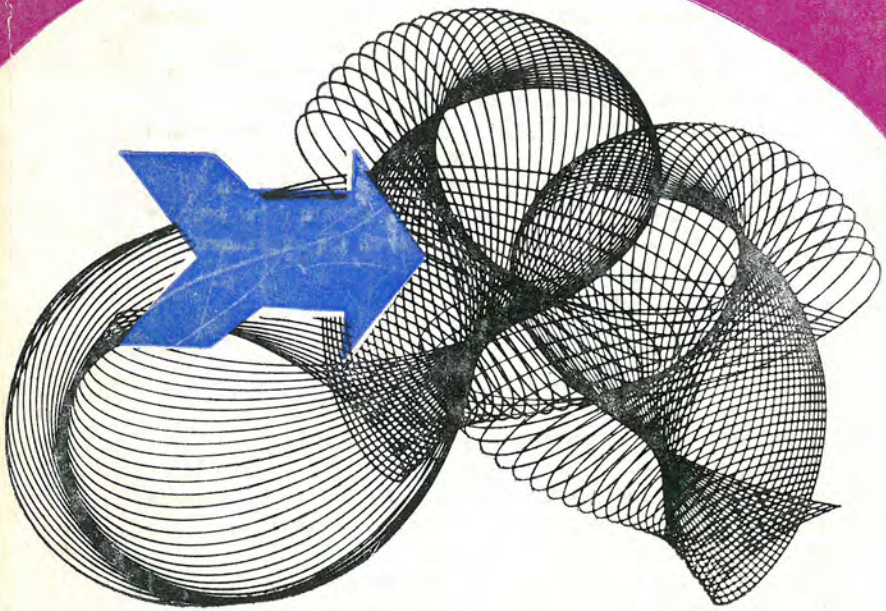
INFORMATYKA W PRAKTYCE



Józef Oleński
Witold Staniszkis

Projektowanie bazy danych

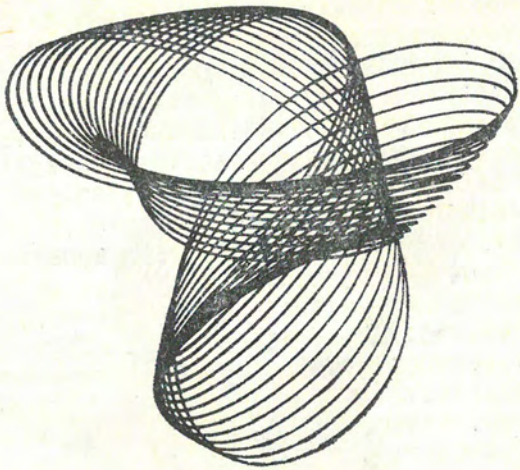
PWE

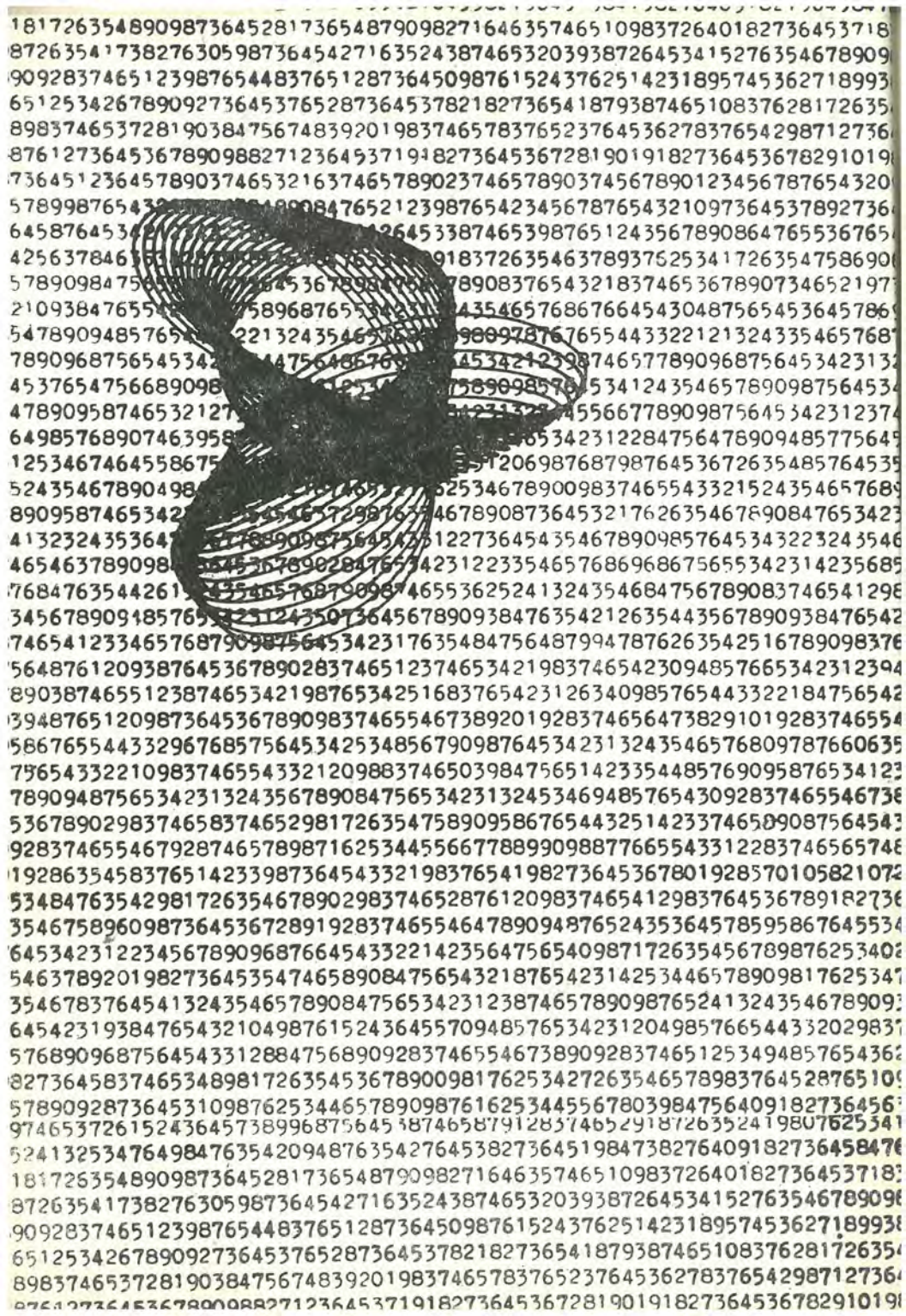
Seria wydawnicza „Informatyka w praktyce" zaspokaja potrzeby poznania literatury fachowej, poświęconej popularyzowaniu i wykorzystywaniu rozwiązań i zastosowań informatyki, a zwłaszcza budowania i funkcjonowania systemów informatycznych w jednostkach gospodarczych oraz doskonalenia sposobów uzyskiwania informacji w zarządzaniu. Poszczególne pozycje serii są przeznaczone dla pracowników służby informatycznej we wszystkich jednostkach gospodarczych, kadr kierowniczych tych jednostek, studentów wyższych uczelni, uczestników kursów podyplomowych oraz osób interesujących się zagadnieniami i rozwojem informatyki.

*

Rozwój metod planowania i zarządzania wymaga coraz sprawniejszej obsługi informacyjnej. Stąd też wiele organizacji gospodarczych podejmuje prace nad bazami danych, które — według ich oczekiwań — powinny zapewnić właściwą jakość tej obsługi. Metody projektowania tradycyjnych systemów informatycznych okazują się jednak niewystarczające i zawodne.
Autorzy postawili sobie za cel wypełnienie luki istniejącej w metodach projektowania systemów informatycznych opartych na wspólnej bazie danych. Przedstawiają w niej instrumenty i metody, jakimi może posługiwać się projektant systemu bazy danych w kolejnych fazach procesu projektowania: modelowaniu semantycznym, modelach pojęciowym, logicznym, fizycznym bazy danych. Zwracają także uwagę na odmienności między projektowaniem systemu opartego na bazie danych a tradycyjnym systemem przetwarzania danych.
Praca przeznaczona jest dla informatyków zainteresowanych praktycznymi aspektami analizy i projektowania systemów informatycznych opartych na bazie danych. Może też być wykorzystywana do celów szkoleniowych, a także przez studentów wyższych szkół ekonomicznych i technicznych.

7N





INFORMATYKA
W PRAKTYCE

Projektowanie bazy danych

Józef Oleński
Witold Staniszkis

Państwowe Wydawnictwo Ekonomiczne
Warszawa 1984

Komitet redakcyjny serii
INFORMATYKA W PRAKTYCE

JANUSZ GOŚCIŃSKI, TADEUSZ JAEGERMAN,
TADEUSZ PECHE (przewodniczący)
WŁADYSŁAW RADZIKOWSKI
TADEUSZ WALCZAK

Okładkę projektował
FRANCISZEK WINIARSKI

Redaktor
WANDA ROWICKA

ISBN 83-208-0350-0

Redaktor techniczny
WŁADYSŁAWA NASTERNAK
Korektor
TERESA RYBICKA
Printed in Poland
Państwowe Wydawnictwo Ekonomiczne
Warszawa 1984

Zlec. 129/70. Wyd. I. Nakład 2850+150 egz.
Ark. wyd. 16,2. Ark. druk. 19
Papier druk. sat. kl. IV, 70 g. Format 61×86/16.
Oddano do składania 16.V.1983 r. Podpisano do druku i druk ukończono w sierpniu 1984 r.
Cena zł 160, — T-69
Zakłady Graficzne w Katowicach, Zakład nr 2, zam. 6555-83.

Spis treści

Wstęp

Z postępu w dziedzinie technologii systemów informatycznych można by sądzić, że w nadchodzących latach dokona się zasadnicza zmiana w podejściu do projektowania systemów informatycznych. Zmiana ta, jak się wydaje, polegać będzie na coraz bardziej powszechnym wykorzystaniu baz danych jako narzędzi gromadzenia, przechowywania, przetwarzania i udostępniania informacji. Z dotychczasowych, stosunkowo skromnych i odcinkowych doświadczeń w tym zakresie wynika, że metody projektowania wypracowane dla systemów przetwarzania danych, zorientowane na przetwarzanie transakcyjne, nie wystarczają już do poprawnego zaprojektowania systemu opartego na bazie danych. Nie dostarczają one bowiem instrumentów niezbędnych do analizy i określenia treści bazy danych, przechowywania i aktualizacji informacji gromadzonych w bazie danych w dłuższym okresie oraz udostępniania informacji użytkownikom finalnym.

W książce tej postawiliśmy sobie za cel przedstawienie procesu projektowania bazy danych i metod do tego przydatnych, koncentrując się w szczególności na tych fazach procesu projektowania, które w wypadku systemu opartego na bazie danych powinny być projektowane w sposób odmienny od stosowanego w informatycznych systemach przetwarzania danych ekonomicznych. Założenie to ma wpływ na zakres i układ książki.

Po wprowadzeniu podstawowych pojęć z zakresu baz danych (rozdz. I), w którym definiujemy te pojęcia w zasadzie zgodnie z ujęciem spotykanym w innych opracowaniach podręcznikowych lub monograficznych, przedstawiamy proces projektowania bazy danych jako ciąg następujących po sobie faz, poczynając od ana-

lizy semantycznej zjawisk lub procesów społeczno-gospodarczych, które mają być opisywane przez informacje gromadzone w bazie danych, a na produkcji oprogramowania kończąc. Rozdział II poświęcony jest w całości metodom analizy i syntezy treści bazy danych. Ta faza w tradycyjnych systemach przetwarzania danych zaliczana bywa zwykle do tzw. prac przedprojektowych. W literaturze przedmiotu coraz częściej włącza się ją jako integralną i ważną część procesu projektowania. Jej wynik — model semantyczny bazy danych — jest podstawą podjęcia prac nad „właściwym" projektem systemu. Narzędzia umożliwiające realizację dalszych faz procesu projektowania omawiamy w rozdziałach III, IV i V, które są poświęcone modelom danych na poziomie pojęciowym, logicznym i fizycznym. Środki tworzenia tych modeli w systemie komputerowym, a więc języki dostępu do danych, architekturę oraz funkcjonowanie systemu zarządzania bazą danych przedstawiamy w rozdziałach VI, VII, VIII. Ostatnie rozdziały (IX, X, XI i XII) przedstawiają, w jaki sposób narzędzia projektowe i programowe, omówione w poprzedniej części, można lub należy stosować w procesie projektowania i wdrażania bazy danych.

Projekt bazy danych jest przedsięwzięciem, w którym biorą udział specjaliści z różnych dziedzin: analitycy, technologowie systemów informatycznych, programiści. Bardzo ważny dla końcowego efektu jest też udział finalnego użytkownika systemu opartego na bazie danych i konsultowanie z nim wszystkich skutków decyzji projektowych, które mogą mieć wpływ na zakres i sposób realizacji funkcji użytkowych przez system. Wiadomo z doświadczeń, że komunikacja między tymi, tak różnymi, w przygotowaniu fachowym i w punktach widzenia na bazę danych, uczestnikami procesu projektowania jest słabym i delikatnym punktem. Stąd też pragnienie, aby książka ta umożliwiła przynajmniej w pewnej mierze lepsze porozumiewanie się i wzajemne zrozumienie osób współpracujących nad projektem baz danych, staraliśmy się podać go w możliwie przystępnej formie. To kryterium wzięto także pod uwagę przy selekcji materiału i wyborze stopnia szczegółowości omawianego tematu. Zainteresowanych bardziej szczegółowym omówieniem poszczególnych zagadnień odsyłamy do literatury.

Książka ta jest przeznaczona dla osób, które zawodowo interesują się przetwarzaniem informacji ekonomicznych. Chodzi tu zarówno o analityków systemowych, projektantów i programistów, jak i ekonomistów, statystyków pracujących w zespołach projektowania systemów informatycznych, opartych na bazie danych lub przygotowujących się do tych zadań. Może się także okazać interesująca dla studentów, słuchaczy kursów z dziedziny informatyki lub wykładowców. Założyliśmy, że czytelnik tej książki ma podstawowe wiadomości z zakresu informatyki, w tym projektowania i programowania systemów przetwarzania danych, zna możliwości współczesnych systemów komputerowych, zwłaszcza odnoszące się do przechowywania i udostępniania informacji. Trzeba jednak zwrócić uwagę na to, że łatwość i komunikatywność opisu narzędzi i metod projektowania bazy danych nie oznacza, że można podejmować się takiego przedsięwzięcia, jakim jest uruchomienie bazy danych, bez właściwego przygotowania podstawowego i praktyki w dziedzinie informatycznych systemów przetwarzania danych. Dopiero opierając się na gruntownej wiedzy, jak i długoletniej praktyce można podejmować się zadań tak odpowiedzialnych, złożonych i kosztownych. Projektowanie bazy danych nie jest dobrym zajęciem dla informatycznych debiutantów.

Podobne zastrzeżenia należy postawić użytkownikom, w szczególności tym, którzy decydują o kierunkach zastosowań informatyki w swoich organizacjach. Technologię bazy danych trzeba widzieć jako etap doskonalenia organizacji i technologii istniejącego systemu informatycznego przedsiębiorstwa. Jest on poprzedzony zwykle powszechnym wprowadzeniem transakcyjnych systemów przetwarzania danych dla poszczególnych obszarów systemu informacyjnego organizacji. Baza danych może funkcjonować efektywnie jedynie w otoczeniu innych systemów informatycznych, zasilających ją lub korzystających z danych w niej nagromadzonych. Bez tych systemów, zwłaszcza bez systemów informatycznych zasilających bazę danych, okazuje się ona zwykle przedsięwzięciem zbyt kosztownym, mało przydatnym, a niekiedy nawet chybionym. Przedwczesna realizacja systemów opartych na bazie danych może utrudniać podejmowanie podobnych inicjatyw w przyszłości, gdy warunki po temu będą już

istniały. Wyjątkiem od reguły mogą być tylko pewne typy systemów informacyjno-wyszukiwawczych w technologii baz danych jako systemy autonomiczne, mające własne zasilenie informacjami.

Technologia baz danych wymaga odpowiedniego sprzętu komputerowego. Wymagania sprzętowe związane z wdrożeniem i eksploatacją bazy danych dotyczą nie tylko konfiguracji komputerowej, która jest wykorzystywana do utrzymania samej bazy danych, lecz także odnoszą się do wszystkich systemów informatycznych tworzących otoczenie bazy danych. Ze względu na małą objętość pracy, problemami tymi nie mogliśmy się szerzej zająć.

Opracowując tę książkę przyjęliśmy jednak założenie, że powinniśmy dać w niej Czytelnikowi przede wszystkim odpowiedź na pytanie, *jak* projektować bazę danych w sytuacji, gdy na pytanie *czy* baza danych jest w ogóle potrzebna, padła wcześniej odpowiedź pozytywna. Sądzimy, że bez wystarczającej wiedzy o tym, jak się projektuje bazę danych, trudno jest dać całkowicie poprawnie odpowiedź na to czy warto, czy trzeba i czy można decydować się na budowanie systemu informacyjnego przedsiębiorstwa opartego na wspólnej bazie danych lub zastosować technologię baz danych do przechowywania i przetwarzania informacji w konkretnych sytuacjach.

Warszawa, styczeń 1983 r.

Autorzy

1. Podstawowe pojęcia

1. Baza danych

Pojęcie bazy danych i systemu bazy danych zostało rozpowszechnione nie tylko przez fachową i naukową literaturę informatyczną, lecz także przez materiały reklamowe i popularyzatorów. Możemy więc założyć, że każdy z Czytelników, zwłaszcza parający się profesjonalnie informatyką, ma pewien ogólny pogląd, „wizję" tego, co to jest baza danych. Na wizję tę składają się z reguły następujące stwierdzenia.

— Baza danych jest zbiorem zapamiętanych, przechowywanych i aktualizowanych danych, dostępnych dla wielu użytkowników (systemów użytkowych).

— Aktualizacja bazy danych odbywa się na podstawie danych pobieranych z innych systemów informatycznych, z reguły transakcyjnych, tzw. systemów wsadowych, tworzących *otoczenie bazy danych.*

— Systemy wsadowe są z reguły autonomicznymi systemami informatycznymi, realizującymi własne funkcje użytkowe: funkcja zasilania bazy danych jest jedną z wielu, nie jedyną i najczęściej nie najważniejszą funkcją takiego systemu.

— Projektowanie i rozwój bazy danych ma na celu m.in. przejęcie przez system bazy danych pewnych funkcji realizowanych przez niektóre przynajmniej systemy wsadowe i udoskonalenie trybu realizacji tych funkcji z punktu widzenia użytkownika. Dopiero w dalszej kolejności rozwijane są nowe funkcje.

— Niekiedy istnieje możliwość kształtowania systemów należących do otoczenia bazy danych w taki sposób, że modyfikuje się systemy wsadowe w sposób zgodny z wymaganiami zasilania

bazy danych lub tworzy się systemy wsadowe służące wyłącznie zasilaniu bazy.

Obecnie mimo tego że systemy użytkowe bazy danych przejmują coraz więcej funkcji tradycyjnie realizowanych przez systemy wsadowe, przystępując do projektowania bazy danych nie możemy zakładać, że potrafimy wymusić takie podporządkowanie systemów wsadowych funkcjom bazy danych, że utracą one cechy autonomiczne, iż wejdą w skład *systemu bazy danych*. Oznacza to, że projektant bazy danych działa w warunkach silnego ograniczenia, które wynika z tego, że w dużej części nie ma wpływu na funkcjonowanie systemów transakcyjnych należących do otoczenia bazy danych. Zwykle musi je akceptować takimi, jakie są. Możliwość zwrotnego oddziaływania bazy danych na te systemy pojawia się z reguły dopiero po dłuższym okresie eksploatacji systemu opartego na bazie danych i przejęciu przez systemy użytkowe bazy danych wielu funkcji użytkowych systemów wsadowych.

W związku z tym projektant bazy powinien wykazać, jakie korzyści przyniesie użytkownikom zastąpienie zbioru tradycyjnych systemów transakcyjnych przez jeden system bazy danych. W ocenie tej bierze się pod uwagę dwie podstawowe zalety: z punktu widzenia użytkownika — możliwość centralnego sterowania danymi, a z punktu widzenia eksploatacji i rozwoju systemu informatycznego bazy danych — niezależność danych od programów.

C. Date [1] wymienia następujące korzyści płynące z możliwości centralnego sterowania danymi.

Zmniejszenie redundancji przechowywanych danych. Jak już mówiliśmy, systemy wsadowe, które zasilają informacjami bazę danych, są z reguły systemami autonomicznymi; może więc występować dublowanie danych między nimi. Więcej kłopotu niż zwykłe dublowanie sprawiają niewielkie z pozoru różnice w sposobie interpretacji danych treściowo podobnych, w nazwach danych, regułach identyfikacji, procedurach kontroli, a przede wszystkim to, że zmiany tych elementów odbywają się w poszczególnych systemach wsadowych niezależnie od siebie. Wskutek tego można łatwo utracić kontrolę nad danymi, przejawiającą się

[1] Por. C. J. Date, *Wprowadzenie do baz danych*, Warszawa 1981, rozdz. II.

w zakłóceniach porównywalności danych, utracie ich ciągłości, kompletności, ogólnie mówiąc — *semantycznej integralności danych*. Jedna baza danych, scalająca dane z różnych systemów wsadowych, zmusza do systematycznej kontroli wszystkich elementów rzutujących na integralność semantyczną danych. Wprawdzie nie zawsze jesteśmy władni doprowadzić do integralności różne systemy wsadowe. Co najmniej jednak integracja danych w bazie daje pełną identyfikację wszystkich zakłóceń i zagrożeń ich integralności.

Eliminacja niezgodności danych w bazie. Identyfikacja zakłóceń integralności danych wprowadzanych do bazy stwarza możliwość eliminacji danych niezgodnych z przyjętymi regułami integracji. Dane redundantne lub nie spełniające innych reguł integracji nie będą po prostu wprowadzane do bazy. Trzeba jednak pamiętać o tym, że nie chroni to systemu bazy danych przed eliminacją danych użytecznych, jeżeli kryteria integracji i selekcji są niewłaściwe lub podlegają dezaktualizacji.

Dzielenie zapamiętanych danych między wielu użytkowników. Zaleta ta jest bardzo ważna. Umożliwia utworzenie zintegrowanej biblioteki programów użytkowych, z których mogą korzystać (w ramach swojego upoważnienia) wszyscy użytkownicy systemu bazy danych. Można też tworzyć nowe programy użytkowe korzystające z istniejącej bazy danych.

Wprowadzanie standardów indentyfikacji danych. Centralne sterowanie danymi w ramach określonej organizacji gospodarczej umożliwia wprowadzenie jednolitych standardów informacyjnych, np. kodów, zasad identyfikacji, nomenklatur i klasyfikacji. W tym zakresie wyjątkowo ważną rolę mogą spełnić centralne bazy danych systemów o zasięgu krajowym, jak system statystyczny, system informacyjny planowania centralnego itp. Stwarza to również możliwość wprowadzenia standardów wymiany informacji między różnymi systemami informacyjnymi, wchodzącymi w szeroko rozumiane otoczenie bazy danych.

Wprowadzenie ograniczenia dostępu do danych. Szerokie możliwości ochrony bazy danych przed niepowołanym dostępem zapewniają wysoki poziom bezpieczeństwa danych w bazie. Administrator Bazy Danych (ABD) zarządzający systemem może ustalać różne procedury dostępu do różnych części bazy danych i róż-

nych rodzajów operacji (wyszukiwanie, aktualizacja, usuwanie itp). Jednak bez tych procedur ochronnych system bazy danych narażony byłby na znacznie większe zakłócenia integralności, niż to grozi danym w autonomicznych systemach transakcyjnych, które można chronić skutecznymi, choć bardzo prostymi metodami.

Zachowanie integralności danych. Problem integralności danych w bazie, jej kontroli i utrzymania należy do podstawowych zagadnień projektowania baz danych. Systemy baz danych, w odróżnieniu od systemów transakcyjnych cząstkowych, mają wiele instrumentów kontroli i utrzymania integralności, w tym poprawności, porównywalności, kompletności danych. Możliwości te, jak i niebezpieczeństwa omówimy dalej.

Optymalizacja. System bazy danych stwarza możliwość takiej konstrukcji, by zmaksymalizować jego użyteczność. Na przykład można wybrać taką organizację pamięci, aby realizacja zadań szczególnie pilnych lub bardzo często powtarzanych była szybka, zwykle kosztem innych zadań, o niższym priorytecie lub rzadziej się pojawiających.

Niezaprzeczalne zalety systemu bazy danych osłabiają konsekwencje płynące z faktu, że bazy danych nie tworzy się w próżni informacyjnej, lecz przez integrację funkcji innych systemów informatycznych tworzących otoczenie bazy danych. Systemy te mają swoich użytkowników, gestorów, właścicieli. Wprowadzenie bazy danych powoduje, że tracą oni swoją dotychczasową pozycję monopolisty w dysponowaniu określonym systemem przetwarzania danych na rzecz anonimowego, wspólnego Administratora Bazy Danych. Trudno jest wytłumaczyć im, że ta utrata pozycji „właściciela" danych jest pozorna. Stąd też projektant bazy danych powinien liczyć się z tą obiektywnie uzasadnioną barierą psychologiczną. Jej zlekceważenie może nawet prowadzić do zanegowania użyteczności bazy danych przez przynajmniej część potencjalnych użytkowników. Może powstać sytuacja, w której nastąpi zaspokajanie potrzeb informacyjnych przy wykorzystaniu systemów wsadowych, podczas gdy zasilanie bazy danych, i częściowo samą bazę danych, traktować się będzie jako *produkt uboczny*. Nieumiejętne rozwiązanie tego problemu prowadzi do obniżenia efektywności przedsięwzięcia utwo-

rzenia bazy danych, a nawet — w skrajnych sytuacjach — może uczynić je nieefektywnym bądź zbytecznym.

System bazy danych jako konstrukcja technologiczna ma jedną bardzo ważną zaletę: niezależność danych od programów użytkowych. Oznacza to, że 1) aktualizacja danych nie wymaga zmian w programach użytkowych, 2) zmiana w programach użytkowych, zwłaszcza dołączanie nowych programów, nie pociągają za sobą konieczności zmian w bazie, 3) jednolita identyfikacja danych czyni opłacalne rozwijanie języków użytkownika finalnego, w tym kosztownych języków interakcyjnego dostępu do danych przez różne grupy użytkowników.

Przystępując do projektowania bazy danych należy wziąć pod uwagę oba obszary potencjalnych korzyści. W najbliższych latach ze względu na sytuację w zakresie sprzętu informatycznego w kraju na pierwszym miejscu należałoby stawiać korzyści w sferze użytkowej, wynikające z integracji danych, ich spójności, standaryzacji; nie wydaje się bowiem, aby istniały duże możliwości kształtowania konfiguracji komputerowych pod konkretne zastosowania bazy danych, a ma to istotne znaczenie dla efektywności eksploatacyjnej bazy danych.

2. Od systemu epd do systemu bazy danych

a. Ewolucja systemów przetwarzania danych

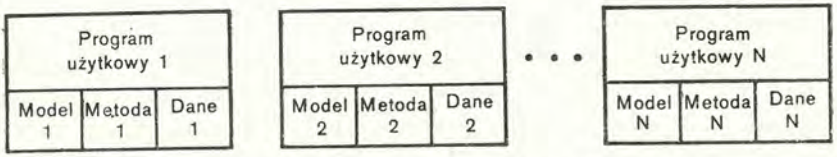
Uporządkujmy pojęcia: dane, metoda, model. W systemie informatycznym dane wejściowe przetwarzane są zgodnie z określoną procedurą. Procedurę tę nazywa się często metodą. Procedura przyjmuje najczęściej postać algorytmiczną. Podkreśla się interdyscyplinarny charakter procedur [2]. Nie chodzi więc o konkretne procedury zorientowane na pojedyncze zastosowanie, lecz o procedury dość uniwersalne, np. podstawowe algorytmy analizy statystycznej, programowania matematycznego. Z kolei przez pojęcie modelu rozumieć będziemy opis obiektu lub zbioru obiek-

[2] Por. B. Meyer, H. Schneider, *Tools for Information System Design and Realisation*, Proceedings of the IFIP TC 8 Working Conference: *Formal Models and Practical Tools of Information Systems Design*, Oxford, 1979. Rysunki do rozdziału I, które ilustrują istotę ewolucji systemów przetwarzania danych, oparto na tym opracowaniu.

tów rzeczywistych lub abstrakcyjnych, w szczególności językowych lub odpowiednich zjawisk, zdarzeń. Opis ten winien być skonstruowany w takim języku, aby umożliwiał identyfikację modelu, tzn. pozwalał na orzeczenie czy model jest optymalny i czy jest dopuszczalny. Nakłada to na język budowy modelu warunek daleko posuniętej precyzji, chociaż jeszcze nie formalizacji. Natomiast w systemie informatycznym modele muszą być skonstruowane już tylko w języku sformalizowanym, np. algebry liniowej, teorii grafów. Podobnie zresztą i dane w systemie informatycznym muszą przyjmować określone formalnie postacie strukturalne. Model określa więc semantyczną i syntaktyczną struktury danych.

Z punktu widzenia tematu pracy specyfikacja tych elementów: danych, metod i modeli, w systemach informatycznych ma istotne znaczenie. Ewolucja architektury systemów przetwarzania danych od systemów informatycznych, obecnie uważanych za tradycyjne, do współczesnych wielodostępnych baz danych polegała na zmianie usytuowania tych trzech elementów w systemie.

W tzw. tradycyjnym systemie przetwarzania danych wszystkie trzy elementy: opis danych, model i metoda były integralną częścią programu użytkowego. Każdy program użytkowy samodzielnie opisywał je wyłącznie dla siebie. Liczba danych była wówczas ograniczona, każdy program użytkowy był realizowany oddzielnie. Sytuację tę przedstawia rysunek 1.



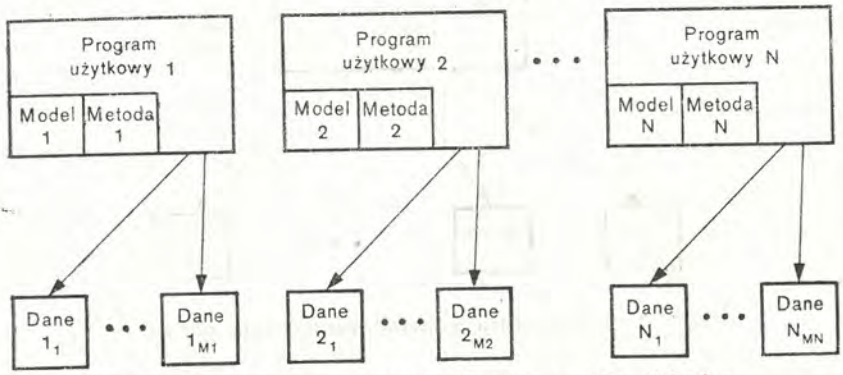
Rys. 1. Tradycyjny system przetwarzania danych

O konsekwencjach takiej organizacji przetwarzania danych dla pracochłonności, kosztów, czasu opracowania i kłopotów aktualizacyjnych nie będziemy pisać. Są one informatykom znane.

Rozwój konfiguracji komputerowych w latach sześćdziesiątych, zwłaszcza rozbudowa pamięci zewnętrznych dały możliwość korzystania przez konkretny program użytkowy z kilku zbiorów danych. Oddzielono więc dane od programu użytkowego, tworząc

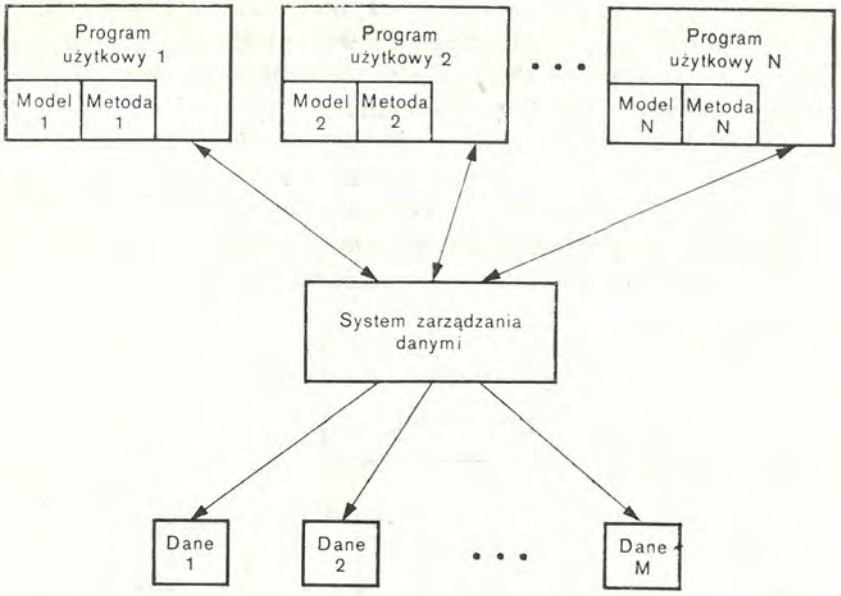
oddzielne zbiory poza programem użytkowym. Niemniej fizyczny opis danych pozostał integralną częścią programu użytkowego. Taka sytuacja obecnie dominuje w systemach partiowego przetwarzania danych, co przedstawiamy na rysunku 2.

Wzrost liczby użytkowników systemów informatycznych oraz techniczne możliwości tworzenia coraz większych zbiorów danych w pamięciach zewnętrznych komputerów stworzyły sytuację, w której kilku użytkowników chciało korzystać z tych samych danych. Niecelowe stało się więc wiązanie danych z programami.



Rys. 2. System przetwarzania danych z wielu zbiorów

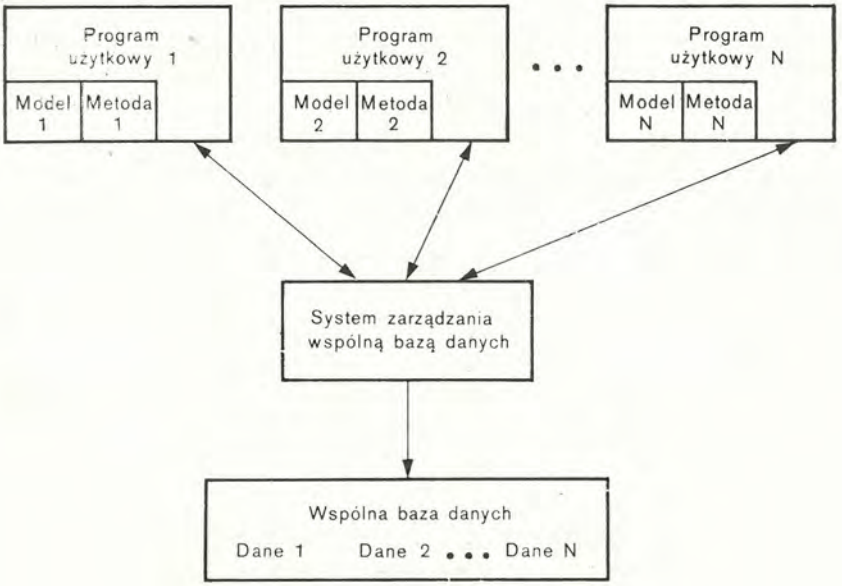
Ekonomika narzuciła potrzebę oddzielenia fizycznych opisów zbiorów od programów użytkowych i poddania ich pod nadzór specjalnego oprogramowania, tzw. systemów zarządzania danymi (*file management systems, data management systems*). Zwalniało to program użytkowy od utrzymywania opisu fizycznego zbioru danych. Wystarczyło podać logiczny opis danych ujęty w modelu. Systemy te umożliwiały też korzystanie przez wiele programów użytkowych, a więc wielu użytkowników, ze wspólnych zbiorów danych. Taki sposób eksploatacji systemów informatycznych, zwłaszcza wobec rozwoju zdalnego dostępu do danych, spowodował konieczność zajęcia się takimi problemami, jak ochrona danych, organizacja przebiegów współbieżnych, likwidowanie konfliktów współbieżnych programów użytkowych. W ówczesnej fazie rozwoju oprogramowania sprawy te w większości załatwiał operator systemu. Schemat organizacji przetwarzania w tym trybie przedstawia rysunek 3.



Rys. 3. Architektura systemu zarządzania danymi

Wadą takiej organizacji systemu informatycznego jest redundancja danych w zbiorach, zwłaszcza w procesach aktualizacji poszczególnych zbiorów. Drugą wadą jest brak instrumentów programowych wspomagających utrzymanie integralności danych, w tym brak *wspólnej bazy metadanych*. W oprogramowaniu systemu zarządzania danymi poprzestawano bowiem na gromadzeniu opisów fizycznych i transformowaniu ich na opisy logiczne (i odwrotnie). Trzeba było wprowadzić instrumenty wspomagające kontrolę semantycznej integracji danych. Temu oczekiwaniu wyszły naprzeciw Systemy Zarządzania Bazą Danych (SZBD). Program użytkowy w warunkach nadzoru nad danymi przez SZBD nie ma możliwości bezpośredniego dostępu do danych. Musi komunikować się z Systemem Zarządzania Bazą Danych, który udostępnia dane lub wykonuje odpowiednie aktualizacje, zgodnie z upoważnieniem programu użytkowego. System Zarządzania Bazą Danych odpowiada za integralność bazy danych we wszystkich aspektach: semantycznym, organizacyjnym i technicznym. W systemach zarządzania bazą danych realizuje się za-

sadę niezależności danych od programów. Oznacza to, że program użytkowy kontaktuje się tylko z logicznym opisem stuktury danych, a odpowiednie operacje tego programu realizowane są przezeń na strukturze logicznej. Tylko System Zarządzania Bazą Danych dysponuje opisem fizycznej struktury danych i dostępem do danych. Struktury: logiczna i fizyczna danych mogą być przeorganizowane bez jakiegokolwiek wpływu na programy użytkowe (por. rys. 4).

