENT'}
LOT called {'SYMBOL' 'ADRES' 'NAZWA' 'SYMBOL-TOWARU' 'ILOŚĆ'
            'DATA'}
NOLOT called {'PRODUCENT' 'IMPORTER'}
             is subtype of NOLOT called 'DOSTAWCA'
IDEA with-first ROLE (called 'stanowi'
                      and on NOLOT called 'TOWAR')
          and with-second ROLE (called 'obejmuje'
          is called ('przesyłka' and on NOLOT called
                     'DOSTAWA-TOWARU')
Uwaga: Pomija się dalsze cztery deklaracje typu IDEA
BRIDGE with-first ROLE (called 'ma'
            and on NOLOT called 'MAGAZYN')
            and with-second ROLE (called 'jest'
                  and on LOT called 'ADRES')
            is called 'położenie'
Uwaga: Pomija się dalsze cztery deklaracje typu BRIDGE
CONSTRAINT identyfikacja - dostawy
is declared as
   (DATA of DOSTAWA-TOWARU and ILOŚĆ of DOSTAWA-TOWARU
   and SYMBOL-TOWARU of TOWAR is unique);
CONSTRAINT podtypy-dostawcy
is declared as
     DOSTAWCA is equal to (PRODUCENT union IMPORTER);
CONSTRAINT rozłączność-podtypów
is declared as
     (PRODUCENT intersection IMPORTER is empty).
```

Schemat 3

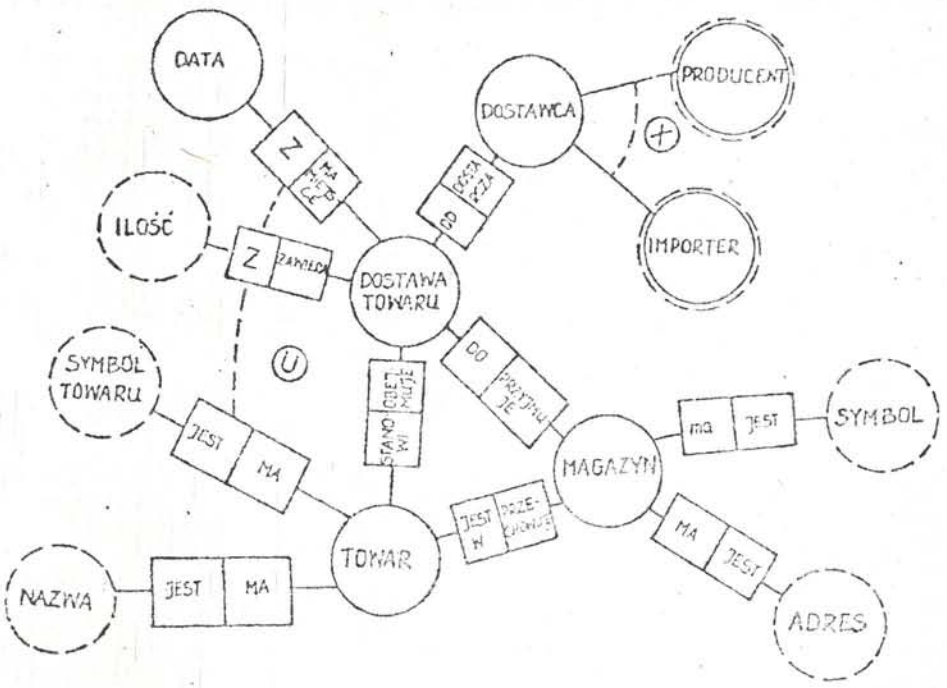


Diagram binarnego schematu konceptualnego
Źródło: Opracowanie własne

W trakcie procesu modelowania binarnego schematu konceptualnego projektant wykonuje trzy zadania: ustala konwencje nazw dla obserwowanych w dziedzinie przedmiotowej obiektów, klasyfikuje je oraz przygotowuje schemat konceptualny. W tym podejściu schemat konceptualny głównie wprowadza klasyfikację i uporządkowanie w opisie dziedziny przedmiotowej natomiast nie jest wyczerpującym opisem na poziomie wystąpień obiektów. Proces modelowania konceptualnego w układzie relacji binarnych, mimo prób "algorytmizacji" jest heurystyczny. Jego punktem wyjścia jest nieformalna percepcja dziedziny przedmiotowej. Po wyborze niezbędnych twierdzeń opisu dziedziny następuje wyodrębnienie obiektów słownikowych i niesłownikowych, binarnych relacji między nimi, ograniczeń a następnie graficzny opis modelu konceptualnego oraz rejestracja w języku opisu schematu. W wyniku usuwania wewnętrznych niezgodności i braków w opisie w kolejnych iteracjach dochodzi się do pożądanego poziomu jakości schematu.

2.4. Rachunek Predykatów

W przypadku logiki rachunku predykatów postrzegana dziedzina przedmiotowa składa się wyłącznie z obiektów, których dotyczą twierdzenia. Schemat konceptualny stanowi opis składający się ze zbioru zdań zakodowanych w języku formalnym opartym o reguły logiki formalnej [STE-80]. Zdania te są złożone z:

- nazw i zmiennych,
- predykatów,

- łączników logicznych,
- kwantyfikatorów.

Nazwy i zmienne dotyczą obiektów w dziedzinie przedmiotowej, a zadania wyrażają twierdzenia o tych obiektach.

Oto przykład zapisu zdań z wykorzystaniem rachunku dla każdego towaru x:

Jeżeli wartość zapasu magazynowego m/x/ obniży się poniżej zapasu zamówieniowego z/x/ należy złożyć zamówienie t/x/ w przeciwnym wypadku zamówienia nie składać.

$$\forall x\ [m/x/ < z/x/ \longrightarrow t/x/] \wedge \forall x\ [m/x/ \geqslant z/x/ \longrightarrow \sim t/x/]$$

Podstawowe zasady podejścia są jednakowo dobrze podatne na modelowanie zarówno statycznych jak i dynamicznych aspektów dziedziny przedmiotowej. Zasady logiki formalnej są szczegółowo omówione w odpowiednich opracowaniach z tego zakresu /np. [MAŁ-70] czy [PAS-80]/, toteż nie są tu bliżej opisywane.

3.5. Abstrakcje Danych

Ten rodzaj modeli nie został wyróżniony w raporcie [GRI-82], choć jest przedmiotem szczegółowych analiz w opracowaniach [BRO-80] i [DE-82]. Inicjatorami konstruowania abstrakcji danych byli Smith'owie /[SMI-77a], [SMI-77b]/. Omawiane podejście znalazło uznanie w implementacjach języków opisu schematu konceptualnego /por. punkt 3.1./.

Zasada tworzenia abstrakcji danych [BRO-80, s.102] polega na usuwaniu nieistotnych cech /atrybutów/ analizowanego obiektu i akceptowaniu cech adekwatnych do aktualnego kontekstu, wynikającego z analizy dziedziny przedmiotowej bądź analizy potrzeb. Smith'owie wyróżnili trzy rodzaje abstrakcji danych: klasyfikację [SMI-80], agregację [SMI-77a] i uogólnianie [SMI-77b]. W wyniku stosowania tych metod każdy wyodrębniony w dziedzinie przedmiotowej obiekt jest albo agregatem albo typem obiektu rodzajowego. Skrótowo istotę poszczególnych metod w typowaniu obiektów można przedstawić następująco w zależnościach:

- klasyfikacja: zbiór wystąpień —→ nowy typ,
- agregacja: związek między typami —→ nowy typ /agregat/,
- uogólnianie: zbiór typów /obiektów składnikowych/ —→ nowy typ /obiektu rodzajowego/.[1/]

Na ogół przyjmuje się agregację i uogólnianie jako podstawowe rodzaje abstrakcji. Odpowiadają im abstrakcje danych: agregaty i obiekty rodzajowe. Są one dość powszechnie stosowane w różnych podejściach do modelowania konceptualnego.

Agregacja polega na tworzeniu agregatu jako związku między różnymi typami obiektów. Agregat jest więc rozumiany jako zbiór typów obiektów traktowanych jak pojedyńczy typ obiektu. Pomijane są w tym przypadku niektóre cechy składników a podkreślone cechy

---

1/Brodie i Silva wprowadzają w systemie ACM/FCM [BRO-82] podobny podział określając klasyfikację mianem asocjacji. Atzeni i inni [ATZ-82], [ATZ-83] wyróżniają związki uogólniania i podzbioru typu obiektu w ramach innego typu obiektu. Natomiast dos Santos i inni [SAN-80] określają 3 typy związków między typami obiektów: suma /odpowiednik obiektu rodzajowego/, produkt /agregat/ i zależność /podzbiór/.

agregatu jako całości, np. związek:

ODBIORCA zamawia TOWAR u DOSTAWCY

tworzy nowy typ obiektu /agregat/:

ZAMÓWIENIE

W procesie uogólniania następuje grupowanie typów obiektów w hierarchiczne zależności - konstruowanie nowych typów obiektów rodzajowych. Oznacza to eliminację różnic pomiędzy kategoriami i podkreślenie cech wspólnych np. zbiór:

SPRZEDAWCA, MAGAZYNIER, REFERENT

pozwala na tworzenie nowego typu obiektu rodzajowego

PRACOWNIK

W odniesieniu do danej dziedziny przedmiotowej tworzone są hierarchie typów obiektów. Elementami tych hierarchii są węzły /typy obiektów/ i łuki. Łuk w przypadku agregacji jest odczytywany jako "jest częścią" a w stosunku do uogólnienia - "jest", w tym drugim przypadku mamy do czynienia z tworzeniem tzw. hierarchii ISA szeroko stosowanych w modelowaniu konceptualnym. Typ obiektu rodzajowego O jest uogólnieniem typów obiektów składnikowych O_1, O_2... O_n jeżeli każde wystąpienie typu obiektu O jest jednocześnie wystąpieniem przynajmniej jednego z typów obiektów O_1, O_2...O_n. Zależności te wyraża się graficznie następująco:

Dla opisu agregatów i obiektów rodzajowych można zastosować prosty język deklaracji obiektów, zaproponowany przez Smith'ów/[SMI-77a] , [SMI- 80]/. Agregat ZAMÓWIENIE zostaje zadeklarowany jako

type zamówienie - agregate O, T, D

O: odbiorca

T: towar

D: dostawca

end

Z bardziej złożonym opisem mamy do czynienia w przypadku obiektów rodzajowych. Umożliwia on ich definicję /def/ przez specyfikację obiektów składnikowych /subs/ i atrybutów /coms/, np:

def pracownik: sub

sprzedawca, magazynier, sekretarka

com

nazwisko, imię, płaca, staż

end

Wszystkie atrybuty obiektu rodzajowego /zwane dalej atrybutami rodzajowymi/ odnoszą się jednocześnie do obiektów składnikowych. Obiekty składnikowe mogą być opisane dodatkowo przez atrybuty składnikowe specyficzne dla danego obiektu.

Tak więc wiązka atrybutów typu obiektu składnikowego obejmuje zarówno atrybuty rodzajowe jak i składnikowe. Dla podanego typu obiektu rodzajowego definicja obiektów składnikowych przedstawia się następująco:

def sprzedawca: com

nr sklepu

wydajność

```
def magazynier: com
                    nr magazynu
                end
```

3. Narzędzia definiowania i użytkowania schematów konceptualnych

3.1. Języki opisu schematu konceptualnego

Rozwój teorii modelowania konceptualnego inspirował konstrukcję języków opisu schematu konceptualnego. Istnieje cały szereg niezależnych, oryginalnych rozwiązań [WRY-83] , mimo koncepcji standardów tych języków, zaprezentowanych w [GRI-82]. Wydaje się celowym przedstawienie wybranych cech niektórych z tych języków. Porównanie to przedstawiono w tablicy 1. Przyporządkowano w niej charakteryzowane języki opisu odpowiednim podejściem do modelowania konceptualnego. Opracowane najwcześniej języki oparte są na podejściu Binarnym a w ostatnich latach - na Abstrakcjach Danych. Rachunek Predykatów jest narzędziem pomocniczym /CSDL, SCHEMAL/ lub autonomicznym nie wymagającym dodatkowych definicji. W tablicy zaznaczono jakie kategorie schematu konceptualnego są opisywane w danym języku. Odpowiadają one podstawowym pojęciom związanych z nimi podejść. Definiowanymi Abstrakcjami Danych są agregaty i obiekty rodzajowe. Język TAXIS wprowadza do opisu schematu aspekt dynamiczny poprzez definicję pojęcia transakcji a język BETA - operacji i transakcji.

Tablica 1

Porównanie charakterystyk języków opisu schematu konceptualnego

Nazwa języka / Charakterystyka	CSL	ENALIM	SCHEMAL	TAXIS	BETA	CSDL	SDM	IRSDL
Publikacje	[BRE-79] [BRE-82]	[NIJ-77] [NIJ-79]	[FRO-83a] [FRO-83b]	[MYL-80]	[BRO-82]	[YEH-78]	[HAM-81]	[KON-80]
Autorzy	Breutman Mauer Falkenberg	Nijssen Nijmeijer, Snijders, van Asche	Frost Whittaker	Mylopoulos, Bernstein, Wong	Brodie, Silva	Roussopoulos, Araya, Chang, Yeh	Hammer, McLeod	Konsynski Manino
Podejście	B	B	B+RP	A D	A D	A D+RP	A D	OAR
Obiekt	X	X	X	X	X	X	X	X
Atrybut				X	X		X	X
Relacja	X	X	X				X	X
Abstrakcje Danych				X		X	X	
Operacja					X			
Transakcja				X	X			
Więzy integralności	X	X	X			X		X
Ograniczenia bezpiecz.								X
Reguły wnioskowania			X					

Źródło: Opracowanie własne.

Więzy integralności są uwzględniane w ograniczonym zakresie i dotyczą w zasadzie określenia typu i zakresu wartości atrybutów. Symboliczny jest opis ograniczeń bezpieczeństwa w języku SECOL, obejmując specyfikację poziomów upoważnień obiektów, relacji i atrybutów. Autorzy języka SCHEMAL wyodrębnili w opisie schematu konceptualnego część zwaną regułami wnioskowania /ang. inference rules/. Specyfikuje ona zapisane z wykorzystaniem Rachunku Predykatów, ograniczenia dotyczące relacji. Zbiór tych ograniczeń może być pusty. Składnię i semantykę oraz przykładowe zastosowania poszczególnych języków opisu schematu konceptualnego zawierają cytowane w tablicy publikacje, jak również częściowo opracowanie [WRY-83].

3.2. Komputerowo-wspomagane konstruowanie schematu konceptualnego

Kwestia modelowania konceptualnego jest zagadnieniem na tyle sformalizowanym, że pojawiły się pierwsze próby opracowania komputerowo-wspomaganych narzędzi konstruowania schematu konceptualnego. Podstawowe funkcje spełniane przez te systemy to:

- konstruowanie schematu konceptualnego bazy danych na zasadzie interaktywnej współpracy z projektantami czy administratorami /dziedziny przedmiotowej zastosowań, bazy danych/,
- sporządzanie raportów składniowej analizy poprawności opisu - sygnalizacja błędów formalnych w definiowaniu poszczególnych składników opisu, korekta niepoprawnych opisów,

- diagnostyka wprowadzanych definicji składników schematu konceptualnego i typowanie redundancji, kandydatów na synonimy i homonimy,
- aktualizacja schematu poprzez operacje tworzenia, skreślania i modyfikacji,
- wyszukiwanie składników schematu w interaktywnym trybie użytkowania systemu przez administratorów i użytkowników,
- integracja podschematów w schemat konceptualny,
- edycja zdefiniowanego w odpowiednim języku opisu schematu konceptualnego lub jego części oraz grafiki schematu,
- generowanie zestawień analitycznych i statystycznych /wytwarzanie dokumentacji schematu/,
- ocena jakości prototypów schematów w kolejnych testach wg przyjętych kryteriów.

Aktualnie najbardziej uznanymi systemami komputerowego wspomagania w zakresie modelowania konceptualnego są ERA/ERE, INCOD-DTE oraz CSDA. Tablica 2 zawiera porównanie wybranych parametrów obydwu systemów.

Wymienione systemy oparte są na modelu OAR, podzielone na moduły i posiadają wyodrębnione kryteria oceny modeli konceptualnych oraz języki opisu schematu konceptualnego. System INCOD-DTE umożliwia opis dynamiki schematu konceptualnego dzięki językom opisu zdarzeń i transakcji, natomiast CSDA - opis transakcji. Systemy wspomaganego komputerem modelowania konceptualnego częściowo automatyzują proces tworzenia schematów konceptualnych, podnoszą poziom ich jakości, zwłaszcza w odniesieniu do złożonych i obszernych dziedzin przedmiotowych.

Tablica 2

Porównanie systemów komputerowego wspomagania modelowania konceptualnego

Lp.	Nazwa	Autor/-zy/ Bibliografia	Model konceptualny	Moduły	Kryteria oceny modeli	Języki opisu schematu konceptualnego
1	ERA/ ERE	Lieberman [LIE-80]	Model OAR	Gromadzenie, Wyszukiwanie	kompletność, spójność	RULES-j. specyfikacji obiektów oraz oceny jakości modeli
2	INCOD -DTE	Atzeni i inni [ATZ-82] [ATZ-83a]	Model OAR /+Abstrakcje Danych/ Model zdarzeń, Model transakcji	Dokumentacja, Projektowanie, Sterowanie, Integracja	spójność	języki realizacji modułów, języki specyfikacji zdarzeń i transakcji
3	CSDA	Sakai i inni [SAK-83]	Model OAR /+Abstrakcje Danych/ Opis transakcji	Tworzenie, Aktualizacja, Wyszukiwanie	kompletność, spójność	CSDL - język opisu schematu konceptualnego

Źródło: Opracowanie własne.

3.3. Języki zapytań

Rozwój różnorodnych narzędzi modelowania konceptualnego obejmuje również języki zapytań. Opracowane projekty składni i semantyki tych języków bazują głównie na modelu OAR. Dotyczy to m.in. języków CABLE [SHO-78], CLER [POO-80] czy GORDAS [ELM-83]. Atzeni i Chen [ATZ-83b] zaproponowali kompletny język zapytań bazujący na modelu OAR. Podstawową operacją tego języka jest wyszukiwanie obiektów lub relacji odpowiednio z typem obiektu lub typu relacji. Wyszukiwanie obiektów może być określone następująco:

a/ wskazanie warunku spełnionego przez atrybuty typu obiektu, np.: FIND pracownik WITH /staż > 5 lat/,

b/ wskazanie relacji danego typu relacji, w której obiekty biorą udział, wybrane zgodnie z warunkiem spełnianym przez typ obiektu, np.:

```
FIND dostawca
     THROUGH HAVING partia towaru
     WITH /ilość > 5 000/,
```

c/ wskazanie relacji danego typu relacji, w której obiekty biorą udział, obejmującego obiekty innego typu obiektu spełniającego warunek WITH, np:

```
FIND  zapas
      THROUGH zapas towaru
      HAVING  towar
      WITH  /symbol = 'x'/
```

Wyszukiwanie relacji może być dokonane w podobny sposób:

- wyszukiwanie relacji poprzez

a/ wskazanie warunku spełnionego przez atrybuty

FIND transport-towaru WITH /data-rozchodu = '5 marzec'/

b/ wyszukiwanie relacji poprzez:
wskazanie obiektów danego typu obiektu
wybranych odpowiednio do warunku. WITH :

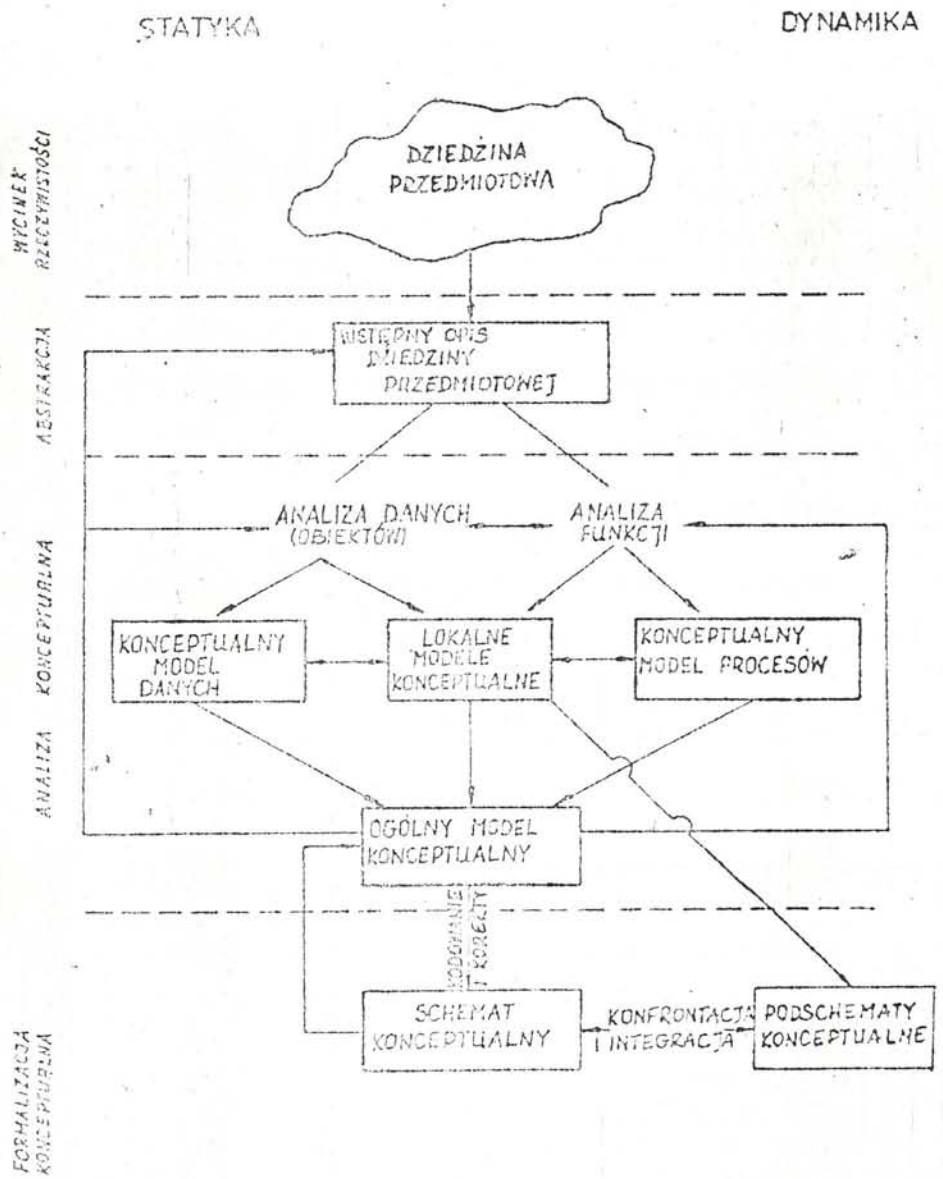
FIND dostawa-towaru
HAVING magazyn WITH /nazwa = 'rozdzielczy'/

Powyższe typy zapytań. są integrowane w konstrukcje bardziej złożone.

4. Proces modelowania konceptualnego

Dotychczasowe rozważania miały charakter porządkujący, zwłaszcza w zakresie terminologicznym oraz opisowy w dziedzinie porównań stosowanych metod i technik modelowania konceptualnego. Wyłania się jednak pytanie - jak w sposób usystematyzowany modelować konceptualnie bazę danych? Jaki zastosować zestaw metod i technik i jak je logicznie uporządkować aby skonstruowany model konceptualny najdokładniej odzwierciedlał badaną dziedzinę przedmiotową? W jaki sposób zapewnić w dostateczny sposób wpływ aspektów dynamicznych na finalną postać modelu a następnie schematu konceptualnego, który zwłaszcza w ujęciu binarnym oraz Obiekt - Atrybut - Relacja zawiera przewagę elementów statycznych ? Propozycję procesu modelowania konceptualnego przedstawiono na schemacie 4.

Schemat 4



Procesy modelowania konceptualnego

Źródło: Opracowanie własne

Przebiega on w trzech etapach : abstrakcji, analizy konceptualnej i formalizacji konceptualnej. Wynikiem realizacji każdej z wymienionych faz są odpowiednio: wstępny opis dziedziny przedmiotowej, model konceptualny i schemat konceptualny. Podstawą metodyczną i narzędziową, służącą przeprowadzeniu każdej z tych faz są: dla abstrakcji - metody opisu i specyfikacja potrzeb, dla analizy - teoria modelowania konceptualnego, a dla formalizacji konceptualnej - języki opisu schematu konceptualnego. Wstępny opis dziedziny przedmiotowej pełni rolę wstępnego modelu konceptualnego. Cały proces modelowania oraz jego poszczególne fazy podlegają ocenie i weryfikacji zgodnie z przyjętymi kryteriami oceny. 1/ W wyniku oceny wstępny opis dziedziny przedmiotowej, model konceptualny bądź schemat konceptualny w kolejnych iteracjach zostają skorygowane celem osiągnięcia ich spójności i kompletności.

1/ Zdaniem Deen'a istnieje bezpośredni związek między stabilnością modelu konceptualnego a zakresem składników odpowiedniej metodyki [DEE-81a] . Poniżej strzałką wskazano kierunek spadku stabilności modelu wraz z uwzględnieniem w metodyce kolejnych elementów opisu:

opis obiektów,
↓ opis relacji,
↓ więzy integralności,
↓ ograniczenia bezpieczeństwa.

Podobny pogląd, określony tzw. "zasadą 80-20", wyrażają Hammer i McLeod. Autorzy [HAM-80, s.193] rozważając kwestię zależności pomiędzy złożonością metodyki modelowania a naturalnością, precyzją i adekwatnością konstruowanej bazy danych formułują zasadę 80-20." Jeśli schemat bazy danych zawiera dużą ilość składników będzie trudny do opanowania i zastosowania choć będzie dokładnym i precyzyjnym modelem badanej rzeczywistości. Z drugiej strony model z minimalnym zbiorem charakterystyk jest łatwiejszy w stosowaniu choć schemat jest mniej dokładnym opisem rzeczywistości. Rozwiązaniem jest tu "zasada 80-20" a mianowicie 80% przypadków modelowania może być zrealizowanych przy użyciu 20% ogólnej liczby właściwości wymaganych przez w pełni precyzyjny mechanizm modelowania.

Przedmiotem modelowania jest zarówno statyka jak i dynamika dziedziny przedmiotowej. Obserwacji dziedziny dokonuje się na podstawie sprecyzowanych przez użytkownika potrzeb informacyjnych. Znajdują one swój wyraz we wstępnym opisie dziedziny przedmiotowej. Opis ten stanowi punkt wyjścia dalszych czynności projektowych podzielonych na dwa równoległe, wzajemnie uzupełniające się strumienie - analizę danych /zwaną również analizą obiektów/ oraz analizę funkcji. W wyniku wykonanych czynności otrzymuje się odpowiednio konceptualny model danych i konceptualny model procesów. Dla poszczególnych procesów, zastosowań czy potrzeb opracowywane są lokalne modele konceptualne zwane również funkcjonalnymi modelami danych.
Integracja tych trzech elementów prowadzi do skonstruowania ogólnego modelu konceptualnego stanowiącego zakończenie fazy analizy konceptualnej. Jego ostateczna postać jest rezultatem kolejnych iteracji realizowanych w ramach stadium analizy konceptualnej. Integracja ta jest procesem twórczym oznaczającym likwidację występujących sprzeczności dla osiągnięcia spójnego, kompletnego modelu konceptualnego. Ogólny model konceptualny jest wejściem do fazy formalizacji konceptualnej czyli definiowania schematu konceptualnego za pomocą instrukcji języka opisu schematu konceptualnego. Poprzez ich kompilację zostaje stworzony schemat konceptualny,podlegający również iteracjom. Może być on transformowany w wyniku przetworzenia przez procesor zewnętrzny lub wewnętrzny na schemat odpowiednio zewnętrzny lub wewnętrzny.

Formalizacja konceptualna może dotyczyć funkcjonalnych modeli konceptualnych. W rezultacie powstają podschematy konceptualne. Są one konfrontowane w stosunku do schematu konceptualnego, dziedziny przedmiotowej /zdefiniowanych typów obiektów i relacji, atrybutów oraz reguł integralności/. Może również mieć miejsce integracja podschematów konceptualnych poszczególnych funkcji we wspólny schemat. 1/

1/ Najbardziej zaawansowane rozwiązania w tym względzie zaprezentowali Batini i inni [BAT-83], Większość propozycji umożliwia integrację jedynie dwu podschematów konceptualnych jednocześnie. Podejście [BAT-83] pozwala na integrację większej ich ilości w iteracyjnym procesie łączenia kolejnych podschematów /n>2/ z dotychczasowym "kumulacyjnym" schematem konceptualnym. Czynności te wprawdzie wspomagane komputerowo w głównej mierze wykonywane są przez projektantów i polegają na analizie konfliktów, tj. wykrywaniu wszystkich sprzeczności, niespójności, jakie istnieją między pojęciami i nazwami integrowanego podschematu a dotychczasowym "kumulacyjnym" schematem konceptualnym. W wyniku dokonanych poprawek /raport poprawek/ opracowuje się skorygowany podschemat, schemat oraz wzajemne zależności. Usunięcie występujących redundacji staje się podstawą ich integracji w nowy "kumulacyjny" schemat konceptualny. Podana sekwencja powtarzana jest aż do włączenia wszystkich schematów do ogólnego spójnego schematu konceptualnego.

BIBLIOGRAFIA

Bazy danych - teoria

[DAT - 81] Date C.J., Wprowadzenie do baz danych, WNT, Warszawa 1981

Opracowania standaryzacyjne

[DE - 81] De P., Haseman W.D., So R.H., Four Schema Approach: An Extended Model for Data Base Architecture, Information Systems, 1981, vol. 6, nr 2, ss. 117-124

[GRI - 82] Griethuysen J.J. i inni /eds/, Concepts and Terminology for the Conceptual Schema and Information Base, ISO TC97/SC5/WG3, February 1982

[JAR - 84] Jordine D.A., Concepts and Terminology for the Conceptual Schema and the Information Base, Computers and Standards, 1984, nr 3, ss. 3-17

[TSI - 78] Tsichritzis D., Klug A., ANSI/X3/SPARC DBMS Framework Report of the Study Group on Database Management Systems, Information Systems, 1978, vol 3, ss.173-191

Teoria organizacji

[PSZ - 78] Pszczołowski T., Mała encyklopedia prakseologii i teorii organizacji, Ossolineum, Wrocław 1978

Taksonomie modeli konceptualnych

[BIL - 77] Biller H., Neuhold E.J., Concepts for Conceptual Schema, w: [NIJ-77, ss. 1-30]

[KER - 77] Kerschberg L., Klug A., Tsichritzis D., A Taxonomy of Data Models, w: [LOC-77, ss. 43-64]

[KUN - 83] Kung C.H., An Analysis of Three Conceptual Models with Time Perspective, w: [OLL-83, ss. 141-167]

[OLI - 83] Olivé A., DADES: A. Methodology for Specification and Design of Information Systems, w: [SCHN-82, ss. 285-334]

Podejście Obiekt - Atrybut - Relacja

[BAC - 69] Bachman C.W., Data Structure Diagrams, w: [FIL-69, ss. 108-116]

[CHE - 77] Chen P.P.S., The Entity Relationship Model: Toward a Unified View of Data, Massachusetts Institute of Technology, Sloan School of Management, Center for Information Systems, Research CISR 30, WP 913-77, Cambridge, March 1977 .

Podejście Binarne

[BRA - 76] Bracchi G., Paolini, P., Pelagatti G., Binary Logical Associations in Data Modelling, w: [NIJ-76, ss. 125-148]

[FAL - 76] Falkenberg E.D., Concepts for Modelling Information, w: [NIJ-76, ss. 95-109]

[NIJ - 76] Nijssen G.M., A Gross Architecture for the Next Generation Database Management Systems, w: [NIJ-76, ss 1-24]

Rachunek Predykatów

[MAŁ - 70] Mała Encyklopedia Logiki, Ossolineum, Wrocław 1970

[PAS - 80] Pasenkiewicz K., Logika ogólna, PWN, Warszawa 1980

[STE - 80] Steel T.B. jr., Report on TSO Activity in Data Base Standardization, w: [DEE-81, ss. 459-467]

Abstrakcje Danych

[BRO - 80] Brodie M.L., Zilles S.N. /eds/, Proceedings of Workshop on Data Abstraction, Data Bases and Conceptual Modelling, Pingree Park, Colorado, 1980, June 23-26

[DE - 82] De P., Sen A., Gudes E., A New Model for Data Base Abstraction Systems, 1982, vol. 7, nr 1, ss.1-12

[SAN - 80] dos Santos C.S., Neuhold E.J., Furtado A.L., A Data Type Approach to the Entity - Relationship Model, w: [CHE-80, ss. 105-119]

[SMI - 77a] Smith J.M., Smith D.C.P., Database Abstractions: Aggregation, Communications of the ACM, June 1977, vol. 20, nr 6, ss. 405-412

[SMI - 77b] Smith, J.M., Smith D.C.P. Database Abstraction: Aggregation and Generalization, ACM Transactions on Database Systems, June 1977, vol. 2, nr 2, ss. 105-133

[SMI - 80] Smith D.C.P., Smith J.M., Conceptual Database Design, w: [DAT-80, ss. 137-162]

Więzy integralności

[LEN - 83] Lenzerini M., Santucci G., Cardinality Constraints in the Entity - Relationship Model, w: [DAV-83, ss. 529-550]

[NIJ - 80] Nijssen G.M., Towards an Ideal Conceptual Model for Data Base Management, w: DEE-80, ss.179-180

Języki opisu schematu konceptualnego

[BRE - 79] Breutman B., Falkenberg E., Mauer E., CSL: A Language for Defining a Conceptual Schema, w: [BRA-79, ss. 237-256]

[BRE - 82] Breutman B., Mauer R., Conceptual Schema Language, Conceptual Manipulation Language for Defining, Retrieving and Updating Conceptual Data, Siemens AG, München 1982

[BRO - 82] Brodie M.L., Silva E., Active and Passive Component Modelling ACM/PCM, w: [OLL-82, ss. 41-91]

[FRO-83a] Frost R.A., Whittaker S., A. Step Towards the Automatic Maintenance of the Semantic Integrity of Databases, The Computer Journal, 1983, vol. 26 nr 2, ss. 124-133

[FRO - 83b] Frost R.A., SCHEMAL: Yet Another Conceptual Schema Definition Language, The Computer Journal, 1983, vol. 26, nr 3, ss. 228-234

[HAM - 81] Hammer M., Mc Leod D., Database Description with SDM: A Semantic Database Model, ACM Transactions on Database Systems, September 1981, vol. 6, nr 3, ss. 351-386

[KON - 80] Konsynski B.R. Manino M., Information Resource Specification and Design Language, w: [CHE-80, ss. 337-351]

[MYL - 80] Mylopoulos J., Bernstein P.A., Wong M.K.T., A Language Facility for Designing Database - Intensive Applications, ACM Transactions on Database Systems, June 1980, vol. 5, nr 2 ss. 185-207

[NIJ - 77a] Nijssen G.M., Current Issues in Conceptual Schema Concepts, w: [NIJ-77, ss. 31-65]

[NIJ - 79] Nijssen G.M., van Asche F.J., Snijders J.J., End-User Tools for Information Systems Requirement Definition, w: SCHN-79, ss. 125-148

[WRY - 83] Wrycza S., Języki opisu schematu konceptualnego, w: [WOJ-83, ss. 53-61]

[YEH - 78] Yeh R.T., Roussopoulos N., Chang P., Data Base Design Approach and Some Issues, w: [DAT-78, ss. 443-475]

Wspomagane komputerem modelowanie konceptualne

[ATZ - 82] Atzeni P., Batini P., de Antonellis V., Lenzerini M., Villanelli F., Zonta B., A. Comuter- Aided Tool for Conceptual Data Base Design, w:[SCHN-82, ss. 85-106]

[ATZ - 83a] Atzeni P., Batini C., Carboni E., de Antonellis V., Lenzerini M., Villanelli F., Zonta B., INCOD-DTE: A System for Interactive Conceptual Design of Data, Transactions and Events, w:[CER-83, ss. 205-228]

[LIE - 80] Lieberman A.Z., Machanized Analysis of Entity-Relationship Design Databases, w : [CHE-80, ss.641-663]

[SAK - 83] Sakai H., Kondo H., Kawasaki Z., A Development of a Conceptual Schema Design Aid in the Entity - Relationship Model, w: [CHE-83, ss. 411-428]

Języki zapytań

[ATZ - 83b] Atzeni P., Chen P.P.S., Completeness of Query Languages for the Entity - Relationship Model, w:[CHE-83, ss. 109 - 122]

[ELM - 83] Elmasri R., Wiederhdd G., Gordas: A. Formal High - Level Query Language for the Entity - Relation - ship Model, w: [CHE-83, ss. 49-72]

[POO - 80] Poonen G., CLEAR: A Conceptual Language for Entitues and Relationships w: [CHE-80]

[SHO - 78] Shoshani A., CABLE: A Language Based on the Entity-Relationship Model, Lawrence Berkeley Laboratory, University of California, January 1978

Proces modelowania konceptualnego

[BAT - 83] Batini C., Lenzerini M., Moscarini M., Views Integration w: [CER-83, ss. 57-84]

[DEE - 81a] Deen S.M., The State of the Art Database Research w: [DEE-81, ss. 1-41]

[HAM - 80] Hammer M., Mc Leod D., On Database Management System Architecture, w: DAT-80, ss. 179-201

[TAR - 80] Tardieu H., Pascot D., Nanci D.,Heckenroth H., A Method, a Formalism and Tools of Data Base Design /Three Years of Experimental Practice/, w: [CHE-80, ss. 384-408]

Opracowania zbiorcze

[BRA - 79] Bracchi G., Nijssen G.M. /eds/, Data Base Architecture Noth - Holland, Amsterdam 1979. Proceedings of the IFIP Working Conference, Venice 26-29 June 1979

[CER - 83] Ceri S., /ed/, Methodology and Tools for Data Base Design, North Holland, Amsterdam 1983

[CHE - 80] Chen P.P.S., /ed./, Entity - Relationship Approach to Systems Analysis and Design, North - Holland, Amsterdam, 1980. Proceedings of the First International Conference on Entity - Relationship Approach, Los Angeles, 1979, December 10-12

[CHE - 83] Chen P.P.S. /ed./, Entity - Relationship Approach to Information Modeling and Analysis, North Holland, Amsterdam 1983. Proceedings of the Second Int. Conf. on Entity-Relationship Approach, Washington D.C., 12-14 Oct. 1981

[DAV - 83] Davis C.G., Jajodia S., Ng P.A. - B., Yeh, R.T. /eds/ Entity - Relationship Approach to Software Engineering, North Holland, Amsterdam 1983. Proceedings of the Third Int. Conf. on Entity - Relationship Approach, Anaheim, 5-7 October 1983

[DAT - 78] Database Technology Infotech State of the Art Report, Maidenhead 1978, vol 1: Analysis and Bibligraphy: vol. 2: Invited Papers

[DAT - 80] Data Design, Infotech State of the Report, Maidenhead 1980, vol. 1: Analysis; vol. 2: Invited Papers

[DEE - 80] Deen S.M., Hammersley P., /eds/, Proceedings International Conference on Data Bases, University of Aberdeen, July 1980

[DEE - 81] Deen S.M., Hammersley P. /eds/, Databases, Pentech Press, London 1981. Proceedings of the First British National Conference on Databases held at Jesus College, Cambridge, 13-14 July 1981

[FIL - 69] File Organization, Selected Papers from File 68 IAG Conference, Swetz and Zeitlinger, Amsterdam 1969

[LOC - 77] Lockeman P.C., Neuhold E.J. /eds/, Systems for Large Data Bases, North - Holland, Amsterdam 1977. Proceeding of the Second International Conference on Very Large Data Bases, Brussells, 8-10 September 1976

[NIJ - 76] Nijssen G.M., /ed./, Modelling in Data Base Management Systems, North-Holland, Amsterdam 1976. Proceedings of the IFIP Working Conference on Modelling in DBMS, Freudanstadt, 5-8 January 1976

[NIJ - 77] Nijssen G.M., /ed./, Architecture and Models in Data Base Management Systems, North- Holland, Amsterdam 1977. Proceedings of the IFIP Working Conference in Data Base Management Systems, Nice, 3-7 January 1977

[OLL - 82] Olle T.W., Sol G.M., Verrijn - Stuart A.A., Information Systems Design Methodologies: A Comparative Review, North-Holland, Amsterdam 1982, Proceedings of the IFIP WG8.1 Working Conference on Comparative Review of Information Systems Design Methodologies, Noordwijkerkont, 10-14 May 1982

[OLL - 83] Olle T.W., Sol H.G., Tully C.J., Information Systems Design Methodologies: A Feature Analysis, North - Holland, Amsterdam 1983. Proceedings of the IFIP WG8.1 Working Conference on Feature Analysis of Information Systems Design Methodologies, New York, 5-7 May 1983

[SCHN - 79] Schneider H.J. /ed./, Formal Models and Practical Tools for Information Systems Desing, North - Holland, Amsterdam 1979. Proceedings of the IFIP TC-8 Working Conference, Oxford, 17-20 April 1979

[SCHN - 82] Schneider H.J., Wasserman A.I. /eds/, Automated Tools for Information Systems Design, North - Holland, Amsterdam 1982. Proceedings of the IFIP WG8.1 Working Conference, New Orlean, 26-28 January 1982.

KOMPUTEROWE WSPOMAGANIE PROJEKTOWANIA SYSTEMÓW BAZ DANYCH

Kazimierz Sycz, Stanisław Zimnocho
Instytut Informatyki
Akademia Ekonomiczna Wrocław
ul. Komandorska 118/120

Praktyczne korzyści, jakie niesie z sobą zastosowanie technologii baz danych w konstruowaniu systemów informatycznych są dosyć oczywiste: systemy powstają szybciej, łatwiej jest je uzupełniać o nowe funkcje użytkowe; można znaleźć przykłady dużych systemów, które bez technologii baz danych w ogóle by nie powstały. Z drugiej strony praca projektantów i programistów takich systemów nie stała się przez to prostsza, często słyszy się zdanie wręcz przeciwne. Informatyka, z takim zapałem usprawniająca pracę różnego rodzaju "użytkowników", do niedawna jeszcze była w sytuacji przysłowiowego szewca bez butów. Ostatnie lata przyniosły, na szczęście, znaczny wzrost zainteresowania szeroko rozumianym wspomaganiem prac projektowo-programistycznych, włączając w to prace związane z eksploatacją, modyfikacją i rozwojem istniejących systemów. Tworzonych jest coraz więcej konkretnych produktów programowych - narzędzi wspomagających określone zadania, jak i zintegrowanych OTOCZEŃ/ŚRODOWISK KONSTRUKCYJNYCH.

Przedmiotem naszego wykładu będą takie właśnie n a r z ę d z i a i ś r o d o w i s k a d l a k o n s t r u k c j i s y s t e m ó w b a z d a n y c h. Interesować nas będą w szczególności produkty programowe specyficzne dla tej klasy systemów (wyczerpujący opis OTOCZEŃ PROGRAMISTY, związanych najczęściej z konkretnym językiem programowania i od baz danych niezależnych, znaleźć można np. w [1] i [2]). Ograniczymy się do najbardziej typowych zastosowań - systemów baz danych organizacji administracyjnych i gospodarczych, nie będziemy się też zajmować szczególnymi cechami systemów rozproszonych. W naszym przeglądzie pojawią się zarówno pakiety używane w praktyce, jak i te z projektów badawczych, które zakończyły się implementacją.

Cykl życia systemu

Głównie po to, by ł a t w i e j k i e r o w a ć tworzeniem i eksploatacją systemów baz danych dla zastosowań gospodarczych i administracyjnych, i w ten sposób unikać zdarzających się tu i ówdzie niepowodzeń, wprowadzono pojęcie cyklu życia systemu. Dzieli ono okres od momentu powstania pierwszej koncepcji systemu do zaprzestania jego eksploatacji, na następujące fazy:

1. analiza systemu przedmiotowego i zebranie zapotrzebowań,
2. modelowanie konceptualne,
3. projektowanie bazy danych i działających nań transakcji,
4. implementacja,
5. eksploatacja,
6. modyfikacje i rozwój systemu b.d.

W fazie pierwszej projektanci, sponsorzy i przyszli użytkownicy systemu wspólnie określają co system ma robić. W wyniku pewnego przetargu - zestawienia cech pożądanych z konsekwencjami wprowadzenia ich do systemu - powstaje k o n t r a k t, czy też sformułowanie celu dla zadania projektowego. Ponadto zebrane zostają i u d o k u m e n t o w a n e wszelkie informacje potencjalnie przydatne w następnych fazach. W szczególności, w dobrze zarządzanym przedsięwzięciu projektowym, sformułowane tu zostają pewne ogólne r e g u ł y będące później podstawą decyzji w sytuacjach konfliktowych (wyboru jednego spośród wielu możliwych rozwiązań).

Chcemy zauważyć, że w interesującej nas klasie systemów już w pierwszej fazie, na etapie zwanym zwykle analizą funkcjonalną, zapadają decyzje o charakterze wyraźnie projektowym, jak np. wyróżnienie (choćby zgrubne) funkcji aplikacyjnych, którym odpowiadają późniejsze programy i transakcje. Z tego m.in. względu w dalszej części wykładu rozpatrujemy także narzędzia wspomagające analizę i zbieranie zapotrzebowań.

Fazę m o d e l o w a n i a k o n c e p t u a l n e g o odróżnia od końcowych etapów poprzedniej przede wszystkim stopień formalizacji. W fazie pierwszej zapotrzebowania są zwykle nieformalne, m.in. dlatego, że muszą być czytelne także dla sponsora i użytkowników. W obecnej fazie przekształca się je w możliwie jak najbardziej formalne i precyzyjne specyfikacje projektowe. (Model konceptualny i specyfikacje projektowe uważamy tutaj za synonimy). Sformalizowanie zapotrzebowań do specyfikacji umożliwia późniejsze stwierdzenie czy rzeczywiście skonstruowano to, o co chodziło.

W fazie p r o j e k t o w a n i a s y s t e m u następuje przekształcenie części modelu konceptualnego dotyczącej danych w implementowany w konkretnym systemie zarządzania bazą danych (SZBD) s c h e m a t b.d. Przy tym spośród wielu możliwych schematów chcemy wybrać taki, który w największym stopniu spełnia sformułowane wcześniej - w kontrakcie - kryteria dobroci czy o p t y m a l n o ś c i. W tej fazie projektowane są również algorytmy bardziej złożonych programów i ich wewnętrzna struktura (podprogramy itp.). Zapadają tu także decyzje co do tego, które z o g r a n i c z e ń i n t e g r a l n o ś c i o w y c h naszych danych będą weryfikowane na bieżąco i w jaki sposób.

Dalsze fazy cyklu życia systemu nie wymagają komentarza.

Wspomaganie pełnego cyklu życia systemu

Każda z faz cyklu życia może być wspomagana odpowiednimi narzędziami programowymi. Pojęcie narzędzia programowego jest najłatwiej "uchwytne" w przypadku narzędzi wspomagających programowanie, takich jak: kompilator, interpreter, debugger, edytor - w szczególności edytor sterowany składnią, analizator statyczny (wykrywający np. użycie niezainicjowanych zmiennych itp.). W ogólnym przypadku na poszczególnych etapach cyklu życia systemu b.d. pożądane są narzędzia wspierające:

- opis problemu projektowego,
- dekompozycję problemu,
- dokumentowanie zapotrzebowań i specyfikacji,
- dokumentowanie podejmowanych decyzji projektowych i przesłanek tych decyzji,
- kontrolę wewnętrznej zgodności dokumentacji i zgodności z przyjętymi standardami,
- wiązanie specyfikacji, decyzji projektowych, programów z zapotrzebowaniami (traceability),
- automatyczną modyfikację dokumentacji w przypadku zmian,
- integrowanie modeli konceptualnych,
- generowanie rozwiązań projektowych,
- ocenę efektów przyjmowanych rozwiązań i obliczenia projektow
- kodowanie/generowanie/derywację programów,
- kontrolę poprawności programów (systematyczne testowanie, formalne dowodzenie poprawności),
- wnioskowanie o programach,
- konfigurowanie oprogramowania, kontrolę wersji programów,
- zarządzanie przedsięwzięciem projektowym/implementacyjnym,

- wzajemne komunikowanie się wszystkich osób związanych z danym przedsięwzięciem (analitycy, projektanci, programiści, użytkownicy),
- monitorowanie i ocenę działającego systemu,
- (pół)automatyczne generowanie dokumentacji użytkowej itp.

Niezmiernie korzystne jest zintegrowanie wspomnianych (i innych) narzędzi w ujednoliconym ŚRODOWISKU KONSTRUKCYJNYM SYSTEMU. Oto kilka oczywistych zalet takiej integracji:

- możliwość użycia - w płynnym ciągu - narzędzi obsługujących poszczególne etapy konstrukcji i w rezultacie znaczne skrócenie cyklu prac,
- zapewnienie zgodności dokumentacji i systemu w niej opisanego,
- stworzenie jednolitej, stale aktualnej i nieredundantnej bazy informacji o projekcie i gotowym systemie (także podstawy późniejszych jego modyfikacji),
- możliwość sterowania procesem projektowym zgodnie z jednolitą metodologią,
- możliwość nauczania, niejako "przy okazji", właściwych metod projektowania i implementacji.

Tego rodzaju środowisko łączy w sobie aspekty techniczne, komunikacyjne, organizacyjne i kierowania przedsięwzięciem; w pewnym sensie może być uważane za szczególnego typu system automatyzacji biura, ukierunkowany tak na bezpośredni wzrost produktywności jak i na polepszenie ogólnych warunków pracy. Jednorazowy wydatek na stworzenie lub kupno środowiska konstrukcyjnego (i dokumentacyjnego, przy okazji) powinien szybko zwrócić się wraz z liczbą zrealizowanych w nim systemów, a przy tym będzie on (ten wydatek) w sumie niższy od wydatków poniesionych na kupno czy stworzenie własnych niezintegrowanych - i przez to najczęściej niekompatybilnych - pojedynczych narzędzi wspomagających (nasuwają się tu luźne skojarzenia z alternatywą: uogólniony system zarządzania bazą danych czy niezależnie zarządzane zbiory).

Żadne ze znanych nam środowisk konstrukcyjnych nie obejmuje przedstawionej wyżej listy pożądanych narzędzi, ani nie spełnia postulowanych przez nas własności. Tę listę narzędzi oraz własności należy więc traktować jako przybliżenie stanu idealnego, ale osiągalnego przy użyciu dostępnej obecnie technologii informatycznej. Chcemy zaznaczyć, że i s t n i e j ą już, używane praktycznie w organizacjach bynajmniej nie o charakterze badawczym, środowiska [illegible]iające wiele z powyższych postulatów; np. [3].

Narzędzia wspomagające analizę i zbieranie zapotrzebowań

Ilość informacji, jaka musi być zebrana przy realizacji średnich i dużych systemów b.d. jest często przytłaczająca. Problemem staje się zarówno s t r u k t u r a l i z a c j a procesu jej gromadzenia jak i d o k u m e n t o w a n i e jej w postaci pozwalającej na panowanie nad ogromną liczbą detali. W tej wstępnej fazie gromadzone dane są często poprawiane i uzupełniane, a każda taka modyfikacja może zakłócić ich spójność (wzajemną zgodność). Tym w jakiej kolejności co zbierać zajmują się różnego rodzaju metodyki analizy zwące same siebie "strukturalnymi". Każda z nich proponuje własne formy gromadzenia i prezentacji informacji, które coraz częściej są komputeryzowane.

Zakres informatycznego wspomagania sprowadza się tu zwykle do zapobiegania błędom przez sformalizowanie składni języka (języków) opisu zapotrzebowań i kontrolę syntaktyczną zdań czy formuł wprowadzanych do wspólnej "bazy informacyjnej". Przez krzyżowe porównywanie takich formuł możliwe jest wykrywanie niektórych rodzajów niespójności. Z "bazą informacyjną" związane są rozmaite języki zapytań i generatory raportów. Jak widać, jest to po prostu rodzaj specjalizowanego systemu b.d., w którym przechowuje się m e t a d a n e o danych i procesach przetwarzania docelowego systemu b.d.

Jako jedno z pierwszych narzędzi tej klasy zaimplementowano pakiet PSL/PSA [4]. PSL (Problem Statement Language) jest językiem opisu zapotrzebowań, którego zdania analizuje PSA (Problem Statement Analyzer), wykrywając niespójności. Późniejsze wersje tego pakietu działają on-line.

Pakiet REVS (Requirements Engineering Validation System) związany z metodyką SREM [5] ma więcej, w stosunku do PSL/PSA, funkcji użytkowych: poza typowymi raportami może generować np. szkielety programów aplikacyjnych dla celów ich symulacji; ma też wyjście graficzne. Po paru modyfikacjach REVS włączono do środowiska SPS [3], jednego z najbardziej "zaawansowanych" środowisk konstrukcyjnych.

Wspólną niezbyt korzystną cechą PSL/PSA i REVS (i wielu podobnych systemów) jest fakt, że sprawdzenie zgromadzonych zapotrzebowań pod kątem ich kompletności i spójności dokonywane jest dopiero po wprowadzeniu w s z y s t k i c h danych do pakietu. Częściowo

wada ta (dokuczliwa szczególnie przy modyfikacjach) jest usunięta w zestawie narzędzi analizy strukturalnej, powstałym w firmie Tektronix[1] [6]. Zestaw ten, jak na Tektronix przystało, ma wejście i wyjście graficzne; pozwala na tworzenie na ekranie i pamiętanie d i a g r a m ó w p r z e p ł y w u d a n y c h. Zestaw stworzono w systemie UNIX. Podobnym narzędziem jest Structural Architect firmy ISDOS [8], dostępny na IBM/PC, który może być też używany jako graficzny inteligentny terminal dla PSL/PSA.

Do omawianej klasy narzędzi można wreszcie zaliczyć wszelkiego rodzaju Słowniki/Skorowidze Danych, których bliżej nie omawiamy.

Inne funkcje fazy analizy, jak np. komunikacja, realizowane są przy pomocy narzędzi tworzonych dla automatyzacji biura, w rodzaju elektronicznej poczty.

Grono użytkowników pakietów takich jak wyżej wymienione wyraźnie wzrasta. Każdy szanujący się dostawca SZBD oferuje słownik/skorowidz lub coś podobnego. Nie bez znaczenia jest też uatrakcyjnienie wejścia (grafika, menu, formularze ekranowe) i ich dostępność na mikro i workstations.

Narzędzia wspomagające analizę i zbieranie zapotrzebowań wymuszają i umożliwiają formalizację odpowiednich języków opisu, co w coraz większym stopniu upodabnia tę fazę do modelowania konceptualnego.

Narzędzia wspomagające modelowanie konceptualne

Zakres wspomagania czynności projektowych w tej fazie jest w zasadzie taki sam jak dla analizy; występujące różnice są konsekwencją stosowania bardziej formalnych języków opisu/modelowania.

Jedną z prostszych i zarazem najbardziej popularnych notacji dla konceptualnego modelowania danych są diagramy entity-relationskip, spopularyzowane w publikacjach Chena. Zwykle kreślone ręcznie, mogą być również tworzone i modyfikowane na ekranie monitora graficznego. Kontrola poprawności tworzonych struktur (oczywiście jest to tylko kontrola syntaktyczna, gdyż interpretacja używanych nazw obiektów i relacji należy w tym formalizmie do projektanta) prowadzona jest zwykle na bieżąco. Prosty system skonstruowany na Uniwersytecie w Toronto [9] działa właśnie w taki sposób. Na wej-

[1] Z drugiej strony, zestaw Tektronixa ma trochę skromniejsze środki opisu (uwzględnia mniej aspektów).

ściu tego systemu korzysta się z klawiatury - do wprowadzania nazw obiektów, relacji i atrybutów, oraz z tzw. tablet-u (urządzenie przypominające pióro świetlne, z tym że wskazywanie i kreślenie odbywa się nie bezpośrednio na ekranie lecz na płaskiej metalowej tabliczce, umieszczonej przed monitorem). Pracą projektanta steruje się przy pomocy menu i podpowiedzi (prompting). Narzucona kolejność tworzenia lub usuwania fragmentów diagramów ułatwia kontrolę spójności i zapobiega błędom. Przykładowo: najpierw muszą być stworzone odpowiednie obiekty, by można je było połączyć relacją; słaby (tzn. egzystencjonalnie zależny) obiekt nie może być dodany do modelu bez uprzedniego zdefiniowania relacji łączącej go z właściwym obiektem nadrzędnym; by usunąć obiekt z modelu należy najpierw usunąć wszystkie relacje w jakich bierze on udział, co jest o tyle bezpieczniejsze od usuwania automatycznego (tzw. trigger) że wszelkie konsekwencje zmian są jawne. Ciąg podpowiedzi systemu przy tworzeniu modelu może wyglądać następująco: wybierz operacje z menu (tworzenie), wskaż miejsce gdzie zostanie wykreślony blok obiektu, wprowadź (z klawiatury) jego nazwę, wymień atrybuty obiektu (min. jeden) (...) wskaż (na menu) typ relacji (1:1, 1:N, M:N itp.).

Na gotowym diagramie można definiować podmodele (views). Późniejsze modyfikacje modelu są automatycznie odzwierciedlane we wszystkich views.

Aby panować nad wielkością/złożonością tworzonego diagramu, system dostarcza w menu komend w rodzaju: powiększ, zmniejsz skalę, powróć do wcześniejszej skali, przeglądaj fragmenty diagramu. Z tego samego powodu nie są wyświetlane nazwy atrybutów ani zbiory wartości, jakie mogą one przyjmować - można je otrzymać na życzenie. Plotter pozwala na kreślenie trwałych kopii views i pełnego modelu danych.

W ramach realizowanego we Włoszech projektu DATAID [11] stworzono dwa niezależne od siebie bardzo interesujące narzędzia do modelowania konceptualnego: INCOD-DTE i Galileo/Dialogo.

INCOD-DTE [12] (a system for INteractive COnceptual Design of Data, Transactions and Events) pozwala na interakcyjne tworzenie pełnego modelu konceptualnego systemu, obejmującego również aspekty dynamiczne - transakcje i zdarzenia. Uwzględnienie transakcji i zdarzeń pozwala na opracowanie specyfikacji dla przyszłych programów użytkowych jak i na znacznie pełniejszą kontrolę kompletności i spójności modelu.