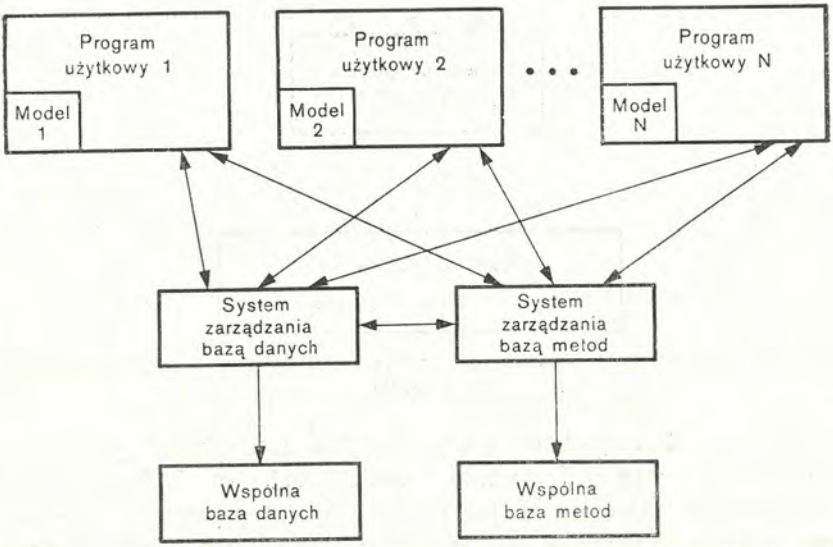


Rys. 4. Architektura systemu informatycznego opartego na wspólnej bazie danych

Systemy Zarządzania Bazą Danych umożliwiły praktyczną i efektywną realizację zasady niezależności danych i programów użytkowych. Umożliwiły też tworzenie zintegrowanych wielkich baz danych, obsługujących wielu różnorodnych użytkowników, szersze wprowadzenie standaryzacji oprogramowania użytkowego, a także wprowadzenie wspólnych systemów metadanych, mianowicie rejestrów i słowników wspólnej bazy danych, zawierających opis danych w języku użytkowników finalnych. Dzięki temu użytkownik ma wgląd nie tylko do zawartości, ale i do

struktury bazy danych, może śledzić jej zmiany, a więc i uczestniczyć w projektowaniu i rozwoju. Zmiana ta miała więc nie tylko znaczenie dla usprawnienia i obniżenia kosztów programowania, skrócenia cyklu programowania, lecz także dla organizacji procesu projektowego.

Kolejnym krokiem ewolucji systemów przetwarzania danych jest usunięcie z programu użytkowego metod przetwarzania informacji. Programy użytkowe mogą korzystać nie tylko ze wspólnej bazy danych, ale i ze wspólnej bazy algorytmów. Organizację przetwarzania opartą na wspólnej bazie danych i algorytmów ich przetwarzania (metod) przedstawiamy na rysunku 5. Poziom ten został osiągnięty w większości nowoczesnych systemów bazy danych. Biblioteki (bazy) algorytmów nowoczesnych systemów komputerowych obejmują z reguły szeroki zakres algorytmów statystyki matematycznej, programowania matematycznego itp.

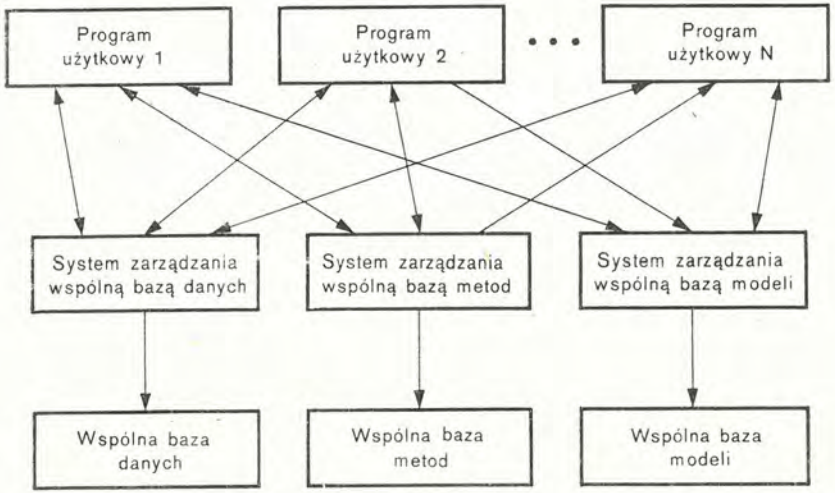


Rys. 5. Architektura systemu informatycznego opartego na wspólnej bazie metod i danych

Można oczekiwać, że następnym krokiem rozwoju będzie wydzielenie z programu użytkowego typowych modeli. Programy użytkowe będą mogły korzystać z bazy zawierającej gotowe

opisy modeli. Wymaga to jednak nie tylko postępu w dziedzinie technik modelowania semantycznego i odwzorowania modeli semantycznych w komputerze (por. rozdz. III), lecz także w upowszechnieniu stosowania modeli matematycznych w praktyce. Architekturę systemu bazy danych, w której wyróżniono bazę danych, bazę metod i bazę modeli przedstawiamy na rysunku 6.

Ten kierunek wydaje się jednak możliwy do realizacji w dalszej przyszłości. Na razie pozostajemy w sferze eksperymentów badawczych. Celowość tworzenia bazy modeli w systemie informatycznym jest ciągle przedmiotem dyskusji i nie ma w tej sprawie jednolitości poglądów. Natomiast wszyscy zgadzają się z innym kierunkiem ewolucji systemu opartego na wspólnej bazie danych, mianowicie zastąpienie programów użytkowych elastycznym językiem użytkownika finalnego, zbliżonym w miarę



Rys. 6. Architektura systemu informatycznego o zintegrowanej bazie danych, metod i modeli

możliwości do jego języka naturalnego, fachowego. Na razie tego typu języki nie stanowią integralnej części systemu bazy danych opartego na uniwersalnych SZBD. Języki te nie akceptują zwykle struktur bazy danych zakładanych przez SZBD, lecz zakładają własne zbiory o określonej strukturze, operują więc własną

metainformacją [3]. Obsługują pewne wyspecjalizowane funkcje użytkowe, jak np. generowanie tablic, modelowanie dynamiki systemów, wyszukiwanie informacji, symulacja procesów. Prowadzi się prace nad wmontowaniem tego typu języków w uniwersalne Systemy Zarządzania Bazą Danych.

[3] Przykładami tego typu języków mogą być języki generowania tablic: TPL *(Table Producing Language)*, TAB 68, a także SEQUEL.

II. Metody analizy i syntezy treści bazy danych

1. Sformułowanie problemu

Z definicji bazy danych wynika, że istotnymi cechami różniącymi metody i technologię baz danych od tradycyjnych systemów przetwarzania danych są integracja i standaryzacja zarówno treści informacji gromadzonych i przetwarzanych w systemie, jak i technologii przetwarzania. W rozdziale tym zajmiemy się problemami związanymi z kształtowaniem treści informacji w bazach danych. Stosowanie odpowiednich metod analizy i syntezy treści informacji w bazach danych jest o tyle istotne, że metody te są również podstawą gospodarowania zasobami danych w toku eksploatacji bazy danych. Tym ostatnim aspektem nie będziemy się szerzej zajmować, ze względu na temat pracy. Warto jednak dodać, że użyteczność metod projektowania bazy danych nie kończy się z chwilą jej uruchomienia, lecz trwa przez cały czas jej istnienia i rozwoju.

Podejmując się określenia zawartości bazy danych musimy wziąć pod uwagę kilka faktów. Po pierwsze, niemal zawsze bazę danych projektuje się w otoczeniu innych funkcjonujących systemów informacyjnych. Baza danych ma za zadanie przejąć lub rozszerzyć możliwości informowania użytkowników realizowane przez istniejące systemy. Stosunkowo niewielki jest obszar obsługi informacyjnej, który przed wprowadzeniem bazy danych nie byłby objęty przez inne tradycyjne systemy informatyczne. Nakłada to na projektanta obowiązek dokładnego poznania owych systemów informatycznych, będących otoczeniem bazy danych. Po drugie, otoczenie bazy danych narzuca projektantowi wiele ograniczeń, w tym determinuje tak istotne sprawy, jak: możli-

wości zasilania i aktualizacja bazy danych, wyjściową formę i zawartość rejestrów i słowników (skorowidzów danych), zasady identyfikacji informacji i kodowania, metody kontroli i korekty danych wejściowych.

Zdarzały się wprawdzie wypadki, że podejmowano próby uruchomienia bez danych zakładając „własne" zasilanie, będące integralną częścią systemu bazy danych. Wszystkie znane nam wypadki takiego podejścia kończyły się albo upadkiem bazy danych, albo „przeproszeniem się" ze wgardzonym otoczeniem bazy danych lub też organizowaniem własnego otoczenia zasilającego kosztem ogromnych nakładów, jakie nie może sobie pozwolić żadna organizacja licząca koszty.

Po trzecie, projektant bazy danych ma niewielki lub żaden wpływ na otoczenie bazy danych. Modyfikacja otoczenia może następować ewolucyjnie, w wyniku przejmowania określonych funkcji systemów należących do otoczenia przez system bazy danych. Następuje to zwykle w długim okresie. Część systemów otoczenia bazy danych pozostaje poza możliwością oddziaływania projektanta bazy danych, np. ogólnokrajowe czy ustalone normami międzynarodowymi zasady kodowania, identyfikacji, nomenklatury, klasyfikacje itp. wchodzące do rejestrów i słowników systemów otoczenia. Te wymagania i ograniczenia wynikające z ustaleń standaryzacyjnych „wyższego rzędu" powinny być projektantowi znane i uwzględniane w projekcie. Narzekanie i udowadnianie nieracjonalności przestrzegania tych ogólnych ustaleń standaryzacyjnych w konkretnej bazie danych na wiele się nie zda, a przy dłuższej eksploatacji bazy danych respektowanie tych ustaleń okazuje się opłacalne, również ekonomicznie.

Po czwarte, trzeba przyjąć do wiadomości, że potrzeb informacyjnych użytkowników bazy danych nigdy nie można do końca określić, i z faktu nieokreśloności potrzeb informacyjnych wyciągnąć odpowiednich wniosków dla elastyczności rozwiązań, zwłaszcza w dalszych, „niższych" fazach procesu projektowego. Względnie precyzyjnemu określeniu i typizacji (np. w formie periodycznie emitowanych zestawień) poddają się tylko pewne standardowe potrzeby informacyjne, wynikające z formalnych obowiązków lub rutynowanych działań użytkowników. Natomiast trzeba pamiętać, że część tych potrzeb, uznanych za rutynowe,

może okazać się nierutynowymi wskutek, a raczej dzięki, wdrożeniu systemu bazy danych. Użytkownicy bowiem w tradycyjnych systemach przetwarzania danych, nie mając możliwości elastycznego dostosowywania żądanych informacji do ich zmiennych potrzeb, godzą się na pewną standaryzację obsługi. Wprowadzenie dzięki bazie danych trybu obsługi „na żądanie", dotyczącego zarówno terminu, formy, jak i treści informacji, skłania ich po pewnym czasie do rezygnacji z rutynowych form obsługi lub co najmniej ich ograniczenia.

Z obserwacji dłuższego użytkowania niektórych baz danych wynika, że dość szybko pojawiają się żądania wzbogacenia treści bazy danych, form udostępniania informacji i procedur przetwarzania. Wskazuje to na zmienność potrzeb, a raczej żądań informacyjnych, wywoływanych właśnie przez wdrożenie bazy danych. Jako metodę badania potrzeb użytkowników zaleca się z reguły ankietowanie. Jednak z naszej praktyki wynika, że należy z dużą rezerwą przyjmować wiarygodność wszelkiego rodzaju ankietowania użytkowników. Chodzi o to, że rutynowe potrzeby informacyjne można łatwo określić bez ankietowania, na podstawie analizy systemów informatycznych, tworzących otoczenie przyszłej bazy danych, a swoich potrzeb niestandardowych użytkownik nie potrafi określić na tyle precyzyjnie, aby były one jakąś podstawą projektowania systemu informatycznego.

Stąd też podkreślamy, że w pracy nad systemem bazy danych należy uwzględnić wnikliwą analizę potencjalnego otoczenia przyszłej bazy danych.

Celem syntezy treści bazy danych jest określenie:

— zakresu podmiotowego bazy danych,

— zakresu przedmiotowego bazy danych,

— zasad podmiotowej integracji danych, pobieranych z otoczenia bazy danych,

— zasad przedmiotowej integracji danych, pobieranych z otoczenia bazy danych.

Przez *zakres podmiotowy* bazy danych rozumiemy zbiór jednoznacznie wydzielonych obiektów lub procesów realnych, które będą opisywane w bazie danych. Wiele baz danych tworzy się w ten sposób, że gromadzi się w nich wartości cech (atrybutów) pewnych klas obiektów. Na przykład w bazie danych o wyro-

bach wytwarzanych przez przedsiębiorstwo czy branżę, obiektem będzie konkretny typ lub grupa wyrobów wyróżniona zgodnie z określoną nomenklaturą, w wypadku bazy danych o pracownikach — obiektem będzie poszczególny pracownik itp. Każdy obiekt opisywany jest za pomocą skończonej liczby cech, których wartości gromadzi się w bazie danych. W zasadzie lista cech jest wspólna dla poszczególnej klasy obiektów. Często też tworzy się bazy danych, w których gromadzi się opisy obiektów należących do kilku klas, np. w kompleksowej bazie danych dla przedsiębiorstwa znajdą się takie obiekty, jak pracownicy, materiały, maszyny i urządzenia, procesy technologiczne.

Przez *zakres przedmiotowy* bazy danych rozumiemy cechę lub zbiór cech obiektów, których wartości gromadzimy w bazie danych. Zakres przedmiotowy danej opisuje jej nazwa.

Integracja podmiotowa danych polega na tym, aby identyfikacja obiektów opisywanych w systemach informatycznych, należących do otoczenia bazy danych, była identyczna z identyfikacją obiektów w bazie danych. Autonomiczny charakter wielu systemów tworzących otoczenie bazy danych oraz odmienne cele systemów otoczenia i samej bazy danych sprawiają, że jest to żmudny i trudny proces.

Z kolei przez *integrację przedmiotową* rozumiemy doprowadzenie do zgodności identyfikacji cech (atrybutów), których wartości gromadzimy w bazie danych, z odpowiednimi cechami (atrybutami), których wartości są gromadzone w systemach należących do otoczenia bazy danych. I ten proces jest często merytorycznie skomplikowany, gdy inne są potrzeby użytkowników systemów otoczenia, a inne użytkowników bazy danych.

Często zarówno w odniesieniu do integracji podmiotowej, jak i integracji przedmiotowej trzeba stosować mniej lub bardziej skomplikowane procedury transformacji formuł identyfikacyjnych obiektów i atrybutów w systemach otoczenia bazy danych na formuły identyfikacyjne tych elementów w samej bazie danych. Ważne jest, aby formuły te zapewniały jednoznaczną transformację również w wypadku zmian, jakie mogą pojawiać się w autonomicznych systemach, tworzących otoczenie bazy danych, na które nie mają wpływu ani projektant bazy danych, ani administrator.

Określając cele i funkcje bazy danych, musimy dysponować zasobem wiedzy o organizacji, dla której projektujemy bazę danych, charakteryzując dwa aspekty tej organizacji:

— potrzeby informacyjne użytkowników, opisane w formie dynamicznego modelu użytkownika,

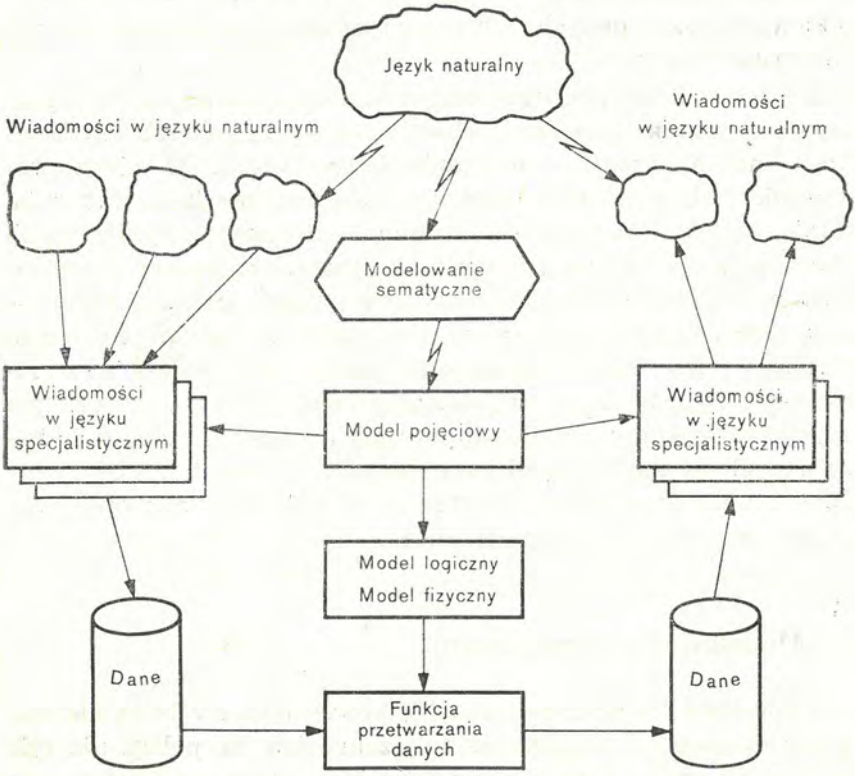
— systemy informatyczne tworzące potencjalne otoczenie projektowanej bazy danych, wyznaczające istniejące możliwości zasilania bazy danych.

Zależnie od specyficznych cech i funkcji systemu informatycznego przeważający może się okazać raz aspekt potrzeb użytkownika, innym razem — otoczenia bazy danych. W pierwszym wypadku odwzorowaniem potrzeb informacyjnych użytkownika finalnego jest jego język; w drugim — decyduje zwykle sposób obserwacji obiektów i ich opisu w systemach, będących potencjalnym otoczeniem bazy danych. Obu tym aspektom odpowiadają inne podejścia do projektowania bazy danych. W pierwszym wypadku właściwe wydaje się zastosowanie tzw. podejścia semiotycznego, w drugim — tzw. podejścia infologicznego. W praktyce częściej spotykamy się z determinującym wpływem otoczenia bazy danych na model treści bazy danych. Na temat ten będziemy dalej mówić (por. rozdz. III pkt 4). Najpierw jednak omówimy pojęcie modelowania semantycznego.

2. Modelowanie semantyczne

Jak już mówiliśmy w rozdziale I, proces budowy bazy danych, w odróżnieniu od procesu przetwarzania danych, polega nie tyle na transformacji danych wejściowych na dane wyjściowe, lecz na stworzeniu możliwie wiernego opisu wybranego wycinka rzeczywistości, by na podstawie tego opisu udostępniać informacje różnym klasom użytkowników. Baza danych jako instrument odzwierciedlenia pewnej zmieniającej się wiedzy o wybranym wycinku rzeczywistości ujmuje w strukturalne formy danych informacje, których treść, strona semantyczna są interpretowane w ramach języka naturalnego lub języka fachowego użytkowników. Dotyczy to zarówno danych wejściowych, źródłowych, jak i danych wynikowych, pojawiających się na wyjściu.

Proces ten przedstawiamy na rysunku 6, na którym modelowanie semantyczne przedstawione jako proces umiejscowiony między językiem naturalnym (lub fachowym), w jakim użytkownik interpretuje dane, a modelem pojęciowym bazy danych, będącym integralną częścią projektu bazy danych.



Rys. 7. Proces transformacji informacji w dane

Modelowanie semantyczne jest procesem definiowania ról i relacji symboli stosowanych do opisu wybranego wycinka rzeczywistości [1].

Do zadań modelowania semantycznego należy:

— wyznaczenie pojęć, za pomocą których opisywać będziemy wybrany fragment rzeczywistości,

[1] Por. R. B. Graves, *Semantic Modelling in Statistics Canada*, Proceedings of ISIS 80 Seminar, VVS, Bratysława 1980.

— określenie możliwie precyzyjnie pól znaczeniowych tych pojęć,

— dobór jednoznaczny nazw (bez synonimów), dla każdego ze zdefiniowanych pojęć w języku możliwie najłatwiej przyswajalnym przez użytkownika,

— zdefiniowanie relacji zachodzących między tymi pojęciami, w tym relacji: podrzędności (nadrzędności), proceduralnych, skojarzeniowych, partytywnych,

— sporządzenie opisu, możliwie pełnego, danego obiektu w języku ograniczonym do wymienionej listy pojęć; nazywane to bywa opracowaniem modelu semantycznego obiektu,

— zidentyfikowanie modelu semantycznego względem obiektu, polegające na stwierdzeniu czy model jest dopuszczalny i optymalny; można tego dokonać przez np. weryfikację opisu przez użytkownika finalnego.

Wynikiem procesu modelowania semantycznego jest opis danej rzeczywistości: obiektu, procesu, zjawiska — w takim języku, aby można było wyrażenia w nim konstruowane interpretować w semantyce języka użytkownika finalnego. Z drugiej strony — ograniczenie języka do pojęć wyznaczonych przez analityka dokonującego modelowania semantycznego staje się filtrem informacji o danej rzeczywistości. Ta funkcja języka modelowania semantycznego jako filtru informacji o rzeczywistości jest bardzo istotna. W związku z tym przy wyznaczaniu pojęć, jakich będziemy używać do skonstruowania modelu semantycznego naszej rzeczywistości, powinniśmy przestrzegać rygorystycznie zasady „brzytwy Ockhama", tzn. nie dopuszczać do wprowadzenia pojęć nie niezbędnych do naszego języka. W przeciwnym razie nieuchronnie w naszej bazie danych pojawi się zbędna redundancja. Zwracamy uwagę na ten aspekt dlatego, że wśród projektantów występuje niejako „naturalna" skłonność do lekceważenia groźby skutków nadmiaru informacji w bazie danych, zarówno w obszarze samych danych, jak i metadanych. Tymczasem nadmiar informacji oznacza, że ponosimy koszty, angażujemy zasoby maszyny, w celu otrzymania zbędnych informacji. W warunkach ograniczonych zasobów ogranicza to możliwość użytecznego rozwoju bazy danych, obniża efektywność eksploatacji, utrudnia użytkownikowi korzystanie z bazy danych. Projektanci mają też

skłonność, uzasadnioną w tradycyjnych systemach przetwarzania danych, do tworzenia projektów zamkniętych. Skłonni są traktować modyfikacje w systemie jako wynik błędów w projektowaniu. W wypadku bazy danych trzeba zmienić nastawienie. Trudno znaleźć taki obszar działalności, dla którego obsługi projektujemy bazę danych, aby w fazie modelowania semantycznego można było sporządzić zamknięty, skończony opis. Trudno też znaleźć — poza wyjątkami, nie wymagającymi korzystania z technologii baz danych — użytkownika, którego potrzeby informacyjne można by dostatecznie wiernie opisać i ująć w modelu semantycznym. Stąd też, zamiast wprowadzać nadmierność danych i ich identyfikacji w modelu semantycznym, lepiej ograniczyć język modelowania semantycznego do pojęć, których niezbędność w systemie informacyjnym możemy potwierdzić. Równocześnie należy zapewnić możliwość elastycznego wprowadzania nowych pojęć w toku dalszych faz procesu projektowania, a zwłaszcza w toku eksploatacji i rozwoju bazy danych. Środki programowe technologii baz danych dają w tym względzie szerokie, ciągle nie dość wykorzystane możliwości.

W pracach nad projektami baz danych proces modelowania semantycznego bywa często nie doceniany. Sprowadza się go nierzadko do prostej analizy istniejącego systemu i analizy potrzeb informacyjnych użytkowników, często metodą ankietowania (!). Bierze się to częściowo stąd, że metody modelowania semantycznego nie są zbyt popularne. Nawet podejścia dość rozpowszechnione nie doczekały się jeszcze opracowań podręcznikowych. Często traktuje się tę fazę prac jako fragment opracowania modelu pojęciowego bazy danych. Zajmiemy się dwoma podejściami, możliwymi do zastosowania w modelu semantycznym bazy danych: podejściem infologicznym i podejściem semiotycznym.

3. Podejście infologiczne

Podejście infologiczne w projektowaniu baz danych wypracowała grupa szwedzkich informatyków (B. Langefors, B. Sundgren, B. Nilsson)[2], zyskując popularność w Kanadzie (R. B. Graves,

[2] Por. B. Langefors, *Infological Models and Information Users Viev, Information Systems*, vol. 5, Pergamon Press London 1980; B. Sundgren, *An Infological*

D. Skuce) [3]; podejście to jest chętnie wykorzystywane przez niektórych projektantów baz danych w wielu krajach [4]. Niemniej jest ono znane przede wszystkim wśród projektantów baz danych statystycznych, planistycznych oraz innych tego typu zastosowań. Mniej lub w ogóle nie jest znane i stosowane w projektowaniu baz danych w systemach zarządzania przedsiębiorstwami. Podejście to może jednak okazać się skuteczne dla jednych, jak i dla drugich. Opracowania na temat podejścia infologicznego nie mają jeszcze charakteru podręczników lub instrukcji metodycznych, chociaż sama metoda w pełni na to zasługuje. Tu chcielibyśmy ją jedynie zasygnalizować i omówić podstawowe założenia metody infologicznej, jak i możliwości jej zastosowania.

Podejście infologiczne opiera się na założeniu, że system bazy danych powinien, z jednej strony, dawać możliwości pełnego i elastycznego opisu wybranego wycinka rzeczywistości, a z drugiej — powinien imitować funkcje umysłu ludzkiego produkowania informacji o tej rzeczywistości na podstawie zawartości pamięci. Informacje o rzeczywistości uzyskujemy obserwując pewne wydzielone i zidentyfikowane obiekty. Tak więc rzeczywistość, którą chcemy opisać w projektowanej bazie danych, jest zbiorem powiązanych ze sobą obiektów. *Obiekt* jest podstawowym pojęciem metody infologicznej.

a. Obiekt

Intuicyjnie obiekt określić można jako „coś", czym jesteśmy zainteresowani, o czym chcemy jak najwięcej wiedzieć i w tym celu gromadzimy o „nim" maksymalnie dużo informacji. Obiektem może być zarówno rzecz fizyczna — realnie istniejąca typu: osoba, budowla, zwierzę, przedmiot, miasto, maszyna, jak też zjawisko psychofizyczne typu: wykształcenie, zainteresowania, zawód, zdarzenie; a więc wszystko to, co jest przedmiotem opisu za

Approach to Date Bases, Urval 7, Stockholm 1974; B. Nilsson, *On Models and Mapping in Date Base Environment*, Urval 9, Stockholm 1980.

[3] Por. R. B. Graves, *Semantic Modelling in Statistics Canada*, Proceedings of ISIS'80 Seminar, VVS, Bratislava 1980; D. Skuce, *An Approach to Defining and Communicating the Conceptual Structure of Data*. University of Ottava, Ottava 1979.

[4] Por. D. Prazenka, *Systemy riadiena bez dat — dvatsat' rokow vyvoja*, „Informacne Systemy" 1981, nr 1.

pomocą informacji. Decyzja o rodzaju i ilości obiektów zawartych w modelu infologicznej bazy danych zapada w momencie jego konkretyzacji, tzn. w toku ustalania:

— jakiego rodzaju decyzje chcemy podejmować za pomocą projektowanej bazy danych,

— jakie są cele ogólne systemu informatycznego, którego fragmentem jest baza danych,

— jakiego rodzaju obiekty chcemy poddać naszej kontroli lub oberwacji za pomocą bazy danych.

Podział analizowanej rzeczywistości na obiekty prowadzi do wydzielenia skończonej listy obiektów prostych, które w końcu nie podlegają dalszemu podziałowi. Fakt czy obiekt jest prosty, czy też jest obiektem złożonym, który można dalej podzielić na inne obiekty proste lub złożone, zależy nie od „materialnej" złożoności obiektu, lecz od relacji między wewnętrzną strukturą obiektu a funkcjami bazy danych. Jeżeli wewnętrzna struktura obiektu nie ma znaczenia dla problemów, dla których rozwiązania projektuje się bazę danych, to obiekt taki powinien być uznany za obiekt prosty, niezależnie od tego czy jest nim prosty wyrób czy cała gospodarka narodowa.

Zbiory obiektów prostych tworzą obiekty złożone. W skład obiektu złożonego mogą wchodzić obiekty proste, jak i obiekty złożone. Te ostatnie jednak również składają się z obiektów, które możemy wyróżniać tak długo, aż dojdziemy wyłącznie do obiektów prostych.

Pojęcie obiektu w podejściu infologicznym nie dotyczy więc materialnej części rzeczywistości. Obiekt jest tworem abstrakcyjnym, wydzielonym według różnych kryteriów, np. organizacyjnych, klasyfikacyjnych, identyfikacyjnych, wypływających z użytkowych funkcji bazy danych, a nie z materialnych cech rzeczywistości.

Bardzo ważne jest dobre zdefiniowanie czasowo-przestrzennych granic obiektu, czyli mówiąc po prostu, gdzie każdy obiekt, prosty lub złożony, ma swój początek i koniec w czasie i przestrzeni. Na przykład trzeba jednoznacznie określić, w jakim punkcie czasowym, czy w jakiej fazie procesu produkcyjnego wyrób powstał jako wydzielony obiekt, w jakim punkcie czasowym przestał istnieć i co go wyróżnia od innych wyrobów. Często jest to

zadanie trudne do wykonania, zwłaszcza wówczas, gdy projektant ma ograniczony wpływ na zdefiniowanie obiektu. Na przykład określenie czasowo-przestrzennych granic takiego obiektu jak zadanie inwestycyjne, przedsiębiorstwo itp. jest bardzo trudne i w ostatecznym rachunku musi być dokonane arbitralnie; często nie satysfakcjonując wszystkich użytkowników bazy danych, z których każdy chciałby dla swoich indywidualnych potrzeb widzieć inny sposób wydzielenia obiektu. Trudności z wydzieleniem i zdefiniowaniem obiektów bazy danych powstają nie tylko wtedy, gdy istnieją sprzeczności interesów użytkowników w zakresie definicji obiektu, lecz również wtedy, gdy w systemach informatycznych tworzących otoczenie bazy danych stosowane są różne definicje tego samego obiektu, inaczej określające jego granice czasowo-przestrzenne. Formułuje się je bowiem zgodnie z potrzebami gestorów systemów informatycznych otoczenia bazy danych [5]. W tym jednak wypadku nie wystarczy uzgodnienie definicji z użytkownikami. Trzeba by jeszcze dokonać odpowiednich modyfikacji wszystkich systemów informatycznych otoczenia bazy danych, w których zastosowano odmienne definicje obiektu. Jest to przedsięwzięcie czasochłonne i zwykle bardzo kosztowne, które może spowodować nieopłacalność całego systemu bazy danych.

Każdy obiekt ma przynajmniej jedną cechę. Przez cechę obiektu rozumiemy stan obiektu, o którym to stanie gromadzimy informację w systemie.

Każdy obiekt istnieje w czasie. Możemy identyfikować obiekt w punkcie czasowym lub w przedziale czasu.

Ta charakterystyka obiektu prowadzi do pojęcia układu elementarnego, bardzo istotnego w podejściu infologicznym. Przez układ elementarny rozumie się trójkę uporządkowaną (x, y, z), gdzie x oznacza obiekt, y oznacza cechę, a z — czas. Zakłada się, że każde zdarzenie opisywane w systemie informatycznym można sprowadzić do tego modelu i, co się z tym wiąże, bazy danych zawierają wyłącznie opisy takich układów elementarnych,

[5] Na przykład różnice w definicji zadania inwestycyjnego stosowane przez inwestora, wykonawcę-przedsiębiorstwo budowlane, bank, aparat planowania i statystykę państwową, które znalazły odzwierciedlenie w systemach informatycznych tych użytkowników, są, jak się wydaje, główną przeszkodą stworzenia zintegrowanej bazy danych dla planowania i kontroli realizacji inwestycji.

z uwzględnieniem ewentualnych powiązań między nimi. Założenie to jest istotną zaletą, ale i ograniczeniem stosowalności podejścia infologicznego.

Trzeba jednak przyznać, że w wielu systemach informatycznych, zwłaszcza w systemach informacji ekonomicznej, model układu elementarnego jest wystarczającym modelem semantycznym. Układ elementarny okazuje się doskonałą formą strukturalizacji wiadomości o obiekcie, niezależnie od stopnia jego złożoności.

Weźmy dla przykładu obiekt *transakcja sprzedaży*, w którym interesuje nas jego cecha *ilość*. Układ elementarny takiego obiektu będzie miał postać trójki uporządkowanej:

$$\langle \textit{transakcja sprzedaży}, \textit{ilość}, \textit{czas} \rangle$$

Analizując obiekt *transakcja sprzedaży*, możemy w nim wyróżnić obiekty: *sprzedający*, *kupujący*, *towar*, przedstawiając otrzymany układ:

$$\langle (\textit{sprzedający}, \textit{kupujący}, \textit{towar}), \textit{ilość}, \textit{czas} \rangle$$

Możemy także podstawiać różne wartości (nazwy) w miejsce cechy, wprowadzając wiele cech charakteryzujących transakcję sprzedaży, jak np. sposób transportu, wartość ubezpieczenia. Ważne jest, że, niezależnie od tych modyfikacji, zawsze, na każdym poziomie, będziemy mieli układ elementarny z wyróżnionymi obiektami i ich cechami identyfikowanymi w czasie.

b. Cechy

Z definicji obiektu wynika, że obiekt jest to „coś", co nas interesuje, o czym chcemy jak najwięcej wiedzieć, o czym zbieramy informacje gromadząc je w bazie danych. Należy więc sobie zadać pytanie, co chcemy wiedzieć o obiekcie, jakie informacje o nim należy gromadzić? Podejście infologiczne daje odpowiedź: poszukiwane informacje o obiekcie sprowadzają się do informacji o cechach obiektu oraz relacjach między danym obiektem a innymi obiektami. Zgodnie z tym, w ramach infologicznej struktury wiadomości, istnieją dwa podstawowe pojęcia: cecha i relacja.

Cechy są nośnikami informacji o obiekcie. Mogą służyć także do generowania cech pochodnych lub nowych. Weźmy dla przykładu dwie cechy obiektu:

P_1 — być wysokim,

P_2 — być szczupłym.

Te dwie cechy, w sytuacji gdy dotyczą tego samego obiektu, np. człowieka, mogą wygenerować trzecią:

P_3 — być wysokim i szczupłym.

Konsekwentnie można by utworzyć dalsze cechy:

P_4 — być wysokim lub szczupłym,

P_5 — być szczupłym i niewysokim,

P_6 — nie być wysokim i nie być szczupłym,

P_7 — być wysokim i nieszczupłym.

Przykłady te przemawiają za tezą o generatywnym charakterze cech obiektów. Wyodrębnienie nowych cech za pomocą cech wyjściowych stawia określone wymagania względem modelu infologicznego: musi on dysponować pewnym zbiorem reguł, według których tworzą się nowe cechy. Trzeba też sformułować kryteria, aby móc wyróżnić cechy pierwotne, tzn. takie, które powstają w wyniku obserwacji i pomiaru „naturalnych" cech obiektu, a których nie można wygenerować z innych cech. Przyjmijmy, że znamy elementy zbioru cech pierwotnych obiektu $P = (p_1, p_2, \ldots, p_n)$ oraz że $G = (g_1, g_2, \ldots, g_m)$ jest zbiorem reguł generowania nowych cech tego samego obiektu. Formuła generowania nowych cech ma postać: $g_i(p_1, p_2, \ldots, p_n) = p_{n+i} \in P$ $(i = 1, 2, \ldots, m)$. Model, w którym występują tylko pierwotne cechy obiektu, nazywa się *bazowym modelem infologicznym.*

Przy określaniu zbioru pierwotnych cech obiektu oraz listy formuł generowania cech pochodnych powstaje problem określenia minimalnego zbioru cech obiektu, które powinny znaleźć odzwierciedlenie w modelu infologicznym. Zbiór wszystkich cech pierwotnych obiektu oraz cech pochodnych, które można wygenerować zgodnie z zasadami g_i ze zbioru cech pierwotnych P, nazywa się *P-generacją cech. P-bazę* stanowi zbiór minimalny elementów należących do zbioru *P-generacji* taki, że żaden z jego elementów nie może być wykluczony ze zbioru cech wchodzących do modelu, bez zmiany możliwości generowania cech pochodnych; *P-generacja* może, ale nie musi być *P-bazą*, natomiast nie

wszystkie elementy należące do zbioru *P-generacji* muszą należeć do zbioru *P-bazy*. Pojęcia te są ważne dla kontroli redundancji w infologicznej bazie danych.

c. Relacje

Cechy są związane z poszczególnymi obiektami i są w tym sensie atrybutami poszczególnego obiektu. Relacje odnoszą się do dwóch lub więcej obiektów jednocześnie i są również częścią modelu infologicznego bazy danych. Określa się je w podobny sposób, jak obiekty lub cechy. Można stwierdzić, że precyzja definicji relacji zależy od precyzji zdefiniowania obiektów i ich cech. Im lepiej i pełniej zdefiniowany jest obiekt, jego cechy, tym lepiej można zdefiniować relacje między różnymi obiektami lub klasami obiektów. Potrzeba określenia relacji wynika ze specyficznych cech rzeczywistości, którą opisujemy w bazie danych, oraz zachowania się obiektów rzeczywistych lub też z potrzeb przetwarzania informacji o obiektach. W pierwszym wypadku mamy do czynienia z relacjami realnymi, mającymi swój desygnat w rzeczywistości, w drugim — z relacjami informacyjnymi, tworzonymi „sztucznie" dla potrzeb organizacji informacji w bazie danych. W praktyce na rozróżnienie tych dwóch klas relacji nie zwraca się większej uwagi, gdyż sposób ich opisu jest taki sam, a kryteria określenia relacji zależą od decyzji projektanta.

Weźmy parę przykładów. Na przykład, jeżeli mamy zbiór osób, którego część zaliczymy do zbioru obiektów nazwanych *pracownikami*, a część — *pracodawcami*, to można stworzyć relację *zatrudnienie* definiowaną jako parę uporządkowaną:

$$\langle pracodawca,\ pracownik \rangle.$$

Jeżeli którykolwiek z elementów pary uporządkowanej zostanie pominięty, relacja przestanie istnieć. Dla relacji prostych, kilkuczłonowych, niebezpieczeństwo niepełnego zdefiniowania relacji nie istnieje. Inaczej ma się rzecz, gdy relacja jest wieloczłonowa, a jej człony zmieniają się dynamicznie w czasie (np. relacja *kooperacja*, obejmująca dostawców, podwykonawców, wyroby, części zamienne dla zakładu produkcyjnego, wytwarzającego skomplikowane wyroby (np. stocznia, wytwórnia komputerów)). Trudności

definiowania skomplikowanej relacji, a zwłaszcza zapewnienia jej aktualizacji, są ograniczeniem dla zastosowań podejścia infologicznego. Prostota obiektów jest tu czynnikiem wyraźnie podnoszącym skuteczność praktycznego stosowania metody infologicznej.