Model konceptualny tworzony jest przyrostowo za pomocą k o m e n d trzech języków modelowania odpowiednio: danych, transakcji i zdarzeń. Język danych korzysta z rozszerzonego (o generalizację i podzbiory) modelu E-R, język transakcji (rozumianych tu jako kwerendy i modyfikacje) jest językiem manipulacji na danych w postaci E-R, zaś zdarzenia opisywane są przy użyciu klasycznie interpretowanych sieci Petriego, w postaci warunek, akcja, rezultat. Przed wykonaniem każda komenda INCOD-a sprawdzana jest pod względem syntaktycznym i "semantycznym" tzn. zgodności z dotychczas zbudowanym modelem. Jeśli dana komenda może zaburzyć spójność modelu, nie jest wykonywana a INCOD proponuje jej zaniechanie lub rozpoczęcie dialogu, w którym niespójność może zostać usunięta.

W takim dialogu system przedstawia użytkownikowi s c e n a r i u s z e k o n f l i k t ó w s p ó j n o ś c i tzn. potencjalne przyczyny niespójności i związane z nimi spójne stany (sposoby rozwiązania konfliktów). Projektant może wybrać jeden z sugerowanych mu stanów spójnych, a INCOD dokonuje odpowiednich modyfikacji. Nie wszystkie tego rodzaju modyfikacje są dokonywane automatycznie, w szczególności nie poprawia się istniejących już transakcji (transakcje takie system oznacza jako "zawieszone" i proponuje ich edycję przy pomocy komend języka transakcji; komendy te muszą być jednak sformułowane przez projektanta).

Potencjalnych źródeł niespójności jest bardzo wiele i trudno wymagać aby wszystkie z nich były ujęte jako możliwości w scenariuszu. Jeśli projektantowi nie odpowiada żadna z sugestii systemu, może nie wybrać żadnego rozwiązania, tym samym rezygnując z komendy, jaka prowadziła do niespójności.

Transakcje mogą być modelowane na trzech poziomach abstrakcji: k o n c e p t u a l n y m (na którym podaje się tylko nazwy i typy używanych w transakcji pojęć - tzn. obiektów, relacji, podzbiorów i generalizacji), n a w i g a c y j n y m (uzupełnia konceptualne o logiczne ścieżki dostępu) i w y k o n a l n y m (pełna specyfikacja fragmentów transakcji związanych z dostępem do danych z bazy, której można użyć w SZBD opartym na modelu ER). Na końcu sesji projektowej procedura walidacyjna sprawdza, czy transakcje wyspecyfikowano w pełni - do poziomu wykonalnego.

Inną ważną cechą INCOD-DTE jest możliwość automatycznej (przy zachowanej spójności) integracji cząstkowych modeli konceptualnych (views).

INCOD-DTE nie zawiera (poza określaniem warunków zdarzeń)

środków specyfikowania ograniczeń integralnościowych. Wejście systemu jest tekstowe, choć przynajmniej do modelowania danych można by użyć grafiki. Procesem modelowania steruje projektant, tzn. nie narzuca się określonej "właściwej" kolejności tworzenia fragmentów modelu konceptualnego. By taka swoboda nie prowadziła do błędów INCOD-DTE oferuje dialogowe - sterowane chwilowo przez system - rozwiązywanie niespójności. Model konceptualny może być tworzony na różnych poziomach abstrakcji.

Mimo drobnych niedociągnięć INCOD-DTE pretenduje do miana najlepszego obecnie narzędzia modelowania konceptualnego.

System DIALOGO [13] oparty na języku Galileo jest jednym ze śmielszych rozwiązań w dziedzinie baz danych. Można uważać go zarówno jako otoczenie języka Galileo, przeznaczone do szybkiego tworzenia prototypów systemów b.d., jak i jako samodzielny system bazy danych, oparty na pojęciu tzw. persistent programming (w którym zakłada się, że program żyje tak długo jak żyją jego dane oraz że wszelkie mechanizmy organizacji/dostępu do danych wbudowane są w kompilator/interpreter danego języka).

Galileo jest językiem "wyrażeniowym" (jak np. LISP), będącym rozszerzeniem (modyfikacją) stworzonego w Uniwersytecie Edynburskim języka ML. Jest językiem z typami, statyczną kontrolą typów, możliwością definiowania własnych typów (klas) i dziedziczeniem atrybutów (przy uszczegóławianiu klas). Przy pomocy Galileo można stworzyć d z i a ł a j ą c y prototyp systemu, a jeśli nie żądamy zbyt wygórowanych osiągów - sam system bazy danych (w którym użytkownik nie widzi różnicy między danymi w PAO a danymi na dyskach). D e f i n i o w a n i e t y p ó w w Galileo odpowiada tworzeniu schematu (danych), pisanie w y r a ż e ń języka - tworzeniu kwerend i transakcji. Wyrażenia służą też do określania ograniczeń integralnościowych - tej ich części, której nie załatwia statyczna kontrola typów i reguły dziedziczenia atrybutów.

Przy pomocy tego samego języka można w Dialogo tworzyć system b.d., zadawać pytania odnośnie definicji schematu i operacji, ładować próbne dane itp. Galileo służy też do personalizacji Dialogo, czyli dostosowania go do stylu pracy konkretnego użytkownika. Dostępne w Dialogo edytory pozwalają - jak np. w INTERLISPie - na tworzenie, w zewnętrznych zbiorach, definicji i wyrażeń Galileo, które potem, na życzenie, mogą zostać wykonane z danymi próbnymi. Środki odpluskwiające dziedziczy Dialogo z języka, w którym został zaimplementowany (jedna z wersji Lispu, powstaje też implementacja w Pascalu).

Dla wielu twórców systemów b.d. Dialogo zbyt brutalnie zrywa z powszechnie przyjętymi w tej dziedzinie pojęciami i zasadami - jaki jest tu model danych, czym się różni projektowanie od implementacji, jaki ma wpływ na osiągi systemu? Wydaje się, że jego dalsze dzieje jako narzędzia wspomagającego modelowanie/projektowanie systemów b.d. zależeć będą od powstania innych narzędzi, np. pozwalających tłumaczyć definicje typów i operacji Galileo na bardziej konwencjonalne pojęcia schematu czy transakcji SZBD. Z drugiej strony, być może przyszłość należy do persistent programming?

Narzędzia wspomagające projektowanie baz danych

W poprzednich fazach procesu konstruowania systemu b.d. określono co system ma robić, jakie dane mają być w nich przechowywane, jakie transakcje działać będą na bazie danych itp. Informacje te sformułowane są (powinny być sformułowane) w terminach niezależnych od używanego SZBD i od konkretnych rozwiązań przyjętych w schemacie, który przecież dopiero ma zostać zaprojektowany. Przykładowo: transakcje nie powinny zakładać określonego pogrupowania danych elementarnych w rekordy ani rodzaju powiązań między rekordami[1]. Zadaniem tej fazy jest stworzyć "najlepszy" projekt.

Narzędzia wspomagające projektowanie baz danych (projektowaniem transakcji nie będziemy się zajmować, narzędzia tu stosowane są w większości tego samego rodzaju co w nie-bazo-danowych systemach) można podzielić z grubsza na następujące grupy:

- transformatory, wspomagające tłumaczenie konceptualnego modelu danych na struktury danych schematu SZBD,
- ewaluatory, które dla podanego na wejściu projektu podają wartości wybranych jego charakterystyk,
- optymalizatory, które z podanych na wejściu informacji (model konceptualny + dane ilościowe) generują optymalny, przy przyjętym z góry kryterium, projekt.

[1] Jak można zauważyć używamy tu terminologii systemów sieciowych - tylko Codasyl sformułował terminy do opisu struktur fizycznych i pamięci. Nie znaczy to, że omawiane przez nas narzędzia mają zastosowanie jedynie do systemów DBTG.

Transformatory są programami nie-numerycznymi, mającymi za zadanie wykrycie takich konstrukcji modelu konceptualnego, które nie mogą być bezpośrednio zaimplementowane w danym SZBD. Tego rodzaju konstrukcje określają punkty decyzyjne, w których mamy do wyboru zwykle kilka sposobów implementacji (im mniej takich punktów tym lepiej, tym lepszy - bo bardziej konceptualny - SZBD). Decyzje w tych punktach mogą być podejmowane automatycznie przez system wspomagający projektanta, bądź są przekazywane do kompetencji człowieka. Transformatory nie gwarantują "optymalnych" schematów, lecz tylko schematy d o p u s z c z a l n e, stąd często łączone są z pozostałymi dwoma rodzajami narzędzi.

Ewaluatory i optymalizatory wymagają na wejściu znacznie więcej informacji, w szczególności ilościowych - jak np. ile danych elementarnych (wystąpień) każdego typu ma być pamiętane w bazie, wielkość każdej danej w znakach lub bajtach, ile danych każdego typu używa każda transakcja, jak często i ewentualnie w jakich grupach (wiązkach) wykonywane będą transakcje itp. Oba narzędzia w tym czy innym stopniu korzystają z wartości cech konkretnego SZBD (na ile SZBD ujawnia, tzn. pozwala na nie wpływać). Niezmiernie istotne są w końcu sposób i kryteria oceny tworzonego projektu.

Jakość projektu możemy określić ex post - po jego zaimplementowaniu, przez zebranie odpowiednich parametrów pracującego systemu, lub a priori - przy użyciu metod analitycznych i/lub symulacyjnych. Projekty możemy przy tym oceniać względnie, przez porównanie konkurencyjnych rozwiązań lub, w wypadku optymalizatorów, wybierać bezwzględnie najlepszy projekt. Optymalizatory wymagają jawnego sformułowania kryterium oceny np. minimalizuj zajętość pamięci dyskowej, minimalizuj sumaryczny czas odpowiedzi systemu, minimalizuj ilość i wolumen przesłań między dyskami a PAO czy minimalizuj funkcję kosztu uwzględniającą wszystkie bądź niektóre z wymienionych właśnie parametrów.

Warto zaznaczyć, że zagadnienie wyboru optymalnego projektu jest dualne w stosunku do zagadnienia optymalizacji kwerend w SZBD. Nie oznacza to jednak, że nasze zadanie można sprowadzić do wygenerowania dopuszczalnego schematu a sprawę optymalizacji powierzyć modułowi optymalizacji kwerend (o ile takowy istnieje w danym SZBD). Sęk m.in. w tym, że zbiór strategii przetwarzania indywidualnie zoptymalizowanych kwerend nie musi dawać optymalnego przetwarzania wszystkich kwerend w danej jednostce czasu.

Historyczne pierwszeństwo w dziedzinie transformatorów należy chyba przyznać Gerritsenowi [14], o ile zgodzimy się na lekkie naciągnięcie definicji takiego narzędzia. Program Gerritsena o nazwie DESIGNER oczekiwał na wejściu nie modelu konceptualnego lecz zdań języka kwerend o nazwie HI-IQ oraz listy nazw danych elementarnych. W kwerendach wolno było odwoływać się tylko do danych z listy i ich dowolnych grup. DESIGNER "wnioskował" na podstawie składni kwerend o typie relacji między grupami danych używanymi w kwerendzie a następnie używał tej informacji do utworzenia na wyjściu struktur Codasylowskich (rekordów i kolekcji (setów)). Ponieważ nie żądano by rekordy struktury wyjściowej pokrywały się z grupami danych używanymi przez HI-IQ, (i nazywanymi też rekordami) można mówić, że DESIGNER umożliwiał integrację "views" (oddzielne "view" dla każdej kwerendy). Tworzona na wyjściu struktura rekordów była nieredundantna - żadna dana elementarna nie mogła się pojawić w więcej niż jednym rekordzie. W przypadku konfliktu DESIGNER tworzył wspólny rekord "nadrzędny" (owner dla obu rekordów, w których miałoby wystąpić redundantnie jakieś pole) i w nim umieszczać ową daną konfliktową. Wejściowe zdania HI-IQ były najczęściej nieadekwatne do działania na stworzonej przez program strukturze.

Najpopularniejszym transformatorem z włączonymi pewnymi mechanizmami optymalizującymi jest IBM-owski DBDA (Data Base Design Aid) [15] wspomagający tworzenie struktur danych dla IMS/VS. Na wejściu tego pakietu korzysta się ze specjalnych formularzy opisu zapotrzebowań (przy odrobinie dobrej woli można się zgodzić, że na formularzach tych koduje się model konceptualny w postaci relacji binarnych). Jeden formularz odpowiada jednemu programowi użytkowemu (jednej transakcji) i zawiera ogólne informacje o tym programie - jak np. liczba przebiegów w jednostce czasu, wsad czy on-line, oraz bardziej szczegółowe informacje o danych w nim wykorzystywanych. Dane opisywane są w sposób niezależny od mającego dopiero powstać schematu bazy (DBD w terminologii IMSu). Konkretnie: podaje się listę danych elementarnych, powiązanych w pary zwane a s o c j a c j a m i (typu 1, M lub C). Asocjacje określają związki identyfikacyjne między danymi, np. asocjacja typu 1 od NR-CZĘŚCI do CENY-JEDNOSTKOWEJ mówi nam że:

a) każda część ma unikalną (z punktu widzenia danej aplikacji) cenę i

b) cena interesuje nas tylko jako pewien "atrybut" części (znów z punktu widzenia danego programu aplikacyjnego).

Asocjacje można więc traktować jako zależności funkcjonalne między danymi albo, jak kto woli, jako logiczne ścieżki dostępu określone na danych elementarnych. Typ M asocjacji oznacza zależność 1:M, typ C to rodzaj zależności 1:1, w którym wystąpienie danej "wskazywanej" jest opcjonalne (jest to typ np. asocjacji MAŁŻONEK - jeśli jest, to tylko jeden). Cały zestaw asocjacji dla określonego programu tworzy więc hierarchiczny "view" z wyróżnionym punktem (punktami) wejścia. Dla każdej danej elementarnej (każdej asocjacji) podawane są też jej charakterystyki ilościowe i rodzaj dostępu - w celu czytania, pisania, modyfikacji.

Zbiór tak opisanych "view" jest integrowany do wspólnego modelu kanonicznego (także binarnego), przy czym zwykle w trakcie takiej integracji wykrywane zostają niespójności globalne. Część tego rodzaju niespójności, jak np. różnie określone typy asocjacji, między tymi samymi danymi elementarnymi, w różnych "view", jest korygowana automatycznie przez LSA, inne są przedstawiane w postaci raportów diagnostycznych.

Model kanoniczny jest następnie analizowany pod kątem zgodności z ograniczeniami IMS/VS - sprawdza się czy nie zawiera on pętli, czy ewentualne związki M:M można zaimplementować (czy istnieje wspólny element podrzędny dla dwóch danych w związku M:M), usuwa się automatycznie związki redundantne (przechodnie).

W końcowym etapie pracy pakietu, DBDA g r u p u j e dane elementarne w segmenty które następnie łączy w h i e r a r c h i e. Powstała struktura hierarchiczna jest częściowo optymalizowana przez: szeregowanie segmentów podrzędnych na tym samym poziomie hierarchii (każda z metod dostępu IMS-a zapewnia szybszy dostęp do segmentów "bardziej na lewo") oraz wybór - w przypadku struktur z relacjami logicznymi - segmentów nadrzędnych fizycznie (jeśli określony segment podrzędny ma dwa nadrzędne, jeden z nich desygnowany jest na nadrzędny fizycznie i przez to uprzywilejowany jeśli chodzi o czas dostępu, drugi staje się nadrzędnym logicznie). Wybierane są również segmenty (pola), dla których pożądane jest utworzenie indeksów pomocniczych.

Przy grupowaniu danych elementarnych w segmenty, DBDA podobnie jak wcześniej omawiany DESIGNER, kieruje się zasadą minimalnej redundancji, nie generuje jednak żadnych dodatkowych segmentów, informując o niezgodnościach przez raporty diagnostyczne.

Wynikiem pracy DBDA oprócz wspomnianych diagnostyk są raporty projektowe, przedstawiające sugerowane przez pakiet struktury danych. Struktury te n i e z a w s z e jednak spełniają wszystkie ograniczenia systemu IMS/VS; DBDA zmusza projektanta do samodzielnej weryfikacji sugerowanych struktur.

Oczekuje się, że pakiet DBDA będzie wykorzystywany w kilku iteracjach. Projektant w szczególności ma możliwość uruchamiania tylko niektórych etapów - np. diagnostyka wejścia. Może on wpływać na wyniki pracy pakietu przy pomocy prostego języka komend wymuszających generowanie określonych struktur.

DBDA można używać do tworzenia nowych projektów, włączania nowych programów do istniejącej bazy (pod warunkiem przekodowania DBD na język komend DBDA), eksperymentów z różnymi zestawami danych wejściowych (dla oceny wpływu nowych antycypowanych aplikacji na strukturę bazy) i w celach szkoleniowych.

Innym dosyć popularnym narzędziem typu transformator jest pakiet o nazwie DATA DESIGNER, rozprowadzany przez firmę J. Martina DDI [16]. Umożliwia on automatyczne integrowanie "views", dając na wyjściu z n o r m a l i z o w a n e - 3NF - r e l a c j e i indeksy pomocnicze. Każda z użytych nazw kontrolowana jest w oparciu o zewnętrzny słownik danych lub katalog nazw budowany przez DATA DESIGNER. Z dostępnego nam opisu trudno zorientować się jak konkretnie wygląda wejście tego programu (prawdopodobnie jakieś zależności funkcjonalne). Oceny projektu dokonuje się manualnie. Zmiany w projekcie możliwe są tylko przez modyfikacje danych wejściowych w pojedynczych views.

DATA DESIGNER włączony został do otoczenia programowo-projektowego SZBD ADR/Datacom System. Jest to tendencja dosyć typowa. Niemal każdy rynkowy SZBD lub Słownik-Skorowidz Danych ma narzędzie tego rodzaju.

Wspomnieć należy jeszcze o programie opisanym w [17], tworzącym w pełni a u t o m a t y c z n i e schemat b.d. dla SZBD ADABAS. Jest on o tyle ciekawy, że jego wejście pobiera dane zgromadzone i przeanalizowane przez PSL/PSA (ściślej: dane z pięciu raportów generowanych przez PSA). Dane te są przekształcane do znormalizowanej postaci r e l a c y j n e j - stąd nazwa programu: Relational Processor, z której następnie tworzone są zdania DDL systemu ADABAS, bez optymalizacji. Użycie przejściowej postaci relacyjnej pozwala na wymianę ostatniego z modułów tak, by tworzyć schematy dla innych SZBD. Z drugiej strony pamiętać należy, że

ADABAS ma bardzo proste schematy b.d., łatwe do generowania automatycznego; przejścia do innych modeli danych mogą być bardziej złożone.

Otrzymany w wyniku transformacji schematu konceptualnego (niekoniecznie przy użyciu narzędzia wspomagającego) projekt schematu, musi być zwykle uzupełniony o szereg parametrów fizycznych, decydujących o osiągach tworzonego systemu b.d. O jakości tak uzupełnionego projektu możemy przekonać się korzystając z e w a l u a t o r ó w.

Ewaluatory projektów baz danych wywodzą się w prostej linii od ewaluatorów plików. Do prekursorów tej grupy narzędzi można zaliczyć FOREM [18], ewaluatory A. Cardenasa [19], K.F. Silera [20] oraz FDA (File Design Analyzer) [21]. Narzędzia te mogą być przydatne do cząstkowej oceny projektu bazy danych w zakresie implementacji plików, dlatego poświęcimy nieco uwagi jednemu (chyba najlepszemu) z nich, mianowicie FDA.

FDA został opracowany w Uniwersytecie Michigan [21] [22]. Na wejściu pakietu podaje się trzy grupy informacji:

- opis danych (długość i liczba rekordów, długość pola klucza, liczba wartości klucza głównego i/lub kluczy pomocniczych itp.),
- opis pamięci systemu komputerowego (czas przesuwu głowic, czas obrotu pakietu, długość ścieżki) oraz
- opis procesu przetwarzania (częstość szukań, aktualizacji, usunięć rekordów, charakterystyka i częstość zapytań itp.).

W programie są zawarte modele pięciu organizacji plików: sekwencyjnej, indeksowo-sekwencyjnej, wielolistowej, odwróconej i bezpośredniej. Stosując metody analityczne, FDA wylicza obszar pamięci oraz czas przetwarzania (bez czasu CPU) w rozbiciu na operacje wyszukiwania (bez zapytań), aktualizacji, usuwania rekordów oraz obsługi zapytań.

Każdy z wyliczonych czasów dotyczy określonego rodzaju dostępu:

- pojedynczego (Single Access), zakładającego pracę jednoprogramową (czas szukania sprzętowy),
- rotacyjnego (Rotational Access), zakładającego pracę pojedynczego programu na pakietach dyskowych zawierających analizowany plik, przy trybie pracy wieloprogramowej całego systemu komputerowego,
- wielodostępu (Multi-Access), gdzie przyjmuje się dzielone wykorzystanie dysków przez wiele programów.

Metoda analizy zastosowana w programie FDA została zaczerpnięta z pracy S.B. Yao [23]. Jej istotą jest określenie, dla każdej organizacji pliku i każdej operacji, tzw. drzewa dostępów (access tree), czyli dekomponowanie poszczególnych operacji (zdefiniowanych dla całego pliku) na podoperacje, wykonywane w określonych obszarach funkcjonalnych pliku (skorowidzu, listach adresowych, obszarze głównym itp.).

Spośród wielu istniejących ewaluatorów projektów baz danych omówimy cztery, naszym zdaniem najbardziej reprezentatywne: DBPROTOTYPE II, DBD-DSS (Data Base Design - Decision Support System), DBDE (Database Design Evaluator) oraz DBAP (Data Base Analyzer and Predictor).

Pakiet o nazwie DBPROTOTYPE II [24] dostarczany przez firmę IBM do oceny projektu bazy danych zarządzanej przez SZBD IMS/VS zawiera trzy grupy programów:

- p o m o c n i c z e, analizujące dane wejściowe, ładujące prototypowe bazy danych (przeznaczone do dalszych badań) oraz podprogramy do modelowania transakcji użytkowych,
- m o d e l u j ą c e, za pomocą technik analitycznych, bazy IMS-owskie oraz pojedyncze transakcje (DBSPACE, DBCAP),
- a n a l i z u j ą c e wykonanie transakcji na próbnie załadowanych bazach danych (DBPROC, DBCALL).

Programy z grupy drugiej i trzeciej otrzymują na wejściu opisy baz danych oraz (wyjątkiem jest DBSPACE) transakcji. Natomiast charakterystyka systemu komputerowego (w tym pamięci) jest zawarta w samych programach.

Program DBSPACE oblicza obszar pamięci dla bazy oraz liczbę dostępów potrzebnych do wykonania elementarnych operacji DL/I.

Zadaniem programu DBCAP jest analityczne wyznaczenie czasu CPU oraz czasu operacji wejścia/wyjścia dla programu aplikacyjnego, w ujęciu globalnym oraz w rozbiciu na kilkanaście operacji niższego poziomu (na przykład operacje na danych, na indeksach pomocniczych, liczba przesyłanych bloków, liczba przesuwów głowic itp.).

DBPROC dostarcza statystyki odwołań programu użytkowego do załadowanej bazy danych w rozbiciu na typy segmentów oraz komendy DL/I. Dodatkową funkcją programu jest statystyka wykorzystania puli buforów oraz czasu wykonania programu.

Program DBCALL rozszerza możliwości DBPROC, umożliwiając zebranie statystyki działania programu przetwarzającego wiele baz fizycznych.

Metody komunikacji i ewaluacji wykorzystane w pakiecie DBPROTOTYPE II powodują, że jest to narzędzie duże i dość trudne w stosowaniu. Konieczność ładowania bazy w celu wykonania programów DBPROC i DBCALL, duża liczba parametrów wejściowych (w tym cały język opisu transakcji o składni innej od DL/I) i konieczność pracy w trybie wsadowym powodują, że ta koncepcja konstruowania ewaluatorów prawdopodobnie nie znajdzie naśladowców. Nie zmienia to jednak faktu, że DBPROTOTYPE II posiada liczne grono użytkowników z powodów, o których nadmieniono wcześniej.

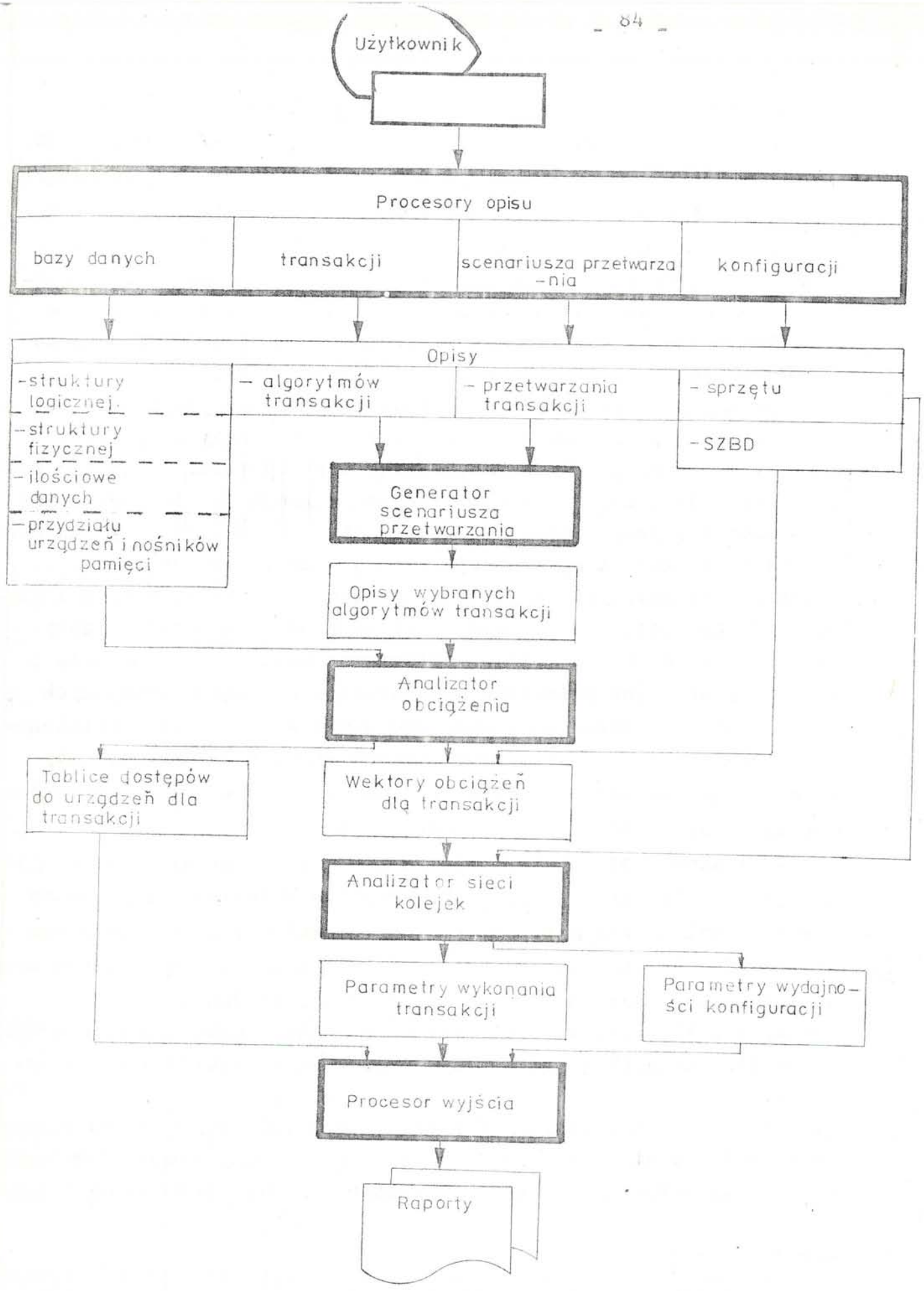
DBD-DSS [25] reprezentuje całkowicie inne podejście niż poprzednio opisany DBPROTOTYPE II. Jest to interakcyjny pakiet, służący do oceny projektu bazy danych typu CODASYL. W trakcie pracy systemu jest możliwa dynamiczna modyfikacja pięciu parametrów projektowych, przy zadanej wcześniej strukturze logicznej danych, opisie transakcji i opisie konfiguracji sprzętowo-programowej. Zmiennymi decyzyjnymi są: metoda lokowania rekordów, metoda implementacji kolekcji (setu), definiowanie punktów wejścia do bazy. Wspomaganie decyzji polega na obliczaniu kosztu przetwarzania i zapamiętywaniu (na polecenie użytkownika) rozwiązań wskazanych (najlepszych). Biorąc dodatkowo pod uwagę dość istotne ograniczenia w opisie bazy, należy uznać system DBD-DSS za interesujący choć niezbyt przydatny w praktyce etap rozwoju koncepcji narzędzi wspomagających tworzenie systemów baz danych.

Ewaluator projektu bazy danych DBDE [26] jest wynikiem kontynuacji (po FDA) prac grupy z Uniwersytetu Michigan. Przy pomocy technik analitycznych szacowany jest oczekiwany czas przetwarzania kwerend (w rozbiciu na czas CPU, operacji wejścia/wyjścia oraz oczekiwania w kolejkach) dla projektu bazy danych w SZBD IDS. Novum, w porównaniu z poprzednio omówionymi ewaluatorami, stanowi tu analiza kolejek w warunkach równoległego przetwarzania transakcji.

Ostatnim z omawianych tu pakietów programowych typu ewaluator jest DBAP[1]. Jest to system do oceny systemu bazy danych IDMS DB/DC, obejmujący wszystkie istotne, z punktu widzenia eksploatacji bazy,

---

[1] Wcześniejsza, prototypowa wersja pakietu DBAP została opracowana w ramach projektu DATAID 11 przez zespół pod kierunkiem W. Staniszkisa w CRAI i jest bardziej znana pod nazwą EOS [28].



Rys. 1. Ogólna architektura systemu DBAP (EOS).

aspekty jej projektu[1]. Architekturę systemu DBAP przedstawiono na rys. 1.

Wejściem do DBAP-a są opisy statyki i dynamiki systemu bazy danych, czyli:

- struktury logicznej (na poziomie JOD schematu),
- struktury fizycznej (na poziomie JOP schematu),
- ilościowy danych,
- przydziału urządzeń i nośników pamięci (na poziomie języka sterowania urządzeniami i nośnikami, np. JCL),
- algorytmów transakcji (na poziomie JMD),
- scenariusza równoległego przetwarzania transakcji,
- konfiguracji sprzętowo-programowej systemu komputerowego.

Najważniejszymy elementami systemu DBAP są analizator obciążenia (Workload Analyzer) oraz analizator sieci kolejek (Queueing Natwork Analyzer).

Analizator obciążenia jest modelem SZBD. Jego głównymi funkcjami są: analiza algorytmu transakcji i określenie częstości wykonania poszczególnych komend JMD, podział każdej komendy JMD na elementarne operacje logiczne, umożliwiające obliczenie kosztu wykonania komendy, oraz synteza wykonanych analiz w postaci łącznego kosztu wykonania transakcji.

Analizator sieci kolejek jest modelem systemu komputerowego i pozwala ocenić funkcjonowanie systemu bazy danych w warunkach równoległego wykonywania wielu transakcji.

Wyniki ewaluacji wyprowadzane przez procesor wyjścia umożliwiają bardzo szczegółową analizę zarówno projektu bazy danych (statycznego) jak i algorytmów transakcji.

Omówiony pakiet DBAP jest, jak do tej pory, najwszechstronniejszym i najbardziej dokładnym ewaluatorem projektu bazy danych [29]. Jego handlowa wersja jest przeznaczona do wspomagania prac projektowych dla SZBD IDMS, lecz architektura i wykorzystane algorytmy umożliwiają stosunkowo łatwą adaptację pakietu do potrzeb innych CODASYL-owskich SZBD, w szczególności RODAN-u.

Trzecią wyróżnioną przez nas grupą narzędzi wspomagających projektowanie baz danych są o p t y m a l i z a t o r y. Łączą one funkcje transformatorów i ewaluatorów, zwiększając tym samym zakres wspomagania.

---

[1] Koncepcja pakietu jest zgodna z propozycją modelu ewaluatora K.C. Sevcik'a [27].

Podstawowe różnice między obecnie stosowanymi optymalizatorami a ewaluatorami są następujące:

- optymalizator oczekuje na wejściu nie gotowego (i w pewnym sensie kompletnego) projektu, lecz specyfikacji projektowych (model konceptualny + dane ilościowe),
- optymalizator nie uwzględnia - broniąc się tym samym przed nadmierną złożonością zadania optymalizacyjnego - wielu istotnych aspektów funkcjonowania systemu b.d. (np. konfiguracji, alokacji danych, wieloprogramowości) a inne z nich formułuje w sposób bardziej abstrakcyjny (uproszczony),
- parametry oceny rozwiązań są znacznie - w stosunku do ewaluatora - uproszczone i zagregowane zwykle do jednej funkcji celu,
- kryteria optymalizacji są, niemal bez wyjątków, sformułowane przez twórców narzędzia i "zaszyte" w kodzie programu.

Wymienione cechy sprawiają, że obie klasy narzędzi będą długo jeszcze pokojowo koegzystować.

Optymalizator opisać można podając:

- zakres informacji koniecznej na wejściu (model konceptualny obejmujący transakcje, parametry ilościowe danych i transakcji),
- zakres optymalizacji, inaczej: kompletność generowanego projektu (przykładowo może on ograniczać się tylko do agregowania danych, metod implementacji kolekcji, metod lokowania itp.),
- zbiór uwzględnianych ograniczeń (np. na zajętość pamięci, czas przetwarzania),
- funkcję celu (czas przetwarzania wszystkich kwerend, zajętość pamięci, koszt),
- metodę szukania rozwiązania optymalnego.

Ograniczenia i funkcja celu są wzajemnie komplementarne np. optymalizuje się czas przetwarzania przy zajętości pamięci nie przekraczającej pewnej wartości.

Cztery pierwsze charakterystyki optymalizatora możemy nazwać zadaniem optymalizacyjnym. Oto przykład zadania optymalizacyjnego tak jak zostało ono sformułowane i rozwiązane przez M. Mitomę [30].

Znany jest zestaw d a n y c h e l e m e n t a r n y c h (charakteryzowanych przez nazwę, długość i liczbę przyjmowanych wartości), z w i ą z k ó w między danymi (opisanych przez nazwę, składowe, liczebność, charakter ilościowy związków) oraz t r a n s a k c j i (rozumianych jako zestaw elementarnych operacji, takich jak np. FIND Y,X,R,f - znajdź wszystkie wartości danej Y połą-

czone z zadaną wartością X poprzez związek R; f oznacza częstość wykonania tej operacji w transakcji. Dla każdego związku R należy wybrać jedną z dopuszczalnych jego implementacji (tu podana jest lista dopuszczalnych implementacji np. redundantne pamiętanie wartości X-a przy każdym Y-ku z którym są związane).

Funkcja celu została zdefiniowana następująco:

$$\text{minimalnej } AC = \sum_{j=1}^{ND} \sum_{i=1}^{NUMF_j} d_{ij}\, AC_{ij}$$

przy ograniczeniach:

$$SC = \sum_{j=1}^{ND} \sum_{i=1}^{NUMF_j} d_{ij}\, SC_{ij} \leqslant MAX_{SC}$$

$$\text{i} \sum_{i=1}^{NUMF_j} d_{ij} = 1 \quad \text{dla każdego } j = 1, 2, \ldots, ND,$$

gdzie: AC - łączny koszt dostępów (logicznych),

AC_{ij} - koszt dostępu przy i-tej implementacji j-tego związku,

SC - łączny koszt pamięci,

SC_{ij} - koszt pamięci przy i-tej implementacji j-tego związku,

ND - liczba związków,

$NUMF_j$ - liczba alternatywnych implementacji j-tego związku,

d_{ij} = 1 jeśli i-ta implementacja została wybrana dla j-tego związku

0 w przeciwnym przypadku.

W istniejących narzędziach klasy optymalizator stosuje się trzy podstawowe metody szukania rozwiązania optymalnego: algorytmy programowania matematycznego, szukanie wyczerpujące w przestrzeni rozwiązań dopuszczalnych oraz szukanie w przestrzeni ograniczonej najczęściej za pomocą przyjętych reguł heurystycznych.

Do narzędzi wykorzystujących pierwszą z wymienionych metod należą trzy zaimplementowane modele powstałe w Uniwersytecie Michigan.

Pierwszym jest praca M.F. Mitomy [30], z której zaczerpnęliśmy przedstawiony przykład formułowania zadania (w kategoriach programowania całkowitoliczbowego). Jest ono rozwiązywane metodami programowania dynamicznego, po przekształceniu go do równoważnego problemu przejścia sieci. Program Mitomy jest przeznaczony do generowania szkieletów CODASYL-owskich baz danych, zawierających wykaz rekordów (wraz z danymi elementarnymi je tworzącymi) oraz wykaz kolekcji.

Istotnym rozszerzeniem pracy Mitomy jest model E. Bereliana [31]. Rozszerzenie to polega na "uczuleniu" optymalizatora na liczbę dostępów fizycznych (zamiast logicznych u Mitorny) oraz na rozszerzeniu zakresu optymalizacji o metody implementacji kolekcji (różne opcje odsyłaczy. Zadanie projektowe jest tu rozwiązywane metodą podziału i ograniczeń.

Pracą rozszerzającą możliwości obydwu poprzednich jest LDO (Logical Database Optimizer) oprogramowany na podstawie badań S. Purkayastha [32]. Najważniejszym usprawnieniem wprowadzonym w LDO jest zmodyfikowanie metod implementowania związków między danymi. Dzięki tym zmianom poszerzono przestrzeń poszukiwań (przy zachowaniu reguł poprawności wygenerowanego schematu) co może prowadzić do znalezienia poprawnego projektu, uznawanego przez narzędzia Mitorny i Bereliana za niepoprawny. Drugą konsekwencją wprowadzonych zmian jest możliwość projektowania zarówno baz sieciowych (CODASYL-owskich) jak i hierarchicznych (IMS).

Podobny sposób formułowania i rozwiązywania problemu optymalizacji jak w poprzednich trzech pracach reprezentuje teoretyczny, nieoprogramowany model P. De, W.D. Hasemana i C.H. Kriebela [33]. Podano tu algorytmy przekształcania schematu relacyjnego w schemat sieciowy, a następnie optymalizujący model implementacji kolekcji w bazie sieciowej ze względu na s u m ę kosztów pamiętania danych i czasu przetwarzania, przy górnych ograniczeniach nałożonych na te dwie wielkości.

Pakiet programowy S. Maroha i D. Severance'a [34], który powstał w Uniwersytecie Minnesota, charakteryzuje się większym, niż w poprzednio opisanych narzędziach, zakresem optymalizacji i większą "czułością" oceny projektu. Optymalizacja dotyczy agregacji danych oraz wyboru metody implementacji pliku (rozumianego tu jako element struktury fizycznej bazy danych). Dzięki zastosowaniu uogólnionej metody opisu struktury pamięci w bazie danych (frame memory [35]), potencjalny zakres zastosowań pakietu jest bardzo sze-

roki. Opis jest adekwatny do różnych systemów organizacji pamięci - IDS, ADABAS, DMS1100, VSAM.

Pakiet składa się z trzech modułów funkcjonalnych:

- segmentacji rekordów,
- wyboru implementacji (organizacji) pliku i algorytmu szukania
- implementacji w pamięci (alokacji) i oceny projektu.

Metoda segmentacji rekordów i wyboru długości bloku, opisana w [36] pozwala dokonać podziału danych elementarnych na rekordy w zależności od częstości ich użycia. Natomiast algorytm wyboru organizacji pliku i metody szukania jest autorstwa R.A. Duhne [37]. Problem sformułowany i rozwiązywany jest klasycznie, jako zadanie programowania matematycznego.

Ponieważ w trakcie działania pakietu sterowanie jest wielokrotnie przekazywane między modułami, generowane są różne rozwiązania pośrednie. Kolejność i ilość wywołań poszczególnych modułów jest ustanowiona arbitralnie (heurystycznie?) przez autorów. Pakiet kończy pracę, gdy dwukrotne kolejne uruchomienie modułu segmentacji nie przynosi zmiany w projekcie. Jako rozwiązanie ostateczne jest przyjmowane najlepsze z badanych (niekoniecznie ostatnie i niekoniecznie optymalne).

J. Carlis i S. March [38] ulepszyli produkt (nazwany obecnie DBDS-Database Design System), dodając heurystyczną procedurę generowania potencjalnie dobrych agregatów danych (rekordów).

W grupie narzędzi optymalizujących stworzono wiele cząstkowych modeli a każdy miesiąc przynosi nowe. Niestety, znaczna część z nich, ze względu na przyjmowane ograniczenia, ma niewielkie znaczenie praktyczne.

Narzędzia wspomagające dla SZBD HADES

System HADES jest "bardzo podobny" do IMS/360, stąd wiele narzędzi jakie IBM przygotował dla IMSu jak DBDA, DBPROTOTYPE, IMSMAP itp. można by wykorzystać do wspomagania prac projektowych w HADESie - gdyby były dostępne. Ponieważ nie są, a zapotrzebowanie na takie narzędzia istnieje, zespół z Akademii Ekonomicznej we Wrocławiu zaprojektował i zaimplementował pakiety programowe LSA i MAP, wzorowane częściowo na DBDA i IMSMAPie.

LSA unika niektórych wad DBDA - jest pakietem konwersacyjnym, w którym język komend zastąpiono punktami decyzyjnymi: projektant jest na bieżąco informowany o sytuacjach, mogących wymagać jego interwencji, dostarcza mu się też podstawowych danych niezbędnych

do podjęcia decyzji. LSA lepiej weryfikuje sugerowane na wyjściu struktury (przypomnijmy, że DBDA żądało, by projektant samodzielnie sprawdził ich poprawność - zgodność z ograniczeniami IMSu) i ma bogatsze wyjście (generuje szkielet DBD, tworzy graf wszelkich możliwych powiązań między segmentami, których wymagają programy użytkowe). Do LSA wprowadzono rozwiązania ułatwiające przystosowanie go do nie HADESowskich modeli danych. Bezpośrednio z DBDA przejęto wejście podstawowe (formularze opisu zapotrzebowań) oraz większość raportów diagnostycznych (ze względu na ograniczenia sprzętowe - mały, niepodzielny ekran monitora - zdecydowano się pozostawić papierowe raporty).

MAP rozszerza funkcje dostępne w IMSMAPie (głównie raportujące) o alokowanie odpowiednich obszarów pamięci dyskowej dla zaprojektowanych baz danych.

Oba wspomniane, a także inne narzędzia zaprojektowane w Akademii Ekonomicznej we Wrocławiu opisane są w [39] (artykuł o LSA jest niekompletny i prezentuje pierwszą, starszą wersję pakietu).

Niektóre trendy rozwojowe w dziedzinie narzędzi wspomagających konstruowanie systemów b.d.

W przedstawionym przez nas przeglądzie niektórych ważniejszych narzędzi wspomagających wyraźnie rzuca się w oczy ich niekompatybilność. Każde (niemal każde) z narzędzi wymyśla od początku rodzaj i format danych potrzebnych mu do realizowania swych funkcji i produkuje wyniki, z których żadne inne programy wspomagające nie mogą bezpośrednio skorzystać. Dzieje się tak nawet w przypadku, gdy kilka produktów wyszło spod ręki tego samego twórcy (np. DESIGNER i DBD-DSS Gerritsena). Ich późniejsze łączenie w "ciągi technologiczne" jest bardzo trudne, czasem niemożliwe bez gruntownego przeprojektowania obu łączonych końców.

Uważamy za ważny trend próby harmonijnego łączenia narzędzi w zintegrowane środowiska konstrukcyjne. Obok spraw "merytorycznych" - odpowiedniej strukturalizacji cyklu życia systemu, definiowania "logicznych" złączy programowych między narzędziami i struktur metabazy przechowującej informacje o projekcie - nie mniej ważne są zagadnienia "implementacyjne" - takiego definiowania "fizycznych" złączy, które zapewnia prostą integrację (przykład Unixa i związanych z nim narzędzi jest tu znamienny; SPS również nieprzypadkowo powstało w otoczeniu Unixa).

Wyraźnie nastaje moda na stosowanie technik sztucznej inteligencji w każdej niemal gałęzi informatyki. Także w zakresie wspomagania konstruowania systemów b.d. pojawiło się kilka publikacji na ten temat.

Holsapple et al. w [40] opisuje prototyp systemu, który na podstawie formalnie opisanych raportów (jakie tworzyć ma powstająca aplikacja), generuje dopuszczalny schemat b.d. dla rozszerzonego modelu sieciowego. Nowością jest tu metoda otrzymania tego schematu: miast zakodowanych klasycznie algorytmów transformujących, system projektowy używa reguł produkcji, tak jak typowe systemy ekspertowe.

Colombetti et al. [41] usiłują stworzyć system, który ze zdań języka naturalnego potrafiłby sformułować bardziej formalne zapotrzebowania użytkowników, a być może wygenerować schemat konceptualny taki jak w INCOD-DTE. Dotychczasowe ich rezultaty są jednak mało spektakularne - problem komunikowania zapotrzebowań w języku naturalnym okazuje się być trudniejszy od problemu dostępu do baz danych w takimż języku.

Obiecujące wydają się doświadczenia z systemem IMT (Information Modelling Tool)[42], służącym do tworzenia i weryfikacji modelu konceptualnego danych. Językiem opisu jest język logiki; tworzony model konceptualny traktowany jest jako teoria logiczna, zaś do kontroli jej spójności używa się programu dowodzącego twierdzenia metodą rezolucji. Podobnych możliwości dostarcza Prolog, umożliwiając też budowanie szybkich prototypów.

Za trzeci istotny trend można uważać pewne prace teoretyczne nad generowaniem projektów optymalnych. Jak pamiętamy zagadnienia stworzenia optymalnego projektu grożą eksplozją kombinatoryczną. Po to, by algorytmy kończyły swe działanie w rozsądnym czasie stosowano mało realistyczne ograniczenia zadania optymalizacyjnego lub arbitralne często metody zmniejszania przeszukiwanej przestrzeni rozwiązań dopuszczalnych. Whang et al. opracowali w tzw. teorii rozdzielności (separability) [43] zestaw własności, jakie muszą być spełnione przez mechanizmy SZBD, aby problem projektowy można było podzielić na rozłączne podproblemy - znacznie mniejsze i łatwiej rozwiązywalne. Podział określony przez warunki teorii rozdzielności ma tę cechę, że zbiór optymalnych rozwiązań podproblemów jest optymalnym rozwiązaniem całego problemu.

Whang et al. wskazują, że znaczne podzbiory mechanizmów w modelach sieciowych i relacyjnych spełniają warunki rozdzielności.

BIBLIOGRAFIA

[1] D. Barstow et al. - Interactive Programming Environments, McGraw-Hill, New York 1983.

[2] Computer vol. 14 no. 4 April 1981 - special edition on programming environments.

[3] B. Boehm et. al. - A Software Development Environment for Improving Productivity. Computer vol. 17 no. 6 June 1984 p. 30-44.

[4] D. Teichroew, E.A. Hershey III - PSL/PSA: A Computer-Aided Technique for Structured Documentation and Analysis of Information Processing Systems. IEEE Trans. on Software Engineering, SE-3 no. 1, January 1977, p. 16-34.

[5] M. Alford - SREM at the Age of Eight. Computer vol. 18 no. 4, April 1985, p. 36-46.

[6] N. Delisle et al. - Tools for Supporting Structured Analysis w: 7 p. 11-20.

[7] H.J. Sohneider, A. Wasserman (eds.) - Automated Tools for Information Systems Design North-Holland, Amsterdam 1982.

[8] Structured Architect - ISDOS Ino., Ann Arbor, Michigan.

[9] E.P. Chau, F. Lochovsky - A Graphical Database Design Aid using the Entity-Relactionskip Model w 10 , p. 295-310.

[10] P.P. Chen (ed.) - Entity-Relationskip Approach to System Analysis and Design, North-Holland, Amsterdam 1980.

[11] S. Ceri(ed.) - Methodology and Tools for Data Base Design, North-Holland, Amsterdam 1983.

[12] P. Atzeni et al. - INCOD-DTE: A System for Interactive Conceptual Design of Data, Transactions and Events w 11 p. 205-228.

[13] A. Albano, R. Orsini - Dialogo: an Interactive Environment for Conceptual Design in Galileo, w 11 p. 229-253.

[14] R. Gerritsen - A Preliminary System for the Design of DBTG Data Structures CACM vol. 18 no. 10, October 1975, p. 551-557.

[15] N. Raver, G. Hubbard - Automated Logical Data Base Design: Concepts and Applications. IBM System Journal, vol. 16 no. 3 p. 287-312.

[16] Data Designer - Database Design Incorporation, 1981.

[17] I. Spiegler - Automating Database Construction. Data Base vol. 14 no. 3, Spring 1983, p. 21-29.

[18] M. Senko, Lum V.Y., Owens P.J.: A File Organisation Evaluation Model (FOREM) Proc. of 1968 IFIP Congress, p. 514-519.

[19] A. Cardenas: Evaluation and Selection of File Organization - A Model and System. CACM, vol. 16, no. 9, September 1973, p. 540-548.

[20] K. Siler: A Stochastic Evaluation Model for Database Organizations in Data Retrieval Systems. CACM, vol. 19, no. 2, February 1976, p. 84-95.

[21] T. Teorey, Sundar-Das K.: Application of Analytical Model to Evaluate Storage Structures. Proc. of 1976 ACM-SIGMOD Cof., p. 9-19.

[22] T. Teorey, Sundar-Das K.: Detailed Specifications for the File Design Analyzer. Syst. Engin. Lab. Tech. Report 87, Univ. of Michigan 1975.

[23] S.B. Yao: An Attribute Based Model for Database Access Cost Analysis. ACM Trans. on Database Systems, vol. 2, no. 1, March 1977, p. 45-67.

[24] DBPROTOTYPE II. Program Description/Operations Manual. PN 5796-PJK, 3rd ed., 1980.

[25] T. Gambino, Gerritsen R.: A Data Base Design Decision Support System. Proc. of 3rd Int. Conf. on VLDB, Tokyo, p. 534-544.

[26] T. Teorey, Oberlander L.B.: Network Database Evaluation Using Analytical Modelling. Proc. of 1978 AFIPS Nat. Comp. Conf.

[27] K. Sevcik: Data Base Performance Prediction Using an Analytical Model. Proc. of 7th Int. Conf. on VLDB, Cannes, 1981, p. 182-198.

[28] W. Staniszkis, Rullo P., Orlando S.: Transaction Workload Analysis in the CODASYL Data Base Performance Predictor EOS. Proc. of 6th Int. Sem. on DBMS, Matrafüred, Hungary 1983, p. 97-120.

[29] W. Staniszkis: Stan prac i trendy rozwojowe w dziedzinie systemów zarządzania bazą danych. Szkoła zastosowań informatyki - IOPM'85. Kołobrzeg 1985.

[30] M. Mitoma, Irani K.B.: Automatic Data Base Schema Design and Optimization. Proc. of 1st Inf. Conf. on VLDB, Framington, Ma. p. 286-328.

[31] E. Berelian, Irani K.B.: Evaluation and Optimization of Database Structures. Proc. of 3rd Int. Conf. on VLDB Tokyo 1977, p. 545-555.

[32] K.B. Irani, S. Purkayastha - A Designer for DBMS - Prossable Logical Database Structures, Proc. of 5th Int. Conf. on VLDB, 1979.

[33] P. De, Haseman W.D., Kriebel C.H.: Toward an Optimal Design of a Network Database from Relational Descriptions. Operations Research, vol. 26, no. 5, September 1978, p. 805-823.

[34] S. March, Severance D.G.: A Mathematical Modeling Approach to the Automatic Selection of Database Designs. Proc. ACM Int. Conf. on Manag. of Data (SIGMOD) 1978, p. 112-126.

[35] S. March et al. - Frame Memory: a Storage Architecture to Support Rapid Design and Implementation of Efficient Databases, ACM Trans. on Database Syst. vol. 6 no. 3, September 1981, p. 441-463.

[36] S. March, Severance D.G.: The Determination of Efficient Record Segmentations and Blocking Factors for Shared Data Files, ACM Trans. on Database Syst., vol. 2, no. 3, September 1977, p. 279-296.

[37] R.A. Duhne, Severance D.G.: Selection of an Efficient Combination of Data Files for a Multiuser Database. Proc. of AFIPS National Computer Conf., 1978.

[38] J.V. Carlis, March S.T., Dickson G.W.: Physical Database Design: A DSS Approach. Information Management, vol. 6, no. 4, 1983, p. 211-224.

[39] Biuletyn Techniczno-Informacyjny MERA, nr 6/1985, czerwiec 1985, w całości poświęcony narzędziom dla systemu HADES.

[40] C. Holsapple et al. - A Consulting System for Data Base Design. Information System vol. 7 no. 3, 1988, p. 281-296.

[41] M. Colombetti et al. - NDLA: A Natural Language Reasoning System for the Analysis of Data Base Requirements, w 11 , p. 163-179.

[42] B. Lundberg - IMT - an Information Modelling Tool, w 7 p. 21-30.

[43] K.Y. Whang et al. - Separability - an Approach to Physical Database Design. Proc. 7th Int. Conf. on VLDB, Cannes 1981, p. 320-332 lub IEEE Trans. on Computers C-33, no. 3, 1984.

P E R C E P T O W Y S Y S T E M O P I S U W I E D Z Y

Michał Sobolewski
Instytut Biocybernetyki i Inżynierii
Biomedycznej PAN
00-818 W-wa, ul.KRN 55

1. Fakty, procesor i rzeczywistość

1.1. Fakty i procesor

Gromadzenie i opisywanie faktów /faktografia/ z określonej dziedziny, przy dużym ich zbiorze, w celu efektywnego wykorzystywania /wyszukiwanie, interpretacja, konsultacja, projektowanie, przewidywanie, planowanie, nadzorowanie itd./, przez licznych użytkowników, wymaga zastosowania procesora /komputera/. Procesor powinien umożliwić zrealizowanie takiego systemu faktograficznego, przy pomocy którego gromadzenie, przechowywanie, modyfikowanie i wyszukiwanie faktów jest dostępne dla szerokiego kręgu użytkowników bez konieczności uczenia się przez nich informatycznej techniki działania i obsługi procesora. Doświadczenia krajów o wysoko rozwiniętej technologii komputerowej wykazały, że konieczność przyswojenia, przez użytkowników systemu faktograficznego, informatycznej techniki i obsługi komputera, które często wychodzą daleko poza ich dziedzinę zawodową, stwarza barierę psychologiczną i w konsekwencji nawet teoretycznie najlepsze systemy /programy/ nie są wówczas stosowane przez użytkowników. Natomiast gdy jednak są stosowane, to nieuwzględnienie czynnika ludzkiego powoduje niską efektywność wykorzystania ze względu na nienaturalny protokół obsługi wyrażony, z punktu widzenia użytkownika, w tajemniczych informatycznych terminach

a nie w terminach ludzkich.

System faktograficzny nie może ograniczać się do technik konwencjonalnego programowania systemów /jako "czarnych skrzynek"/ udzielających dobrych odpowiedzi na sformułowane przez klienta zadanie. Jest to niepełny obraz problemu, nie uwzględniający następujących istotnych elementów /rys.1.1/ [11,12,13,14,15] :

1. użytkownik musi komunikować się z systemem w sposób jak najbardziej dla niego naturalny we wszystkich trybach użytkowania.
2. klient żąda wyjaśnień co do działania systemu na równi z odpowiedziami.
3. system musi posiadać łatwość rozszerzania i polepszania bazy faktów w procesie interakcyjnego współdziałania przy udziale specjalistów przedmiotowej dziedziny /a nie projektantów systemu/.
4. system powinien dostarczać nagromadzone skodyfikowane fakty dla ludzi zajmujących się przedmiotową dziedziną.

Wymienione cztery cechy wymagają specyficznych języków programowania systemów faktograficznych, które umożliwiają względnie łatwe ich organizowanie i jak najbardziej naturalne komunikowanie się z nimi we wszystkich trybach ich użycia. Cechy te jednocześnie wskazują na bezpośrednią nieprzydatność konwencjonalnych języków programowania do tego celu. Przykładami języków przeznaczonych do programowania systemów faktograficznych są: RITA [1,8], ROSIE[2,8], AGE[3], EXPERT[5,6,9], R1/XCON/[7], Hearsay-III[10], czy EMYCIN[4]. Języki tego typu umożliwiają programowanie zadań rozwiązywanych w oparciu o bazę faktów /nazywaną w sztucznej inteligencji bazą wiedzy/ a nie, jak w przypadku konwencjonalnych języków, o znany algorytm. Otrzymujemy więc nową jakość, praktyczne rozwiązanie wielu trudnych

zadań, dla których algorytm explicite nie musi być znany.

Sytemy faktograficzne charakteryzujące się powyższymi czterema cechami powinny być oparte na spójnej metodzie reprezentacji danych na wszystkich poziomach organizacji bazy wiedzy przy założeniu, że możliwe będzie przetworzenie zadań języka konwersacji /np. zdań języka naturalnego o uproszczonej składni/ na wewnętrzną reprezentację danych systemu. W swej istocie procesor może tylko przetwarzać dane symboliczne /obiekty procesora/, aczkolwiek o dowolnej strukturze i w dowolny sposób. Powstaje zasadnicze pytanie z punktu widzenia procesora - jakie obiekty powinien przetwarzać procesor systemu faktograficznego i w jaki sposób, przy założeniu, że danymi wejściowymi i wyjściowymi procesora są zadania języka konwersacji, będące lingwistyczną interpretacją faktów i celów /wymieniona wcześniej pierwsza cecha/. Ciąg zdań wyrażonych tekstem lub mową nazwiemy interpretem lingwistycznym.

Struktura interpretów lingwistycznych jest hierarchiczna. Ustopniowany układ, który można wyróżnić w każdym interprecie zawiera poziom leksykalny, syntaktyczny i semantyczny /nazywamy również poziomem wiedzy/. Pierwsze dwa poziomy wyrażają opis interpretu jako tworu lingwistycznego, natomiast poziom trzeci wyraża wiedzę o faktach, które zachodzą lub celach, które chcemy osiągnąć. Na poziomie wiedzy interpretów lingwistycznych można wyróżnić cztery podstawowe podpoziomy: deklaratywny, proceduralny, sterujący i celów. Podpoziom pierwszy wyraża własności /deklaracje/ bytów rzeczywistości. Drugi wyraża stwierdzenia /reguły/ określające sposób przetwarzania deklaracji i celów, dokładniej, z jakich znanych deklaracji można otrzymać nowe i jakie cele można zastąpić nowymi podcelami. Trzeci podpoziom wyraża stwierdzenia /sterowania/ wyrażające sposób użycia reguł do przetwarzania deklaracji i celów. Ostatni, czwarty podpoziom wyraża pytanie /py-

tania/, na które poszukujemy odpowiedzi i od którego /których/ zależy sposób przetwarzania danych poziomu deklaratywnego i celów opisywany na poziomie sterującym.