Każda relacja, w zależności od liczby obiektów, które wiąże, ma określony stopień. Najmniejszy będzie drugi stopień, gdyż relacja musi wiązać co najmniej dwa obiekty; relację taką nazywamy *relacją binarną*. Na przykład w obiekcie złożonym *rodzina* występują obiekty proste, członkowie rodziny. Między obiektem *ojciec* a obiektem *syn* zajdzie relacja *bycie ojcem*. Relacja ta nie jest symetryczna, gdyż relacja odwrotna między *synem* a *ojcem* będzie: *bycie synem* lub *bycie dzieckiem*, zależnie od tego czy w systemie istotne jest odróżnienie płci dzieci, czy nie. Interesujące jest, że może zachodzić relacja między obiektem prostym a obiektem złożonym, w skład którego wchodzi tenże obiekt prosty, np. między *głową rodziny* a *rodziną* zachodzi relacja *bycia głową rodziny*, *głównym żywicielem*, relacja między *pracownikiem* a *przedsiębiorstwem* itp.

Przykładem relacji trzeciego stopnia będzie relacja *sprzedaż*, zachodząca między trzema obiektami

$$\langle sprzedający,\ kupujący,\ towar \rangle.$$

Z przykładu tego wynika również względność podziału elementów modelu infologicznego na relacje i obiekty. Poprzednio *sprzedaż* zdefiniowaliśmy jako obiekt złożony z trzech obiektów prostych, teraz definiujemy ją jako relację między trzema obiektami. Nie ma w tym żadnej sprzeczności. Odwrotnie, jest to zaleta metody infologicznej, umożliwiającej sprowadzenie do jednolitej formy strukturalnej różnych semantycznie elementów.

Podobnie jak cechy, tak i relacje mogą na podstawie pewnych zasad generowania tworzyć nowe relacje, będące pochodnymi starych. Jeśli $(r_1, r_2, r_3, ..., r_n)$ są elementami zbioru R-relacji obiektu umieszczonego w modelu infologicznym i g jest elementem zbioru G — zasad generowania nowych relacji dla tego samego modelu oraz utworzono funkcję $g_i\ (r_1, r_2, r_3, ..., r_n) = r_{n+i}$, to wtedy $r_{n+i} \in R$. Natomiast, gdy w modelu infologicznym określone są zbiory: P, R oraz G, to wtedy możliwe jest utworze-

nie funkcji $y = g_i\ (p_1, p_2, ..., p_m, r_1, r_2, ..., r_n)$ przez działanie g_i na zbiorach P i R i wtedy $y \in R$ lub $y \in P$. O ile jednak możliwe jest przedstawienie konkretnego zbioru cech obiektów zawartych w modelu bazy danych, to jest to zawsze trudne lub niemożliwe w wypadku zbioru relacji.

Pochodne relacje dotyczące obiektu, podobnie jak pochodne cechy mogą być formalnie zdefiniowane. Natomiast bazowe relacje obiektu, podobnie jak bazowe cechy mogą być definiowane jedynie nieformalnie. Są one określone słownie za pomocą pojęć zewnętrznych w stosunku do modelu bazy danych i powinny być zrozumiałe dla użytkowników systemu informatycznego. Powstaje więc problem, jak wyrazić formalną definicję relacji pochodnych. Wydaje się, iż jedyną możliwą drogą jest zastosowanie rachunku predykatowego. Na przykład, jeśli określimy p_1: *być kobietą* jako bazową własność oraz r_1: być *rodzicem* jako bazową relację obiektów i jeśli są określone zasady generowania, to nowo utworzoną relację r_2: *być babcią* można potraktować jako relację pochodną od relacji bazowych r_1 i p_1. Tę pochodną relację możemy łatwo zdefiniować za pomocą algebry zbiorów.

d. Czas

Kolejnym pierwotnym pojęciem modelu infologicznego jest *czas*. Każde zjawisko zawarte w bazie danych, jeśli uznamy to za konieczne, będzie obejmowało element czasu. Typowym przykładem może być chociażby układ elementarny.

Sposób przedstawiania elementu czasu, w zależności od poszczególnych zjawisk i sytuacji, może być różny. Dla lepszego poznania problemu rozpatrzmy sytuację opisaną w zdaniu:

„Kowalski był w Koluszkach w ostatni wtorek".

Jak powinno być takie zdarzenie traktowane w modelu infologicznym? Przede wszystkim jest to zależne od celu, jaki postawimy do spełnienia w tej informacji. Jako obiekty można potraktować tu „Kowalskiego" i „Koluszki", a fakt, że Kowalski „był", jako binarną relację tych obiektów. Z kolei należy uwzględnić element czasu, jakim jest zwrot „w ostatni wtorek". Z wielu możliwości odnoszących się do tego jak potraktować element czasu, najprostszy wydaje się ten, który przyjmuje dzień jako punkt

czasu i wtedy problem jest rozwiązany jednoznacznie dla projektanta sytemu. Jeśliby jednak, ze względu na potrzeby modelu, dzień został potraktowany jako przedział czasowy, to w połączeniu z „lokalizacją geograficzną" człowieka, podane zdanie staje się niejasne. Sytuacja ta może być wtedy następująca:
„Kowalski był w Koluszkach w czasie całego ostatniego wtorku", tzn. od godziny 0 do godziny 24 lub
„Kowalski był w Koluszkach w ostatni wtorek po południu" lub o innej porze dnia.
Należy zachować wyjątkową ostrożność i dokładność przy ustalaniu, czy i jak powinien być umieszczany w modelu element czasu.

Można też wyobrazić sobie inną sytuację, którą przedstawimy za pomocą zdania:

„Polska wyeksportowała X ton siarki w 1980 r."

W tym wypadku wydaje się, iż najbardziej naturalne będzie potraktowanie czasu w postaci kolejnych przedziałów, np.:
„Polska wyeksportowała X ton siarki w ciągu całego 1980 r."

Wprowadzenie czasu jako jednego z podstawowych elementów struktury infologicznej pozwoliło na rozszerzenie możliwości opisu obiektów w modelu infologicznym, zwłaszcza na uwzględnienie ich dynamiki. Obiekty charakteryzują się typowymi zmianami, podczas których zyskują, zachowują lub zatracają pewne właściwości lub zmieniają ich wartości.

e. Grupa obiektów

Następnym pojęciem infologicznej bazy danych jest *grupa obiektów*, którą można zdefiniować w sposób następujący:

$Q(p)$ to grupa obiektów utworzona ze względu na cechę p. Przez grupę obiektów więc rozumiemy zbiór wszystkich obiektów, które miały, mają, mieć będą cechę p, tzn.

$$Q(p) = \{q_i | t : \langle q_i, p, t \rangle \in F\}.$$

Zbiór obiektów Q jest grupą obiektów wtedy i tylko wtedy, gdy istnieje taka własność p, że:

$$Q = Q(p).$$

Tworząc grupy obiektów należy uwzględnić składnik czasu t grupy obiektów $Q_t(p)$. Jest to taki podzbiór zbioru $Q(p)$, który charakteryzuje się tym, że obejmuje on wszystkie obiekty posiadające w danym momencie wspólną cechę, tzn.

$$Q_t(p) = \{q_i | \langle q_i, p, t \rangle \in F\}.$$

Jeśli warunek t jest sprzeczny z p, co jest możliwe, to wtedy oczywiście $Q_t(p) = \emptyset$. Sytuacja taka może mieć miejsce, gdy czas występuje w postaci punktów, a model wymaga przedziałów lub gdy w danym punkcie lub przedziale czasu każdy obiekt ma inne — różne cechy. Łatwe do wykazania jest również to, że każda grupa obiektów $Q(p)$ jest zbiorem wszystkich $Q_t(p)$, tzn.:

$$Q(p) = \bigcup_t Q_t(p).$$

Dla własności zmiennych w czasie lub też czasowo zależnych postać grupy obiektów nie ulega zmianie.

Jako przykład grup obiektów można posłużyć się sytuacjami przedstawionymi w zdaniach:

„Kobiety urodzone w XX w., a w 1970 r. pracujące na statkach PLO na stanowisku kapitana".

Jest to grupa jednoobiektowa, gdyż istniał tylko jeden taki obiekt, a mianowicie kpt. Kobylińska. Inne przykłady grup wieloelementowych:

„Ludzie urodzeni w 1960 r., chodzący do szkoły w 1970 r. i mający na nazwisko Kowalski"

„Dzieci urodzone w 1978 r., które pójdą do szkół specjalnych w 1985 r."

Na podstawie przedstawionych na początku podrozdziału definicji i określeń można stwierdzić, że każda cecha z modelu infologicznego może stanowić grupę cech odnoszącą się do pewnej grupy obiektów. Dwie niezależne grupy obiektów mogą być związane ze sobą relacjami w trojaki sposób:

— mogą nie mieć wspólnego obiektu (związek przez relację opisaną na cechach),

— mogą się częściowo pokrywać (relacja między obiektami bez odwoływania się do cech),

— jedna może być podgrupą drugiej (zwykle relacja hierarchicznego podporządkowania).

f. Atrybuty obiektu

Atrybuty obiektu są to wszystkie te dane, które są związane z obiektem, a mianowicie:

— atrybut kluczowy — dana będąca kluczem dostępu do obiektu,

— atrybut proceduralny — dana biorąca udział w przetwarzaniu,

— atrybut opisowy — dana opisująca obiekt.

Chcąc jednak dokładnie wyjaśnić te pojęcia, w stosunku do innych już poznanych, należy je przedstawić w kategoriach określonych w modelu infologicznym.

Jeśli przyjmiemy, że $Q(p)$ jest pewną grupą obiektów utworzoną przez własność p, $A = \{v_i\}$ jest zbiorem cech, a dla wszystkich przedziałów czasu t każdy obiekt, który jest zawarty w $Q_t(p)$, jest także *zawarty w jednym* z przedziałów czasu $t(v_i)$ utworzonych przez własność v_i ze zbioru A, to ten właśnie zbiór A nazywać będziemy atrybutem związanym z grupą obiektów $Q(p)$, a elementy $v_i \in A$ — wartościami atrybutu A. Zbiór A jest atrybutem wtedy i tylko wtedy, gdy istnieje pewna grupa obiektów z nim związana; grupa obiektów jest nazywana wtedy związkiem atrybutu A. Jeżeli przy określaniu atrybutu zastąpimy zwrot „w jednym ze" za pomocą „w tylko jednym", to otrzymamy atrybut jednowartościowy, czyli zmienną. Atrybuty, które nie są zmiennymi, będą nazywane wielowartościowymi.

Zwracamy uwagę na różnicę między pojęciem cechy obiektu i atrybutu. Mówiąc o cesze, mamy na myśli jakościowe charakterystyki obiektu, natomiast gdy mówimy o atrybucie, chodzi nam o mierzalne w sposób skwantyfikowany lub nie — wartości cech [6]. Zatem jednej cechy może dotyczyć kilka atrybutów. Sytuacja odwrotna raczej nie wchodzi w rachubę.

---

[6] Por. W. Belke, D. Graichen, M. Starrus, *Nichtmetrische Klassifizierung von Informationen*, Akademie Verlag, Berlin 1979, rozdz. 2. Zbliżone podejście znajdujemy w tej pracy. Autorzy wprowadzają pojęcia: *cechy nominalnej (nominales Merkmal), cechy porządkowej (ordinales Merkmal)* i *cechy kardynalnej (kardinales Merkmal)*. Dwie ostatnie pokrywają się z pojęciem atrybutu w metodzie infologicznej, pierwsza — z pojęciem cechy obiektu.

g. Informacje i metainformacje

W metodzie infologicznej przyjęto intuicyjnie rozumienie pojęcia informacji. Natomiast zwraca się uwagę na to, że informacje są zawsze opisami wyróżnionych obiektów, zgodnie z podanym już podejściem, oraz że jeden i ten sam obiekt może być opisany w różny sposób. Dobór zaś sposobu opisu ma duży wpływ na efektywność systemu informacyjnego. Weźmy dla przykładu dwie informacje:

I. Jan został powołany na dyrektora przedsiębiorstwa X w dniu 2 maja 1980 r.

II. Zięć Piotra został powołany na dyrektora na miejsce szwagra Anny w 30 dni po tym, jak jej brat został zastąpiony przez Piotra na stanowisku podsekretarza stanu.

Te dwa zupełnie różne opisy dotyczą jednego i tego samego zdarzenia, lecz w sposób zupełnie różny. Zięć Piotra = Jan, dzień odwołania brata Anny ze stanowiska +30 dni = 2 maja 1980 r., przedsiębiorstwo X = przedsiębiorstwo, w którym dyrektorem do dnia 2 maja był szwagier Anny itd. Łatwo sobie wyobrazić wiele innych odmian tych zdań. Fakt istnienia różnych sposobów prezentacji informacji o obiektach wskazuje, że istnienie różnic w formie przekazywanych informacji nie musi oznaczać faktycznych różnic, jeśli chodzi o ich treść w odniesieniu do obiektu i jego cech. Dwa wymienione zdania, mimo że różne „infologicznie", przekazują te same treści, jeżeli chodzi o samo konkretne zdarzenie.

Rozróżnienie między obiektem a informacją o obiekcie, np. jego nazwą, powinno być zaznaczone w dokumentacji modelu infologicznego. Proponuje się dla obiektów cech, relacji, używać liter małych, zaś dla obiektów informacyjnych — wersalików. Na przykład Jan = zięć Piotra prezentuje relację obiektów, podczas gdy JAN = ZIĘĆ PIOTRA oznacza pewną informację.

W metodzie infologicznej starannie rozróżnia się:

— zjawiska świata rzeczywistego (obiekty, ich cechy, relacje zachodzące między obiektami),

— informacje o obiektach,

— dane reprezentujące te informacje w systemie informatycznym,

— informacje o informacjach, czyli metainformacje,
— dane o danych zawartych w bazie, czyli metadane.

Przez metainformacje rozumie się w metodzie infologicznej:
— informacje o elementach modelu infologicznego, a więc np. formalne i nieformalne definicje typów obiektów, atrybutów,
— informacje o danych zawartych w bazie, np. słownik deskryptorów.

Metadane są znakowymi reprezentacjami metainformacji.

h. Wiadomości

Ogólna definicja wiadomości przyjęta w metodzie infologicznej pokrywa się z definicją powszechnie przyjętą w informatyce. Tutaj zakłada się jednak ograniczenie formy strukturalnej wiadomości do układu elementarnego, wyznaczonego przez trójkę uporządkowaną (x, y, z), czyli (*obiekt, cecha, czas*) x jest nazwą lub zbiorem nazw obiektów, y — jest zbiorem nazw atrybutów, a więc cech lub relacji, z stanowi punkt lub przedział czasu. Inaczej mówiąc x jest składową obiektową, y — składową predykatową, a z — składową czasową. Wiadomość, w której usunięcie jakiejkolwiek składowej powoduje, że przestaje ona być wiadomością, nazywa się wiadomością elementarną (e — wiadomość).

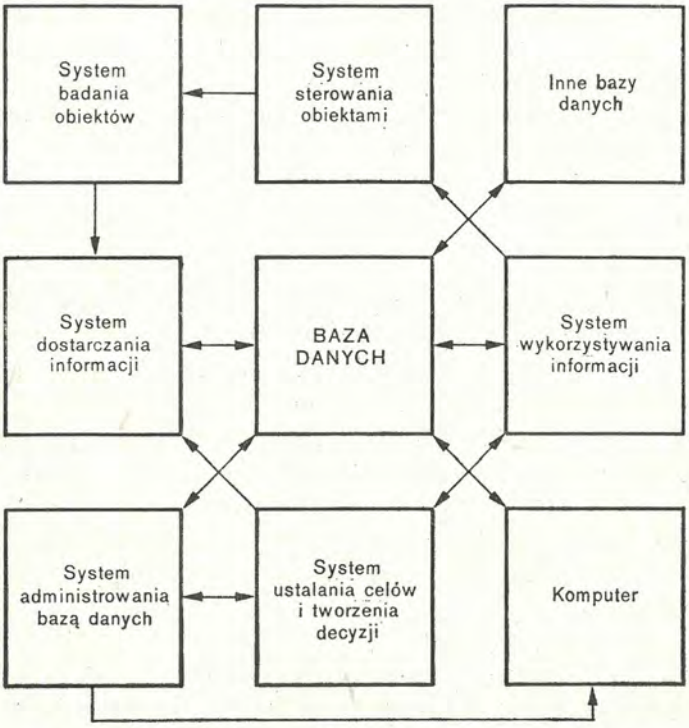
i. Infologiczna baza danych

Infologiczny model bazy danych jest głównym produktem stosowania tej metody w procesie określania treści bazy danych i strukturalizacji tej treści. Składa się on z dwóch części: modelu otoczenia bazy danych oraz modelu właściwego (głównego) bazy danych. Opracowanie tych modeli musi odbywać się równolegle, ponieważ analiza informacji i funkcji otoczenia bazy danych wpływa na definiowanie elementów modelu samej bazy, i odwrotnie, prace nad infologicznym modelem bazy danych będą miały wpływ na sposób analizy systemów należących do otoczenia bazy danych.

Budując model infologiczny bazy danych, zgodnie z wprowadzonymi już pojęciami, musimy sporządzić opis wybranego wycinka rzeczywistości definiując obiekty, ich cechy, relacje, dokonać transformacji tego opisu na dane, przypisując odpowiednie

nazwy tak, aby można było je traktować jako znaki formalne. Można sporządzić listę klas wiadomości elementarnych, procedury generowania cech pochodnych i relacji pochodnych itd.

Prowadząc analizę otoczenia bazy danych, w celu skonstruowania jej modelu infologicznego, musimy dysponować przynajmniej wstępnymi definicjami obiektu bazy danych, listy cech obiektów i typów relacji, jakie trzeba będzie odwzorować w bazie. Natomiast sam model infologiczny bazy danych traktujemy jako czarną skrzynkę, której struktura wewnętrzna nas nie interesuje. Jest to bardzo ważne. Umożliwia bowiem zachowanie w projekcie niezależności wewnętrznej struktury przyszłej bazy danych od zmian zachodzących w otoczeniu bazy danych, przez stworzenie odpowiedniego systemu definiowania i identyfikowania odpowiednich obiektów, cech i relacji w bazie danych i w otoczeniu bazy danych.



Rys. 8. Struktura otoczenia infologicznej bazy danych

Zakres analizy otoczenia zależy od konkretnych potrzeb użytkownika i specyficznych cech oraz funkcji systemu informatycznego, organizacji, dla której baza jest projektowana. Hipotetyczny zakres możliwej analizy otoczenia bazy danych przedstawia rysunek 8.

Wiadomości wprowadzane są do bazy danych z jej otoczenia. Powstaje pytanie, czy powinien być to ten sam zbiór informacji, jaki pojawia się na wejściu do „czarnej skrzynki", jedynie pomniejszony o informacje zbędne, czy też poddane one muszą być dalszym przekształceniom.

W analizie tego problemu należy uwzględnić fakt, iż baza danych powinna być zasilana wiadomościami elementarnymi. Weźmy prosty przykład:

a — pies,
b — Zosia,
c — Franek,
d — brat *b*,
e — właściciel *a*,

⟨*a* pogryzł wczoraj *d*⟩

czteroelementowa wiadomość jest wprowadzona do bazy danych. Jednak można twierdzić, że zawiera ona nie tylko te informacje. Na przykład można wykazać, że zawiera też elementarną wiadomość o nowej treści: „Pies Franka pogryzł wczoraj brata Zosi", „Franek jest właścicielem psa", ,Zosia ma brata" itd., mimo iż wiadomość w tej formie nie była wprowadzona. Zatem można stwierdzić, że „czarna skrzynka" ma nie tylko zdolności reprodukcyjne, tzn. postrzega, przechowuje i reprodukuje informacje na każde życzenie użytkownika, lecz również ma właściwości dedukcyjne — tworzenie informacji o nowej wartości. Dedukcja jest realizowana według reguły dedukcyjnej, którą nazwano *schematem*. Wiadomości, które za pomocą *schematu* są uzyskiwane (dedukowane) z innych, nie wymagają gromadzenia. Są one możliwe do odtworzenia w każdym momencie czasu. Istnieją jednak dwa czynniki ograniczające *schemat*, a mianowicie: szybkość odtwarzania oraz kontrola jakości uzyskiwanych wartości. Są one konieczne dla zapobiegania redundancji informacji.

Obok *schematu* infologiczna baza danych ma jeszcze dwa podsystemy, a mianowicie: *rdzeń* i *filtr*. W świetle tych pojęć można formalnie zdefiniować infologiczną bazę danych:

$$IBD = \langle S, R, F \rangle,$$

gdzie:
S — schemat,
R — rdzeń,
F — filtr.

Schemat. Z infologicznego punktu widzenia *schemat* bazy danych jest identyczny, jeśli chodzi o opis z głównym modelem, tzn. zawiera wykaz zbiorów, jak również informację o typach obiektów, atrybutach, relacjach obiektów, regułach generowania, typach układów elementarnych oraz definicjach zewnętrznych i wewnętrznych.

Schemat ma dwie podfunkcje: semantyczną i dedukcyjną. Pierwsza z nich polega na komunikowaniu użytkownikom faktycznej treści wiadomości uzyskiwanych z bazy danych. Jest to zadanie niełatwe, zważywszy, że istnieje wiele osób zajmujących się tworzeniem, ochroną i użytkowaniem potencjału informacyjnego bazy danych i na pewno każda z nich ma własne wymagania, różne od innych, jeśli chodzi o strukturę wewnętrzną posiadanych wiadomości. Dodatkową trudność sprawia fakt, że struktury te ulegają stałym zmianom.

Dedukcyjna podfunkcja schematu jest związana ze zbiorem zasad generowania informacji, który determinuje, jakie wiadomości zostały wyprowadzone z innych i nie pochodzą z zewnątrz. Uważa się, że rozwinięte bazy danych są w stanie wygenerować nawet pewne prawa empiryczne. Wydaje się, że raczej jest to sprawa przyszłości, zależna od sukcesu zastosowań metod tzw. sztucznej inteligencji. Prawa te otrzymywane mogłyby być w postaci wniosków wynikających z treści tych informacji. Takie systemy mogą mieć zastosowanie jedynie w takich dziedzinach czy działach gospodarki, gdzie dane są bardzo liczne. Prawa empiryczne lub nawet słaba korelacja między atrybutami obiektu może być bardzo pomocna dla zupełnie innych celów.

W bardziej rozwiniętych bazach danych mogą pojawić się struktury danych, w których atrybuty są mocno skorelowane. Powinno wówczas następować ich grupowanie, scalanie i reduk-

cja. Stara struktura powinna być zastąpiona nową, uwzględniającą opis rzeczywistości z jedną lub kilkoma kartotekami, zawierającymi wszystkie wystąpienia atrybutów rzeczywistych oraz niektóre wystąpienia atrybutów nie mających odpowiedników w rzeczywistości, ale możliwych do wydedukowania przez bazę.

Rdzeń. Rdzeniem nazywa się podstawowy zbiór informacji, które w kombinacji ze *schematem* wystarczają do utworzenia wiadomości o określonej treści. Rdzeń zawiera np. wartości numeryczne wskaźników. Poprawnie skonstruowany rdzeń bazy danych jest nieredundantny. Z infologicznego punktu widzenia nie zawsze można o konkretnej wiadomości jednoznacznie orzec czy jest częścią *rdzenia* czy też nie. Może istnieć wiele różnych zbiorów informacji spełniających takie warunki. Wybór jednego nieredundantnego lub redundantnego rdzenia kandydującego do miana „głównego" w bazie danych jest warunkowany potrzebami użytkowników systemu. Tak więc dokładne rozgraniczenie bazy danych i jej rdzenia będzie do pewnego stopnia decyzją arbitralną. Natomiast bardziej precyzyjnie można rozstrzygnąć, które wiadomości powinny się znaleźć w „rdzeniu", a które powinny być jedynie odzyskiwane za pomocą zasad generowania.

Rdzeń bazy danych może obejmować każdy rodzaj wiadomości: kompletną i niekompletną, elementarną i złożoną, pierwotną i wtórną itd. W praktyce *rdzeń* jest równoznaczny ze zbiorem kompletnych, elementarnych i zrozumiałych wiadomości, najczęściej w postaci kartoteki.

Filtr. Funkcją *filtra* infologicznej bazy danych winno być przede wszystkim ochronienie jej i jej użytkowników przed wiadomościami fałszywymi oraz takimi, których treść nie jest zgodna z opisami i definicjami, zawartymi w *schemacie*. Działalność filtra może przejawiać się m.in. w różny sposób:

— może odmówić wprowadzenia do bazy danych wiadomości, których struktura jest sprzeczna z modelem infologicznym,

— kontroluje informacje wychodzące,

— wyłapuje i gromadzi „neologizmy" informacyjne.

Na przykład funkcja *filtra* może być użyta do ochrony bazy danych i jej użytkowników przed fałszywymi wiadomościami, natomiast Administrator Bazy Danych może za jej pomocą tworzyć nowe struktury logiczne bazy.

Strefą jego działania nie są jedynie obszary wejścia i wyjścia. Ilekroć system jest „nieczynny", tzn. kiedy nie ma procesu transakcji między bazą a otoczeniem, to może dzięki swoim właściwościom próbować „oczyszczać" sam siebie z pewnych treści, które zatrzymał w toku pracy. Dodatkowo filtr spełnia rolę ochrony informacji, uniemożliwiając ujawnienie informacji nie upoważnionym użytkownikom. Dzięki tym zdolnościom jest w stanie ukazać tę samą bazę danych różnym użytkownikom w całkowicie odmienny sposób.

j. Dynamiczny i transakcyjny charakter bazy danych

Infologiczna Baza Danych została formalnie zdefiniowana jako:

$$IBD = \langle schemat, rdzeń, filtr \rangle.$$

Tak więc przekształcenie się jednego ze składników powodować może zmianę całej bazy. Najczęściej spotykaną, typową zmianą jest modyfikacja *rdzenia*, która może następować przez wprowadzenie lub kasowanie pewnych wiadomości. Nie chodzi tu oczywiście o kasowanie wiadomości redundantnych. Zmiana *schematu* może, choć nie musi, zmieniać treści bazy danych (w pewnym momencie czasu). Na przykład zastąpienie jednej definicji drugą może najprawdopodobniej wpływać na zbiór wiadomości, podczas gdy wprowadzenie nowego atrybutu nie będzie miało żadnego efektu na tym polu. Filtr i jego zmiany mają wpływ znikomy, jakkolwiek wszelkie niedokładności w jego działaniu, zwłaszcza w zakresie ochrony tajności danych, mogą bezpośrednio i gwałtownie oddziaływać na efektywność użytkową bazy danych.

Reasumując, dynamiczny charakter bazy wyraża się ciągłymi zmianami i modyfikacjami poszczególnych jej członów. Proces ten ma na celu stałą modyfikację, unowocześnienie i dopasowywanie danych do zmieniających się warunków zewnętrznych.

Transakcję można określić jako wszelką wymianę danych między bazą a jej otoczeniem. Formalnie transakcję, w warunkach Infologicznej Bazy Danych (IBDt), można przedstawić w postaci dwuelementowego zbioru:

$$IBDt = \langle operator, parametr \rangle,$$

gdzie:

— operator — jest to bodziec, który inicjuje pewien rodzaj postępowania otoczenia lub bazy danych, polegający na wprowadzeniu lub wyprowadzeniu pewnej wartości transakcji,

— parametr — zmienia postępowanie bazy danych lub otoczenia za pomocą procesów inicjowanych przez operator wyposażony dodatkowo w pewne wartości wejścia lub wyjścia.

Klasyfikacja transakcji opiera się na strukturze podsystemów otaczających Infologiczną Bazę Danych. Część z nich jest w większym stopniu zaangażowana w wymianę danych i informacji niż pozostałe. Są to (por. rys. 8):

— podsystem dostarczania informacji,

— podsystem wykorzystania informacji,

— podsystem administrujący bazą danych obejmuje: planowanie, systematyzację, programowanie oraz kontrolę decyzji i funkcjonowania,

— podsystem komputera obejmuje „sztuczną inteligencję", komputer obsługujący bazę danych oraz ludzi i procesy zaangażowane w obsługę i zarządzanie komputerem,

— podsystem innych baz danych.

Można stwierdzić, że każdy z wymienionych podsystemów w pewien sposób charakteryzuje transakcję. Jednak największy udział mają trzy pierwsze, gdyż na ich podstawie ustalono następujące transakcje:

— dostarczanie informacji,

— wykorzystanie informacji,

— administrowanie bazą danych.

Każdą z tych funkcji transakcyjnych można uzupełnić podziałem na podfunkcje. Są one zgodne i ściśle określone w warunkach funkcji wyjściowych. Wiele z tych podfunkcji jak: kodowanie, programowanie może dziedziczyć swoje nazwy z funkcji już istniejących lub im podobnych. Pozostałe mogą być dowolnie nazywane w zależności od potrzeb użytkowników. Należy jednak uważać, aby uniknąć chaosu pojęciowego. Dwiema podfunkcjami administracji bazy danych są: struktura informacji i struktura danych. Zadania struktry informacji związane są

z modelem infologicznym, który jest podstawą bazy danych, co oznacza jednocześnie powiązanie ze *schematem, filtrem* i *czarną skrzynką*. Struktura informacji powinna odpowiadać specyfice zastosowania, zawartości i logicznej zgodności modelu z opisywaną rzeczywistością oraz w miarę potrzeb powinna umożliwiać zmiany w zakresie specyfikacji nowej grupy obiektów, własności, atrybutów, relacji obiektów, definicji i nazw. Zadaniem jej również jest informowanie o zawartości bazy danych, możliwościach korzystania z niej oraz aktywizacja potencjalnych użytkowników. Struktura danych natomiast powinna dotyczyć zadań związanych z wewnętrzną — „datalogiczną" strukturą bazy danych. Jeśli infologiczne lub technologiczne otoczenie bazy danych zmienia się, to struktura danych powinna również podlegać odpowiednim zmianom.

Metoda infologiczna doczekała się praktycznych implementacji i języków, umożliwiających opis infologicznej bazy danych zgodnie z wymaganiami formalizacji odpowiadającej modelowi pojęciowemu. Przykładem takiej implementacji jest język LESK [7]. Jak to wynika z tej implementacji, metoda infologiczna okazuje się skuteczna wówczas, gdy możemy definiować obiekt bazy danych jako odwzorowanie obiektów występujących w realnej rzeczywistości oraz gdy obiekty te nie mają zbyt wielu cech, ani nie są powiązane zbyt wieloma relacjami.

4. Podejście semiotyczne

W wielu zastosowaniach trudno jest zdefiniować obiekt rzeczywisty, o którym informacje chcemy gromadzić w bazie danych. Trudno też wprowadzić obiekt zastępczy, abstrakcyjny, mogący zastąpić brak obiektu rzeczywistego. Może wreszcie powstać sytuacja, w której istnieje bardzo wiele klas obiektów, na dodatek podlegających różnym zmianom. Brak możliwości sprowadzenia wszystkich różnorodnych obiektów do paru klas, dla których tworzylibyśmy takie „obiektowe" bazy danych, minimalizuje pożytek, jaki możemy odnieść z metody infologicznej.

W takich wypadkach skuteczna może okazać się metoda semiotyczna. Nazwa tej metody pochodzi stąd, że za punkt wyjścia

[7] Por. D. Skuce, op. cit.

przyjmuje się model strukturalny nazwy wskaźnika ekonomicznego jako wyrażenia skonstruowanego w określonym języku, w tym wypadku w języku ekonomicznym. Ów językowy charakter modelu znajduje wyraz w nazwie metody, jak i modelu, zwanym semiotycznym modelem wskaźnika. Stanowi on metodyczną podstawę strukturalizacji wszelkich wiadomości skwantyfikowanych w systemach ekonomicznych.

W metodzie semiotycznej obiektem, którego opisy gromadzimy w bazie danych, jest wskaźnik zespolony [8]. W praktyce wygodnie jest stosować technikę modelowania wskaźnika zespolonego, wykorzystując metodę graficzną, stosowaną do modelowania szablonów semantycznych w tzw. sztucznej inteligencji, ewentualnie uzupełnioną o odpowiedni zapis teoriomnogościowy. Alternatywną metodą modelowania jest traktowanie modelu wskaźnika ekonomicznego jako formuły fasetowej języka ekonomicznego, traktowanego jako język informacyjny. Oba podejścia mają swoje uzasadnienia. Oba też mają ograniczenia. Omówimy pokrótce tryb postępowania obu tych podejść.

Najpierw jednak kilka uwag ogólnych. W metodzie semiotycznej w strukturze bazy danych odwzorowujemy, w odróżnieniu od metody infologicznej, nie relacje między obiektami, lecz związki między wskaźnikami a ich atrybutami. Związki te mogą więc zachodzić między różnymi wskaźnikami zespolonymi lub między różnymi wskaźnikami elementarnymi. Mogą to być związki między atrybutami w ramach tego samego wskaźnika zespolonego lub między atrybutami różnych wskaźników zespolonych. Podstawowe relacje, zachodzące między wskaźnikami a atrybutami, są relacjami typu językowego, relacje między wyrażeniami *jednego* języka informacyjnego. Są to relacje:

— podrzędności-nadrzędności,
— części-całości,
— kojarzeniowe,
— proceduralne.

Relacje podrzędności-nadrzędności występują wówczas, gdy wyznaczone są one przez zasady hierarchii przyjęte w klasyfikacji

[8] Na temat modelu semiotycznego wskaźnika ekonomicznego oraz definicji wskaźnika zespolonego por. J. Oleński, *Języki problemowo-zorientowane — podstawy projektowania*. W: Informacyjne problemy planowania, praca zbiorowa pod red. W. Maciejewskiego, Warszawa 1982, s. 140—163.

lub w innym języku deskryptorowym, ujmującym hierarchię deskryptorów. Nie chodzi tu jednak o hierarchię w ramach samej klasyfikacji lub języka deskryptorowego, lecz o relacje wynikające z zaklasyfikowania za pomocą wyrażeń z różnych poziomów hierarchii, innych wyrażeń lub obiektów. Chodzi o relacje między tymi klasyfikowanymi obiektami. W wypadku wskaźników, o relacjach nadrzędności-podrzędności możemy mówić tylko w odniesieniu do wskaźników należących do jednego wskaźnika zespolonego. Na przykład wskaźnik X charakteryzujący dział gospodarki narodowej będzie w tym sensie nadrzędny w stosunku do analogicznego wskaźnika odnoszącego się do branży gospodarki w tymże dziale.

Często spotykaną i odwzorowywaną w strukturze baz danych jest relacja część-całość, tzw. relacja partytywna. Wskaźnik x jest częścią wskaźnika y, jeżeli pole semantyczne wskaźnika y zawiera pole semantyczne wskaźnika x.

Relacje kojarzeniowe, to takie, w których dwa wskaźniki mają identyczne pewne atrybuty. Jest to relacja kojarzeniowa za pośrednictwem atrybutu. Szczególnie interesujące są relacje kojarzeniowe wyznaczane przez atrybuty pełniące funkcję deskryptorów tematyczno-dziedzinowych, zwłaszcza dziedzin wspólnych, takich jak efektywność, wydajność pracy, inwestycje, koszty. Wiążą one wskaźniki z odległych nawet dziedzin, ale charakteryzujących ten sam aspekt analityczny (np. wydajność pracy w przemyśle rolno-spożywczym z wydajnością pracy w transporcie, zatrudnienie w transporcie wewnątrzzakładowym w różnych zakładach czy przedsiębiorstwach). Relacje kojarzeniowe są szczególnie przydatne dla wybierania i prezentowania informacji do analiz nie opisanych formalnymi procedurami.

Jeżeli wymagania analizy lub zestawu informacji możemy opisać za pomocą sformalizowanej procedury, to możemy opisać w bazie danych relacje kojarzenia przez procedurę (relację proceduralną). Wiąże ona wskaźniki występujące jako podstawa obliczeń wskaźników pochodnych według określonych procedur. Twierdzi się, że jeżeli procedury te są dostatecznie często powtarzane, opłaca się odwzorować w strukturze bazy danych relacje między wskaźnikami, wynikające z takiej procedury. Dodajmy, że efektywność takich przedsięwzięć nie jest wystarczająco zba-

dana, aby można było postępowanie takie zalecić jako rutynę. Sądzimy, że zależeć to będzie od złożoności obliczeń, liczby wskaźników elementarnych zaangażowanych do realizacji procedury, liczby wskaźników zespolonych, jak i innych czynników (częstotliwość korzystania z procedury itp.).

Przedstawimy formalizację opisu różnych relacji w modelowaniu semantycznym bazy danych. Formalizacja taka ułatwia opracowanie modelu pojęciowego bazy danych. Odpowiada ona *metodzie szablonów semantycznych* (frame'ów).

Przyjmijmy, że wszystkie wskaźniki tworzą zbiór R. W zbiorze tym można utworzyć, według z góry określonej zasady, podzbiory wskaźników. Zasadą wyodrębnienia podzbiorów nie musi być hierarchia, ale przykładowo wyróżnić można podzbiór wskaźników dotyczących jakiegoś pojedynczego obiektu.

Do określenia wzajemnych zależności wskaźników wykorzystano elementy teorii mnogości [9]. W celu przeniesienia relacji zachodzących w teorii mnogości na zbiory wskaźników, musi być określone *pole wskaźnika*. Tak więc pole wskaźnika będą tworzyły wszystkie obiekty lub zjawiska ekonomiczne, których dotyczy dany wskaźnik. Pole to będzie oznaczane symbolem $V(R)$.

Między zbiorami wskaźników ekonomicznych zachodzą zależności analogiczne do zależności zachodzących w teorii mnogości.

1. Mówimy, że wskaźnik R_i obejmuje wskaźnik R_j, jeżeli pole wskaźnika R_i obejmuje pole wskaźnika R_j, co zapisuje się:

$$V(R_i) \cap V(R_j) = V(R_j).$$

Formalnie zależność ta będzie zapisywana $R_i \supset R_j$, gdzie symbol $\supset$ jest symbolem obejmowania pól wskaźników.

Przykład:

R_i — sprzedaż wyrobów, robót i usług własnej produkcji w cenach zbytu w tys. zł przedsiębiorstw i zakładów przemysłowych uspołecznionych działu przemysł,

$V(R_i)$ — wyroby, roboty i usługi własnej produkcji przeznaczone na sprzedaż,

[9] Por. A. Wasilianskas, i inni, Osnowy awtomatizacyi usprawlienija narodnym choziajstwom riespubliki, Wilnius 1975.

R_j — sprzedaż wyrobów, robót i usług własnej produkcji w cenach zbytu w tys. zł przedsiębiorstw i zakładów przemysłowych uspołecznionych działu przemysł,

$V(R_j)$ — wyroby, roboty i usługi własnej produkcji przeznaczone na eksport.