Obserwator procesora powinien zdawać sobie sprawę, że przypisując procesorowi przetwarzanie interpretów lingwistycznych, w rzeczywistości, należy przypisać mu przetwarzanie obiektów je reprezentujących i to takich, z których procesor powinien wiedzieć wszystko to, co wie obserwator z interpretów lingwistycznych. Jeśli obserwator zna hierarchię interpretu lingwistycznego, to powinna się ona przenieść na obiekt procesora /rys.1.2./.

Złożone i trudne jest reprezentowanie poziomu wiedzy interpretu ze względu na złożoną gramatykę języków naturalnych i niezupełnie jednoznaczną ich semantykę. Gramatyki języków naturalnych są w zasadzie sztuką wstawiania odpowiednich słów w odpowiednie miejsca. Sztuką nie są systemy logiczne, będące systemami sformalizowanymi o jednoznacznej składni i semantyce. Dlatego też zwykle posługujemy się zdaniami specjalnie skonstruowanego systemu logicznego zamiast trudnych do analizy zdań języka naturalnego. Interprety lingwistyczne wyrażone zdaniami języka logicznego będziemy nazywali interpretami logicznymi. Można spodziewać się, że dla interpretów logicznych o jednoznacznej składni i semantyce łatwiej będzie znaleźć właściwą reprezentację /obiekty procesora/.

Wprowadzenie interpretów logicznych, jako pośrednich obiektów pomiędzy interpretami lingwistycznymi a obiektami procesora, wyznacza specyficzną rolę logiki. Rola ta sprowadza się do opisu poziomu wiedzy interpretów lingwistycznych z jednej strony i wskazania, zrozumiałych w jej terminach, struktur obiektów procesora z drugiej strony. Zatem logika powinna być traktowana jako system opisu reprezentowanych obiektów /struktura interpre-

tów logicznych jest liniowa/, a nie jako system reprezentacji obiektów procesora /o strukturze hierarchicznej/ w ogóle.

Z powyższych rozważań wynika, że krokiem ukierunkowującym problem reprezentacji wiedzy jest sposób opisu reprezentowanych obiektów. Poniżej rozważamy istotę pojęcia faktu, w odniesieniu do rzeczywistości, w celu wskazania semantyki języka sformalizowanego, który stosować będziemy do wyrażania faktów i celów.

1.2. Rzeczywistość i fakty

Coś odpowiada rzeczywistości, jeśli to coś jest zgodne z faktami. Przez zgodność z faktami zwykle rozumiemy zgodność z tym, co zaszło lub zachodzi w rzeczywistości.

Chcąc reprezentować w procesorze to, co zaszło lub zachodzi w rzeczywistości /fakty/, musimy odpowiedzieć czego te zajścia dotyczą. Odpowiedź prowadzi do pojęcia bytu - wszystkiego cokolwiek istnieje, jednocześnie wyzwalając odwieczne dysputy na temat charakteru i struktury rzeczywistości.

Nie będziemy prowadzili rozważań ontologicznych, bo nie jest możliwe wprowadzenie do procesora bytów rzeczywistości, lecz jedynie symboli je reprezentujących. Symboliczna reprezentacja bytów, a więc określona ich interpretacja /nie wdając się w to jakie one są i czy w ogóle istnieją/ jest wynikiem złożonego procesu poznawczego nazywanego percepcją. Proces ten polega na odzwierciedleniu przez człowieka przedmiotów, zjawisk i procesów zachodzących wskutek działania na narządy zmysłowe określonych bodźców. Te bodźce, czy też byty te bodźce wytwarzające - jest to w naszych rozważaniach bez znaczenia, nazywać będziemy istnieniami i oznaczać przez ENT a pojedyncze istnienia przez e, e_1, e_2, ... Reakcje na nie, będące wynikiem percepcji nazwiemy perceptami oznaczając przez PER, zaś pojedyncze percepty przez pe, pe_1, pe_2, ... Dalej przez percepcję rozumieć będziemy relację per, $per \subseteq ENT \times PER$ określającą czy dane istnienie e i percept pe pozostają w relacji percepcji per, $/e,pe/ \in per$.

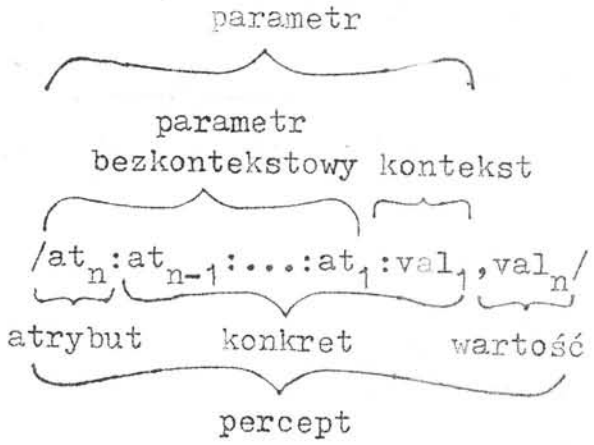
Zakładamy, że percept jest parą /pa,val/, której pierwszy element jest parametrem perceptu pa charakteryzującym pewne istnie-

nie e z wartością val będącą drugim elementem perceptu, takiego że /e,/pa,val//$\in$ per. Zbiór parametrów perceptów PER oznaczać będziemy przez PAR, a jego elementy przez pa, pa_1, pa_2, ... , natomiast zbiór wartości przez VAL a jego elementy przez val, val_1, val_2, ... Wychodząc z założenia, że istnienie wskazać możemy wyłącznie przez podanie wartości i atrybutów mu przypisanych, parametr /wskazujący istnienie/ traktujemy jako ciąg, którego pierwszym elementem jest wartość nazywana kontekstem, pozostałe jego elementy są atrybutami. Zbiór atrybutów oznaczamy przez AT a jego elementy przez at, at_1, at_2, ...

W celu oddzielenia od siebie elementów parametrów używać będziemy znaku dwukropka /:/, jako separatora elementów ciągu. Zazwyczaj n-ty atrybut parametru /n+1-szy element parametru/ oznaczać będziemy przez at_n, a elementy ciągu będziemy zapisywali od strony prawej do lewej. Atrybuty, kontekst i wartość perceptu zapisywać będziemy dużymi literami. Zgodnie z przyjętym sposobem zapisu czteroelementowy parametr z kontekstem K1 i atrybutami KOCIOŁ, PARA i CIŚNIENIE, jako drugi, trzeci i czwarty element parametru odpowiednio, zapiszemy w postaci

CIŚNIENIE:PARA:KOCIOŁ:K1

Powyższy parametr czytamy:"ciśnienie pary w kotle K1".
Jeśli pa jest parametrem perceptu pe, to ostatni element parametru pa nazywać będziemy atrybutem perceptu pe, natomiast parametr pa bez ostatniego elementu konkretem perceptu pe, przyczym konkret jest również parametrem /wskazuje coś konkretnego/. Zbiór konkretów parametrów PAR oznaczać będziemy przez CON a jego elementy przez c, c_1, c_2, ...
Wprowadzoną powyżej terminologię perceptu ilustruje poniższy schemat:



Zwróćmy uwagę, że w języku potocznym zwykle parametrem nazywamy ciąg atrybutów nie przywiązując uwagi do jego kontekstu. Np. mówiąc "ciśnienie pary jest parametrem kotła" /w naszej notacji CIŚNIENIE:PARA:KOCIOŁ/ mamy na uwadze jakiś ustalony kocioł lub dowolny kocioł.

Parametry z opuszczonym kontekstem /np.CIŚNIENIE:PARA:KOCIOŁ/ będziemy nazywali parametrami bezkontekstowymi i zbiór tych parametrów oznaczali przez CFPAR, a jego elementy przez par, par$_1$, par$_2$, ...

Jeśli /at$_n$:at$_{n-1}$:...:at$_1$:val$_1$,val$_n$/ jest perceptem to val$_1$ jest wartością atrybutu at$_1$ a val$_n$ wartością atrybutu at$_n$. Z powyzszego wynika, że atrybut at$_1$ wyznacza typ kontekstu a atrybut at$_n$ typ wartości parametru. Z określenia perceptu wynika, że jeśli najprostszy konkret ma postać at$_1$:val$_1$ /z jednym tylko atrybutem/, to najprostszy parametr perceptu ma postać at$_2$:at$_1$:val$_1$, tzn. zawiera conajmniej dwa atrybuty. Percepty postaci /at:at:val,val/ wyrażają fakt, że val jest wartością atrybutu at lub, że "val jest at-em". Np./KOCIOŁ:KOCIOŁ:K1,K1/- "K1 jest kotłem", /KOLOR:KOLOR:CZERWONY,CZERWONY/-"czerwony jest kolorem".

Fakty dotyczące kotła K1, który jest koloru czerwonego, w kształcie walca, o objętości 2m^3 i zawierający parę o ciśnieniu

i temperaturze 2kPa i 130°C odpowiednio, można wyrazić przez następujące percepty:

/KOLOR:KOCIOŁ:K1,CZERWONY/

/KSZTAŁT:KOCIOŁ:K1,WALEC/

/OBJĘTOŚĆ:KOCIOŁ:K1,2m^3/

/TEMPERATURA:PARA:KOCIOŁ:K1,130°C/

/CIŚNIENIE:PARA:KOCIOŁ:K1,2kPa/

/OBJĘTOŚĆ:PARA:KOCIOŁ:K1,2m^3/

Pierwsze trzy percepty dotyczą konkretu KOCIOŁ:K1 natomiast pozostałe trzy konkretu PARA:KOCIOŁ:K1.

Przyglądając się słoniowi Trąbalskiemu możemy fakty o nim wyrazić np. przez następujące percepty:

/KOLOR:LEWE-OKO:SŁOŃ:TRĄBALSKI,CIEMNO-SZARY/

/KOLOR:PRAWE-OKO:SŁOŃ:TRĄBALSKI,CZARNY/

/ILOŚĆ:NOGI:SŁOŃ:TRĄBALSKI,CZTERY/

/SSAK:SŁOŃ:TRĄBALSKI,TRĄBALSKI/.

Ostatni percept wyraża fakt, że słoń Trąbalski jest również ssakiem.

Niech pe_i=/at_i:at_{i-1}:...:at_1:val_1,val_i/,

pe_j=/at_j:at_{j-1}:...:at_1:val_1,val_j/

będą perceptami i c konkretem relacji percepcji per. Zbiór perceptów relacji per, jej przeciwdziedzinę, oznaczamy przez PE, PE=/DNT/per. Przez $\approx_c$ oznaczamy relację równoważności w zbiorze perceptów PE, taką że $pe_i \approx_c pe_j$ wtedy i tylko wtedy, gdy dla konkretu c spełnione są następujące warunki:

pe_i=/at_i:at_{i-1}:...:c,val_i/,

pe_j=/at_j:at_{j-1}:...c,val_j/.

Niech PE/c oznacza zbiór wszystkich perceptów $pe \in PE$, które

pozostają w relacji $\approx_c$. Zbiór PE/c nazywamy znaczeniem konkretu c. Zbiór konkretów CON relacji percepcji per wyznacza rodzinę podzbiorów perceptów $PE/CON = \{PE/c : c \in CON\}$.

Utożsamiając rodzinę PE/CON ze zbiorem istnień ENT, możemy istnienia potraktować jako wtórność konkretów CON, które wyznaczają również zbiór perceptów PE.

Mając na uwadze powyższe, w dalszych rozważaniach ograniczymy się wyłącznie do zbioru perceptów PE, jednocześnie eliminując z dalszych rozważań istnienia i wszelkie kwestie z nimi związane.

W dalszym ciągu określimy formalnie język służący do wyrażania faktów elementarnych /perceptów/ jak i faktów złożonych utworzonych z perceptów. Język ten będziemy nazywali językiem perceptowym. Zdefiniujemy aparat logiczny tego języka uzyskując teorię sformalizowaną nazwaną rachunkiem perceptowym.

Rozwarzania nasze, na temat faktów w stosunku do procesora i rzeczywistości, ustaliły uwagę na perceptach - obiektach pośredniego poziomu, poziomu wyobrażenia rzeczywistości.

W terminach perceptów rozumieć będziemy interprety lingwistyczne czy graficzne, jak również interprety logiczne. Wcześniej stwierdziliśmy, że struktura interpretów lingwistycznych, czy graficznych jest na tyle złożona i niejednoznaczna, że analiza ich i poszukiwanie właściwych struktur dla obiektów procesora, jest zbyt zawiłe. Z tego powodu proponujemy reprezentację interpretów logicznych, wyrażających poziom wiedzy np. interpretów lingwistycznych, jako wyrażeń o jednoznacznej składni i semantyce.

Zakładamy, że rozumiejąc w terminach perceptów, z jednej strony, obiekty procesora poprzez interprety logiczne i interprety lingwistyczne z drugiej strony, będziemy w stanie zrozumieć, w terminach perceptów, bezpośrednią transformację z interpretów lingwistycznych na obiekty procesora /rys.1.3./.

2. Rachunek perceptowy

2.1. Składnia, semantyka i aparat logiczny

Zbiorem symboli języka PL jest suma następujących niepustych i parami rozłącznych zbiorów:

NF - zbiór zero i jednoargumentowych funktorów nazwotwórczych, $NF=NF_0 \cup NF_1$. Elementy zbioru NF_0 /funktory zeroargumentowe/ oznaczamy przez $v_1, v_2, \ldots$ natomist elementy zbioru NF_1 /funktory jednoargumentowe/ oznaczamy przez $a_1, a_2, \ldots$. Dwie wyróżnione stałe $\perp, \top$ należą do NF_0. Zbiory NF_0 i NF_1 są rozłączne, $NF_0 = \bigcup_{a \in NF_1} NF_a \cup \{\perp, \top\}$.

VAR - zbiór zmiennych oznaczonych przez $x, y, x_1, y_1, \ldots$, $VAR = \bigcup_{a \in NF_1} VAR_a$.

$\{\neg, \vee, \wedge, \Rightarrow, \Leftrightarrow\}$ - zbiór spójników zewnętrznych: negacji, alternatywy, koniunkcji, implikacji i równoważności odpowiednio;

$\{-, +, \cdot, \rightarrow, \leftrightarrow\}$ - zbiór spójników wewnętrznych;

$\{\exists, \forall\}$ - zbiór kwantyfikatorów nazywanych szczegółowym i ogólnym odpowiednio;

$\{:, (,)\}$ - zbiór znaków pomocniczych.

Elementy zbioru NF_0 nazywamy słowami. Będziemy posługiwali się również wyrażeniami utworzonymi ze słów, zmiennych i spójników wewnętrznych zgodnie z poniższą definicją.

Zbiór TR termów jest najmniejszym zbiorem spełniającym, dla każdego $a \in NF_1$, poniższych pięć warunków:

(T1) jeśli $x \in VAR_a$, to $x \in TR_a$

(T2) $\perp, \top \in TR_a$

(T3) jeśli $v \in NF_a$, to $v \in TR_a$

(T4) jeśli $t_1, t_2 \in TR_a$, to $-t_1, t_1 + t_2, t_1 \cdot t_2, t_1 \rightarrow t_2$, $t_1 \leftrightarrow t_2 \in TR_a$

(T5) jeśli $t \in TR_a$, to $t \in TR$

Elementy zbioru TR oznaczać będziemy przez $t, t_1, t_2, \ldots$.

Zbiór deskryptorów DES utworzony z symboli języka PL jest najmniejszym zbiorem D spełniającym następujące warunki:

/D1/ jeśli $x \in VAR_a$, $v \in NF_a$, $a \in NF_1$ to $a:x, a:v \in D$

/D2/ jeśli $d \in D$ i $a \in NF_1$ to $a:d \in D$

Zbiór deskryptorów bez zmiennych będziemy oznaczali przez DES_o, tj. dodając indeks "o".

Zbiór formuł FORPL utworzony z symboli języka PL jest najmniejszym zbiorem F spełniającym następujące warunki:

/F1/ jeśli $a:d \in DES$ i $t \in TR_a$, to $/a:d, t/ \in F$

/F2/ jeśli $A \in F$, to $\neg A \in F$

/F3/ jeśli $A, B \in F$, to $A \vee B$, $A \wedge B$, $A \Rightarrow B$, $A \Leftrightarrow B \in F$

/F4/ jeśli $A \in F$, $x \in VAR$, to $\exists xA$, $\forall xA \in F$.

Formuły określone przez F1 nazywamy elementarnymi i oznaczamy przez $FORPL_e$. Przyjmujemy zwykłą definicję zmiennych wolnych i związanych. Zmienna w formule jest związana, jeśli znajduje się w zasięgu kwantyfikatora z tą samą nazwą zmiennej, w przeciwnym wypadku występuje jako zmienna wolna. Formuła bez zmiennych wolnych nazywana jest zdaniem. Zdania zbioru FORPL oznaczamy przez $FORPL_o$. Elementy zbioru FORPL oznaczać będziemy przez A, B, C, w miarę potrzeby z indeksami.

Ażeby uniknąć zbytniego nagromadzenia nawiasów, przyjmujemy zwykle stosowaną umowę, że kwantyfikatory wiążą mocniej niż

spójniki zewnętrzne i będziemy opuszczać nawiasy zawsze wtedy, gdy to nie spowoduje niejasności przy odczytywaniu wyrażeń, w których występują kwantyfikatory.

Najprostsze formuły elementarne /ze zbioru $FORPL_{eo}$/ są parą /d,v/, gdzie d jest deskryptorem i v słowem.

Zatem semantyka języka powinna być zdefiniowana przez użycie uniwersum dwusortowego i funkcję oznaczającą m, która każdemu funktorowi nazwotwórczemu języka przypisuje element uniwersum.

Niech PAR i VAL będą niepustymi zbiorami nazywanymi parametrami i wartościami odpowiednio. Zbiór parametrów PAR jest skończony. AT jest zbiorem atrybutów występujących w parametrach zbioru PAR. Dla każdego atrybutu at $\in$ AT niech VAL_{at} będzie zbiorem wartości atrybutu at. Dwie wyróżnione wartości $\perp$ /nieokreśloność - wartość nieokreślona/ i $\top$ /nadokreśloność - jakakolwiek wartość określona/ należą do VAL, $VAL=\bigcup_{at\in AT} VAL_{at} \cup \{\perp,\top\}$. Parę U=/PAR,VAL/ nazywamy uniwersum. Odwzorowanie $m:NF\to VAL \cup AT$ takie, że $m/\perp\!\!\!\perp/=\perp$, $m/\top/=\top$, $m/a/\in AT$, dla każdego $a\in NF_1$ i jeśli $v\in NF_a$, to $m/v/\in VAL_{m/a/}$, nazywamy funkcją oznaczającą w uniwersum U. Funkcję m rozszerzamy na zbiór deskryptorów DES_o w sposób naturalny, tj.

$$m/a_n:a_{n-1}:\ldots:v/=m/a_n/:m/a_{n-1}/\ldots:m/v/.$$

Funkcję $w:VAR\to VAL$ spełniającą warunek, jeśli $x\in VAR_a$, $a\in NF_1$, to $w/x/\in VAL_{m/a/}$, nazywamy wartościowaniem języka PL. Przez realizację języka PL rozumiemy system R=/U,m,PE/, gdzie U jest uniwersum, m jest funkcją oznaczającą w U i PE jest zbiorem perceptów utworzonych z parametrów i wartości uniwersum U.

Niech R=/U,m,PE/ będzie realizacją, w wartościowaniem w U. Powiemy, że formuła A jest spełniona w realizacji R przez wartościowanie w /R, w sat A/ jeśli zachodzą następujące warunki:

(S1) R, w sat/d,v/ iff /m/d/,m/v//$\in$ PE, d $\in DES_o$, v $\in NF_o$

(S2) R, w sat/d,x/ iff /m/w/d//,w/x//$\in$ PE, d$\in$ DES

(S3) R, w sat/d,-t/ iff R, w sat/$\neg$/d,t//

(S4) R, w sat/d,t_i+t_j/ iff R, w sat//d,t_i/$\vee$ /d,t_j//

(S5) R, w sat/d,t_i·t_j/ iff R, w sat//d,t_i/$\wedge$ /d,t_j//

(S6) R, w sat/d,$t_i\rightarrow t_j$/ iff R, w sat//d,t_i/$\Rightarrow$/d,t_j//

(S7) R, w sat/d,$t_i\leftrightarrow t_j$/ iff R, w sat//d,t_i/$\Leftrightarrow$/d,t_j//

(S8) R, w sat $\neg$A iff nie R, w sat A

(S9) R, w sat A$\vee$B iff R, w sat A lub R, w sat B

(S10) R, w sat A$\wedge$B iff R, w sat A i R, w sat B

(S11) R, w sat A$\Rightarrow$B iff nie R, w sat A lub R, w sat B

(S12) R, w sat A$\Leftrightarrow$B iff R, w sat A$\Rightarrow$B i R, w sat B$\Rightarrow$A

(S13) R, w sat $\exists$ xA iff jeśli x$\in$ VAR_a, a$\in$ NF_1, to istnieje val$\in$ $VAL_{m/a/}$ takie, że R, w_{val} sat A

(S14) R, w sat $\forall$ xA iff jeśli x$\in$ VAR_a, a$\in$ NF_1, to dla wszystkich val$\in$ $VAL_{m/a/}$ mamy R, w_{val} sat A

gdzie napis iff oznacza zwrot "wtedy i tylko wtedy, gdy"; m/w/d//=m/d/ dla d$\in$ DES_{eo} i m/w/d//=m/a_n/:m/a_{n-1}/:...:w/x/ dla d=a_n:a_{n-1}:...:a_1:x; w_{val} jest wartościowaniem takim, że w_{val}/x/=val i w_{val}/y/=w/y/ dla y$\neq$x.

Ponadto przyjmujemy następujące definicje spójników wewnętrznych -, +, ·, $\rightarrow$, $\leftrightarrow$, określając je przez spójniki zewnętrzne $\neg$, $\vee$, $\wedge$, $\Rightarrow$, $\Leftrightarrow$:

(CD1) $\neg$/d,t_n/=/d,-t_n/

(CD2) /d,t_i+t_j/=/d,t_i/$\vee$ /d,t_j/

(CD3) /d,t_i·t_j/=/d,t_i/$\wedge$ /d,t_j/

(CD4) /d,$t_i\rightarrow t_j$/=/d,t_i/$\Rightarrow$ /d,t_j/

(CD5) /d,$t_i\leftrightarrow t_j$/=/d,t_i/$\Leftrightarrow$ /d,t_j/

Formuła A jest <u>spełnialna</u> w realizacji R=/U,m,PE/, gdy istnieje wartościowanie w_* dla którego zachodzi R, w sat A. Natomiast, gdy

istnieje takie wartościowanie w, dla którego A nie jest spełniona, to formuła A jest falsykowalna. Formuła A jest prawdziwa w realizacji R $/\models_R A/$ wtedy i tylko wtedy, gdy dla każdego wartościowania w formuła A jest spełnialna. W takim przypadku powiemy, że realizacja R jest modelem dla formuły A. Z iór formuł FS języka PL jest prawdziwy w realizacji R, jeśli każda formuła A ze zbioru FS jest prawdziwa w realizacji R. W tej sytuacji powiemy, że realizacja R jest modelem dla zbioru formuł FS.
Formuła A prawdziwa w każdej realizacji R jest tautologią.
Formuły A i B są równoważne $/A \equiv B/$, jeśli w każdej realizacji R są jednocześnie spełnialne lub falsykowalne. Z przyjętych definicji wynika, że formuły A i B są równoważne wtedy i tylko wtedy, gdy formuła $A \Leftrightarrow B$ jest tautologią.

Wśród wszystkich formuł FORPL języka perceptowego PL wyróżniamy te, które traktujemy jako odpowiedniki aksjomatów rachunku perceptowego. Nazywamy je aksjomatami perceptowymi. Za aksjomaty perceptowe PA przyjmujemy następujące formuły:

(PA1) $\forall x(a:a:x,x)$

(PA2) $(a_n:a_{n-1}:\ldots:a_1:v,\top) \Leftrightarrow \forall x(a_n:a_{n-1}:\ldots:a_1:v,x)$

(PA3) $(a_n:a_{n-1}:\ldots:a_1:\top,v) \Leftrightarrow \forall x(a_n:a_{n-1}:\ldots:a_1:x,v)$

(PA4) $(a_n:a_{n-1}:\ldots:a_1:v,\bot) \Leftrightarrow \neg\exists x(a_n:a_{n-1}:\ldots:a_1:v,x)$

(PA5) $(a_n:a_{n-1}:\ldots:a_1:\bot,v) \Leftrightarrow \neg\exists x(a_n:a_{n-1}:\ldots:a_1:x,v)$

(PA6) $\forall x_1 \forall x_n \forall x_i$
$(((a_i:a_{i-1}:\ldots:a_1:x_1,x_i) \wedge (a_n:a_{n-1}:\ldots:a_i:x_i,x_n)) \Rightarrow$
$(a_n:a_{n-1}:\ldots:a_i:a_{i-1}:\ldots:a_1:x_1,x_n))$, dla $n \geqslant i \geqslant 2$

(PA7) $\forall x_1 \forall x_n((a_n:a_{n-1}:\ldots:a_1:x_1,x_n) \Leftrightarrow$
$\exists x_i ((a_i:a_{i-1}:\ldots:a_1:x_1,x_i) \wedge (a_n:a_{n-1}:\ldots:a_i:x_i,x_n)))$,
dla $n \geqslant i \geqslant 2$

(PA8) $\forall x_n \forall x_{n+k} \forall x_1$
$(((a_n:a_{n-1}:\ldots:a_1:x_1, x_n) \wedge$
$(a_{n+k}:a_{n+k-1}:\ldots:a_n:a_{n-1}:\ldots:a_1:x_1, x_{n+k})) \Rightarrow$
$(a_{n+k}:a_{n+k-1}:\ldots:a_n:x_n, x_{n+k}))$, dla $n \geqslant 2$, $k \geqslant 0$

(PA9) $\forall x_1 \forall x_k \forall x_n$
$(((a_n:a_{n-1}:\ldots:a_k:x_k, x_n) \wedge (a_n:a_{n-1}:\ldots:a_1:x_1, x_n)) \Rightarrow$
$(a_k:a_{k-1}:\ldots a_1:x_1, x_k))$, dla $n \geqslant k \geqslant 2$

(PA10) $\forall x ((a_1:a_1:x,x) \Rightarrow (a_2:a_2:x,x)) \Leftrightarrow \forall x (a_2:a_1:x,x)$

gdzie $v \in NF_0, a, a_i \in NF_1$, $i=1,2,\ldots,n$, $x, x_1, x_n, x_k, x_{n+k} \in VAR$.

Aksjomat PA1 wyznacza zbiór /sort/ NF_a, $a \in NF_1$; aksjomaty PA2 - PA5 określają nadokreśloność i nieokreśloność; aksjomat PA6 określa łączenie deskryptorów, aksjomat PA7 określa rozdzielenie deskryptora, aksjomat PA8 i PA9 określają odcinanie deskryptora, aksjomat PA10 określa zawieranie się /jednoargumentowych/ funktorów nazwotwórczych.

Następnym etapem formalizacji rachunku perceptowego jest wybór aksjomatów logicznych LA i reguł dowodzenia, które łącznie stanowią jego aparat logiczny. Aksjomaty logiczne wybieramy spośród tych formuł, które są tautologiami.
Za aksjomaty logiczne przyjmujemy wszystkie formuły języka perceptowego PC, postaci

(LA1) $A \Rightarrow /B \Rightarrow A/$

(LA2) $/A \Rightarrow /B \Rightarrow C// \Rightarrow //A \Rightarrow B/ \Rightarrow /A \Rightarrow C//$

(LA3) $\neg A \Rightarrow /A \Rightarrow B/$

(LA4) $/\neg A \Rightarrow A/ \Rightarrow A$

(LA5) $\forall x/A \Rightarrow B/x// \Rightarrow /A \Rightarrow \forall x B/x//$

(LA6) $\forall x A/x/ \Rightarrow A/v/$, dla każdego $v \in NF_a$, gdy $x \in VAR_a$

gdzie A,B,C FORPL, przy czym w aksjomacie LA5 zakłada się, że w formule A nie występuje zmienna wolna x.

Za reguły dowodzenia przy takim układzie aksjomatów logicznych przyjmujemy regułę odrywania

$$(DR1) \qquad \frac{A, A \Rightarrow B}{B}$$

oraz regułę uogólniania

$$(DR2) \qquad \frac{A/x/}{\forall x A/x/}$$

Ponadto przyjmujemy definicje spójników $\vee$, $\wedge$, $\Leftrightarrow$ i kwantyfikatora szczegółowego $\exists$ określając je przez implikację, negację i kwantyfikator ogólny, w sposób następujący:

(LD1) $A \vee B = \neg A \Rightarrow B$

(LD2) $A \wedge B = \neg / A \Rightarrow \neg B /$

(LD3) $A \Leftrightarrow B = / A \Rightarrow B / \wedge / B \Rightarrow A /$

(LD4) $\exists x A = \neg \forall x \neg A$

Ustaliwszy zbiór aksjomatów logicznych i reguł dowodzenia precyzujemy intuicyjne pojęcie dowodu.

Dowodem formalnym, dla dowolnej formuły A ze zbioru formuł FS, nazywamy każdy skończony ciąg formuł rachunku perceptowego, taki, że każda z formuł tego ciągu jest bądź formułą ze zbioru FS, bądź aksjomatem perceptowym, bądź aksjomatem logicznym, bądź też powstaje z wcześniejszych formuł w tym ciągu przez stosowanie reguł dowodzenia, przy czym ostatnią formułą w tym ciągu jest formuła A. Powiemy, że formuła A jest wyprowadzalna /syntaktycznie/ ze zbioru formuł FS, jeśli istnieje dla niej dowód formalny /$FS \vdash A$/.

Obecnie sprecyzujemy operację wyprowadzania D, która jest formalizacją intuicyjnego pojęcia wyprowadzalności. Dla każdego zbioru formuł FS definiujemy D/FS/ jako zbiór wszystkich formuł wyprowadzalnych z FS, tj. $D/FS/ = \{A \in FORPL : FS \vdash A\}$.

Zbiór formuł FS jest sprzeczny wtedy i tylko wtedy, gdy istnieje taka formuła A, że $A \in D/FS/$ i $\neg A \in D/FS/$. Powiemy wówczas, że formuła A jest sprzeczna.

Gdy formuła taka nie istnieje, zbiór FS nazywamy niesprzecznym.

Jeśli zbiór formuł FS jest zbiorem pustym $\emptyset$, to elementy zbioru $D/\emptyset/$ nazywamy twierdzeniami rachunku perceptowego.

Napis $\vdash A$ czytamy: A jest twierdzeniem rachunku perceptowego.

Niech $R=/U,m,PE/$ będzie realizacją rachunku perceptowego. Realizację R nazywamy modelem rachunku perceptowego, jeśli realizacja R jest modelem dla zbioru aksjomatów perceptowych PA. Określimy obecnie klasę modeli rachunku perceptowego przez sprecyzowanie zbioru PE dowolnej realizacji R.

Niech AT będzie zbiorem atrybutów i VAL zbiorem wartości, $VAL=\bigcup_{at \in AT} VAL_{at} \cup \{\perp, \top\}$. Przez zbiór perceptów MPE oznaczać będziemy najmniejszy zbiór perceptów spełniających, dla $at_i \in AT$, $i=1,2,\ldots,n$ i $val, val_1, val_n, val_k, val_{n+k} \in VAL - \{\perp, \top\}$, poniższe warunki:

/M1/ $/at:at:val,val/ \in MPE$,

dla każdego $at \in AT$ i $val \in VAL_{at}$

/M2/ dla każdego $val_1 \in VAL_{at_1}$

$/at_n:at_{n-1}:\ldots:at_1:val_1,val_n/ \in MPE$

wtedy i tylko wtedy, gdy

$/at_n:at_{n-1}:\ldots:at_1:\top,val_n/ \in MPE$

/M3/ dla każdego $val_n \in VAL_{at_n}$

$/at_n:at_{n-1}:\ldots:at_1:val_1,val_n/ \in MPE$

wtedy i tylko wtedy, gdy

$/at_n:at_{n-1}:\ldots:at_1:val_1,\top/ \in MPE$

/M4/ dla każdego $val_1 \in VAL_{at_1}$

$/at_n:at_{n-1}:\ldots:at_1:val_1,val_n/ \notin MPE$

wtedy i tylko wtedy, gdy

$/at_n:at_{n-1}:\ldots:at_1:\perp, val_n/ \in MPE$

/M5/ dla każdego $val_n \in VAL_{at_n}$

$/at_n:at_{n-1}:\ldots:at_1:val_1, val_n/ \notin MPE$

wtedy i tylko wtedy, gdy

$/at_n:at_{n-1}:\ldots:at_1:val_1, \perp / \in MPE$

/M6/ jeśli $/at_i:at_{i-1}:\ldots:at_1:val_1, val_i/ \in MPE$

i $/at_n:at_{n-1}:\ldots:at_i:val_i, val_n/ \in MPE$, dla $n \geq i \geq 2$

to $/at_n:at_{n-1}:\ldots:at_i:at_{i-1}:\ldots:at_1:val_1, val_n/ \in MPE$

/M7/ jeśli $/at_n:at_{n-1}:\ldots:at_1:val_1, val_n/ \in MPE$, to istnieje $val_i \in VAL_{at_i}$, takie że

$/at_i:at_{i-1}:\ldots:at_1:val_1, val_i/ \in MPE$

i $/at_n:at_{n+1}:\ldots:at_i:val_i, val_n/ \in MPE$, dla $n \geq i \geq 2$

/M8/ jeśli $/at_n:at_{n-1}:\ldots:at_1:val_1, val_n/ \in MPE$

i $/a_{n+k}:a_{n+k-1}:\ldots:at_n:at_{n-1}:\ldots:at_1:val_1, val_{n+k}/ \in MPE$,

to $/at_{n+k}:at_{n+k-1}:\ldots:at_n:val_n, val_{n+k}/ \in MPE$, dla $n \geq 2$, $k \geq 0$

/M9/ jeśli $/at_n:at_{n-1}:\ldots:a_k:val_k, val_n/ \in MPE$

i $/at_n:at_{n-1}:\ldots:a_1:val_1, val_n/ \in MPE$,

to $/at_k:at_{k-1}:\ldots:at_1:val_1, val_k/ \in MPE$, dla $n \geq k \geq 2$

/M10/ jeśli dla każdego $val_1 \in VAL_{at_1}$, $/at_2:at_2:val_1, val_1/ \in MPE$

to $/at_2:at_1:val_1, val_1/ \in MPE$, dla każdego $val_1 \in VAL_{at_1}$.

Łatwo sprawdzić, że warunki określające zbiór MPE zostały tak, dobrane, że dowolna realizacja /U,m,MPE/ jest modelem dla zbioru aksjomatów perceptowych PA. Realizację /U,m,MPE/, której zbiór perceptów MPE spełnia warunki M1-M10 nazywać będziemy <u>realizacją perceptową</u> i oznaczać przez M. Wobec powyższego każda realizacja M jest modelem rachunku perceptowego. Formuła A jest <u>pewna</u> w rachunku perceptowym /$\models$ A/, jeśli jest prawdziwa w każdym modelu M. Formuła A <u>wynika</u> /semantycznie/ ze zbioru formuł FS /FS$\models$A/ wtedy i tylko wtedy, gdy każdy model M dla FS jest modelem dla A.

2.2. Podstawowe własności rachunku perceptowego

Jeśli z aksjomatów rachunku perceptowego usuniemy aksjomaty perceptowe PA, to operacja wyprowadzalności D ograniczona będzie wyłącznie do aksjomatów logicznych LA i stosowania dwóch przyjętych reguł dowodzenia DR1 i DR2. Aksjomaty logiczne LA i przyjęte reguły dowodzenia są podstawieniami aksjomatów i reguł rachunku predykatów pierwszego rzędu. Ponieważ każda formuła wyprowadzalna w rachunku predykatów jest tautologią, wobec powyższego, zbiór formuł wyprowadzalnych, z pominięciem aksjomatów perceptowych, będzie również zbiorem tautologii.

Z własności elementarnych teorii sformalizowanych [31] otrzymujemy stwierdzenie, że rachunek perceptowy, teoria PC=/PL,D/, gdzie PL jest językiem perceptowym i D operacją wyprowadzania, jest niesprzeczny wtedy i tylko wtedy, gdy istnieje realizacja perceptowa M dla PL.

Stwierdzenie powyższe określa warunki praktycznych zastosowań teorii perceptowych, ograniczając je do modeli teorii. Zatem, konstruując percepty określonej dziedziny zastosowań, należy mieć na uwadze warunki M1-M10 wyznaczające zbiór perceptów modelu.

Intuicje związane z tymi warunkami wynikają z roli perceptów w rozumieniu interpretów lingwistycznych z jednej strony i reprezentacji interpretów logicznych z drugiej strony.

Przedstawiony formalizm rachunku perceptowego stosować będziemy do wyrażania faktów, których opisy utożsamiać będziemy z pewnym zbiorem formuł FS. Rozwiązywane zadanie, w oparciu o opisy faktów FS, traktować będziemy jako cel, którego opisem będzie formuła G. Dane zadanie opisane przez G będzie miało rozwiązanie, jeśli FS⊨G. Zatem z punktu widzenia operacji wyprowadzania /czy procesora tą operację wykonującego/ istotne

(ST1) $\vdash G$ implikuje $\models G$

(ST2) $FS \vdash G$ implikuje $FS \models G$

i po drugie

(CT1) $\models G$ implikuje $\vdash G$

(CT2) $FS \models G$ implikuje $FS \vdash G$.

Powyższe cztery stwierdzenia są spełnione w rachunku perceptowym. Pierwsze dwa odnoszone są do twierdzenia o poprawności, ostatnie dwa do twierdzenia o pełności. Dowody tych twierdzeń można znaleźć w [30].

Następujące własności operacji wyprowadzania D są znanymi w teoriach sformalizowanych [31,33].

(DP1) $FS \subseteq D/FS/$

(DP2) $D/D/T//=D/T/$

(DP3) $FS_1 \subseteq FS_2$ implikuje $D/FS_1/ \subseteq D/FS_2/$

(DP4) $D/FS_1 \cup FS_2/=D/FS_1/ \cup D/FS_2/$

Poniższa lista wylicza inne użyteczne własności operacji D.

(DP5) $\vdash G$ implikuje $G \in D/FS/$, dla każdego $FS \subseteq FORPL$

(DP6) $F \in D/FS/$ i $/F \to G/ \in D/FS/$ implikuje $G \in D/FS/$

(DP7) $F \in D/FS/$ implikuje $\forall x F \in D/FS/$

(DP8) $\forall x F \in D/FS/$ implikuje $F/v/ \in D/FS/$, dla każdego $v \in NE_a$ jeśli $x \in VAR_a$

(DP9) $G \in D/FS \cup \{F\}/$ wtedy i tylko wtedy, gdy $/F \to G/ \in D/FS/$

(DP10) $G \in D/FS/$ wtedy i tylko wtedy, gdy $FS \cup \{\neg G\}$ jest sprzeczny.

2.3. Postać Skolema

Każde odwzorowanie $s:VAR \to TR$ takie, że $s/x/ \in TR_a$ dla każdego $x \in VAR_a$ oraz zbiór $\{x:s/x/\neq x\}$ jest skończony, nazywać będziemy podstawieniem.

Ponieważ zbiór zmiennych, którym podstawienie przypisuje różne od nich termy, jest zawsze skończony, możemy więc opisywać podstawienie podając explicite wykaz takich zmiennych wraz z przypisanymi im termami. Zatem $\{x_1/t_1, x_2/t_2, \ldots, x_n/t_n\}$ oznaczać będzie podstawienie, które zmiennej x_i przypisuje term t_i dla $i=1,2,\ldots,n$.

Każde podstawienie daje się w naturalny sposób przedłużyć na dowolne termy i formuły.

Niech $EXP=FORPL \cup TR$. Oznaczamy przez $e/x_1/t_1,\ldots,x_n/t_n/$ wyrażenie powstałe z $e \in EXP$ jako wynik zastępienia zmiennych $x_1, x_2, \ldots, x_n$ odpowiednio termami $t_1, t_2, \ldots, t_n$. Wówczas dla podstawienia $s=\{x_1/t_1, x_2/t_2, \ldots x_n/t_n\}$ przez $s/e/$ rozumieć będziemy warażenie $e/x_1/t_1', x_2/t_2', \ldots, x_n/t_n'/$, gdzie

$$t_i' = \begin{cases} x_i, & \text{gdy } x_i \text{ jest zmienną związaną} \\ t_i, & \text{w przeciwnym przypadku.} \end{cases}$$

Formułę $A \in FORPL$ nie zawierającą spójników $\Leftrightarrow$, $\Rightarrow$ i $\neg$ nazywamy formułą koniunkcyjno-alternatywną.

Formuła koniunkcyjno-alternatywna A jest postacią koniunkcyjno-alternatywną formuły B, jeśli $A \Leftrightarrow B$.

/CDF/ Dla każdego zdania $A \in FORPL$ istnieje efektywnie zdanie $B \in FORPL$ w postaci koniunkcyjno-alternatywnej.

Dowód - konstrukcja formuły B

1. Z formuły A eliminujemy spójniki $\Leftrightarrow$ i $\Rightarrow$ posługując się skończoną ilość razy prawami:

$A \Leftrightarrow B \equiv A \Rightarrow B \wedge B \Rightarrow A$ /LD3/

$A \Rightarrow B \equiv \neg A \vee B$ /LD1/

2. Wprowadzamy negacje bezpośrednio przed formuły elementarne posługując się skończoną ilość razy prawami de Morgana.

3. Wprowadzamy negacje do wewnątrz formuł elementarnych posługując się skończoną ilość razy prawem

$\neg /d,t/ \equiv /d,-t/$ /CD1/

Otrzymana w ten sposób formuła wynikowa spełnia tezę twierdzenia, co jednocześnie kończy dowód.

Powiemy, że formuła $A \in$ FORPL nie zawierającą kwantyfikatorów jest formułą <u>otwartą</u> lub w <u>postaci Skolema</u>.

Formułę A zawierającą wszystkie kwantyfikatory /prefiks/ na początku, której pozostała część /matryca/ jest formułą otwartą, nazywamy <u>formułą preneksową.</u> Prefiks i matrycę formuły A oznaczamy przez p/A/ i m/A/ odpowiednio.

Formuła preneksowa A jest <u>postacią preneksową</u> formuły B, jeśli $A \Leftrightarrow B$.

/PCDF/ Dla każdego zdania $A \in FORPL_0$ istnieje postać preneksowa $B \in FORPL_0$, taka że m/B/ jest formułą koniunkcyjno-alternatywną.

Dowód - konstrukcja

1. Tworzymy postać koniunkcyjno-alternatywną zgodnie z konstrukcją podaną wcześniej.

2. Niech Θ oznacza dowolny kwantyfikator, natomiast ε - jeden z symboli $\exists$, $\forall$. Obecnie wydobywamy kwantyfikatory na początek formuły posługując się skończoną ilość razy prawami:

a/ $\Theta x A/x/ \varepsilon\ \Theta y B/y/ \equiv \Theta x\, \Theta y /A/x/ \varepsilon\ B/Y//$

Zwróćmy uwagę, że A i B są zdaniami wobec czego x jako zmienna wolna nie może występować w B podobnie jak y nie może w A. Jeśli x występuje w B lub y w A to tylko jako zmienne związane. Taka sytuacja podpada zawsze pod jeden z następujących schematów:

b/ $\forall x A/x/ \wedge \forall x B/x/ \equiv \forall x /A/x/ \wedge B/x//$

c/ $\exists x A/x/ \vee \exists x B/x/ \equiv \exists x/A/x/ \vee B/x//$

d/ $\forall x A/x/ \vee \Theta x B/x/ \equiv \forall x \Theta z/A/x/ \wedge \{x/z\} /B//$

e/ $\exists x A/x/ \wedge \Theta x B/x/ \equiv \exists x \Theta z/A/x/ \vee \{x/z\} /B//$

gdzie z jest nową zmienną nie występującą w A ani w B i taką, że jeśli $x \in VAR_a$ to $z \in VAR_a$.

f/ $\Theta x A/x/ \varepsilon B \equiv \Theta x/A/x/ \varepsilon B/$ jeśli x nie występuje w B.

Otrzymana w ten sposób formuła wynikowa spełnia tezę twierdzenia, co kończy jednocześnie jego dowód.

Powiemy, że formuła $A \in FORPL$ ma <u>postać normalną koniunkcyjno-alternatywną</u> wtedy i tylko wtedy, gdy A przedstawia się jako

$$/A_1^1 \vee \ldots \vee A_{m1}^1/ \wedge \ldots \vee /A_1^k \vee \ldots \vee A_{mk}^k/$$

i wszystkie A_j^i są formułami elementarnymi.

/PNCDF/ Dla każdego zdania $A \in FORPL_o$ istnieje postać preneksowa $B \in FORPL_o$, taka że m/B/ ma postać normalną koniunkcyjno-alternatywną.

Dowód - konstrukcja

1. Tworzymy preneksową postać koniunkcyjno-alternatywną zgodnie z konstrukcją PCDF.

2. Matrycę doprowadzamy do koniunkcji alternatyw formuł elementarnych stosując skończoną ilość razy prawo rozdzielności

$$/A \wedge B/ \vee C \equiv /A \vee C/ \wedge /B \vee C/$$

Otrzymana w ten sposób formuła wynikowa spełnia tezę twierdzenia.

Formułę elementarną postaci $/a_2:a_1:t_1,t_2/$, $t_1 \in VAL_{a_1} \cup VAR_{a_1}$, $t_2 \in TR_{a_2}$, nazywamy <u>tetradą</u>.

Powiemy, że formuła A ma postać <u>tetradowo-normalną koniunkcyjno-alternatywną</u> wtedy i tylko wtedy, gdy A przedstawia się jak jako

$$/A_1^1 \vee \ldots \vee A_{m1}^1/ \wedge \ldots \wedge /A_1^k \vee \ldots \vee A_{mk}^k/$$

i wszystkie A_j^i są tetradami.

/TCDF/ Dla każdego zdania $A \in FORPL_0$ istnieje postać preneksowa $B \in FORPL_0$, taka że m/B/ ma postać tetradowo-normalną koniunkcyjno-alternatywną.

Dowód - konstrukcja

1. Z formuły A eliminujemy wszystkie spójniki wewnętrzne $-, +, \cdot, \rightarrow, \leftrightarrow$ posługując się definicjami CD1-CD5.

2. Na mocy aksjomatu perceptowego PA7, każdą formułę elementarną postaci $/a_n:a_{n-1}:\ldots:a_1:t_1,t_2/$ zastępujemy równoważną formułą

$$\exists x_2 \exists x_3 \ldots \exists x_{n-1} //a_2:a_1:t_1,x_2/ \wedge /a_3:a_2:x_2,x_3/ \wedge \ldots \wedge /a_n:a_{n-1}:x_{n-1},t_n//$$

3. Dalej postępujemy zgodnie z konstrukcją twierdzenia PNCDF.

Otrzymana w ten sposób formuła wynikowa spełnia tezę twierdzenia, co jednocześnie kończy jego dowód.

Formuły A i B nazwiemy <u>modelowo równoważnymi</u>, gdy prawdziwe jest następujące zdanie:

"istnieje model dla A wtedy i tylko wtedy, gdy istnieje model dla B".

/SKL/ Dla każdego zdania $A \in FORPL_0$ istnieje efektywnie formuła $B \in FORPL$ w postaci Skolema modelowo równoważna formule A.

Dowód - konstrukcja

1. Tworzymy dla zdania A postać preneksową C.

2. Kładziemy $B_0=C$, k=0 i przechodzimy do punktu 3.

3. Jeśli pośród kwantyfikatorów formuły B_k nie występuje "$\exists$" udajemy się do 7. W przeciwnym przypadku - niech Q_{ik} będzie pierwszym od lewej kwantyfikatorem szczegółowym formuły B_k. Znajdujemy wówczas zbiór $V_k = \{ x_1^k, x_2^k, \ldots, x_{jk}^k \}$ zmiennych występujących za kwantyfikatorami ogólnymi poprzedzającymi Θ_{ik} i przechodzimy do 4.

4. Dodajemy do alfabetu języka PL nowy, nie występujący symbol o_k j_k-argumentowego funktora /Skolema/ $o_k \in FF_a^{jk}$, dla $a \in NF_1$, jeśli $x_{ik} \in VAR_a$. Kładziemy $t_k = o_k/x_1^k, x_2^k, \ldots, x_{jk}^k/$ oraz dodajemy t_k do zbioru termów FTR_a^{jk}, po czym przechodzimy do 5.

5. Kładziemy $B_{k+1} = \{x_{ik}/t_k\}$ $/B_k/$ i usuwamy z B_{k+1} tekst "$\exists x_{ik}$".

6. Kładziemy k=k+1 i udajemy się do punktu 2.

7. Podstawiamy $B=B_k$, usuwamy z formuły B prefiks i kończymy proces.

Dowód, że formuły A i B są modelowo równoważne wynika z dowodu podanego w [32] dla tzw. normalnej postaci skolema.

Wyrażenia t_k tworzone w punkcie 4 nazywać będziemy <u>termami funkcyjnymi</u> /Skolema/. Zwróćmy uwagę, że termy funkcyjne nie występują w składni rachunku perceptowego i obecnie pojawiły się jako wyrażenia techniczne związane z postacią Skolema. Aby je formalnie potraktować w rachunku perceptowym, należy przyjąć poniższe definicje.

Dodajemy do alfabetu języka perceptowego PL zbiór <u>funkcyjnych funktorów nazwotwórczych</u> $FF = \bigcup_{a \in NF_1} \bigcup_{i=0}^{\infty} FF_a^i$, gdzie FF_a^i jest zbiorem i-argumentowych funktorów oznaczonych przez o, w miarę potrzeby z indeksami.

Określimy zbiór <u>termów funkcyjnych</u> FTR w sposób następujący:

a/ jeśli $o \in FF_a^i$, to $o/x_1, x_2, \ldots, x_i/ \in FTR_a^i$

b/ jeśli $ov \in FTR_a^i$, to $ov \in FTR$.

Zbiór $FTR = \bigcup_{a \in AT_1} \bigcup_{i=0}^{\infty} FTR_a^i$ traktować będziemy jako podzbiór NF_o, w ten sposób, że $FTR_a \subseteq NF_a$, $FTR_a = \bigcup_{i=0}^{\infty} FTR_a^i$.

Uniwersum systemu perceptowego rozszerzamy o zbiór funkcji /operacji/ $OP = \bigcup_{at \in AT} \bigcup_{i=0}^{\infty} OP_{at}^i$, takich że dla każdego i=1,2,..., $f \in OP_{at}^i$ jest i-argumentową funkcją $f : VAL^i \to VAL_{at}$.

Funkcję oznaczającą m w uniwersum U=/PAR,VAL,OP/ rozszerzamy na zbiór funkcyjnych funktorów nazwotwórczych w sposób następujący:

dla każdego $o \in FF_a^k$ $m/o/ \in OP^k_{m/a/}$, dla $a \in NF_1$, k=1,2,... .

/DSKL/ Przez dualną postać Skolema formuły A rozumieć będziemy formułę B będącą wynikiem podanej konstrukcji SKL, w której punkt 3 zastępujemy poniższym punktem 3´:

3´. Jeśli pośród kwantyfikatorów formuły B_k nie występuje "$\forall$" udajemy się do 7. W przeciwnym przypadku - niech Q_{ik} będzie pierwszym od lewej kwantyfikatorem ogólnym formuły B_k. Znajdujemy wówczas zbiór $V_k = \{x_1^k, x_2^k, \ldots, x_{jk}^k\}$ zmiennych występujących za kwantyfikatorami szczegółowymi poprzedzającymi Q_{ik} i przechodzimy do 4.

Procesy opisane przez dowody twierdzeń SKL i DSKL nazywamy skolemizacją /skl/ i dualną skolemizację /dskl/ odpowiednio. Gdy w punkcie 1 konstrukcji tych procesów dla zdania A tworzymy koniunkcyjno-alternatywną, normalną koniunkcyjno-alternatywną lub tetradową koniunkcyjno-alternatywną postać preneksową C /zgodnie z podanymi konstrukcjami/, to mówimy o koniunkcyjno-alternatywnej, normalnej koniunkcyjno-alternatywnej lub tetradowej koniunkcyjno-alternatywnej skolemizacji /dualnej skolemizacji/ odpowiednio.

Wynikiem koniunkcyjno-alternatywnej, normalnej koniunkcyjno-alternatywnej, tetradowej koniunkcyjno-alternatywnej skolemizacji /dualnej skolemizacji/ jest formuła w koniunkcyjno-alternatywnej, normalnej koniunkcyjno-alternatywnej, tetradowej koniunkcyjno-alternatywnej postaci Skolema /dualnej postaci Skolema/ odpowiednio.

3. Systemy perceptowe

3.1. Określenie systemu perceptowego

Formuły języka PL można potraktować jako schematy zdań, które wyrażają wiedzę o konkretach modelu M=/U,m,MPE/, gdzie U=/PAR,VAL/ jest uniwersum, m funkcją oznaczającą taką, że $m/NF_0/=VAL$, $m/NF_1/=AT$ i MPE jest zbiorem perceptów.

Określimy zbiór wyrażeń FORPL/U/ otrzymanych z formuł języka PL przez podstawienie za funktory nazwotwórcze NF wartości funkcji m dla tych funktorów. Wyrażenia zbioru FORPL/U/ nazywać będziemy U-formułami, a zbioru $FORPL_e/U/$ U-perceptami. U- percepty z dwuatrybutowym parametrem nazywać będziemy U-tetradami. U-formuły bez zmiennych wolnych, ze zbioru $FORPL_0/U/$, nazywamy U-zdaniami. U-zdania wyrażają wiedzę o faktach lub celach dotyczących konkretów modelu M. TR/U/ oznacza zbiór U-termów otrzymany w wyniku podstawienia za funktory zeroargumentowe wartości funkcji m dla tych funktorów w termach ze zbioru TR. Elementy zbioru VAR_{at} traktujemy jako zmienne przyjmujące wartości ze zbioru VAL_{at}, $VAR=\bigcup_{at\in AT} VAR_{at}$.

Pojęcie wyprowadzalności rozszerzamy w sposób naturalny na U-formuły. Powiemy, że U-formuła jest wyprowadzalna ze zbioru U-formuł FS jeśli może być otrzymana z U-formuł mających postać aksjomatów /perceptowych lub logicznych/ rachunku perceptowego PC lub elementów zbioru FS przez kolejne stosowanie reguł dowodzenia rachunku PC. Używać będziemy taką samą notację jak dla formuł FORPL, mianowicie zapiszemy $FS\vdash A$ jeśli U-formuła A jest wyprowadzalna ze zbioru FS U-formuł.

Interprety logiczne wyrażone przez U-formuły nazywać będziemy interpretami perceptowymi.

Niech U=/PAR,VAL/ będzie uniwersum, FS i GS zbiorami U-zdań, FS jest zbiorem niepustym. Zakładamy, że PAR jest skończonym

niepustym zbiorem parametrów i VAL jest niepustym zbiorem wartości.

Systemem perceptowym w uniwersum U nazwiemy trójkę

$$S=/U,FS,GS/$$

składającą się z uniwersum U, niepustego skończonego zbioru FS U-zdań wyrażających fakty o konkretach parametrów PAR i skończonego zbioru GS U-zdań wyrażających cele /pytania/ dotyczące konkretów parametrów PAR.

Zauważmy, że zgodnie z definicją perceptu, jeśli zbiór VAL określimy przez podanie jego podzbiorów VAL_{at}, $at \in AT$, to w celu określenia zbioru PAR, wystarczy podać zbiór parametrów bezkontekstowych CFPAR. Sort kontekstu perceptu $/at_n:at_{n-1}:\ldots:at_1:val_1,val_n/$, tj. zbiór VAL_{at_1} jest określony przez pierwszy atrybut at_1 danego parametru. A więc w miejsce /PAR,VAL/ możemy rozpatrywać

$/CFPAR,\ VAL_{at_1},VAL_{at_2},\ldots,VAL_{at_n}\ /$, gdzie $\{at_1,at_2,\ldots,at_n\}=AT$.

U-zdania będące opisami faktów nazywać będziemy U-faktami, a U-formuły opisujące cele U-celami.

Poniższy przykład wyjaśnia jak U-fakty zbioru FS /nazywanego ciałem wiedzy faktów/ i U-cele zbioru GS /nazywanego ciałem wiedzy celów/ mogą być wyrażone w systemie perceptowym.

Rozpatrzmy fakty wyrażone następującym interpretem lingwistycznym:

Pies As szczeka.

Zwierzę Mruczek miauczy.

Jeśli pies merda ogonem, to jest przyjazny.

Jeśli pies szczeka na kota, to kot obawia się psa.

Pies jest zwierzęciem.

Jeśli zwierzę miauczy, to jest kotem.

i cel wyrażony poniższym interpretem lingwistycznym:

Czy istnieją kot i pies, takie że kot obawia się psa?

Podzbiór parametrów bezkontekstowych, podzbiór atrybutów, podzbiory wartości atrybutów i podzbiory zmiennych dla powyższych interpretów lingwistycznych, mogą być określone w sposób następujący:

$\{$PIES:PIES, KOT:KOT, ZWIERZĘ:ZWIERZĘ,
ZACHOWANIE:PIES, ZACHOWANIE:ZWIERZĘ, USPOSOBIENIE:PIES,
PIES:KOGO-CZEGO:OBAWIANIE-SIĘ:KTO-CO:KOT, PIES:ZWIERZĘ$\} \subseteq CFPAR_1$

$\{$PIES, KOT, ZWIERZĘ, ZACHOWANIE, USPOSOBIENIE$\} \subseteq AT_1$

$\{AS\} \subseteq VAL_{PIES}$, $\{x1, x2, x4, x8\} \subseteq VAR_{PIES}$

$\{$MRUCZEK$\} \subseteq VAL_{KOT}$, $\{x3, x5, x7\} \subseteq VAR_{KOT}$

$VAL_{ZWIERZĘ} = VAL_{PIES}\ VAL_{KOT}$, $\{x4, x5\} \subseteq VAR_{ZWIERZĘ}$

$\{$MERDA-OGONEM, SZCZEKA, MIAUCZY$\} \subseteq VAL_{ZACHOWANIE}$

$\{$PRZYJAZNY$\} \subseteq VAL_{USPOSOBIENIE}$

Zbiór parametrów bezkontekstowych $CFPAR_1$ i zbiory wartości atrybutów AT_1 wyznaczają uniwersum U_1 systemu perceptowego S_1.

Fakty wyrażone rozpatrywanym interpretem lingwistycznym mogą być opisane przez następujące U_1-fakty:

(D1) /ZACHOWANIE:PIES:AS, SZCZEKA/

(D2) /ZACHOWANIE:ZWIERZĘ:MRUCZEK, MIAUCZY/

(R1) $\forall$x1//ZACHOWANIE:PIES:x1, MERDA-OGONEM/ $\Rightarrow$
/USPOSOBIENIE:PIES:x1, PRZYJAZNY//

(R2) $\forall$x2 $\forall$x3///ZACHOWANIE:PIES:x2, SZCZEKA/ $\wedge$ /KOT:KOT:x3, x3//$\Rightarrow$
/PIES:KOGO-CZEGO:OBAWIANIE-SIĘ:KTO-CO:KOT:x3, x2//

(R3) $\forall$x4//PIES:PIES:x4, x4/$\Rightarrow$/ZWIERZĘ:ZWIERZĘ:x4, x4//

(R4) $\forall$x5//ZACHOWANIE:ZWIERZĘ:x5, MIAUCZY/$\Rightarrow$/KOT:KOT:x5, x5//

Rozpatrywany interpret lingwistyczny celu wyrażamy przez U_1-cel:

(G1) $\exists$x7 $\exists$x8/PIES:KOGO-CZEGO:OBAWIANIE-SIĘ:KTO-CO:KOT:x7, x8/

Powyższe U_1-fakty podzieliliśmy na dwie grupy.

Wśród U_1-faktów wyróżniliśmy zdania w postaci implikacyjnej R1-R4, które nazywamy U_1-regułami i oznaczamy przez RS_1. U_1-fakty D1 i D2 stwierdzające własności konkretów nazywać będziemy U_1-deklaracjami i oznaczać przez DS_1, $FS_1=DS_1 \cup RS_1$.

W systemie $S_1=/U_1, FS_1, GS_1/$ przy zastosowaniu aparatu deduckcyjnego rachuunku perceptowego PC, implicite dane są między innymi poniższe U_1-deklaracje:

(D4) /KOT:KOT:MRUCZEK,MRUCZEK/ /D2,R4,LA6,DR1/

(D5) /PIES:KOGO-CZEGO:OBAWIANIE-SIĘ:KTO-CO:KOT:MRUCZEK,AS/
/D1,D4,R2,LA6,DR1/

Przy powyższych U_1-deklaracjach wskazano w nawiasach na sposób ich wyprowadzania. U_1-reguły R2,R4 i aksjomat logiczny LA6 były stosowane w schemacie reguły odrywania DR1 do wyprowadzenia U_1-deklaracji D4 i D5.

Jak stwierdziliśmy wcześniej, interprety lingwistyczne można rozpatrywać na trzech poziomach: leksykalnym, syntaktycznym i semantycznym/wiedzy/. Ten ostatni z kolei można rozdzielić na cztery podpoziomy: deklaratywny, proceduralny, sterujący i celów.

System perceptowy jest formalizmem dającym możliwość wyrażania poziomu wiedzy interpretów lingwistycznych przez interprety perceptowe FS i GS traktowane jako podzbiory U-formuł systemu perceptowego $S=/U,FS,GS/$.

Poziom deklaratywny systemu perceptowego wyrażamy przez zbiór DS jego U-deklaracji, poziom proceduralny przez zbiór RS jego U-reguł a poziom celów wyrażamy przez zbiór GS jego U-celów. Trzy te poziomy używają atrybutów charakteryzujących konkrety rozważanej dziedziny. Natomiast poziom sterujący zawiera U-zdania /U-sterowania/ CS, $CS \subset RS \cup DS=FS$, które określają kiedy i w jaki sposób są stosowane U-deklaracje, U-cele i U-reguły w dowodach U-celów GS.

W rozpatrywanym przykładzie do poziomu deklaratywnego systemu

perceptowego S_1 zaliczymy zbiór U_1-deklaracji D1 i D2 do poziomu proceduralnego U_1-reguły R1-R4, a do poziomu celów U_1-cel G_1. Poziom sterujący nie rozpatrywano by zbytnio nie komplikować przykładu.

Skolemizację rozszerzamy na U-formuły w sposób naturalny, przy czym zbiór U-termów funkcyjnych FTR/U/ nazywać będziemy zbiorem wartości funkcyjnych i oznaczać przez FVAL, a jego elementy przez fval, w miarę potrzeby z indeksami, $FVAL = \bigcup_{at \in AT} FVAL_{at}$, $FVAL_{at} = \bigcup_{i=0}^{\infty} FVAL_{at}^{i}$.

3.2. Równoważność systemów perceptowych

W systemie perceptowym S=/U,FS,GS/ zbiór FS możemy traktować jako zbiór U-faktów danych explicite w systemie S, a zbiór D/FS/-FS U-faktów wyprowadzalnych z FS i nie należących do FS jako U-faktów danych explicite. Zbiór D/FS/ traktujemy jako zbiór wszystkich U-faktów w systemie S.

Niech $SS=\{S_i\}_{i\in I}$ będzie rodziną systemów perceptowych postaci $S_i=/U,FS_i,GS_i/$ takich, że wszystkie systemy mają to samo uniwersum U=/PAR,VAL/. Wprowadzamy relację porządkującą $\leq$ w rodzinie SS określoną w sposób poniższy:

$S_1 \leq S_2$ wtedy i tylko wtedy, gdy $D/FS_1/ \subseteq D/FS_2/$ i $GS_1 \subseteq GS_2$.

Jeśli $S_1 \leq S_2$ to S_1 nazywamy <u>podsystemem</u> S_2.

Zatem S_1 jest podsystemem S_2 wtedy i tylko wtedy, gdy zbiór $D/FS_1/$ U_1-faktów w S_1 jest zawarty w zbiorze $D/FS_2/$ U_2-faktów systemu S_2 i zbiór GS_1 U_1-celów jest zawarty w zbiorze GS_2 U_2-celów. Zaznaczmy, że zgodnie z definicją systemu perceptowego, w przeciwieństwie do zbiorów FS_i, zbiory GS_i mogą być puste.

Powiemy, że systemy S_1 i S_2 są <u>równoważne</u> $/S_1 \sim S_2/$ wtedy i tylko wtedy, gdy $S_1 \leq S_2$ i $S_2 \leq S_1$.

Systemy S_1 i S_2 mają <u>równoważną wiedzę</u> jeśli są równoważne.

Zauważmy, że dopiero w tym miejscu podaliśmy pośrednie określenie wiedzy poprzez pojęcie równoważności systemów. Wynika to z faktu, iż dwa różne systemy, dokładniej z różnymi zbiorami U-faktów, mogą mieć tą samą wiedzę.

Utwórzmy system perceptowy S_2 modyfikując system S_1 opisujący zachowanie psa Asa i kota Mruczka, dodając U_1-deklarację D6 o zachowaniu psa Azora

/D6/ /ZACHOWANIE:PIES:AZOR,MERDA-OGONEM/.

Jest oczywiste, że system $S_1=/U_1,FS_1,GS_1/$ jest podsystem

$S_2=/U_1, FS_1 \cup \{D6\}, GS_1/$ i nie jest podsystem $S_3=/U_1, FS_1 \cup \{D6\}, \emptyset/$.

Oznaczmy przez S_4 system S_1 uzupełniony o dwie U_1-deklaracje

/D4/ /KOT:KOT:MRUCZEK,MRUCZEK/

/D7/ /PIES:AS,AS/

wyrażające stwierdzenia "Mruczek jest kotem" i "As jest psem". Jest zrozumiałe, że system S_1 i system $S_4=/U_1, FS_1 \cup \{D4,D7\}, GS_1/$ są równoważne, jako że deklaracje D4,D7 dane explicite w systemie S_4 są dane implicite w systemie S_1.

Poniższe pierwsze trzy twierdzenia podają warunki na otrzymanie systemów równoważnych, a czwarte na otrzymanie podsystemów danego systemu [33].

/E01/ $D/\{A,B\} = D/\{A \wedge B\}$

/E02/ $D/FS_1 = D/FS_1'$ i $D/FS_2 = D/FS_2'$ implikuje $D/FS_1 \cup FS_2 = D/FS_1' \cup FS_2'$

/E03/ jeśli $FS_1=D/FS_1$, $FS_2=D/FS_2$ i $FS=D/FS$, to $FS=FS_1 \cup FS_2$ implikuje $FS=FS_1$ lub $FS=FS_2$

/E04/ jeśli $FS_1=D/FS_1$, $FS_2=D/FS_2$ i $FS=D/FS$, to $FS \subseteq FS_1 \cup FS_2$ implikuje $FS \subseteq FS_1$ lub $FS \subseteq FS_2$

3.3. Modelowa równoważność systemów perceptowych

Niech $S_1=/U_1, FS_1, GS_1/$ i $S_2=/U_2, FS_2, GS_2/$ będą systemami perceptowymi, tr różnowartością funkcją przyporządkującą każdemu U_1-zdaniu A U_2-zdanie B, takie że A i B są modelowo równoważne. Funkcję tr nazywamy transformacją modelowo równoważną.

Jeśli $tr/A/=B$, to powiemy, że U_2-zdanie B jest odpowiednikiem modelowo równoważnym U_1-zdania A.

Zbiór U_2-zdań BS jest odpowiednikiem modelowo równoważnym zbioru U_1-zdań AS wtedy i tylko wtedy, gdy dla każdego $A \in AS$, $tr/A/ \in BS$ i dla każdego $B \in BS$, $tr^{-1}/B/ \in AS$.

Jeśli zbiór BS jest odpowiednikiem modelowo równoważnym zbioru AS, to powiemy, że zbiory AS i BS są modelowo równoważne.

Pojęcie wyprowadzalności w systemie perceptowym jest naturalnym rozszerzeniem tego pojęcia z rachunku perceptowego na U-formuły. Obecnie pojęcie to rozszerzymy na U_2-formuły systemu perceptowego $S_2=/U_2, FS_2, GS_2/$, którego zbiór U_2-formuł $FS_2 \cup GS_2$ jest odpowiednikiem modelowo równoważnym zbioru U_1-formuł $FS_1 \cup GS_1$ systemu perceptowego $S_1=/U_1, FS_1, GS_2/$.

Operację wyprowadzania D określiliśmy w rachunku perceptowym przez zbiór aksjomatów logicznych LA i perceptowych PA oraz dwie reguły dowodzenia.

Zbiór formuł $D/\emptyset/$ jest zbiorem twierdzeń /praw/ rachunku perceptowego. Twierdzenia te są schematami formuł zbudowanymi z formuł składowych /wyjściowych/. Twierdzenia te traktowane jako U-formuły są twierdzeniami systemu perceptowego.

Jeśli w dowolnym U_1-twierdzeniu T zastąpimy wszystkie U_1-formuły składowe schematu T przez U_2-formuły będące odpowiednikami modelowo równoważnymi odpowiednich U_1-formuł, to otrzymamy U_2-formułę, którą nazywać będziemy odpowiednikiem modelowo równoważnym

twierdzenia T . W sposób analogiczny możemy potraktować reguły dowodzenia. Pod sformułowaniem "dozwolona reguła dowodzenia" rozumieć będziemy: "każda reguła dowodzenia w systemie perceptowym". Powiemy zatem, że reguła dowodzenia DR_2 jest dozwolonym odpowiednikiem modelowo równoważnym reguły DR_1, jeśli została otrzymana w wyniku zastąpienia U_1-formuł składowych schematu DR_1 przez U_2-formuły będące ich odpowiednikami modelowo równoważnymi.

W celu zilustrowania powyższych określeń rozpatrzmy odpowiedniki modelowo równoważne dla prawa /wynikającego z aksjomatu logicznego LA6/

/PT6/ $\forall xA/x/ \Rightarrow A/val/$, dla $val \in VAL_{at}$, jeśli $x \in VAR_{at}$

i dozwolonej reguły

/PR3/ $\dfrac{A/val/}{\exists xA/x/}$, dla $x \in VAR_{at}$, jeśli $val \in VAL_{at}$

Załóżmy, że transformacją modelowo równoważną jest skolemizacja skl. Skolemizacja skl nie jest jednoznaczna i właściwie nie powinniśmy jej traktować jako transformacji modelowo równoważnej. Jednakże w konkretnych przypadkach nie prowadzi to do nieporozumień, można bowiem zawsze dla konkretnie równoważnych zbiorów formuł uważać odpowiedniki skolemowe za ad hoc wzajemnie jednoznacznie wyznaczone.

Jeśli w prawie PT6 U_1-formuła $A/x/$ jest otwarta to jego odpowiednikiem skolemowym jest

$A/x/ \Rightarrow A/val/$, dla $val \in VAL_{at}$, jeśli $x \in VAR_{at}$.

Jednak w ogólnym przypadku, gdy

$skl/\forall xA/x//=$

$B/x,x_1,\ldots,x_k,f_1/x,x_1,\ldots,x_k/,\ldots,f_t/x,x_1,\ldots,x_k//$, to

$skl/A/val/=B/val,x_1,\ldots,x_k,g_1/x_1,\ldots,x_k/,\ldots,g_t/x_1,\ldots,x_k//$,

dla $f_i \in OP^{k+1}_{at_i}$, $g_i \in OP^{k}_{at_i}$, $i=1,2,\ldots,t$ i $val \in VAL_{at}$, jeśli $x \in VAL_{at}$.

Zatem odpowiednik skolemowy prawa PT6 ma postać

$$/SPT6/\quad B/x,x_1,\ldots,x_k,f_1/x,x_1,\ldots,x_k/,\ldots,f_t/x,x_1,\ldots,x_k/ \Rightarrow$$
$$B/val,x_1,\ldots,x_k,g_1/x_1,\ldots,x_k/,\ldots,g_t/x_1,\ldots,x_k/,$$
$$\text{dla } val \in VAL_{at},\ \text{jeśli } x \in VAR_{at}\ .$$

W przypadku reguły DR3, jeśli skl/A/val//=B/val/, to mamy

$$/SDR3/\quad \frac{B/val/}{B/fval/} \quad \text{dla } fval \in FVAL^{o}_{at}\ ,\ \text{jeśli } val \in VAL_{at}$$

Zauważmy, że z prawa SPT6 i reguły odrywania dostajemy dozwoloną regułę dowodzenia

$$/SDR4/\quad \frac{B/x,x_1,\ldots,x_k,f_1/x,x_1,\ldots,x_k/,\ldots,f_t/x,x_1,\ldots,x_k/}{B/val,x_1,\ldots,x_k,g_1/x_1,\ldots,x_k/,\ldots,g_t/x_1,\ldots,x_k/}$$

będącą odpowiednikiem skolemowym dozwolonej reguły

$$\frac{\forall xA/x/}{A/val/} \quad \text{dla } val \in VAL_{at},\ \text{jeśli } x \in VAR_{at}.$$

System $S_2=/U_2,FS_2,GS_2/$ jest <u>odpowiednikiem modelowo równoważnym systemu</u> $S_1=/U_1,FS_1,GS_1/$ wtedy i tylko wtedy, gdy zbiór $FS_2 \cup GS_2$ jest odpowiednikiem modelowo równoważnym zbioru $FS_1 \cup GS_1$, oraz operacja wyprowadzania D_2 w systemie S_2 jest rozszerzeniem operacji D_1 w systemie S_1 takim, że $D_2/\emptyset/$ jest zbiorem odpowiedników modelowo równoważnych twierdzeń zbioru $D_1/\emptyset/$ i ponadto regułami dowodzenia operacji D_2 są odpowiedniki modelowo równoważne dozwolonych reguł dowodzenia operacji D_1.

Zwróćmy uwagę, że jeśli system perceptowy S_2 jest odpowiednikiem modelowo równoważnym systemu S_1, to system S_2 posiada podstawowe własności logiczne systemu S_1 /poprawność i pełność/, jako że prawdziwość twierdzeń i reguł dowodzenia wynika jedynie ze struktury, czyli ze sposobu, w jaki są zbudowane, a nie z treści występujących w nich pojęć pozalogicznych.

Jeśli system S_2 jest odpowiednikiem modelowo równoważnym

systemu S_1 to powiemy, że systemy S_1 i S_2 są modelowo równoważne.

Twierdzenie o modelowej równoważności systemów perceptowych.

Jeśli system $S_1=/U_1, FS_1, GS_1/$ i $S_2=/U_2, FS_2, GS_2/$ są modelowo równoważne i zbiory FS_1 i FS_2 są niesprzeczne, to

$G_1 \in D_1/FS_1/$ wtedy i tylko wtrdy, gdy $G_2 \in D_2/FS_2/$,

Dowód twierdzenia

Załóżmy, że $G_1 \in D_1/FS_1/$, $\{F_{11}, F_{12}, \ldots, F_{1n}\} = FS_1$ i $\{F_{21}, F_{22}, \ldots, F_{2n}\} = FS_2$. Wówczas z twierdzenia /o dedukcji/DP9 $//F_{11} \wedge F_{12} \wedge \ldots \wedge F_{1n}/\Rightarrow G_1/ \in D_1/\emptyset/$. Zatem nie istnieje model M dla $\neg //F_{11} \wedge F_{12} \wedge \ldots \wedge F_{1n}/\Rightarrow G_1/$ i na mocy modelowej równoważności S_1 i S_2 nie istnieje model M dla $\neg //F_{21} \wedge F_{22} \wedge \ldots \wedge F_{2n}/\Rightarrow G_2/$. Z nieistnienia modelu dla U_2-formuły $\neg //F_{21} \wedge F_{22} \wedge \ldots \wedge F_{2n}/\Rightarrow G_2/$ wynika, że U_2-formuła $/F_{21} \wedge F_{22} \wedge \ldots \wedge F_{2n}/\Rightarrow G_2$ jest prawdziwa w każdej realizacji M, więc na mocy twierdzenia /o pełności/CT1 mamy $//F_{21} \wedge F_{22} \wedge \ldots \wedge F_{2n}/\Rightarrow G_2/ \in D_2/\emptyset/$. Korzystając ponownie z twierdzenia o dedukcji /w drugą stronę/ otrzymujemy $G_2 \in D_2/\{F_{21} \wedge F_{22} \wedge \ldots \wedge F_{2n}$ /, co równoważne jest tezie. Dowód w drugą stronę w sposób analogiczny.

Z twierdzenia o modelowej równoważności systemów perceptowych wynika, że zagadnienie reprezentacji ciała wiedzy systemu perceptowego koniecznie nie musi być związane z wyjściowym systemem perceptowym. Może np. okazać się, że przy przyjętym kryterium na schemat reprezentacji, reprezentacja ciała wiedzy w wyjściowym systemie perceptowym jest zbyt skomplikowana lub wręcz niemożliwa. Wówczas, na mocy powyższego twierdzenia można konstruować modelowo równoważny system perceptowy, w którym daje się to w ogóle zrobić lub zrobić efektywniej, z jednoczesną gwarancją na poprawne wnioskowanie.

3.4. Dedukcja regułowa

Niech $S_1=/U_1, FS_1, GS_1/$ będzie systemem perceptowym opisującym zachowanie psa Asa i kota Mruczka, a $S_1'=/U_1', FS_1', GS_1'/$ odpowiednikiem modelowo równoważnym otrzymanym z S_1 w sposób następujący:

1. odpowiednikiem U_1 deklaracji $D \in DS_1$, $DS_1 \in FS_1$ i U_1-celu $G \in GS_1$ są $skl/D/ \in DS_1'$, $DS_1' \subseteq FS_1'$ i $skl/G/ \in GS_1'$ odpowiednio
2. odpowiednikiem U_1-reguły $R \in RS_1$, $R=/A \Rightarrow B/$, $RS_1 \subseteq FS_1$, jest $/skl/A/ \Rightarrow skl/B// \in RS_1'$, $RS_1' \subseteq FS_1'$
   gdzie skl jest konstrukcją postaci Skolema.

System S_1' będziemy nazywali odpowiednikiem skolemowym systemu S_1.

Ciało wiedzy systemu S_1' wyraża się zatem przez następujące U_1'-formuły:

/D1'/ /ZACHOWANIE:PIES:AS,SZCZEKA/

/D2'/ /ZACHOWANIE:ZWIERZĘ:MRUCZEK,MIAUCZY/

/R1'/ /ZACHOWANIE:PIES:x1,MERDA-OGONEM/

/R2'/ /ZACHOWANIE:PIES:x2,SZCZEKA/ $\wedge$ /KOT:KOT:x3,x3/ $\Rightarrow$
/PIES:KOGO-CZEGO:OBAWIANIE-SIĘ:KTO-CO:KOT:x3,x2/

/R3'/ /PIES:PIES:x4,x4/$\Rightarrow$/ZWIERZĘ:ZWIERZĘ:x4,x4/

/R4'/ /ZACHOWANIE:ZWIERZĘ:x5,MIAUCZY/$\Rightarrow$/KOT:KOT:x5,x5/

/G1'/ /PIES:KOGO-CZEGO:OBAWIANIE-SIĘ:KTO-CO:KOT:K1,P1/
gdzie K1 i P1 są stałymi Skolema, $K1 \in FVAL^{o}_{KOT}$, $P1 \in FVAL^{o}_{PIES}$.

Wyprowadźmy U_1'-cel G1' z powyższego ciała wiedzy systemu S_1'. Wyprowadzimy najpierw U_1'-deklarację /KOT:KOT:MRUCZEK,MRUCZEK/, po czym korzystając z niej U_1'-cel G1'.

1. Stosując odpowiednik skolemowy aksjomatu logicznego LA6 /prawo SPT6/ dla R2' dostajemy
   /ZACHOWANIE:ZWIERZĘ:MRUCZEK,MRUCZEK/$\Rightarrow$/KOT:KOT:MRUCZEK,MRUCZEK
2. W schemacie reguły odrywania PR1 dla przesłanek D2' i U_1'-formuły otrzymanej w punkcie 1 otrzymujemy wniosek

/D4´/ /KOT:KOT:MRUCZEK,MRUCZEK/

Zastanówmy się jaki schemat postępowania wyznaczają dwa powyższe punkty wyrażające dowód U_1'-formuły D4´. Istota sprawy redukuje się do poszukiwania takiego podstawienia s za zmienne wolne U_1'-reguły R4´, by jej poprzednik stał się identyczny z jedną z U_1'-deklaracji. Wtedy to właśnie możemy oderwać następnik U_1'-reguły R4´ i przyjąć go za U_1'-formułę wyprowadzalną w S_1'. Zatem powyższe trzy punkty redukują się do poniższego schematu stosowania U_1'-reguł:

$\frac{A, B \Rightarrow C}{s/C/}$, jeśli istnieje podstawienie s, dla którego

$s/A/ = s/B/$

gdzie $A \in DS_1'$, $/B \Rightarrow C/ \in RS_1'$, $s/C/$ włączamy do DS_1'.

Podstawieniem s obejmujemy również U_1'-deklarację A, gdyż w ogólnym przypadku może ona zawierać zmienne wolne /którym odpowiadają zmienne związane kwantyfikatorem ogólnym w odpowiedniej U_1-deklaracji systemu S_1/.

Z powyższego schematu wynika, że kluczową kwestią jego stosowania, staje się wyznaczanie podstawienia s spełniającego warunek $s/A/ = s/B/$. Poszukiwanie takiego podstawienia jest ważnym procesem w systemach sztucznej inteligencji, który nazywany jest unifikacją. Jeśli istnieje takie s, dla którego $s/A/ = s/B/$ to mówimy, że A i B unifikują się, a podstawienie s nazywamy unifikatorem dla A i B.

Stosując obecnie powyższy schemat dla przesłanek U_1'-deklaracji D1´ i D4´ oraz U_1'-reguły R2´, przy podstawieniu $\{x3/MRUCZEK, x2/AS\}$, otrzymujemy U_1'-deklarację

/D5´/ /PIES:KOGO-CZEGO:OBAWIANIE-SIĘ:KTO-CO:KOT:MRUCZEK,AS/

Końcowym krokiem dowodu U_1'-celu G1´ jest zastosowanie dozwolonej reguły SDR3

$$\frac{B/val/}{B/fval/}\ ,\ \text{dla } fval \in FVAL^{o}_{at},\ \text{jeśli } val \in VAL_{at}$$

Stosując powyższą regułę do /D5´/ otrzymujemy U_1´-cel G_1´, co jednocześnie kończy jego dowód.

Zazwyczaj nie tylko interesuje nas dowód celu reprezentującego pytanie, lecz przede wszystkim odpowiedź na to pytanie tj. wskazanie takich wartości dla których U_1´-formuła celu jest spełniona. Innymi słowy interesuje nas dowód konstruktywny, wskazujący na wartości parametrów, dla których cel jest prawdziwy.

Aby otrzymać dowód konstruktywny należałoby porównać cele z deklaracjami i w wyniku zwrócić warunek spełniający kryteruim ich zgodności. Kryterium to jednocześnie powinno wskazywać na dowód celu.

Jeśli zgodność U_1´-celu i U_1´-deklaracji sprowadzimy do ich unifikacji, to warunkiem ich zgodności może być unifikator. Jednak by móc rozpatrywać cele w tej samej płaszczyźnie co fakty musimy je potraktować dualnie. To znaczy, że jeśli wartości o które pytamy są wynikiem podstawienia za zmienne wolne, a w U_1´-celu ich miejsce zastępują termy skolemowe, to należy dla U_1-celów rozpatrywać, w miejsce ich odpowiednika skolemowego, ich dualny odpowiednik skolemowy, określony przez konstrukcję dualnej postaci skolema /dskl/.

Rozpatrzmy obecnie dowód U_1´´-celu G_1´´ w systemie perceptowym S_1´´=/U_1´´, FS_1´´, GS_1´´/ który otrzymujemy z systemu S_1=/U_1, FS_1, GS_1/ w sposób następujący:

1. odpowiednikiem U_1-deklaracji $D \in DS_1$, $DS_1 \subseteq FS_1$, jest $skl/D/ \in DS_1$´´, DS_1´´$\subseteq FS_1$´´
2. odpowiednikiem U_1-celu $G \in GS_1$ jest $dskl/D/ \in GS_1$´´
3. odpowiednikiem U_1-reguły $R \in RS_1$, $RS_1 \subseteq FS_1$, R=A⇒B, jest

/skl/A/⇒skl/B//∈ RS_1'', $RS_1'' \subseteq FS_1''$.

Obecnie dowód U_1''-celu G_1''

/G_1''/ /PIES:KOGO-CZEGO:OBAWIANIE-SIĘ:KTO-CO:KOT:x7,x8/

przebiega w sposób następujący:

1. Dla podstawienia $s_1 = \{x5/MRUCZEK\}$ z D2´i R4´ otrzymujemy U_1''-deklarację D4´.
2. Dla podstawienia $s_2 = \{x2/AS, x3/MRUCZEK\}$ z D1´, D4´ i R2´ dostajemy U_1''-deklarację D5´.

   Jeśli istnieje podstawienie s i U_1''-deklaracja F, takie że $s/G_1''/ = s/F/$, to podstawienie s wyznacza odpowiedź na pytanie reprezentowane przez U_1''-cel G_1'', przy czym odpowiedź ta reprezentowana jest przez U_1''-formułę $s/G_1''/$.
3. Dla podstawienia s_2 U_1''-cel G_1'' unifikuje się U_1''-deklaracją D5´, dając tym samym odpowiedź: "Kot Mruczek obawia się kota Asa".

Przedstawiony dla powyższego przykładu schemat operacji wyprowadzania w systemie perceptowym S_1'' nazywamy dedukcją regułową. Obecnie zróbmy podsumowanie podstawowych założeń tego schematu:

Dedukcja regułowa

1. Określona dla systemu perceptowego $S_2 = /U_2, FS_2, GS_2/$ otrzymanego z systemu wyjściowego $S_1 = /U_1, FS_1, GS_1/$ w poniższy sposób:

   a/ DS_2, $DS_2 \subseteq FS_2$, jest odpowiednikiem skolemowym DS_1, $DS_1 \subseteq FS_1$

   b/ RS_2, $RS_2 \subseteq FS_2$, jest odpowiednikiem skolemowym U_1 reguł RS_1, $RS_1 \subseteq FS_1$, określonym w sposób następujący:

   dla każdej U_1-reguły $R = /A \Rightarrow B/$, $R \in RS_1$, /skl/A/⇒skl/B//∈ RS_2

   c/ GS_2 jest dualnym odpowiednikiem skolemowym GS_1.
2. Schemat dedukcji regułowej:

$\frac{A, B \Rightarrow C}{s/C/}$, jeśli istnieje podstawienie s, dla którego $s/A/ = s/B/$

gdzie $A \in DS_2$, $/B \Rightarrow C/ \in RS_2$,

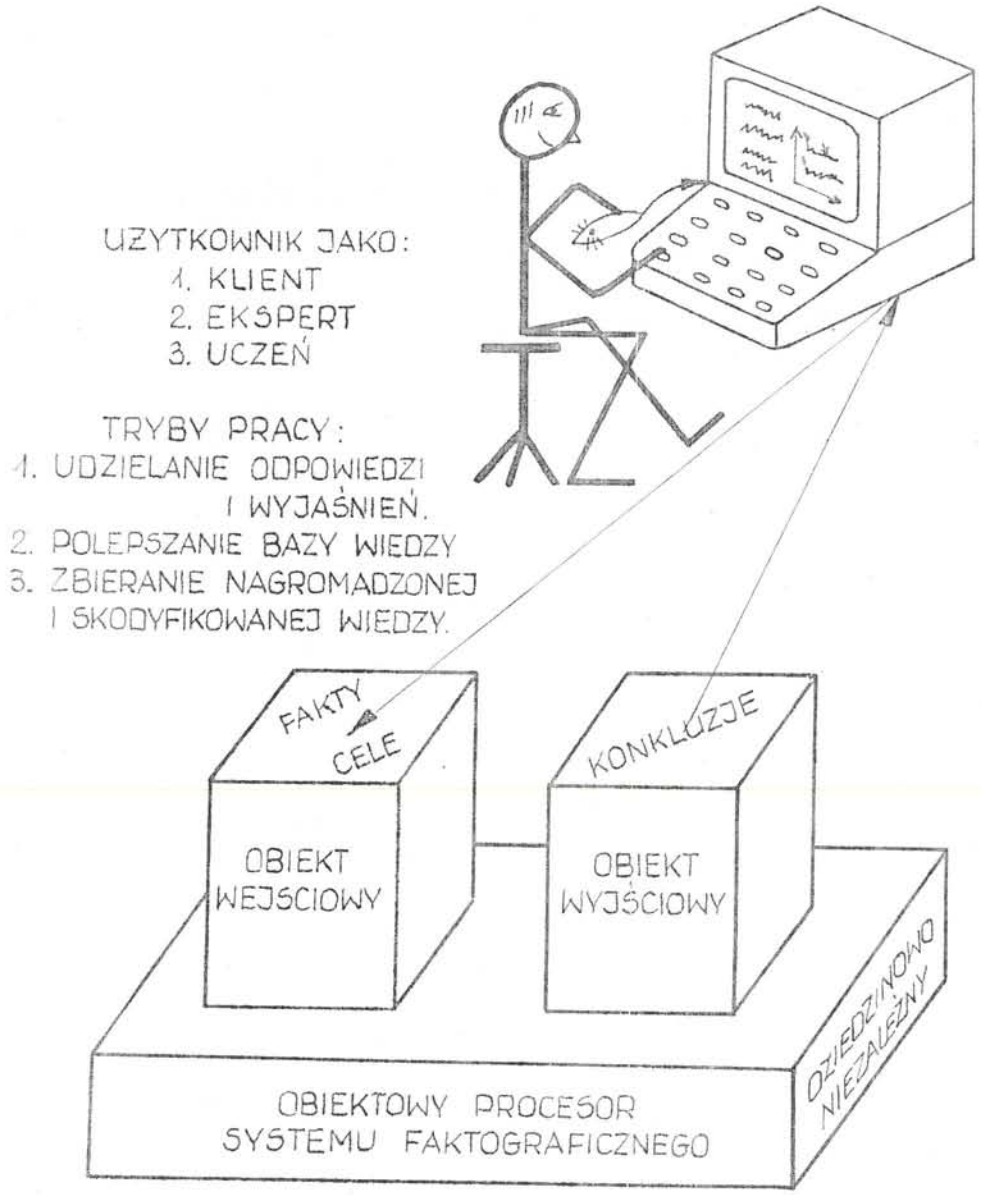
$s/C/$ włączamy do DS_2.

3. Warunek końcowy dedukcji regułowej:

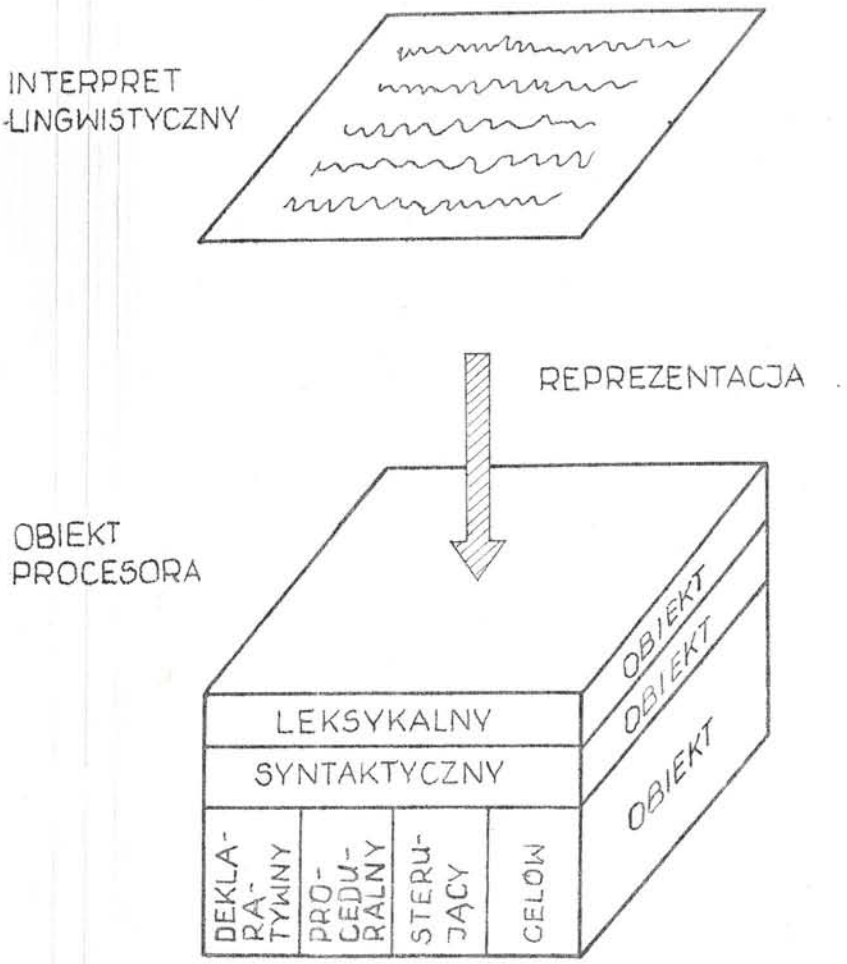
jeśli istnieje podstawienie s i U_2-deklaracja $D \in DS_2$,

takie że $s/D/ = s/G/$, dla $G \in GS_2$,

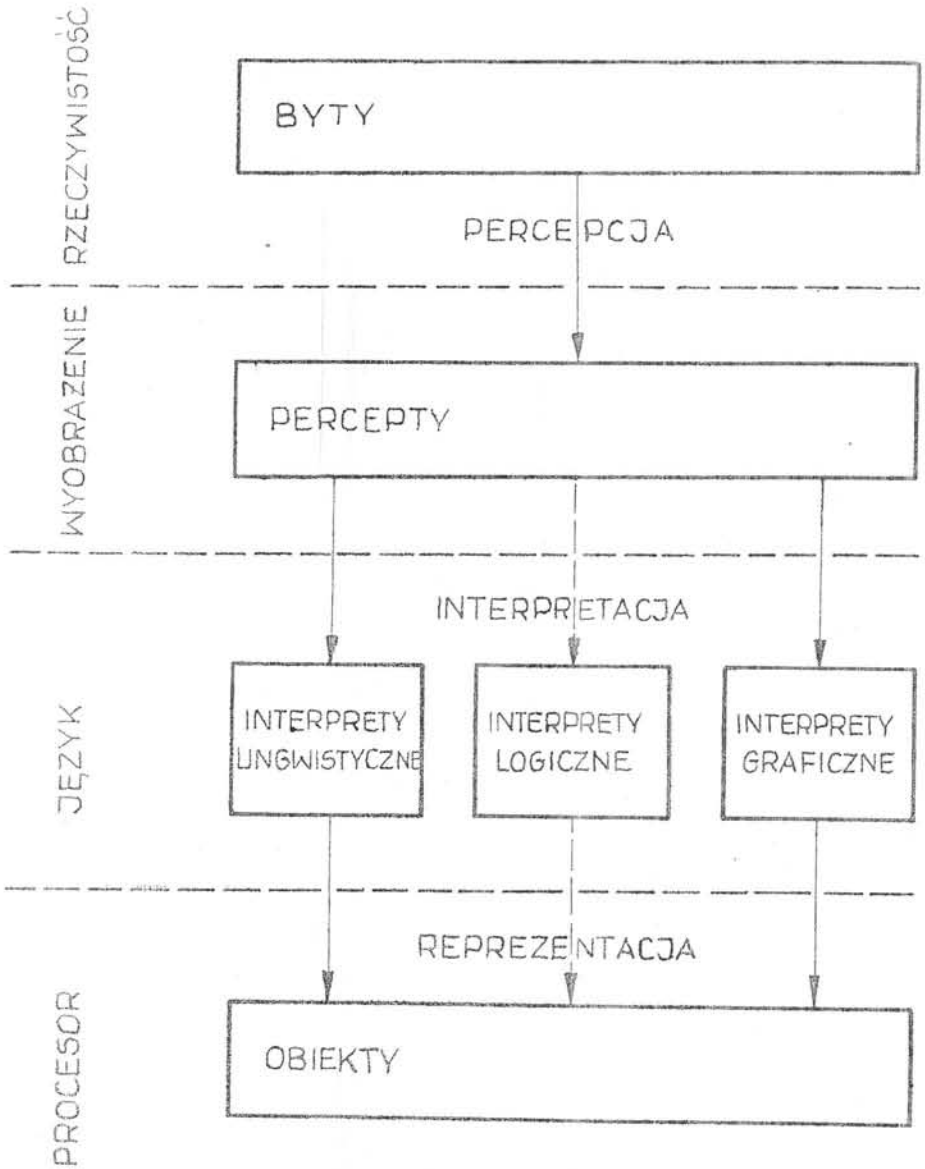
to G jest wyprowadzany w S_2.



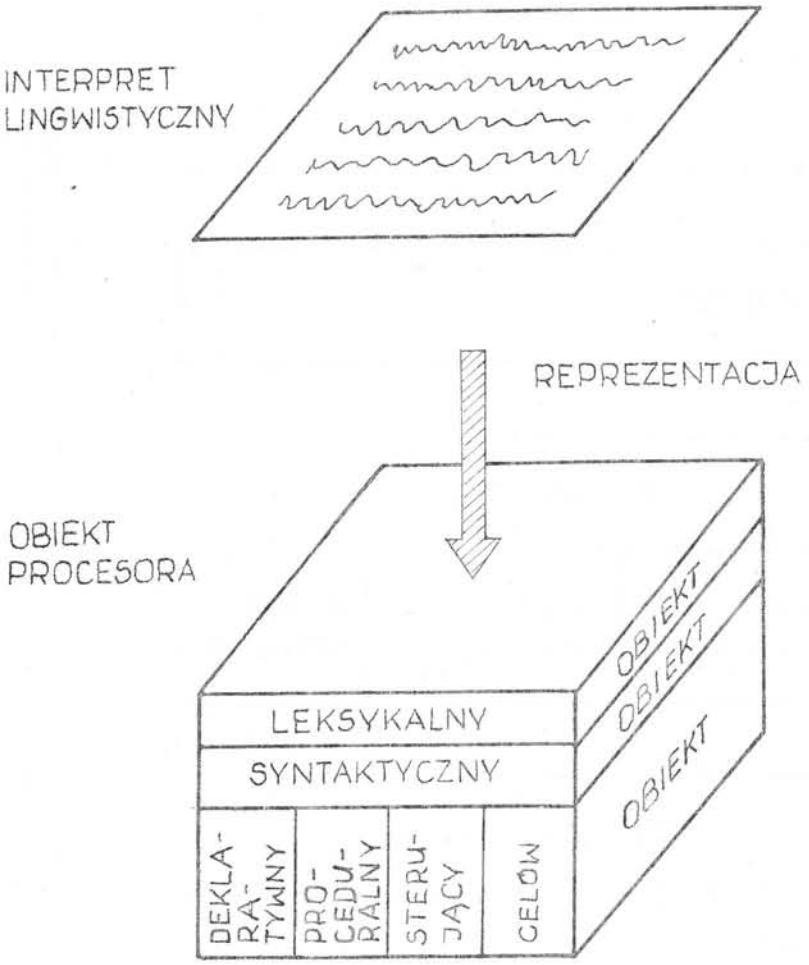
Rys.1.1. Sposób i tryby użycia systemu faktograficznego



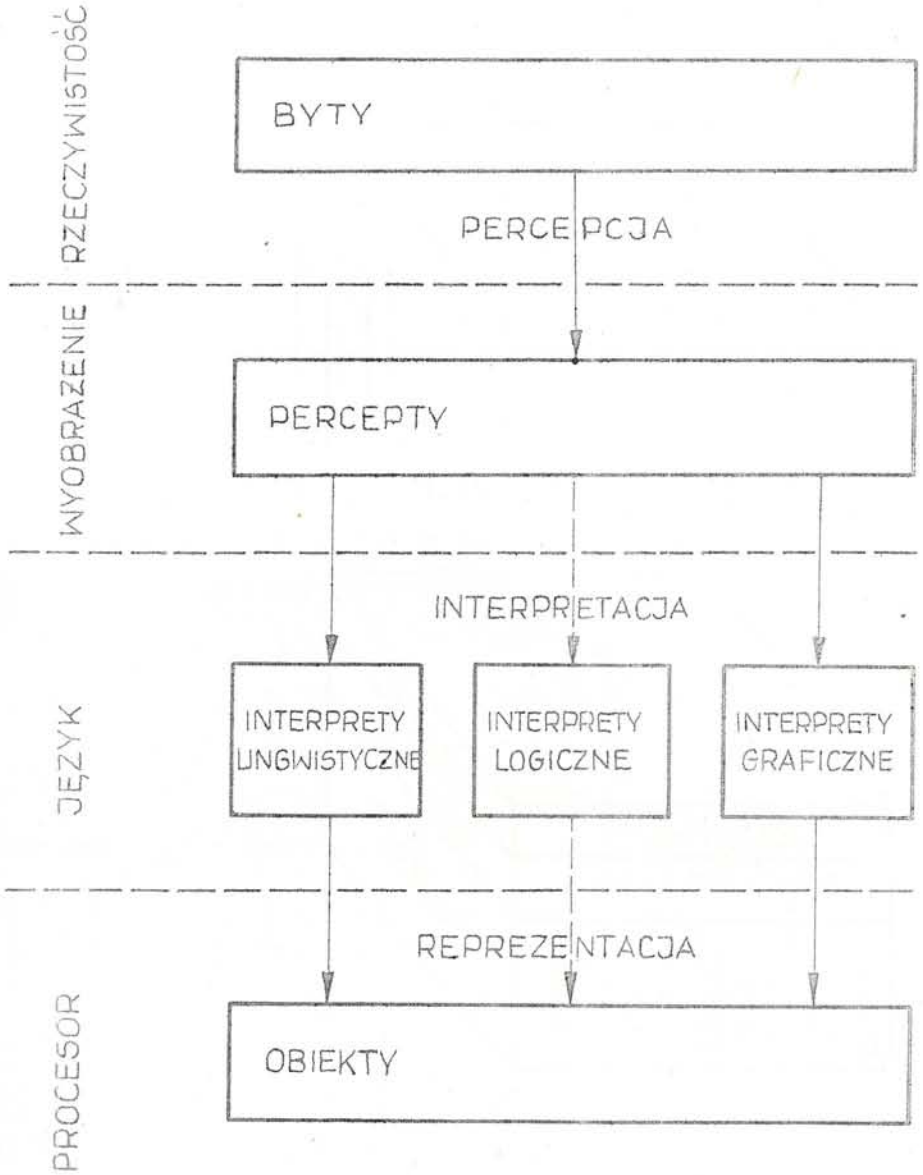
Rys.1.2. Poziomy obietu procesora



Rys. 1.3. Zależności między bytami a obiektami



Rys.1.2. Poziomy obietu procesora



Rys.1.3. Zależności między bytami a obiektami

Bibliografia

1. Anderson R.H. and Gillogly J.J./1977/. RAND Inteligent Terminal Agent /RITA/:Design Philosophy. Rand rep R-1809-ARPA.
2. Waterman D.A. /1979/. Design of a Rule-oriented System for Implementing Expertise. Rand Note N-1158-1-ARPA.
3. Nii H.P. and Aiello N./1979/. AGE/Attempt to Generalize/: A Knowledge-based Program for Building Knowledge-based Programs.Proc.Int.Sixth Joint Conf. on Artificial Intelligence, vol.6., pp.645-655.
4. Van Melle W./1979/. A Domain-independent Production Rule System for Consultation Programs.Proc.Sixth Int.Conf. on Artificial Intelligence, pp.942-947.
5. Weiss S.M. and Kulikowski C.A. /1979/. EXPERT:A System for Developing Consultation Models. Proc. Sixth Int. Joint Conf. on Artificial Intelligence, pp.942-947.
6. Kulikowski C.A. /1980/. Artificial Intelligence Methods and Systems for Medical Consultation. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.PAMI-2,No.5, pp.464-476.
7. McDermot J./1980/.R1:An Expert in the Computer System Domain.Proc. First Annual Nat. Conf. on Artificial Intelligence, pp.269-271.
8. Waterman D.A. /1981/. Rule-based Expert Systems. In Machine Intelligence, pp.323-338. Infotech State of The Art Report.Series 9, No.3.
9. Weiss S.M. and Kulikowski C.A./1981/ Expert Consultation Systems: The Expert and Casnet projects. In Machine Intelligence pp.339-352. Infotech State of The Art Report, Series 9 3.

10. Erman L.D., London P.E. and Fickas S.F./1981/. The Design and an Example Use of Hearsay-III. Proc.Seventh Int. Joint Conf. on Artificial Intelligence, pp.409-415.

11. Michie D. /1980/. Knowledge-based Systems. Report UIUCDCS-R-80-1001. University of Illinois.

12. Buchanan B.G. /1982/. Research on Expert Systems. In Machine Intelligence:10, Hayes J.,Michie D. and Pao Y-H. /eds/. Chichester: Ellis Horwood.

13. Stefik M. et al. /1982/. The Organization of Expert Systems, A Tutorial. Artificial Intelligence Journal 18, pp.135-173.

14. Nau D.S. /1983/. Expert Computer Systems. IEEE Computer, February 1983, pp.63-85.

15. Hayes-Roth F. /1984/. The Knowledge-Based Expert System: A Tutorial. IEEE Computer, September 1984, pp.11-28.

16. Shortliffe E.M. /1976/. Computer-based Medical Consultations: MYCIN, Elsevier, New YorK.

17. Waterman D.A and Hayes-Roth F. /eds/ /1978/. Pattern Directed Inference Systems. Academic Press.

18. Lindsay R.K. et al. /1980/. Applications of Artificial Intelligence for Organic Chemistry: The Dendral Project. McGraw-Hill, New York.

19. Bond A.M. /1981/. Machine Intelligence. Infotech State of The Art Report, Series 9, No.3.

20. Barr A. and Feingenbaum. The Handbook of Artificial Intelligece. William Kaufman, Menlo Park, California,Vol.1.1981, Vol.2,1982.

21. Nilsson N.J. /1982/. Principles of Artificial Intelligence. Springer Verlag.

22. Davis R. and Lenat D.B. /1982/. Knowledge Based Systems. McGrow-Hill.

23. Hayes-Roth F., Waterman D.A., and Lenat D.B. /1983/. Building Expert Systems. Addison-Wesly, Reading, Massachusetts

24. Winston P.H. /1984/. Artificial Intelligence /2nd ed./, Addison-Wesly, Reading, Massachusetts.

25. Charniak E., McDermott D./1985/. Introduction to Artificial Intelligence. Addison-Wesley.

26. Sobolewski M. /1984/ Structured Object Representations in Many-sorted Attribute Systems. Proc.First IFAC Int.Symp. on Artificial Intelligence, Leningrad. Pergamon Press, Oxford.

27. Sobolewski M. and Kulpa Z. /1984/. From Sentences to Attribute Networks. In Artificial Intelligence and Information-Control Systems of Robots, pp.345-348, North-Holland.

28. Sobolewski M. /1984/. Systemy perceptowe-logiczne systemy faktograficzne. Opracowanie IBIB PAN, temat 11/06.9.

29. Sobolewski M. /1984/. Sieci atrybutowe-obiekty procesora systemu faktograficznego. Opracowanie IBIB PAN, temat 11/06.9.

30. Górski M., Olszewski K. /1985/. Podstawowe własności logiczne rachunku perceptowego. Opracowanie IBIB PAN, temat 11/06.9.

31. Rasiowa H. and Sikorski R. /1970/. The Mathematics of Metamathematics. PWN-Polish Scientific Publishers.

32. Gburzyński P./1978/. Automatyczne dowodzenie twierdzeń z wykorzystaniem zasady rezolucji. Prace IPI PAN, Nr.319.

33. Orłowska E. and Pawlak Z. /1984/, Logical Foudations of Knowledge Representation. ICS PAS REPORTS No.537.

34. Turner R. /1984/. Logics for Artificial Intelligence. Ellis Horwood.

SYNCHRONIZACJA TRANSAKCJI W ROZPROSZONYCH BAZACH DANYCH

Tadeusz Morzy
Instytut Automatyki
Politechniki Poznańskiej
Piotrowo 3A
tel.78-23-69

1. WSTĘP

Jednym z najistotniejszych problemów stojących przed projektantem systemu zarządzania bazą danych /SZBD/ jest zagwarantowanie odpowiedniej efektywności działania systemu bazy danych. Osiągnięcie odpowiedniego współczynnika efektywności działania systemu bazy danych /SBD/ możliwe jest, przede wszystkim, poprzez zapewnienie, wielu użytkownikom, jednoczesnego, współbieżnego dostępu do bazy danych.

Problem zagwarantowania poprawności współbieżnej realizacji wielu zadań użytkowników /tzw.transakcji/, żądających jednoczesnego dostępu do bazy danych, nazywamy problemem synchronizacji. Zarządzanie współbieżnym dostępem do danych realizowane jest przez SZBD zgodnie z przyjętym, w systemie, algorytmem synchronizacji.

Poprawnie zaprojektowany algorytm synchronizacji winien zapewnić spójność bazy danych, jak również zagwarantować zrealizowanie każdej transakcji w skończonym czasie.

Złożoność problemu synchronizacji transakcji w systemach rozproszonych baz danych wynika przede wszystkim z faktu, że dowolna transakcja może żądać jednoczesnego dostępu do wielu lokalnych baz danych, zlokalizowanych na różnych, wzajemnie oddalonych stanowiskach komputerowych. Wobec powyższego, w stosun-

ku do problemu synchronizacji w systemach scentralizowanych baz danych, który polegał na konieczności zapewnienia spójności wewnętrznej lokalnej bazy danych, problem synchronizacji transakcji w systemach rozproszonych baz danych jest rozszerzony o konieczność zapewnienia spójności wewnętrznej danych należących do różnych lokalnych baz danych oraz o konieczność zapewnienia spójności zewnętrznej, rozumianej jako identyczność wszystkich kopii fizycznych tej samej danej logicznej.

Dodatkową trudność konstrukcji algorytmów synchronizacji transakcji w systemach rozproszonych baz danych stanowi fakt, że w systemach takich żadne stanowisko komputerowe nie dysponuje pełną informacją o globalnym stanie całego systemu. Stąd konieczność podejmowania decyzji zarządzających na danym stanowisku komputerowym na podstawie niepełnej i nie w pełni aktualnej informacji o aktywności pozostałych stanowisk komputerowych.

W problemie synchronizacji transakcji w systemach rozproszonych baz danych można wyróżnić dwa aspekty: aspekt spójności rozproszonej bazy danych oraz aspekt konfliktów działania systemu. Oba te aspekty przedstawimy szczegółowo w rozdziałach 3 i 4, po wprowadzeniu w rozdziale 2 podstawowych pojęć i definicji. Rozdział 5 zawiera przegląd znanych metod synchronizacji w systemach rozproszonych baz danych na przykładzie wybranego algorytmu synchronizacji. Na zakończenie, w rozdziale 6, omówiono i scharakteryzowano krótko nowe tendencje w zakresie metod synchronizacji transakcji w systemach rozproszonych baz danych.

2. PODSTAWOWE POJĘCIA I DEFINICJE

W tym punkcie krótko przedstawimy podstawowe pojęcia i definicje odnoszące się do rozważanego w pracy, modelu systemu rozproszonej bazy danych i modelu transakcji. Podstawę naszych roz-

ważań stanowić będzie formalny model rozproszonej bazy danych /RBD/ przedstawiony w pracach [22 , 23 , 59].

Rozproszoną bazą danych /RBD/ nazywać będziemy czwórkę /ST, $\mathcal{L}$, D, $\mathcal{L}oc$/, gdzie $ST = \{ST_1,\ldots,ST_N\}$ jest zbiorem stanowisk komputerowych przechowywania danych, $\mathcal{L} = \{X_1,\ldots,X_n\}$ jest zbiorem danych logicznych, nazywanym logiczną bazą danych, $D = \{x_{11},\ldots,x_{1m_1}, x_{21},\ldots,x_{2m_2},\ldots,x_{n1},\ldots,x_{nm_n}\}$ jest zbiorem danych fizycznych, nazywanym fizyczną danych, $\mathcal{L}oc : D \longrightarrow ST$ jest funkcją, określającą lokalizację danej fizycznej $x \in D$ w zbiorze ST. Logiczna baza danych $\mathcal{L}$ reprezentuje RBD z punktu widzenia jej użytkownika, który w terminach logicznej bazy danych definiuje swoje zadania zwane transakcjami. W klasycznym modelu RBD, który będziemy rozważać w tej pracy, użytkownik nie rozróżnia rozproszonej bazy danych od scentralizowanej bazy danych i pozostaje nieświadomy faktu fizycznego rozproszenia i ewentualnej duplikacji danych, do których żąda dostępu[1/].

Daną logiczną interpretujemy jako najmniejszą jednostkę RBD, o którą mogą współubiegać się transakcje użytkowników. W zależności od systemu mogą to być: relacje, krotki, pliki, zbiory, itp.

Dowolna dana logiczna $_i$ może występować w RBD w postaci pojedynczej kopii fizycznej x_{i1}, przechowywanej na dowolnym stanowisku komputerowym, lub w postaci kilku kopii fizycznych $x_{i1},\ldots,x_{im_i}$ przechowywanych na m_i - różnych stanowiskach komputerowych. Dla dowolnych k,l, $1 \leq k < l \leq m_i$, zachodzi $\mathcal{L}oc/x_{ik}/ \neq \mathcal{L}oc/x_{il}/$.

Fizyczna baza danych D stanowi implementację logicznej bazy danych $\mathcal{L}$ dla przyjętej konfiguracji systemu rozproszonego.

[1/] W tak zwanym modelu wielobazowym /ang.multibase model/ [25, 26] użytkownik jest świadomy faktu rozproszenia i duplikacji danych i sam określa strategię dostępu do lokalnych baz danych

Pojedyncza dana fizyczna $x \in D$ implementowana jest postaci jednej lub kilku tzw.jednostek dostępu /ang.access unit/. Przez jednostkę dostępu rozumiemy najmniejszą jednostkę bazy danych dostępną dla systemu zarządzania RBD. W praktyce tą najmniejszą jednostką dostępu jest strona pamięci wirtualnej [7, 8, 22, 23, 25, 29, 59, 82].

Zbiór wszystkich danych fizycznych zlokalizowanych na pojedynczym stanowisku komputerowym ST_i, $\{x_{ij}: 1 \leq i \leq n, 1 \leq j \leq m_i, Loc(x_{ij}) = ST_i\}$, nazywamy lokalną bazą danych /LBD_i/. Każda LBD zarządzana jest autonomicznie przez system zarządzania lokalną bazą danych /SZLBD/.

Zarządzanie RBD jest realizowane przez system zarządzania rozproszoną bazą danych /SZRBD/, który organizuje współpracę równorzędnych i autonomicznych SZLBD.

Zauważmy, że pojęcie rozproszenia odnosi się w rónej mierze do modelu danych jak i do systemu zarządzania RBD. Należy podkreślić, że w powyższym modelu RBD dane zdefiniowano jako niezależne obiekty reprezentujące ten sam poziom abstrakcji. W dalszej części artykułu przedstawimy rozwiązania, w których warunek ten został osłabiony i zakłada się bardziej złożone struktury danych.

Jak już wspomnieliśmy elementarną jednostką interakcji użytkownika z SZRBD jest transakcja. Transakcję winny cechować z punktu widzenia użytkownika, następujące własności: spójność, niepodzielność, trwałość i niezależność od efektu domina [23, 35, 36, 39, 60, 75, 76]. Koncepcja transakcji pozwala rozłącznie rozpatrywać problemy synchronizacji, poprawności dostępu do bazy danych oraz niezawodności.

Formalnie transakcję T_i definiujemy jako parę $T_i = (\overline{T}_i, \overline{\prec}_{T_i})$, gdzie $\overline{T}_i$ jest zbiorem operacji odczytu i/lub zapisu danych, natomiast $\overline{\prec}_{T_i}$ jest relacją częściowo porządkującą zbiór $\overline{T}_i$. Przez

$r_{ij}^{ST_k}(x)$ ($W_{ij}^{ST_k}(x)$) oznaczać będziemy operację odczytu /zapisu/ danej x na stanowisku ST_k przez transakcję T_i. Jeżeli nie będzie to prowadziło do niejasności, lub nie będzie to konieczne, pomijać będziemy w powyższym zapisie oznaczenie stanowiska komputerowego.