Łatwo zauważyć, że pole $V(R_i)$ obejmuje pole $V(R_j)$, gdyż wyroby, roboty i usługi przeznaczone na eksport są częścią wyrobów, robót i usług sprzedawanych przez przedsiębiorstwa. Oznacza to, że sprzedaż eksportowa mieści się w ogólnej sprzedaży tych przedsiębiorstw i zakładów.

2. Mówimy, że zakres wskaźnika R_i jest zawarty we wskaźniku R_j, jeżeli pole wskaźnika R_i jest zawarte w polu wskaźnika R_j, co zapisuje się:

$$V(R_i) \cap V(R_j) = V(R_i).$$

Formalny zapis ma postać $R_i \subset R_j$.

Przykład:

R_i — sprzedaż wyrobów, robót i usług własnej produkcji w cenach zbytu w tys. zł przedsiębiorstw i zakładów przemysłowych uspołecznionych działu przemysł w miesiącu sprawozdawczym,

$V(R_i)$ — wyroby, roboty i usługi własnej produkcji przeznaczone na sprzedaż w miesiącu sprawozdawczym,

R_j — sprzedaż wyrobów, robót i usług własnej produkcji w cenach zbytu w tys. zł przedsiębiorstw i zakładów przemysłowych uspołecznionych działu przemysł od początku roku do końca miesiąca sprawozdawczego,

$V(R_j)$ — wyroby, roboty i usługi własnej produkcji przeznaczone na sprzedaż od początku roku do końca miesiąca sprawozdawczego.

W przykładzie pole wskaźnika $V(R_i)$ zawarte jest w polu wskaźnika $V(R_j)$, czyli zachodzi zależność ze $R_i \subset R_j$, gdyż sprzedaż wyrobów, robót i usług w miesiącu sprawozdawczym będzie częścią sprzedaży od początku roku do końca miesiąca sprawozdawczego.

3. Dwa zbiory wskaźników ekonomicznych R_i oraz R_j przecinają się, jeżeli:

$$V(R_i) \cap V(R_j) \neq \phi.$$

Oznacza to, że zbiory te przecinają się, jeżeli część wspólna ich pól nie jest zbiorem pustym.

Przykład:

R_i — sprzedaż wyrobów produkcji przemysłowej w tys. zł w przedsiębiorstwach należących do działu transportu,

$V(R_i)$ — wyroby produkcji przemysłowej przeznaczone na sprzedaż wytworzone w dziale transportu,

R_j — sprzedaż wyrobów produkcji przemysłowej w tys. zł w przedsiębiorstwach wykonujących produkcję przemysłową,

$V(R_j)$ — wyroby produkcji przemysłowej produkowane w przedsiębiorstwach wykonujących produkcję przemysłową.

Częścią wspólną pól wskaźników $V(R_i)$ oraz $V(R_j)$ może być przykładowo produkcja części zamiennych do pojazdów, uznawana za produkcję przemysłową, a produkowana w dziale transportu.

4. Dwa zbiory wskaźników R_i oraz R_j są rozłączne, jeżeli ich pola nie mają wspólnych elementów:

$$V(R_i) \cap V(R_j) = \phi.$$

Przykład:

R_i — sprzedaż wyrobów, robót i usług własnej produkcji w cenach zbytu w tys. zł przedsiębiorstw działu przemysł, w miesiącu sprawozdawczym na zaopatrzenie rynku,

$V(R_i)$ — wyroby, roboty i usługi własnej produkcji przeznaczone na zaopatrzenie rynku,

R_j — sprzedaż wyrobów, robót i usług własnej produkcji w cenach zbytu w tys. zł przedsiębiorstw działu przemysł, w miesiącu sprawozdawczym na eksport,

$V(R_j)$ — wyroby, roboty i usługi własnej produkcji przeznaczone na eksport.

Pola omawianych zbiorów nie mają części wspólnej, gdyż wyroby sprzedane na eksport nie są przeznaczone na zaopatrzenie rynku.

Przedstawimy dalej tworzenie modelu bazy danych zgodnie z podejściem semiotycznym.

Przyjmijmy, że zbiór wskaźników ekonomicznych R, które chcemy gromadzić w bazie danych można rozpatrywać jako całość złożoną z podzbiorów $R = \{R_1 \ldots R_n\}$, między którymi zachodzą relacje obejmowania i zawierania zbiorów. W tak skonstruowanym zbiorze wskaźników ekonomicznych R można utworzyć hierarchiczną strukturę zależności między podzbiorami wskaźników $R_1 \ldots R_n$. W tym celu należy określić ciąg rekursywny podzbiorów, którego s-ty wyraz ma postać:

$$Z_s(R) = \{R_i / R_i \subset R_j;\ R_i,\ R_j \in Z_{s-1}(R)\},\quad s = 1 \ldots n$$

przy czym przyjmuje się, że $Z_0(R) = R$.

Zapis ten oznacza, że $Z_1(R)$ jest takim podzbiorem R, że zawiera przynajmniej jeden element ze zbioru R. Analogicznie $Z_2(R)$ jest takim podzbiorem $Z_1(R)$, że zawiera przynajmniej jeden jego element itd. Przyjmujemy, że $h(R)$ jest najmniejszą liczbą całkowitą, dla której jest spełniony warunek:

$$Z_{h(R)+1}(R) = \phi.$$

Liczbę $h(R)+1$ nazywać będziemy *głębokością zbioru wskaźników ekonomicznych* R.

W zbiorze R możemy zdefiniować taki podzbiór, że:

$$Y_s(R) = Z_{s-1}(R) \setminus Z_s(R) \quad s = 1, \ldots, h(R)+1,$$

gdzie $\setminus$ jest symbolem odejmowania zbiorów.

Zbiór R, w którym utworzona została hierarchiczna struktura zależności na podstawie opisanego już ciągu rekursywnego, i w którym zdefiniowana jest operacja odejmowania, można przedstawić za pomocą grafu.

Dalszy etap postępowania polega na określeniu *pola asocjacyjnego* podzbiorów wskaźników ekonomicznych R_i, które spełniają określone już relacje obejmowania i zawierania. W tym celu zostały określone dwa następujące zbiory:

$$\underline{Z_1}(R_i) = \{R_j / R_j \subset R_i;\quad j = 1 \ldots n,\ i \neq j\},$$

co oznacza, że elementami zbioru $\underline{Z_1}(R_i)$ są takie podzbiory R_j, których zakres zawarty jest w podzbiorze R_i.

$$\bar{Z}_1(R) = \{R_j / R_j \subset R_i;\quad j = 1 \ldots n,\ i \neq j\},$$

co oznacza, że elementami zbioru $\bar{Z}_1(R_i)$ są takie podzbiory R_j, które obejmują zakres podzbioru R_i.

Polem asocjacyjnym podzbioru wskaźników R_i w zbiorowości wskaźników R nazywany jest następująco zdefiniowany zbiór:

$$Z(R_i) = \underline{Z}_1(R_i) \cup \bar{Z}_1(R_i),$$

co oznacza, że pole asocjacyjne podzbioru R_i tworzą wszystkie takie podzbiory, które należą do zbioru R_i lub które obejmują podzbiór R_i.

$\underline{Z}_1(R_i)$ jest *górną warstwą pola asocjacyjnego*,

a $\bar{Z}_1(R_i)$ jest *dolną warstwą pola asocjacyjnego* podzbioru wskaźników R_i ze zbiorowości R.

Na podstawie określonych już działań i własności zbioru R oraz jego podzbiorów można zbudować hierarchiczną strukturę pola asocjacyjnego podzbioru R_i. W tym celu wykorzystując wprowadzoną już zależność rekursywną określa się następujące zależności dla podzbiorów R_i:

$$\underline{Z}_s(R_i) = \{R_j | R_j \subset R_k;\ R_j,\ R_k \in \underline{Z}_{s-1}(R_i)\}, \quad s = 2, ..., n.$$

Jest to zależność zachodząca w górnej warstwie pola asocjacyjnego podzbioru R_i. Oznacza to, że wyrazami ciągu są takie podzbiory R_j, które są zawarte w podzbiorze R_k, a obydwa podzbiory R_j oraz R_k należą do poprzedniego elementu ciągu.

Analogiczna zależność zachodzi w dolnej warstwie pola asocjacyjnego:

$$\bar{Z}_s(R_i) = \{R_j | R_j \supset R_k;\ R_j,\ R_k \in \bar{Z}_{s-1}(R_j)\}, \quad s = 2, ..., n.$$

Wyrazami ciągu są takie podzbiory R_j, które obejmują podzbiór R_k, a obydwa podzbiory R_j oraz R_k należą do poprzedniego elementu ciągu.

Przez $\underline{h}(R_i)$ oraz $h(R_i)$ będą oznaczone najmniejsze liczby całkowite, dla których $[\underline{h}(R_i)+1]$ element ciągu będzie zbiorem pustym.

$$\underline{Z}[\underline{h}(R_i)+1](R_i) = \phi,$$

analogicznie

$$Z[h(R_i)+1](R_i) = \phi.$$

Liczbę $\underline{h}(R_i)$ będziemy nazywać *stopniem agregacji*, a liczbę $\bar{h}(R_i)$ — *stopniem dezagregacji* podzbioru R_i w zbiorze wskaźników R.

W dolnej i górnej warstwie pola asocjacyjnego podzbioru R_i określa się kolejność podzbiorów zgodnie z następującym wzorem:

$$\underline{Y}_s(R_i) = \underline{Z}_s(R_i) \setminus Z_{s+1}(R_i), \quad s = 1 \ldots \underline{h}(R_i),$$
$$\bar{Y}_s(R_i) = \bar{Z}_s(R_i) \setminus Z_{s+1}(R_i), \quad s = 1 \ldots \bar{h}(R_i).$$

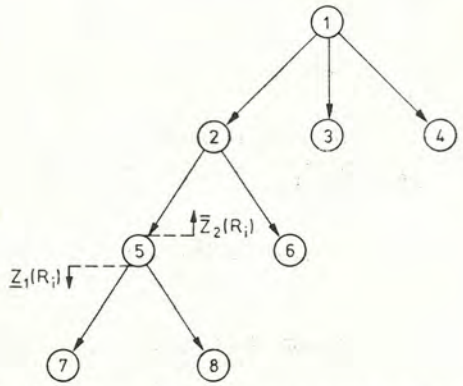
Zbiór wskaźników ekonomicznych R wraz z zachodzącymi między podzbiorami $\{R_1 \ldots R_n\}$ zależnościami zawierania i obejmowania zakresów można przedstawić w postaci grafu. Przy budowie takiego grafu, dla uproszczenia, nie wyodrębnia się górnej i dolnej warstwy pola asocjacyjnego, tylko wykorzystuje się wzór uproszczony:

$$Z(R_i) = \underline{Z}_1(R_i) \cup \bar{Z}_1(R_i).$$

Model grafu można jeszcze bardziej uprościć do postaci przedstawionej na rysunku 9, stosując regułę określającą kolejność podzbiorów w postaci:

$$Y(R_i) = \underline{Y}_1(R_i) \cup \bar{Y}_1(R_i).$$

Usystematyzowany zbiór wskaźników R ma budowę hierarchiczną, a struktura jego jest bardzo skomplikowana. Powstaje problem, który polega na tym, że zachodzą bezpośrednie zależności między elementami podzbiorów, będących na różnych poziomach hierarchii, a nie tylko na poziomach sąsiadujących ze sobą.



Rys. 9. Graf uproszczony z uwzględnionym polem asocjacyjnym

W celu uproszczenia struktury hierarchicznej i wprowdzenia zależności między sąsiednimi stopniami hierarchii, wprowadza się pojęcie zbioru ciągłego, który jest zdefiniowany w następujący sposób: *zbiorem ciągłym* wskaźników ekonomicznych nazywamy taki zbiór R, że dla każdego R_i należącego do R spełniony jest następujący warunek:

$$\underline{Y}_1(R_i) \subset Y[\bar{h}(R_i)+2]\,(R).$$

Konsekwencją wprowadzenia swego rodzaju „ciągłości" w zbiorze wskaźników R jest to, że w modelu geometrycznym istnieją tylko te krawędzie grafu, które łączą wierzchołki sąsiadujących ze sobą poziomów. Założenie to ma duże znaczenie dla budowania struktury zbioru, która znacznie się upraszcza.

Wprowadzenie warunku „ciągłości" pociąga za sobą przebudowę i reorganizację całego zbioru; aby zbiór nieciągły zamienić na ciągły, zachodzi konieczność wprowadzenia dodatkowych podzbiorów. Nowo wprowadzone wierzchołki uzupełniają „brakujące" szczeble hierarchii i z punktu widzenia informacyjnego nie przedstawiają sobą żadnej wartości (są to wierzchołki „puste"). Wprowadza się także wierzchołki, które dublują treść informacyjną innych wierzchołków. Wierzchołki te po to zostały wprowadzone, aby zachować powiązania między istniejącymi szczeblami hierarchii, przy zachowaniu warunku ciągłości.

Narzucenie warunku „ciągłości" na zbiór wskaźników ekonomicznych R powoduje pojawienie się nowych zależności, zachodzących między jego podzbiorami.

1. W ciągłym zbiorze wskaźników ekonomicznych R dla $R_j \subset \underline{Y}_s(R_i)$ zachodzi odpowiednio:

$$\bar{h}(R_j) = \bar{h}(R_i)+s.$$

Własność ta wiąże stopień dezagregacji dwóch podzbiorów wskaźników ekonomicznych, wyłonionych ze zbioru R, w tym sensie, że jeżeli znany jest stopień dezagregacji jednego podzbioru wskaźników, stopień zajmowany w hierarchii oraz inny podzbiór w polu asocjacyjnym pierwszego, to można określić stopień dezagregacji drugiego podzbioru.

Przykład:

101 — sprzedaż wyrobów, robót i usług własnej produkcji w cenach zbytu w tys. zł przedsiębiorstw i zakładów prze-

mysłowych uspołecznionych działu przemysł w miesiącu sprawozdawczym,

z tego:

103 — na zaopatrzenie rynku,

105 — na eksport,

107 — na cele zaopatrzenia materiałowo-technicznego,

w tym:

108 — przedsiębiorstwom przemysłowym,

w tym:

109 — dostawy kooperacyjne,

111 — jednostkom handlu zaopatrzeniowego,

112 — na cele inwestycji krajowych,

114 — na inne cele,

w tym:

115 — konsumentom zbiorowym.

$R_j = \{108, 111\}$ $R_j \in \underline{Y}_2(R_i)$,

$R_i = \{103, 105, 107, 112, 114\}$,

$\bar{h}R_i = 1$; $\bar{h}(R_j) = 1+2 = 3$.

2. W ciągłym zbiorze wskaźników ekonomicznych R dla $R_j \in \bar{Y}_s(R_i)$ zachodzi odpowiednio:

$$\bar{h}(R_j) = \bar{h}(R_i - s, \quad s = 1 \ldots \bar{h}(R_i).$$

Własność ta jest analogiczna do poprzednio omówionej. Dotyczy zależności zachodzących w dolnej warstwie pola asocjacyjnego podzbioru R_i, jeżeli podzbiór wskaźników ekonomicznych R_j zajmuje s-ty stopień w hierarchii zbioru R oraz jest elementem tworzącym górną warstwę pola asocjacyjnego podzbioru R_i. Stopień dezagregacji podzbioru wskaźników R_j jest równy stopniowi dezagregacji podzbioru R_i pomniejszonemu o 3.

Przykład:

Weźmy poprzedni przykład. Jeżeli za R_i przyjmiemy 109, a za R_j przyjmiemy 107, to $R_j \in \bar{Y}_2(R_i)$, a $\bar{h}(R_i) = 4$, to

$$\bar{h}(R_j) = 4-2 = 2.$$

3. W ciągłym zbiorze wskaźników ekonomicznych górna warstwa pola asocjacyjnego podzbioru R_i, będącego na s-tym stopniu

hierarchii, jest zawarta w podzbiorze elementów zbioru R znajdujących się na $\bar{h}(R_i)+s+1$ stopniu hierarchii:

$$Y_s(R_i) \subset Y_{\bar{h}(R_i)+s+1}(R), \quad s = 1 \ldots h(R_i).$$

Właściwość ta mówi o elementach podzbioru R_i, znajdujących się na s-tym stopniu w górnej warstwie pola asocjacyjnego R_i. Elementy te są zawarte w elementach będących na $\bar{h}(R_i)+s+1$ stopniu hierarchii zbioru R.

Przykład:

$109 \in Y_2(R_i)$
$\{103, 105, 107, 112, 114\} = R_i$
$\bar{h}(R_i) = 1$
Wynika z tego, godnie z tą regułą, że 109

$$109 \in Y_{\bar{h}(R_i)+s+1\,(R)} = Y_{1+2+1\,(R)} = Y_{4(R)},$$

czyli 109 należy do czwartego stopnia hierarchii zbioru R.

4. Jest to właściwość zbudowana analogicznie do własności 3. W ciągłym zbiorze wskaźników ekonomicznych R zachodzi:

$$Y_s(R_i) \subset Y_{\bar{h}(R_i)-s+1\,(R)}.$$

Własność ta mówi o tym, że elementy podzbioru R_i należące do jego górnej warstwy pola asocjacyjnego i będące w nim na s-tym stopniu hierarchii są równocześnie zawarte w elementach zbioru R należących do $\bar{h}(R_i)_{-s+1}$ stopni jego hierarchii.

Przykład:

$109 \in (R_i)$
$\{103, 105, 107, 112, 114\} \in \bar{Y}_2(R_i)$
$\bar{h}(R_i) = 3$
Wynika stąd, że zgodnie z regułą z pkt. 4:

$$107 \in Y_{\bar{h}(R_i)-s+1(R)} = Y_{3-2+1}(R) = Y_2(R).$$

5. W ciągłym zbiorze wskaźników ekonomicznych R, elementy będące na $s+1$ stopniu jego hierarchii, równe są sumie elementów z pierwszej dolnej warstwy pola asocjacyjnego podzbioru R_i, jeżeli podzbiór R_i jest złożony z elementów zbioru R, będących na s-tym stopniu jego hierarchii, co zapisuje się:

$$Y_{s+1}(R) = \bigcup_{R_i \in Y_s(R)} \underline{Y}_1(R_i), \quad s = 1, ..., h(R)$$

Przykład:

$s = 2$
$Y_3(R) = \{108, 111, 115\}$
$R_i \in Y_2(R) = \{103, 105, 107, 112, 114\}$
$R_1 = \{107\}$
$R_2 = \{114\}$
$\underline{Y}_1(R_1) = \{108, 111\}$
$\underline{Y}_1(R_2) = \{115\}$
$\underline{Y}_1(R_i) = \{108,, 111, 115\}$
R_i

W celu uproszczenia, przy wykorzystaniu podanych własności graficznego modelu ciągłego zbioru wskaźników ekonomicznych wprowadzimy pojęcie *zbioru ciągłego o strukturze drzewiastej.*

Zbiór ciągły wskaźników ekonomicznych ma *strukturę drzewiastą*, jeżeli:

$$\underline{Y}_1(R_i) \cup \underline{Y}_1(R_j) = \phi \text{ dla wszystkich } R_i, R_j \in Y_s(R); \quad s = 1, ..., h(R).$$

Oznacza to, że ciągły zbiór wszystkich ekonomicznych R ma strukturę drzewiastą, jeżeli pierwsze stopnie hierarchii górnych warstw pól asocjacyjnych wskaźników ekonomicznych należących do tego samego stopnia hierarchii zbioru R nie przecinają się.

W geometrycznej interpretacji zbiór ma strukturę drzewa, jeżeli nie posiada wierzchołków, od których odchodzi w dół więcej niż jedna krawędź.

Drzewiastą *strukturę* zbioru wskaźników ekonomicznych nazywa się *hierarchicznie uporządkowaną*, jeżeli:

$$V(R_i) \cap V(R_j) = \phi \text{ dla wszystkich } R_i, R_j \in Y_s(R); \quad s = 1, ..., h(R)+1.$$

Oznacza to, że drzewiasta struktura wskaźników ekonomicznych R jest hierarchicznie uporządkowana, gdy wskaźniki ekonomiczne należące do tego samego stopnia hierarchii zbioru R nie przecinają się polami.

Drzewiasta *struktura* ciągłego zbioru wskaźników ekonomicznych jest *proporcjonalnie uporządkowana*, jeżeli zbiór jest hierarchicznie uporządkowany oraz zachodzi:

$$\bigcup_{R_j \in Y_1(R_i)} V(R_j) = V(R_i) \text{ dla wszystkich } R_i \in R \setminus Y_{h(R)+1}(R)$$

tzn., jeżeli pole podzbioru wskaźników R_j należy do pierwszego stopnia hierarchii górnej warstwy pola asocjacyjnego podzbioru wskaźników R_i.

Niezwykle istotną cechą wzajemnych stosunków między poszczególnymi podzbiorami wskaźników ekonomicznych jest ich *niespójność*.

Dwa podzbiory wskaźników ekonomicznych będą uważane za *niespójne*, jeżeli do określania wskaźników ekonomicznych zawartych w poszczególnych podzbiorach używane są różnorodne (nie te same) atrybuty. Jeżeli do zbioru R należą przynajmniej dwa niespójne wskaźniki, to zbiór ten nazywany jest *zbiorem niespójnych wskaźników*.

Formalny zapis takiego zbioru pozostaje bez zmian: $R = \{R_1 \ldots R_n\}$. Własności zawierania i obejmowania zakresów zbiorów są określone analogicznie, jak w poprzedniej części rozdziału. Określone są także zbiory $Z_s(R)$ oraz $Y_s(R)$, a także głębokość zbioru $h(R)+1$.

Zbiory $Z_s(R)$ oraz $Y_s(R)$ są to zbiory nieprzecinające się, odpowiadające podzbiorom zgodnych wskaźników ekonomicznych $R^{(i)}$ co oznacza, że:

$$R = \bigcup_{i=1}^{m} R^{(i)}$$

oraz

$$Z_s(R) = \bigcup_{i=1}^{m} Z_s(R^{(i)})$$

$$Y_s(R) = \bigcup_{i=1}^{m} Y_s(R^{(i)})$$

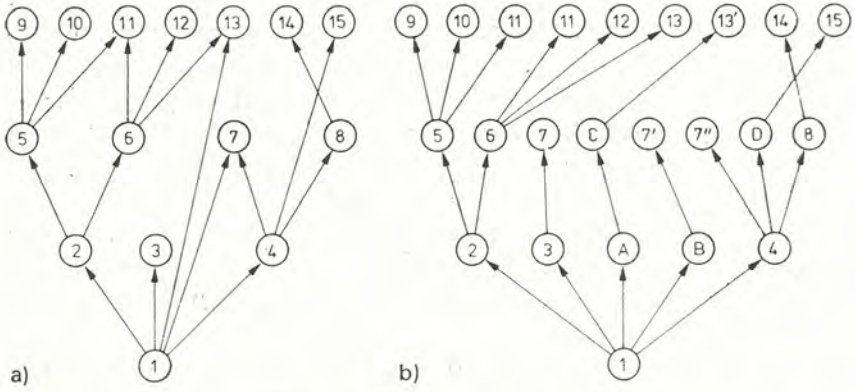
gdzie m jest liczbą podzbiorów zbioru R.

Głębokość zbioru R jest równa maksymalnej głębokości podzbioru zgodnych wskaźników ekonomicznych:

$$h(R) = \max_{1 \leqslant i \leqslant m} h(R^{(i)})$$

Graf zbioru niespójnych wskaźników ekonomicznych „rozpada się" na m oddzielnych podgrafów, odpowiadających podzbiorom spójnych wskaźników ekonomicznych, które tworzą zespolone wskaźniki ekonomiczne.

Zbiór wskaźników ekonomicznych R jest cykliczny, jeżeli można go rozbić na niewielkie grupy ekwiwalentnych, nie przecinających się podzbiorów $R^{(i)}$, $i = 1 \ldots m$, ale zachowujące własności zawierania i obejmowania zakresów zbiorów. Takimi nie przecinającymi się ekwiwalentnymi podzbiorami są wskaźniki zespolone. Ilustracją cykliczności w zbiorze wskaźników ekonomicznych jest graf z rysunku 10(a).



Rys. 10. Porównanie grafów zbiorów: nieciągłego (a) i ciągłego (b)

Poszczególne elementy zbioru ciągłego (niezależnie od stopnia hierarchii) są opisywane przez identyczny system wskaźników, tzn. produkcję na każdym szczeblu określają wskaźniki o identycznych nazwach.

Odwzorowując opisane relacje w strukturze bazy danych otrzymujemy bazę odpowiadającą strukturze języka użytkownika; profilowi zapytań formułowanych przez użytkownika. O ile więc infologiczna baza danych jest zorientowana na odwzorowanie struktury rzeczywistości, o tyle „semiotyczna" baza danych jest zorientowana na odwzorowanie potrzeb użytkownika finalnego, odzwierciedlonych przez strukturę jego języka.

Jak już mówiliśmy, dość wygodną formą opisu struktury treści

bazy danych jest formuła fasetowa. Przez formułę fasetową rozumiemy kolejność występowania klas deskryptorów w opisie danego obiektu, w tym wypadku wskaźnika ekonomicznego. Dla każdego wskaźnika zespolonego możemy opracować taką formułę fasetową. Każdą z faset możemy rozpisać na deskryptory. Organizacja deskryptorów w fasecie może przyjmować różne formy: klasyfikacji, nomenklatury, systematyki, listy słów kluczowych itp. Natomiast same fasety między sobą są powiązane jedynie relacją „następowania po", wskutek czego tracimy możliwość odzwierciedlenia bardziej złożonych powiązań między wskaźnikami, które daje formuła szablonu semantycznego. Natomiast formuła fasetowa jako podstawa organizacji bazy danych jest cenna wówczas, gdy dominującą funkcją systemu bazy danych jest wyszukiwanie informacji. Stąd też polecamy formułę fasetową jako podstawę budowy modelu semantycznego bazy danych w autonomicznych systemach metadanych, np. rejestrach obiektów, rejestrach osób, słownikach, systemach informatycznych prowadzenia nomenklatur i klasyfikacji. Formuła ta zapewnia bowiem możliwość łatwego przejścia na którykolwiek model bazy danych stosowany w systemach wyszukiwania informacji, a równocześnie dostarcza pełny i precyzyjny opis języka użytkownika finalnego. Wadą jest natomiast trudność wmontowania w taki język procedur obliczeń numerycznych na wartościach danych, gdyż wykraczają one poza funkcję wyszukiwania informacji [10].

*

Omówione tu dwa podejścia analizy i syntezy semantycznej baz danych znalazły już zastosowania, mimo że nie doczekały się podręcznikowego omówienia. Sposoby ich wykorzystania w procesie projektowania modelu semantycznego bazy danych omawiamy w rozdziale IX.

Wynikiem prac analitycznych i syntezy treści bazy danych jest semantyczny model bazy danych. Jest on sformułowany w języku finalnego użytkownika systemu informatycznego i wprowadza jednak niezbędne ujednolicenia i standardy języka, zmniejszające różnorodność form opisu i ograniczające zakres informacji, jaki może być ujmowany w bazie danych. Mo-

[10] Por. *Ekonomiczeskaja informacyja*, praca zbiorowa, Moskwa 1974.

del semantyczny bazy danych odwzorowuje więc z jednej strony — potrzeby informacyjne użytkownika, a z drugiej — opisuje wybrany wycinek rzeczywistości w sposób umożliwiający zredukowanie informacji (które mają być gromadzone w bazie danych) do zakresu i formy strukturalnej minimalnych, niezbędnych z punktu widzenia funkcji, spełnianych przez system bazy danych. Metody infologiczna i semiotyczna dostarczają praktycznych wskazówek i metod budowy modelu semantycznego bazy danych.

Należy zwrócić uwagę również na fakt, że często bagatelizuje się znaczenie tej fazy pracy. Czynią to „technologicznie" ukierunkowani projektanci systemów informatycznych, którzy oczekują, że nie tylko model semantyczny, ale i model pojęciowy bazy danych zostaną im doręczone i będą niezmienne. Poprawne skonstruowanie, przejrzyste udokumentowanie modelu semantycznego zapewniają dobre podstawy do dalszych faz projektowania, dostarczają wiedzy niezbędnej do przeprowadzenia kompetentnej oceny projektu bazy danych i materiału do dalszego rozwoju systemu bazy danych.

III. Pojęciowy model danych

1. Struktura modelu semantycznego

Zadaniem systemu informacyjnego jest opis wybranego fragmentu rzeczywistości. W tym sensie system informacyjny jest reprezentacją fragmentu rzeczywistości, objętej zakresem jego działania. System informacyjny dokonuje opisu rzeczywistości przez zbiór wartości wybranych parametrów uznanych za charakterystyczne dla tej rzeczywistości. Zbiór parametrów określa się przez funkcje systemu. Wynika z tego, że jeden i ten sam fragment rzeczywistości może być w różnych systemach informacyjnych opisywany za pomocą różnych parametrów.

Opis rzeczywistości jest zbiorem powiązanych wzajemnie wartości parametrów. Wartości te ustala się dla określonych punktów czasowych. Mamy więc do czynienia ze statycznym modelem rzeczywistości. Równie istotnym aspektem systemu informacyjnego jest odwzorowanie zmian rzeczywistości, odwzorowanie zjawisk dynamicznych zachodzących w badanej rzeczywistości i znajdujących swoje odzwierciedlenie w zmianach wartości parametrów. Zmiany te prowadzą do zmian stanów systemu informacyjnego.

Zmiany dynamiczne, zachodzące w rzeczywistości, uwzględniamy w systemie informacyjnym w ten sposób, że wprowadzamy odpowiednie zmiany w modelu statycznym, wykonując odpowiednie modyfikacje na jego strukturze lub wartościach parametrów. Opis wszystkich operacji zmian, dokonywanych na elementach struktury systemu informacyjnego, stanowi model dynamiczny, nazywany również modelem funkcjonalnym. Tak więc każdy system informacyjny powinien obejmować zawsze pewien model

informacyjny (statyczny), model funkcjonalny (dynamiczny), które tworzą semantyczny model systemu oraz odpowiednią konfigurację sprzętu informatycznego.

Projektując bazę danych, musimy przede wszystkim rozwiązać wszystkie problemy, związane z konstrukcją modelu semantycznego. Możliwe podejścia do konstruowania tego modelu oraz strukturalne formy, jakie może przyjmować informacja w bazie danych, już omówiliśmy. W tym rozdziale skoncentrujemy się przede wszystkim na metodzie konstruowania modelu bazy danych, umożliwiającej przejście do dalszych faz projektowania bazy danych. Wiąże się to przede wszystkim z doborem technik dokumentowania modelu semantycznego. Proponowane techniki nie są jedynymi, jakie wypracowano w tej dziedzinie. Wybór metody notacji graficznej podyktowany był przede wszystkim tym, że jest ona względnie uniwersalna i nie wiąże projektanta z konkretnymi instrumentami informatycznymi komputerowego wspomagania prac projektowych, które w naszych warunkach są stosunkowo mało rozpowszechnione.

Przystępując do budowy modelu semantycznego musimy przede wszystkim jasno i precyzyjnie zdefiniować obiekt bazy danych. O problemach i trudnościach z tym związanych już mówiliśmy (por. rozdz. II). Zdefiniowanie obiektu baz danych chętnie zalicza się do prac „przedprojktowych" i zrzuca na barki użytkownika lub zamawiającego system informatyczny. Wydaje się jednak, że dopóki projektanci baz danych będą unikać aktywnego włączania się w tę fazę prac, a użytkownicy „zamawiający" bazę danych biernie godzić się na redukowanie tej fazy do minimum umożliwiającego przejście do faz następnych, powstanie niejedna chybiona baza danych.

W celu zdefiniowania obiektu bazy danych należy precyzyjnie i jednoznacznie określić zadania i funkcje systemu informatycznego i listę żądań jego użytkowników. Duża złożoność obiektów ekonomicznych sprawia, że sprostanie wymaganiom tej fazy jest trudne, wymaga dużego doświadczenia i wiedzy. Projektując bazę danych dla przedsiębiorstwa przemysłowego musimy widzieć to przedsiębiorstwo zarówno z punktu widzenia wykonywanej w nim produkcji, uzyskiwanych wyników ekonomicznych, ale także — jego oddziaływania na środowisko, powiązania z lokalną

infrastrukturą, wreszcie widzieć to przedsiębiorstwo w procesie jego rozwoju, w którym inwestowanie i doskonalenie technologii wytwarzania jest zaledwie jednym z elementów. Tak wieloaspektowego spojrzenia nie wymaga się od projektanta odcinkowych zastosowań systemów informatycznych. Projektantom „wychowanym" na zastosowaniach odcinkowych, zwłaszcza polegających na przetwarzaniu danych, przestawienie się tak wielostronne i kompleksowe spojrzenie na projektowany system informatyczny sprawia niejakie trudności. Stąd też, biorąc się za projektowanie bazy danych, trzeba robić to w uzasadnionym przekonaniu, że tak wysokim wymaganiom potrafi się sprostać.

Etap prac projektowych zmierzający do skonstruowania modelu semantycznego nazywamy analizą danych. Jego głównym celem jest dokonanie transformacji informacji, które powinny być ujęte w projektowanym systemie informacyjnym, na dane, jakie powinny znaleźć się w bazie danych. Przystępując do tego etapu powinno się zatem dysponować odpowiednią wiedzą o wybranym wycinku rzeczywistości, o funkcjach systemu informatycznego i potrzebach informacyjnych użytkowników. Charakterystyki te mogą być formułowane w języku naturalnym, a dla doświadczonego projektanta, znającego dobrze organizację, konstruowanie takiego opisu w języku naturalnym nie wydaje się niezbędne. Konieczna jest natomiast pełna i jasna specyfikacja funkcji systemu informatycznego, która uwzględnia wymagania użytkowników finalnych i wskazuje wzajemne uwarunkowania realizacji poszczególnych funkcji. Chodzi zwłaszcza o wskazanie, jakie funkcje systemu warunkują realizację innych funkcji użytkowych.

Kluczowym problemem analizy danych jest identyfikacja elementów występujących w wycinku rzeczywistości objętej systemem informatycznym.

Podstawowe fazy procesu analizy danych są następujące:

— selekcja elementów rzeczywistości, które uznajemy za istotne z punktu widzenia systemu informatycznego,

— wyznaczenie pojęć prototypowych, wyróżnionych za pomocą jednoznacznej nazwy, w formie zbioru nazw atrybutów obiektów należących do badanej rzeczywistości,

— wieloaspektowe poklasyfikowanie wszystkich wybranych

obiektów rzeczywistości, które opisywać chcemy w systemie informatycznym, na podstawie przyjętych pojęć prototypowych.

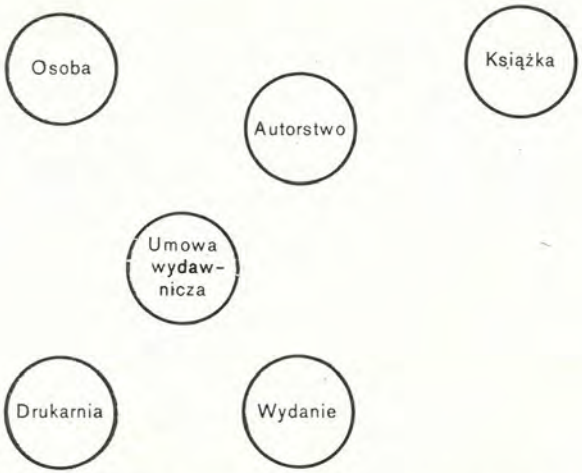
Objaśnimy to na prostym przykładzie. Przypuśćmy, że projektujemy bazę danych o pracownikach jakiejś organizacji. Ograniczenie się do pojęcia prototypowego „człowiek" może okazać się niewystarczające. Dla bardziej precyzyjnej identyfikacji zbioru obiektów opisywanych w bazie danych potrzebne są dodatkowe pojęcia prototypowe, takie jak: *mężczyzna, kobieta, dziecko* (według kryteriów płci i wieku). Wprowadzenie takich pojęć prototypowych pozwala na podział zbioru wszystkich ludzi zatrudnionych w tej organizacji na trzy podzbiory: zbiór mężczyzn, zbiór kobiet i zbiór dzieci.

Kryterium klasyfikacji oparte na pojęciach prototypowych pozwoli na określenie klas abstrakcji w zbiorze obiektów. Zaliczenie wybranego obiektu rzeczywistego do określonej klasy wynika z posiadania przez ten obiekt cech wspólnych z pojęciem prototypowym, przyjętym za kryterium wydzielania danej klasy elementów. Zwracamy uwagę na to, że, aby klasyfikować elementy według kryterium pojęć prototypowych, lista pojęć prototypowych nie musi spełniać warunków klasyfikacji. Ważne jest tylko, aby pole znaczeniowe tych pojęć było na tyle dobrze zdefiniowane, by można było jednoznacznie orzec w stosunku do każdego obiektu czy należy do klasy obiektów wyznaczonej przez dane pojęcie prototypowe, czy też nie.

Nazwa przyjęta dla danego pojęcia prototypowego może być wykorzystana jako nazwa klasy elementów wydzielonych według kryterium tego pojęcia. Metoda ta jest powszechnie stosowana w matematyce jako metoda identyfikacji elementów równoważnych.

Weźmy inny przykład. Przyjmijmy, że prowadzimy analizy danych w oficynie wydawniczej, a nasze rozważania doprowadziły do wyróżnienia takich pojęć prototypowych, jak *osoba, autorstwo, książka, umowa wydawnicza, wydanie i drukarnia.* Odpowiednio możemy dokonać klasyfikacji elementów rzeczywistości, zaliczając je do odpowiednich klas elementów. Zauważmy, że rozważone klasy elementów obejmują elementy, będące fizycznymi obiektami istniejącymi w rzeczywistości, takie jak *osoba,*

książka czy *drukarnia*, jak i pojęcia abstrakcyjne, jak *autorstwo* lub akty prawne, jak *umowa wydawnicza*. Istotną sprawą jest fakt, że każdy z rozważonych elementów ma wystarczająco wyraźne cechy, na podstawie których następuje zaliczenie go do odpowiedniej klasy elementów. Przyjęte klasy elementów przedstawia rysunek 11.