$$T_i: \quad r_{11}^{ST_1}(x) \longrightarrow W_{13}^{ST_1}(x), \qquad r_{12}^{ST_2}(y) \nearrow W_{13}^{ST_1}(x)$$

gdzie $\longrightarrow$ łączy dwie operacje O_{ij} oraz O_{ik}, wtedy gdy $O_{ij} \prec_{T_i} O_{ik}$

Zbiór transakcji aktualnie wykonywanych w systemie $T_1, T_2, \ldots, T$ oznaczać będziemy przez τ.

Parę (RDB, τ), gdzie RBD oznacza rozproszoną bazę danych, $\tau = (T_1, T_2, \ldots, T_n)$ zbiór niezależnych transakcji, nazywać będziemy systemem rozproszonej bazy danych (SRBD). Przejdziemy obecnie, w następnych rozdziałach, do sformułowania i przedstawienia problemu poprawnego zarządzania współbieżnym wykonywaniem transakcji w środowisku rozproszonym.

3. PROBLEM WSPOŁBIEŻNEJ REALIZACJI TRANSAKCJI

Jak wiadomo, niekontrolowana współbieżna realizacja zbioru transakcji w bazie danych może prowadzić do naruszenia spójności i integralności tej bazy danych. Formalnym opisem wykonania zbioru transakcji τ w bazie danych jest tzw.realizacja [6, 7, 23, 25, 45, 46, 49, 59, 62, 75].

Realizacją r zbioru transakcji τ nazywać będziemy częściowo uporządkowany zbiór $(\bar{T}, \prec_r)$, gdzie $\bar{T} = \left\{ \bar{T}_{ij} : 1 \leq i \leq m, 1 \leq j \leq m_i \right\}$ jest zbiorem wszystkich elementarnych operacji, natomiast $\prec_r$ jest relacją częściowego porządku, dla której spełnione są następujące warunki :

(i) $\bigwedge_{1 \leq j < k \leq m_i}^{j,k} (T_{ij} \prec_{T_i} T_{ik}) \Rightarrow (T_{ij} \prec_r T_{ik})$, (ii) dla dowolnych

T_{ik}, T_{jl}, $1 \leq i \leq m$, $1 \leq j \leq m$, $1 \leq k \leq m_i$, $1 \leq l \leq m_j$, takich że $\mathcal{L}oc(T_{ik}) = \mathcal{L}oc(T_{jl})$ i $i \neq j$, zachodzi $(T_{ik} \overset{r}{\prec} T_{jl}) \vee (T_{jl} \overset{r}{\prec} T_{ik})$. W istocie realizacja r stanowi opis dostępu do bazy danych. Jednocześnie, dana realizacja jest wynikiem działania określonego algorytmu synchronizacji.

W SRBD realizację r zbioru transakcji τ na N stanowiskach systemu modelujemy w postaci zbioru N realizacji lokalnych $r = (r_1, r_2, \ldots, r_N)$ będących opisem dostępu do lokalnychh baz danych $LBD_1, \ldots, LBD_N$. Ogólnie przyjętym kryterium poprawności współbieżnej realizacji r jest tzw. kryterium uszeregowalności [6, 8, 25, 45, 46, 49, 62], zgodnie z którym dowolna realizacja współbieżna r zbioru transakcji τ jest poprawna, jeżeli jest ona równoważna dowolnej realizacji sekwencyjnej zbioru τ[2/]. Jednakże, jak wiadomo, problem uszeregowalności jest problemem NP-zupełnym. W praktyce, za podstawowe kryterium poprawności przyjęto kryterium D-uszeregowalności, które stanowi warunek dostateczny uszeregowalności [6,8, 26, 45, 46, 62, 72, 75, 87]. Zgodnie z kryterium D-uszeregowalności, dowolna realizacja współbieżna r jest poprawna jeżeli jest ona równoważna, w sensie rozwiązania konfliktów dostępu, dowolnej realizacji sekwencyjnej. Mówimy, że operacje T_{ik} i T_{jl} znajdują się w konflikcie dostępu w stosunku do danej x, co zapisujemy $T_{ik} \overset{©}{x} T_{jl}$, jeżeli żądają dostępu do tej samej danej fizycznej x i co najmniej jedna z tych operacji jest operacją zapisu. Mówimy dalej, że dwie transakcje T_i i T_j znajdują się w konflikcie dostępu, co zapisujemy $T_i \,©\, T_j$, jeżeli istnieje taka dana x i takie elementarne operacje $T_{jk} \in T_i$ i $T_{jl} \in T_j$, że $T_{ik} \overset{©}{x} T_{jl}$. Transakcje znajdujące się w konflikcie muszą być wykonane ściśle sekwencyjnie, innymi słowy, muszą pozostawać względem

2/ W pracy [49] wykazano, że kryterium uszeregowalności jest najsłabszym /tj.najmniej ograniczającym/ kryterium poprawności realizacji współbieżnej r, gdy poprawność tę rozpatruje się w kontekście informacji czysto syntaktycznej

siebie w odpowiedniej relacji poprzedzania /oznaczonej dalej przez $\longrightarrow$/. Relację tę zdefiniujemy następująco.

Mówimy, że dwie operacje elementarne T_{ik}, T_{jl} znajdują się w relacji poprzedzania dla danej x, co zapisujemy $T_{ik} \xrightarrow{x} T_{jl}$, jeżeli: $(T_{ik} \overset{\copyright}{x} T_{jl}) \wedge (T_{ik} \overset{r}{\prec} T_{jl})$. Mówimy dalej, że dwie transakcje znajdują się w relacji poprzedzania, co zapisujemy $T_i \longrightarrow T_j$, jeżeli istnieje taka dana x i takie operacje elementarne T_{ik}, T_{jl}, $T_{ik} \in T_i$, $T_{jl} \in T_j$, że $T_{ik} \xrightarrow{x} T_{jl}$. Łatwo zauważyć, że relacja $\longrightarrow$ jest relacją częściowo porządkującą zbiór transakcji τ. Korzystając z relacji $\longrightarrow$ możemy obecnie, w nieco innej postaci, sformułować kryterium D-uszeregowalności realizacji r [6, 7, 8, 25, 30, 46, 62, 72, 80, 86, 87].

Kryterium D-uszeregowalności. Dowolna realizacja r zbioru transakcji $\tau = \{T_1, \ldots, T_m\}$ jest uszeregowalna, jeżeli relacja poprzedzania $\longrightarrow$ jest acykliczna. Porządek wykonywania transakcji wprowadzony przez acykliczną relację $\longrightarrow$ nazywać będziemy porządkiem realizacji [8]. Jak już wspomnieliśmy, zdefiniowane wyżej kryteria uszeregowalności i D-uszeregowalności zostały sformułowane przy wykorzystaniu informacji wyłącznie czysto syntaktycznej. W przypadku, gdy dysponujemy pewną dodatkową informacją, np.: o strukturze danych, o ograniczeniach integralnościowych czy zbiorze przetwarzanych transakcji, kryterium uszeregowalności można osłabić uzyskując w wyniku większy stopień współbieżności wykonywania transakcji [9, 28, 34, 46, 48, 49, 54, 70, 72, 73, 85]

4. PROBLEM KONFLIKTÓW DZIAŁANIA SYSTEMÓW ROZPROSZONEJ BAZY DANYCH

Kryterium uszeregowalności nie jest wystarczającym kryterium poprawności działania systemów rozproszonej bazy danych, lub innymi słowy, poprawności synchronizacji transakcji w SRBD. Jak wspomnieliśmy we wstępie, problem poprawności działania SRBD rozpatrywać będziemy w ujęciu globalnym, jako: (i) problem zapewnie-

nia spójności i integralności RBD, (ii) problem zapewnienia zrealizowania każdej transakcji zainicjowanej w systemie. Przyjęcie określonej metody synchronizacji transakcji w SRBD zapewniającej spójność wewnętrzną i zewnętrzną RBD (warunek (i)) może pociągnąć za sobą możliwość wystąpienia w systemie tzw. konfliktów działania SRBD /ang.system performance failures/. Wyróżniamy cztery typy tych konfliktów, a mianowicie: martwy punkt, cykliczny restart, stałe zablokowanie i stałe restartowanie. Wystąpienie w systemie któregokolwiek z powyższych konfliktów działania prowadzi w konsekwencji do niespełnienia warunku (ii), tj. do niezakończenia wykonywania transakcji lub zbioru transakcji zainicjowanych w systemie. Niezrealizowanie transakcji prowadzi w pewnym sensie również do naruszenia spójności RBD, rozumianej teraz szereko jako relacji odpowiedniości pomiędzy "światem zewnętrznym" a jego abstrakcyjnym odzwierciedleniem, jakim jest RBD. Obecnie krótko scharakteryzujemy wspomniane konflikty. Konflikt martwego punktu definiujemy jako stan systemu, w którym dwie lub więcej transakcji oczekuje wzajemnie na uwolnienie danych niezbędnych do zakończenia ich wykonywania. Problem martwego punktu był szeroko rozważany w literaturze [8, 17, 19, 25, 37, 41, 43, 44, 56, 68, 81, 85]. Warto zwrócić uwagę na fakt, że w SRBD stan martwego punktu może wystąpić lokalnie, na pojedynczym stanowisku komputerowym, lub globalnie, w obrębie całego SRBD, obejmując jednocześnie wiele stanowisk komputerowych. W konsekwencji rozwiązanie tego problemu jest znacznie trudniejsze niż w przypadku systemów scentralizowanych baz danych [7, 43]. Wyróżnia się dwie ogólne metody rozwiązania tego problemu w SRBD, mianowicie, metodę wykrywania i likwidacji oraz metodę unikania.

Metoda wykrywania i likwidacji polega na okresowym sprawdzaniu czy SRBD nie znajduje się w stanie martwego punktu. W tym celu konstruowany jest graf czekania transakcji i badane jest ist-

nienie cyklu w tym grafie. Zasadniczą trudnością w przypadku implementacji tej metody w SRBD jest problem efektywnego konstruowania grafu czekania transakcji, który jest rozproszony pomiędzy stanowiskami komputerowymi [7, 8, 37, 41, 46].

Metoda unikania martwego punktu polega na sprawdzaniu w momencie żądania ze strony transakcji T_j niekompatybilnego dostępu do danej przydzielonej transakcji T_i, czy może ona czekać na zakończenie lub wycofanie transakcji T_i tak, aby nie spowodować wystąpienia w SRBD stanu martwego punktu [8, 17, 19]. Najprostszym przykładem procedury testującej o wymaganej właściwości jest procedura, która każdorazowo w momencie zgłoszenia żądania niekompatybilnego dostępu do danej powoduje natychmiastowe wycofanie transakcji zgłaszającej takie żądanie [12, 71]. Najciekawszym przykładem realizacji metody unikania są dwie procedury, oznaczone przez WD i WW, zaproponowane w pracy [68], które jak wykazano, minimalizują liczbę restartów transakcji.

Drugi ze wspomnianych konfliktów - konflikt stałego zablokowania zachodzi wówczas, gdy na skutek nieskończonego strumienia nowych transakcji inicjowanych w systemie, w nieznanych a priori momentach, transakcja lub zbiór transakcji nigdy nie uzyskają dostępu do wszystkich danych, niezbędnych do ich wykonania i tym samym nie zostaną nigdy zrealizowane [17, 18, 43, 67, 68]. Stosowane w SRBD metody ochrony przed wystąpieniem konfliktu stałego zablokowania nie różnią się w istocie od znanych metod stosowanych w systemach scentralizowanych, a mianowicie metody zapobiegania, polegającej na ustalaniu kolejności dostępu do danych zgodnie z kolejnością inicjowania transakcji w SRBD [17, 18, 67, 68] lub metody wykrywania i likwidacji, polegającej na pomiarze parametrów transakcji przebywających w systemie i określeniu na tej podstawie, czy transakcję należy traktować jako stale zablokowaną. W przypadku wykrycia takiej transakcji, likwidacja stanu

stałego zablokowania polega najczęściej na odcięciu strumienia nowych transakcji [17, 18]. Jednakże specyfika systemu rozproszonego, charakteryzującego się rozproszeniem fizycznych zasobów systemowych oraz autonomią stanowisk komputerowych, powoduje, że podstawowym mechanizmem rozwiązania konfliktów dostępu jest odrzucanie transakcji zdalnych. Prowadzi to w konsekwencji do możliwości wystąpienia dwóch pozostałych konfliktów, mianowicie: konfliktu cyklicznego restartowania i stałego restartowania. Konflikt cyklicznego restartowania zachodzi wówczas, gdy dwie lub więcej transakcji powoduje wzajemne wycofanie lub wywłaszczenie [7, 8, 12, 21, 23, 43, 67, 68] .

Konflikt stałego restartowania zachodzi wówczas, gdy na skutek nieskończonego strumienia nowych transakcji inicjowanych w systemie w nieznanych a priori momentach, transakcja lub zbiór transakcji są stale wycofywane lub wywłaszczane, i tym samym nie zostaną nigdy zrealizowane.

Przedstawimy obecnie krótko metodę rozwiązywania konfliktów: cyklicznego restartu i stałego restartowania opartą na metodzie wykrywania i likwidacji zaproponowanq w pracy [21].

Każda transakcja T_i otrzymuje jednoznaczny identyfikator oznaczany przez $TS(T_i)$, związany z czasem jej inicjowania (identyfikatorem tym może być etykieta czasowa inicjowania). Ponadto w systemie określa się liczbę restartów transakcji, począwszy od której uważać się będzie transakcję za znajdującą się w stanie stałego restartowania. W przypadku przekroczenia tej liczby, czyli wykrycia konfliktu stałego restartowania lub cyklicznego restartu, następuje zamarkowanie restartującej transakcji oraz zamarkowanie wszystkich danych, do których zamarkowana transakcja żąda dostępu. Markowanie danych polega na przydzielaniu im znaczników czasowych (oznaczonych -- mark(x)) równych identyfikatorowi $TS(T_i)$ zamarkowan[illegible] transak[illegible].

Wprowadzamy warunek wstępny dostępności danej x dla dowolnej transakcji T_i w postaci: $TS(T_i) \leqslant mark(x)$.

Znacznik czasowy niezamarkowanej danej x jest równy $+\infty$ i wówczas zachodzi $\bigwedge_{1 \leqslant i \leqslant n} TS(T_i) < mark(x)$. Markowanie danych realizowane jest zgodnie z następującym schematem:

```
if    mark(x) < TS(T_i)
   then    mark(x) <-- TS(T_i),
```

gdzie $TS(T_i)$ oznacza etykietę czasową zamarkowanej transakcji T_i. Włączenie powyższej metody rozwiązania konfliktów cyklicznego i stałego restartowania do metody blokowania jest natychmiastowe. Transakcja T_i może założyć blokadę danej x jeżeli spełnione są dwa warunki: (1) $TS(T_i) \leqslant mark(x)$, (2) typ blokady żądanej przez T_i jest kompatybilny z aktualną blokadą danej x. Powyższy warunek (1) wprowadza dodatkowe ograniczenie dostępności danej, odcinając dostęp do niej strumieniowi nowo zainicjowanych transakcji. Jednakże odcięcie strumienia nowo zainicjowanych transakcji nie jest całkowite - transakcje żądające dostępu do innych danych są normalnie wykonywane. Jak wynika z przeprowadzonych badań symulacyjnych [21, 59] metoda ta może posłużyć do poprawy efektywności działania systemu, nawet gdy nie zachodzi przypadek stałego restartowania transakcji. Ogranicza ona bowiem liczbę restartów transakcji co prowadzi do poprawy takich kryteriów oceny działania SRBD jak średni czas odpowiedzi oraz średnia liczba restartów. Skuteczność tej metody zależy od właściwego doboru granicy liczby restartów przyjmowanej za symptom wystąpienia konfliktów: stałego restartowania lub cyklicznego restartu transakcji.

5. METODY SYNCHRONIZACJI

Znane w literaturze algorytmy synchronizacji można sklasyfikować jako należące do trzech podstawowych metod, mianowicie:

metody blokowania, metody porządkowania transakcji według etykiet czasowych oraz metody walidacji. Istnieje szereg algorytmów, które stanowią próbę połączenia wymienionych, podstawowych metod synchronizacji. Wynika to z faktu, iż metody te są w pewnej mierze wzajemnie komplementarne. Algorytmy te zaliczono do tzw.metody mieszanej i omówiono w następnym rozdziale. Obecnie krótko scharakteryzujemy trzy podstawowe metody synchronizacji. Bardziej szczegółową charakterystykę tych metod można znaleźć w pracy przeglądowej [23],

5.1. Metoda blokowania

Metoda blokowania jest najpowszechniej stosowaną metodą synchronizacji i posiada bogatą literaturę [1, 2, 3, 7, 8, 12, 25, 30, 32, 59, 68, 73, 81, 85, 86, 87]. W metodzie blokowania zakłada się, że każda transakcja przed odczytem lub zapisem danej musi uzyskać założenie odpowiedniej blokady tej danej oraz, że blokady wszystkich danych, do których transakcja żądała dostępu zostaną zdjęte przed zakończeniem jej wykonania. Zakładanie blokad danych, do których transakcja żąda dostępu, ma na celu niedopuszczenie do odczytu lub zapisu tych danych przez inne transakcje w czasie, gdy ograniczenia integralnościowe RBD mogą być przejściowe naruszone. Najszerzej stosowanym w praktyce algorytmem metody blokowania jest algorytm blokowania dwufazowego (2PL). Istotą tego algorytmu jest wymaganie, aby wykonywanie każdej transakcji przebiegało w dwóch fazach: w fazie blokowania (kiedy transakcja żąda dostępu do kolejnych danych i nie uwalnia żadnych zasobów) oraz w fazie odblokowania (kiedy transakcja odblokowuje kolejne dane).

W pracach [49, 87] udowodniono, że algorytm 2PL jest optymalnym algorytmem blokowania w tym sensie, że powoduje on zablokowanie minimalnej liczby danych na minimalne odcinki czasu. Główną wadą algorytmu 2PL jest to, że nie rozwiązuje on samodzielnie prob-

lemu konfliktów działania i musi być uzupełniony odpowiednimi metodami opisanymi w rozdziale 4. Ponadto, należy podkreślić, że wprowadzenie blokad ogranicza dostępność danych. Operacje założenia blokad i zdjęcia blokad szeregują każdą parę transakcji w przypadku jeżeli istnieje chociażby jedna dana, do której obie transakcje żądają dostępu /nawet jeżeli nie prowadzi to do naruszenia kryterium uszeregowalności/. Zmniejsza to oczywiście stopień współbieżności wykonywania transakcji. Warto zauważyć, że jeżeli projektant SRBD dysponuje a priori informacją o logicznej strukturze danych /np.o ograniczeniach integralnościowych/ lub fizycznej strukturze danych /np. o drzewiastej strukturze danych/, to wymaganie dwufazowej realizacji transakcji może być osłabione. W pracach [46, 48, 72, 73, 85, 87], zaproponowano szereg nie-dwufazowych algorytmów blokowania dla SRBD, w których struktura danych może być przedstawiona za pomocą acyklicznego grafu skierowanego. Algorytmy te nazwano ogólnie algorytmami drzewiastymi. Podejście to zostało następnie uogólnione w pracach [85, 87] na przypadek, gdy struktura danych może być przedstawiona za pomocą hypergrafu. Mówimy w tym przypadku o algorytmach hypergrafowych. Na zakończenie omawiania metody blokowania należy krótko wspomnieć o problemie ziarnistości blokad [16, 48, 66] i hierarchicznej wersji algorytmu blokowania [16, 38, 48]. Problem ziarnistości blokad polega na doborze rozmiaru blokowanej jednostki. Wiąże się on z problemem realizacji w SRBD transakcji o różnych liczbach danych, do których dokonywany jest dostęp. Maksymalizacja współbieżności zbioru małych transakcji wymaga małych jednostek blokady, natomiast minimalizacja kosztu blokowania danych dla dużych transakcji wymaga dużych jednostek blokady. W pracach [38, 48] zaproponowano koncepcję hierarchicznego blokowania, w której przyjęto hierarchiczną strukturę jednostek blokowania. Zgodnie z tą koncepcją baza danych widziana jest jako hierarchia jednostek blokowania /tzw.

ziaren/. Zakładając blokadę na określonym poziomie hierarchii blokujemy, implicite, wszystkie następniki tej jednostki w hierarchii. Ponieważ transakcje realizują blokowanie na różnych poziomach hierarchii blokad wprowadzono koncepcję tzw.blokad intencjonalnych [38, 48, 66]. Założenie blokady intencjonalnej oznacza, że blokowanie właściwe realizowane jest na niższym poziomie hierarchii blokad.

5.2. Metoda porządkowania transakcji według etykiet czasowych

Metoda porządkowania transakcji według etykiet czasowych (T/O) stanowi drugą z podstawowych metod synchronizacji transakcji w SRBD [7, 8, 16, 23, 25, 32, 40, 46, 64, 65, 78].

Podstawowym pojęciem wykorzystywanym w tej metodzie jest pojęcie etykiety czasowej inicjowania transakcji. Etykieta czasowa inicjowania transakcji T_i, oznaczona przez $TS(T_i)$, jest konkatenacją jednoznacznego identyfikatora stanowiska komputerowego, na którym transakcja T_i jest inicjowana i stanu zegara fizycznego na tym stanowisku w chwili inicjowania transakcji [7, 8, 50].

Istota metody T/O polega na przyjęciu a priori porządku realizacji zbioru transakcji w SRBD. Porządek ten jest jednoznacznie określony na podstawie etykiet czasowych inicjowania. Zadaniem procedur synchronizacji jest testowanie czy porządek ten nie został w trakcie współbieżnej realizacji zbioru transakcji naruszony /np. na skutek opóźnień w sieci komputerowej/.

Przedstawimy obecnie podstawową wersję algorytmu metody T/O [7, 8]. Z każdą daną $x \in D$ związane są dwie etykiety czasowe: etykieta czasowa odczytu R_ts(x) oraz etykieta czasowa zapisu W_ts(x). Etykiety te przyjmują wartość etykiety czasowej inicjowania transakcji, która jako ostatnia zrealizowała odczyt lub zapis tej danej. Procedury odczytu i zapisu dla uproszczonej podstawowej wer-

sji algorytmu metody T/O mają następującą postać:

```
procedure readreq (Ti,x);
      if  TS(Ti) < W_ts(x)
          then <wycofaj transakcję Ti i restartuj ją
                z nową etykietą czasową inicjowania>;
          else <zrealizuj żądanie odczytu> ;
               R_ts(x) := max{TS(Ti), R_ts(x)};
end;
procedure writereq (Ti, x);
      if TS(Ti) < R_ts(x) or TS(Ti) < W_ts(x)
         then <wycofaj transakcję Ti i restartuj ją
               ponownie z nową etykietą czasową inicjowania>;
         else <zrealizuj żądanie zapisu> ;
              W_ts(x) := TS(Ti);
end.
```

Zauważmy, że w metodzie porządkowania transakcji według etykiet czasowych a priori porządek realizacji transakcji prowadzi do stałego wycofywania i restartowania transakcji żądających dostępu do danych niezgodnie z porządkiem etykiet czasowych inicjowania. Sytuacja taka ma miejsce szczególnie często w przypadku transakcji generujących dynamicznie żądania dostępu do danych w trakcie swego wykonywania. Stąd podstawowym problemem metody T/O jest problem stałego restartowania transakcji oraz minimalizacji liczby restartów. W przypadku tzw.konserwatywnych algorytmów T/O [8, 36], żądanie dostępu do dowolnej lokalnej bazy danych jest zrealizowane jedynie wówczas, gdy istnieje pewność, że akceptacja żądania dostępu nie spowoduje restartu jakiejkolwiek transakcji w systemie. Wymaga to komunikacji pomiędzy stanowiskiem komputerowym, na którym realizowane jest żądanie dostępu, a wszystkimi pozostałymi. Konserwatywne algorytmy metody T/O oraz szczegółowe omówienie heurystyk minimalizujących koszt oraz praw-

dopodobieństwo restartów przedstawiono w pracy [7]. Hierarchiczna wersja algorytmu metody T/O została przedstawiona w pracy [16] (patrz również [23]). Jednakże przedstawiona hierarchiczna wersja algorytmu nadal nie rozwiązuje w sposób efektywny problemu zarządzania etykietami czasowymi danych. Problem opracowania bardziej efektywnej strategii zarządzania etykietami pozostaje nadal otwarty.

5.3. Metoda walidacji

Do grupy algorytmów opartych na metodzie walidacji można zaliczyć algorytmy zaproponowane w pracach [10, 11, 24, 67, 69].

Trzy zasadnicze cechy [7] wyróżniają podejście walidacyjne. Po pierwsze, podstawową opcją rozwiązania konfliktów dostępu jest wycofanie transakcji [7, 24, 67]. Po drugie, decyzja o akceptacji bądź wycofaniu transakcji podejmowana jest po wykonaniu transakcji. Po trzecie, zapytania do RBD nie są nigdy odrzucane. Podstawową przesłanką, która legła u podstaw konstrukcji algorytmów walidacyjnych, jest założenie, że konflikty dostępu pomiędzy transakcjami występują na tyle rzadko, że większość współbieżnych realizacji jest uszeregowalna i nie wymaga synchronizacji. W celu zagwarantowania uszeregowalności dowolnej realizacji wystarczy ograniczyć się zatem do wykonania procedury walidacyjnej, testującej czy w trakcie wykonywania transakcji nie wystąpiły konflikty dostępu mogące prowadzić do naruszenia spójności RBD. W przypadku gdy wynik testu jest negatywny transakcja jest wycofywana i restartowana. Przy powyższym, optymistycznym założeniu odnośnie prawdopodobieństwa wystąpienia konfliktów dostępu liczba restartów powinna być niewielka.

Istnieje wiele możliwych algorytmów metody walidacji. Jeden z nich polega na konstrukcji i testowaniu acykliczności grafu relacji $\rightarrow$ [7], Uaktualnienie grafu relacji $\rightarrow$ następuje każdorazowo w przypadku realizacji żądania odczytu lub zapisu danej w RBD.

Acykliczność grafu relacji $\longrightarrow$ jest, jak wiadomo z rozdziału 3, warunkiem dostatecznym uszeregowalności zbioru transakcji τ. Podstawowym problemem związanym z implementacją powyższego algorytmu w SRBD jest problem efektywnej konstrukcji i uaktualniania grafu relacji $\longrightarrow$.

Inne algorytmy metody walidacji zaproponowano w [11, 13, 24, 60]. Polegają one na wprowadzeniu dodatkowej fazy realizacji transakcji, a mianowicie, fazy walidacji. Omówimy to krótko na przykładzie algorytmu Kunga-Robinsona [67]. W algorytmie tym wykonywanie transakcji przebiega w trzech fazach: odczytu, walidacji oraz zapisu. W fazie odczytu transakcja wykonywana jest lokalnie, w obszarze roboczym lokalnej bazy danych. W momencie zakończenia wykonywania transakcji, rozumianego jako zakończenie wykonywania transakcji logicznej, następuje przejście do fazy drugiej - fazy walidacji. W fazie tej sprawdza się, czy zaakceptowanie transakcji nie naruszy spójności RBD, innymi słowy, czy po zaakceptowaniu transakcji wynikowa realizacja /uwzględniająca operację akceptowanej transakcji/ pozostanie uszeregowalna. Jeżeli realizacja pozostanie uszeregowalna to transakcja jest akceptowana i przechodzi do wykonywania fazy zapisu. W przeciwnym razie następuje wycofanie transakcji i jej ponowny restart.

Algorytm Kunga-Robinsona, podobnie jak inne algorytmy metody walidacji, wymaga, w celu walidacji transakcji, dysponowania pełną i aktualną informacją o globalnym stanie całego systemu. Informacja ta może być gromadzona na jednym wybranym stanowisku komputerowym SRBD lub na wszystkich stanowiskach. Można zatem stwierdzić, że efektywne rozwiązanie problemu walidacji, nie wymagające każdorazowo kosztownej komunikacji pomiędzy rozproszonymi stanowiskami SRBD, warunkuje przydatność metody walidacji w SRBD. Wniosek ten potwierdzają badania symulacyjne i analityczne dotyczące

oceny efektywności metod synchronizacji w SRBD [7, 10, 14, 15, 32, 57, 67].

Ponadto, istotnym problemem wymagającym rozwiązania w większości zaproponowanych algorytmów walidacyjnych jest problem wystąpienia konfliktu stałego restartowania. Wynika to z przyjęcia do rozwiązania konfliktów dostępu, metody wycofywania i ponownego restartowania transakcji. Problem ten może być rozwiązany metodą opisaną w rozdziale 4.

Algorytmy walidacyjne nie prowadzą natomiast do wystąpienia konfliktu cyklicznego restartu, który, jak wiadomo, w ogólności, może mieć miejsce w przypadku przyjęcia takiego rozwiązania konfliktów dostępu. Wynika to z faktu wprowadzenia jednoznacznych identyfikatorów liniowo porządkujących zbiór transakcji $\mathcal{T}$, wykorzystywanych następnie w fazie walidacji do rozwiązania konfliktów dostępu.

6. AKTUALNE TENDENCJE W ZAKRESIE ALGORYTMÓW SYNCHRONIZACJI W SRBD

Aktualnie, prace prowadzone w zakresie problematyki synchronizacji w systemach baz danych, zarówno scentralizowanych jak i rozproszonych, koncentrują się zasadniczo na trzech kierunkach badawczych:

- konstrukcji nowych algorytmów, stanowiących integrację podstawowych metod synchronizacji, z bezpośrednim włączenie do nich aspektu niezawodnościowego (m.in [2, 3, 9, 13, 21, 23, 83, 84]),
- konstrukcji algorytmów synchronizacji z uwzględnieniem semantyki danych (patrz [13, 28, 34, 70, 75, 76]),
- konstrukcji tzw.algorytmów wielowersyjnych [8, 27, 51, 61, 63, 64, 65, 67, 74, 77, 78].

Krótko scharakteryzujemy niektóre z tych tendencji na przykładzie wybranych algorytmów. Porównanie omówionych podstawowych metod synchronizacji pozwala stwierdzić, że wady i zalety tych metod są w dużym stopniu komplementarne. Stąd naturalne dążenie do integracji tych metod i opracowania metod hybrydowych [2, 3, 9, 13, 21, 23, 83, 84]. Przykładami algorytmu hybrydowego jest algorytm BOL łączący cechy metody blokowania i metody T/O.

Algorytm ten posiada wszystkie zalety algorytmu 2PL, rozwiązuje poprawnie wszystkie konflikty działania i ponadto dopuszcza: (i) współbieżność operacji zapisu, (ii) możliwość odczytu tzw. "starej" wersji danej, którą zablokowano uprzednio dla zapisu, oraz (iii) możliwość zdjęcia, przez niektóre transakcje, blokad danych założonych przez inne transakcje, a następnie ich późniejsze "odnowienie".

Idea algorytmu BOL jest następująca. Zauważmy, że w celu zachowania zalet algorytmu 2PL konieczne jest przyjęcie podstawowych mechanizmów blokowania oraz dwufazowość transakcji. Z drugiej strony efektywność algorytmów metody T/O wynika z możliwości współbieżnego przygotowywania wielu kopii danych poprzez dopuszczenie do współbieżnej realizacji operacji zapisu wstępnego, Powyższe spostrzeżenia stały u podstaw algorytmu BOL. W celu połączenia metody blokowania oraz metody T/O, oraz w celu rozwiązania konfliktów działania, przyjęto ideę podwójnego porządkowania zbioru transakcji τ. Pierwsze uporządkowanie o charakterze statycznym, zdefiniowane jest w oparciu o etykiety czasowe inicjowania transakcji. Uporządkowanie to wyznacza a priori priorytety transakcji i pozwala na rozwiązywanie konfliktów działania, które mogłyby wystąpić w trakcie realizacji fazy blokowania. W przeciwieństwie do klasycznego algorytmu 2PL, dopuszcza się w algorytmie kompatybilność blokad dla zapisu, dzięki czemu możliwe jest współbieżne

przygotowywanie kopii danych przez operacje zapisu wstępnego. Niekompatybilne pozostają jednak nadal blokady dla odczytu i zapisu. Jednakże, algorytm dopuszcza, gdy nie prowadzi to do naruszenia kryterium uszeregowalności, odczyt danej zablokowanej dla zapisu. Odczyt taki powoduje wywłaszczenie transakcji, która założyła blokady dla zapisu i konwersję blokady dla zapisu na blokadę dla odczytu. Jeżeli wywłaszczona transakcja żąda wykonania operacji zapisu właściwego, a transakcja wywłaszczająca /tj.realizująca aktualnie odczyt tej danej/ jeszcze się nie zakończyła, to wywłaszczona transakcja zostaje wycofana i restartowana z zachowaniem starej etykiety czasowej inicjowania. W przypadku, gdy wywłaszczająca transakcja się zakończy i blokada dla odczytu zostanie przez nią zdjęta, to wywłaszczona blokada dla zapisu zostaje odnowiona, a żądanie zapisu właściwego dla wywłaszczonej transakcji jest akceptowane. Pozwala to na zwiększenie stopnia współbieżności wykonywanych transakcji.

W momencie osiągnięcia przez transakcję punktu akceptacji, transakcja otrzymuje nową jednoznaczną etykietę czasową tzw. etykietę czasową akceptacji. Relacja porządku określona przez etykiety czasowe akceptacji na zbiorze transakcji zaakceptowanych, jak łatwo zauważyć, ma charakter dynamiczny. Porządek realizacji operacji zapisu właściwego dla zbioru transakcji jest zgodny z nowym porządkiem etykiet czasowych akceptacji. Dzięki temu, spójność bazy danych pozostaje zachowana mimo kompatybilności blokad dla zapisu. Porządek realizacji zbioru transakcji zgodny jest z porządkiem realizacji operacji zapisu właściwego. Zauważmy, że ze względu na zachowanie w algorytmie BOL podstawowego mechanizmu blokowania, koncepcja blokowania hierarchicznego, oparta na blokadach intencjonalnych, może być w pełni zachowana, co pozwala na łatwe skonstruowanie hierarchicznej wersji algorytmu BOL.

Włączenie semantyki danych (a ściślej obiektów) może mieć różnorodny charakter. Najprostszym przykładem uwzględnienia semantyki danych jest zwiększenie zbioru operacji elementarnych np. o operacje DEC i INR [9], których funkcją jest zwiększenie lub zmniejszenie wartości danej o 1. Dla szerokiej klasy zastosowań (np.wszelkiego rodzaju systemy rezerwacji) pozwala to istotnie zwiększyć efektywność działania systemu. Bardziej zaawansowane techniki polegają na definiowaniu danych, zbiorów danych oraz innych zasobów systemowych jako abstrakcyjnych typów danych. Dowolny abstrakcyjny typ danych posiada definicję zbioru dopuszczalnych operatorów (tzw.syntaktyka abstrakcyjnych typów danych) umożliwiających manipulację na obiekcie.

Na zakończenie, krótko wspomnimy o bardzo atrakcyjnym, zarówno z teoretycznego jak i praktycznego punktu widzenia, kierunku badań w zakresie synchronizacji, a mianowicie - algorytmach wielowersyjnych.

Idea wielowersyjnych algorytmów synchronizacji jest następująca:

> Transakcja żądająca dostępu do danej x w celu zapisu, nie uaktualnia stanu danej x, lecz tworzy nową wersję tej danej. Stąd, kolejna transakcja, żądająca dostępu do danej x w celu odczytu, ma możliwość odczytu jednej z dwu, aktualnie dostępnych wersji danych.

Intuicyjnie oczywisty jest fakt, że zwiększając liczbę wersji danej, dostępnych w systemie, zwiększamy tym samym stopień współbieżności realizacji transakcji. Powyższa obserwacja została formalnie udowodniona w postaci twierdzenia o nieskończonej hierarchii klas tzw.realizacji wielowersyjnych [63].

W literaturze znanych jest cały szereg algorytmów wielowersyjnych np. [8, 26, 27, 51, 61, 64, 67, 74, 77, 78, 84]. Szcze-

gółowe omówienie i przedstawienie problemów synchronizacji w systemach rozproszonych baz danych wraz z przedstawieniem podstawowych metod i algorytmów wielowersyjnych można znaleźć w pracy [88].

LITERATURA

1 Adiba M., Andrade J.M., Decitre P., Fernandez F., Nguen G.T., Polypheme: and experience in distributed database system design and implementation, w: C.Delobel, W.Litwin /red./, Distributed Data Bases, North Holland Pub.Co., 1980, 67-85.

2 Agraval R., Carey N.J., Livny M., Models for Studying Coucurrency Control Performance: Alternatives and Implications, Techn.Rep.AT T Bell Laboratories NJ 07901, 1984, p.25.

3 Agraval R., Dewitt D., Integrated Coucurrency Control and Recovery Mechanisms: Design and Performance Evaluation, Tech. Rep.No 497, Comp.Sc.Dept., University of Wiscousin, 1983.

4 Augustin R., Scholten H.A., Praedel U., Modelling Database Concurrency Control Algorithms Using a General Purpose Performance Evaluation Tool, Proc.of 10H Inl.Symp.Performance 84' Paris, 1984, pp.69-88.

5 Bayer R., Heller H., Reiser A., Parallelism and Recovery in DatabasesSystems, ACM TODS, Vol.5, 2,/June 1980/, pp.139-156.

6 Bernstein P.A., Shipman D.W., Wong W.S., Formal aspects of serializability in database concurrency control, IEEE Trans. on Soft.Eng. SE-5, 3, 1979, 203-216.

7 Bernstein P.A., Goodman N., Fundamental algorithms for concurrency control in distributed databases, Tech.Rep.CCA-80-05, CCA, 1980.

8 Bernstein P.A., Goodman N., Concurrency Control in Distributed Database Systems, ACH Comp.Surveys, Vol.13, No 2, 1981, pp. 185-221.

9 Bernstein P.A., Goodman N., Lai M.L., Analysing Concurrency Control Algorithms When User and System Operations Differ, IEEE Trans.on Soft.Eng.Vol.SE-9, 3 /May 1983/, 233-239.

10 Bhargava B., Performance Evaluation of the Optimistic Approach to Distributed Database Systems and Its Comparison to Locking, Proc.of 3rd Int.Conf.on Distributed Comp.Systems, Miami, Florida, 1982, pp.508-517.

11 Boksenbaum C., Cart M., Ferrie J., Pus J.F., Certification by Intervals of Timestamps in Distributed Database Systems, Proc.of 10th Int.Conf.on VLDB, Singapore, 1984, pp.377-387.

12 Bouchet P., Chesnais A., Feuvre J.M., Jomier G., Kurnickx A., PEPIN: an experimental multi-microcomputer data base management system, Proc.2nd Int.Conf.Distributed Computing Systems, Paris, IEEE Computer Society Press, 1981, 211-217.

13 Bragger R.P., Reimer M., Predicative Scheduling: Integration of Locking and Optimistic Methods, Tech.Rep.53, ETH, Institut fur Informatik, 1983.

14 Carey M.J., Stonebraker M.R., The Performance of Concurrency Control Algorithms For Database Management Systems, Proc.of 10th Conference on VLDB, Singapore, 1984, pp.107-118.

15 Carey M.J., Au Abstract Model of Database Concurrency Control Algorithms, Proc.of ACM-SIGMOD Conf.,San Jose, 1983,pp.97-107.

16 Carey M.J., Granularity Hierarchies in Concurrency Control, Proc.2nd ACM SIGACT-SIGMOD Symp.on Principles on Database Systems, Atlanta /March 1983/, 156-166.

17 Cellary W.: Rozdział zasobów w systemach komputerowych - próby podejścia globalnego. Wyd.Politechniki Poznańskiej, ser.Rozprawy nr 127, Poznań, 1981.

18 Cellary W.; Problem stałego zablokowania w systemach z nieprzywłaszczalnymi zasobami, Archiwum Automatyki i Telemechaniki 26, 4, 1981.

19 Cellary W., Admission test for deadlock avoidance in single resource systems. Computer Performance Vol.3, 4 /Dec.1982/, pp.240-246.

20 Cellary W., Morzy T., Bi-ordering approach to concurrency control in distributed database systems, /Przedłożone do druku/.

21 Cellary W., Morzy T., Locking with Preventation of Cyclic and Infinite Restarting in Distributed Database Systems, Proc.of 11th Int.Conf.VLDB, Stockholm, 1985, p.26.

22 Cellary W., Rozproszone bazy danych, Materiały II Jesiennej Szkoły PTI, Mrągowo, Listopad 1985.

23 Cellary W., Morzy T., Problemy i metody synchronizacji w systemach rozproszonych baz danych, /przyjęto do druku w Archiwum Automatyki i Telemechaniki/.

24 Ceri S., Owicki S., On the Use of Optimistic Methods for Concurrency Control in Distributed Databases, Proc.6th Berkeley Workshop on Distributed Data Management and Computer Networks, /Feb.1982/.

25 Ceri S., Pelagatti G., Distributed Databases: Principles and Systems, McGraw-Hill Book Company, 1984.

26 Chan A., Fox S., Lin W.T., Nori A., Ries D., The Implementation of an Integrated Concurrency Control and Recovery Scheme, Proc.of ACM-SIGMOD Conf., 1982, pp.184-191.

27 Chan A., Gray R., Implementing Distributed Read-Only Transactions, IEEE Trans.on Soft.Eng., Vol.SE-11, No 2, 1985, pp. 205-212.

28 Codron R., Garcia-Molina M,, The Performance of a Concurrency Control Mechanism That Exploits Semantic Knowledge, Tech.Rep. 324, University of Princeton, Sept.1984.

29 Date C.J., An Introduction to Database Systems Volume II, Addison-Wesley Pub.Co., 1982.

30 Eswaran K.P., Gray J.N., Lorie R.A., Traiger I.L., The notion of consistency and predicate locks in a database system, Comm.of ACM 19, 11. 1976, 624-633.

31 Franaszek P., Robinson J., Limitations of Concurrency in Transaction Processing, Res.Rep.IBM RC 10151, 1983.

32 Galler B.I., Concurrency control performance issues, Ph.D. thesis, Techn.Rep.CSRG-147, Toronto, 1982.

33 Garcia-Molina H., Performance of update algorithms for replicated data in a distributed databases, Ph.D.thesis, Tech. Rep.STAN-CS-79-744, Stanford, 1979.

34 Garcia-Molina H., Using Semantic Knowledge for Transaction in a Distributed Database, ACM Trans.on Database Systems, Vol.8, No 2, 1983, pp.186-213.

35 Gardarin G., Integrity, consistency, concurrency, reliability in distributed database management systems, w: C.Delobel, W.Litwin /red./, Distributed Data Bases, North Holland Pub. Co., 1980, 335-351.

36 Gifford D.K., Donahue J.E., Coordinating Independent Atomic Actions, Proc.of Compcon'85, 1985, 10p.

37 Gligor V.D., Shatluck S.H., On deadlock detection in distributed systems, IEEE Trans.on Soft.Eng.SE-6, 5 /1980/, 435-440.

38 Gray J., Notes on data base operating systems, w: R.Bayer, R.M.Graham, G.Seegmuller /eds./, Operating Systems: An Advance Course, Springer Verlag, 1978, 393-481.

39 Gray J., Transaction concept: virtues and limitations, Proc. 7th Int.Conf.on VLDB, Cannes, IEEE Computer Society Press, 1981, 145-154.

40 Herman D., Verjus J.P., An algorithm for maintaining the consistency of multiple copies, Proc.1st Int.Conf.Distributed Computing Systems, Huntsville, 1979.

41 Ho G.S., Ramamoorthy C.V., Protocols for Deadlock Detection in Distributed Database Systems, IEEE Trans.on Soft. Eng.Vol.SE-8, 6, /Nov.1982/, 554-557.

42 Holler E., Multiple Copy Update, in: Laupson /ed./ Distributed Systems Architecture and Implementation, Lecture Notes in Comp.Sc., Vol.105, Springer UNC 1981.

43 Isloor S.S., Marsland T.A., The deadlock problem: an overview, IEEE Computer 9, 1980, 58-78.

44 Jagannathan J.R., Vasuderan R., Comments on Protocols for Deadlock Detection in Distributed Database Systems, IEEE Trans.on Soft.Eng.SE-9, 3, /May 1983/, p.371.

45 Kelter U., On the Inadequacy of Serializability, Tech.Rep. No 172, Universitat Dortmund, 1983.

46 Kohler W.H., A survey of techniques for synchronization and recovery in decentralized computer systems, Computing Surveys 13, 2, 1981, 149-183.

47 Kohler W.H., Wilner K.C., Stankovic J.A., An Experimental Comparison of Locking Policies in a Testbed Database System, Proc.of ACM-SIGMOD Conf., San Jose, 1983, pp.108-119.

48 Korth H.F., A deadlock-free, variable granularity locking protocol, Tech.Rep.TR 268, Princeton University, 1980.

49 Kung H. ., Papadimitriou C.H., An optimality theory of concurrency control for databases, Proc.ACM-SIGMOD Int.Conf. Management of Data, 1979, 116-126.

50 Lamport L., Time, clocks and the orderings of events in a distributed systems. Comm.of ACM 21, 7, 1978, 558-565.

51 Lausen G., Formal Aspects of Optimistic Concurrency Control in a Multiple Version Database System, Information Systems, Vol.8, No 4, 1983, pp.291-301.

52 Lin W.K., Performance evaluation of two concurrency control mechanisms in a distributed database system, Proc. ACM - SIGMOD Int.Conf.Management of Data, 1981, 84-92.

53 Lin W.K., Nolte J., Communication Delay and Two Phose Locking, Proc.of 3rd Int.Conf.on Distributed Comp.Systems, Miami, 1982, pp.502-507.

54 Lynch N., Multilevel atomicity - a new correctness criterion for database concurrency control, Tech.Rep.MIT/LCS/TR-281, MIT, /August 1982/.

55 Madelaine J., Evaluation d'algorithms de controle de concurrence, INRIA Res.Rep.164, 1983.

56 Menasce D.A., Muntz R.R., Locking and deadlock detection in distributed data bases, IEEE Trans.on Soft.Eng., Vol.SE-5, 3, 1979, 195-201.

57 Morris R.J.T., Wong W.S., Performance of Concurrency Control Algorithms with Nonexclusive Access, Proc.of 10th Int.Symp. "Performance 84", Paris, 1984, pp.87-101.

58 Morzy T., Ordered-transaction approach to performance evaluation of concurrency control algorithms for distributed database systems, w: J.Akoka /red./, Management of Distributed Data Processing, North Holland Pub.Co., 1982, 253-269.

59 Morzy T., Algorytmy synchronizacji w systemach rozproszonych baz danych, Praca doktorska, Poznań, 1983.

60 Moss J.E., Nested Transactions: Au Approach to Reliable Distributed Computing, Rep.TR-260, MIT, 1981.

61 Muro S., Kameda T., Monoura T., Multi-version Concurrency Control Scheme for a Database System, Journal of Computer and System Sciences, Vol.29, 1984, pp.207-224.

62 Papadimitriou C.H., Serializability of concurrent database updates, Journal of ACM 26, 4, 1979, 631-653.

63 Papadimitriou C.H., Kanellakis P.C., On Concurrency Control by Multiple Versious, ACM Trans.on Database Systems, Vol.9, No 1, 1984, pp.89-99.

64 Reed D.P., Naming and Synchronization in a Decentralized Computer System, Ph.D.Thesis, MIF, 1978.

65 Reed D., Svobodowa L., SWALLOW - a distributed data starage system for a local network, w: Proc.Int.Workshop on Local Networks, Zurich, 1980.

66 Ries D., The effect of concurrency control on database management system performance, Ph.D.thesis, Computer Sc.Dep., University of California, 1979.

67 Robinson J.T., Design of concurrency control for transaction processing systems. Ph.D.thesis, Rep.CMU-CS-82-114, Cornegie-Mellon University, Pittsburgh, 1982.

68 Rosenkrantz D.J., Stearns R.E., Lewis P.M., System level concurrency control for distributed database systems, ACM Trans.Database Systems 3, 2, 1978, 178-198.

69 Schlageter G., Optimistic methods for concurrency control in distributed database systems, w: C.Zaniolo, C.Delobel /red./, Proc.7th Int.Conf.VLDB, Cannes, IEEE Computer Society Press, 1981, 125-130.

70 Schwarz P.M., Spector A.Z., Synchronizing Shared Abstract Types, Rep.CMU-CS-83-151, Cornegie-Mellon University, Pittsburgh, 1983.

71 Shum A.W., Spirakis P.G., Performance analysis of concurrency control methods in database systems, w: F.J.Kylstra /red./, Comp.Performance Modelling and Evaluation, North, Holland Pub.Co., 1981, pp.1-20.

72 Silberschatz A., Kedem Z., Consistency in hierarchical database systems, Journal of ACM 27, 1, 1980, 72-80.

73 Silberschatz A., Kedem Z., A Family of Locking Protocols for Database Systems that are modelled by directed graphs, IEEE Trans.on Soft.Eng.Vol.SE-8, 6 /Nov.1982/.

74 Sinha M.K., Timepad: A Performance Improving Synchronisation Mechanism for Distributed Systems, MIT Res.Rep. MIT/LCS/TM-177, 1980.

75 Sha L., Jensen D.E., Rashid R.F., Northentt J.D., Distributed Co-operating Processes and Transactions, z: Synchronization, Control and Communication in Distributed Computing Systems, ed.Y.Parker, Academic Press, 1983.

76 Spector A.Z., Schwarz P.M., Transactions: A Construct for Reliable Distributed Computing, Operating Systems Review, vol.17, No 2, 1983, pp.18-35.

77 Stearns R.E., Rosenkrantz D.J., Distributed Database Concurrency Controls Using Before-Values, Proc.of ACM-SIGMOD Conf., 1981, pp.74-83.

78 Takagi A., Concurrent and reliable updates of distributed databases, MIT Res.Rep.MIT/LCS/TR-144, Laboratory for Comp. Sc.MIT, 1979.

79 Tay Y., A Mean Value Performance Model for Locking in Databases, Ph.D.Thesis, Comp.Sc.Dept.Harvard University, 1984.

80 Thomas R.H., A majority concensus approach to concurrency control for multiple copy databases, ACM Trans.Database Systems 4,2, 1979, 180-209.

81 Traiger I.L., et al.Transactions and Consistency in Distributed Database Systems, ACM Trans.on Database Systems, vol.7, No 3, 1982, pp.323-342.

82 Ullman J.D., Principles of database systems, Computer Society Press, Pitman, 1980.

83 Vidyasankar K., Raghavan V.V., Highly Flexible Integration of the Locking and the Optimistic Approaches of Concurrency Control, Tech.Rep.8402, University of Regina, Canada, March 1984.

84 Viemont Y.H., Gardarin G., A distributed concurrency control based on transaction commit ordering, Proc.Fault Tolerant Comp.Symp., Los Angeles, 1982.

85 Yannakakis M., Papadimitriou C.H., Kung H.T., Locking Policies: Safety and Freedom from Deadlocks, Proc.20th IEEE Symp.on the Found of Comp.Sc., 1979, 286-297.

86 Yannakakis M., Issues of correctness in database concurrency control by locking, Proc.13th Int.Conf.ACM-STOC, 1981, 363-367.

87 Yannakakis M., A Theory of Safe Locking Policies in Database Systems, Journal of ACM, Vol.29, 3 /July 1982/, 718-740.

88 Morzy T., Wielowersyjne algorytmy synchronizacji w systemach rozproszonych baz danych, Materiały Konferencji "Architektura systemów rozproszonych" DSA'85, Bierutowice, październik 85.

OPTYMALIZACJA TRANSAKCJI W ROZPROSZONYCH BAZACH DANYCH

Zbyszko Królikowski
Instytut Automatyki Politechniki Poznańskiej
ul. Piotrowo 3A, 60-965 Poznań

1. Wstęp

W ostatnich latach obserwuje się gwałtowny rozwój systemów rozproszonych baz danych (SRBD), i to zarówno w aspekcie intensyfikacji prac badawczych z tego zakresu jak i zastosowań.

Osiągnięcie potencjalnych korzyści płynących z realizacji SRBD uwarunkowane jest rozwiązaniem szeregu jakościowo nowych, trudnych problemów związanych z zarządzaniem takimi systemami. Jednym z najistotniejszych i najtrudniejszych problemów stojących przed projektantem systemu zarządzania rozproszoną bazą danych (SZRBD) jest problem optymalizacji planów wykonywania transakcji. Znaczenie tej problematyki wynika z konieczności zapewnienia efektywnego wykonywania transakcji co decyduje o użytkowości całego systemu, a co jest szczególnie utrudnione w systemach rozproszonych baz danych, ze względu na wymaganie zarządzania rozproszonego z jednej strony oraz konieczność transmisji i przetwarzania dużych woluminów danych - z drugiej.Doskonałą ilustracją istotnego znaczenia problematyki optymalizacji planów wykonywania transakcji w SRBD jest przykład przedstawiony w [DATE 83]. W przykładzie tym, C.J. Date analizując kilka różnych planów wykonania prostej transakcji, pokazał jak dramatyczny spadek efektywności systemu może mieć miejsce w przypadku źle skonstruowanego planu wykonania transakcji.

Najogólniej problem optymalizacji planów wykonywania transakcji można sformułować następująco. Dla danej transakcji, wymagającej wykonania szeregu operacji na danych zlokalizowanych na różnych stanowiskach komputerowych systemu rozproszonej bazy danych, należy określić sekwencję wykonywania tych operacji oraz ich lokalizację tak aby optymalizować wybrane kryterium oceny działania systemu.

Dotychczas opracowano cały szereg algorytmów generowania planów wykonywania transakcji w systemach rozproszonych baz danych [APER 83, ASTR 80, BER 81d, BLAC 81, CELL 80, CHEN 84, CHIU 84, CHU 82, CHUN 84, DANI 82,DAYA 82, EPST 78, GOLD 84, GOUD 81, GRIF 79, HEVN 85, KAMB 82, KAMB 84, KERS 82, LUK 83, PERR 84, SELI 80, TOAN 81, VARD 84, WONG 77, YOSH 84, YU 80, YU 83].

Przedstawimy obecnie podstawowe założenia dotychczasowych algorytmów generowania planów wykonywania transakcji w SRBD.

2. Podstawowe pojęcia i założenia

Wspólną cechą opracowanych dotychczas algorytmów generowania planów wykonywania transakcji jest bardzo uproszczony model kosztów wykonywania transakcji w SRBD. W modelu tym uwzględnia się jedynie koszt transmisji danych, pomijając koszty ich przetwarzania na stanowiskach komputerowych, w szczególności koszt przygotowania transmisji, wykonywania operacji relacyjnych, dekompozycji transakcji, generowania planów rozproszonego wykonywania transakcji oraz koszty systemowe związane z obsługą synchronizacji transakcji, zabezpieczeniem przed upadkiem systemu itp. Przyjęcie takiego modelu kosztów jest równoznaczne z założeniem, że wąskim gardłem systemu rozproszonej bazy danych jest transmisja danych.

Odnośnie modelu transmisji zakłada się, że:

(i) koszt transmisji jest niezależny od odległości pomiędzy stanowiskami systemu oraz obciążenia sieci komunikacyjnej,

(ii) koszt transmisji jest liniową funkcją rozmiaru woluminu przesyłanych danych, tj. $c(x) = ax + b$, gdzie b jest kosztem związanym z przygotowaniem sesji transmisyjnej, a x jest rozmiarem woluminu przesłanych danych.

Założenie dotyczące liniowości funkcji kosztów transmisji potwierdzają, wyniki pomiarów [BOUC 83, ROLI 82, SARF 81] przeprowadzonych w kilku SRBD. Z przedstawionych w [ROLI 82] wykresów, zależności kosztów transmisji od rozmiarów przesyłanych danych wynika, że po dokonaniu aproksymacji i uśrednienia kosztów dla stanowiska ekspediującego dane i stanowiska komputerowego SRBD przyjmującego dane można przyjąć, że koszt transmisji jest liniową funkcją rozmiaru przesyłanych danych.

W problematyce optymalizacji wykonywania transakcji w SRBD przyjmuje się następującą definicję rozproszonej bazy danych.

Rozproszoną bazą danych (RBD) nazywamy czwórkę (ST, DL, DF, Loc), gdzie:

$ST = \{ST_1, \ldots, ST_n\}$ jest zbiorem stanowisk komputerowych,

$DL = \{S_1 S_2, \ldots, S_m\}$ jest zbiorem danych logicznych, nazywanym logiczną bazą danych,

$DF = \{R_1, R_2, \ldots, R_m\}$ jest zbiorem danych fizycznych, nazywanym fizyczną bazą danych.

$Loc : DF \rightarrow ST$ jest funkcją określającą lokalizację danej fizycznej $R_i \in DF$ w zbiorze ST.

Logiczna baza danych DL reprezentuje RBD z punktu widzenia użytkownika, który w terminach logicznej bazy danych definiuje swoje zadanie nazywane transakcją. Użytkownik pozostaje nieświadomy faktu fizycznego rozproszenia danych, do

których żąda dostępu, a zatem nie rozróżnia rozproszonej bazy danych od scentralizowanej. Logiczna baza danych jest widziana przez użytkownika, jako relacyjny model danych.

Zakłada się, że relacja R_i odpowiadająca schematowi S_i jest niepodzielna w tym sensie, że jest zlokalizowana na pojedyńczym stanowisku Loc (R_i). Innymi słowy, zakładamy, że relacja stanowi jednostkę rozproszenia w RBD. Powyższe założenie jest ogólnie przyjęte w literaturze [APER 83, BER 81d, BLAC 81 CHEN 84, CHIU 84, CHU 82, CHUN 84, EPST 78, HEVN 79, KAMB 82, KAMB 84, KERS 82, LUK 83, PERR 84, SELI 80, WONG 77, YOSH 84, YO 80, YO 83]. Wynika to z następującej obserwacji. Dzieląc relację R_i na rozłączne fragmenty $R_{i1}, R_{i2}, \ldots, R_{im_i}$, pamiętane na m_i - różnych stanowiskach komputerowych, uzyskujemy nowy stan RBD, dla którego DF' $=\{ R_1, \ldots, R_{i-1}, R_{i+1}, \ldots, R_n,$

$$R_{n+1}, R_{n+2}, \ldots, R_{n+m_1} \} .$$

Fragmentowi R_{imj}, gdzie $1 \leq m_j \leq m_i$, odpowiada w DF' relacja R_{n+mj}. Zadanie użytkownika żądające dostępu do relacji R_i, w bazie danych DF, można obecnie zdefiniować, w terminach bazy danych DF', jako zadanie żądające jednoczesnego dostępu do realizacji $R_{n+1}, R_{n+2}, \ldots, R_{n+mi}$.

Powszechnie przyjmuje się następujące założenia dotyczące relacji:

- jednorodność rozkładu wartości atrybutów, tzn. wszystkie wartości jakiegokolwiek atrybutu pojawiają się z równą częstotliwością,
- niezależność atrybutów, tzn. prawdopodobieństwo $P(V_1 \& V_2)$, że atrybut 1 ma wartość V_1 i atrybut 2 ma wartość V_2 w tej samej krotce relacji, jest równe prawdopodobieństwu $P_1(V_1)$, że atrybut 1 ma wartość V_1, razy prawdopodobieństwo $P_2(V_2)$, że atrybut 2 przybiera wartość V_2, tzn. $P(V_1 \& V_2) = P_1(V_1) * P_2(V_2)$.

Założenia te były przedmiotem szczegółowej analizy przeprowadzonej w cyklu prac Christodoulakisa (m.in. [CHR 84a, CHR 84b]).

Jak wspomnieliśmy uprzednio transakcja jest elementarną jednostką interakcji użytkownika z SRBD. Koncepcja transakcji jako podstawowej jednostki programowej w systemie rozproszonej bazy danych znalazła szerokie uznanie i akceptację wśród projektantów systemów baz danych [GRAY 81, SCHW 84, SPEC 83]. Koncepcja ta pozwala bowiem rozłącznie rozpatrywać problem dostępu do bazy danych pojedyńczego użytkownika oraz problemy: synchronizacji współbieżnego dostępu wielu użytkowników i zagwarantowania odpowiedniego stopnia niezawodności systemu [MOSS 81, ALLC 83, SPEC 83].

Przejdziemy obecnie do szczegółowego przedstawienia modelu transakcji przyjętego w piśmiennictwie.

Zakładamy, że każda transakcja T zawiera dwa składniki:

- kwalifikację transakcji (ang. qualification) Q(T), będącą koniunkcją operacji połączenia postaci $R_i \cdot A = R_j \cdot B$, tj.

$$Q(T) = \{(R_i \cdot A_k = R_j \cdot A_l) \wedge (R_k \cdot A_m = R_l \cdot A_n) \wedge \ldots \wedge$$

$$\wedge (R_m \cdot A_p = R_n \cdot A_r)\},$$

gdzie pary atrybutów: (A_k, A_l), (A_m, A_n),..., (A_p, A_r)
są zdefiniowane na wspólnych domenach,

- specyfikację relacji wynikowej (ang. target list) T(T)- zawiera listę atrybutów, których wartości spełniające kwalifikację transakcji Q(T), są przesyłane na stanowisko wynikowe.

Zgodnie z obowiązującymi w piśmiennictwie tendencjami [APER 83, BER 81d, BLAC 81, CHEN 84, CHIU 84, GOLD 84, GOUD 81, HEVN 79, HUAN 85, KAMB 82, KAMB 84, LUK 83, PERR 84, WONG 77, YOSH 84, YO 80, YU 83], zakładać będziemy dalej, że kwalifikacja transakcji Q(T) spełnia następujący warunek:

$R_k, R_l \in Q(T)$, Loc $(R_k) \neq$ Loc (R_l) dla k,l = i,j,..., m,n

Dla dowolnej transakcji T można skonstruować, na podstawie jej kwalifikacji tzw. graf transakcji G(T). Zbiór wierzchołków grafu G(T) odpowiada zbiorowi relacji ujętych w Q(T), natomiast krawędzie grafu G(T) wskazują operacje połączenia na zbiorze wspólnych atrybutów połączeniowych tych relacji. W zależności od struktury grafu transakcji, wyróżnia się następujące typy transakcji: drzewowe, łańcuchowe, cykliczne i gwiaździste. Wyszczególnione wyżej klasy transakcji, można również podzielić w zależności od liczby atrybutów połączeniowych, występujących w poszczególnych elementach Q(T).
Wyróżniamy wówczas:

- transakcje proste - każda operacja połączenia w Q(T) jest wykonywana na jednym atrybucie połączeniowym,
- transakcja złożona - operacje połączenia w Q(T)mogą być wykonywane na więcej niż jednym atrybucie połączeniowym.

Oprócz wymienionych wyżej klas transakcji, jest używane również pojęcie: transakcja ogólna. Pod tym pojęciem rozumiemy transakcję, której graf G(T)ma dowolną postać i nie ma żadnych ograniczeń co do liczby atrybutów połączeniowych operacji połączenia ujętych w Q(T).

Każdej transakcji T odpowiada niepusty zbiór możliwych algorytmów jej wykonania w SRBD, które nazywamy planami wykonywania transakcji. Plany wykonywania transakcji różnią się między sobą lokalizacją i kolejnością operacji wykonywanych na poszczególnych stanowiskach komputerowych oraz kierunkiem i kolejnością przesyłania danych.
Wybór określonego planu wykonywania transakcji dokonywany jest przez procedury optymalizacyjne SZRBD, na podstawie przyjętych kryteriów oceny działania SRBD.

Przyjęto dwa kryteria oceny działania systemu rozproszonej bazy danych: czas odpowiedzi oraz sumaryczne obciążenie systemu. W definicjach obu tych kryteriów używa się pojęcia kosztu, przy czym zakłada się, że koszt może być wyrażony w jednostkach czasu lub liczbie instrukcji kodu maszynowego oprogramowania SZRBD.

Zauważmy, że tę ostatnią jednostkę, można łatwo przeliczyć na jednoski czasu. Tak więc, pojęcie kosztu jest synonimem słowa czas i nie rozważa się innych aspektów oceny działania systemu rozproszonej bazy danych, jak na przykład: ocena z punktu widzenia kosztów finansowych, ponoszonych przez użytkowników systemu.

Wspomniane wyżej kryteria oceny działania SRBD definiowane są w następujący sposób:

Czas odpowiedzi (ang. response time) - jest sumą kosztów od momentu startu pierwszej operacji wykonywanej w ramach realizacji transakcji, do momentu zakończenia ostatniej operacji transakcji. W przypadku równoległego wykonywania pewnych operacji (np. równoległa transmisja danych), do kosztów czasu odpowiedzi wlicza się koszt tej spośród operacji równoległych, która jest "najdroższa";

Sumaryczne obciążenie systemu (ang. total time) - jest sumą kosztów wszystkich operacji realizowanych w ramach wykonywania transakcji - współbieżność operacji nie jest uwzględniana.

Kryterium czasu odpowiedzi reprezentuje punkt widzenia użytkownika, natomiast kryterium sumarycznego obciążenia systemu - punkt widzenia administratora SZRBD.

Podstawowy schemat wykonywania transakcji w SRBD ma następującą postać [APER 83, BER 81d, BLAC 81, CHEN 84, CHIU 84, HEVN 79, JAR 84b, KAMB 82, LUK 83, PERR 84, WONG 77, YOSH 84, YO 80]:

- wykonaj operacje, które wymagają tylko danych lokalnych,
- wykonaj wstępne przetwarzanie, w celu redukcji relacji przetwarzanych w następnym kroku,
- wykonaj operacje, które wymagają danych z więcej niż jednego stanowiska komputerowego (operacje binarne),
- wykonaj operacje końcowe na stanowisku wynikowym, zgodnie ze specyfikacją relacji wynikowej transakcji, w celu odpowiedniej prezentacji wyniku.

Wykazano, [CHU 82, SMIT 75, UUM 80], że powyższy schemat wykonywania transakcji, zgodnie z którym w pierwszej kolejności wykonane są operacje unarne, a następnie operacje binarne minimalizuje koszty wykonywania transakcji. Wykazano ponadto [BER 81a, BLAC 81, BITT 83, CERI 84, CHU 82, JAR 84a, JAR 84b, LUK 81, LUK 83, ULLM 80], że koszt wykonywania transakcji, zasadniczo zdeterminowany jest przez koszt operacji binarnych, a w szczególności operacji połączenia.

W środowisku systemów rozproszonych baz danych rozwiązaniem alternatywnym w stosunku do operacji połączenia jest operacja pół-połączenia (semi-join). Definicja tej operacji jest następująca:

Operacją pół-połączenia, relacji R_i z relacją R_k, nazywamy połączenie relacji R_i z wynikiem operacji projekcji relacji R_k, na wspólnych atrybutach połączeniowych obu relacji, tzn.

$$R_i \ltimes R_k = \{r_i \in R_i \mid (\exists\, r_k \in R_k)\ (r_i\,[R_i \cap R_k] = r_k\,[R_i \cap R_k])\}$$

Na oznaczenie operacji półpołączenia, używa się symbolu $\ltimes$ oraz dodatkowo tam, gdzie jest konieczne wskazanie kierunku transmisji i atrybutu połączeniowego, np. A, - symbolu $\xrightarrow[A]{\ltimes}$.

Jak wynika z definicji, operacja półpołączenia w przeciwieństwie do operacji połączenia jest asymetryczna, tzn. $R_i \xrightarrow{\ltimes} R_k \neq R_k \xrightarrow{\ltimes} R_i$. Operacja półpołączenia jest kompozycją operacji relacyjnych: projekcji i połączenia. Po wykonaniu operacji projekcji relacji R_k, według atrybutu połączeniowego, jej wynik tzn. wszystkie unikalne (bez duplikatów) wartości atrybutu połączeniowego są łączone z relacją R_i. Rozmiar relacji R_i zostaje w ten sposób zredukowany tylko do tych krotek, które biorą bezpośredni udział w relacji wynikowej a połączenia obu relacji, tj. R_i i R_k.

Operacja półpołączenia jako alternatywa w stosunku do operacji połączenia, jest preferowana z dwóch powodów [BER 81d]:

1° Operacja ta posiada silne właściwości redukcyjne ($R_i \ltimes R_k \subseteq R_k$). Dla kontrastu operacja połączenia może powodować wzrost rozmiaru bazy danych, w najgorszym przypadku:

$$\text{size}\,(R_i \bowtie R_k) = \text{size}\,(R_i) * \text{size}\,(R_k)$$

2° Operacja półpołączenia może być wykonywana przy mniejszych kosztach transmisji danych niż operacja połączenia. Do wykonania operacji półpołączenia $R_i \ltimes R_k$, konieczna jest jedynie transmisja wyniku operacji projekcji na relacji R_k, natomiast do wykonania operacji połączenia $R_i \bowtie R_k$ konieczna jest transmisja całej relacji R_i lub R_k.

Oczywiście operacja półpołączenia może, w niektórych sytuacjach dawać mniejszy efekt niż operacja połączenia, ponieważ operacja $R_i \ltimes R_k$ redukuje tylko relację R_i, a operacja $R_i \bowtie R_k$ może redukować obie relacje R_i i R_k. Jednakże, w takim przypadku pojedyńcza operacja półpołączenia, może być zastąpiona dwiema operacjami półpołączenia, zazwyczaj przy koszcie niższym niż operacji połączenia (pokazano to w [BER 81d]).

Przedstawimy obecnie, skrótowo podstawowe wyniki uzyskane w badaniach nad efektywnością operacji półpołączenia, dla wybranych klas transmisji [BEER 81, BEER 83, BER 81a, Ber 81b, BER 81c, FAG 83a, FAG 83b, FAGI 84, GOO 82a, GOO 82b, GOO 82c, GOO 83a, GOO 83b, GOOD 84].

W szeregu prac [BER 81a, BER 81b, BER 81c, GOO 82a] przeprowadzono dokładną analizę możliwości redukcyjnych operacji półpołączenia dla transakcji drzewowych i cyklicznych. Wnioski wynikające z tych badań (opublikowane w fundamentalnej pracy z tego zakresu [BER 81a]) można zreasumować w następujący sposób: wykorzystując sekwencję operacji półpołączenia dla realizacji transakcji drzewowej, można osiągnąć stan pełnej redukcji relacji, na których wykonywana jest ta transakcja.

Stan pełnej redukcji, rozumiemy intuicyjnie jako ograniczenie rozmiarów łączonych relacji, tylko do tych krotek, które są niezbędne dla realizacji operacji połączenia. Innymi słowy wszystkie te krotki, które nie znajdą się w relacji wynikowej są eliminowane przez operację półpołączenia. Warto zaznaczyć, że powyższy wniosek dotyczy zarówno transakcji prostych jak i złożonych [BER 81b, GOOD 84].

Dla klasy transakcji cyklicznydh problem redukcji rozmiarów relacji przez sekwencję operacji półpołączenia jest znacznie trudniejszy i należy do klasy problemów silnie NP -zupełnych [BEER 81, BER 81a, BER 81b, GOO 82c].

Dla transakcji cyklicznych otrzymanie stanu pełnej redukcji (jeżeli w ogóle jest możliwe otrzymanie takiego stanu), wiąże się z poniesieniem znacznie większych kosztów, niż w przypadku transakcji drzewowych.

W konsekwencji wspomnianego wcześniej założenia o dominacji kosztów transmisji w ogólnych kosztach wykonywania transakcji w SRBD, zdecydowana większość opracowanych dotychczas algorytmów generowania planów wykonywania transakcji, koncentruje się wyłącznie na redukcji woluminów danych przesyłanych pomiędzy stanowiskami systemu rozproszonej bazy danych i wykorzystuje w tym celu strategię operacji półpołączenia.

Dokonamy obecnie przeglądu opracowanych do tej pory algorytmów generowania planów wykonywania transakcji w SRBD.

3. Algorytmy generowania planów wykonywania transakcji w SRBD - Stan wiedzy

Jak wspomnieliśmy, problem opracowania efektywnego algorytmu generowania planów wykonywania transakcji jest jednym z najtrudniejszych i najbardziej złożonych problemów, stojących przed projektantem SZRBD. Złożoność tego problemu wynika przede wszystkim z dwóch faktów:

- możliwości równoległego wykonywania transakcji w SRBD,
- konieczności transmisji danych pomiędzy stanowiskami komputerowymi SRBD.

Problem generowania planów wykonywania transakcji należy do klasy problemów silnie NP-zupełnych [APER 83, CERI 84, HUAN 85, GOO 82b, LUK 83, YU 82]. Nawet w przypadku przyjęcia omówionych wcześniej silnych założeń upraszczających, algorytmy generowania planów wykonywania transakcji, z natury rzeczy, są algorytmami heurystycznymi. Znane w literaturze algorytmy są trudno porównywalne, ze względu na: po pierwsze - różnorodność przyjmowanych założeń odnośnie modelu RBD i sieci komunikacyjnych oraz po drugie, ze względu na założenia dotyczące informacji dostępnej w procesie generowania planu, a dotyczącej: stanu RBD i aktualnego stanu wykonywanej transakcji. Czynnik drugi, tj. zakres oraz dokładność dostępnej informacji o stanie bazy danych w istotny sposób determinuje efektywność danego algorytmu generowania planów wykonywania transakcji w SRBD.

Algorytmy generowania planów wykonywania transakcji można podzielić na dwie klasy: algorytmy dynamiczne, dla których plan wykonywania transakcji generowany jest w trakcie wykonywania transakcji oraz algorytmy statyczne, dla których plan generowany jest w trakcie kompilacji transakcji. Pomimo potencjalnych zalet algorytmów dynamicznych związanych z możliwością wykorzystania precyzyjnych informacji o stanie bazy danych i stanie wykonywania transakcji, wysoki koszt implementacji tych algorytmów w SRBD, wynikający z konieczności zbierania i przesyłania tych informacji, czyni je nieprzydatnymi w chwili obecnej. Dotychczas opracowano tylko kilka algorytmów dynamicznych [TOAN 81, BALD 79].

Zdecydowana większość znanych z piśmiennictwa algorytmów należy do klasy algorytmów statycznych i algorytmami z tej klasy będziemy się zajmować w dalszym ciągu. Opracowane do tej pory algorytmy ograniczają obszar poszukiwania planów wykonywania transakcji, przez przyjęcie założeń upraszczających. Uzasadnienie takiego podejścia podano w [BLAC 81], wykazując, że przykładowo, dla stosunkowo prostego przypadku transakcji wymagającej wykonania operacji połączenia 3 relacji na dwóch atrybutach połączeniowych, wykorzystując technikę operacji półpołączenia dla znalezienia optymalnego planu wykonywania tej transakcji należałoby przeszukać około 1.3 miliarda unikalnych sekwencji operacji półpołączenia, co byłoby rozwiązaniem zdecydowania nieefektywnym.

Algorytmami, które mają fundamentalne znaczenie w problematyce optymalizacji planów wykonywania transakcji w SRBD są:
algorytm stosowany w systemie rozproszonej bazy danych SDD-1, który jest produktem Computer Corporation of America [BER 81a], algorytm Apears'a, Hevnera i Yao - algorytm AHY [APER 83] oraz jego odmiana zaproponowana przez Perrizo - algorytm - S [PERR 84]. Algorytmy te dotyczą klasy transakcji ogólnych, co wydatnie nobilituje ich wartość w porównaniu z innymi algorytmami znanymi w piśmiennictwa [BLAC 81, CHEN 84, CHIU 84, CHEU 82, CHUN 84, GOUD 81, KAMB 82,

KERS 82, LUK 83, YOSH 84, YU 80, YU 83], które dotyczą szczególnych klas transakcji i które w wielu przypadkach są przystosowaniem do specyfiki szczególnej klasy transakcji jednego z algorytmów SDD-1 lub AHY.

Algorytm SDD-1

W algorytmie SDD-1, wykorzystano wyniki prac E. Wonga, dokonując modyfikacji i rozszerzenia zaproponowanego w [WONG 77] algorytmu generowania planów wykonywania transakcji w SRBD. Użytkownicy systemu rozproszonej bazy SDD-1, kontaktują się z systemem za pomocą proceduralnego języka wysokiego poziomu Datalanquage. Transakcja w systemie SDD-1 jest realizowana w następujących krokach [BER 81a]:

1° Przekształcenie transakcji T zapisanej w Datalanquage do postaci transakcji wyrażonej w algebrze relacyjnej, przy jednoczesnym określeniu lokalizacji relacji, wyszczególnionych w Q(T) na stanowiskach komputerowych SRBD.

2° Konstrukcja programu P, stanowiącego sekwencję operacji półpołączenia oraz wybór stanowiska komputerowego, na którym po wykonaniu programu P, znajdują się dane o największym rozmiarze.

3° Przesłanie danych wymaganych do wykonania transakcji na stanowisko wybrane w kroku 2°; wykonanie operacji końcowych na tym stanowisku komputerowym i przesłanie ostatecznego wyniku na stanowisko wynikowe.

Kluczowym punktem tego schematu jest punkt 2°, tj. konstrukcja programu P. Dla wspomagania wyboru odpowiedniego wariantu programu P, szacuje się koszty, efekty i zyski dla poszczególnych operacji półpołączenia, tworzących dany wariant.

Koszt operacji półpołączenia $R_i \overset{\ltimes}{X} R_j$, jest zdefiniowany jako rozmiar danych (size) przesyłanych pomiędzy stanowiskami komputerowymi SRBD: ST_i i ST_j dla wykonania tej operacji, tzn. jest to: size $(R_i \cdot X)$.

Efekt wykonania operacji półpołączenia $R_i \overset{\ltimes}{X} R_j$ jest rozumiany jako rozmiar relacji R_j, po wykonaniu tej operacji.

Zysk wykonania operacji półpołączenia $R_i \overset{\ltimes}{X} R_j$, jest rozumiany jako rozmiar danych wyeliminowanych z bazy danych, tzn.: size (R_j) przed $\ltimes$ $-(R_j)$ po $\ltimes$

Do oszacowania efektów operacji półpołączenia wykorzystuje się teorię współczynnika selektywności, przy założeniu, że atrybuty niepołączeniowe są redukowane według aproksymacji funkcji Yao [YAO 77].

Algorytm SDD-1 można przedstawić w następujący sposób.

1° Wykonaj wszystkie operacje lokalne.

2° Dla zbioru wszystkich możliwych operacji półpołączenia dla danej transakcji T, wykonaj następujące kroki:

(a) - oszacuj koszty, efekty i zyski operacji półpołączenia,

(b) - wybierz operację półpołączenia, która daje największy zysk,

(c) - uaktualnij dane potrzebne do oszacowania kosztów, efektów

i zysków operacji półpołączenia w następnej iteracji,

(d) - ze zbioru badanych operacji, wyeliminuj operację wybraną w punkcie (b),

(e) - powtarzaj od kroku (a) aż zyski ze wszystkich operacji półpołączenia będą równe zeru lub zbiór badanych operacji półpołączenia będzie zbiorem pustym.

3° Wybierz stanowisko komputerowe, na którym znajdują się dane o największym rozmiarze i na to stanowisko prześlij dane wyszczególnione w specyfikacji relacji wynikowej; wykonaj operacje końcowe i prześlij ostateczny wynik na stanowisko wynikowe.

Algorytm AHY

Podstawową ideą algorytmu AHY jest dekompozycja transakcji złożonej do zbioru podtransakcji prostych, których plany wykonywania, jak wykazano w [APER 83, HEVN 79] można łatwo optymalizować. Algorytm AHY polega na generowaniu planów przetwarzania każdej relacji osobno, przy czym przez plan przetwarzania relacji rozumiemy sekwencję operacji półpołączenia użytych do redukcji tej relacji. Plan wykonywania transakcji, zawiera plany dla wszystkich relacji, które są wyszczególnione w kwalifikacji transakcji. Algorytm AHY ma następującą postać [APER 83]:

1° Wykonaj wszystkie operacje lokalne,

2° Wygeneruj kandydujące plany dla poszczególnych relacji - wyizoluj każdy atrybut, na którym wykonywana jest operacja połączenia i rozważ każdy z nich dla zdefiniowania podtransakcji prostych z nieokreślonym stanowiskiem wynikowym;

- w celu zminimalizowania czasu odpowiedzi, zastosuj procedurę PARALLEL, dla każdej podtransakcji prostej, zachowaj wszystkie kandydujące plany, dla ich integracji w kroku 3°;
- w celu zminimalizowania sumarycznego obciążenia systemu, zastosuj procedurę SERIAL dla każdej podtransakcji prostej;
- z uzyskanych planów dla każdej podtransakcji prostej utwórz kandydujące plany dla każdego atrybutu, na którym wykonywane jest połączenie.

3° Dokonaj integracji kandydujących planów; dla każdej relacji wyszczególnionej w kwalifikacji, kandydujące plany są integrowane do formy planu przetwarzania tej relacji. W zależności od kryterium optymalizacji, integracja jest wykonywana przez procedury RESPONSE oraz TOTAL lub COLLECTIVE.

4° Usuń plany nadmiarowe tzn. te plany przetwarzania relacji, których transmisje są uwzględnione w planach dla innej relacji.

Ze względu na obszerność i wysoką złożoność - uniemożliwiającą skrótowe przedstawienie - poszczególnych procedur algorytmu AHY, ograniczymy się dalej do omówienia tylko procedury RESPONSE, dla której, autorzy algorytmu AHY wykazali

optymalność generowanych planów wykonywania transakcji. Do twierdzenia o optymalności algorytmu AHY (RESPONSE) i specyficznych założeń metody AHY, ustosunkowano się w [CEL 85a, LAKH 84].

Wykorzystywana w kroku 3^{o} algorytmu AHY, procedura RESPONSE, przedstawia się następująco:

Procedura RESPONSE

1. Uporządkowanie kandydujących planów.
   Dla każdej relacji uporządkuj według rosnących czasów kandydujące plany dla atrybutu połączeniowego oznaczanego d_{ij}, $j = 1,2,\ldots,NAP_i$ jest liczbą atrybutów połączeniowych relacji - i.
2. Integracja planów.
   Z kandydujących planów, uporządkowanych według rosnących czasów, skonstruuj zintegrowane plany dla relacji R_i, które zawierają równoległe transmisje wszystkich uporządkowanych kandydujących planów. Wybierz plan z minimalnym czasem odpowiedzi.

W celu minimalizacji czasu odpowiedzi i sumarycznego obciążenia systemu kandydujących planów algorytmu AHY wykorzystuje odpowiednio procedury PARALLEL i SERIAL [HEVN 79]. Procedury te mają następującą postać:

Procedura PARALLEL

1. Uporządkuj relacje R_i, wyszczególnione w kwalifikacji transakcji, tak aby size $(R_1) \leq$ size $(R_2) \leq \ldots \leq$ size (R_n).
2. Rozważ każdą z relacji R_i, w kolejności ich rosnących rozmiarów. Dla każdej relacji R_i, akonstruuj plan, który zawiera równoległe transmisje planów dla relacji R_j, gdzie $j < i$; wybierz plan z minimalnym czasem odpowiedzi.

Procedura SERIAL

1. Uporządkuj relacje R_i, wyszczególnione w kwalifikacji transakcji tak, aby size $(R_1) \leq$ size $(R_2) \leq \ldots \leq$ size (R_n).
2. Jeśli na stanowisku wynikowym nie ma relacji, wówczas wybierz strategię;
   $R_1 \rightarrow R_2 \rightarrow \ldots \rightarrow R_n \rightarrow$ stanowisko wynikowe.
3. Jeśli R_r jest relacją na stanowisku wynikowym, wówczas spośród dwóch strategii:
   $R_1 \rightarrow R_2 \rightarrow \ldots \rightarrow R_r \rightarrow \ldots \rightarrow R_n \rightarrow R_r$ lub
   $R_1 \rightarrow R_2 \rightarrow \ldots \rightarrow R_{r-1} \rightarrow R_{r+1} \rightarrow \ldots R_n \rightarrow R_r$
   wybierz tę, która daje minimum sumarycznego obciążenia systemu.

Algorytm - S

Przedstawiony w [PERR 84] algorytm - S jest odmianą algorytmu AHY. Istotną zmianą w stosunku do algorytmu AHY jest odrzucenie założenia o nie-

redukowalności atrybutów połączeniowych. Algorytm - S do konstrukcji planu dla danej relacji wykorzystuje operacje półpołączenia wykonywane zarówno na atrybutach połączeniowych i niepołączeniowych. Szacowanie rozmiarów redukcji relacji i atrybutów, po wykonaniu operacji półpołączenia, w algorytmie - S jest dokonywane na podstawie rozwiniętej teorii współczynnika selektywności [PERR 84].

Z takich samych względów jak w przypadku algorytmu AHY, ograniczymy się tutaj do przedstawienia algorytmu - S w wersji RESPONSE. Algorytm - S przedstawia się następująco.

1. Konstrukcja kandydujących planów dla poszczególnych atrybutów połączeniowych.
   - wykonaj sortowanie atrybutów połączeniowych według ich rosnących rozmiarów,
   - dla wszystkich kolejnych atrybutów połączeniowych wykonaj następujące czynności:
     - weź kolejny (pierwszy) atrybut,
     - dla tego atrybutu skonstruuj plany, w których uwzględnij plan wybrane dla atrybutów poprzedzających go biorąc pod uwagę równoległe i szeregowe wykonywanie operacji półpołączenia,
     - wybierz plan z minimalnym czasem odpowiedzi,
     - wykonaj sortowanie skonstruowanych planów według rosnących czasów odpowiedzi.
2. Dla każdej relacji R_i wyszczególnionej w kwalifikacji danej transakcji, skonstruuj zintegrowane plany przetwarzania, w których uwzględnij równoległe wykonywanie skonstruowanych i uporządkowanych w kroku 1. planów kandydujących dla atrybutów.
3. Ze skonstruowanych w ten sposób planów przetwarzania dla relacji, usuń operacje nadmiarowe.

Jak widać, idea algorytmu - S jest identyczna jak algorytmu AHY. Jednakże, wprowadzenie do konstrukcji planów operacji półpołączenia na atrybutach niepołączeniowych oraz wielokrotne sortowanie planów według czasu odpowiedzi, umożliwiło uzyskanie wyników lepszych nawet o 50%, w stosunku do planów generowanych przez algorytm AHY.

Oprócz omówionych powyżej metod, w piśmiennictwie znanych jest szereg innych algorytmów generowania planów wykonywania transakcji w SRBD. Jednakże algorytmy te mają znacznie mniejsze znaczenie dla omawianej problematyki, ponieważ albo dotyczą tylko wybranych klas transakcji, albo są odmianą algorytmów SDD-1 i AHY, lub zakładają pewne, dodatkowe uproszczenia w środowisku SRBD. Tak więc, algorytmy te mają znacznie niższe wartości aplikacyjne niż omówione wcześniej metody SDD-1, AHY i S.

Spośród algorytmów, które są odmianami metod SDD-1 i AHY, na szczególną uwagę zasługuje algorytm zaproponowany w [BLAC 81, LUK 83]. Jest on rozwiązaniem pośrednim pomiędzy starą wersją algorytmu SDD-1 z 1979 roku, a algorytmem - G przedstawionym przez Hevnera i Yao w [HEVN 79]. Istotnym novum, w stosunku do algorytmów SDD-1 i AHY jest to, że wybór operacji półpołączenia, które mają być włączane do programu redukcyjnego, jest wykonywany na podstawie dwóch warunków:

1° Koszt operacji półpołączenia jest minimalny,

2° Redukowana relacja, po wykonaniu operacji półpołączenia ma najmniejszą liczność atrybutu połączeniowego.

Dla klasy transakcji prostych, znalezienie operacji półpołączenia spełnającej jednocześnie oba powyższe kryteria, jest możliwe. Jednakże, nie jest to możliwe dla transakcji złożonych.

Black i Luk proponują wówczas formułę pośrednią pomiędzy kryteriami 1° i 2°, tzn. podstawą wyboru operacji półpołączenia jest wówczas minimum sumy składników występujących w kryteriach 1° i 2°. Na podstawie badań symulacyjnych [BLAC 81] stwierdzono, dla klasy transakcji prostych, przewagę tego algorytmu nad wersją algorytmu SDD-1 z 1979 roku i algorytmu G-Hevnera i Yao również z 1979 roku [HEVN 79]. Jednakże, dla transakcji o wysokiej złożoności algorytm Black'a i luk'a staje się nieefektywny. W odniesieniu do nowych wersji algorytmów SDD-1 i AHY, badań porównawczych nie przeprowadzono.

W [CHEU 82] zaproponowano algorytm - D, który jest odmianą algorytmu AHY. W algorytmie tym odrzucono założenie o nieredukowalności atrybutów niepołączeniowych. Algorytm ten ma w chwili obecnej, znaczenie już tylko historyczne, bowiem Perrizo w [PERR 84], zaproponował ulepszenie i rozwinięcie algorytmu - D.

W piśmiennictwie zaproponowano również cały szereg algorytmów generowania planów wykonywania wybranych klas transakcji. Algorytmy te nie są zdolne do rozwiązywania transakcji ogólnych.

W [KAMB 82] zaproponowano algorytm generowania planów, dla klasy transakcji cyklicznych, w [CHEN 84, KERS 82] dla klasy transakcji gwiaździstych, w [CHIU 84] dla klasy transakcji łańcuchowych a w [YU 82] dla klasy transakcji drzewowych. Wszystkie te algorytmy, w poszukiwaniu optymalnego planu wykonywania danego typu transakcji, stosują technikę zbliżoną do zaproponowanej w algorytmie SDD-1.

Szereg prac dotyczy optymalizacji planów wykonywania transakcji w SRBD, w których wykorzystuje się sieci komunikacyjne o specyficznej konfiguracji, albo zakłada się dodatkowe uproszczenia w SRBD.

W [CHU 82] zaproponowano algorytm generowania planów wykonywania transakcji, minimalizujący koszt transmisji i przetwarzania w funkcji lokalizacji stanowisk komputerowych i sekwencji wykonywanych operacji relacyjnych. Dana transakcja jest dekomponowana do sekwencji operacji relacyjnych. Bazując na powiązaniach pomiędzy tymi operacjami oraz na ich przechodności, generuje się dla danej transakcji wiele wstępnych równoważnych drzew transakcji. Następnie na podstawie wyprowadzonych w [CHU 82] twierdzeń ogranicza się obszar poszukiwań rozwiązania optymalnego.

Podstawową wadą tej metody jest jej niewielomianowa złożoność obliczeniowa, co czyni tę metodę nieefektywną obliczeniowo i bardzo trudną w implementacji. W modelu kosztów, wykorzystywanym w tej metodzie, zakłada się między innymi znajomość tzw. funkcji redukcyjnych danych. Funkcje te, są trudne do określenia dla poszczególnych kroków planu wykonywania transakcji, a uzyskiwanie ich wartości na drodze pomiarów symulacyjnych może być powodem nieprzewidzianych błędów i utraty optymalności. W algorytmie tym zakłada się, że na stanowiskach komputerowych SRBD znajdują się komputery tego samego typu oraz identyczne systemy zarządzania lokalną bazą danych. Ograniczenie stosowalności tego algorytmu tylko dla klasy homogenicznych SRBD, w świetle obowiązujących tendencji projektowania heterogenicznych SRBD, degraduje wartość tego algorytmu.

W [EPST 78] zaprezentowano algorytm generowania planów wykonywania transakcji, stosowany w rozproszonej wersji systemu INGRES. Jest ona rozwinięciem idei dekompozycji zapytań przedstawionej w [WONG 76]. Dla wyboru planu wykonywania transakcji stosuje się kryterium minimalizacji transmisji w sieci komunikacyjnej, tzn. dokonuje się wyboru stanowisk komputerowych na których ma być wykonane przetwarzanie, tak by przesyłane woluminy danych były jak najmniejsze.

W systemie R [ASTR 80, DANI 82, GRIF 79, SELI 80] problem optymalizacji planów wykonywania transakcji został sprowadzony do wyboru najtańszego algorytmu wykonywania operacji połączenia.

W [KERS 82] zaproponowano algorytm generowania planów wykonywania transakcji w homogenicznych systemach rozproszonych baz danych - typu "gwiaździstego", a w [DAYA 82] dla SRBD typu CODASYL.

Wspomnijmy na zakończenie, o pewnych algorytmach w których odrzucono część powszechnie akceptowanych założeń. W [YU 83, VARD 84] zaproponowano pewne algorytmy generowania planów wykonywania transakcji w SRBD, przy założeniu, że istnieją redundantne kopie relacji na różnych stanowiskach komputerowych SRBD oraz przy założeniu, że relacje mogą być podzielone na mniejsze fragmenty. Prace te mają jednak charakter wstępny i jak na razie nie mają większego znaczenia praktycznego.

4. Problematyka optymalizacji wykonywania transakcji w SRBD - nowe podejście

Postęp technologiczny w dziedzinie sieci komputerowych oraz tendencje rozwojowe SRBD prowadzą do odwrócenia proporcji pomiędzy kosztami transmisji a kosztami przetwarzania transakcji. Przypomnijmy, że dotychczas zakładano zdecydowaną dominaję czasów transmisji nad czasami przetwarzania.
Podejście takie było uzasadnione w rozproszonych baz danych starszego typu, wykorzystujących sieci komputerowe o małej szybkości transmisji i duże komputery w roli stanowisk SRBD. Wówczas bowiem mamy rzeczywiście do czynienia z silną dominacją czasów transmisji nad czasami przetwarzania. Jednakże postęp technologiczny odwrócił proporcje tych czasów. Z jednej strony obserwujemy bowiem gwałtowny wzrost szybkości transmisji w sieciach komputerowych, przy czym przyrost ten jest znacznie szybszy niż przyrost mocy obliczeniowej komputerów. Z drugiej strony natomiast, obserwuje się silną tendencję do stosowania małych komputerów (mini i mikrokomputerów) w roli stanowisk SRBD. Oznacza to relatywne zmniejszenie mocy obliczeniowej stanowisk w stosunku do możliwości transmisji sieci, a więc innymi słowy, wzrost kosztów przetwarzania w stosunku do kosztów transmisji. Wniosek ten uzasadnia również szczegółowa analiza pomiarów przeprowadzonych w eksperymentalnych i komercyjnych SRBD [BOUC 83, BITT 83, ROLI 82, SARF 81]. Na jej podstawie, przy uwzględnieniu wspomnianych powyżej tendencji rozwojowych można stwierdzić, że wąskim gardłem SRBD są stanowiska komputerowe, których obciążenie sięga 95%, podczas gdy obciążenie sieci komunikacyjnych sięga do 30%. Wymienione powyżej czynniki sprawiają, że konieczne jest nowe podejście do problemu optymalizacji planów wykonywania transakcji w SRBD.

Próbę takiego nowego podejścia przedstawiono w [KRO 85e], rozważając problem optymalizacji planów wykonywania transakcji w SRBD, zdefiniowany w sposób bardziej ogólny, niż było to dotychczas przyjęte w literaturze. Rozszerzenie zakresu problemu dotyczy kosztów wykonywania transakcji.

W [CEL 85b, KRO 85a, KRO 85c] przedstawiono nowy model kosztów wykonywania transakcji, który dostosowany jest do nowoczesnych SRBD.

W modelu tym uwzględniono następujące trzy grupy kosztów:

1° transmisji danych, pomiędzy stanowiskami komputerowymi SRBD;

2° przetwarzania - koszty wykonywania operacji relacyjnych, w tym koszty dostępu do pamięci zewnętrznej;

3° systemowe - koszty generowania i wykonywania akcji lokalnych, gdzie przez akcję lokalną rozumiemy tutaj operację nadzoru rozproszonego wykonywania transakcji, koszty organizacji synchronizacji transakcji, koszty zabezpieczeń przed upadkiem systemu oraz koszty tworzenia tymczasowych plików danych.

Jak widać w modelu uwzględniono wszyskie istotne składniki kosztów, biorąc jednocześnie pod uwagę rzeczywiste, potwierdzone empirycznie relacje pomiędzy nimi.

Jak wiadomo, koszty wykonywania transakcji w SRBD są zdominowane przez koszty wykonywania operacji połączenia. Jak wspomniano wcześniej operacja połączenia może być wykonywana zgodnie z jedną z dwóch strategii: jako połączenie naturalne lub jako sekwencja operacji półpołączenia. Wcześniej, wskazaliśmy również na zmianę, w stosunku do czynionych do tej pory założeń, proporcji pomiędzy poszczególnymi składnikami kosztu wykonywania transakcji w SRBD. Zmiana taka ma silny wpływ na efektywność poszczególrych strategii wykonywania operacji połączenia. Zachodzi więc potrzeba dokonania kosztów wykonywania transakcji, stosujących jedną lub drugą strategię.

W [CEL 85b, KRO 85b, KRO 85e] dokonano takiego porównania. Wykonano analizę efektywności strategii połączenia i półpołączenia w funkcji rozmiarów relacji i współczynników selektywności atrybutów połączeniowych.

Wykazano, że dla danego współczynnika selektywności, istnieje pewna wartość progowa rozmiaru łączonych relacji, powyżej której strategia półpołączenia jest bardziej efektywna od strategii połączenia naturalnego. Poniżej tej wartości progowej koszty strategii półpołączenia przewyższają koszty strategii połączenia. Uwagi powyższe dotyczą obu kryteriów, zarówno czasu odpowiedzi jak i sumarycznego obciążenia systemu; przy czym, dla kryterium sumarycznego obciążenia systemu, wartość progowa dla danej wartości współczynnika SF, występuje dla większych rozmiarów relacji, niż dla kryterium czasu odpowiedzi. Na rys. 4.1. przedstawiono procentowy rozkład kosztów wykonywania operacji półpołączenia i połączenia, dla kilku wybranych rozmiarów łączonych relacji, dla wartości współczynnika selektywności atrybutu połączeniowego równej 0.1, a więc dla przypadku szczególnie korzystnego dla strategii półpołączenia.

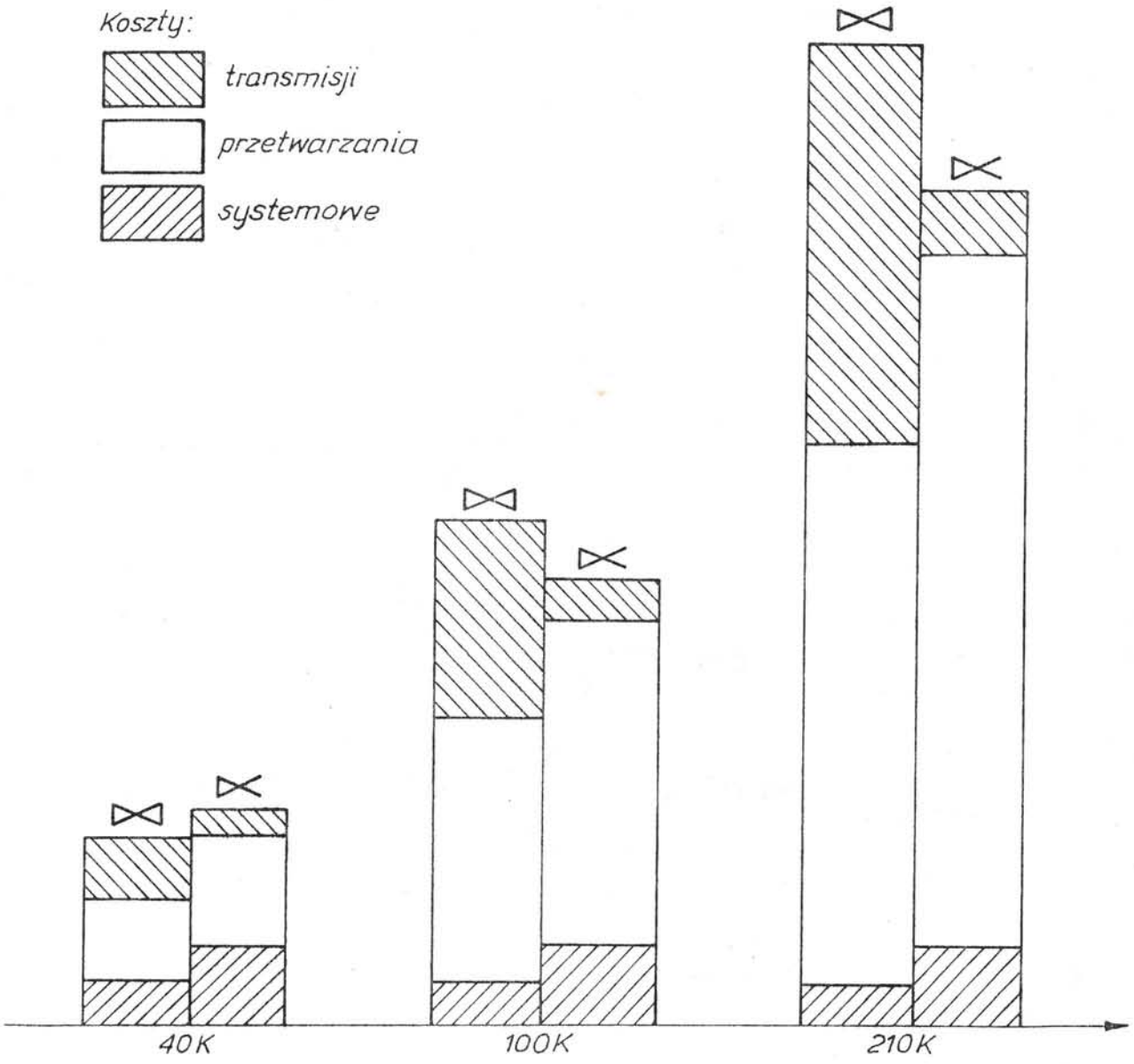
Z przedstawionych na Rys. 4.1. diagramów wynika, że strategia półpołączenia prowadzi do oczekiwanej redukcji kosztów transmisji danych. Jednakże prowadzi również do dodatkowego obciążenia stanowisk komputerowych kosztami systemowymi i kosztami przetwarzania.

Dokładna analiza porównawcza strategii połączenia i półpołączenia, wykazała, że:

(i) procentowy udział kosztów transmisji danych w kosztach wykonywania transakcji, osiąga około 30% (patrz - Rys. 4.1.). Strategia półpołączenia redukuje koszty transmisji danych, jednakże prowadzi równocześnie do wzrostu kosztów przetwarzania i kosztów systemowych,

(ii) koszty przetwarzania transakcji silnie zależną do takich parametrów jak: rozmiary łączonych relacji i współczynniki selektywności ich atrybutów połączeniowych.

Generalny wniosek, nasuwający się na podstawie przeprowadzonych w [CEL 85b, KRO 85d, KRO 85e] rozważań jest następujący: dotychczasowe metody generowania planów wykonywania transakcji w SRBD, konstruowane w oparciu o operację pół-



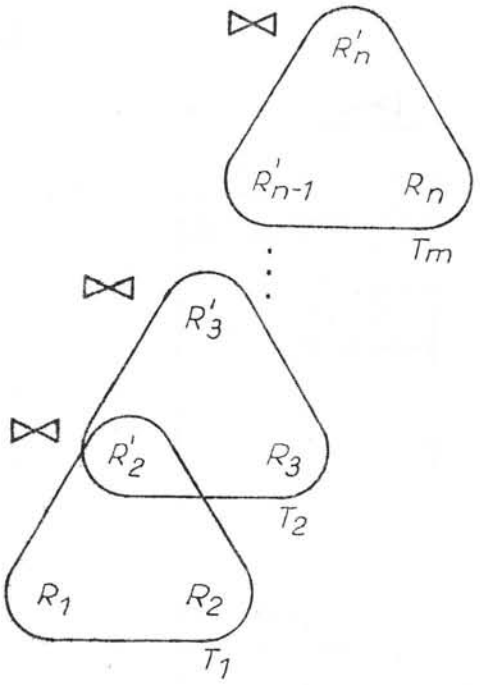
Rys. 4.1. Procentowy rozkład kosztów wykonywania strategii półpołączenia i połączenia, dla wartości SF = 0.1 (dla kryterium czasu odpowiedzi)

połączenia, w pewnych sytuacjach mogą okazać się nieefektywne, a co więcej mogą powodować wzrost kosztów wykonywania transakcji.

Wyniki omówionej powyżej analizy były podstawą opracowania nowego algorytmu generowania planów wykonywania transakcji w SRBD, nazwanego Algorytmem - Z [KRO 85e] . Podstawową ideą tego algorytmu jest stosowanie zmiennej strategii wykonywania operacji połączenia.

Ogólna idea Algorytmu - Z przedstawia się następująco. Najpierw dekomponuje się transakcję T, do zbioru podtransakcji $\{T_1, T_2, \ldots, T_m\}$, gdzie $m = n - 1$, a n jest liczbą relacji wyszczególnionych w kwalifikacji transakcji T. Podtransakcja T_i dotyczy pary relacji (R_j, R_k). Dekompozycja transakcji T jest wykonana przez procedury: SEKWENCJA i ROWNOLEGŁOSC, które generują zbiory podtransakcji dalej wykonywalnych sekwencyjnie i równolegle.

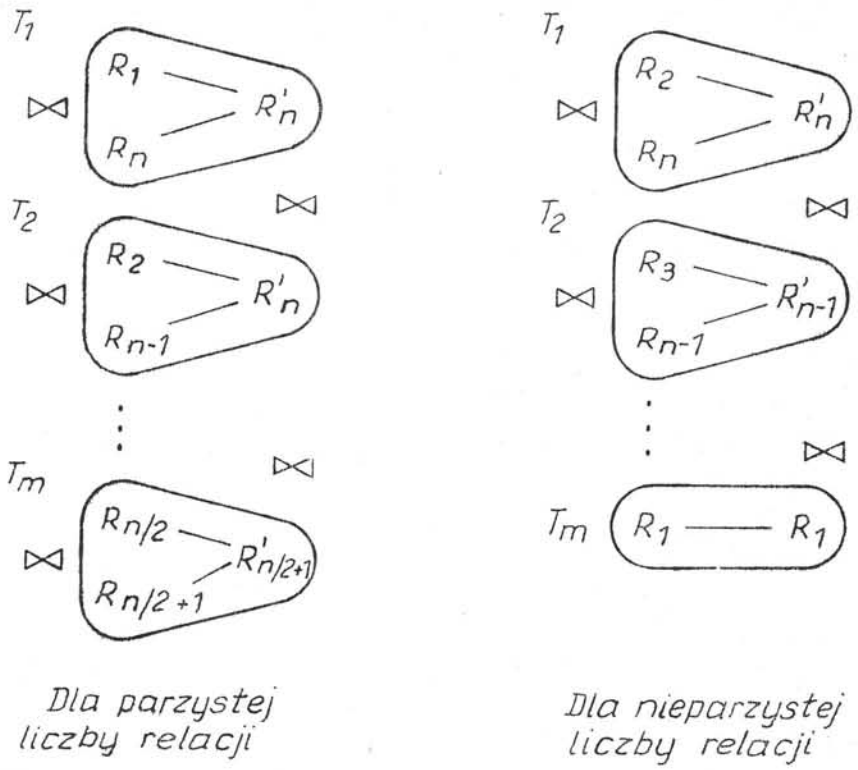
Dekompozycja transakcji T, przez procedurę SEKWENCJA jest wykonywana w następujący sposób. Najpierw dokonuje się uporządkowania relacji według ich rosnących rozmiarów, tzn. size $(R_i) \leq$ size (R_{i+1}). W przypadku SEKWENCJA, podtransakcja T_i dotyczy pary relacji (R_i, R_{i+1}) (patrz - Rys. 4.2.). Kwalifikacja podtransakcji T_i zawiera operację połączenia dwóch relacji, tj. $R_i \bowtie R_{i+1}$.



Rys. 4.2. Dekompozycja transakcji T, przez procedurę SEKWENCJA

Jak widać, w procedurze SEKWENCJA zakłada się, że kolejne podtransakcje T_i są wykonywane sekwencyjnie.

W procedurze ROWNOLEGŁOSC, podobnie jak w przypadku procedury SEKWENCJA , najpierw dokonuje się uporządkowania relacji według ich rosnących rozmiarów, tzn. size $(R_i) \leq$ size (R_{i+1}). Kolejne wygenerowane przez procedurę ROWNOLEGŁOSC , podtransakcje T_i dotyczą odpowiednio, przy parzystej i nieparzystej liczbie relacji, następujących par relacji: (R_1, R_n), $(R_2, R_{n-1}), \ldots,$ $(R_{n/2}, R_{n/2+1})$ oraz (R_2, R_n), $(R_3, R_{n-1}), \ldots, (R_1)$ (patrz Rys. 4.3.).



Rys. 4.3. Dekompozycja transakcji T, przez procedurę ROWNOLEGŁOSC

W procedurze ROWNOLEGŁOSC , zakłada się, że podtransakcje T_i są wykonywane równolegle. Relacje otrzymane w wyniku realizacji podtransakcji T_i są przesyłane na stanowisko wynikowe i tam realizowana jest operacja połączenia końcowego. Spośród podtransakcji wygenerowanych przez procedury SEKWENCJA i ROWNOLEGŁOSC dla każdej podtransakcji T_i określony jest zbiór możliwych strategii operacji półpołączenia (tzn. dla każdej podtransakcji T_i generowany jest zbiór: $2 * AP_i$ strategii operacji półpołączenia, gdzie AP_i - jest liczbą atrybutów połączeniowych pary relacji podtransakcji T_i). Koszty tych operacji są obliczane według modelu kosztów przedstawionego w [KRO 85e]. Koszty strategii operacji półpołączenia są porównywane z kosztem strategii operacji połączenia naturalnego relacji, rozważanych dla danej podtransakcji T_i. W przypadku gdy dla danych rozmiarów relacji i ich współczynników selektywności efektywniejsza jest strategia operacji półpołączenia, do wykonania jest wybierana operacja charakteryzująca się minimalnym kosztem. Natomiast, w przypadku przeciwnym wybierana jest następująca strategia operacji połączenia:
$R_j \xrightarrow{\bowtie} R_k$ (size $(R_i) \leq$ size (R_k)).

Czynności te są wykonywane dla każdej podtransakcji ze zbiorów wygenerowanych przez procedury SEKWENCJA i ROWNOLEGŁOSC a następnie, po określeniu strategii wykonywania danej podtransakcji uaktualniane są dane dotyczące stanu bazy danych. Do ostatecznego wykonania wybierany jest ten spośród planów wygenerowanych dla danej transakcji T, który dla danego kryterium optymalizacji, charakteryzuje się minimalnym kosztem.

Algorytm-Z w porównaniu z dotychczasowymi algorytmami generowania planów wykonywania transakcji w SRBD, można scharakteryzować następująco:

- algorytm-Z jest algorytmem uniwersalnym, ponieważ może generować plany dla transakcji ogólnych,
- algorytm-Z uwzględnia znacznie większą liczbę przypadków jakie mogą wystąpić w trakcie realizacji transakcji, niż algorytmy dotychczasowe,
- jest on algorytmem o niskiej złożoności obliczeniowej oraz jest stosunkowo prosty w implementacji,
- algorytm-Z zapewnia wysoką precyzję szacowania rozmiarów realizacji wynikowych, po wykonaniu operacji połączenia i półpołączenia. Uzyskano to dzięki wykorzystaniu wyników najnowszych badań z tego zakresu,
- jest algorytmem statycznym, jednakże w prosty sposób może być przekształcony w algorytm dynamiczny. Zmiana taka wymagałyby zastąpienia fragmentu algorytmu, w którym szacuje się parametry wynikowe RBD, po wykonaniu danego kroku algorytmu przez procedurę uaktualniającą te parametry na podstawie informacji otrzymanych od SZRBD.

W [KRO 85e, KRO 85f] dokonano oceny efektywności zaproponowanego algorytmu. Głównym celem tej oceny było porównanie efektywności Algorytmu - Z z algorytmami SDD-1, AHY, S. Wybór tych algorytmów jako algorytmów odniesienia jest wynikiem szczególnego znaczenia tych algorytmów w problematyce optymalizacji planów wykonywania transakcji w SRBD.

Przeprowadzony eksperyment obliczeniowy dowiódł, że zastosowanie algorytmu ze zmienną strategią wykonywania operacji połączenia dla SRBD implementowanych na szybkich sieciach lokalnych z mikrokomputerami w roli stanowisk komputerowych prowadzi do znacznej poprawy efektywności działania tych systemów, z punktu widzenia obu kryteriów oceny ich działania.

Reasumując - dotychczas uzyskane wyniki w problematyce optymalizacji wykonywania transakcji w SRBD, wskazują na pewne nowe kierunki badań w tej dziedzinie.

Pierwszy z nich dotyczy duplikatów relacji na różnych stanowiskach komputerowych. Powchechnie akceptowanym w literaturze założeniem, jest założenie, że w ramach transakcji rozpatruje się tylko jedną kopię każdej relacji. Celowe jest przebadanie, czy biorąc pod uwagę nowy model kosztów wykonywania transakcji, uwzględnienie faktu istnienia w ogólności wielu kopii relacji w SRBD może doprowadzić do dalszej poprawy efektywności algorytmów generowania planów transakcji.

Innym zagadnieniem wartym dalszych badań jest problem wpływu współbieżnego wykonywania transakcji w SRBD na efektywność algorytmów generowania planów ich wykonywania. Celowe wydaje się uwzględnienie w przyszłości ewentualnych nierównomierności obciążenia SRBD w algorytmach generowania planów wykonywania transakcji.

Jeszcze innym kierunkiem badawczym w omawianej dziedzinie jest zastosowanie podejścia wielokryterialnego i poszukiwanie algorytmów generowania planów wykonywania transakcji w SRBD, zapewniających właściwy kompromis pomiędzy różnymi kryteriami oceny działania systemu.

5. Piśmiennictwo

[ALLC 83] Allchin J.E., McKendry M.S., Synchronization and recovery of actions, w: Proc.of 2nd Annual ACM Symp. on Principles of Distributed computing, 1983, s. 31-44.

[APER 83] Apers P.M., Hevner A.R., Yao S.B., Optimization algorithms for distributed queries, IEEE Trans. on Software Engineering, vol. SE-9, No 1, styczeń 1983, s. 57-67.

[ASTR 80] Astrahan M., Kim W., Schkolnick M., Evaluation of the system R access path selection mechanism, Research Raport RJ 2798, IBM Reasearch Laboratory, San Jose, 1980.

[BALD 79] Baldiddera C., Bracchi G., Ceri S.,A query processing strategy for distributed data bases, w: Proc. EURO-IFIP 79, 1979, s. 667-678.

[BEER 81] Beeri C., i inni; Properties of acyclic database schemes, w: Proc.Thirteenth Annual ACM Symposium on Theory of Computation, New York, maj 1981, s. 355-363.

[BEER 83] Beeri C., Fagin R., Maier D., Yannakakis M., On the desirability of acyclic database schemas, J. Assoc. Comput. Mach., vol. 30, No 3, 1983, s. 479-513.

[BER 81a] Bernstein P.A., Chiu D.W., Using semi-joins to solve relational queries, J. Assoc. Comput. Mach., Vol. 28, No 1, 1981, s. 25-40.

[BER 81b] Bernstein P.A., Goodman N., The power of natural semi-joins, SIAM J.Comput. vol. 10, No 4, 1981, s. 751-771.

[BER 81c] Bernstein P.A., Goodman N., The power of inequality semi-joins, Information Systems, vol. 6, No 4, 1981, s. 225-265.

[BER 81d] Bernstein P.A., i inni, Query processing in a system for distributed databases /SDD-1/, ACM TODS, vol. 6, No 4, 1981, s. 602-625.

[BLAC 81] Black P.A., Luk W.S., A new heuristic for generating semi-joins programs for distributed query processing, w: Proc.Conf. IEEE COMPSAC 1981, s. 581-588.

[BITT 83] Bitton D., DeWitt D.J., Turbyfill C., Benchmarking database systems a systematic approach, Computer Sciences Technical Raport No. 526, Univ. Wisconsin-Madison, grudzień 1983,

[BOUC 83] Bouchet P. i inni, Mesures de comportement en onvironment centralize et reparti sur SGBD relational-PEPIN, Raport de recherche No. 08, ISEM, Univ. Paris-Sud, 1983.

[CELL 80] Cellary W., Meyer D., A multi-query approach to distributed processing in a relational distributed data base management system, w: Proc. Intern. Symposium on Distributed Data Bases, ed.C.Delobel, W. Litwin, Paryż 1980, s. 99-121.

[CEL 85a] Cellary W., Królikowski Z., Morzy T., Others comments on "Optimization algorithms for distributed queries". IEEE Transaction on Software Enginnering, 1985 /przedstawiono do druku/.

[CEL 85b] Cellary W., Królikowski Z., Morzy T., Comparison of semi-join and join distributed strategies using a new analytical cost model, Pacyfic Computer Communication Symposium, Seoul, 1985.

[CERI 84] Ceri S., Pelagatti G., Distributed Databases-Principlrs and Systems, McGraw-Hill Book Company, 1984.

[CHEN 84] Chen A.L.P., Li V.O.K., Optimizing Star queries in distributed database system, w: Proc. 10th Intern.Conf. on VLDB, 1984, s.344-347.

[CHIU 84] Chiu Dah-Ming, Bernstein P.A., Ho Yu-Chi, Optimizing ohain queries in distributed database systems, SIAM J. Comput., vol. 13, No 1, s. 116-134.

[CHU 82] Chu W.W., Hurley P., Optimal query processing for distributed database systems, IEEE Trans.on Computers, vol. C-31, No 9, 1982, s. 835-850.