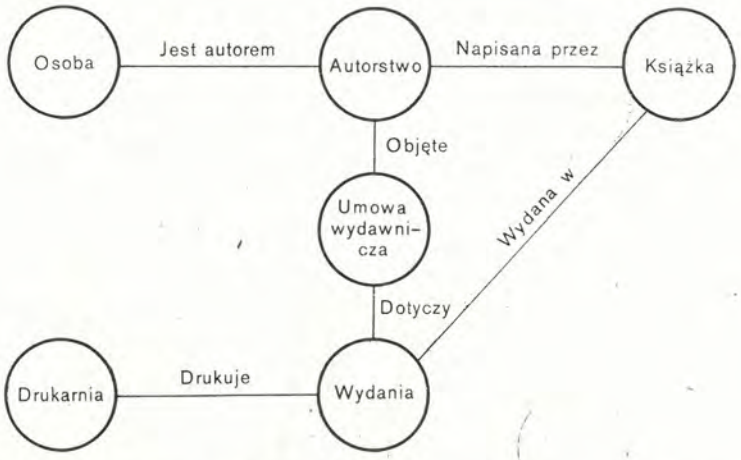


Rys. 11. Klasy elementów modelu semantycznego oficyny wydawniczej

Nietrudno zauważyć, że zawartość informacyjna tak przedstawionego modelu jest bardzo ograniczona. Informacje możliwe do uzyskania na jego podstawie ograniczają się jedynie do listy autorów współpracujących z danym wydawnictwem, listy książek wydanych i drukarni, z których korzysta to wydawnictwo. Trudno jest odpowiedzieć natomiast na tak proste pytanie jak, kto jest autorem danej książki, z jakimi osobami zawarto umowy wydawnicze lub ile wydań miała już określona książka. Możliwość wzbogacenia zawartości informacyjnej modelu daje wprowadzenie dodatkowej charakterystyki prezentowanych elementów przez określenie ich wzajemnych związków. Przyjmijmy, że związki między elementami modelu informacyjnego są relacjami dwuargumentowymi w iloczynie kartezjańskim zbiorów elementów, należących do dwóch różnych klas elementów. Rysunek 12

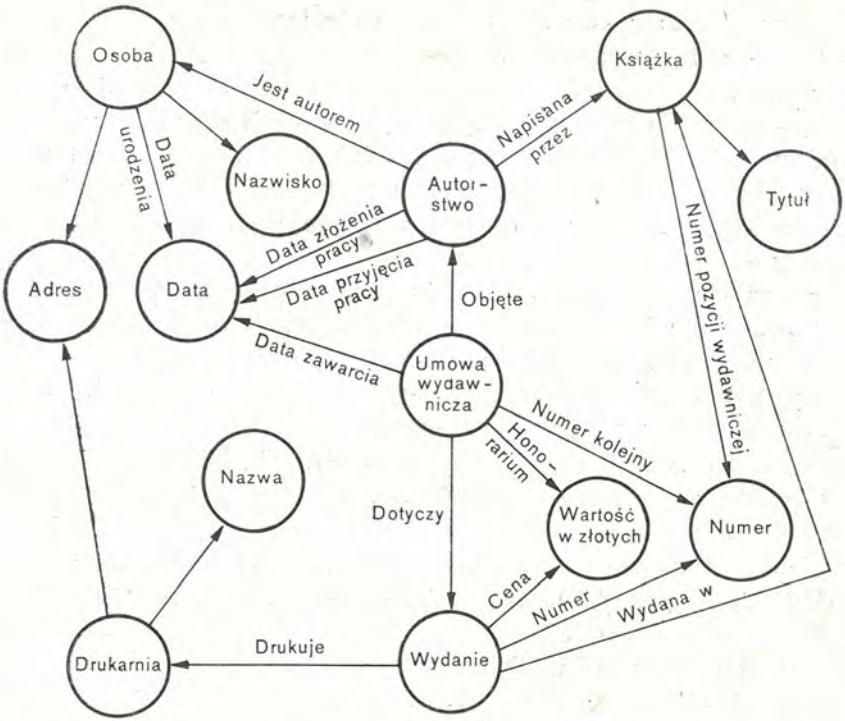
przedstawia takie rozwinięcie rozważanego modelu. Dodajmy, że również związki między elementami modelu informacyjnego mogą mieć przypisane nazwy.

Nietrudno spostrzec, że na podstawie takiego modelu można już odpowiedzieć na postawione poprzednio pytania.



Rys. 12. Związki klas elementów w modelu semantycznym oficyny wydawniczej

Mówiąc o pojęciach prototypowych stwierdziliśmy, że każde takie pojęcie jest związane z pewnym zespołem cech charakterystycznych, a elementy określone przez taki sam zbiór cech zaliczamy do wspólnej klasy elementów. Każdy z elementów (przedstawionych na rys. 11 i 12) należący do klas elementów można zapisać jako pewien zbiór wartości charakteryzujących go cech. Każda taka wartość może być potraktowana również jako element analizowanego fragmentu świata rzeczywistego. Dzięki temu możemy, w znany już sposób, wyróżnić dodatkowe klasy elementów, przy czym każda z takich klas obejmuje wszystkie te elementy, które mają przyjętą dla niej własność. W wypadku omawianego modelu informacyjnego oficyny wydawniczej zupełnie naturalne będzie występowanie takich klas elementów, jak *nazwisko, data, tytuł książki, numer, wartość w złotych* czy *adres*. Rozszerzona o te elementy struktura modelu semantycznego oficyny wydawniczej jest przedstawiona na rysunku 13.



Rys. 13. Rozwinięta struktura modelu semantycznego oficyny wydawniczej

Dodatkową cechą tego schematu w porównaniu z rysunkiem 12 jest oznaczenie strzałkami związków między obiektami. Wprowadzenie takiego oznaczenia wynika z faktu, że prezentowane związki elementów są relacjami dwuargumentowymi, a kierunek strzałki wyraża kolejność elementów pary uporządkowanej (od poprzednika do następnika), czyli określa dziedzinę i przeciwdziedzinę relacji dwuargumentowej. Zgodnie z definicją relacji dwuargumentowej określa ona pewną właściwość uporządkowanej pary elementów.

Zdanie „osoba jest autorem" oznacza, że ta osoba ma pewne prawa wynikające z faktu autorstwa napisanego tekstu. Natomiast zdanie „książka jest napisana przez" mówi nam, że istnieją pewne ograniczenia, w stosunku do prawa dysponowania tekstem dzieła, wynikające z zaistnienia autorstwa i określone przez pra-

wo autorskie. Gdyby zbiór osób objętych omawianym modelem informacyjnym rozszerzyć na wszystkie inne osoby (poza autorami) współpracujące z wydawnictwem, to określenie relacji „jest autorem" pozwala na stworzenie podzbioru elementów należących do klasy osób i mających tę właśnie cechę, czyli osób będących autorami. Możemy powiedzieć, że relacja dwuargumentowa określa rolę, jaką pełni element należący do jednej klasy elementów w stosunku do elementu należącego do drugiej klasy elementów.

Rozwinięcie struktury modelu informacyjnego polegało na wprowadzeniu klas elementów, których wartości charakteryzują inne klasy elementów. Z przedstawionego na rysunku 13 schematu wynika, że istnieją następujące zbiory cech charakterystycznych, odpowiadające poszczególnym klasom elementów:

OSOBA = NAZWISKO, DATA URODZENIA, ADRES
AUTORSTWO = DATA ZŁOŻENIA PRACY, DATA PRZYJĘCIA PRACY
KSIĄŻKA = TYTUŁ, NUMER POZYCJI WYDAWNICZEJ
UMOWA WYDAWNICZA = DATA ZAWARCIA, NUMER KOLEJNY, HONORARIUM
WYDANIE = NUMER, CENA
DRUKARNIA = NAZWA, ADRES

Łatwo zauważyć, że niektóre wartości cech charakterystycznych, mimo iż mają różny sens, pochodzą z tej samej klasy elementów. Na przykład *data złożenia pracy* i *data przyjęcia pracy*, charakteryzując elementy należące do klasy elementów *autorstwo*, mają zupełnie różny sens, mimo iż są elementami tego samego zbioru wartości. O ich sensie decyduje rola, jaką spełnia element opisujący jakąś własność w stosunku do elementu opisywanego. Dla uproszczenia nie wprowadzono nazw relacji w wypadku istnienia jednoznaczności.

Z rozważań tych wynika, że mamy do czynienia z różnymi funkcjami poszczególnych klas elementów w budowaniu struktury modelu semantycznego. Niektóre klasy elementów, jak *osoba, książka* czy *drukarnia* reprezentują obiekty istniejące w rozpatrywanej rzeczywistości, inne natomiast reprezentują powiązania między tymi obiektami. I tak klasa elementów *autorstwo* reprezentuje powiązanie między klasami elementów *osoba, książ-*

ka i *umowa wydawnicza*, a z kolei *umowa wydawnicza* reprezentuje powiązanie między klasami elementów *autorstwo* i *wydanie*. Natomiast *wydanie* reprezentuje powiązanie między klasą elementów *książka* a klasą elementów *drukarnia*. Jak widać, powiązania mogą zachodzić zarówno między klasami elementów reprezentujących obiekty istniejące w rzeczywistości, jak i klasami elementów, które reprezentują inne powiązania. Możliwość tworzenia takich powiązań pozwala na budowanie dowolnie złożonych struktur modelu semantycznego.

Elementy spełniające rolę cech charakterystycznych innych elementów (należących do innej klasy elementów) są wartościami prostymi, to jest wartościami, które są semantycznie niepodzielnymi fragmentami informacji. O ich sensie decyduje semantyka związku z charakteryzowaną klasą elementów. Opis struktury modelu semantycznego jest istotnym krokiem w kierunku stworzenia formalnego opisu struktury fragmentu świata rzeczywistego objętego projektowanym systemem informatycznym. Nie jest on jednak wystarczający dla przekazania wszystkich istotnych treści. Ważnym etapem budowy modelu semantycznego jest identyfikacja wszystkich istotnych warunków, jakie muszą spełniać jego elementy. Przykładem takich warunków mogą być dla modelu semantycznego takie zdania, jak „Autor nie może mieć mniej niż 15 lat" lub „Honorarium w ramach jednej umowy nie może przekraczać 50 tys. zł". Istnieje wiele takich warunków. Ich zapisanie i powiązanie z odpowiednimi klasami elementów daje dopiero możliwość pełnego opisu modelu semantycznego.

2. Podstawowe elementy pojęciowego modelu danych

Pojęciowy model danych jest formalnym opisem modelu semantycznego obejmującym wszystkie te elementy tego modelu, które są istotne z punktu widzenia projektowanego systemu informatycznego. Podstawowym wymaganiem stawianym przed takim formalnym opisem jest możliwość jednoznacznej identyfikacji związków między jego elementami a elementami opisywanego modelu semantycznego, przy czym przejście od modelu informacyjnego semantycznego do pojęciowego modelu danych nie może

powodować straty informacji. Drugim istotnym warunkiem jest niezależność pojęciowego modelu danych od technik implementacyjnych stosowanych na dalszych etapach tworzenia systemu informatycznego.

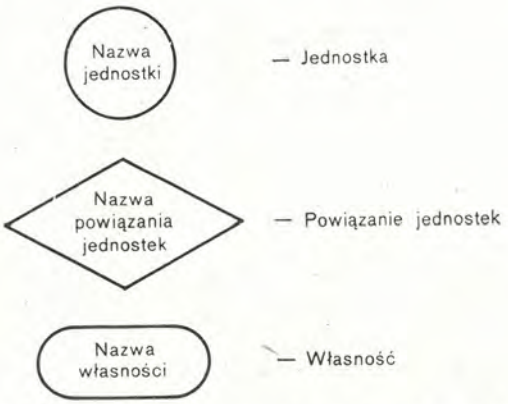
W praktyce mamy do czynienia z takim samym pojęciowym modelem danych dla systemu informatycznego budowanego tradycyjnymi technikami i dla systemu opartego na wspólnej bazie danych. Inaczej natomiast będzie w obu tych wypadkach wyglądało odwzorowanie modelu semantycznego na niższych poziomach systemu informatycznego.

W wypadku zastosowania systemu zarządzania bazą danych treść pojęciowego modelu danych powinna być przeniesiona w pełni do opisu logicznej struktury danych. W wypadku implementacji systemu informatycznego technikami tradycyjnymi semantyka pojęciowego modelu danych jest „zapisywana" w algorytmach oprogramowania systemu.

Omawiając elementy modelu semantycznego zwróciliśmy uwagę na różnorodność funkcji poszczególnych klas elementów z punktu widzenia tworzenia struktury tego modelu. Mieliśmy do czynienia z klasami elementów reprezentującymi obiekty istniejące w rzeczywistości, powiązania tych obiektów czy wreszcie ich cechy charakterystyczne. Stworzenie opisu pojęciowego modelu danych wymaga przyjęcia konwencji zapisu poszczególnych jego elementów. Najbardziej czytelna jest, naszym zdaniem, technika graficzna, przy czym przejście z tej techniki na dowolny język opisu pojęciowego modelu danych nie wydaje się zadaniem zbyt trudnym. Poszczególne typy elementów opisu pojęciowego modelu danych będą odpowiadały poszczególnym funkcjom tworzenia struktury modelu informacyjnego tworzącym odpowiednie grupy klas elementów tego modelu.

W naszych dalszych rozważaniach będziemy posługiwali się pojęciem *obiektu* lub *jednostki* (*entity*) jako reprezentacją klasy elementów modelu semantycznego odpowiadającym obiektom istniejącym i wyróżnionym w otaczającej nas rzeczywistości lub pojęciom abstrakcyjnym, związanym z rozważaną przez nas rzeczywistością, pojęciem *powiązania obiektów* jako reprezentanta tych klas elementów modelu informacyjnego, które odpowiadają powiązaniom obiektów występujących w rzeczywistości, oraz

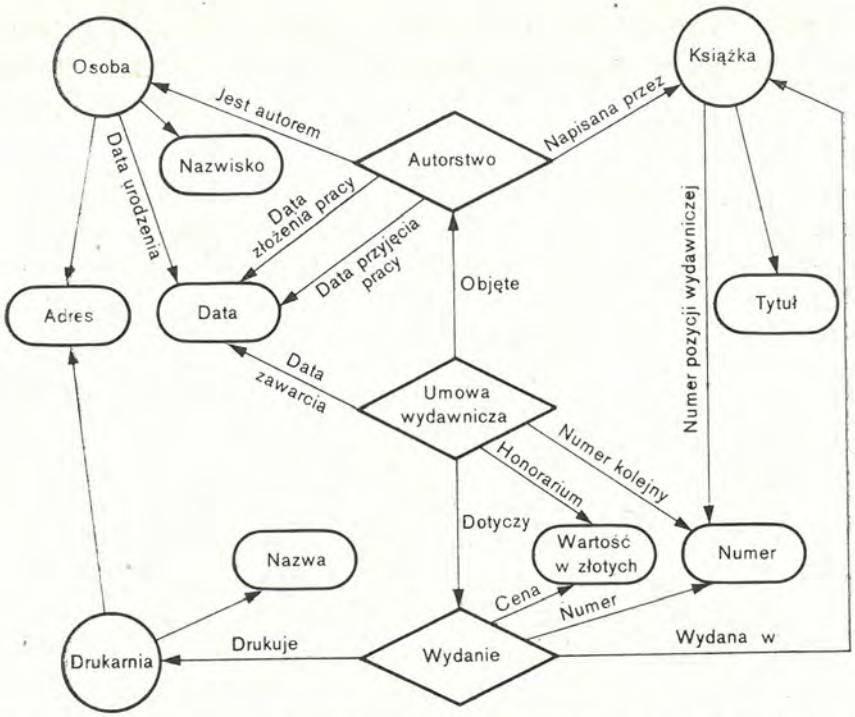
pojęciem *własności* jako reprezentanta tych klas elementów, które spełniają rolę cech charakterystycznych. Odpowiednie symbole graficzne przyjęte dla poszczególnych typów elementów opisu pojęciowego modelu danych przedstawiamy na rysunku 14. Podobnie jak w wypadku schematu struktury modelu semantycznego (por. rys. 13) strzałki łączące symbole graficzne reprezentują relacje dwuargumentowe, których elementami są pary uporządkowane elementów należących do klas elementów oznaczanych za pomocą tych symboli. Nazwy strzałek opisują rolę, jaką pełnią połączone nimi elementy.



Rys. 14. Symbole graficzne elementów opisu modelu pojęciowego bazy danych

Graficzne przedstawienie pojęciowego modelu danych odpowiadającego strukturze modelu informacyjnego oficyny wydawniczej (por. rys. 13) daje rysunek 15. Z porównania dwóch rysunków 13 i 15 wynika, że zawierają one te same informacje na temat struktury obiektów analizowanego fragmentu oficyny wydawniczej. Wyróżnienie różnych typów klas elementów modelu informacyjnego pozwoliło na uzyskanie czytelniejszego opisu. Zwróciliśmy już uwagę, że informacja w modelu semantycznym może być bardziej złożona, niż mogą to pokazać dotychczas stosowane schematy jego struktury.

Na podstawie analizy omawianego modelu semantycznego oficyny wydawniczej można sformułować następujące zasady semantyczne:



Rys. 15. Graficzna reprezentacja modelu pojęciowego oficyny wydawniczej

1) autor może napisać wiele książek,

2) książka może mieć wielu autorów,

3) każdy autor zawiera jedną umowę wydawniczą na jedną książkę, przy czym umowa dotyczy jednego wydania,

4) jedna książka może mieć wiele wydań drukowanych w różnych drukarniach,

5) osoba jest jednoznacznie identyfikowana przez jej nazwisko, natomiast książka przez numer pozycji wydawniczej.

Oczywiście można zidentyfikować dodatkowo wiele innych zasad semantycznych dotyczących omawianego modelu, dla uproszczenia przyjmiemy jednak, że zasady te wyczerpują wymagania projektowanego systemu.

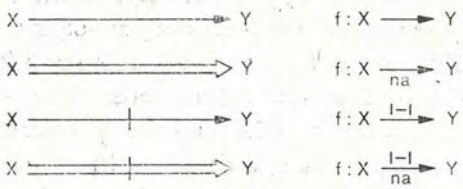
Z pierwszych dwóch zasad semantycznych wynika, że jedna osoba może mieć wiele różnych uprawnień wynikających z faktu autorstwa wielu książek oraz że ograniczenia założone z punktu

widzenia prawa autorskiego na jedną książkę wynikają z uprawnień wielu autorów. Wyraźnie widać, że te ograniczenia dotyczą jednostek *osoba* i *książka* oraz powiązania jednostek *autorstwo* i mogą być one wyrażone przez nałożenie odpowiednich warunków na związki między tymi pojęciami. W dotychczasowych rozważaniach przyjmowaliśmy, że związki między pojęciami opisu pojęciowego modelu danych są relacjami dwuargumentowymi. Zgodnie z tym związek między klasą elementów *osoba* i klasą elementów *autorstwo* może być wyrażony jako zbiór par uporządkowanych, gdzie pierwszy element pary zależy od klasy elementów *autorstwo*, a drugi od klasy elementów *osoba*. Z przyjętych zasad semantycznych wynika, że jeden element należący do klasy elementów *osoba* może być związany z wieloma elementami należącymi do klasy elementów *autorstwo*, natomiast fakt *autorstwa*, a raczej uprawnień wynikających z *autorstwa*, może być związany tylko z jedną *osobą*.

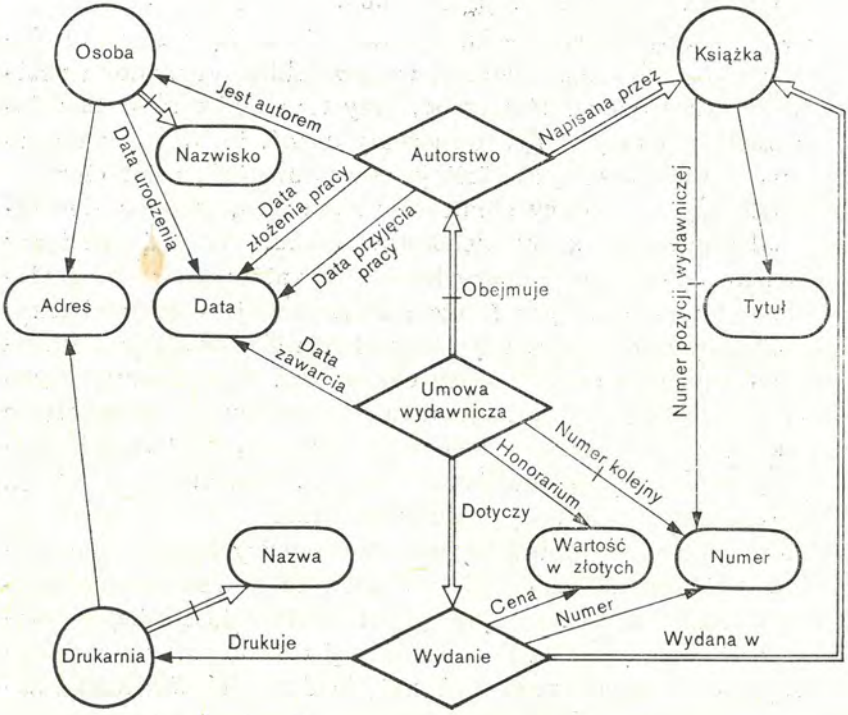
Jeżeli dodamy jeszcze, że każdy element należący do klasy elementów *autorstwo* musi być związany z jakimś elementem należącym do klasy elementów *osoba*, przy czym fakt *autorstwa* dotyczy zawsze jednej osoby, to możemy uznać, że rozważana relacja dwuargumentowa jest funkcją odwzorowującą zbiór elementów, należących do klasy elementów *autorstwo*, w zbiór elementów należących do klasy elementów *osoba*. Podobnie możemy powiedzieć, że związek między klasą elementów *autorstwo* i klasą elementów *książka* jest funkcją odwzorowującą zbiór elementów należących do pierwszej klasy elementów w zbiór elementów należących do drugiej klasy elementów. Wprowadźmy ograniczenie, że wszystkie relacje dwuargumentowe reprezentujące związki między klasami elementów będziemy traktowali jako funkcję, której zbiorem argumentów jest dziedzina tej relacji, a zbiorem wartości jest jej przeciwdziedzina.

Zaletą takiego traktowania związków między klasami elementów modelu semantycznego jest możliwość odwzorowania omawianych zasad semantycznych przez dobieranie funkcji o odpowiednich własnościach. Pojęcie funkcji możemy zawęzić, wprowadzając do naszych rozważań pojęcia funkcji różnowartościowej, gdzie jednemu elementowi zbioru wartości funkcji może odpowiadać tylko jeden element zbioru jej argumentów, oraz funk-

cji „na", gdzie każdemu elementowi zbioru wartości funkcji musi odpowiadać co najmniej jeden element zbioru jej argumentów. Symbole graficzne dla poszczególnych rodzajów funkcji reprezentujących związki klas elementów w opisie pojęciowego modelu danych przedstawia rysunek 16. Wprowadzenie funkcji jako



Rys. 16. Graficzne symbole funkcji reprezentujących związki między klasami elementów w modelu pojęciowym bazy danych



Rys. 17. Rozwinięty model pojęciowy danych oficyny wydawniczej

reprezentantów związków między klasami elementów modelu semantycznego jest zupełnie wystarczające dla opisania pojęciowego modelu danych, wyrażającego przyjęte zasady semantyczne. Fakt ten postaramy się dalej uzasadnić.

Rozwinięty pojęciowy model danych przedstawia rysunek 17.

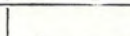
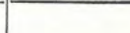
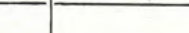
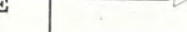
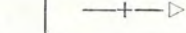
Powiązanie jednostek *autorstwo* jest powiązaniem typu N : M między klasą jednostek *osoba* a klasą jednostek *książka*, spełniając tym samym dwie pierwsze z przyjętych zasad semantycznych. Funkcje odpowiadające związkom tych klas obiektów z klasą powiązań obiektów *autorstwo* nie są różnowartościowe, czyli wiele faktów *autorstwa* wiąże jedną *książkę* z wieloma *osobami* i odwrotnie. Z przedstawionego schematu wynikają dodatkowe zasady semantyczne dotyczące omawianych klas obiektów, a mianowicie, każda *książka* ma przynajmniej jednego *autora* i nie wszystkie *osoby* są autorami. Pierwsza z tych zasad wynika z faktu, że związek *napisana przez* jest reprezentowany przez funkcję *na*, a funkcja określająca związek *jest autorem* nie ma tej własności.

Zasada mówiąca o tym, że każdy autor zawiera jedną umowę wydawniczą na jedną książkę, przy czym umowa może obejmować jedno wydanie, jest zapisana przez przedstawienie związku *obejmująca* jako funkcji jednoznacznej (funkcja różnowartościowa i *na*), a związku =dotyczy= jako funkcji *na*. Tak dobrana charakterystyka tych związków mówi nam, że jednej osoby autora i jednej napisanej przez niego książki dotyczy zawsze jedna umowa wydawnicza na jedno wydanie (z definicji funkcji). Natomiast jedno wydanie może być objęte wieloma umowami wydawniczymi (wielu autorów tej samej książki). Zauważmy, że z omawianego opisu semantycznego modelu danych wynika, iż sporządzenie każdego nowego wydania tej samej książki wymaga zawarcia nowej umowy wydawniczej. Odczytanie pozostałych zasad semantycznych wynikających z omawianego schematu pozostawiamy Czytelnikowi.

Jednym z istotnych problemów opisu pojęciowego modelu danych jest możliwość wyrażenia wszystkich wypadków struktur zachodzących w modelu semantycznym. Pojęcie powiązania obiektów spełnia naczelną rolę w budowaniu struktury semantycznego modelu danych, rozważmy więc możliwe warianty ich opisu przedstawione w tablicy 1.

Tablica 1

Możliwe warianty opisu powiązań jednostek w pojęciowym modelu danych

KLASA JEDNOSTEK A	ZWIĄZEK	KLASA POWIĄZAŃ JEDNOSTEK A—B	ZWIĄZEK	KLASA JEDNOSTEK B
1	2	3	4	5
V		N : M SŁABE		V
Λ		N : M PRAWO-STRONNIE SŁABE		V
V		N : M LEWO-STRONNIE SŁABE		Λ
Λ		N : M SILNE		Λ
V		1 : N SŁABE		V
Λ		1 : N PRAWO-STRONNIE SŁABE		V
V		1 : N LEWO-STRONNIE SŁABE		Λ
Λ		1 : N SILNE		Λ
V		1:1 SŁABE		V
Λ		1:1 PRAWO-STRONNIE SŁABE		V

c.d. tablicy 1

1	2	3	4	5
V		1:1 LEWO-STRONNIE SŁABE		Λ
Λ		1:1 SILNE		Λ

Symbole Λ i V oznaczają odpowiednio:
Λ — wszystkie elementy należące do danej klasy elementów,
V — niektóre elementy należące do danej klasy elementów
oraz N > 1, M > 1 odpowiadają liczbie elementów danej klasy biorących udział w powiązaniu

Nietrudno zauważyć, że przedstawione typy powiązań jednostek wyczerpują wszystkie możliwe kombinacje. Jeżeli jedna lub obie z powiązanych klas elementów są zbiorami pustymi, to powiązanie jednostek nie zachodzi. Dla ułatwienia późniejszego odwoływania się do przedstawionych typów powiązań jednostek wprowadziliśmy ich nazwy oraz oznaczenie stosunku liczb elementów biorących udział w jednym powiązaniu.

Jeżeli powiązanie obiektów dotyczy wszystkich elementów należących do obu klas obiektów, to określamy je jako silne powiązanie. Przykładem takiego powiązania jest powiązanie człowieka z krajem rodzinnym. W każdym kraju rodzą się ludzie i każdy człowiek urodził się w jakimś kraju. Znacznie częściej mamy do czynienia z powiązaniami, które obejmują pewne podzbiory elementów rozważanych klas obiektów. Wprowadzenie trzech różnych typów takich powiązań obiektów (lewo- i prawostronnie słabe i słabe) jest bardzo przydatne z punktu widzenia opisu semantyki modelu pojęciowego.

Rozpatrując np. powiązanie między osobami dorosłymi a dziećmi możemy przekazać dosyć istotną informację, że wszystkie dzieci mają (lub miały) rodziców, natomiast nie wszystkie osoby dorosłe mają dzieci. Stwierdzenie, że małżeństwo jest słabym powiązaniem 1 : 1 pozwala wykluczyć bigamię jako dopuszczalny stan rozpatrywanego modelu, nie wprowadzając jednocześnie obowiązku małżeństwa ani zakazu rozwodów.

Wracając do przedstawionego (por. rys. 17) pojęciowego modelu danych oficyny wydawniczej zauważmy, że powiązanie jednostek =*autorstwo*= jest powiązaniem N : M lewostronnie słabym. Możemy przyjąć, że w wypadku stosunku N : M powiązania lewostronnie i prawostronnie słabe są równoważne. Natomiast powiązanie jednostek =*umowa wydawnicza*= jest silnym powiązaniem 1 : N, gdzie z jednym wydaniem książki jest związanych wiele faktów autorstwa. Jak wynika z tego przykładu możliwości budowania opisu złożonych struktur są znacznie rozszerzone przez dopuszczenie opisu powiązań jednostek wiążących inne takie powiązania. Omawiając zagadnienie modelu semantycznego, zwróciliśmy uwagę na konieczność wprowadzenia, poza opisem struktury modelu, również wszystkich warunków, jakie muszą spełniać poszczególne zbiory jego elementów. Z dotychczas wprowadzonych form opisu pojęciowego modelu danych można opisać wiele istotnych warunków (zasad semantycznych); dotyczyły one jednak wyłącznie powiązań obiektów lub pewnych szczególnych własności obiektów, jak np. fakt identyfikacji osoby na podstawie jej nazwiska.

Ważnym problemem opisu pojęciowego modelu danych jest określenie własności elementów tworzących poszczególne klasy jednostek, powiązań jednostek oraz własności.

Na przykład rozważając klasę własności *data* nie interesujemy się przestrzenią wartości tej własności, lecz określonym jej podzbiorem. Przyjmijmy, że w wypadku omawianego modelu oficyny wydawniczej interesują nas daty zawarte między 01.01. 1850 r. i 31.12.1999 r., czyli okres 150 lat. Powiedzmy również, że jeżeli własność =data= spełnia rolę =data urodzenia= charakteryzując elementy należące do klasy jednostek =osoba=, to interesujący nas zakres wartości ogranicza się do okresu między 01.01.1850 r. a 31.12.1980 r. Rozpatrując natomiast klasę własności =wartość w złotych= możemy powiedzieć, że honorarium wynikające z jednej umowy wydawniczej nie może przekroczyć 100 tys. złotych, a cena jednego egzemplarza określonego wydania książki nie może być większa niż 1 tys. złotych. Z ograniczeń tych widać wyraźnie, że inny może być zakres zbioru wartości właściwości wynikający z faktu ich przynależności do określonej klasy, a inny z faktu występowania w określonej roli w stosunku

do charakteryzowanej klasy jednostek. Tak rozumianą rolę własności będziemy dalej nazywali *atrybutem obiektu*.

Podobnie rola obiektów w powiązaniu jednostek może być ograniczona przez wartości jej atrybutów. Na przykład możemy powiedzieć, że żaden autor nie może mieć mniej niż 20 lat. Z rozważań tych wynika, że uzyskanie pełnego opisu pojęciowego modelu danych wymaga uzupełnienia jego graficznej prezentacji odpowiednim komentarzem zawierającym wszystkie istotne ograniczenia.

Opis zbiorów wartości własności i atrybutów jednostek może być wyrażony w języku naturalnym lub w formalny sposób przy wykorzystaniu funkcji zdaniowych lub wyrażeń rachunku zdań. Forma opisu takich własności będzie wynikała z formy języka opisu pojęciowego modelu danych, a poszczególne wyrażenia tego języka powinny wynikać z omówionych już elementów opisu pojęciowego modelu danych.

Rozważmy dla przykładu możliwe do przyjęcia w języku opisu pojęciowego modelu danych zdanie definiujące powiązanie obiektów. Format ogólny oraz znaczenie poszczególnych faz zdania opisu powiązania obiektów przedstawiamy w tablicy 2.

W podobny sposób można rozwinąć pozostałe elementy opisu pojęciowego modelu danych, otrzymując tym samym pełny język formalny takiego opisu pojęciowego modelu danych oraz znacznie ułatwia jego wykorzystanie w dalszym procesie projektowania systemu informatycznego. Istotną cechą tego języka jest jego niezależność od przyjętych w późniejszych fazach realizacji systemu technik implementacyjnych. Ważnym problemem projektowania dolnych poziomów modelu danych, a szczególnie logicznej struktury danych, jest takie odwzorowanie pojęciowego modelu danych, by nie powstało niebezpieczeństwo straty informacji. Możliwości takiego opisu logicznej struktury danych wynikają bezpośrednio z klasy struktury danych (sieciowa, hierarchiczna lub relacyjna), jaką umożliwia język wykorzystywany do tworzenia takiego opisu. Zagadnienie to jest bardzo istotne z punktu widzenia wyboru odpowiedniego systemu zarządzania bazą danych [1].

[1] Na temat zasad opisu pojęciowego modelu danych por. M. Anderson i W. Staniszkis, *Systems Development Facility*, Long Range Desing Report, London 1979.

Tablica 2

Przykład zdania opisu powiązania jednostek

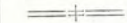
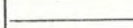
FORMAT OGÓLNY

POWIĄZANIE[a] JEDNOSTEK nazwa powiązania — jednostek

TWORZONE PRZEZ { [WSZYSTKIE] { JEDNOSTKA nazwa — jednostek / POWIĄZANIA JEDNOSTEK nazwa — powiązania jednostek } }

{ POWIĄZANE Z / [JEDNOZNACZNIE] OKREŚLAJĄCE } { [WSZYSTKIE] { JEDNOSTKI nazwa — jednostki / POWIĄZANIA JEDNOSTEK nazwa powiązania jednostek } } =

Fraza zdania	POWIĄZANIE JEDNOSTEK		Typ powiązania jednostek
	lewa strona	prawa strona	
1	2	3	4
POWIĄZANE Z			N : M SŁABE
WSZYSTKIE POWIĄZANE Z			N : M PRAWOSTRONNIE SŁABE
POWIĄZANE Z WSZYSTKIE			N : M LEWOSTRONNIE SŁABE
WSZYSTKIE POWIĄZANE Z WSZYSTKIE			N : M SILNE
OKREŚLAJĄCE			1 : N SŁABE
WSZYSTKIE OKREŚLAJĄCE			1 : N PRAWOSTRONNIE SŁABE

1	2	3	4
OKREŚLAJĄCE WSZYSTKIE			1 : N LEWOSTRONNIE SŁABE
WSZYSTKIE OKREŚLAJĄCE WSZYSTKIE			1 : N SILNE
JEDNOZNACZNIE OKREŚLAJĄCE			1 : 1 SŁABE
WSZYSTKIE JEDNOZNACZNIE OKREŚLAJĄCE			1 : 1 PRAWOSTRONNIE SŁABE
JEDNOZNACZNIE OKREŚLAJĄCE WSZYSTKIE			1 : 1 LEWOSTRONNIE SŁABE
WSZYSTKIE JEDNOZNACZNIE OKREŚLAJĄCE WSZYSTKIE			1 : 1 SILNE

Zasady notacji:

[*a*] — *a* opcjonalne

$\left\{ \begin{matrix} a \\ b \end{matrix} \right\}$ — albo *a* albo *b*

[a] Podkreślone słowa kluczowe są obowiązkowe

Kończąc omawianie problemów związanych z opisem pojęciowego modelu danych przedstawiamy nieco bardziej rozbudowany przykład opisu pojęciowego modelu danych domu towarowego (por. rys. 18). Przykład ten będzie służył dalej jako ilustracja możliwości różnych typów języków opisu danych oraz języków dostępu do danych. Model domu towarowego obejmuje następujące klasy jednostek=

PIĘTRO = NUMER-PIĘTRA, POWIERZCHNIA
DZIAŁ = NUMER-DZIAŁU, TYP-ASORTYMENTU
PRACOWNIK = IMIĘ, NAZWISKO, FUNKCJA, POBORY
DOSTAWCA = NAZWA-DOSTAWCY, WARTOŚĆ-DOSTAW
TOWAR = NAZWA-TOWARU, KOD, CENA

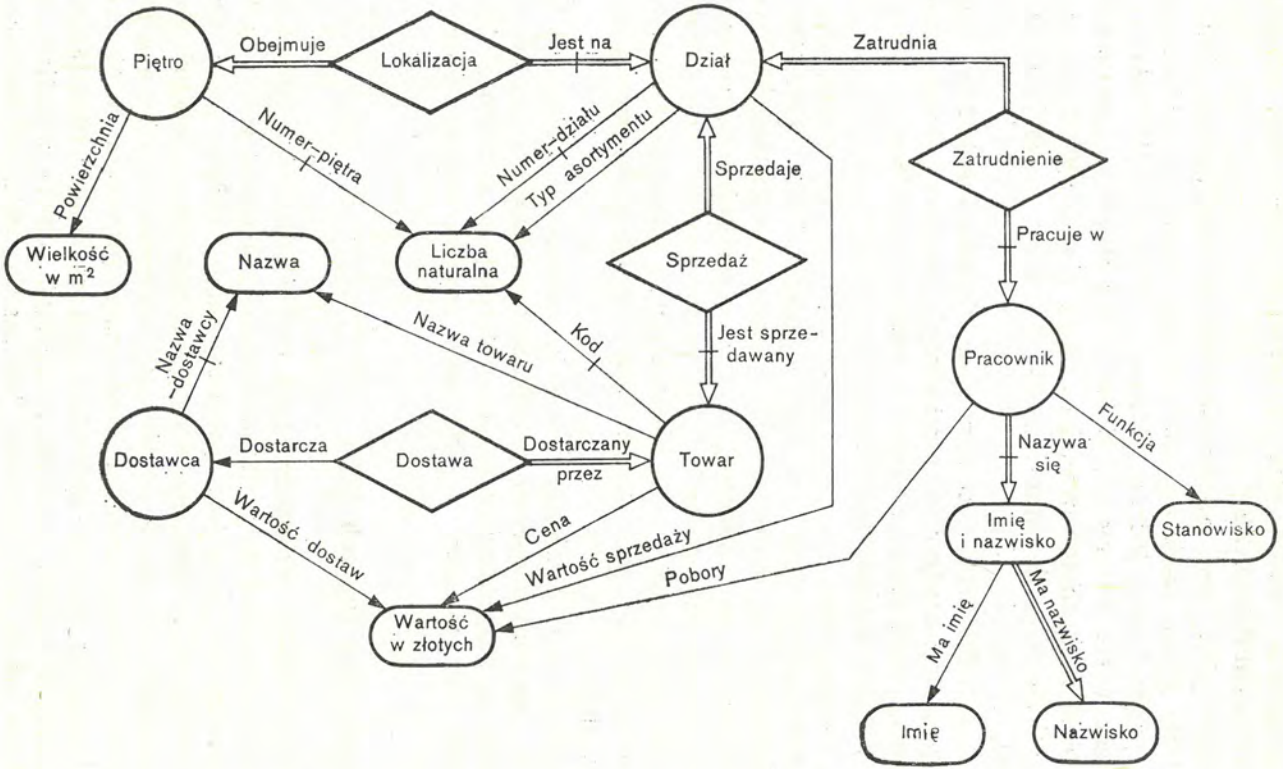
Podkreślone atrybuty obiektów pozwalają na jednoznaczną selekcję określonego wystąpienia obiektów spośród wszystkich innych obiektów należących do tej samej klasy, są więc identyfikatorem w ramach danej klasy obiektów. Odpowiednie związki między klasą obiektów a klasą własności mają charakter funkcji 1-1 lub funkcji 1-1 i *na*. W tym drugim wypadku zbiór wartości własności obejmuje wyłącznie wartości identyfikatora danej klasy jednostek (np. =imię i nazwisko=). Jeśli identyfikator obejmuje kilka atrybutów elementarnych, to możemy go przedstawić jako atrybut zbiorowy (=nazywa się=) określając jednocześnie jego związki z atrybutami elementarnymi (*ma imię, ma nazwisko*). Jest to nieznaczne odstępstwo od wprowadzonej zasady, że wartość własności musi być semantycznie niepodzielna.