[CHEU 82] Cheung T.Y., A method for equijoin queries in distributed relational databases, IEEE Trans.Comput., vol. C-31, 1982.

[CHUN 84] Chung Ch.W., Irani K.G., A semi-join strategy for distributed query optimization, w: Proc. 4th Intern.Conf. on Distributed Computing, San Francisco, 1984, s. 368-377.

[CHR 84a] Christodoulakis S., Estimating block selectivities, Inform. Systems, vol. 9, Nr 1, 1984, s. 69-81.

[CHR 84b] Christodoulakis S., Implication of certain assumptions in database performance evaluation, ACM TODS, vol. 9, Nr 2, 1984, s. 163-186.

[DATE 83] Date C.J., An introduction to database systems, vol. II, Addison-Wesley Publishing Company, 1983.

[DANI 82] Daniels D., Query compilation in a distributed database system, IBM Res. Report, RJ 8423, IBM Research Laboratories, San Jose, 1982.

[DAYA 82] Dayal U., Goodman N., Query optimization for Codasyl Database Systems, w: Proc. ACM-AIGMOD conf.on Management of Data, Orlando, 1982, s. 138-150.

[EPST 78] Epstein R., Stonebraker M.R., Wong E., Distributed query processing in a relational data base system, w: Proc. ACM SIGMOD Intern. Conf.on Management of Data, Austine, USA, 1978, s. 169-180.

[FAG 83a] Fagin R., Deggrees of acyclity for hyp-ergraphs and relational database systems, J.Assoc.Comput.Mach., vol. 30, Nr 3, 1983, s. 514-550.

[FAG 83b] Fagin R., Acyclic database schemes /of various degrees/: a painlese introduction, Research Report RJ 3800/43497/ Computer Science, IBM Research Laboratory, San Jose, 1983.

[FAGI 84] Fagin R., The theory of data dependencies - a survey, Research Report RJ 4321/47149/ Computer Science, IBM Research Laboratory, San Jose, 1984.

[GOO 82a] Goodman M., Shmueli O., Tree queries: a simple class of relational queries, ACM TODS vol. 7, Nr 3, 1982, s. 653-677.

[GOO 82b] Goodman N., Shmueli O., The tree property is fundamental for query processing, w: Proc.ACM SIGACT-SIGMOD Conf.on Principles of database Systems, Los Angeles, marzec 1982, s. 40-48.

[GOO 82c] Goodman N., Shmueli O., Transforming cyclic schemas into trees, w: Proc.ACM SIGACT-SIGMOD Conf.on Principles of Database Systems, Los Angeles, 1982, s. 49-54.

[GOO 83a] Goodman N., Shmueli O., Syntactic characterization of tree database schemas, J.Assoc.Comput.Mach., vol. 30, Nr 4, 1983, s. 767-786.

[GOO 83b] Goodman N., Shmueli O., NP-complete problems simplified on tree schemas, Acta Informatica, Springer-Verlag, Nr 20, 1983, s.171-178.

[GOOD 84] Goodman N., Shmueli O., The tree projection theorem and relational query processing, Journal of Computer and System Sciences, vol. 28, Nr 1, 1984, s. 60-79.

[GOLD 84] Goldhirsch D., Yedwab L., Processing read only queries - over views with generalization, w: Proc. 10th Intern. Conf.on VLDB, 1984, s. 344-347.

[GOUD 81] Gouda M.G., Dayal U., Optimal semijoin schedules for query processing in local distributed database systems, w: Proc.ACM-SIGMOD Intern.Conf.on Management of Data, Ann Arbor, New York, 1981, s. 164-175.

[GRAY 81] Gray J.N., The transaction concept: virtues and limitations, w: Proc.of 7th VLDB Conf.Cannes, 1981, s. 144-154.

[GRIF 79] Griffiths, i inni, Access path selection in a relational database management system, IBM Research Rep., San Jose, RJ 2429, 1979.

[HEVN 79] Hevner A.R., Yao S.B., Query processing in distributed database systems, IEEE Trans.on Software Engineering, vol. SE-5, Nr 5, 1979, s. 177-187.

[HUAN 85] Huang T.K., A cost evaluation model for processing distributed query, Proc.of 18th Hawaii International Conf.on System Sciences, 1985.

[JAR 84a] Jarke M., Koch J., Schmidt J.W., Introduction to query processing, Working Paper Series, CRIs-73-GBA 84-48/CR, Graduate School of Business Administration, New York University, Marzec 1984.

[JAR 84b] Jarke M., Koch J., Query optimization in database systems, Computing Surveys, vol. 16, Nr 2, 1984, s. 111-152.

[KAMB 82] Kambayashi Y., i inni, Query processing for distributed database using generalized semi-joins,w: Proc. ACM-SIGMOD Intern.Confer.on Management of Data, Orlando, 1982, s. 151-160.

[KAMB 84] Kambayashi Y., Query processing using the consecutive retrieval property, IBM Res.Lab., RJ 4302, San Jose, 1984.

[KERS 82] Kerschberg L., Ting P.D., Yao S.B., Query optimization in star computer networks, ACM TODS, vol. 7, Nr 4, 1982, s. 678-711.

[KRO 85a] Królikowski Z., Some Remarks on Query Processing in Distributed Data Base Management Syrems, 8th International Seminar on DBMS, Piestany, Czechosłowacja, 1985.

[KRO 85b] Królikowski Z., Query Processing in Distributed Data Base Systems, w: Proc. 7th International Symposium - Computer at the University, Cavtat Zagreb, Jugosławia, 1985.

[KRO 85c] Królikowski Z., Towards Improved Strategies of Query Processing in Distributed Data Base Systems, w: Proc. 9th Symposium in Informatics Jahorina 85, Sarajevo, Jugosławia, 1985.

[KRO 85d] Królikowski Z., Szymańska M., Distribution Technique and Performance of Relational Database in Decentralized Health Care Environments, w: Proc. 6th International Congress of the European Federation for Medical Informatics, Helsinki, Finlandia, 1985.

[KRO 85e] Królikowski Z., Optymalizacja planów wykonywania transakcji w systemach rozproszonych baz danych, Rozprawa doktorska, Politechnika Gdańska, 1985.

[KRO 85f] Królikowski Z., Morzy T., Szulczyński J., Modelling and simulation analysis of query processing in distributed database systems, Intern.Conf. AMSE "Modelling and Simulation" Monastir, Tunezja, 1985.

[LAKH 84] Lakhani G.D., Wang J.S., Comments on "Optimization Algorithms for Distributed Queries", IEEE Transactions on Software Engineering, vol. SE-10, Nr 4, 1984.

[LUK 83] Luk W.S., Luk L., Optimizing semi-join programe for distributed query processing, w: Proc.Conf. ICOD-2, Cambridge, Anglia, 1983, s. 298-316.

[PERR 84] Perrizo W., A method for processing distributed database queries, IEEE Transactions on Software Engineering, vol. SE-10, No 4, 1984, s. 466-470.

[ROLI 82] Rolin P., Rapport sur la campagne de mesure de performance du prototype SIRIUS-DELTA, Rapports de Recherche No. 175, INRIA, Francja, 1982.

[SARF 81] Sarfati J., Rolin P., Measure on the SIRIUS-DELTA distributed data base prototype, w: Proc. 7th Int.Conf.on VLDB, Cannes, 1981.

[SCHW 84] Schwarz P.M., Spector A.Z., Synchronizing Shared abstract types, IEEE Trans.on Computer Systems, vol. 2, Nr 3, 1984.

[SELI 80] Selinger P.G., Adiba M.E., Access path selection in distributed data base management systems, Conf. ICOD, Aberdeen, Szkocja, Ed.S.M.Deen and P.Hammersley and Sons Ltd., 1980.

[SMIT 75] Smith J.M., Chang P.Y., Optimizing the performance of a relational algebra database interface, Comm.Ass.Comput, Machinery, vol. 18, Nr 10, 1975, s. 568-579.

[SPEC 83] Spector A.Z., Schwarz P.M., Transaction: a construct for reliable distributed computing, Operating Systems Review, vol. 17, Nr 2, 1983, s. 18-35.

[TOAN 81] Toan N.G., Distributed query management for a local network database system, w: Proc. 2nd Intern.Conference on Distributed Computing Systems, Paryż, 1981.

[ULLM 80] Ullman J., Principles of database systems, Computer Sciences Press, 1980.

[VARD 84] Vard Y.L., Vrbsky V., Distributed query processing allowing for redundant data, w: Proc. 4th Intern.Conf.on Distributed Computing, 1984, s. 389-396.

[WONG 77] Wong E., Retrieving dispersed data from SDD-1: a system for distributed data bases, w: Proc. 2nd Berkeley Conf.on Distributed Data Management and Computer Networks, 1977, s. 217-235.

[WONG 76] Wong E., Youssefi K., Decomposition - a strategy for query processing, ACM TODS, vol. 1, Nr 3, 1976, s. 223-241.

[YAO 77] Yao S.B., Approximating block accesses in database organizations, CACM, vol. 20, Nr 4, 1977, s. 260-261.

[YOSH 84] Yoshikawa M., Kambayashi Y., Processing inequality queries based on generalized semi-join, w: Proc. 10th Intern.Conf.on VLDB, 1984, s. 416-428.

[YO 80] Yu C.T., Lam K., Ozsoyoglu M., Distributed query optimization for tree queries, Technical Report, Dep. of Information Engineering, Univ.of Illinois at Chicago Circle, 1980, s. 1-55.

[YU 83] Yu C.T., Chang C.C., On the design of a query processing strategy in a distributed database environment, w: Proc. SIGMOD 83 Conf., New York, 1983, s. 30-39.

Modyfikowanie "views" w relacyjnych bazach danych

Janusz Getta

Instytut Informacji Naukowej, Technicznej i Ekonomicznej

00-926 Warszawa ul. Żurawia 3/5

Streszczenie

W pracy opisano metody translacji modyfikacji wykonywanych w relacjach schematu zewnętrznego / "view" / na modyfikacje w relacjach bazy danych.

Przyjmijmy, że schemat zewnętrzny zdefinowany jest w postaci następującego wyrażenia algebry relacyjnej:

/1/ $v = f(r_1, r_2, \ldots, r_n)$

gdzie v jest relacją widzianą przez użytkownika, natomiast $r_1, r_2, \ldots, r_n$ są relacjami występującymi w bazie danych. Taka metoda definiowania schematu zwenętrznego dostarcza jednocześnie informacji o tym w jaki sposób powinny być obliczane wartości relacji v występującej w pytaniu użytkownika. Znalezienie wartości v polega na obliczeniu wartości wyrażenia $f(r_1, r_2, \ldots r_n)$. Zatem, jeżeli w relacjach bazy danych wkonane zostały jakiekolwiek zmiany będą się one automatycznie "przenosiły" do relacji schematu zewnętrznego. Znacznie trudniejszym problemem jest odwzorowanie zmian wykonanych w relacjach schematu zewnętrznego na zmiany w relacjach bazy danych.

W pracy omówione zostały kryteria istnienia i jednoznaczności takich odwzorowań. Przedstawione zostały metody translacji modyfikacji w oparciu o tzw. schemat dopełniający i układy równań w modelu algebry relacyjnej.

1. Wstęp

W większości systemów baz danych istnieje możliwość definiowania przez użytkownika swojego "spojrzenia" na bazę danych. Mechanizm ten umożliwia wybranie z bazy danych interesujących użytkownika fragmentów i ewentualne przekształcenie ich do zadanej postaci. Takie "spojrzenie" użytkownika przyjęto nazywać schematem zewnętrznym lub schematem użytkownika. Mechanizm definiowania schematów zewnętrznych posiada następujące zalety:

- umożliwia uproszczenie współpracy użytkownika z systemem bazy danych

przez zignorowanie tych danych, które nie są w obszarze jego zainteresowań,

- wymusza niezależność danych od sposobu ich reprezentacji, tzn. sytuację w której zmiany w strukturze bazy danych nie wpływają na zmiany w schemacie zewnętrznym,
- dostarcza mechanizmów ochrony danych przez niedopuszczenie użytkownika do korzystania z obszarów bazy danych będących poza jego schematem.

Schemat zewnętrzny definiowany jest w postaci sekwencji operacji przekształcających schemat bazy danych. Taka sekwencja jest nazywana funkcją definiującą. W modelu relacyjnym funkcja definiująca schemat zewnętrzny pozwala na zmiany nazw atrybutów, zmiany sposobu reprezentacji wartości atrybutów, tworzenie atrybutów wyliczanych oraz obliczanie wartości wyrażeń algebry relacyjnej. Ponieważ definicja schematu zewnętrznego jest funkcją, wszystkie zmiany w relacjach bazy danych mogą być w jednoznaczny sposób przetłumaczone na zmiany w relacjach schematu zewnętrznego. Aby mechanizm schematu zewnętrznego był w pełni wykorzystany system bazy danych powinien zapewniać zarówno możliwości wyszukiwania w relacjach schematu zewnętrznego jak i modyfikację tych relacji. Realizacja wyszukiwania w najprostszym przypadku nie stanowi tutaj specjalnego problemu. Metody obliczania zapytań dla relacji schematu zewnętrznego zostały omówione w pracy [Os80]. Przypadek realizacji zapytań gdy schemat zewnętrzny jest tworzony w oparciu o rekurencyjne definicje został omówiony w pracach [Ch81], [HeNa84]. Realizacja modyfikacji w relacjach schematu zewnętrznego wymaga ich jednoznacznej translacji na modyfikacje w relacjach bazy danych. Okazuje się jednak, że w wielu przypadkach taka translacja jest niemożliwa lub niejednoznaczna. W celu dokładniejszego omówienia powstających tutaj problemów przedstawiony zostanie przykład bazy danych i schematu zewnętrznego zaczerpnięty z pracy [DaBe78]. Schemat bazy danych, schemat zewnętrzny oraz odpowiednie relacje zostały przedstawione poniżej.

Schemat bazy danych:

ED (EMPNAME, DEPT)
ESA (EMPNAME. SALARY, AGE)
DM (DEPT, MGR)

DM*

DEPT	MGR
Normalization	Bernstein
Normalization	Codd
Concurrency	Marill
Implementation	Marill

ESA*

EMPNAME	SALARY	AGE
Beeri	55K	32
Rothnie	75K	35
Goodman	39K	35

ED*

EMPNAME	DEPT
Beeri	Normalization
Rothnie	Concurrency
Goodman	Implementation
Gagliardi	Performance
Fagin	Normalization
Mylopulos	AI
Shipman	Concurrency

Schemat zewnętrzny zdefiniowany jest następująco:

EM : $\pi_{EMPNAME,MGR}$ (ED ⋈ DM)

EDS : $\pi_{EMPNAME,DEPT,SALARY}$ (ED ⋈ ESA

EDM : ED ⋈ DM

Relacje schematu zewnętrznego:

EM*

EMPNAME	MGR
Beeri	Bernstein
Beeri	Codd
Rothnie	Marill
Goodman	Marill
Fagin	Bernstein
Fagin	Codd
Shipman	Marill

EDM*

EMPNAME	DEPT	MGR
Beeri	Normalization	Bernstein
Beeri	Normalization	Codd
Rothnie	Concurrency	Marill
Goodman	Implementation	Marill
Fagin	Normalization	Bernstein
Fagin	Normalization	Codd
Shipman	Concurrency	Marill

EDS*

EMPNAME	DEPT	SALARY
Beeri	Normalization	65k
Rothnie	Concurrency	75k
Goodman	Implementation	39k

Zmiany wykonane w relacjach schematu zewnętrznego mogą być przetłumaczone na zmiany w relacjach bazy danych gdy spełnione są następujące warunki:

- modyfikacje nie powodują powstania wartości nieokreślonych w relacjach bazy danych,
- modyfikacje nie powodują powstania efektów ubocznych,
- modyfikacje nie naruszają logicznych ograniczeń zgodności narzuconych na relacje bazy danych,
- modyfikacje mogą być przeprowadzone w sposób jednoznaczny.

Wartości nieokreślone mogą się pojawić w relacji ESA* gdy dodamy krotkę ⟨Ullman , Theory , 75k⟩ do relacji EDS* . Wówczas do relacji ED* dodajemy krotkę ⟨Ullman , Theory⟩natomiast do ESA* ⟨Ullman , 75k , -⟩. Wartość atrybutu AGE jest w tym przypadku nieokreślona. Przykładem występowania efektów ubocznych może być sytuacja , w której wstawienie krotki ⟨Buzen , Performance , Marill⟩do relacji EDM* powoduje wstawienie krotek ⟨Buzen , Performance⟩ do ED* oraz ⟨Performance , Marill⟩

do relacji DM^*. Teraz , jeżeli powtórnie obliczymy wartość relacji EDM^* to okaże się, że dodana został do niej także krotka <Gagliardi , Performance , Marill > . Załóżmy, że w relacji ED^* atrybut EMPNAME jest kluczem. Przykładem naruszenia logicznych ograniczeń zgodności jest wstawienie krotki <Shipman , Implementation , Marill>do EDM^*. Powoduje to dodanie krotki < Shipman , Implementation> do ED^* co jest sprzeczne z warunkiem mówiącym, że atrybut EMPNAME jest kluczem tej relacji. Przykładem powstania niejednoznaczności w translacji zmian jest sytuacja , w której dodanie krotki do relacj EDS^* powoduje powstanie wartości nieokreślonych w relacji ESA^*. Wartość atrybutu AGE może być tutaj wstawiona na wiele sposobów. Podobna sytuacja występuje w przypadku usunięcia krotki z relacji zdefiniowanej jako różnica dwóch relacji z bazy danych.

2. Translacja modyfikacji dla schematów elementarnych

Przez schemat elementarny będziemy rozumieli taki schemat zewnętrzny, który powstaje przez jednokrotne wykorzystanie operacji algebry relacyjnej w jego definicji. Przykładami schematów elementarnych mogą być wyniki operacji ograniczenia relacji, rzutowania relacji, połączenia relacji itp. Translacja zmian dla przypadku schematów elementarnych została omówiona w pracy [Ma84] . Jako podstawowe operacje algebry relacyjnej przyjęto tutaj iloczyn kartezjański dwóch relacji , sumę i różnicę teoriomnogościową , rzutowanie oraz operację Θ- ograniczenia relacji. Definicje tych operacji przedstawione zostały poniżej.

$v = u \times w : \forall t \in dom(v)\ (M\_of\_v(t) = (M\_of\_U(t[att(U)]) \text{ and } M\_of\_w(t[att(W)]))$

$v = u \cup w : \forall t \in dom(v)\ (M\_of\_v(t) = (M\_of\_U(t) \text{ or } M\_of\_w(t)))$

$v = u - w : \forall t \in dom(v)\ (M\_of\_v(t) = (M\_of\_U(t) \text{ and } M\_of\_w(t)))$

$v = u[X] : \forall t \in dom(v)\ (M\_of\_v(t) = (\exists u \in dom(u)\ (u[X] = t \text{ and } M\_of\_u(t) = true)))$

$v = u[X \Theta Y] : \forall t \in dom(v)\ (M\_of\_v(t) = (M\_of\_u(t) \text{ and } t[X] \Theta t[Y]))$

Zbiór wszystkich możliwych modyfikacji podzielono na dwie klasy: usuwanie krotek , dodawanie krotek. Następnie dla każdej z klas oraz dla wszystkich schematów elementarnych przedyskutowano możliwe translacje.

Usuwanie krotek

Niech schemat zewnętrzny będzie zdefiniowany jako iloczyn kartezjański relacji u i w, v = u×w . Usunięcie krotki jest w tym przypadku możliwe tylko wtedy gdy zmodyfikowana relacja v' spełnia zależność połączeniową $*[X , Y]$ gdzie X,Y są schematami relacji u i w. Wtedy zmodyfikowane relacje u' i w' mogą być obliczone w następujący sposób:

$$u' = \Pi_X(v') \quad , \quad w' = \Pi_Y(v')$$

Niech schemat zewnętrzny będzie sumą relacji u i w , v = u∪w .

Wtedy usunięcie krotki z relacji v może zostać jednoznacznie przetłumaczone na usunięcie tej samej krotki z u i w .

Niech schemat zewnętrzny będzie zdefiniowany jako różnica relacji, v = u w . Wtedy usunięcie krotki z v może zostać przetłumaczone na następujące modyfikacje w relacjach bazy danych:

- usunięcie krotki z relacji u,
- dodanie krotki do relacji w,
- usunięcie krotki z relacji u i dodanie krotki do relacji w.

Niech schemat zewnętrzny będzie zdefiniowany jako rzut relacji na podschemat Z , $v = \pi_Z(u)$. Wówczas usunięcie krotki z v odpowiada usunięciu tych wszystkich krotek z relacji u , których rzut na schemat Z jest identyczny z usuwaną krotką.

Niech schemat zewnętrzny będzie zdefiniowany jako Θ- ograniczenie, $v = u[X\Theta Y]$. Wtedy usunięciu krotki z v odpowiada usunięcie odpowiedniej krotki z u.

Dodawanie krotek

W przypadku gdy schemat zewnętrzny jest zdefiniowany jako $v = u \cap w$ dodanie krotki do v może zostać przetłumaczone jednoznacznie na dodanie krotek do relacji u i w tylko wtedy gdy v' spełnia zależność połączeniową $\bowtie [X , Y]$.

W przypadku gdy $v = u \cup w$ dodanie krotki do v może zostać przetłumaczone na:

- dodanie krotki do u,
- dodanie krotki do w,
- dodanie krotki do u i do w jednocześnie.

Niech schemat zewnętrzny będzie zdefiniowany jako różnica relacji v = u-w. Wtedy dodanie krotki do v może zostać przetłumaczone na dodanie tej samej krotki do u i usunięcie jej z w.

Niech schemat zewnętrzny będzie zdefiniowany jako $v = \pi_Z(u)$. Wtedy przetłumaczenie modyfikacji jest możliwe tylko wtedy gdy jesteśmy w stanie odtworzyć wartości atrybutów X-Z.

Niech schemat zewnętrzny będzie zdefiniowany jako $v = u[X\Theta Y]$. Wtedy dodanie krotki do v może zostać przetłumaczone na dodanie tej samej krotki do u pod warunkiem, że spełnia ona ograniczenie $[X\Theta Y]$.

Powstaje tutaj pytanie: czy określenie wszystkich możliwych translacji dla schematów elementarnych jest wystraczające dla określenia translacji dla schematów złożonych. Okazuje się, że zbiór translacji dla schematów elementarnych nie jest całkowicie wystarczający. Rozważmy następujący przykład. Niech schemat zewnętrzny będzie zdefiniowany w

następujący sposób: $v = u[X\Theta Y]$; $u = w_1 \times w_2$ gdzie w_1 , w_2 są relacjami z bazy danych o schematach W_1 , W_2 odpowiednio. Załóżmy , że modyfikacja polega na usunięciu krotki z relacji v. Odpowiada to usunięciu krotki z relacji u, a następnie usunięciu odpowiednich krotek z w_1 i w_2 . przypuśćmy, że taka modyfikacja narusza zależność połączeniową $*[W_1 , W_2]$ w relacji u. Zatem wydaje się , że translacja tej zmiany w ogólnym przypadku nie może być wykonana. Jednakże pomimo naruszenia zależności połączeniowej $*[W_1 , W_2]$translacja w pewnych przypadkach jest możliwa. Ponieważ usunięcie krotki z relacjiu u narusza $*[W_1 , W_2]$to dodatkowo usuwamy takie krotki z u aby zależność $*[W_1 , W_2]$była spełniona i jednocześnie usunięte krotki nie spełniały warunku $[X\Theta Y]$. W tym przypadku po wykonaniu uperacji $u[X\Theta Y]$otrzymamy wymaganą wartość relacji v. Rozważmy następujący przykład. Niech w_1 , w_2 , u podane są poniżej.

w_1

SUPPLIER	PART
s1	p1
s2	p2
s3	p1
s3	p2
s3	p3

w_2

PART	CUSTOMER
p1	c1
p1	c2
p2	c2

u

SUPPLIER	w_1.PART	w_2.PART	CUSTOM.
s1	p1	p1	c1
s2	p2	p1	c1
s3	p1	p1	c1
s3	p2	p1	c1
s3	p3	p1	c1
s1	p1	p1	c2
s2	p2	p1	c2
s3	p1	p1	c2
s3	p2	p1	c2
s3	p3	p1	c2
s1	p1	p2	c2
s2	p2	p2	c2
s3	p1	p2	c2
s3	p2	p2	c2
s3	p3	p2	c2

Załóżmy następujący warunek ograniczenia w_1.PART = w_2.PART . Relacja v podana jest poniżej.

v

SUPPLIER	w_1.PART	w_2.PART	CUSTOMER
s1	p1	p1	c1
s1	p1	p1	c2
s2	p2	p2	c2
s3	p1	p1	c1
s3	p1	p1	c2
s3	p2	p2	c2

Modyfikacja wykonywana na relacji v polega na usunięciu z niej krotek: $\langle s1 , p1 , c1\rangle$, $\langle s1 , p1 , c2\rangle$, $\langle s3 , p1 , c1\rangle$, $\langle s3 , p1 , c2\rangle$. Jeżeli postępując zgodnie z regułami translacji dla schematów elementarnych usuniemy te krotki z relacji u to nie będzie spełniona tam zależ-

ność połączeniowa $*[SUPPLIER\ w_1.PART\ ,\ w_2.PART\ CUSTOMER]$. Natomiast zależność ta będzie spełniona jeżeli dodatkowo usuniemy z u krotki: $\langle s1\ ,\ p1\ ,\ p2\ ,\ c2\rangle$ oraz $\langle s3\ ,\ p1\ ,\ p2\ ,\ c2\rangle$. Ponieważ krotki te nie spełniają warunku $[w_1.PART = w_2.PART]$ ich usunięcie nie prowadzi do powstania efektów ubocznych. Zmiany które powinny być wykonane w relacjach w_1 i w_2 to usunięcie krotek $\langle s_1\ ,\ p_1\rangle$, $\langle s_3\ ,\ p1\rangle$ z relacji w_1 i $\langle p1\ ,\ c1\rangle$, $\langle p1\ ,\ c_2\rangle$ z relacji w_2 .

3. Translacja modyfikacji z wykorzystaniem schematu dopełniającego.

W pracy [EaSp81] zaproponowano aby wraz z definicją schematu zewnętrznego określać tzw. schemat dopełniający , w taki sposób aby para schemat zewnętrzny, schemat dopełniający pozwalała na odtworzenie wszystkich relacji z bazy danych. Zakłada się również, że translacje zmian dla ustalonego schematu zewnętrznego definiujemy tak aby relacje ze schematu dopełniającego pozostały stałe. W omawianej pracy odowodniono, że kompletny zbiór U modyfikacji w relacjach schematu zewnętrznego posiada translację wtedy i tylko wtedy gdy istnieje schemat dopełniający i każde u U jest translowalne przy stałym schemacie dopełniającym.

Przyjmijmy następujące oznaczenia. Niech f będzie funkcją definiującą schemat zewnętrzny, S będzie zbiorem stanów bazy danych, wtedy modyfikacja relacji w bazie danych będzie zdefiniowana jako funkcja $t : S \to S$. Niech U_1 będzie zbiorem modyfikacji wykonywanych w relacjach bazy danych , U_f będzie zbiorem modyfikacji w schemacie zewnętrznym.

Aby nie występowały efekty uboczne następujący warunek powinien być spełniony: $\forall s \in S\ \ f(t(s)) = u(f(s))$, gdzie $u \in U_f$.

Niech $u \in U_f$ oraz $t_u \in U_1$, wtedy t_u jest nazywane translacją gdy spełnione są następujące warunki:

- $f(t_u) = u(f)$
- $\forall s \in S\ \ u\ f(s) = f(s) \Rightarrow t_u(s) = s$

Niech $U \subset U_f$ będzie kompletnym zbiorem zmian. Odwzorowanie $T : U \to U_1$ jest nazywane translatorem wtedy i tylko wtedy gdy:

- $\forall u \in U\ \ T(u)$ jest translacją u,
- $\forall u \in U \forall v \in U\ \ T(uv) = T(u)\ T(v)$.

Niech $M(s)$ będzie zbiorem definicji schematów zewnętrznych. Niech $f, g \in M(s)$. Mówimy, że f jest większe niż g lub że f określa g $(f \geq g)$ wtedy i tylko wtedy gdy $\forall s \in S\ \forall s' \in S\ \ f(s) = f(s') \Rightarrow g(s) = g(s')$.

Niech $f, g \in M(s)$. Mówimy ,że f, g są równoważne $(f \equiv g)$ wtedy i tylko wtedy gdy $f \geq g$ i $g \leq f$.

Niech f, g$\in$M(s). Produkt f i g oznaczamy przez f×g i definiujemy jako $f \times g = \langle f(s), g(s) \rangle \forall s \in S$.

Niech f$\in$M(s) . Mówimy , że g$\in$M(s) jest dopełnieniem f wtedy i tylko wtedy gdy $f \times g \cong 1$, gdzie 1 oznacza identycznościowe przekształcenie schematu bazy danych. Oznacza to, że z pary $\langle f , g \rangle$ możemy odtworzyć wszystkie relacje z bazy danych.

Przykładowo niech schemat bazy danych będzie zdefiniowany następująco: r(A , B), s(B , C) i niech schemat zewnętrzny będzie zdefiniowany jako f : $v = \pi_{A,C}(r \bowtie s)$. Niech g : s(B , C). Schemat g jest schematem dopełniającym gdyż pozwala na odtworzenie wartości relacji r i s z relacji v . ($r = \pi_{A,B}(v \bowtie s)$).

Niech u U_f, niech g jest dopełnieniem f , zmiana u jest g-translowalna wtedy i tylko wtedy gdy:

$\forall s \in S \; \exists s' \in S$ $f(s') = u(f(s))$ brak efektów ubocznych

$g(s') = g(s)$ stałe dopełnienie

Niech g będzie dopełnieniem f, niech u$\in U_f$ i u jest g-translowalne oraz $\gamma_u = (f \times g)^{-1} (u (f \times g))$.

Wtedy γ_u jest translacją u ,

$g \gamma_u = g$

Rozważmy następujący przykład. Schemat bazy danych zdefiniowany jest następująco: s(EMP,DEP), r(DEP,MGR), schemat zewnętrzny jest zdefinowany jako $v = \pi_{EMP,MGR}(s \bowtie r)$. Wartości relacji s , r , v są następujące:

s

EMP	DEP
A	1
B	1
C	m

r

DEP	MGR
1	X
m	Y

v

EMP	MGR
A	X
B	X
C	Y

Zakładamy modyfikację u : [A] $\Rightarrow$ [F] oraz dopełnienie g : r

Wtedy translacja γ_u jest zdefiniowana następująco:

$s' = \pi_{EMP,DEP}(r \bowtie v')$

$r' = r$

gdzie v' jest relacją schematu zewnętrznego po wykonaniu modyfikacji u.

s

EMP	DEP
A	1
B	1
C	m

r

DEP	MGR
1	X
m	Y

$\xrightarrow{\gamma_u}$

s'

EMP	DEP
F	1
B	1
C	m

r'

DEP	MGR
1	X
m	Y

$\downarrow f$

v

EMP	MGR
A	X
B	X
C	Y

$\xrightarrow{u}$

v'

EMP	MGR
F	X
B	X
C	Y

Jednoznaczność g-translacji jest implikowana przez założenie o niezmienności relacji schematu dopełniającego. Z wyborem dopełnienia jest ściśle związany spoób modyfikacji relacji z bazy danych. Zależność sposobu modyfikacji od dopełnienia pokazana została w poniższym przykładzie zaczerpniętym z pracy [BaSp81].

Niech schemat bazy danych będzie zdefiniowany następująco:

r (PRODUCT, COST, SALEPRICE, PROFIT, PROFITRATE)

oraz niech następujące logiczne ograniczenia zgodności będą narzucone na r:

c_1 : PRODUCT $\longrightarrow$ COST, SALEPRICE, PROFIT, PROFITRATE

c_2 : $\forall x \in r$ x.COST $\geq$ 0

x.SALEPRICE $\geq$ x.COST

x.PROFIT $\geq$ x.SALEPRICE - x.COST

x.PROFITRATE = x.PROFIT / x.COST

Niech schemat zewnętrzny będzie zdefiniowany następująco:

$v = \pi_{\text{PRODUCT, COST}}(r)$

Rozważmy następujące trzy schematy dopełniające:

g_1 : $d_1 = \pi_{\text{PRODUCT, SALEPRICE}}(r)$

g_2 : $d_2 = \pi_{\text{PRODUCT, PROFIT}}(r)$

g_3 : $d_3 = \pi_{\text{PRODUCT, PROFITRATE}}(r)$

Załóżmy, że w relacji v wykonujemy następującą zmianę: "zmniejszamy koszt wszystkich produktów o 10%".

Zauważmy że i=1,2,3 g_i jest dopełnieniem f i że u jest g_i-translowalne. Zmiana u może zostać przetłumaczona na następujące zmiany u_1, u_2, u_3 w relacji r w zależności od wybranego dopełnienia.

u_1 : SALEPRICE pozostaje bez zmian, wartości PROGIT i PROFITRATE zwiększają się,

u_2 : PROFIT pozostaje bez zmian, SALEPRICE maleje, PROFITRATE rośnie,

u_3 : PROFITRATE pozostaje bez zmian , PROFIT , SALEPRICE maleją.

W omawianej pracy autorzy proponują następujący sposób przeprowadzania translacji zmian:

- wybierz dopełnienie g dla danego schematu zewnętrznego f,
- sprawdź czy zmiany należące do zbioru zmian U nie powodują modyfikacji w relacjach schematu dopełniającego g,
- dla każdej zmiany u U zbuduj translator $\gamma_u = (f \times g)^{-1}(u(f \times g))$.

4. Translacja modyfikacji przez rozwiązywanie układów równań

Przyjmijmy, że schemat zewnętrzny zdefiniowany jest w postaci następującego wyrażenia algebry relacyjnej:

/1/ $v = f(r_1, r_2, \ldots, r_n)$

gdzie v jest relacją "widzianą" przez użytkownika natomiast $r_1, r_2, \ldots, r_n$ są relacjami występującymi w bazie danych. Załżmy że w relacji v wykonano pewne modyfikacje. Niech u będzie relacją powstałą po wykonaniu zmian oraz $q_1, q_2, \ldots, q_n$ będą odpowiednio zmodyfikowanymi relacjami $r_1, r_2, \ldots, r_n$ bazy danych. Modyfikacja relacji z bazy danych powinna spełniać następujące warunki:

(i) nie mogą powstawać wartości nieokreślone w relacjach bazy danych,

(ii) sposób modyfikacji powinien być jednoznaczny,

(iii) wykonanie modyfikacji nie powinno powodować powstania efektów ubocznych.

Z warunku (iii) wynika konieczność spełniania następującego równania:

/2/ $u = f(q_1, q_2, \ldots, q_n)$

Warunek (i) jest równoważny istnieniu rozwiązania równania /2/, natomiast warunek (ii) odpowiada istnieniu dokładnie jednego rozwiązania. Jeżeli $q_1, q_2, \ldots, q_n$ potraktujemy jako niewiadome to rozwiązując równanie /2/ ze względu na $q_1, q_2, \ldots, q_n$ znajdziemy jednocześnie sposób obliczenia wartości relacji bazowych po modyfikacji. Jednoznaczność zmian odpowiada jednoznaczności rozwiązania równania /2/. Oczywiście w większości przypadków nie możemy spodziewać się istnienia jednoznacznego rozwiązania. W tym przypadku do równania /2/ dodajemy dodatkowe równania wynikające np. z logicznych ograniczeń zgodności narzuconych na relacje z bazy danych. Otrzymujemy w ten sposób układ równań, którego rozwiązanie opisuje w jednoznaczny sposób translację zmian. Jednocześnie uwzględniony jest warunek aby wykonane zmiany nie naruszały logicznych ograniczeń zgodności. Ograniczenia pełnią tutaj rolę reduktora przestrzeni rozwiązań.

W dalszej części pracy przyjęto definicje podstawowych operacji algebry relacyjnej podane w pracy [Ma83]. Ponadto zdefiniowane zostały dwa następujące dodatkowe operatory:

operacja częściowej różnicy:

$r \dot{-} s = \{t : t \in r \wedge \forall u \in s \quad t[Z] \neq u[Z]\}$ gdzie r,s są relacjami o schematach X, Y oraz $Z \neq \emptyset$, $Z = X \cap Y$.

operacja częściowej sumy:

$r \dot{+} s = \{t : t \in r \vee \exists u \in s \ (u[Z] = t[Z])\}$

Przyjęto także oznaczać największą możliwą relację o schemacie X przez P(X)

Niech schemat użytkownika będzie zdefiniowany jako ograniczenie relacji r ? schemacie X przez warunek c.

/3/ $v = \sigma_c(r)$

Sytuacja po wykonaniu modyfikacji w relacji v może być opisana następującym równaniem:

/4/ $u = \sigma_c(q)$

gdzie u jest relacją schematu zewnętrznego po wykonaniu modyfikacji natomiast q jest nieznaną relacją bazy danych odpowiadającą relacji r po wykonaniu modyfikacji.

Metoda rozwiązywania tego typu równań pokazana zostanie dalej, natomiast obecnie przeprowadzona zostanie dyskusja otrzymanych rozwiązań. Przestrzeń rozwiązań tego równania jest kratą zupełną. Dlatego możemy ją opisać podając najmniejsze i największe rozwiązanie.

Najmniejszym rozwiązaniem równania /4/ jest $q_* = u$, największym rozwiązaniem jest $q^* = P(X) \supset \left(\sigma_c\left(P(X) \supset u\right)\right) = u \cup \sigma_{\bar{c}}\left(P(X)\right)$

Zatem rozwiązanie równania /4/ istnieje ale nie jest jednoznaczne. Rozwiązanie byłoby jednoznaczne tylko w wyjątkowym przypadku gdy warunek c spełniałby zależność:

$$\sigma_{\bar{c}}\left(P(X)\right) = \emptyset$$

tzn. nigdy nie powodowałby ograniczenia relacji r.

Załóżmy teraz, że zapamiętana zostanie ta część relacji, która jest z niej usuwana w wyniku działania $\sigma_c(r)$. Oznaczmy ją przez $\bar{r} = \sigma_c(r)$. Zatem relacja r musi się składać z dwóch rozłącznych zbiorów:

/5/ $r = \sigma_c(r) \cup \bar{r}$

Równanie /5/ możemy potraktować jako logiczne ograniczenie zgodności dla relacji r. Powykonaniu zmian w relacji v równanie /5/ przepisujemy do postaci:

/6/ $q = \sigma_c(q) \cup \bar{r}$

Teraz otrzymujemy układ równań:

$$\begin{cases} u = \sigma_c(q) & /7.1/ \\ q = \sigma_c(q) \cup \bar{r} & /7.2/ \end{cases}$$

Rozwiązanie równania 7.1 podane jest poniżej:

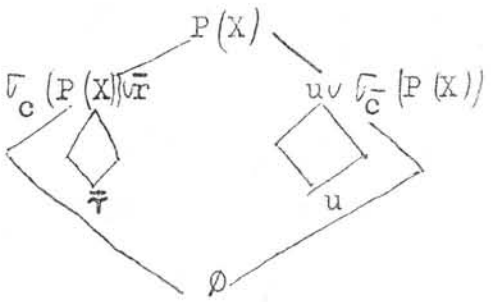
$q^* = u \cup \sigma_{\bar{c}} P(X)$

$q_* = u$

Rozwiązanie /7.2/ :

$q^* = \sigma_c\left(P(X)\right) \cup \bar{r}$

$q_* = \bar{r}$



Z połączenia tych rozwiązań otrzymujemy:

$q^{*'} = \left(\overline{\Gamma}_c(P(X)) \cup \bar{r}\right) \cap \left(u \cup \overline{\Gamma}_{\bar{c}}(P(X))\right)$

$q'_{*} = \bar{r} \cup u$

Ponieważ $\bar{r} \subseteq \overline{\Gamma}_{\bar{c}} P(X)$ oraz $u \subseteq \overline{\Gamma}_c (P(X))$ to $q^{*'}$ możemy przekształcić do postaci $\bar{r} \cup u$. Stąd otrzymujemy jednoznaczne rozwiązanie

/8/ $q^{*} = \bar{r} \cup u$

Rozwiązanie to możemy interpretować w następujący sposób: "jeżeli zapamiętana została część relacji r wyeliminowana przez operację ograniczenia to możliwa jest jednoznaczna translacja zmian, i opisuje ją równanie /8/ ".

Rozwiązywanie równań o postaci /2/ możemy wykonać przez przekształcenie równania do tzw $\emptyset$-równań a następnie do równań stałopunktowych.

Niech dane będzie równanie o postaci:

/9/ $f(q) = h(q)$

Równanie /8/ jest równoważne układowi $\emptyset$-równań:

$f(q) \rightharpoonup h(q) = \emptyset$

$h(q) \rightharpoonup f(q) = \emptyset$

Niech dane będzie równanie :

/10/ $f(q) = \emptyset$

Równanie /10/ jest równoważne równaniu stałopunktowemu:

/11/ $q = q \rightharpoonup f(q)$

wtedy i tylko wtedy gdy:

$$\forall q \left(f(q) \neq \emptyset \Rightarrow f(q) \ltimes q \neq \emptyset\right)$$

Równanie /10/ jest równoważne równaniu stałopunktowemu:

/12/ $q = q \oplus f(q)$

wtedy i tylko wtedy gdy:

$$\forall q \left(f(q) \neq \emptyset \Rightarrow f(q) \ltimes q \neq f(q)\right)$$

Powyższe warunki można osłabić otrzymując jednocześnie kryteria równoważności równań wygodniejsze do zastosowania.

Jeżeli $\forall q \quad f(q) \subseteq q$ to równanie /10/ jest równoważne równaniu /11/.

Jeżeli $\forall q \neq \emptyset \quad f(q) \not\subseteq q \wedge f(\emptyset) = \emptyset$ to równanie /10/ jest równoważne równaniu /12/

Rozwiązywanie równań stałopunktowych możemy wykonywać w oparciu o tw. Kleene [Kl52] , [Ta55].

Niech $f(q)$ będzie rosnąco i malejąco ciągłą funkcją . Najmniejszy /największy/ punkt stały funkcji $f(q)$ istnieje i jest równy:

$$q_{*} = \bigcup_{i=0}^{\infty} f^i(\emptyset)$$

$$/\ q^{*} = \bigcap_{i=0}^{\infty} f^i(P(X))\ /$$

gdzie $f^i(q)$ jest zdefiniowane następująco:

$$f^0(q) = q$$

$$f^1(q) = f(q)$$
$$f^i(q) = f(f^{i-1}(q))$$

Korzystając z powyższych wzorów rozwiązujemy równanie /7.2/ w następujący sposób:

$$f(q) = \Gamma_c(q) \cup \bar{r}$$

$$f^0(\emptyset) = \emptyset \qquad f^0(P(X)) = P(X)$$
$$f^1(\emptyset) = \bar{r} \qquad f^1(P(X)) = \Gamma_c(P(X)) \cup \bar{r}$$
$$f^2(\emptyset) = \bar{r} \qquad f^2(P(X)) = \Gamma_c(\Gamma_c(P(X)) \cup \bar{r}) \cup \bar{r} = \Gamma_c(P(X)) \cup \bar{r}$$

$$q_* = \bar{r} \qquad q^* \equiv \Gamma_c(P(X)) \cup \bar{r}$$

Pokazane zostaną teraz przykłady znajdowania translacji zmian przez rozwiązywanie układów równań dla schematów elementarnych.

Niech schemat zewnętrzny będzie zdefiniowany w następujący sposób:

/13/ $v = \Pi_Z(r)$

gdzie r jest relacją o schemacie X i $Z \subseteq X$.

Po wykonaniu modyfikacji w v otrzymujemy następujące równanie opisujące sposób odtworzenia relacji r

/14/ $u = \Pi_Z(q)$

Równanie to możemy przekształcić do równoważnej postaci

/15.1/ $u \supset \Pi_Z(q) = \emptyset$

/15.2/ $\Pi_Z(q) \supset u = \emptyset$

Równanie /15.1/ przekształcamy do postaci:

/15.1'/ $q = q \oplus (u \supset \Pi_Z(q))$

Dla powyższego równania nie istnieje najmniejsze rozwiązanie. Możemy określić tutaj jedynie dolne ograniczenie:

$$q_\sim = \{t : \Pi_Z(t) = u \wedge t(Z \to X-Z)\}$$

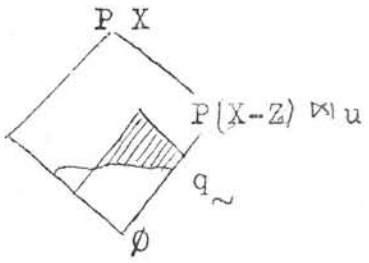
/ Notacja $t(Z \to X-Z)$ oznacza konieczność spełniania zależności funkcjonalnej $Z \to X-Z$ przez relację t /

Największym rozwiązaniem jest $q^* = P(X)$, oraz wszystkie relacje należące do zbioru rozwiązań muszą spełniać warunek $\Pi_Z\, t \supseteq u$

Rozwiązanie równania /15.2/ jest następujące:

$$q^* = P(X-Z) \bowtie u$$
$$q_* = \emptyset$$

Łącząc powyższe wyniki otrzymujemy:



Zatem w ogólnym przypadku nie istnieje jednoznaczne rozwiązanie dla równania /14/. Oznacza to niejednoznaczność translacji zmian w przypadku elementarnego schematu rzutowania. Istnieje szczególny przypadek w którym translacja jest możliwa, a mianowicie gdy $P(X-Z)\bowtie u$ położone jest na granicy $q_{\sim}$. Wtedy w $P(X-Z)\bowtie u$ powinna być spełniona zależność funkcjonalna $Z \rightarrow X-Z$. Dla $P(X-Z)\bowtie u$ jest to możliwe tylko wtedy gdy $\|P(X-Z)\| = 1$. W tym przypadku wszystkie krotki z u mogą być połączone tylko z jedną wartością z $P(X-Y)$ co pozwala na jednoznaczne odtworzenie relacji z bazy danych.

W ogólnym przypadku aby otrzymać możliwość jednoznacznej translacji należy na relację r narzucić dodatkowe ograniczenia zgodności logicznej. Załóżmy przykładowo że w r spełniona jest zależność połączeniowa $\bowtie[Z\ ,\ X-Z]$ Oznaczmy przez t rzut relacji r na schemat X-Z , $t = \Pi_{X-Z}(r)$

Powyższą zależność połączeniową możemy opisać równaniem:

$$/16/ \quad q = \Pi_Z(q) \bowtie t$$

Rozwiązanie tego równania jest następujące:

$$P(Z) \diamond_{\emptyset} \bowtie t$$

Łącząc to rozwiązanie z rozwiązaniem równania 15.2 otrzymujemy:

$$\left(P(X-Z) \diamond_{\emptyset} \bowtie u \right) \bowtie \left(P(Z) \diamond_{\emptyset} \bowtie t \right) = u \bowtie t$$

co jest jednoznacznym rozwiązaniem. Zatem relację r możemy w tym przypadku odtworzyć ze wzoru $r = u \bowtie t$.

Niech schemat zewnętrzny będzie zdefiniowany jako suma dwóch relacji r i s.

$$/17/ \quad v = r \cup s$$

Po wykonaniu modyfikacji w v niewiadomymi stają się wystąpienia relacji r i s.

$$/18/ \quad u = q_1 \cup q_2$$

Prowadzi to do równań:

$$/19.1/ \quad u \sqsupset (q_1 \cup q_2) = \emptyset$$
$$/19.2/ \quad (q_1 \cup q_2) \sqsupset u = \emptyset$$

Powyższe równania rozwiązujemy w przestrzeni która jest iloczynem kartezjańskim przestrzeni relacji r i s.

Najmniejsze rozwiązanie równania /19.1/ nie istnieje, możemy jedynie znaleźć dolne ograniczenie:

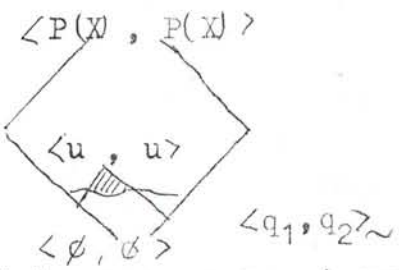
$$\langle q_1, q_2 \rangle_{\Pi} = \{\langle \alpha, \beta \rangle : \alpha \cup \beta = \cup \wedge \alpha \cap \beta = \emptyset\}$$
$$\langle q_1, q_2 \rangle^{\Pi} = \langle P(X),\ P(X) \rangle$$

Rozwiązanie równania /19.2/ jest następujące:

$$\langle q_1, q_2 \rangle_* = \langle \emptyset , \emptyset \rangle$$

$$\langle q_1, q_2 \rangle^* = \langle u , u \rangle$$

Łącząc rozwiązania otrzymujemy:



Wynika stąd , że jednoznaczne rozwiązanie w tym przypadku nie istnieje. Jednoznaczną translację można byłoby przeprowadzić tylko wtedy gdy $u = \emptyset$. Wtedy $r,s = \emptyset$.

Rozważmy teraz przypadek ,w którym do równania /18/ dodamy następujące dwa równania wynikające z logicznych ograniczeń zgodności narzuconych na relacje r i s.

/20/ $q_1 = \Gamma_c(q_1)$

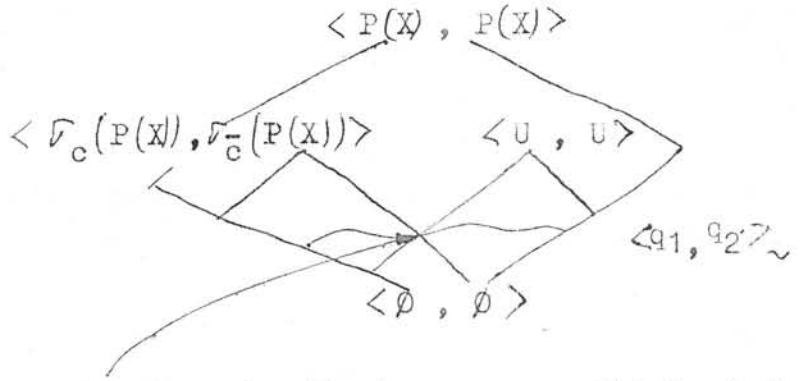
/21/ $q_2 = \Gamma_{\bar{c}}(q_2)$

Rozwiązania równań /20/ i /21/ są następujące:

$$q_1^* = \Gamma_c(P(X)) \qquad q_2^* = \Gamma_{\bar{c}}(P(X))$$

$$q_{1*} = \emptyset \qquad q_{2*} = \emptyset$$

Łącząc te rozwiązania z poprzednimi wynikami otrzymujemy:



$\langle \Gamma_c(P(X)), \Gamma_{\bar{c}}(P(X)) \rangle \cap \langle u , u \rangle = \langle \Gamma_c(u) , \Gamma_{\bar{c}}(u) \rangle \in \langle q_1 , q_2 \rangle_\sim$ gdyż

spełnione są warunki: $\Gamma_c(u) \cup \Gamma_{\bar{c}}(u) = u$

$\Gamma_c(u) \cap \Gamma_{\bar{c}}(u) = \emptyset$

Stąd otrzymujemy jednoznaczne rozwiązanie opisujące możliwą translację zmian:

$$\langle q_1 , q_2 \rangle^* = \langle \Gamma_c(u), \Gamma_{\bar{c}}(u) \rangle$$

Niech schemat zewnętrzny będzie zdefiniowany w następujący sposób:

$$v = r \cap s$$

Po wykonaniu modyfikacji w relacji v otrzymujemy relację u oraz równanie opisujące metodę odtworzenia relacji r i s.

/22/ $u = q_1 \cap q_2$

Stąd otrzymujemy równania:

/23.1/ $u \sqcup (q_1 \cap q_2) = \emptyset$

/23.2/ $(q_1 \cap q_2) \sqcup u = \emptyset$

Rozwiązanie równania /23.1/ jest następujące:

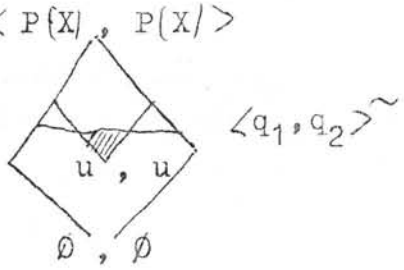
$$\langle q_1, q_2 \rangle^{*} = \langle P(X), P(X) \rangle$$

$$\langle q_1, q_2 \rangle_{*} = \langle u, u \rangle$$

Największe rozwiązanie równania /23.2/ nie istnieje, możemy jedynie podać górne ograniczenie:

$$\langle q_1, q_2 \rangle^{\sim} = \{ \langle \alpha, \beta \rangle : \langle \alpha \cup u, \beta \cup u \rangle \wedge \alpha \cap \beta = \emptyset \}$$

Po połączeniu wyników otrzymujemy:



Wynika stąd, że jednoznaczne rozwiązanie równania /22/ nie istnieje. Jeżeli zapamiętamy usunięte części relacji r i s i założymy, że muszą się one zawierać w q_1 i q_2 możemy sformułować następujące dodatkowe równania:

/24/ $r' \sqcup q_1 = \emptyset$

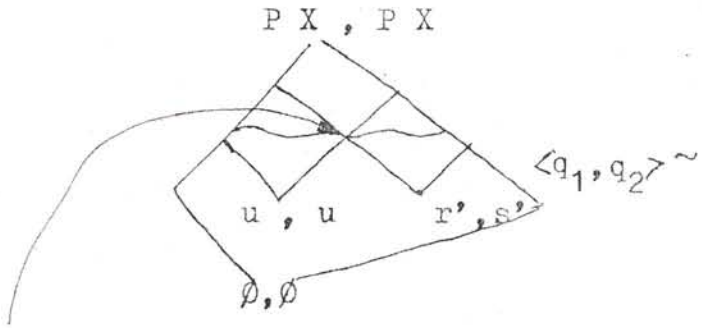
/25/ $s' \sqcup q_2 = \emptyset$

Rozwiązania tych równań są następujące:

$$q_1^{*} = P(X) \qquad q_2^{*} = P(X)$$

$$q_{1*} = r' \qquad q_{2*} = s'$$

łącząc te rozwiązania z rozwiązaniami równania /22/ otrzymujemy:



$\langle u \cup r' , u \cup s' \rangle \in \langle q_1, q_2 \rangle^{\sim}$ gdyż $r' \cap s' = \emptyset$

Zatem jednoznaczne rozwiązanie istnieje i jest następujące:

$$\langle q_1 , q_2 \rangle^{\circ} = \langle u \cup r' , u \cup s' \rangle$$

Powyższy wzór opisuje jednocześnie sposób odtworzenia relacji r i s z bazy danych.

Bibliografia

[BaSp81] F.Bancilhon, N.Spyratos Update semantics of relational view ACM Trans. on Database Systems , vol. 6 , no. 4 , 1981

[CaAr79] C.R.Carlson , A,K.Arora The updatability of relational views based on function de endencies , 3rd Intl. Comp. Conf. Software and Applications ., 1979

[Ch81] C,L.Chang On evaluation of queries containing derived relations in a relational databases , w Advances in Data Base Theory ed. H.Gallaire , J.Minker , J.M.Nicolas , Plenum Press 1981

[CoPa84] S.S.Cosmadakis, C.H.Papadimitriou Updates of relational views Journal of ACM vol. 31 no. 4 , 1984

[DaBe78] U.Dayal , P.A.Bernstein Updatability of relational views , Proc 4th VLDB Conf. W.Berlin , 1978

[DaBe82] U.Dayal , P.A.Bernstein On the correct translation of update operations on relational views , ACM Trans. on Database Systems vol. 7 , no. 3 , 1982

[FSD79] A.L.Furtado, K.C.Sevcik , C.S.Dos Santos Permitting updates through views of data bases , Information Systems , vol. 4 , no. 4 , 1979

[HeNa84] L.J.Hensen , S.A.Nagvi On compiling queries in recursive first-order databases , Journal of ACM , vol. 31 , no. 1 , 1984

[Ke81] A.M.Keller updates to relational database through views involving joins , IBM Res. Rep. RJ3282 . October 1981

[Kl52] S.L.Kleene Introduction to methamatematics , North Holland , 1952

[Ma84] Y.Masunaga A relational database view update translation mechanism, Proc 10th VLDB Conf. Singapore 1984

[Os80] I.M.Osman Updating relational views, an informal approach, RAIRO Informatique , Comp. Sci. vol. 14 , no. 4 , 1980

[Sp80] N.Spyratos Translation structures of relational views , Proc 6th VLDB Conf. Montreal 1980

[Ta55] A.Tarski A lattice theoretical fixpoint theorem and its application . Pacific Journal of Math. vol. 5 , no. 2 , 1955

[Ma83] D.Maier The theory of relational databases, Comp. Sci. Press 1983

ON FIRST ORDER LOGIC DATABASES

by

Henryk Rybiński *

---

* Istitute for Scientific Technical and Economic Information
Warsaw, Zurawia 3/5 , .Poland

ABSTRACT

The use of first order logic as database logic is shown to be powerful enough for formalizing and implementing not only relational but also hierarchical and network type databases. It enables to treat all the types of databases in a uniform manner. The discussion focusses on the database language for heterogeneous databases. The language is shown to be general enough to specify constraints specific for particular type of database. It means that a specification of a database type can be "translated" to the specification in the database language, creating a "logical environment" for different views that can be defined by users. By the fact that any database schema is seen as a first order theory expressed by a finite set of sentences the problems concerned with completness and compactness of database logic discussed in [18] are avoided.

Categories and Subject descriptors: F.4.1 [Mathematical Logic and Formal Languages] ; H.2.1. [Database Management] : Logical design - data models; H.2.3 [Database Management]: Languages - data manipulation languages

1. Introduction

One of the major functions of database systems is to describe a part of reality in such a way that manipulations on data structures enable database users to gain information about the reality. Tools which are applied to model a reality are different in various types of databases. One of the basic problems in designing the tools for modelling of data is to choose such mechanisms which are simple enough and clear in construting data structures being conceptually a natural reflection of the described reality. The problem of modelling reality in data base systems has been one of the main subjects of database researches over the past 15 years. One of the most intensively studied solutions is the Codd's relational approach [10]. The simplicity and homogeneity of data structures are ones of the advantages of this approach. Another chief benefit derived from working with relational approach is that it can be closed in the formalism of first order predicate logic [22,15]. As a result many important topics (such as database modelling [11,2], proving of program correctness [6], deduction augmented databases [14,19,20] can be treated mathematically.