W ramach rozpatrywanego pojęciowego modelu danych wyróżniliśmy wiele powiązań obiektów, a z charakteru typu powiązań wynikają zasady semantyczne przyjęte w modelu.

Oto przykłady powiązań obiektów:

— LOKALIZACJA jest silnym powiązaniem obiektów typu 1 : N, ponieważ na każdym piętrze można umieścić wiele działów, a każdy dział musi w całości znajdować się na jednym piętrze.

— SPRZEDAŻ jest silnym powiązaniem obiektów typu 1 : N, ponieważ każdy towar może być sprzedawany tylko w jednym dziale, a każdy dział musi sprzedawać przynajmniej jeden towar.



Rys. 18. Graficzna reprezentacja modelu pojęciowego domu towarowego

To powiązanie jest ponadto charakterystyczne przez atrybut WARTOŚĆ-SPRZEDAŻY.

— DOSTAWA jest lewostronnie słabym powiązaniem jednostek typu N : M, ponieważ każdy towar musi być dostarczony przez przynajmniej jednego dostawcę, a może być dostarczany przez wielu dostawców. Natomiast dostawca może dostarczać wiele różnych towarów, lecz mogą również wystąpić dostawcy, którzy nie dostarczają żadnego towaru (potencjalni dostawcy).

— ZATRUDNIENIE jest silnym powiązaniem obiektów typu 1 : N, ponieważ każdy dział musi zatrudniać pracowników, natomiast pracownik musi pracować w jakimś dziale, lecz nie może pracować w wielu działach jednocześnie.

Dla tak opisanego pojęciowego modelu danych można przyjąć wiele różnych ograniczeń wyznaczających zbiory dopuszczalnych wartości atrybutów jednostek lub własności. Możemy np. powiedzieć, że cena każdego towaru, z wyjątkiem sprzedawanych w dziale, którego asortymentem są aparaty fotograficzne, nie może przekroczyć 50 tys. złotych. Innym ograniczeniem może być maksymalna wysokość zarobku pracownika wynosząca np. 30 tys. złotych.

Przedstawiona konwencja tworzenia opisu pojęciowego modelu danych jest jednym z wielu możliwych rozwiązań tego problemu. Jest ona — naszym zdaniem — wystarczająco mocnym i elastycznym instrumentem wspomagania procesu analizy danych.

3. Skorowidz Danych (SD)

Rozwój systemów informatycznych doprowadził w ciągu ostatniego dziesięciolecia do powstania wielu problemów związanych z utrzymaniem i rozwojem oprogramowania i związanych z nim zbiorów danych. Obecne systemy informatyczne, obejmujące wiele tysięcy programów użytkowych i wieleset zbiorów danych są rzeczą całkiem normalną. Utrzymanie sprawności eksploatacyjnej oraz umożliwienie rozwoju takich systemów jest trudnym problemem z punktu widzenia zarządzania. Jedną z możliwych dróg rozwiązania tego problemu jest zastosowanie systemu Sko-

rowidza Danych (*Data Dictionary System*). Z dotychczasowych doświadczeń wynika, że zastosowanie systemu skorowidza danych ma równie duże znaczenie dla systemów informatycznych, budowanych na podstawie konwencjonalnej technologii, jak i dla systemów opartych na wspólnej bazie danych.

Skorowidz Danych jest zbiorem informacji o definicjach, strukturze i sposobie wykorzystania danych. Z definicji tej wynika, że skorowidz danych może być prowadzony bez wykorzystania komputera; ważne jest jednak by zawierał wystarczająco dużo informacji o zasobach danych oraz ich wykorzystaniu przez odpowiednie grupy użytkowników systemów informatycznych, działających w danej organizacji. Ze względu na dużą standaryzację funkcji systemów skorowidza danych jest rzeczą naturalną, że podobnie jak w wypadku innych problemów z dziedziny zarządzania podjęto realizację pakietów oprogramowania wspomagających zarządzanie zasobami danych i oprogramowania użytkowego. Zastosowanie takiego pakietu może przynieść równie duże efekty zarówno w konwencjonalnych systemach informatycznych, jak i w systemach opartych na wspólnej bazie danych.

Do podstawowych funkcji systemu skorowidza danych należą:

— ewidencja danych elementarnych obejmująca ich charakterystykę, nazwy i znaczenie,

— utrzymanie powiązań między synonimami,

— ewidencja zasad spójności logicznej wartości danych elementarnych,

— ewidencja wykorzystania danych elementarnych w strukturze danych obejmująca:

  — struktury rekordów,

  — struktury zbiorów danych,

  — struktury dokumentów wejściowych,

  — struktury raportów wynikowych,

— ewidencja wykorzystania danych elementarnych, rekordów i zbiorów danych przez moduły, programy i systemy użytkowe,

— wspomaganie bezpośredniego dostępu do skorowidza danych,

— ochrona tajności informacji o zasobach danych,

— automatyczna ewidencja na podstawie analizy deklaracji danych i deklaracji zbiorów danych zawartych w istniejącym oprogramowaniu użytkowym,

Tablica 3

Wybrane systemy Skorowidza Danych

Producent	System Skorowidza Danych	Współpracuje z systemem ZBD	Wymaga systemu ZBD
MRI	CONTROL 2000	System 2000	System 2000
Synergetics	Data Catalogue	IMS TOTAL	nie
Cincon	Data Dictionary	TOTAL	TOTAL
MSP, Inc	Datamanager	IMS TOTAL IDMS ADABAS	nie
IBM	DB/DC Dictionary	IMS DOS/DL/I	IMS lub DOS DL/I
Cullinane	IDMS Dictionary	IDMS	IDMS
Arthur Anderson	Lexicon	IMS TOTAL IDMS	nie
University Computing	UCC IO	IMS	IMS

Źródło: G. Schüssel, *The Roll of The Data Dictionary*, „Datamation" 1979, nr 6,

— wspomaganie konstrukcji programów użytkowych przez generowanie deklaracji danych i zbiorów danych (zwykle dla takich języków programowania jak COBOL, PL/1 i FORTRAN).

Systemy skorowidza danych mogą zwykle współpracować z wybranymi systemami zarządzania bazą danych, a w niektórych wypadkach wręcz wymagają wykorzystania takiego systemu. Charakterystykę najbardziej znanych systemów skorowidza danych zawiera tablica 3.

Ochrona tajności	Dostęp bezpośredni	Automatyczna ewidencja danych	Generacja deklaracji danych
tak	TP 2000	tak	COBOL, PL/I, FORTRAN, ASSEMBLER
tak	TSO CICS	tak	COBOL, PL/I, ASSEMBLER
tak	nie	nie	nie
tak	Intercomm Taskmaster ROSCOE IMS/DC CICS TSO CMS ETSS	tak	COBOL, PL/I, ASSEMBLER, MARK IV
nie	IMS/DC(VS)	tak	COBOL, PL/I
nie	TSO IDMS Query	tak	COBOL, PL/I
nie	TSO	nie	COBOL, PL/I
nie	IMS/DC	nie	COBOL

vol. 23.

Rozwój systemów zarządzania bazy danych doprowadził do zmiany wymagań w stosunku do systemów skorowidza danych. Mniejsze znaczenie mają obecnie możliwości automatycznego generowania opisów struktury danych w programach użytkowych (ta funkcja jest zwykle realizowana przez system ZBD).

Największe znaczenie mają te elementy systemu skorowidza danych, które wspomagają proces projektowania, tworzenia i eksploatacji bazy danych i realizowanych na ich podstawie zasto-

sowań. Mimo iż obecnie osiągalne systemy skorowidza danych są bardzo użyteczne w zastosowaniach opartych na wspólnej bazie danych, ich przydatność ogranicza się do wspomagania działalności Administratora Bazy Danych. Brak jest natomiast funkcji wpływających na automatyzację prac nad projektem i oprogramowaniem systemu informatycznego.

Do najbardziej poważnych wad dostępnych obecnie systemów skorowidza danych zalicza się następujące:

— brak możliwości formalnego opisu semantyki systemu informacyjnego,

— brak w istniejących językach SD wyraźnego podziału na poziomy systemu informatycznego (pojęciowy, logiczny, fizyczny,

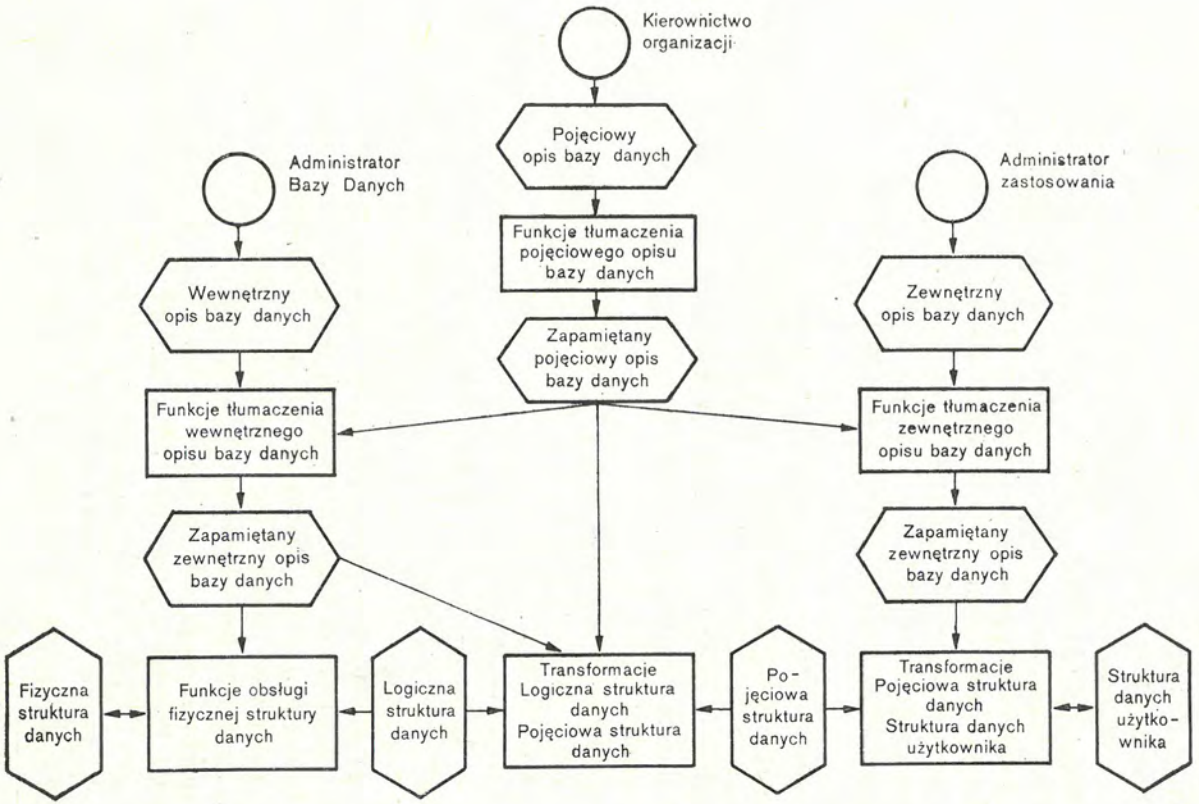
— stosowane mechanizmy opisu danych oraz stopień ich interpretacji z systemem ZBD nie prowadzą do rozwiązania problemu niezależności od danych,

— separacja opisu pojęciowego (Skorowidz Danych) od opisu logicznej struktury danych (schemat bazy danych) może doprowadzić do naruszenia spójności logicznej tych opisów, a w konsekwencji do naruszenia spójności logicznej bazy danych,

— brak odpowiednich narzędzi wspomagających projektowanie odwzorowania modelu pojęciowego danych w logicznej i fizycznej strukturze danych.

Rozwiązanie tych problemów może nastąpić przez rozwój architektury systemów skorowidza danych, ich pełną integrację ze współczesnymi systemami ZBD lub przez rozwój architektury systemów ZBD. Przykładem tego drugiego podejścia jest propozycja architektury systemu ZBD opracowana przez Grupę Roboczą ds. Baz Danych ANSI/SPARC. Ogólny schemat proponowanej architektury systemu ZBD przedstawia rysunek 19.

Centralnym elementem architektury ANSI/SPARC jest pojęciowy model bazy danych pozwalający na tworzenie formalnego opisu semantyki systemu informacyjnego. Ponieważ podstawowym użytkownikiem języka opisu pojęciowego jest kierownictwo organizacji, istotnym warunkiem stawianym przed takim opisem jest łatwość jego formułowania i czytelność. Jednocześnie język powinien być formalny, tzn. powinien składać się ze zbioru dopuszczalnych wyrażeń o ściśle określonej semantyce,



Rys. 19. Struktura systemu ZBD według propozycji ANSI/SPARC

przy czym każde z tych wyrażeń musi być rozpoznawalne jednocześnie na podstawie jego składni.

Drugim istotnym elementem propozycji ANSI/SPARC jest pośrednia zależność struktury danych użytkownika (zewnętrzny opis bazy danych) od logicznej i fizycznej struktury danych (wewnętrzny opis bazy danych). Takie rozwiązanie prowadzi do wzmocnienia niezależności od danych, pod warunkiem odpowiednio elastycznych odwzorowań danych. W obecnej sytuacji w dziedzinie oprogramowania systemów ZBD bardziej realna wydaje się być ewolucja funkcji systemów skorowidza danych w kierunku proponowanego przez ANSI/SPARC modelu pojęciowego, niż tworzenie systemu ZBD, zgodnego z propozycją tej organizacji.

Bardzo istotne z punktu widzenia rozwoju funkcji systemów SD były prace prowadzone przez działający w ramach *British Computer Society* Grupę Roboczą ds. Skorowidz Danych[2]. W raporcie będącym wynikiem prac tej grupy, sformułowano następujące podstawowe postulaty w stosunku do systemów SD. Są to m.in.:

— wspomaganie tworzenia dokumentacji projektowej we wszystkich stadiach realizacji systemu informatycznego (analiza, projekt, wdrożenie),

— wspomaganie aktualizacji i rewizji dokumentacji projektowej — brak takiej możliwości praktycznie przekreśla celowość wykorzystywania systemu SD dla tworzenia takiej informacji,

— wspomaganie bieżącego wykorzystania informacji projektowo-programowych przez elastyczny aparat selekcji,

— wspomaganie strukturalnego projektowania przez utrzymanie odpowiednio silnej dyscypliny dokumentacyjnej,

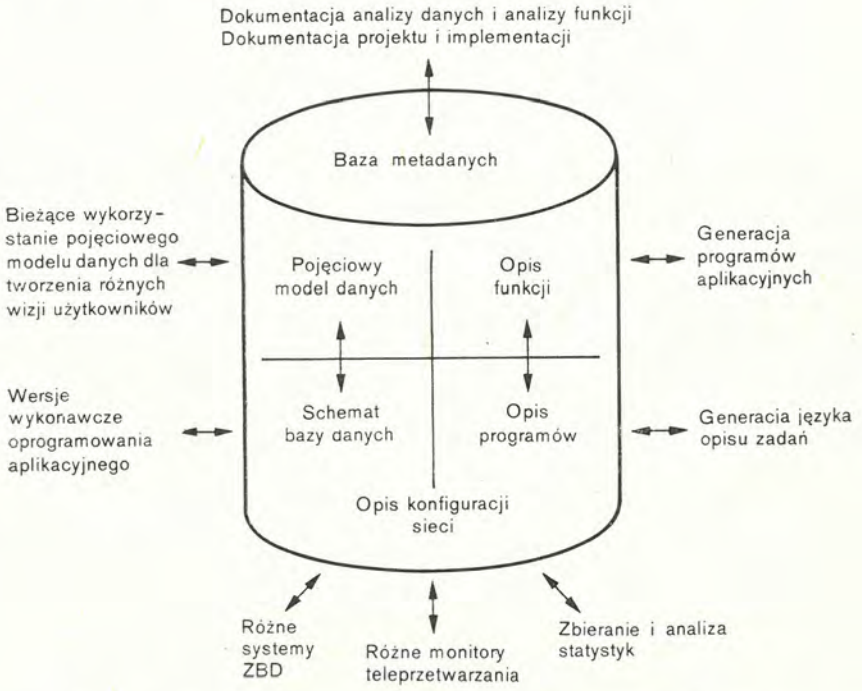
— możliwość elastycznego rozwoju systemu skorowidza danych zgodnie z powstającymi wymaganiami.

Z postulatów przedstawionych przez Grupę Roboczą ds. Skorowidza Danych wynika, że zakres funkcji tych systemów ulega istotnym zmianom i że stają się one coraz częściej narzędziem projektowania i wdrażania systemów informatycznych. W tej sytuacji Skorowidz Danych stanie się zbiorem informacji projek-

[2] Por. *Data Dictionary Systems, Working Party Report*, The British Computer Society, 1977.

towych (metainformacji), dostępnych nie tylko dla użytkowników systemu, lecz również dla innych elementów oprogramowania działających w ramach systemu informatycznego (system ZBD, monitor teleprzetwarzania, system operacyjny).

Na rysunku 20 przedstawiamy objęty takim Skorowidzem Danych (metabaza danych) zbiór informacji projektowych, zbiór metainformacji powinien być jądrem systemu informatycznego integrującym wszystkie jego funkcje.



Rys. 20. Zakres informacji zawartych w Skorowidzu Danych

Zakres informacji zawarty w Skorowidzu Danych obejmuje m.in. opis semantycznego modelu danych. Ponieważ systemy Skorowidza Danych służą do wspomagania pełnego cyklu prac projektowo-programowych, zakres informacji musi być stopniowo rozszerzony w miarę postępu prac nad tworzeniem systemu informatycznego. Formalizacja opisu pojęciowego modelu danych

umożliwia automatyczne generowanie elementów opisu logicznej struktury danych (Schemat Bazy Danych). Możliwość ta jest bardzo istotna dla utrzymania spójności logicznej obu tych poziomów opisu struktury danych oraz dla zapobiegania stracie informacji w toku przejścia z jednego modelu danych na drugi. Zasady takiego odwzorowania dla poszczególnych klas logicznej struktury danych (sieciowa, hierarchiczna, relacyjna) omówimy dalej.

IV. Logiczny model danych

1. Podstawowe elementy logicznych modeli danych

Omawiając pojęciowy model danych wprowadziliśmy wiele elementów jego opisu wskazując jednocześnie na rolę tych elementów w budowaniu struktury, będącej reprezentacją rzeczywistości objętej projektowanym systemem informatycznym. Istotną cechą takiego opisu jest możliwość jednoznacznego przedstawienia semantyki modelu informacyjnego. Niezależnie od formy opisu, zawsze istnieje problem przeniesienia zawartych w nim informacji na poziom logicznego projektu systemu informatycznego. W wypadku systemów tradycyjnych, taki projekt logiczny zawiera opis plików danych oraz ogólny opis procedur zakładania, aktualizacji i wyszukiwania danych. Istotną cechą takiego projektu jest fakt, iż cały opis struktury powiązań między plikami danych jest zawarty w opisie procedur. Na przykład określone w modelu oficyny wydawniczej powiązanie jednostek *autorstwo* zostanie opisane jako procedura dobierania elementów plików danych, obejmujących *osoby* i *książki*. W wypadku systemów wykorzystujących złożone struktury danych umieszczenie tak istotnych informacji, jak opis powiązań elementów tej struktury w opisie procedur, w poważnym stopniu utrudnia projektowanie, realizację i testowanie oprogramowania systemu. Jednocześnie takie podejście do projektowania plików danych prowadzi do konieczności powtarzania tych samych wartości danych (redundancja) w wielu różnych plikach tak, aby istniała możliwość odpowiedniego ich łączenia.

W omawianym przykładzie zapis dotyczący każdej osoby, która jest autorem, będzie musiał zawierać wszystkie tytuły książek

napisanych przez te osoby, a zapis dotyczący książek będzie musiał zawierać imiona i nazwiska jej autorów.

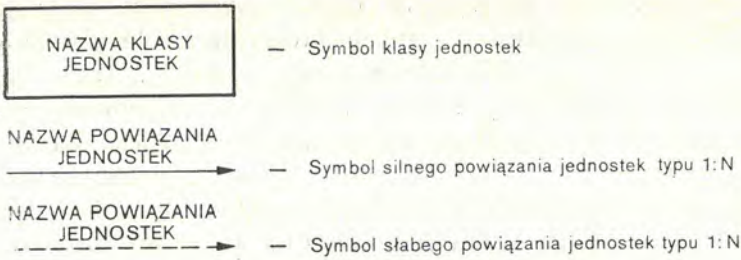
Reprezentacja złożonych struktur danych w postaci wielu plików danych prowadzi również do poważnych trudności w utrzymaniu logicznej spójności danych. Rosnące znaczenie oraz wielkość i stopień złożoności struktur zasobów danych współczesnych systemów informatycznych spowodowały powstanie specjalistycznego oprogramowania, którego podstawową funkcją jest zarządzanie tymi zasobami.

Takim oprogramowaniem są współcześnie eksploatowane systemy zarządzania bazą danych. Wspólną cechą wszystkich tych systemów jest możliwość opisu powiązań poszczególnych elementów danych w sposób niezależny od stosowanych technik ich zapisu na nośnikach urządzeń pamięci zewnętrznej. Taki opis struktury danych nazywamy *logicznym modelem danych*. Podstawowym celem opisu logicznego modelu danych jest zebranie wszystkich istotnych informacji o strukturze i treści fragmentu rzeczywistości objętego projektowaną bazą danych.

Ważną więc cechą dostępnego w ramach określonego systemu ZBD logicznego modelu danych jest możliwość odwzorowania pojęciowego modelu danych bez straty informacji. Jeżeli logiczny model danych nie daje takich możliwości, to informacje te muszą być zawarte w oprogramowaniu użytkowym systemu informatycznego. W ramach współczesnych systemów ZBD dostępne są trzy różne logiczne modele danych, a mianowicie: sieciowy, hierarchiczny i relacyjny. Każdy z tych modeli danych charakteryzuje się odmiennym podejściem do opisu struktur danych, zawiera różne ograniczenia w zakresie budowania tych struktur oraz zasady dostępu do danych.

Omawiając poszczególne modele danych zwrócimy więc uwagę na problem ich strukturalnej zupełności, to jest możliwości reprezentacji informacji strukturalnej i zasad semantycznych zawartych w pojęciowym modelu danych. Podobnie, jak w wypadku opisu pojęciowego modelu danych, posłużymy się konwencją graficznej reprezentacji struktury danych. Najbardziej rozpowszechnioną konwencją graficzną jest tzw. notacja Bachmana [1], której elementy przedstawiamy na rysunku 21.

[1] Por. C. W. Bachman, *Data Structure Diagrams*, „Data Base" 1969, nr 2, vol. 1.



Rys. 21. Symbole graficzne notacji Bachmana

Jak wynika z omawianego już (por. rozdz. III, pkt. 2) opisu pojęciowego modelu danych, powiązanie jednostek typu 1 : N, wiąże pewną grupę jednostek należących do jednej klasy jednostek z jednostką należącą do innej klasy jednostek. Strzałka określa kierunek podporządkowania, przy czym jednostkę nadrzędną nazywamy właścicielem, a jednostki podrzędne członkami powiązania jednostek.

Już z samej notacji Bachmana wynikają ograniczenia w stosunku do możliwości wprowadzonych w przedstawionej konwencji reprezentacji pojęciowego modelu danych. Na przykład powiązanie jednostek typu 1 : 1 jest tutaj szczególnym wypadkiem powiązania jednostek typu 1 : N (gdzie N = 1). Ponieważ jednak przedstawiona notacja zawiera wszystkie możliwości omawianych logicznych modeli danych, nie wprowadzimy dodatkowych symboli graficznych, a ograniczenia w stosunku do opisu pojęciowego modelu danych omówimy osobno dla każdej z prezentowanych klas struktur danych.

2. Sieciowy model danych

Sieciowy model danych jest strukturą, w której każda jednostka, należąca do określonej klasy jednostek, może jednocześnie uczestniczyć w wielu powiązaniach jednostek należących do tej samej lub różnych klas powiązań jednostek, występując w roli nadrzędnej (właściciel) lub roli podrzędnej (członek).

Ta ogólna definicja sieciowego modelu danych nie nakłada żadnych ograniczeń na konstruowanie powiązań między jednostka-

mi. Stosowane w istniejących obecnie systemach ZBD języki opisu sieciowej struktury danych zawierają wiele dodatkowych ograniczeń na zasady konstrukcji tej struktury. Takie ograniczenia są zwykle podyktowane względami implementacyjnymi, a szczególnie koniecznością zachowania jednoznaczności w tworzeniu, utrzymaniu i wyszukiwaniu elementów struktury danych. Klasycznym przykładem sieciowego modelu danych jest struktura danych proponowana w Raporcie Grupy Roboczej ds. Baz Danych Komitetu CODASYL, opublikowanym w kwietniu 1971 r.[2], a następnie rozwinięta przez Komitet ds. Języka Opisu Danych, działającego również w ramach Komitetu CODASYL. Propozycje Komitetu ds. JOD zawarte są w raportach opublikowanych w 1973 r. i 1978 r.[3]. Wszystkie te propozycje były publikowane w formie języków opisu danych dla sieciowego modelu danych. Raporty Komitetu CODASYL doczekały się wielu implementacji, a obecnie wszystkie najbardziej rozpowszechnione typy komputerów są wyposażone w systemy ZBD zrealizowane na podstawie tych propozycji.

W Polsce najbardziej rozpowszechnionym systemem ZBD, eksploatowanym na komputerach Jednolitego Systemu, jest system RODAN[4] będący implementacją wymienionych raportów Komitetu CODASYL.

W proponowanym Języku Opisu Danych przyjęto następujące definicje elementów struktury danych:

— *dana elementarna* — najmniejsza i niepodzielna struktura danych, składająca się z nazwy i atrybutów opisu,

— *grupa powtarzalna* — zbiór danych elementarnych, dla którego określono nazwę i atrybuty opisu; grupa powtarzalna może być również zbiorem grup powtarzalnych,

— *rekord* — zbiór danych elementarnych i grup powtarzalnych dla którego określono nazwę i atrybuty opisu,

— *zbiór strukturalny* — powiązanie podporządkowujące grupę

[2] Por. *Codasyl Systems Committee, Feature Analysis of GDBMS*, ACM, New York 1971; *Codasyl Data Base Task Group*, DBTG, April 1971 Report, ACM, New York 1972.

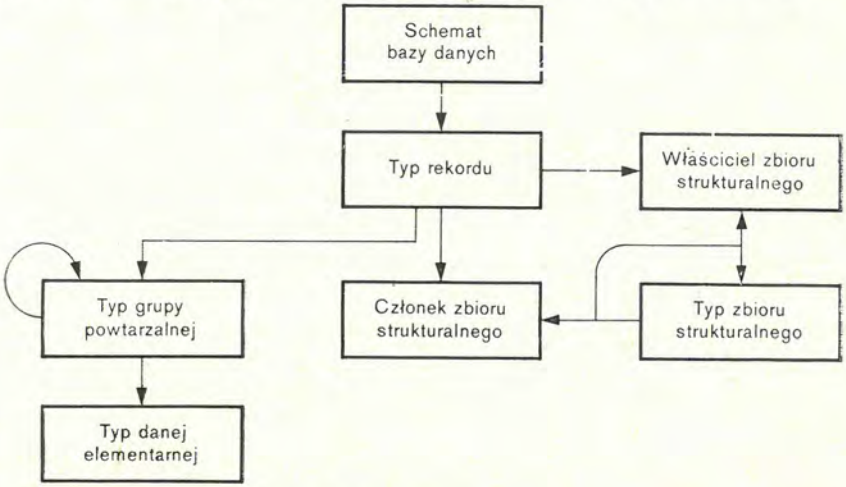
[3] Por. *Codasyl Data Description Language Committee*, „Data Description Language „Journal of Development US Dept. of Commerce NBS" 1973; *Codasyl DDLC, Data Description Language*, „Journal of Development Dept. of Supply and Services", Ottawa 1978.

[4] Por. Techniczna dokumentacja CPiZI, *System ZBD RODAN*, Dokumentacja użytkowa, Warszawa 1980.

rekordów jednego typu jednemu rekordowi innego typu, przy czym rekordy podporządkowane nazywamy członkami zbioru strukturalnego, a rekord nadrzędny właścicielem zbioru strukturalnego.

Dla zbioru strukturalnego również jest określona nazwa i atrybuty opisu.

Atrybutami opisu elementu struktury danych są określone (poza nazwą) w opisie typu tego elementu jego cechy charakterystyczne, wspólne dla wszystkich elementów, należących do tego samego typu elementów. Przykładem atrybutów opisu może być opis danej elementarnej, zawierający takie informacje, jak długość danej, typ wartości (ciąg znaków, liczba dwójkowa itp.) lub ograniczenia przyjęte dla zbioru jej dopuszczalnych wartości.



[a] Ten sam typ rekordu nie może być właścicielem i członkiem tego samego typu zbioru strukturalnego.

Rys. 22. Zasady konstrukcji struktury danych według raportów Komitetu CODASYL

Istotnymi pojęciami określającymi związek opisu bazy danych z bazą danych są:

— *schemat bazy danych* — zawierający nazwy i atrybuty opisu wszystkich elementów struktury danych,

— *baza danych* — składająca się ze wszystkich danych, rekordów i zbiorów strukturalnych opisanych w jej schemacie.

Zasady konstrukcji sieciowego modelu danych opartego na propozycjach Komitetu CODASYL przedstawiono za pomocą notacji Bachmana na rysunku 22.

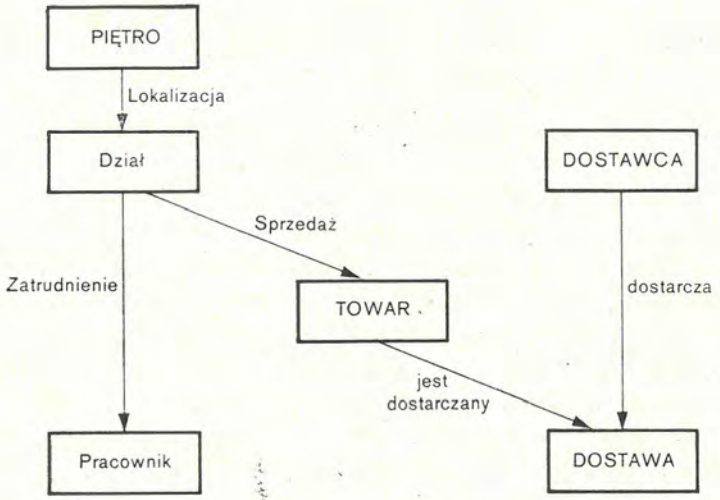
Schemat Bazy Danych może zawierać opis dowolnej liczby typów rekordów, które z kolei mogą zawierać dowolną liczbę typów grup powtarzalnych. Każdy typ grupy powtarzalnej może obejmować dowolną liczbę innych typów grup powtarzalnych oraz danych elementarnych. Taka możliwość pozwala na budowanie tablic wielowymiarowych. Każdy typ rekordów może występować w roli właściciela lub członka dowolnej liczby typów zbioru strukturalnego, przy czym każdy typ zbioru strukturalnego może obejmować dowolną liczbę typów rekordów członkowskich.

Dopiero w raporcie Komitetu ds. JOD z maja 1978 r. zniesiono zaznaczone na schemacie ograniczenie, mówiące o tym, że żaden typ rekordu nie może być jednocześnie właścicielem i członkiem danego typu zbioru strukturalnego. Oczywiście, zgodnie z zasadami notacji Bachmana, żadne wystąpienie danego typu rekordu nie może partycypować w roli właściciela lub członka w więcej niż jednym wystąpieniu zbioru strukturalnego.

Należy zwrócić uwagę na fakt, że w ogólnej definicji sieciowego modelu danych, przytoczonej na początku, wymienione ograniczenia nie występowały. Zostały one wprowadzone ze względu na konieczność zachowania jednoznaczności niezbędnej dla prawidłowego wykonywania komend Języka Manipulacji Danymi (JMD).

Omawiając ogólne problemy logicznych modeli danych zwróciliśmy uwagę na fakt, że najbardziej istotnym zagadnieniem projektowania bazy danych na poziomie logicznym jest przeniesienie opisu semantyki zawartego w pojęciowym modelu danych. Przedstawione już reguły budowy sieciowej struktury danych oparte na propozycjach Komitetu CODASYL wskazują na istnienie wielu ograniczeń w porównaniu z możliwościami przedstawionego w rozdziale III opisu pojęciowego modelu danych. Zanim przejdziemy do szczegółowego omówienia możliwości odwzorowania tego opisu w sieciowym modelu danych, przedstawimy przykład sieciowej struktury danych domu towarowego, odpowiadającej graficznej reprezentacji pojęciowego modelu danych,

przedstawionego na rysunku 18. Schemat tej struktury przedstawionej w notacji Bachmana i zgodnej z zasadami budowy sieciowego modelu danych Komitetu CODASYL przedstawiamy na rysunku 23.



Rys. 23. Schemat struktury sieciowej danych domu towarowego

Łatwo zauważyć, że schemat w notacji Bachmana zawiera umownie mniej informacji na temat domu towarowego niż odpowiadająca mu graficzna reprezentacja pojęciowego modelu danych (por. rys. 18). Brak jest przede wszystkim artybutów (danych elementarnych) przedstawionych klas jednostek (typów rekordów). Nie ma również możliwości przedstawienia zbiorów wartości, na których są określone takie atrybuty (liczby naturalne, wartość w złotych itp.). Zwykle taki schemat w notacji Bachmana jest uzupełniany specyfikacją zbiorów atrybutów (danych elementarnych), charakteryzujących poszczególne klasy jednostek (typów rekordów).

W wypadku omawianego sieciowego modelu danych domu towarowego poszczególne typy rekordów mają następującą strukturę:

PIĘTRO = NUMER-PIĘTRA, POWIERZCHNIA
DZIAŁ = NUMER-DZIAŁU, TYP-ASORTYMENTU
WARTOŚĆ SPRZEDAŻY
PRACOWNIK = IMIĘ, NAZWISKO, FUNKCJA, POBORY
DOSTAWCA = NAZWA-DOSTAWCY, WARTOŚĆ-DOSTAW
TOWAR = NAZWA-TOWARU, KOD, CENA
DOSTAWA = TOWAR, ILOŚĆ, DOSTAWCA, NUMER-
-DOSTAWY

Podkreślone dane elementarne jednoznacznie identyfikują wystąpienia rekordów w ramach opisywanego typu rekordu. Wszystkie klasy jednostek przedstawione w opisie pojęciowego modelu danych są opisane jako typy rekordów w modelu sieciowym. Natomiast powiązania jednostek typu 1 : N są opisane jako zbiory strukturalne. W opisie pojęciowego modelu danych powiązanie jednostek „sprzedaż" ma atrybut „wartość sprzedaży". Ponieważ jednak zbiór strukturalny nie może zawierać danych elementarnych oraz ze względu na to, że dana elementarna „wartość-sprzedaży" dotyczy całego działu, została ona zapisana w ramach typu rekordu („dział"), który jest właścicielem tego zbioru strukturalnego.

Sieciowy model danych Komitetu CODASYL, jak również notacja Bachmana nie dają możliwości bezpośredniego odwzorowania powiązania jednostek typu N : M. W takim wypadku zachodzi konieczność wprowadzenia dodatkowego typu rekordu, który reprezentuje powiązanie jednostek, uczestnicząc w dwóch zbiorach strukturalnych jako rekord członkowski. W omawianym przykładzie taką rolę pełni typ rekordu „dostawa", który reprezentuje powiązanie typu N : M między rekordami „towar" i „dostawca".

Omawiając symbole graficzne notacji Bachmana wyróżniliśmy dwa rodzaje powiązań typu 1 : N, a mianowicie silne i słabe powiązanie jednostek. Podobne rozwiązania istnieją w modelu danych Komitetu CODASYL, gdzie wprowadzono pojęcia silnego i słabego zbioru strukturalnego oraz wyróżniono dodatkowo przypadek pustego zbioru strukturalnego. W wypadku silnego zbioru strukturalnego wszystkie wystąpienia rekordu (lub rekordów) zdefiniowanego jako rekord członkowski tego zbioru struktural-

nego muszą uczestniczyć w jakimś jego wystąpieniu. Natomiast w słabym zbiorze strukturalnym istnieje możliwość opcjonalnego członkostwa, tzn. sytuacji, w której nie wszystkie wystąpienia rekordu członkowskiego muszą uczestniczyć w jakimś wystąpieniu tego zbioru strukturalnego. Z pustym zbiorem strukturalnym mamy do czynienia wtedy, gdy z danym wystąpieniem rekordu, opisanego jako rekord właściciel tego zbioru strukturalnego, nie jest powiązane, w ramach tego typu zbioru, żadne wystąpienie rekordu członkowskiego. Silny zbiór strukturalny w powiązaniu z pojęciem pustego zbioru strukturalnego odpowiada wprowadzonemu w pojęciowym modelu danych lewostronnie słabemu powiązaniu jednostek typu 1 : N.