On the other side most systems today are , in general, not relational. Some of them , like hierarchical IMS [17] or CODASYL standard databases [12] are in wide use for traditional reasons, other new non-relational proposals based on the entity-relationship approach are still appearing ([8,4,28,16,3])

All these new proposals seem to be justified by the fact, that the entity-relationship approach looks to be more characteristic of the way humans model the world.

Because of the variety of modelling tools there exists a need for a formalism that will capture all major approaches. This is in fact the idea of the ANSI/SPARC proposals [1]. The essential progress in this direction has been made by Jacobs [18]. However, the formalism applied in [18] is based on the second order predicate calculus. Additionally, some problems concerned with this approach refer to the fact that for some database schema the completness and compactness of the database logic fail. This gives rise to restrictions in applying the proposed formalism for non-CODASYL network databases,because some properties of databases specified by schemas of a kind are not expressible in the database language.

In the presented paper the first order predicate calculus is shown to be appropriate enough for applying for a wide range of database types. A database schema is defined here as a set of sentences in a database language. The notion of "database schema" plays the same role as the notion of "theory" in predicate calculus. A model of this "theory"closely corresponds to a database, which can be seen at any instant of time as a mathematical object. It is worthy of note that for the typical database types (relational, hierarchical and CODASYL-like) one can define, using the database language, some sets of formulas which specify constraints specific to particular type of the database. The set of axioms which determine a type of a database is called "characteristic set" for given type. So, what is losely under-

stood as a database model becomes in this approach a class of mathematically defined "semantical models". The characteristic set of formulas can be attached to a user-specified schema ("view") every time when a particular type of the database is "declared". Hence, such a "declaration" provides a "logical environment" for a particular type of database. It gives a posibility of controling the consistency of given application with the assumed database model.

By using the first order predicate calculus we avoid in this approach the problems concerned with completness and compactness of the database logic presented in[18]. Additionally a widely penetrated domain of automatic theorem proving [7] can be used for a large class of the database systems. Some of the applications concerned with this domain can be analogical to these ones proposed in [19,22], extended over other types of databases another ones can result from heterogeneous database environment for which this formalism is shown to be appropriate.

An outline of this paper is as follows.

In Section 2 we introduce the notions of a database domain, and a database in the domain. The notion of database domain corresponds to the notion of database schema in some simple DBMS implementations. Here, we see it close to an intensional specification of objects and attributes to be used for object descriptions. The notion of a database can be seen as a real occurence of a set of objects having assigned values to particular

attributes. It is close to object-attribute approach of Pawlak information system [23]. In fact it can be seen as an extension of Pawlak information retrieval system idea over the database area.

In Section 3 a database language and its logical realization are defined. A database schema is a set of formulas (sentences) which specify integrity constraints imposed on the database. All constraints are shown to be expressible in the same language: the ones resulting from choosing a particular type of a database (i.e. relational, hierarchical or CODASYL-like), and those resulting from particular application. It is shown that a database can be seen in terms of logic as a structure in a realization of the database language. Finally some general properties of the database logic are discussed.

2. Database domains and databases

The notion of a database, introduced below, is close to Pawlak idea of information system [23]. We attempt to start from the notions of constructs, which seem to be conceptually natural to be employed by designers when they start thinking of a DBMS application. The primitives used here are objects and attributes. Indeed, independently of what kind of database model is assumed to be used, a designer in his first approach to an application distinguishes a universum $\mathfrak{X}$ of objects which are planned to be describable in the database. Usually such a

universum is considered by the designer as consisting of heterogeneous sets of objects. Therefore a number of types of objects are distinguished. For each type of objects in $\mathbf{X}$ some attributes have to be predetermined. To each attribute a particular meaning is assigned. Losely speaking, by specifying a universum $\mathbf{X}$, its distribution over sets of different types X_t, $t \in T$, and assigning to each type $t \in T$ a set of attributes A_t we say that a conceptual domain of a database is defined.

Definition 1 . A domain of a database is an ordered quadruple $D = \langle \mathbf{X}, T, \{A_t\}_{t \in T}, \{\mathbf{V}_a\}_{\substack{a \in A_T \\ t \in T}} \rangle$ where :

- $\mathbf{X}$ is a universum of objects being the subject of a database application;
- T is a finite set of names of object types in X:
- for any $t \in T$ the set $A_t = \{a_{t_1}, a_{t_2}, \ldots a_{t_k}\}$ is a set of attributes names; A_t is said to be a characteristics of objects from X_t (of the type t);
- $\mathbf{V}_a$ is a set of possible values of the attribute a , $a \in A_t$ (the attribute domain).

Below we denote by A_D a set of all attributes in the database domain D, i.e. $A_D = \bigcup_{t \in T} A_t$. Additionally, we say that the set $X \cup \bigcup_{a \in A_d} \mathbf{V}_a$ is the universum of the database domain, and we denote it by U_D.

A database domain can be understood as a formal specification of the notion of conceptual view. A particular database domain determines a class of databases. A particular database in the domain D can be specified by choosing a finite set of objects and assigning to the objects some values of particular attributes. In the definition above we do not specify any restrictions for particular sets $\mathbb{V}_a$ of attributes values. In some cases it has however the essential meaning for properties of the database domains. Especially by imposing some restrictions on sets $\mathbb{V}_a$ one can specify classes of the domains. Therefore two kinds of attributes are distinguished below:

Definition 2. We say that an attribute $a \in A_D$ is nonterminal iff there is a type $t \in T$ such that $\mathbb{V}_a = \mathbb{X}_t$; the attribute $a \in A_D$ is the terminal one iff $\mathbb{V}_a \cap \mathbb{X} = \emptyset$.

In the paper we consider the database domains of which attributes are either terminal or nonterminal. To illustrate the notions defined above let us consider the following example of a database domain:

Example 1. We describe a database domain for an application that can be designed for needs of a university; so, in the universum $\mathbb{X}$ we distinguish objects of the types STUDENT, LECTURE, PROFESSOR, e.t.c. For these types the following characteristics can be assumed:

$A_{STUDENT} = \{ S\_AGE, S\_NAME, S\_SEX, S\_DEPT, LECT\_ATT \}$

$A_{PROFESSOR} = \{ P\_AGE, P\_NAME, P\_SEX, LECT\_DEL \}$

$A_{LECTURE} = \{L\_NAME, LEVEL, HOURS, REF\_LECT\}$

For the considered characteristics we specify:

$\mathbb{V}_{S\_AGE} = \{ x \in INTEGER : x \geq 16 \ \& \ x \leq 30 \}$

$\mathbb{V}_{P\_AGE} = \{ x \in INTEGER : x \geq 25 \ \& \ x \leq 60 \}$

$\mathbb{V}_{S\_SEX} = \mathbb{V}_{P\_SEX} = \{ 'M', 'F' \}$

$\mathbb{V}_{LECT\_ATT} = \mathbb{V}_{LECT\_DELIV} = \mathbb{V}_{REF\_LECT} = \mathbb{X}_{LECTURE}$

$\mathbb{V}_{LEVEL} = \{'I', 'II', 'III'\}$

As one can easily notice the attributes LECT_ATT, LECT_DELIV, REF_LECT are nonterminal. All the other attributes are the terminal ones.

By imposing on a database domain certain special restrictions we can define different classes of database domains. In particular we define below the classes of relational, hierarchical and CODASYL-like domains. To do this, first let us introduce for a domain $D = \langle \mathbb{X}, T, \{A_t\}_{t \in T}, \{\mathbb{V}_a\}_{\substack{a \in A_t \\ t \in T}} \rangle$ a notion of transition graph $G_D = \langle N, \rightsquigarrow \rangle$ where $N = A_D \cup T$ is the set of nodes, and $\rightsquigarrow$ is defined as follows:

(1) $t \rightsquigarrow a$ iff $a \in A_t$ in D ;

(2) $a \rightsquigarrow t$ iff $\mathbb{V}_a = \mathbb{X}_t$ in D (i.e. the attribute a is nonterminal, and its domain is defined in $\mathbb{X}_t$) ;

(3) no other pairs $x, y \in N$ are in the relation $\rightsquigarrow$.

As one can notice , there is no transition from the nodes representing terminal attributes.

Definition 3. A database domain $D = \langle \mathbb{X}, T, \{A_t\}_{t \in T}, \{v_a\}_{\substack{a \in A_t \\ t \in T}} \rangle$ is relational iff all the attributes from A_D are terminal.

The class of relational database domains will be denoted by RD. Roughly speaking - to describe a relational domain we have to specify types of objects (relations names) and for each type t - a characteristics A_t, consisting of terminal attributes.

Fig.1

Definition 4. A database domain $D = \langle X, T, \{A_t\}_{t \in T}, \{\mathbb{V}_a\}_{\substack{a \in A_t \\ t \in T}} \rangle$ is hierarchical if:

(1) for any $t \in T$ not $t \overset{*}{\rightsquigarrow} t$, where $\overset{*}{\rightsquigarrow}$ denotes the closure of the relation $\rightsquigarrow$ in the transition graph G_D;
(2) any nonterminal attribute node $a \in A_D$ in the graph G_D is reached only by one edge.

Fig.2

The class of hierarchical database domains will be denoted by HD. One can easily notice that transition graphs for relational and hierarchical domains do not contain loops. In addition from the definitions 3 and 4 above the following corollary results directly:

Corollary 1. If a database domain D belongs to the class RD it belongs also to HD.

Finally, let us introduce the CODASYL-like database domain:

Definition 5. A database domain $D = \langle \mathbb{X}, T, \{A_t\}_{t\in T}, \{\mathbb{V}_a\}_{\substack{a\in A_t \\ t\in T}} \rangle$ is the CODASYL database domain iff for any pair $x,y\in N$ in the transition graph G_D for this domain $x \rightsquigarrow y$ implies not $y \rightsquigarrow x$.

The class of CODASYL database domains will be denoted by CD. The restriction from the definition above is a reflection of the fact that in CODASYL databases records of a given type can not create SET constructs with records of the same type. From the definitions 4 and 5 we have immediately:

Corollary 2. If a database domain belongs to the class HD it belongs also to CD.

In figures 1-3 examples of transition graphs of database domains are presented.

Fig 3.

As mentioned above, by determining a database domain one can roughly specify a class of databases over this domain. A particular database in this class consists of a set X of objects from the universum $\mathbb{X}$ and a set of functions $\{\varphi_{ta}\}_{\substack{a\in A_t \\ t\in T}}$ which assign to particular objects $x\in X$ the values of given attributes. To be more precise in defining the concept of database we introduce the notion of description function

$$\varphi_{ta}: \quad X_t \rightarrow 2^{V_a}$$

Analogically to [8] we have to distinguish two notions: the intension of the description function and the extension of it. The intension of a function φ corresponds to its intended meaning. The intension imposes some general properties on the function which should be expressed in a database language. On the other hand the extension of a function φ_{ta} is a mathematical function - defined on a fixed set of objects. Saying on the semantics of a function φ_{ta} we assume that if $x_1 \in \varphi_{ta}(x_2)$ then "the object x_2 of the type t has the a-property x_1". To be exact we have to add, however, what does mean $x_1 \notin \varphi_{ta}(x_2)$. In our approach we assume that "the object x_2 has not the a-property x_1". It closely corresponds to Reiter's "close world assumption" [25], which indeed is implicitely presupposed in most of practical implementations of database systems.

Definition 6. A database d in a domain $D = \langle \mathbb{X}, T, \{A_t\}_{t\in T}, \{V_a\}_{a\in A_t, t\in T} \rangle$ is a pair $\langle U^e, \{\varphi_{ta}\}_{\substack{t\in T \\ a\in A_t}} \rangle$, where $U^e = X \cup \bigcup_{a\in A_D} V_a$, and

φ_{ta} is an extension of the description function for $t\in T$, and $a \in A_t$, defined on X_t. For each $a \in A_D$ V_a is a set from the attribute domain $\mathbf{V}_a$.

In some cases it is convenient to consider relations between objects and attributes values rather than the functions φ_{ta}: we say that for a database $d = \langle U^e, \{\Phi_{ta}\}_{t\in T} \rangle$ there is $x_1 \overrightarrow{\varphi_{ta}} x_2$ iff $x_1 \in X_t$ and $x_2 \in \varphi_{ta}(x_1)$. By $\overset{a\in A_t}{\underset{\Phi}{\pm\!\!\rightarrow}}$ we denote

the closure of the set of relations $\Phi = \{\overrightarrow{\varphi_{ta}}\}$.

One can easily notice that this definition allows to construct a database in a domain of a given class, which is not appropriate for databases of given type. It means that requirements induced by the type of domain are not sufficient to assure the same type of databases in the domain. Therefore we specify below the requirements for classes of relational, hierarchical and CODASYL-like databases.

Definition 7 . A database $d = \langle U^e, \Phi \rangle$ is relational iff

(1) the domain D is relational;

(2) for any $t \in T$ and $a \in A_t$ card $(\varphi_{ta}(x)) = 1$

(3) for any type $t \in T$ objects in X_t are distinguishable by values of attributes i.e. for any pair $x_1, x_2 \in X_t$, $x_1 \neq x_2$, for at least one attribute $a \in A_t$ $\varphi_{ta}(x_1) \cap \varphi_{ta}(x_2) = \emptyset$

Using the terminology of [23](3) means that the relational database is a selective information system. It is obvious that the set X_t of objects corresponds to a set of tuples in usual understanding of relational databases.

Definition 8. A database $d = \langle U^e, \Phi \rangle$ in a domain D is hierarchical iff

(1) the domain D is hierarchical

(2) $x_3 \overset{*}{\overrightarrow{\Phi}} x_1$ and $x_2 \overset{*}{\overrightarrow{\Phi}} x_1$ imply $x_2 = x_3$

Directly from the definition above and the definition of hierarchical domain results that for no object in X the condition $x \xrightarrow[\Phi]{*} x$ can be fulfilled.

Definition 9. A database $d = \langle U^{\Theta}, \mathcal{C} \rangle$ in a database domain of any type is a CODASYL database iff:

(1) for any $t \in T$ and for any $x_1, x_2 \in X_t$ the condition $x_1 \xrightarrow{\varphi_{ta}} x_2$ can not be fulfilled;

(2) if for some $x_1, x_2 \in X$ there is $x_1 \xrightarrow{\varphi_{ta}} x_2$, then for any x_3 $x_3 \neq x_1$, the condition $x_3 \xrightarrow{\varphi_{ta}} x_2$ cannot be fulfilled.

One can notice that the condition (1) is satisfied if the domain D belongs to the class CD. The requirement (2) is a reflection of the restriction imposed on CODASYL databases, which says that a record can not occur as a MEMBER in two occurences of the same SET type (c.f.[12]).

From the definitions above one can conclude :

Corollary 3. The class of CODASYL databases covers the class of hierarchical databases , which subsequently covers the class of relational databases.

In Fig. 4 examples of various types of databases are presented as graphs. The nodes represent objects and attributes values, the edges are labelled by the attributes names. As one can see the relational database is represented as a set of disjoint subgraphs. All subgraphs are trees. The trees corresponding to the objects of the same type are

isomorphic to each other. In the case of the graph representing hierarchical database all subgraphs are still trees, but their heights can be arbitrary for different databases (however established for a particular database). More general graph can correspond to a CODASYL-like database. In this case cycles in the graph are allowed. Still, however, nodes representing objects of the same type can not link each other. Moreover, any node in the graph can be reached by no more then one edge with given label (c.f. (2), Def. 9).

Fig.

Restrictions of all considered above database types force, when designed a particular application, to introduce some artificial types of objects, what, in general, give rise to a redundancy of data contained in the database. Additionally, this causes some difficulties in understanding data structures and in manipulating them by application users. In particular the natural descriptional power of network entity-relationship approach has effected since a few years in an observable tendency of creating more general network-approached databases (c.f. [4,8,21,28,16]). In these proposals the restrictions of CODASYL databases are usually avoided. Additionally, as in case of relational databases the network databases can be provided with a database language ground on the first order predicate calculus. One of the languages of this type is presented in [16]. In the next section we define a database language which is not concerned with a real implementation. However, because of its independence of a model of database it

can be seen as a model language for applications in heterogeneous database environment.

3. Database language and its realization

The language presented below is based on first order predicate calculus. It is not oriented towards a particular type of a database domain. Particularly one can express in this language the special requirements which are satisfied by a database of particular type (relational, hierarchical or CODASYL-like). In our approach a database can be considered as an interpretation of some formulas given in this language. Let us first introduce the alphabet of the database language. In the alphabet we distinguish the following symbols:

A1. the logical symbols $\{\cdot, +, \sim, \rightarrow\}$ and the quantifiers ∇ (universal), and Δ (existential);

A2. countable set of variables $\mathfrak{X} = \{x^{(1)}, x^{(2)}, \ldots\}$

A3. the "path" predicate symbol Ψ^* (binary);

A4. the set of binary predicate $\{\Psi_{ta}\}_{\substack{t \in T \\ a \in A_t}}$

A5. the set of unary predicate symbols $\{\tau_t\}_{t \in T}$

A6. equality symbol "="

Some special symbols of the type COUNT, MEAN, MAX, MIN could be included to the alphabet as term symbols. For simplicity of our considerations we do not use them below. In the examples below we siplify notation, using for example the text STUDENT.AGE$(x^{(1)}, x^{(2)})$ rather then $\Psi_{\text{STUDENT,AGE}}(x^{(1)}, x^{(2)})$. Similar-

isomorphic to each other. In the case of the graph representing hierarchical database all subgraphs are still trees, but their heights can be arbitrary for different databases (however established for a particular database). More general graph can correspond to a CODASYL-like database. In this case cycles in the graph are allowed. Still, however, nodes representing objects of the same type can not link each other. Moreover, any node in the graph can be reached by no more then one edge with given label (c.f. (2), Def. 9).

Fig.

Restrictions of all considered above database types force, when designed a particular application, to introduce some artificial types of objects, what, in general, give rise to a redundancy of data contained in the database. Additionally, this causes some difficulties in understanding data structures and in manipulating them by application users. In particular the natural descriptional power of network entity-relationship approach has effected since a few years in an observable tendency of creating more general network-approached databases (c.f. [4,8,21,28,16]). In these proposals the restrictions of CODASYL databases are usually avoided. Additionally, as in case of relational databases the network databases can be provided with a database language ground on the first order predicate calculus. One of the languages of this type is presented in [16]. In the next section we define a database language which is not concerned with a real implementation. However, because of its independence of a model of database it

can be seen as a model language for applications in heterogeneous database environment.

3. Database language and its realization

The language presented below is based on first order predicate calculus. It is not oriented towards a particular type of a database domain. Particularly one can express in this language the special requirements which are satisfied by a database of particular type (relational, hierarchical or CODASYL-like). In our approach a database can be considered as an interpretation of some formulas given in this language. Let us first introduce the alphabet of the database language. In the alphabet we distinguish the following symbols:

A1. the logical symbols $\{\cdot, +, \sim, \rightarrow\}$ and the quantifiers ∇ (universal), and Δ (existential);

A2. countable set of variables $\mathfrak{X} = \{x^{(1)}, x^{(2)}, \ldots\}$

A3. the "path" predicate symbol Ψ^* (binary);

A4. the set of binary predicate $\{\Psi_{ta}\}_{t \in T, a \in A_t}$

A5. the set of unary predicate symbols $\{\tau_t\}_{t \in T}$

A6. equality symbol "="

Some special symbols of the type COUNT, MEAN, MAX, MIN could be included to the alphabet as term symbols. For simplicity of our considerations we do not use them below. In the examples below we siplify notation, using for example the text STUDENT.AGE$(x^{(1)}, x^{(2)})$ rather then $\Psi_{\text{STUDENT,AGE}}(x^{(1)}, x^{(2)})$. Similar-

ly, instead of writing $\tau_{STUDENT}(\alpha)$ we simply write STUDENT(α). The set of well formed formulas can be defined in a usual way (the formal definition is given in Appendix 1). So, we distinguish atomic formulas e.g. STUDENT(α), PERSON.AGE(α,'30') STUDENT.LECT_ATT(α, 'MATH') e.t.c. Using atomic formulas we can construct the quatifierless formulas, for example:

STUDENT(α) $\rightarrow \sim$ PROFESSOR(α)

PERSON.SEX(α,'M') $\rightarrow \sim$PERSON.SEX(α,'F')

By applying quantifiers we can specify more sophisticated formulas. If a formula F does not contain free variables, it is called sentence. Examples of sentences are given below:

$\forall\, \alpha^{(1)}$ PERSON($\alpha^{(1)}$) $\rightarrow \Delta\, \alpha^{(2)}$ PERSON.AGE($\alpha^{(1)}, \alpha^{(2)}$)

$\forall\, \alpha^{(1)}$ PROFESSOR($\alpha^{(1)}$) $\rightarrow \Delta \alpha^{(2)}$ LECTURE ($\alpha^{(2)}$) · PROFESSOR.LECT($\alpha^{(1)}, \alpha^{(2)}$)

Possibilities of using a database language of this kind as a query language for non-relational databases are shown in [16,28] Especially, the formulas with free variables are usefull for querying the database. The formulas being sentences express some properties of a database and therefore play the essential role in formulating database shemas, user views, translation schema between databases (cf [18]). Below we define a notion of realization of the database language. We do that in a style of [24].

Definition 10. We say that a mapping R is a realization of the language L_D in the Boolean algebra $B = \langle \{\mathbb{0}, \mathbb{1}\}, \vee, \wedge, \Rightarrow, \sup, \inf \rangle$ if for any database $d = \langle U^e, \Phi \rangle$ in the domain D it assigns:

(1) to the predicate symbol Ψ^* the function $\Psi^{*(R)}$

$$\Psi^{*(R)} : (U^e)^2 \longrightarrow \{\mathbb{0}, \mathbb{1}\}$$

such that

$$\Psi^{*(R)}(x_1, x_2) = \begin{cases} \mathbb{1} & \text{iff } x_1 \overset{*}{\underset{\Phi}{\rightarrow}} x_2 \\ \mathbb{0} & \text{otherwise} \end{cases}$$

(2) to each predicate symbol τ_t, $t \in T$, the function $\tau_t^{(R)} : U^e \rightarrow \{\mathbb{0}, \mathbb{1}\}$ such that

$$\tau_t^{(R)}(x) = \begin{cases} \mathbb{1} & \text{iff } x \in X_t \\ \mathbb{0} & \text{otherwise} \end{cases}$$

(3) to each predicate symbol Ψ_{ta}, $t \in T$, $a \in A_t$, the function $\Psi_{ta} : U^e \times U^e \longrightarrow \{0,1\}$ such that

$$\Psi_{ta}^{(R)}(x_1, x_2) = \begin{cases} 0 & \text{iff } x_1 \notin X_t \ \underline{\text{or}} \ x_2 \notin \varphi_{ta}(x_1) \\ \mathbb{1} & \text{otherwise} \end{cases}$$

(4) to the symbol "=" a function which is a realization of identity relation.

We can extend this definition to realizations of formulas in a standard way (Appendix 1). According to this extension realizations of sentences do not depend on valuations of variables. We say that a realization R is a <u>semantic model</u> of the sentence F iff $F^{(R)} = \mathbb{1}$. In this case we also say that the formula F is valid in R. Moreover, we say that the database $d = \langle U^e, \Phi \rangle$ is an interpretation of the sentence F. Any sentence in L_D determines a set of databases, which are interpretations of this sentence in some realizations. Therefore it is quite natural to use a set of sentences for specifying a space of databases being interpretations of these in-

terpretations. To be more precise, we say that a realization R is a semantic model of a set of sentences $\mathbb{F}$ iff this realization is a model of each sentence from $\mathbb{F}$. We write $\mathbb{F} \models F$ iff any model of $\mathbb{F}$ is also the model of F.

Definition 11. A pair $S = \langle L_D, \mathbb{F} \rangle$ is a database schema iff the set $\mathbb{F}$ is a nonempty, finite set of sentences given in the database language L_D.

The formulas from $\mathbb{F}$ are called the integrity constraints of given database. They play role of axioms of the database. A very important implication concerned with applying the database language refers to a possibility of using a formal deductive system FDS , which in some cases can be used for controlling consistency of database schemas, and for avoiding redundant formulas in schemas. It is in real an extension of some researches concerned with relational databases over other types of databases. As we do not go beyond the first order predicate calculus the formal deductive system can be defined in a usual way as a system applying logical axioms, equality axioms (cf.[24]),and inference rules. In practice , results of theorem proving using the Resolution Principle ([26]) can be used. However, additional set of axioms should be taken into account in a deduction process. These axioms are specific for the described here database logic, and result from the definition of realization of the database language L_D:

AX1: $\forall x^{(1)} \forall x^{(2)} \Psi_{ta}(x^{(1)}, x^{(2)}) \rightarrow \Psi^*(x^{(1)}, x^{(2)})$

AX2: $\forall x^{(1)} \forall x^{(2)} \forall x^{(3)} \; \Psi^*(x^{(1)}, x^{(2)}) \cdot \Psi^*(x^{(2)}, x^{(3)}) \rightarrow \Psi^*(x^{(1)}, x^{(3)})$

AX3: $\forall x^{(1)} \forall x^{(2)} (\Psi_{ta}(x^{(1)}, x^{(2)}) \rightarrow \tau_t(x^{(1)}))$

The axiom AX1 results from the definition of realization of the predicate symbol Ψ^* and from the definition of relation $\xrightarrow[\Phi]{*}$ The axiom AX2 is a reflection of transitivity property of the relation $\xrightarrow[\Phi]{*}$. The axiom AX3 results directly from the definition of the realization of the symbols Ψ_{ta}, and τ_t, $t \in T$, $a \in A_t$. The additional axioms, defined below, can be useful in the case when the types of a database universum $\mathbb{X}$ are not disjoint and inclusions between the types are allowed (c.f. 28,16)

AX4: $(\forall x^{(1)} \tau_{t_1}(x^{(1)}) \rightarrow \tau_{t_2}(x^{(2)})) \cdot (\forall x^{(1)} \tau_{t_2}(x^{(1)}) \rightarrow \tau_{t_3}(x^{(1)})) \rightarrow$

$(\forall x^{(1)} \tau_{t_1}(x^{(1)}) \rightarrow \tau_{t_3}(x^{(1)}))$

AX5: $\left(\forall x^{(1)} \tau_{t_1}(x^{(1)}) \rightarrow \tau_{t_2}(x^{(1)})\right) \rightarrow \left(\bigsqcap_{a \in A_{t_2}} \forall x^{(2)} \forall x^{(3)} \Psi_{t_2 a}(x^{(2)}, x^{(3)}) \rightarrow \right.$

$\left. \Psi_{t_1 a}(x^{(2)}, x^{(3)})) \right)$

As one can observe, the axiom AX4 reflects the transitivity property of the relation "$\subset$". The axiom AX5 is a result of the natural assumption that for the types $t_1, t_2 \in T$, such that $\mathbb{X}_{t_1} \subset \mathbb{X}_{t_2}$, the description functions $\varphi_{t_1 a}$ and $\varphi_{t_2 a}$ are the same in the domain $\mathbf{X}_{t_1}$.

One of the most important features of the database language described above is the possibility of expressing properties of databases of different types, and hence treating them unifor-

mly. It makes possible to compare different views in different types of databases and create views which refer to a number of different databases. An example of such approach is presented in [3] .

We show below that the database language allows to express properties specific for particular domain.

Definition 12. A finite and consistent set of sentences $\mathbb{F}_D$ is the characteristic set for given database domain D iff:

(1) for any semantical model of the schema $S_D = \langle L_D, \mathbb{F}_D \rangle$ the nonempty database $d = \langle U^e, \Phi \rangle$ in this model is in the domain D (i.e. any semantical model is consistent with the domain specification);

(2) for any other database schema $S' = \langle L_D, \mathbb{F}' \rangle$ if there is a semantical model R for this schema for which the database d_S, is in the domain D then R is also a model of the schema S_D.

(3) no proper subset of $\mathbb{F}_D$ holds (1) and (2).

Existing of characteristic sets of formulas for the considered types of databases may be very attractive from the point of vie of database implementator . It would provide natural and simple facilities of realizing a universal database system which could cover different types of databases. For end-users particular database could be seen by the logical database language interface. The appropriate characteristic set $\mathbb{F}_C$ of sentences could be attached to the user specifications every time when a particular type of a database is declared. As shown in [27] for any considered type of database domain one can choose

a set of sentences, which is characteristic for given domain. For example, the specification LECT_ATT $\rightsquigarrow$ LECTURE for the domain D from Example 1, which says that the attribute LECT_ATT is not terminal and its domain is $\mathbb{X}_{LECTURE}$, can be formulated in the database language L_D as follows:

$$\nabla x^{(1)} \nabla x^{(2)} \; \text{STUDENT.LECT\_ATT}(x^{(1)}, x^{(2)}) \longrightarrow \text{LECTURE}(x^{(2)})$$

In a heterogeneous database environment it would be very attractive to have also facilities for checking whether a user specification given in a database language L_D does not go beyond a specified database type.

Definition 13. For a database language L a finite and consistent set of sentences $\mathbb{F}_c$ is the characteristic set for relational (hierarchical, CODASYL) type of databases iff it is characteristic set for any relational (hierarchical, CODASYL) type domain.

Theorem 1 [27] For any database language L_D there exist sets of sentences characterisic for relational, hierarchical and CODASYL databases.

Examples of characteristic sets for relational and CODASYL types of databases are presented in Appendix 2. So, a choice of particular type of database can be easily translated to a set of sentences (or clauses - if used for theorem proving) which attached to the formulas specific for given application constitute a "logical environment" of given database. It may be useful in checking whether user-defined formulas specific

for given application are consistent with a declared type of the database. Additional advantage can be derrived from the fact that the same language is used to express both database restrictions and user-specified restrictions , what enables to treat all specifications uniformly. Formally, the consistency of a schema can be seen twofold: we say that a schema $S = \langle L_D, \mathbb{F} \rangle$ is m-consistent if there exist a semantical model of this schema; or we say that the schema $S = \langle L_D, \mathbb{F} \rangle$ is f-consistent if for the formal logical system (FDS) taking into account logical axioms ,equality axioms , the axioms AX1 - AX5 and the sentences from $\mathbb{F}$ there exist no formula F such that F and $\sim F$ are provable in FDS (if a formula F is provable from $\mathbb{F}$ we write $\mathbb{F} \vdash F$). As in defining the database logic we do not go beyond the first order predicate calculus, all the main theorems of first order logic can be directly applied for the database logic. Hence, directly from the well known Gödel theorem [24] we have:

Theorem 2. If a database schema $S = \langle L_D, \mathbb{F} \rangle$ is f-consistent it is also m-consistent, and in a semantical model of this schema the set U^e of a database d is countable.

And conversely, it is well known that:

Theorem 3. If a database schema $S = \langle L_D, \mathbb{F} \rangle$ is m-consistent it is also f-consistent, and additionally for any sentence F $\mathbb{F} \models F$ iff $\mathbb{F} \vdash F$

In comparison with the approach presented in [18], here any se-

mantic model of a database schema is induced only by formulas from the schema given in the language L_D, rather then by constructs which are outside of the language. This enables to avoid the problems discussedin [18]which are concerned with completness and compactness of Jacobs database logic in general case(c.f. Cor.4.5 and 4.6 in [18]). It means that the database language presented here can be used for specifying schemas for network databases , where relations between objects of the same types are allowed. The theorems 2 and 3 assure us that also for these cases completness holds for the database logic defined above. However the existence of a finite semantic model of a database schema $S = \langle L_D, \mathbb{F} \rangle$ with a finite set of axioms $\mathbb{F}$ is in general case not so evident. To illustrate this fact let us consider two examples: the first one for a network database, the second one for relational database.

Example 2 . Let us consider a database schema $S^* = \langle L_D, \mathbb{F} \rangle$ with the following formulas in $\mathbb{F}$:

$F_1 = \forall\, x^{(1)} \Delta x^{(2)} \Psi^* (x^{(1)}, x^{(2)})$

$F_2 = \forall\, x^{(1)} \sim \Psi^* (x^{(1)}, x^{(1)})$

One can see from the formula F_2 and the definition of $\xrightarrow[\Phi]{*}$ (see also AX2) that in any semantic model of this schema $\xrightarrow[\Phi]{*}$ is an ordering relation on U^e . Additionally, from the definition of realization of L_D results that F_1 is valid in a realization R iff in this realization for any $x \in X$ there exist $x' \in X$ greater then x in the meaning of $\xrightarrow[\Phi]{*}$, what means that X is in this model infinite.

The next example refers to the relational database:

Example 3 . Assume, that a relational database has to contain a relation with attributes Ψ_1 and Ψ_2. Let us incorporate to a relational schema S_R the following formulas:

$$F_1 = \nabla x^{(1)} \nabla x^{(2)} \nabla x^{(3)} \Delta x^{(4)} (\Psi_1(x^{(1)},x^{(2)}) \cdot \Psi_2(x^{(1)},x^{(3)})) \longrightarrow \Psi_1(x^{(4)},x^{(3)})$$

$$F_2 = \nabla x^{(1)} \nabla x^{(2)} \sim (\Psi_1(x^{(1)},x^{(2)}) \cdot \Psi_2(x^{(1)},x^{(2)}))$$

$$F_3 = \nabla x^{(1)} \nabla x^{(2)} \nabla x^{(3)} \nabla x^{(4)} \nabla x^{(5)} \nabla x^{(6)} (\Psi_1(x^{(1)},x^{(4)}) \cdot \Psi_2(x^{(1)},x^{(5)}) \cdot \Psi_1(x^{(2)},x^{(5)}) \cdot \Psi_2(x^{(2)},x^{(6)})) \longrightarrow (\Psi_1(x^{(3)},x^{(4)}) \cdot \Psi_2(x^{(3)},x^{(6)}))$$

A part of a database being interpretation of this schema is given in a table form below. As one can see the formulas F_1, F_2 and F_3 force infinite expanding of this table.

Table 1

X	Ψ_1	Ψ_2
x_1	a	b
x_2	b	c
x_3	b	d
x_4	a	c
x_5	a	d
x_6	c	e
x_7	a	e
x_8	b	e
..	...	...
..	...	...

A definition of a schema which does not have any finite semantical model can be considered as errorneous. One can ask whether it is possible to detect errors of this kind in a general case. Basing on the fundamental results of 9 and 29 we can say immediately that generally the problem of existing a finite model is undecidable. However, one can define a subclass of formulas for which this problem is recursive (c.f.[13]). Possibilities of applying this subclass for specifying database schemas are discussed in [27].

5. Conclusions

As shown, the first order logic can be applied as the database logic for a wide range of database types: it is powerfull enough for formalizing and implementing not only relational , but also hierarchical and network databases, providing a tool for treating all the types of databases in a uniform manner.

The database language defined in the paper can be used in a heterogeneous database environment. It can be applied not only for specifying user constraints and his views, but additionally it can be used to express constraints resulting from using a particular type of database type. It means that a specification of the database type given with the objects and attributes names can be "translated" to the specifications in the database language L_D , creating a "logical environment" for different views that can be defined by users. So, from the practical point of view this kind of language can be used as a natural

end-user interface to databases of different types.

Additionally, as the language L_D is based on first order logic it is possible to explore well developped techniques of theorem proving for providing end-users with powerfull deduction tools in heterogeneous database environment.

APPENDIX 1

A1.1 Formal definition of well formed formulas

In the database language L_D a term is a variable or constant symbol (constant symbols are assigned uniquely to objects from U^e). The well formed formulas (wffs) are defined as follows:

(a) for each type $t \in T$ the expression $\tau_t(z)$ is a wff iff z is a term;

(b) for each type $t \in T$, and $a \in A_t$ the expression $\Psi_{ta}(z_1, z_2)$ is a wff iff z_1 and z_2 are terms;

(c) the expression $\Psi^*(z_1, z_2)$ is wff iff z_1 and z_2 are terms;

(d) if the expressions F_1 and F_2 are wffs then $\sim F_1$, $F_1 + F_2$, $F_1 \cdot F_2$, $F_1 \rightarrow F_2$ are wffs;

(e) for any variable symbol α, and F being wff the expressions $\nabla\alpha\, F$ and $\Delta\alpha\, F$ are wffs;

(f) no other constructs are wffs.

A1.2 Realization of well formed formulas

Let us denote by Arg(F) a set of all variables and constants occuring in the formula F. Any function $f \in U^{e\,Arg(F)}$ is called valuation of arguments in F, assuming that for any constant symbol c $f(c)=c$. A mapping

$$F^{(R)} : U^{e\,Arg(F)} \longrightarrow \{0,1\}$$

is the realization of the formula F iff the following holds:

if F is one of the atomic formulas $\Psi^*(z_1,z_2)$, $\Psi_{ta}(z_1,z_2)$, $\tau_t(z)$. then respectively:

$$\Psi^{*(R)}(z_1,z_2)[f] = \Psi^{*(R)}(f(z_1),f(z_2))$$

$$\Psi^{(R)}_{ta}(z_1,z_2)[f] = \Psi^{(R)}_{ta}(f(z_1),f(z_2))$$

$$\tau^{(R)}_t(z)[f] = \tau^{(R)}_t(f(z))$$

$$(z_1 = z_2)^{(R)}[f] = (f(z_1) = f(z_2))^{(R)}$$

If F is not atomic formula then $F^{(R)}$ is defined by induction as follows:

$$(F_1 + F_2)^{(R)}[f] = F_1^{(R)}[f] \vee F_2^{(R)}[f]$$

$$(F_1 \cdot F_2)^{(R)}[f] = F_1^{(R)}[f] \wedge F_2^{(R)}[f]$$

$$(F_1 \rightarrow F_2)^{(R)}[f] = \neg(F_1^{(R)}[f]) \vee F_2^{(R)}[f]$$

$$(\sim F)^{(R)}[f] = \neg(F^{(R)}[f])$$

where $\vee$, $\wedge$ and $\neg$ are the operators of the Boolean algebra $B = \langle \{0,1\}, \vee, \wedge, \neg, \sup, \inf \rangle$. Moreover :

$$(\nabla z\ F)^{(R)}[f] = \inf_{f_{x_o} \in Val_z} \{ F^{(R)}[f_{x_o}] \}$$

$$(\exists z\ F)^{(R)}[f] = \sup_{f_{x_o} \in Val_z} \{ F^{(R)}[f_{x_o}] \}$$

where sup and inf are the operators of upper and lower bounds (respectively) of the Boolean algebra B and Val_z is defined as a set of valuations f_{x_o}, $x_o \in U^e$, such that

$$f_{x_o}(s) = \begin{cases} f(s) & \text{if } s \neq z \\ x_o & \text{if } s = z \end{cases}$$

APPENDIX 2

A2.1 Characteristic sets of formulas for relational and CODASYL-like databases

Let $D = \langle \mathbb{X}, T, \{A_t\}_{t\in T}, \{\mathbb{V}_a\}_{a\in A_t,\ t\in T} \rangle$ be a database domain, and $d = \langle U^e, \Phi \rangle$ be a database in this domain. We specify below the characteristic sets of formulas for relational and CODASYL types databases.

AR Relational database axioms

Domain axioms

AR1 Type completness

$$\nabla x \sum_{t\in T} \tau_t(x)$$

AR2 Completness of characteristics for particular types $t \in T$:

$$\nabla x^{(1)} \nabla x^{(2)} \tau_t(x^{(1)}) \cdot \Psi^*(x^{(1)}, x^{(2)}) \rightarrow \sum_{a\in A_t} \Psi_{ta}(x^{(1)}, x^{(2)})$$

Database axioms

AR3 The attributes are terminal :

$$\nabla x^{(1)} \nabla x^{(2)} \nabla x^{(3)} \Psi_{ta}(x^{(1)}, x^{(2)}) \rightarrow \sim \Psi^*(x^{(2)}, x^{(3)})$$

AR4 Descriptions of every two objects in X_t are distinguish-

able [1] :

$$\nabla x^{(1)} \nabla \vec{v}^{(1)}_{A_t} \nabla x^{(2)} \nabla \vec{v}^{(2)}_{A_t} \Big(\prod_{a \in A_t} \Psi_{ta}(x^{(1)}, v^{(1)}_a) \cdot \Psi_{ta}(x^{(2)}, v^{(2)}_a) \cdot (x^{(1)} \neq x^{(2)}) \rightarrow \sum_{a \in A_t} v^{(1)}_a \neq v^{(2)}_a$$

AR5 Descriptions of the objects of any type t are in 1 NF [11]:

$$\nabla x^{(1)} \nabla v^{(1)} \nabla v^{(2)} \Psi_{ta}(x^{(1)}, v^{(1)}) \rightarrow (\Psi_{ta}(x^{(1)}, v^{(2)}) \rightarrow (v^{(1)} = v^{(2)}))$$

AC CODASYL database axioms

The axioms specifying type completness and distinction of particular types are the same as AR1 and AR2. The axiom specifying completness of characteristics is the same as AH4. Additionally we specify :

AC4 SET definition requirement :

$$\nabla x^{(1)} \nabla x^{(2)} (\Psi_{ta}(x^{(1)}, x^{(2)}) \rightarrow \sim \tau_t(x^{(2)}))$$

(SET construct can not be defined for records of the same type);

AC5 SET occurence axioms , $t \in T$ and $a \in A_t$ is nonterminal:

$$\nabla x^{(1)} \nabla x^{(2)} \nabla x^{(3)} (\Psi_{ta}(x^{(1)}, x^{(2)}) \cdot \Psi_{ta}(x^{(3)}, x^{(2)})) \rightarrow x^{(1)} = x^{(3)}$$

(no record can occur in two occurences of a SET of given type).

---

[1] To simplify notation in this formula we write $\nabla \vec{v}^{(i)}_A$ instead of writing $\nabla v^{(i)}_{a_1} \ldots \nabla v^{(i)}_{a_k}$ (assuming that $A = \{a_1, \ldots, a_k\}$)

REFERENCES

1. ANSI/X3/SPARC Study Group on DBMS, Interim Report, SIGMOD FDT Bull. 7, 2, 1975

2. Beeri, C., Bernstein, P.A., A sophisticated introduction to database normalization theory, Proc. 4th Int'l Conf. on VLDB, Berlin,1978

3. Brzeziński, Z., et al. UNIBASE - an integrated access to databases, 10th Int'l Conf. on VLDB, Singapoore, 1984.

4. Buneman, P., Frankel,R.E., FQL - a functional query language, Proc. ACM SIGMOD Conf., Boston, 1979

5. Cadiou, J.M., On semantic Issues in Relational Model of Data, Proc. Int'l Symp. on Math. Foundations of Comp. Sc., Gdańsk, Poland, 1976, Lect. Notes in Comp. Sc., Springer Verlag

6. Casanova, M.A., Bernstein, P.A., A logic of a relational data manipulation language, 6th Ann. ACM Symp. on Principles of Progr. Lang., San Antonio, 1979

7. Chang, C.L., Lee, R.C.T., Symbolic Logic and Mechanical Theorem Proving, Academic Press, New York, 1973

8. Chen, P.P.S., The entity - relationship model, towards a unified view of data, ACM TODS, vol.1, No.1, 1976

9. Church, A., Introduction to Mathematical Logic, Part 1, Ann. of Math. Studies, No.13, Princeton University Press, 1944

10. Codd, E.F., A relational model of data for large shared data banks, Comm. ACM, 13(6), 1970

11. Codd, E.F., Further normalization of a database relational model, in Database Systems, (ed.Rustin R.), Prentice Hall 1972

12. CODASYL Data Base Task Group report, ACM, New York, 1971.

13. Di Paola, R. : The Solvability of the Decision Problem for classes of proper formulas and related results, Joun. ACM, vol.20, No.1, 1973

14. Gallaire H. : Impacts of logic on Databases, Proc. of the 7th International Conf. on VLDB, 1981, Cannes, France

15. Gallaire, H., Minker, J., Nicolas, J. : Logic and Databases: A deductive Approach, Comp. Surv., vol.16, No.2, 1984

16. Getta J., Rybiński, H.: HOLMES: A deduction augmented Database Management System, Inform. Systems, Vol.9, No.2, 1984

17. IBM Corp.: Information Management System / Virtual Storage General Information Manual, IBM, Form No. GH20

18. Jacobs, B.E., On Database Logic, Journ. ACM, vol.29, No. 2, 1982

19. Minker, J., On deductive relational databases, Proc. of the 5th Int'l Conference on Collective Phenomena (ed. J.L. Lebowitz) , Ann. of the New York Academy of Sc., vol.10 1983

20. Minker, J., An experimental relational database system based on logic, In Logic and Databases (ed. H.Gallaire, J.Minker) , Plenum, New York, 1978

21. Munz, R., The WELL system: A multilevel database system based on the binary relationships and graph patern matching, Inform. Systems, (3), 1978

22. Nicolas, J.-M., First Order Logic Formalization for Functional, Multivalued and Mutual Dependencies, ACM SIGMOD Int'l Conference on Management of Data, 1978

23. Pawlak, Z., Information systems - theoretical foundations, Inform. Syst., vol.6, No.3, 1981

24. Rasiowa H., Sikorski R., The Mathematics of Metamathematics, Polish Ac. of Sc., Polish Sc. Publ., 1963, Warsaw

25. Reiter , R., On Closed World Databases, Logic and Databases, (ed. H.Gallaire and J.Minker), Plenum Press, New York, 1978

26. Robinson , G.A., Wos, L., Paramodulation and theorem proving in first order theories with equality, Machine Inteligence, vol.4, (ed. B.Meltzer and D.Mitchie).

27. Rybiński, H. : Logical aproach to database systems (to appear , in Polish)

28. Shipman D.W. : The functional data model and the language DAPLEX, ACM TODS, vol. 6, No. 1, 1981

29. Trachtenbrot, B.A. : Impossibility of an algorithm for the d the decision problem in finite classes, Dokl. Akad. Nauk SSSR, 70 ,1950, (translated in Amer. Math. Soc. Transl. Ser.2, Vol.23, 1963

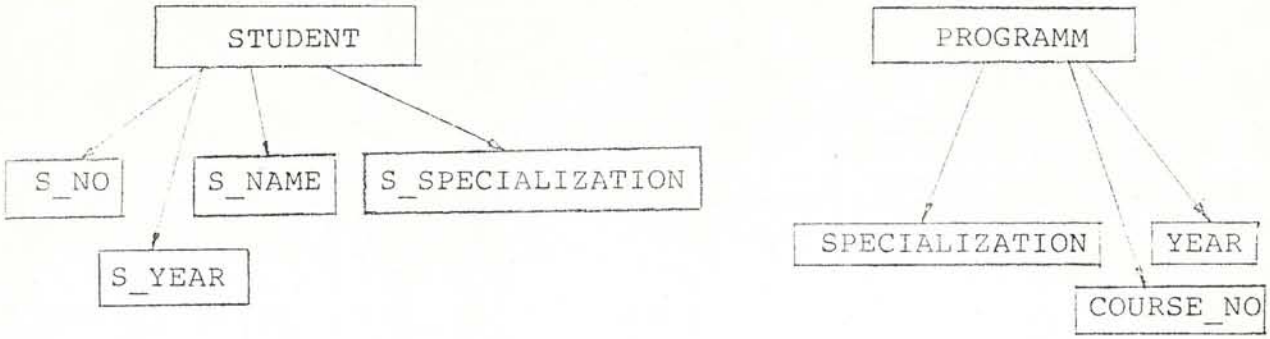


Fig 1. Relational database domain

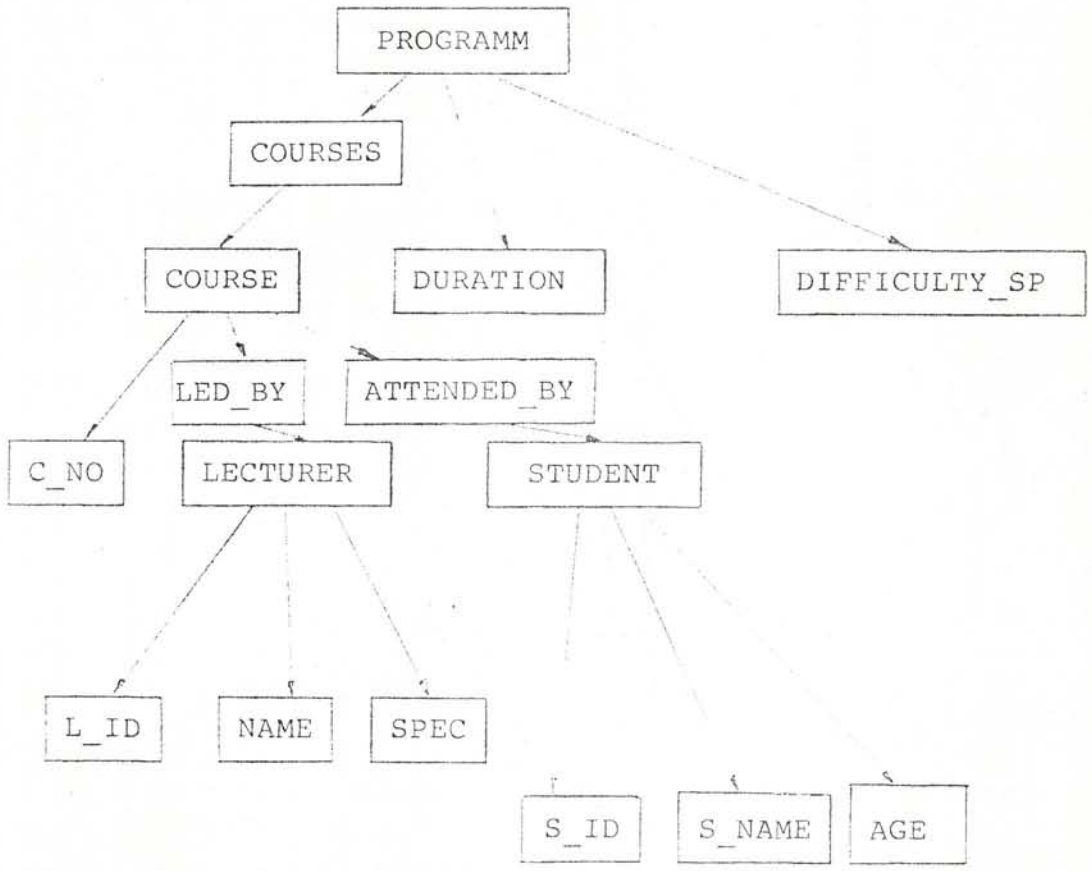


Fig 2. Hierarchical database domain

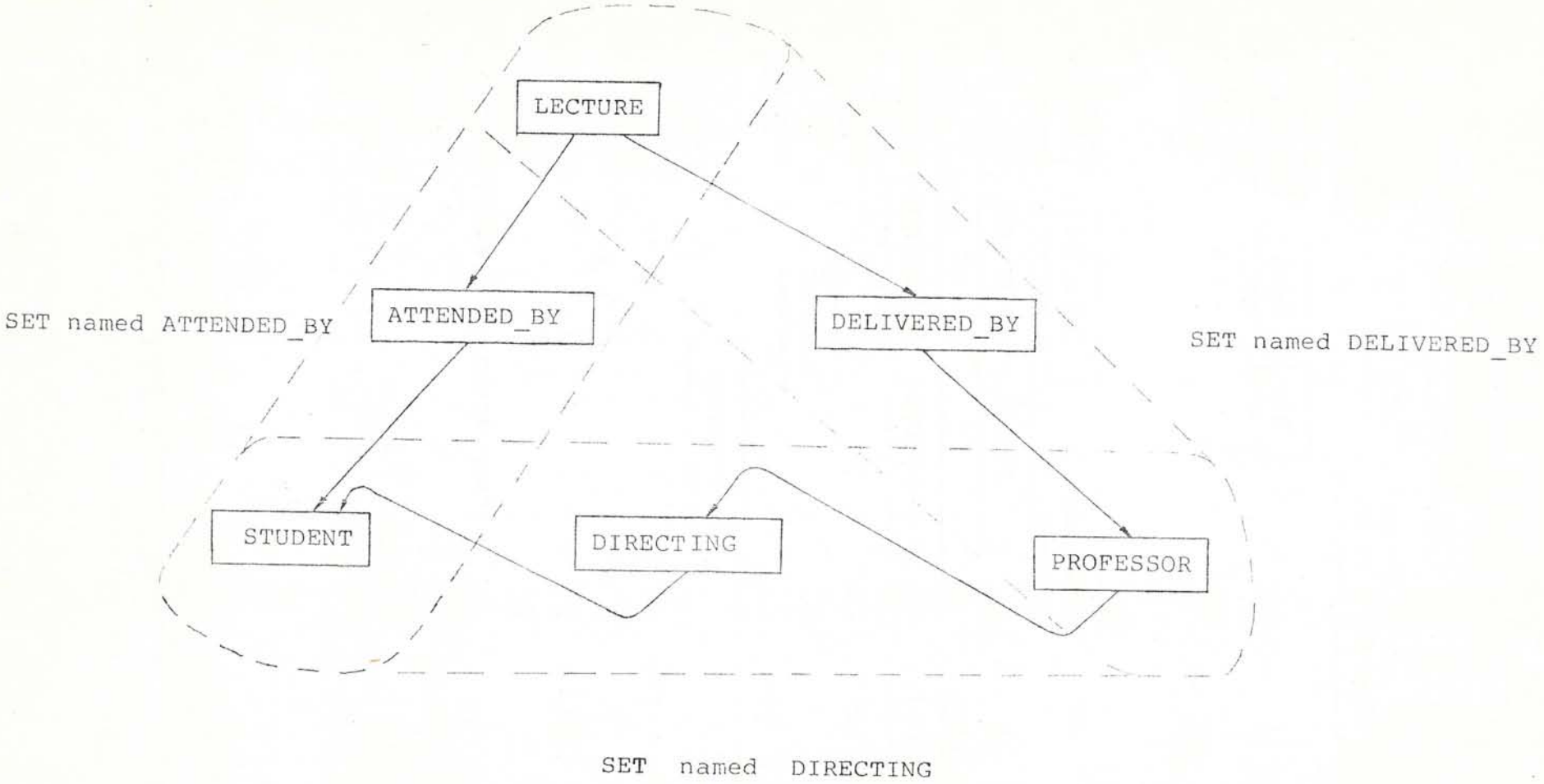
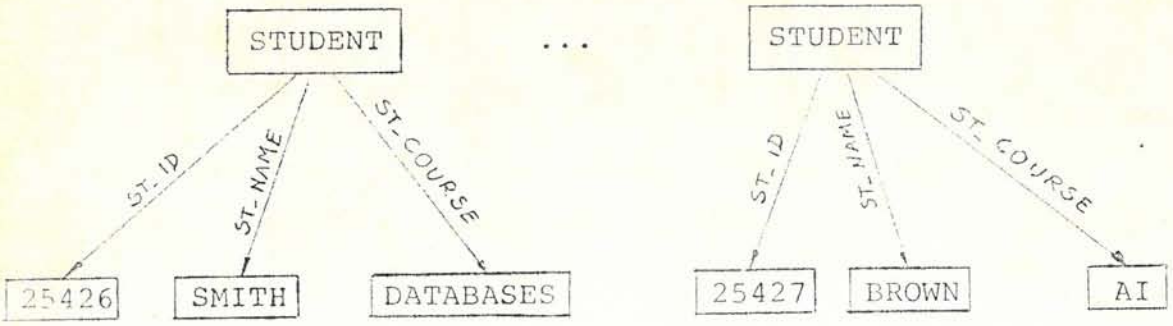
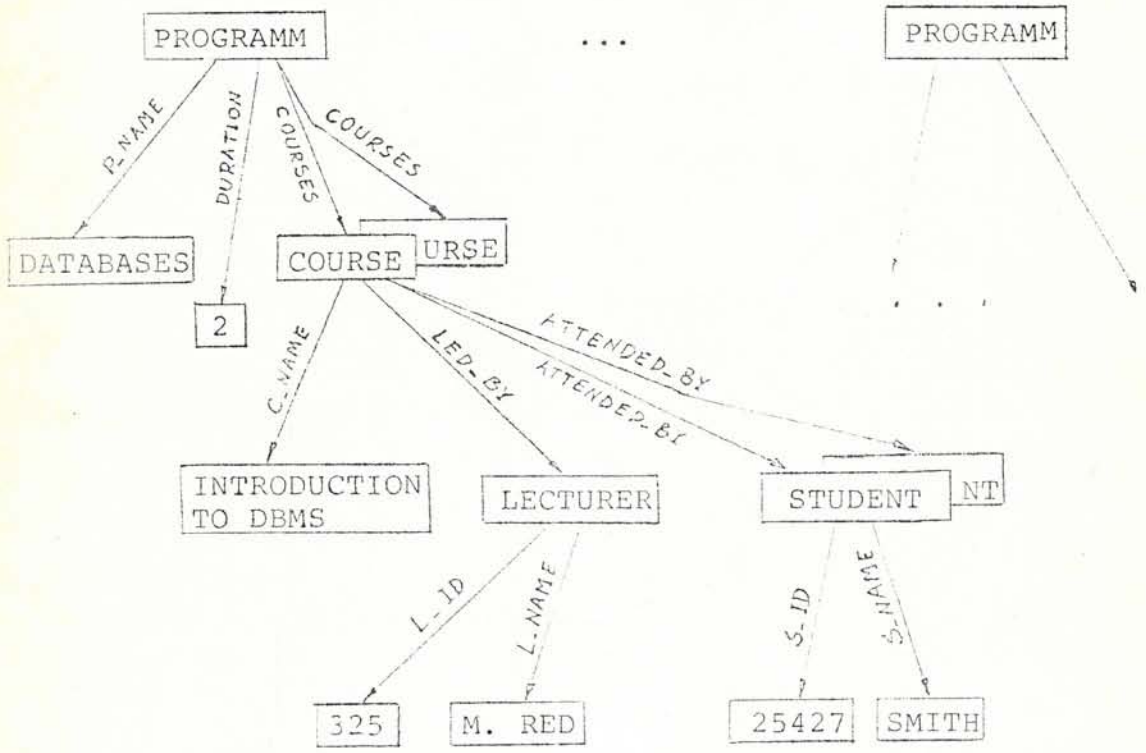


Fig. 3 A part of CODASYL database domain



A. A part of relational database



B. A part of hierarchical database

Fig. 4 Examples of databases