Ponieważ nie ma możliwości wyeliminowania pustych wystąpień zbioru strukturalnego przez odpowiedni element opisu sieciowego modelu danych, odwzorowanie silnego powiązania jednostek typu 1 : N nie daje się opisać w tym modelu danych. Pojęcie słabego zbioru strukturalnego odpowiada natomiast słabemu powiązaniu jednostek typu 1 : N. Ograniczenia te odnoszą się również do powiązań jednostek typu N : M i 1 : 1. W razie powiązań typu 1 : 1 są one traktowane jako specjalny wypadek zbioru strukturalnego (tylko jedno wystąpienie rekordu członkowskiego), przy czym nie ma możliwości zaznaczania tego faktu w opisie zbioru strukturalnego. Trudności w pełnym (bez straty informacji) odwzorowaniu pojęciowego modelu danych w modelu logicznym są wyraźnie widoczne na omawianym przykładzie bazy danych dla domu towarowego, gdzie w opisie pojęciowego modelu danych stwierdzono, iż towar musi być zawsze dostarczany przez jakiegoś dostawcę, natomiast dostawca może nie dostarczać żadnego towaru. Ta zasada semantyczna była reprezentowana w modelu danych przez lewostronnie słabe powiązanie typu N : M „dostawa" (por. rys. 19). W wypadku sieciowego modelu danych mamy do czynienia z dwoma zbiorami strukturalnymi „jest dostarczany" i „dostarcza" i nie istnieje możliwość zapisania, że ten pierwszy nie może mieć nigdy pustych wystąpień.

Wprowadzenie takiego ograniczenia nie jest możliwe przy istniejących zasadach manipulacji danymi w sieciowej strukturze danych. Na przykład nie mamy obecnie możliwości jednoczesnego (w ramach jednej komendy Języka Manipulacji Dany-

mi) zapisania rekordu-właściciela i co najmniej jednego wystąpienia rekordu członkowskiego. W tablicy 4 zebrano zasady odwzorowania elementów pojęciowego modelu danych w sieciowym modelu danych Komitetu CODASYL.

Tablica 4

Zasady odwzorowania elementów pojęciowego modelu danych w sieciowym modelu danych Komitetu CODASYL

Pojęciowy model danych	Sieciowy model danych
Jednostka	rekord
Silne powiązanie jednostek N : M	—
Lewostronne słabe powiązanie jednostek N : M	—
Prawostronnie słabe powiązanie jednostek N : M	—
Słabe powiązanie jednostek N : M	rekord relacji opisanej jako rekord członkowski w dwóch silnych zbiorach strukturalnych
Silne powiązanie jednostek 1 : N	grupa powtarzalna reprezentująca podporządkowaną jednostkę
Lewostronnie słabe powiązanie jednostek 1 : N	silny zbiór strukturalny, grupa powtarzalna o zmiennej ilości powtórzeń
Prawostronnie słabe powiązanie jednostek 1 : N	—
Słabe powiązanie jednostek 1 : N	słaby zbiór strukturalny
Silne powiązanie jednostek 1 : 1	grupa powtarzalna reprezentująca podporządkowaną jednostkę o ilości powtórzeń równej jeden
Lewostronnie słabe powiązanie jednostek 1 : 1	—
Prawostronnie słabe powiązanie jednostek 1 : 1	—
Słabe powiązanie jednostek 1 : 1	—
Atrybut jednostki	dane elementarne
Atrybut identyfikujący jednostkę	dane elementarne opisane jako identyfikator rekordu
Zbiór wartości własności	—

W wypadku powiązań jednostek typu N : M ograniczenia sieciowego modelu danych pozwalają na bezpośrednie odwzorowanie jedynie słabego powiązania. Ponieważ reprezentujący to powiązanie jednostek rekord relacji nie może istnieć samodzielnie (np. nie ma dostawy bez towaru i dostawcy), to musi on być opisany jako rekord członkowski dwóch silnych zbiorów strukturalnych. Silne powiązanie jednostek typu 1 : N może być reprezentowane wyłącznie przez grupę powtarzalną; ponieważ została ona opisana w danym typie rekordu, to musi istnieć w każdym jego wystąpieniu. Jest to mało praktyczny sposób reprezentowania powiązań jednostek typu 1 : N, jeśli liczba zależnych jednostek występujących w jednym powiązaniu może być duża. Podobnie można opisać silne powiązanie jednostek typu 1 : 1 deklarując ilość wystąpień grupy powtarzalnej równą jeden. Zasady reprezentacji słabego i lewostronnie słabego powiązania jednostek typu 1 : N zostały już omówione.

W sieciowym modelu danych Komitetu CODASYL nie występuje pojęcie zbioru wartości, w którym mogą być określone dane elementarne. W tej sytuacji pojęcie zbioru wartości własności wykorzystywane w pojęciowym modelu danych nie ma swojego odpowiednika. Wszystkie te zasady semantyczne, których nie możemy bezpośrednio zapisać w sieciowym modelu danych, muszą być uwzględnione w oprogramowaniu użytkowym systemu.

3. Hierarchiczny model danych

Hierarchiczny model danych jest strukturą, w której każda jednostka zwana segmentem, należąca do określonej klasy jednostek (segmentów), może uczestniczyć w roli podrzędnej w co najwyżej jednym powiązaniu jednostek, a w roli nadrzędnej w dowolnej liczbie takich powiązań. Strukturę danych uzupełniającą takie ograniczenia nazywamy również strukturą drzewiastą. Istotną cechą struktury drzewiastej jest to, iż zawsze musi istnieć jeden taki jej węzeł, który nie jest podporządkowany żadnemu węzłowi tej struktury. Taki węzeł struktury drzewiastej nazywamy korzeniem drzewa.

Klasycznym przykładem systemu ZBD opartego na hierarchicznym modelu danych jest *Information Management System*

(IMS) firmy IBM. IMS był opracowany w 1968 r. przez firmę amerykańską ROCKWELL INTERNATIONAL, a następnie zakupiony i rozwijany przez firmę IBM. Obecnie istnieją dwie podstawowe wersje tego systemu: IMS 2 dla komputerów IBM serii 360 oraz IMS/VS dla komputerów serii 370. Ze względu na ogromną popularność sprzętu komputerowego firmy IBM, system ZBD ma obecnie najwięcej eksploatowanych zastosowań.

Podstawowymi pojęciami modelu danych, obsługiwanego przez IMS, są fizyczna i logiczna baza danych oraz fizyczny i logiczny rekord. Fizyczny rekord jest drzewiastą strukturą danych, zawierającą dowolną ilość wystąpień segmentów, przy czym jeden z nich musi być segmentem-korzeniem (*root segment*), rozmieszczonych na co najwyżej 15 poziomach hierarchicznych. Można odwzorować 255 różnych typów (28-1) segmentów w ramach jednego typu rekordu fizycznego. Fizyczna baza danych może zawierać wystąpienia jednego lub więcej typów rekordów fizycznych. Zgodnie z definicją struktury drzewiastej każdy segment (z wyjątkiem korzenia) jest podporządkowany tylko jednemu segmentowi w ramach tego rekordu fizycznego. Takie powiązanie segmentów rekordu nazywamy powiązaniem fizycznym, a segmenty nadrzędny i podrzędny w ramach takiego powiązania nazywamy odpowiednio fizycznym ojcem i fizycznym dzieckiem. Ze względu na to, iż między dwoma różnymi typami segmentów może istnieć, w ramach jednego typu rekordu fizycznego, tylko jeden typ powiązania, nie było potrzeby wprowadzania nazwy takiego powiązania.

Jak wiemy, wprowadzenie nazw zbiorów strukturalnych (powiązań rekordów), w sieciowym modelu danych, było konieczne dla jednoznacznej interpretacji takiej struktury danych.

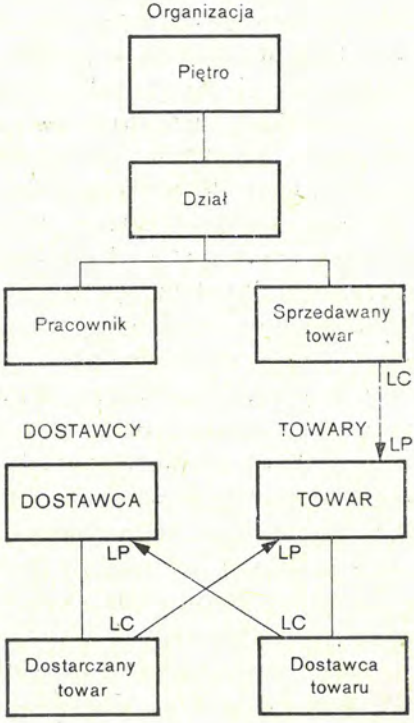
Fizyczne powiązania segmentów reprezentują lewostronne słabe powiązania jednostek typu 1 : N pojęciowego modelu danych. Fakt ten wynika z zasad budowy drzewiastego rekordu, gdzie każdy segment (z wyjątkiem korzenia) musi mieć fizycznego ojca, natomiast nie musi być związany z wystąpieniami segmentów typu opisanego jako jego fizyczne dziecko. Poprzestanie na takiej możliwości budowania struktury danych doprowadziłoby do poważnych ograniczeń w zastosowaniach systemu, a w najlepszym razie do dużej powtarzalności danych (redundancji).

Rozszerzeniem możliwości logicznej struktury danych IMS było wprowadzenie pojęć logicznego rekordu oraz logicznej bazy danych. Zasady budowy logicznego rekordu są takie same, jak w wypadku fizycznego, to jest stanowi on strukturę drzewiastą o określonej maksymalnej liczbie poziomów hierarchicznych. Istotną różnicą w stosunku do rekordu fizycznego jest możliwość budowania struktury drzewiastej rekordu logicznego, w skład której wchodzą segmenty różnych rekordów fizycznych, zapamiętanych w tej samej lub różnych fizycznych bazach danych. Zbiór logicznych rekordów nazywamy logiczną bazą danych. Logiczny rekord oraz logiczna baza danych są pojęciami abstrakcyjnymi, ponieważ w odróżnieniu od ich fizycznych odpowiedników nie są przechowywane na nośnikach urządzeń pamięci zewnętrznej. Materializacja logicznych rekordów przez ich odwzorowanie w zapisanych w komputerze fizycznych rekordach jest jedną z podstawowych funkcji IMS. Możliwość takiej materializacji wynika z utrzymywania przez system logicznych powiązań między rekordami fizycznymi. Występujące w ramach takiego powiązania segmenty nadrzędny i podrzędny nazywamy odpowiednio logicznym ojcem i logicznym dzieckiem.

Logiczny rekord jest obrazem struktury danych wykorzystywanej przez użytkownika, a raczej przez jego program użytkowy. Każdy program użytkowy może równolegle korzystać z wielu logicznych rekordów. Tak więc wizja struktury danych, jaką ma użytkownik IMS, jest zawsze hierarchiczna, natomiast dzięki tworzeniu logicznych powiązań możemy w bazie danych odnotować powiązania charakterystyczne dla sieciowej struktury danych (np. powiązanie jednostek typu N : M). Logiczne powiązania są realizowane przez łączniki adresowe lub przez wartości pól kluczowych. W tym pierwszym wypadku segment wiążący zawiera łączniki adresowe, a w drugim — odpowiednie wartości pól. W obu wypadkach segment wiążący może zawierać informacje charakterystyczne dla reprezentowanego powiązania (np. cena jednostkowa konkretnego towaru sprzedawanego w określonym dziale).

Powiązanie typu 1 : N między segmentami rekordów fizycznych jest reprezentowane przez jednokierunkowe logiczne powiązanie segmentów, natomiast powiązanie typu N : M wymaga zadekla-

rowania dwukierunkowego logicznego powiązania segmentów. Zanim omówimy szczegółowo możliwości odwzorowania opisu pojęciowego modelu danych w hierarchicznym modelu danych, przedstawimy przykład hierarchicznej struktury danych domu towarowego, odpowiadającej graficznej reprezentacji pojęciowego modelu danych, przedstawionego na rysunku 18. Schemat tej struktury w konwencji stosowanej w IMS przedstawiamy na rysunku 24.



Rys. 24. Schemat hierarchicznej struktury danych domu towarowego w systemie IMS

Struktura danych domu towarowego została opisana jako trzy fizyczne rekordy: ORGANIZACJA, DOSTAWCY I TOWARY, powiązane między sobą logicznie. Rekord fizyczny ORGANIZACJA jest związany jednokierunkowym powiązaniem segmentów

z rekordem fizycznym TOWARY, przy czym logicznym ojcem (LP) jest segment TOWAR, a logicznym dzieckiem — segment SPRZEDAWANY TOWAR (LC). Zauważmy, że jeżeli strzałka jest skierowana odwrotnie niż w notacji Bachmana, to logiczne powiązanie segmentów reprezentuje opisane w pojęciowym modelu danych silne powiązanie jednostek typu 1 : N „sprzedaż". W tym wypadku mamy do czynienia ze stratą informacji, ponieważ zgodnie z zasadami budowy fizycznych rekordów segment SPRZEDAWANY TOWAR nie musi wystąpić w fizycznym rekordzie ORGANIZACJA, co oznacza, że może wystąpić dział, który nie sprzedaje żadnego towaru. Natomiast w pojęciowym modelu danych opisaliśmy zasadę semantyczną, z której wynika, że każdy dział musi sprzedawać przynajmniej jeden towar. Oczywiście ta zasada semantyczna musi, w tej sytuacji, być realizowana przez oprogramowanie użytkowe, utrzymujące omawianą strukturę danych. Rekordy fizyczne DOSTAWCY I TOWARY są powiązane dwukierunkowo, logicznymi ojcami w tym powiązaniu są segmenty DOSTAWCA i TOWAR, a logicznymi dziećmi segmenty DOSTARCZANY TOWAR i DOSTAWCA TOWARU.

To dwukierunkowe powiązanie segmentów reprezentuje opisane w pojęciowym modelu danych lewostronnie słabe powiązanie jednostek typu N : M „dostawa". Podobnie jak w wypadku poprzedniego powiązania jednostek zachodzi strata informacji, ponieważ mamy do czynienia ze słabym powiązaniem. Na podstawie zasad budowania rekordu fizycznego nie możemy powiedzieć — że segmenty wiążące DOSTARCZANY TOWAR i DOSTAWCA TOWARU muszą zawsze występować odpowiednio w rekordach fizycznych DOSTAWCY i TOWAR.

Pozostałe powiązania jednostek opisane w pojęciowym modelu danych domu towarowego, a mianowicie „lokalizacja" oraz „zatrudnienie" są reprezentowane jako fizyczne powiązania segmentów w ramach rekordu fizycznego ORGANIZACJA. I w tym wypadku gubimy informację o fakcie silnego powiązania jednostek. W omawianym hierarchicznym modelu danych domu towarowego możemy przyjąć następującą strukturę segmentów:

PIĘTRO = NUMER-PIĘTRA, POWIERZCHNIA
DZIAŁ = NUMER DZIAŁU, TYP-ASORTYMENTU, WARTOŚĆ SPRZEDAŻY

PRACOWNIK = IMIĘ, NAZWISKO, FUNKCJA, POBORY

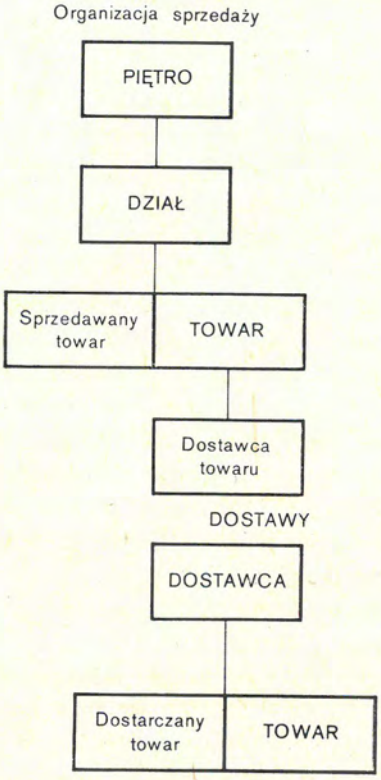
DOSTAWCA = NAZWA-DOSTAWCY, WARTOŚĆ--DOSTAW

TOWAR SPRZEDAWANY = NAZWA TOWARU, KOD, CENA

TOWAR DOSTARCZANY = KOD

TOWAR = łączniki adresowe

DOSTAWCA TOWARU = łączniki adresowe



Rys. 25. Przykłady logicznych rekordów w strukturze danych systemu IMS

Tablica 5

Zasady odwzorowania elementów pojęciowego modelu danych w hierarchicznym modelu danych (IMS)

Pojęciowy model danych	Hierarchiczny model danych (IMS)
Jednostka	Segment
Silne powiązanie jednostek N : M	—
Lewostronnie słabe powiązanie jednostek N : M	—
Prawostronnie słabe powiązanie jednostek N : M	—
Słabe powiązanie jednostek N : M	dwukierunkowe logiczne powiązanie segmentów
Silne powiązanie jednostek 1 : N	—
Lewostronnie słabe powiązanie jednostek 1 : N	fizyczne powiązanie segmentów w ramach rekordu drzewiastego
Prawostronnie słabe powiązanie jednostek 1 : N	—
Słabe powiązanie jednostek 1 : N	jednokierunkowe logiczne powiązanie segmentów
Silne powiązanie jednostek 1 : 1	—
Lewostronnie słabe powiązanie jednostek 1 : 1	—
Prawostronnie słabe powiązanie jednostek 1 : 1	—
Słabe powiązanie jednostek 1 : 1	—
Atrybut jednostki	pole
Atrybut identyfikujący jednostkę	pole kluczowe

W wypadku segmentu wiążącego SPRZEDAWANY TOWAR przyjęliśmy realizację logicznego powiązania segmentów przez wartość pola kluczowego (KOD), a dwukierunkowe logiczne powiązanie rekordów DOSTAWCY i TOWARY jest odwzorowane za pomocą łączników adresowych. Na podstawie omawianego przykładu widać wyraźnie, w jaki sposób można w hierarchicznym modelu danych IMS uniknąć redundancji danych.

Na rysunku 25 pokazano dwa rekordy logiczne, możliwe do opisania na przedstawionej strukturze rekordów fizycznych. Naturalnie takich logicznych rekordów można opisać znacznie więcej. Materializacja tych rekordów logicznych następuje na podstawie istniejących w fizycznych bazach danych odpowiednich powiązań fizycznych i logicznych. Wiążące segmenty są konkatenowane w rekordach logicznych, co umożliwia fizyczny zapis segmentów bez zbędnie powtarzanych pól.

W tablicy 5 zebrano zasady odwzorowania elementów pojęciowego modelu danych w hierarchicznym modelu danych IMS.

Wszystkie pokazane w tablicy zasady odwzorowania zostały już omówione. Podobnie jak w wypadku sieciowego modelu danych wszystkie te zasady semantyczne, których nie możemy bezpośrednio zapisać w hierarchicznym modelu danych, muszą być uwzględnione w oprogramowaniu użytkowym systemu informatycznego.

4. Relacyjny model danych

Wykorzystanie modeli danych (sieciowy, hierarchiczny) już omówionych prowadzi zwykle do powstawania złożonych struktur danych, obejmujących wiele różnych typów rekordów oraz powiązań między tymi rekordami. Wykorzystanie takiej struktury danych wymaga dobrej znajomości jej elementów, przynajmniej w zakresie działania programu użytkowego. Takiej znajomości rzeczy można wymagać od użytkowników profesjonalnych (programista, projektant, administrator bazy danych), natomiast trudno jest oczekiwać, by użytkownik końcowy, działający bezpośrednio na bazie danych, mógł bez trudu zrozumieć i właściwie wykorzystać złożoną strukturę danych. Ponieważ bezpośrednia praca użytkowników końcowych na podstawie bazy danych jest jednym z podstawowych wymagań stawianych współczesnym systemom ZBD, wprowadzenie modelu danych spełniającego wymagania tych użytkowników było konieczne.,

Relacyjny model danych sformułował E. F. Codd, pracownik badawczy firmy IBM, publikując od 1970 r. serię artykułów [5],

[5] Por. E. F. Codd, *A Relational Model of Data for Large Shared Data Banks*, CACM nr 6, vol. 13; E. F. Codd, *Relational Completeness of Data Base Sublanguages*,

obejmujących podstawy teoretyczne i charakterystykę użytkową tego modelu danych. W tym okresie opracowano wiele systemów ZBD, działających na podstawie relacyjnej struktury danych lub uwzględniających ją jako jedną z dopuszczalnych klas logicznej struktury danych. Wiele z nich było eksperymentalnymi systemami opracowanymi w ośrodkach badawczych, natomiast takie systemy, jak MAGNOM czy QUERY BY EXAMPLE są obecnie dostępne w wersjach handlowych. System MAGNOM [6] opracowano w 1975 roku przez firmę TYMSHARE Ine w USA, a QUERY BY EXAMPLE [7] przez firmę IBM. Do najbardziej znanych systemów relacyjnych należą trzy eksperymentalne systemy SYSTEM-R [8] PRTB (*Peterlec Relational Test Velsicle* [9] oraz INGRES [10]. Dwa pierwsze systemy były opracowane w ośrodkach badawczych firmy IBM w Stanach Zjednoczonych i w Wielkiej Brytanii, a trzeci na Uniwersytecie w Berteley. W Polsce opracowano System ZBD RODAN, który dopuszcza relacyjny model danych jako logiczną strukturę danych użytkowników końcowych [11].

Relacyjne podejście do struktury danych ma swoje teoretyczne podłoże w podstawowych pojęciach logiki matematycznej i teorii mnogości. Dla dowolnych zbiorów $S_1, S_2, ..., S_n$ (niekoniecznie rozłącznych) R jest relacją określoną na tych n zbiorach, jeżeli jest ona zbiorem n-tek takich, że pierwszy element każdej z nich należy do zbioru S_1, drugi element należy do zbioru S_2 itd. Mówimy, że zbiór S_j jest j-tą dziedziną relacji R ($j = 1, ..., n$). Zgodnie z tą definicją relacja R jest n-tego stopnia.

Relacja R może być formalnie zdefiniowana jako podzbiór iloczynu kartezjańskiego zbiorów $S_1 \times S_2 \times, ..., \times S_n$. Rozważając relację R z punktu widzenia pojęciowego modelu danych możemy

---

Proc. Courant Computer Science, Symposium 6, 1971, Prentice Hall; E. F. Codd, *Recent Investigations in Relational Data Base Systems*, Proc. IFIP Congress 1974, North Holland.

[6] Por. Won Kim, *Relational Database Systems*, ACM, „Computing Surveys" 1979, nr 3, vol. 11.

[7] Por. M. M. Astraham, D. D. Chamberlin, *Implementation of a Structured English Query Language*, Proc. SICMOD 75, ACM, New York.

[8] Por. T. M. Astraham, *System R: Relational Approach to Database Management*, ACM TODS, 1976, nr 2, vol. 1.

[9] Por. Won Kim, op. cit.

[10] Por. M. Stonebrauer, E. Wong, P. Krebs, G. Held, *The Design and Implementation of IGRES*, ACM TODS, 1976, nr 3, vol. 1.

[11] Por. Techniczna dokumentacja CPiZI, *System ZBD RODAN*, wyd. cyt.

powiedzieć, że taka relacja jest reprezentacją pewnej klasy jednostek, a poszczególne jej n-tki odpowiadają jednostkom należącym do tej klasy. Taka interpretacja relacji wymaga przyjęcia założenia jej zmienności w czasie, tym samym jest istotnym odstępstwem od matematycznej interpretacji tego pojęcia. Relacyjną strukturę danych możemy rozpatrywać jako pewien zbiór tablic reprezentujących w tej strukturze danych relacje, przy czym dla dowolnej relacji n-tego stopnia R odpowiadająca jej tablica spełnia warunki:

1) każdy wiersz tablicy odpowiada n-tce relacji R,
2) kolejność wierszy nie ma znaczenia,
3) wszystkie wiersze są różne,
4) kolejność kolumn ma istotne znaczenie i odpowiada porządkowi $S_1, S_2, ..., S_n$ dziedzin, na których jest określona relacja R,
5) znaczenie każdej z kolumn wynika częściowo z jej nazwy odpowiadającej nazwie odpowiedniej dziedziny.

Przykładem relacji n-tego stopnia jest relacja „dostawa", obejmująca wszystkie dostawy materiałów dla poszczególnych zleceń produkcyjnych, wykonywane przez określonych dostawców.

DOSTAWA	DOSTAWCA	MATERIAŁ	ZLECENIE	ILOŚĆ
1	2	5	17
1	3	5	23
2	3	7	9
2	7	5	4
4	1	1	12

Jak nietrudno zauważyć, że każda kolumna tablicy ma swoją nazwę, określającą pewien zbiór wartości (dziedzinę relacji). Powstaje więc pytanie, dlaczego kolejność kolumn ma istotne znaczenie. Rozważmy inny przykład dotyczący problemu tzw. rozwinięć technologicznych. Relacja trzeciego stopnia PODZESPÓŁ obejmuje trzy dziedziny, przy czym dwie pierwsze nazwijmy CZĘŚĆ, a trzecią ILOŚĆ. Przyjmijmy, że n-tka (x, y, z) oznacza, że część y wchodzi bezpośrednio w skład części x w ilości z.

Dwie pierwsze dziedziny są identyczne (CZĘŚĆ, CZĘŚĆ) i obejmują zbiór wszystkich występujących części. Istotną sprawą jest rola przyjęta dla poszczególnych dziedzin relacji (w tym wypadku część nadrzędna i podrzędna). W naszym przykładzie rola

PODZE-SPÓŁ	CZĘŚĆ	CZĘŚĆ	ILOŚĆ
	1	5	9
	2	5	7
	3	5	2
	2	6	12
	3	6	3
	4	7	1
	6	7	1

przyjęta dla poszczególnych dziedzin relacji wynika z ich kolejności. Jeżeli przyjmiemy, że relacja R jest określona na zbiorach $D_1, D_2, ..., D_n$ stanowiących dziedziny tej relacji, to możemy powiedzieć, że każdy taki zbiór jest zbiorem własności jednostek, należących do klasy jednostek opisanych przez tę relację.

Każdej własności jednostek możemy przypisać nazwę. Zbiór nazw własności jednostek będziemy nazywali zbiorem atrybutów relacji. W wypadku jednak, gdy mamy do czynienia z relacją, której dziedziny są określone na tym samym zbiorze wartości własności (np. CZĘŚĆ), dla uzyskania jednoznacznych atrybutów musimy określić rolę, jaką spełniają te dziedziny w rozważanej relacji. Jedną z możliwych konwencji jest dodawanie przedrostków do nazw poszczególnych własności, określających ich rolę w danej relacji. Tak więc relacja PODZESPÓŁ będzie wyglądała następująco:

PODZE-SPÓŁ	NADRZĘDNA CZĘŚĆ	PODRZĘDNA CZĘŚĆ	ILOŚĆ
	1	5	9
	2	5	7
	3	5	2
	2	6	12
	3	6	3
	4	7	1
	6	7	1

W takim wypadku możemy przyjąć, że kolejność kolumn tablicy będącej odwzorowaniem relacji nie ma znaczenia. Zgodnie z przyjętą definicją dla każdej relacji w bazie danych musi ist-

nieć atrybut lub zbiór atrybutów, których wartości jednoznacznie identyfikują każdą n-tkę relacji. Klucz K relacji R jest podzbiorem atrybutów tej relacji taki, że spełnione są następujące warunki:

1) dla każdej n-tki relacji R wartość K jednoznacznie identyfikuje tę n-tkę,

2) żaden atrybut zawarty w K nie może zostać pominięty bez naruszenia warunku identyfikalności 1.

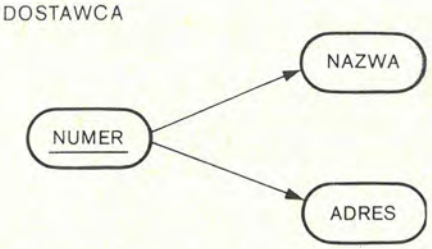
Dla każdej relacji może istnieć pewna liczba atrybutów lub zbiorów atrybutów spełniających te warunki. Takie atrybuty nazywamy potencjalnymi kluczami relacji, przy czym jeden z takich potencjalnych kluczy musi być określony jako główny klucz relacji. Klucz główny może być traktowany jako jednoznaczny identyfikator jednostek należących do klasy jednostek opisanych za pomocą danej relacji. Atrybuty wchodzące w skład klucza głównego relacji nazywamy atrybutami głównymi. Natomiast atrybuty, które nie są elementami klucza głównego relacji nazwiemy atrybutami zwykłymi. Ponieważ wartości atrybutów głównych jednoznacznie wyróżniają każdą n-tkę relacji spośród innych n-tek tej relacji, muszą być one zawsze określone. Inaczej mówiąc, dla głównych atrybutów relacji nie są dopuszczane wartości puste.

Istotnym problemem projektowania relacyjnej struktury danych jest określenie wzajemnych związków między atrybutami relacji. Istnieją wypadki, w których wartość jednego atrybutu relacji w pełni determinuje wartości innych atrybutów. Taki związek zachodzący między dziedzinami jednej relacji nazywamy zależnością funkcyjną.

Dla dowolnej relacji T mówimy, że dziedzina Y relacji R jest funkcyjnie zależna od dziedziny X relacji R wtedy i tylko wtedy, gdy każdej wartości X w relacji R jest przypisana dokładnie jedna wartość Y w relacji R. Rozważmy relację DOSTAWCA (NUMER, NAZWA, ADRES), w której wartości atrybutu NUMER jednoznacznie identyfikują każdego dostawcę i determinują jego NAZWĘ i ADRES. Zależności funkcyjne występujące między atrybutami tej relacji przedstawiamy na rysunku 26.

Z zależności funkcyjnej atrybutu ADRES od atrybutu NUMER wynika, że każdy dostawca może być zlokalizowany pod jednym tylko adresem. Jest to ograniczenie występujące w opisywanej

rzeczywistości. Dowolność występowania zależności funkcyjnych między atrybutami relacji może doprowadzić do istotnych problemów w wypadku wykonywania operacji aktualizujących te relacje. Skutki uboczne takich operacji omówimy w kontekście procesu normalizacji struktury relacyjnej bazy danych.



Rys. 26. Zależności funkcyjne w relacji DOSTAWCA

Normalizacja jest procesem stopniowego zastępowania dowolnego zbioru relacji, przez kolejne zbiory relacji, zmierzającym do uzyskania prostszej i bardziej regularnej struktury danych. Odwracalność tego procesu gwarantuje możliwość powrotu do wyjściowego zbioru relacji, nie dopuszczając tym samym do straty informacji.

Rozważając proces normalizacji omówimy trzy tzw. normalne formy relacji oraz *Optymalną Trzecią Normalną Formę* (określaną niekiedy jako IV forma normalna lub *Boyce-Codd Normal Form* (BCNF).

Pierwsza Normalna Forma (1NF). Relacja jest w pierwszej normalnej formie, jeżeli każdy atrybut tej relacji jest określony na prostej dziedzinie. Pierwsza normalna forma dotyczy struktury relacji, przy czym relacja w 1NF nie może obejmować żadnej dziedziny, która sama jest relacją. Istnienie nieznormalizowanych relacji w bazie danych w poważnym stopniu utrudnia stosowanie rachunku relacji, a w niektórych wypadkach uniemożliwia jednoznaczne wybieranie *n*-tek tej relacji.

Rozważmy relację:

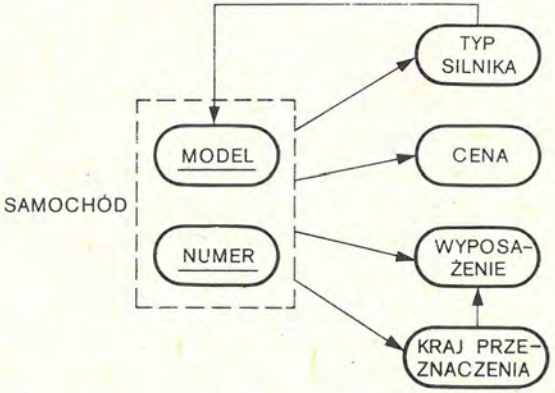
SAMOCHÓD (MODEL, NUMER, TYP SILNIKA, CENA, DOSTAWA)

Jeżeli DOSTAWA jest relacją (WYPOSAŻENIE, KRAJ PRZEZ-

NACZENIA), to relacja SAMOCHÓD nie jest w 1NF. Każda relacja może być przepisana na relację w 1NF w prosty algorytmiczny sposób. Relacja SAMOCHÓD (MODEL, NUMER, TYP SILNIKA, CENA, WYPOSAŻENIE, KRAJ PRZEZNACZENIA) jest relacją w 1NF.

Przyjmijmy, że w opisywanej przez relację SAMOCHÓD modelu danych rzeczywistości obowiązują następujące zasady spójności logicznej:

— każdy samochód jest zaopatrzony w jednoznaczny numer w ramach produkowanego modelu, czyli identyfikatorem samochodu są atrybuty MODEL i NUMER,



Rys. 27. Graficzne zależności funkcyjne w relacji SAMOCHÓD

— atrybuty TYP SILNIKA, CENA, WYPOSAŻENIE i KRAJ PRZEZNACZENIA są determinowane przez identyfikator (MODEL, NUMER),

— atrybut WYPOSAŻENIE jest determinowany przez KRAJ PRZEZNACZENIA,

— atrybut MODEL jest determinowany przez TYP SILNIKA.

Zależności funkcyjne, zachodzące w relacji SAMOCHÓD, przedstawia rysunek 27, przy czym klucz główny relacji obejmuje atrybuty MODEL, NUMER.

Druga Normalna Forma (2NF). Znormalizowana relacja *R* jest w drugiej normalnej formie wtedy i tylko wtedy, gdy wszystkie

zwykłe atrybuty tej relacji są funkcjonalnie zależne od jej klucza głównego. Z zależności funkcyjnych w relacji SAMOCHÓD wynika, że ta relacja spełnia wymagania określone dla 2NF. Niezależnie jednak od funkcyjnych zależności atrybutów tej relacji od klucza głównego występują zależności między jej zwykłymi atrybutami. W takiej sytuacji mówimy, że zachodzi funkcyjna zależność przechodnia między zwykłymi atrybutami a kluczem głównym relacji.

W wypadku omawianej relacji atrybut WYPOSAŻENIE jest zależny przechodnio od klucza głównego przez atrybut KRAJ-PRZEZNACZENIA, a atrybut MODEL przez atrybut TYP-SILNIKA. Aktualizacja relacji w 2NF może doprowadzić do wielu niepożądanych skutków. Omawiając operacje aktualizacji wyróżniamy trzy podstawowe ich typy: dodawanie n-tki, skreślanie n-tki oraz aktualizację wartości atrybutów n-tki.

Dodawanie. W naszym przykładzie zależność między atrybutem KRAJ-PRZEZNACZENIA a atrybutem WYPOSAŻENIE jest istotną informacją. Jest rzeczą zupełnie prawdopodobną, że w wypadku dostaw do nowego kraju może powstać potrzeba wprowadzenia informacji dotyczących wyposażenia wymaganego na tym rynku, zanim zostanie podjęta produkcja wysyłanych tam samochodów. Wprowadzenie takiej informacji wymagałoby utworzenia następującej n-tki:

L, —,—,—,—, specjalne, USA,

gdzie "—" oznacza wartość pustą.

Wprowadzenie takiej n-tki do relacji SAMOCHÓD nie jest jednak możliwe, ponieważ wartości atrybutów, wchodzących w skład klucza głównego, nie mogą zawierać wartości pustych.

Skreślanie. Jeżeli skreślimy ostatni samochód dostarczany do danego kraju, to tracimy także istotną dla nas informację dotyczącą wyposażenia samochodu wymaganego w tym kraju.

Aktualizacja. Jeżeli nastąpi zmiana wymagań w stosunku do wyposażenia samochodu eksportowanego do danego kraju, to wymagana jest aktualizacja tych wszystkich n-tek relacji, które zawierają odpowiednie wartości. Ponieważ mamy do czynienia z wielokrotną aktualizacją, wynikającą z jednej zmiany, to opuszczenie jakiejkolwiek n-tki doprowadzi do pozostawienia sprzecznych informacji w bazie danych. Dalszy proces normalizacji po-

winien doprowadzić do zlikwidowania problemów związanych z aktualizacją.

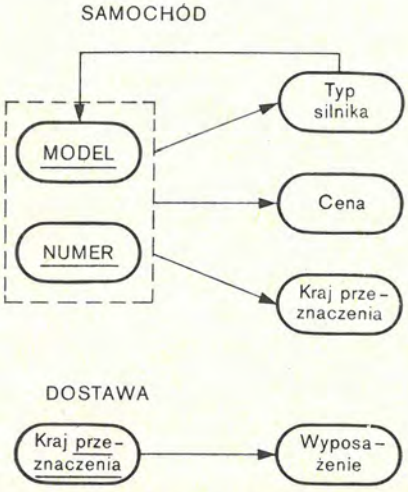
Trzecia Normalna Forma (3NF). Relacja R jest w Trzeciej Normalnej Formie, jeżeli jest ona w 2NF i żaden zwykły atrybut tej relacji nie jest zależny przechodnio od jakiegokolwiek z jej kluczy.

Doprowadzenie relacji SAMOCHÓD do 3NF wymaga rozbicia jej na dwie następujące relacje:

SAMOCHÓD (MODEL, NUMER, TYP SILNIKA, CENA, KRAJ PRZEZNACZENIA)

DOSTAWA (KRAJ PRZEZNACZENIA, WYPOSAŻENIE).

Zależności funkcyjne w tych relacjach przedstawiamy na rysunku 28.



Rys. 28. Zależności funkcyjne między relacjami SAMOCHÓD i DOSTAWA

Łatwo zauważyć, że obie relacje są w 3NF, a pozostawiona w relacji SAMOCHÓD przechodnia zależność między atrybutami TYP SILNIKA i MODEL jest zgodna z przytoczoną definicją 3NF, ponieważ atrybut MODEL jest głównym atrybutem (wchodzi w skład klucza relacji SAMOCHÓD). Nie zachodzi również niebezpieczeństwo straty informacji, ponieważ można odtworzyć relację SAMOCHÓD w jej pierwotnej formie: przez wykonanie

operacji łączenia relacji na podstawie dziedziny KRAJ PRZEZNACZENIA (operacja JOIN). Normalizacja relacji SAMOCHÓD doprowadziła do uzyskania pełnej wzajemnej niezależności wszystkich jej niegłównych atrybutów. W tej sytuacji nie zachodzą omówione już problemy aktualizacji relacji SAMOCHÓD, wynikające z zależności między atrybutami KRAJ PRZEZNACZENIA a WYPOSAŻENIE. Okazuje się jednak, że zgodna z definicją 3NF przechodnia zależność atrybutu głównego MODEL (przez atrybut TYP SILNIKA) od klucza relacji (MODEL, NUMER) może również doprowadzić do powstania problemów związanych z aktualizacją tej relacji. Zgodnie z przyjętą zasadą semantyczną model samochodu (MODEL) jest determinowany przez TYP SILNIKA. W tej sytuacji *aktualizacja*, a szczególnie *dodawanie* i *skreślanie*, może doprowadzić do podobnych problemów, jakie wystąpiły w wypadku relacji 2NF. Ze schematu zależności funkcyjnych wynika, że atrybuty MODEL i TYP SILNIKA pozostają w związku typu 1 : 1, czyli potencjalnymi kluczami tej relacji mogą być pary atrybutów MODEL, NUMER i TYP SILNIKA, NUMER.

Optymalna Trzecia Normalna Forma (Boyce- Codd Normal Forn (BCNF)). Relacja *R* jest w optymalnej trzeciej normalnej formie (BCNF), jeżeli jest w 1NF i dla każdego zbioru atrybutów *C* relacji *R*, jeżeli jakikolwiek atrybut nie zawarty w *C* jest funkcyjnie zależny od *C*, to wszystkie atrybuty relacji *R* są funkcyjnie zależne od *C*.

Z definicji tej wynika, że wszystkie atrybuty nie zawarte w kluczu są niezależne od siebie oraz dodatkowo wszystkie klucze są niezależne od niegłównych atrybutów i funkcyjnie zależne od innych kluczy.

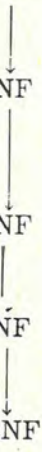
Zaletą tej definicji jest uniknięcie wprowadzenia pojęcia głównych atrybutów i zależności przechodniej. Projektowanie relacyjnej struktury danych zgodnie z zasadami optymalnej 3NF przyczynia się do zniknięcia redundancji kluczy relacji i prowadzi do bardziej regularnej struktury. Doprowadzenie do optymalnej 3NF wymaga dalszego rozbicia relacji SAMOCHÓD, co sprzyja powstaniu następujących relacji:

SAMOCHÓD (MODEL, NUMER, CENA, KRAJ PRZEZNACZENIA)

DOSTAWA (KRAJ PRZEZNACZENIA, WYPOSAŻENIE)
WERSJA (MODEL, TYP SILNIKA)

Projektowanie relacyjnej struktury danych wymaga przejścia przez następujące etapy normalizacji:

FORMA NIEZNORMALIZOWANA



1NF	ELIMINACJA ATRYBUTÓW STANOWIĄCYCH RELACJE
2NF	USTALENIE FUNKCYJNEJ ZALEŻNOŚCI ZWYKŁYCH ATRYBUTÓW OD KLUCZY
3NF	ELIMINACJA PRZECHODNICH ZALEŻNOŚCI ZWYKŁYCH ATRYBUTÓW OD KLUCZY
BCNF	ELIMINACJA REDUNDANCJI KLUCZY

Proces normalizacji prowadzi do uzyskania pożądanej struktury wewnątrz projektowanych relacji.

Omawiany w rozdziale III pojęciowy model danych domu towarowego możemy przedstawić jako relacyjną strukturę danych w następujący sposób:

Relacje	Atrybuty
DOSTAWCA	NAZWA-DOSTAWCY, WARTOŚĆ-DOSTAW
DOSTAWA	NAZWA-DOSTAWCY, KOD
TOWAR	NAZWA-TOWARU, KOD, CENA, NUMER-DZIAŁU
PIĘTRO	NUMER-PIĘTRA, POWIERZCHNIA
DZIAŁ	NUMER-DZIAŁU, TYP-ASORTYMENTU, WARTOŚĆ-SPRZEDAŻY, NUMER-PIĘTRA
PRACOWNIK	IMIĘ, NAZWISKO, FUNKCJA POBORY, NUMER-DZIAŁU

Atrybuty główne poszczególnych relacji są zaznaczone przez podkreślenie. Jak już mówiliśmy, pojęcie relacji, w relacyjnym

modelu danych, odpowiada pojęciu klasy jednostek. Natomiast powiązanie jednostek może być reprezentowane w strukturze relacyjnej jako osobna relacja (np. DOSTAWA) lub wynikać z atrybutów obu powiązanych relacji. W tym drugim wypadku materializacja powiązania relacji odbywa się przez wykonanie operacji łączenia relacji. Rodzaj reprezentowanych powiązań jednostek można przedstawić w strukturze relacyjnej przez przyjęcie odpowiednich ograniczeń dotyczących dziedzin relacji, będących argumentami operacji łączenia lub dziedzin relacji dwuczłonowej. Szersze omówienie zagadnienia reprezentacji powiązań jednostek w relacyjnym modelu danych wymaga wprowadzenia dwóch operacji algebry relacji, a mianowicie operacji łączenia i operacji rzutu. Niech Θ oznacza jakikolwiek z operatorów relacyjnych $=$, $\neq$, $<$, $\leqslant$, $>$ i $\geqslant$. Operację Θ - łączenia (Θ -join) relacji R na dziedzinę A z relacją S na dziedzinę B zapisujemy w sposób następujący:

$$R[A\Theta B]S = \{(r \cap s) : r \in R \wedge s \in S \wedge (r[A]\Theta s[B])\},$$

przy czym każdy element zbioru $R[A]$ musi być Θ — porównywalny z każdym elementem zbioru $S[B]$.
Niech r będzie n-tką relacji n-tego stopnia R.
Dla $j = 1, 2, ..., n$ zapis r [j] wyznacza j-ty składnik r.
Jeżeli $A = (j_1, j_2, ... j_k)$ jest ciągiem dodatnich liczb całkowitych ze zbioru $\{1, 2, ..., n\}$, to:

$$r[A] = (r[j_1], r[j_2], ..., r[j_k]).$$

Rzut relacji na listę dziedzin A zapisujemy:

$$R[A] = \{r[A] : r \in R\},$$

przy czym, jeżeli A jest zbiorem pustym, to $r[A] = r$.

Łatwo zauważyć, że jeżeli A jest permutacją listy $1, 2, ..., n$, to wynikiem operacji rzutowania jest relacja R o zmienionym porządku dziedzin. Lista dodatnich liczb jest zbiorem numerów dziedzin relacji R, które łatwo możemy zastąpić nazwami tych dziedzin.

Wracając do omawianego przykładu, zobaczmy, jakie należy przyjąć ograniczenia na dziedziny relacji DOSTAWA, by relacja ta była reprezentacją lewostronnie słabego powiązania jednostek typu N : M. Przede wszystkim, jeśli jeden dostawca może dostar-

czać wiele towarów i jeden towar może być dostarczany przez wielu dostawców, to każda n-tka relacji DOSTAWA może być jednoznacznie identyfikowana dopiero przez wartości obu atrybutów (NAZWA-DOSTAWCY, KOD). Dodatkowo lewostronnie słabe powiązanie jednostek wyraża warunek, że każdy towar musi być dostarczany przez przynajmniej jednego dostawcę. Korzystając z wprowadzonego zapisu operacji rzutowania, możemy wyrazić taki warunek w następujący sposób:

DOSTAWA (KOD) = TOWAR (KOD),

przy czym znak = wyraża równość zbiorów, jakie tworzą odpowiednio dziedziny relacji DOSTAWA i TOWAR oznaczone atrybutem KOD.

Warunek ten oznacza, że każda wartość kodu towaru, występująca w relacji TOWAR, musi wystąpić przynajmniej raz w relacji DOSTAWA. Jak już mówiliśmy, powiązania jednostek mogą być materializowane w strukturze relacyjnej przez operację łączenia relacji, czyli nie muszą być one opisane w postaci osobnych relacji. I w tym wypadku należy jednak określić charakter powiązania jednostek, nakładając odpowiednie warunki na dziedziny powiązanych relacji. Zatem w wypadku struktury danych domu towarowego musimy przyjąć następujące warunki:

PIĘTRO (NUMER-PIĘTRA) = DZIAŁ (NUMER-PIĘTRA)
DZIAŁ (NUMER-DZIAŁU) = TOWAR (NUMER-DZIAŁU)
DZIAŁ (NUMER-DZIAŁU) = PRACOWNIK (NUMER-
-DZIAŁU)

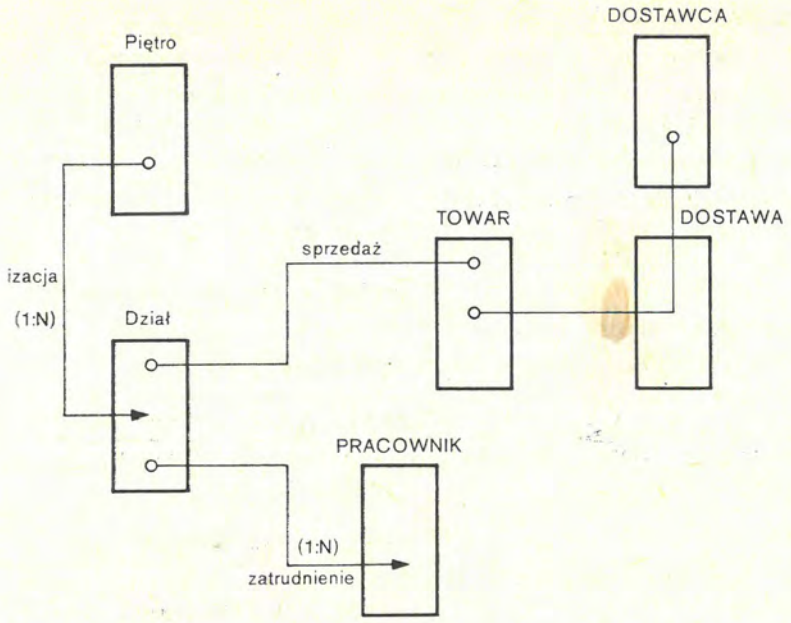
Odpowiednie operacje łączenia relacji zapisujemy:

PIĘTRO (NUMER-PIĘTRA = NUMER-PIĘTRA) DZIAŁ
DZIAŁ (NUMER-DZIAŁU = NUMER-DZIAŁU) TOWAR
DZIAŁ (NUMER-DZIAŁU = NUMER-DZIAŁU) PRACOWNIK

Tak zapisane operacje łączenia relacji w powiązaniu z już podanymi warunkami reprezentują odpowiednio silne powiązania jednostek typu 1 : N LOKALIZACJA, SPRZEDAŻ i ZATRUDNIENIE. Zauważmy, że jeden z argumentów operacji łączenia musi być w tym wypadku kluczem relacji.

Odpowiednią strukturę powiązań relacji w modelu danych domu towarowego przedstawia rysunek 29.

W większości znanych języków opisu relacyjnego modelu da-



Rys. 29. Struktura powiązań w relacyjnej strukturze danych domu towarowego

nych można łatwo zapisywać konieczne warunki, jakie muszą spełniać dziedziny poszczególnych relacji. W związku z tym uwzględniono tę możliwość w zebranych zasadach (por. tabl. 6) odwzorowania pojęciowego modelu danych.

Tablica 6

Zasady odwzorowania elementów pojęciowego modelu danych w relacyjnym modelu danych

Pojęciowy model danych	Relacyjny model danych
Jednostka	relacja
Silne powiązanie jednostek N : M	relacja dwuczłonowa R—S [A, B] taka, że R—S [A] = R [A] i R—S [B] = S [B]
Lewostronnie słabe powiązanie jednostek N : M	relacja dwuczłonowa R—S [A, B] taka, że R—S [B] = S [B]

c.d. tablicy 6

Pojęciowy model danych	Relacyjny model danych
Prawostronnie słabe powiązanie jednostek N : M	relacja dwuczłonowa R—S [$\underline{A}$, $\underline{B}$] taka, że R—S [A] = R [A]
Słabe powiązanie jednostek N : M	relacja dwuczłonowa R—S [A, B]
Silne powiązanie jednostek 1 : N	operacja łączenia R [$A \Theta B$] S, gdzie R [A] = S [B] i atrybut A jest kluczem relacji R
Lewostronnie słabe powiązanie jednostek 1 : N	operacja łączenia R [$A \Theta B$] S, gdzie R [A] $\underline{\supseteq}$ S [B] i atrybut A jest kluczem relacji R
Prawostronnie słabe powiązanie jednostek 1 : N	operacja łączenia R [$A \Theta B$] S, gdzie R [A] $\underline{\subseteq}$ S [B] i atrybut A jest kluczem relacji R
Słabe powiązanie jednostek 1 : N	operacja łączenia R [$A \Theta B$] S, gdzie atrybut A jest kluczem relacji R
Silne powiązanie jednostek 1 : 1	operacja łączenia R [$A \Theta B$] S, gdzie R (A) = S (B) i atrybuty A i B są odpowiednio kluczami relacji R i S
Lewostronnie słabe powiązanie jednostek 1 : 1	operacja łączenia R [$A \Theta B$] S, gdzie R (A) $\underline{\supseteq}$ S (B) i atrybuty A i B są odpowiednio kluczami relacji R i S
Prawostronnie słabe powiązanie jednostek 1 : 1	operacja łączenia R [$A \Theta B$] S, gdzie R (A) $\underline{\subseteq}$ S (B) i atrybuty A i B są odpowiednio kluczami relacji R i S
Słabe powiązanie jednostek 1 : 1	operacja łączenia R [$A \Theta B$] S, gdzie atrybuty A i B są odpowiednio kluczami relacji R i S
Atrybut jednostki	atrybut zwykły relacji
Atrybut identyfikujący jednostkę	atrybut główny relacji
Zbiór wartości własności	zbiór wartości, na którym jest określona dziedzina relacji

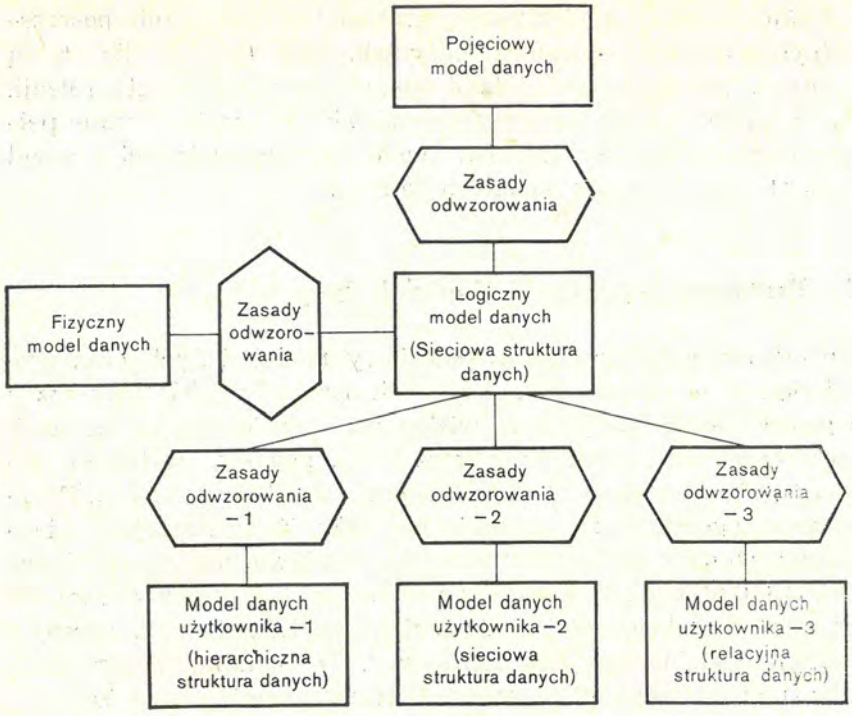
Symbole zapisu koniecznych warunków odwzorowania poszczególnych rodzajów powiązań jednostek, przyjęte w tablicy 6, są zgodne z podanymi definicjami operacji łączenia i rzutu relacji. Jak wynika z przedstawionych zasad odwzorowania, istnieje pełna zgodność możliwości opisu struktury rzeczywistości i zasad semantycznych w obu modelach danych.

5. Problemy koegzystencji modeli danych

Dotychczas przedstawiliśmy pojęciowy model danych oraz wynikające z możliwości współczesnych systemów ZBD trzy różne logiczne modele danych. W większości obecnie eksploatowanych systemów ZBD użytkownik korzysta z pewnego podzbioru logicznego modelu danych określonego dla całej bazy danych. Przykładem tego stanu rzeczy może być fakt, iż przeważającą większość systemów ZBD opracowano na podstawie propozycji Komitetu CODASYL lub IMS, gdzie użytkownik systemu obraca się w ramach jednego względnie stałego modelu danych, operując jedynie podzbiorem jego elementów. W niektórych systemach umożliwia się również ograniczone transformacje elementów modelu danych, jak np. zmiana struktury rekordu lub zmiana reprezentacji danej elementarnej w sieciowym modelu danych.

Omawiając relacyjny model danych zwróciliśmy uwagę na znacznie lepsze przystosowanie tej klasy struktury danych do potrzeb użytkownika końcowego. Rozwój zastosowań systemów ZBD, a szczególnie integracja zasobów informacyjnych, wykorzystywanych tradycyjnie przez różne grupy użytkowników, w ramach jednej bazy danych, doprowadziły do powstania systemów obejmujących wiele różnych logicznych modeli danych. Przykładem takiego podejścia do architektury systemu ZBD jest zaprezentowana architektura ANSI/SPARC (por. rozdz. III). W podobnym kierunku zmierzały prace badawcze G. M. Nijssena [12] z *Control Data Corporation*, który zaproponował architekturę systemu ZBD, obejmującą wiele modeli danych. Taki system nazywa on modelem koegzystencji. Istotną cechą zaproponowa-

[12] Por. *Codasyl Data Base Task Group*, DBTG, April 1971 Report, ACM, New York 1972.



Rys. 30. Wzajemne powiązania różnych modeli danych

nego tego modelu koegzystencji jest możliwość automatycznej realizacji odwzorowań między różnymi logicznymi modelami danych. Przykładem takiej architektury systemu ZBD jest schemat powiązań różnych modeli danych, przedstawiony na rysunku 30.

Ważną zmianą w stosunku do tradycyjnie stosowanych rozwiązań jest propozycja włączenia pojęciowego modelu danych do ogólnej architektury systemu ZBD. Zasady odwzorowania tego modelu danych w logiczny model danych już omówiliśmy.

Załóżmy, że w praktyce model koegzystencji przyjmie sieciową strukturę danych jako logiczny model oraz pozwoli na deklarowanie każdego z trzech podstawowych logicznych modeli danych, to jest sieciowego, hierarchicznego i relacyjnego modelu danych.

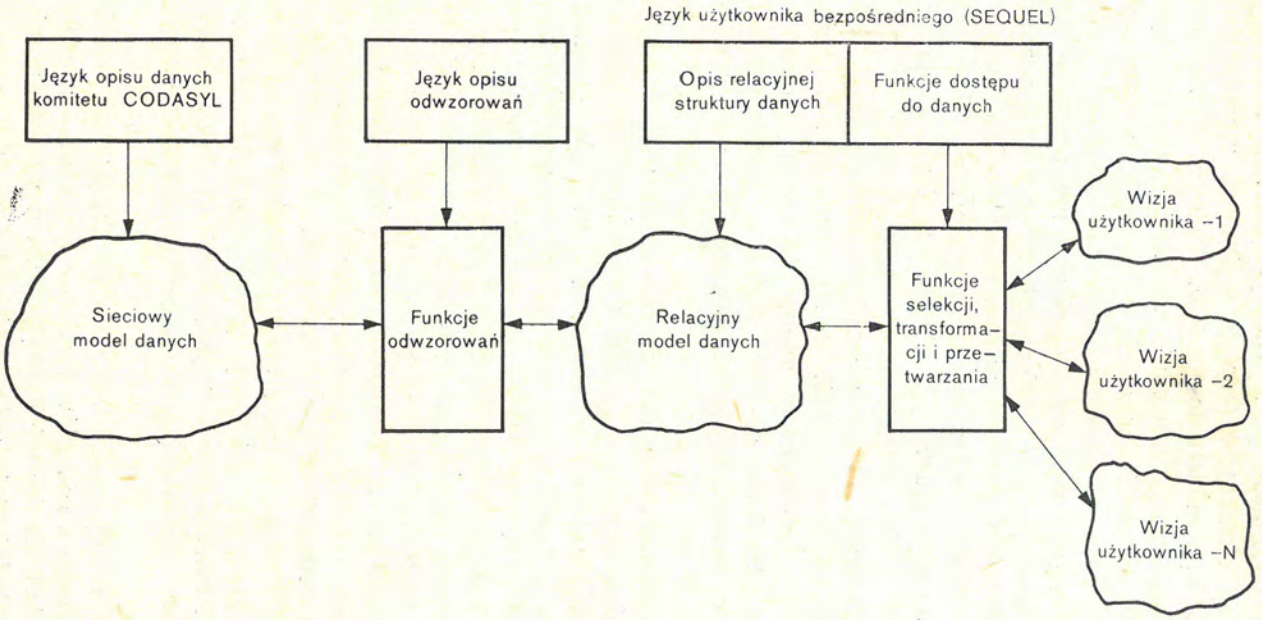
Problem transformacji logicznego modelu danych w fizyczny model danych jest przedmiotem następnego rozdziału. Zasta-

nówmy się natomiast, jakie istnieją możliwości automatycznej transformacji elementów jednego logicznego modelu danych na elementy innego modelu logicznego. Ważnym warunkiem takiej transformacji jest zapobieganie stracie informacji, jaka może nastąpić w momencie przejścia z jednego modelu na drugi. W wypadku przejścia między logicznymi modelami danych tego samego typu (np. sieciowy — sieciowy) nie powstaje takie niebezpieczeństwo, ponieważ informacje są wyrażone przez te same typy elementów tej struktury (np. zbiór strukturalny). Również w razie przejścia między modelem sieciowym a hierarchicznym modelem danych mamy jedynie do czynienia z ograniczeniem wizji użytkownika. Porównując oba te modele danych można łatwo zauważyć, że możliwości opisu struktury hierarchicznej są podzbiorem możliwości sieciowej struktury danych. Inaczej wygląda problem odwzorowań między relacyjną strukturą danych a sieciową strukturą danych. Jedną z podstawowych różnic między tymi modelami danych jest sposób opisu powiązań jednostek.

W pierwszym modelu danych rolę elementów wiążących spełniają atrybuty relacji, natomiast w drugim — zbiór strukturalny, oraz w wypadku powiązań jednostek typu N : M — rekord relacji. Wynikająca z tych różnic w zasadach budowy obu modeli danych niejednoznaczność zasad odwzorowań wymaga wprowadzenia dodatkowego opisu. Język takiego opisu jest zwykle nazywany językiem opisu odwzorowań. Przykładem takiego rozwiązania jest system ZBD RODAN, w którym zrealizowano możliwość odwzorowania sieciowej struktury danych Komitetu CODASYL w relacyjną strukturę danych. Architekturę tego rozwiązania przedstawiamy na rysunku 31.

Podstawowym elementem rozwiązania jest wprowadzony język opisu odwzorowań, pozwalający na jednoznaczną interpretację sieciowej struktury danych jako struktury relacyjnej. Podstawowe zasady, na których oparty jest proponowany język, są następujące:

1. Jedna relacja może być obrazem fragmentu modelu sieciowego złożonego z rekordów i zbiorów strukturalnych tworzących ciąg (ścieżkę), na której każdy rekord z wyjątkiem pierwszego (zgodnie z kolejnością opisywania) jest członkiem zbioru struk-



Rys. 31. Koegzystencja różnych logicznych modeli danych w SZBD RODAN

turalnego, którego właścicielem jest poprzedni rekord na ścieżce. Ścieżkę tę nazwano ścieżką główną.

2. Dodatkowo, dla każdego rekordu leżącego na ścieżce głównej, odpowiadającej danej relacji, fragment sieciowego modelu danych można rozszerzyć o rekordy i zbiory strukturalne tworzące ciągłe ścieżki, w których pierwszym rekordem jest dany rekord ścieżki głównej, a każdy następny jest właścicielem zbioru, którego członkiem jest poprzedni rekord ścieżki. Ścieżki te nazwano ścieżkami bocznymi.

Składnię języka opisu odwzorowań zrealizowanego w systemie RODAN przedstawiono w tablicy 7.

Tablica 7

Składnia języka opisu odwzorowań w systemie RODAN

MAPPING[a] FOR SCHEMA nazwa-modelu-sieciowego
IS nazwa-modelu-relacyjnego;
TABLE nazwa-relacji;

[{INCLUDE [RECORD nazwa-rekordu]
[{OWNER / MEMBER} OF nazwa-zbioru SET];} ...
[DOMAIN nazwa-kolumny-relacji
FROM {nazwa-danej [(lista-indeksów)] / PROCEDURE nazwa-procedury} ...]]

Zasady notacji:
(*a*) — *a* opcjonalne
{ *a* / *b* } — albo *a* albo *b*,
... — dowolna liczba powtórzeń

[a] Podkreślone frazy są obowiązkowe.

Podstawowe zasady dotyczące składni i semantyki języka opisu odwzorowań są następujące:

— fragment dotyczący opisu każdej relacji musi zaczynać się zdaniem TABLE,

— pierwszym zdaniem po zdaniu TABLE musi być zdanie INCLUDE opisujące pierwszy rekord ścieżki głównej,

— w zdaniu INCLUDE, opisującym pierwszy rekord ścieżki głównej, fraza OWNER/MEMBER może być pominięta,

— jeśli zdanie INCLUDE zawiera frazy MEMBER i opisywany w tym zdaniu zbiór strukturalny zawiera wiele typów członów, wówczas fraza RECORD musi być użyta dla zapewnienia jednoznaczności wyboru rekordu-członka,

— po każdym zdaniu INCLUDE, opisującym rekord ścieżki głównej (lub bocznej), może wystąpić ciąg zdań DOMAIN opisujących kolumny relacji, których wartości mają pochodzić z danych tego rekordu wymienionych odpowiednio we frazie FROM,

— jeśli we frazie FROM wymieniona jest nazwa powtarzalnej grupy danych, to musi zostać opisana lista indeksów identyfikujących pojedynczy element jej grupy danych,

— wyspecyfikowane we frazie FROM nazwy procedury oznacza, że wartości danej kolumny relacji mają być obliczane przez tę procedurę wołaną przez system każdorazowo w chwili materializacji kolejnego wiersza relacji (n-tki relacji [13]).

Dobrą ilustracją możliwości języka opisu odwzorowań systemu RODAN jest podany przykład opisu odwzorowania sieciowego modelu danych domu towarowego (por. rys. 29) w relacyjny model danych przedstawiony w punkcie 4.4. Opis reguł odwzorowania przedstawia tablica 8.

Tablica 8

Opis reguł odwzorowania sieciowej struktury danych domu towarowego w relacyjną strukturę danych

```
MAPPING FOR SCHEMA sieciowy-model-domu-towarowego
IS relacyjny-model-domu-towarowego;
TABLE PIĘTRO;
        INCLUDE RECORD PIĘTRO;
            DOMAIN NUMER-PIĘTRA FROM NUMER-PIĘTRA;
            DOMAIN POWIERZCHNIA FROM POWIERZCHNIA;
TABLE DZIAŁ;
            INCLUDE RECORD PIĘTRO;
            DOMAIN NUMER-PIĘTRA FROM NUMER-PIĘTRA;
            INCLUDE RECORD DZIAŁ MEMBER
            OF LOKALIZACJA SET;
            DOMAIN NUMER-DZIAŁU FROM NUMER-DZIAŁU;
```

[13] W pracy R. C. Date, *Wprowadzenie do baz danych* — tłumacze użyli terminu *krotka*.

```
        DOMAIN TYP-ASORTYMENTU FROM TYP-ASORTYMENTU;
        DOMAIN WARTOŚĆ-SPRZEDAŻY FROM WARTOŚĆ-
        -SPRZEDAŻY;
TABLE PRACOWNIK;
      INCLUDE RECORD DZIAŁ;
        DOMAIN NUMER-DZIAŁU FROM NUMER-DZIAŁU;
      INCLUDE RECORD PRACOWNIK MEMBER
        OF ZATRUDNIENIE SET;
        DOMAIN IMIĘ FROM IMIĘ;
        DOMAIN NAZWISKO FROM NAZWISKO;
        DOMAIN FUNKCJA FROM FUNKCJA;
        DOMAIN POBORY FROM POBORY;
TABLE TOWAR;
      INCLUDE RECORD DZIAŁ;
        DOMAIN NUMER-DZIAŁU FROM NUMER-DZIAŁU;
      INCLUDE RECORD TOWAR MEMBER
        OF ZATRUDNIENIE SET;
        DOMAIN NAZWA-TOWARU FROM NAZWA TOWARU;
        DOMAIN KOD FROM KOD;
        DOMAIN CENA FROM CENA;
TABLE DOSTAWA;
      INCLUDE RECORD TOWAR;
        DOMAIN KOD FROM KOD;
      INCLUDE RECORD DOSTAWA MEMBER
        OF JEST DOSTARCZANY SET;
      INCLUDE RECORD DOSTAWCA OWNER
        OF DOSTARCZA SET;
        DOMAIN NAZWA-DOSTAWCY FROM NAZWA-DOSTAWCY;
TABLE DOSTAWCA;
      INCLUDE RECORD DOSTAWCA;
        DOMAIN NAZWA-DOSTAWCY FROM NAZWA-DOSTAWCY;
        DOMAIN WARTOŚĆ-DOSTAW FROM WARTOŚĆ-DOSTAW.
```

Dzięki przedstawionemu w tablicy 8 opisowi odwzorowań można przenieść wszystkie informacje zawarte w sieciowym modelu danych do modelu relacyjnego. Nie ma natomiast możliwości uzupełnienia tych informacji. Jak wiemy, sieciowy model danych domu towarowego (por. pkt 2) nie zawierał wszystkich informacji przedstawionych w opisie pojęciowym modelu danych (por. rys. 15).

Omawiając relacyjny model danych, opisaliśmy, dzięki wprowadzeniu odpowiednich ograniczeń w stosunku do wartości niektórych atrybutów relacji, relacyjną strukturę danych domu

towarowego, będącego pełną reprezentacją odpowiedniego pojęciowego modelu danych. Jeżeli elementem systemu ZBD jest również pojęciowy model danych, jak w wypadku ANSI/SPARC lub modelu koegzystencji, takie różnice w możliwościach odwzorowania rzeczywistości między sieciowym a relacyjnym modelem danych mają duże znaczenie.

W wypadku systemu RODAN sieciowy model danych jest punktem wyjściowym do opisu relacyjnego modelu danych i możliwość przeniesienia wszystkich informacji jest wtedy zupełnie wystarczająca. Relacyjnym językiem bezpośredniego użytkownika jest w systemie RODAN język SEQUEL (*Structured English Query Language*) opracowany w dwóch wersjach składniowych i leksykalnych: polskiej i angielskiej. SEQUEL [14] był opracowany w ośrodku badawczym firmy IBM w Kalifornii, USA. Dalszy rozwój systemów ZBD realizujących model koegzystencji może mieć duże znaczenie dla systemów informatycznych opartych na rozproszonej bazie danych.

[14] Por. A. Szymański, *SEQUEL — Język zapytań dla systemu RODAN*, Warszawa 1980.

V. Fizyczny model danych

Z fizycznej struktury danych przyjętej w określonym systemie zarządzania bazą danych wyłania się zakres możliwości odwzorowania logicznej struktury danych na urządzeniach pamięci o dostępie bezpośrednim.

Możliwości techniczne tych urządzeń determinują rozwiązania przyjęte w zakresie fizycznej struktury danych. Taka sytuacja doprowadziła do powstania wielu typowych rozwiązań, przyjętych w większości obecnie eksploatowanych systemów zarządzania bazą danych.

Wybór odpowiednich fizycznych odwzorowań i, co się z tym wiąże, odpowiedniej fizycznej struktury danych ma istotny wpływ na parametry eksploatacyjne zastosowań systemów baz danych. Rozważając poszczególne elementy fizycznej struktury danych przyjmiemy pewien zbiór podstawowych pojęć. Podstawowe pojęcia fizycznej struktury danych wynikają z rozwiązań przyjętych w systemach ZBD oraz z rozwiązań technicznych, stosowanych w urządzeniach pamięci zewnętrznej i oprogramowania obsługi tych urządzeń, zrealizowanych w ramach systemu operacyjnego.

System Zarządzania Bazą Danych:

Obszar — nazwana część adresowalnego zakresu pamięci zewnętrznej przydzielonego dla bazy danych i zawierającego wystąpienia rekordów oraz powiązań między tymi rekordami.

Strona — element fizycznej struktury danych charakteryzujący się ciągłą adresacją w pamięci operacyjnej.

Rekord fizyczny — nazwany zbiór zera, jednej lub więcej danych i/lub grup danych.

Pole — jednostka pamięci służąca do przechowywania wartości. Wartość może być reprezentowana jako ciąg znaków, liczba, zmienna boolowska lub dana, która może być interpretowana jako wartość lub stanowić adres wartości.

Urządzenia pamięci zewnętrznej:

Wolumen — jednostka nośnika pamięci zewnętrznej zdolna do przechowywania wartości i umożliwiająca odczytanie tych wartości.

Cylinder — zakres nośnika pamięci zewnętrznej charakteryzujący się tym, że przenoszenie zapamiętanych w ramach jednego cylindra wartości jest znacznie szybsze niż przenoszenie wartości zapamiętanych na dwóch kolejnych cylindrach.

Ścieżka — jest jednostką podziału cylindra, umożliwiającą najszybsze przeniesienie zapamiętanych na niej informacji.

Blok — jednostka podziału ścieżki, umożliwiająca operację czytania/pisania bez konieczności wykonania tych operacji w stosunku do sąsiednich bloków, umieszczonych na tej samej ścieżce.

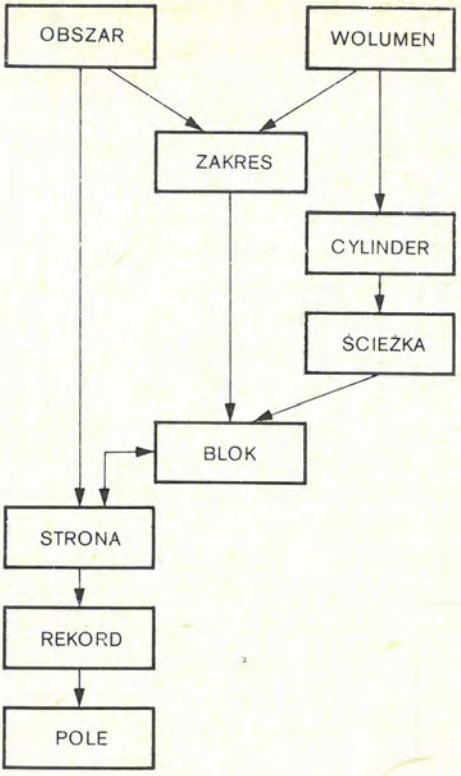
Zakres — zbiór bloków przydzielonych w ramach jednego wolumenu dla zapisania określonego obszaru lub części obszaru.

Z definicji tych wynika m.in., że pojęcie obszaru bazy danych odpowiada pojęciu pliku w systemie operacyjnym. Wzajemne powiązania podstawowych elementów fizycznej struktury danych przedstawiamy posługując się notacją Bachmana [1], na rysunku 32.

Istotnym problemem z punktu widzenia systemu zarządzania bazą danych jest odwzorowanie fizyczne poszczególnych elementów logicznego modelu danych. W wypadku sieciowego modelu danych mówimy o zasadach fizycznego odwzorowania rekordów, grup powtarzalnych, danych elementarnych oraz zbiorów strukturalnych. Podobnie w odniesieniu do relacyjnego modelu danych możemy mówić o fizycznym odwzorowaniu *n*-tki relacji, atrybutów relacji oraz o fizycznym wspomaganiu operacji na relacjach. Przykładem takiego wspomagania może być wykorzystanie elementów fizycznego modelu danych w łączeniu relacji.

Stosowane metody rozmieszczania i odwzorowania danych wynikają przede wszystkim z możliwości technicznych urządzeń

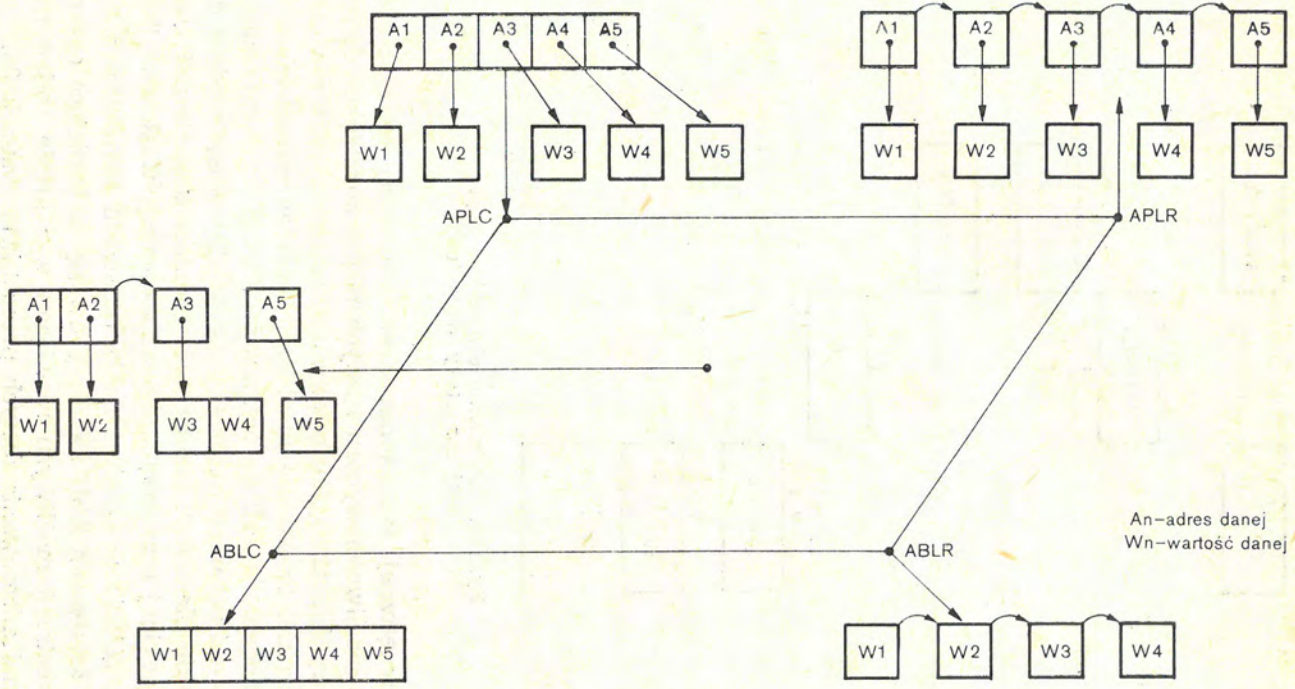
[1] Por. C. W. Bachman, *The Evolution of Storage Structures*, CACM, vol. 15, 1972, nr 7.



Rys. 32. Powiązanie elementów fizycznej struktury danych

pamięci zewnętrznej. Ważną natomiast własnością systemu ZBD jest zakres możliwości wyboru różnych metod konstrukcji fizycznej struktury danych. Jeżeli możliwości takiego wyboru są dostatecznie szerokie, to istnieje możliwość lepszego dopasowania fizycznego modelu danych do wymagań systemu informatycznego. Podstawowymi parametrami oceny jakości fizycznego modelu danych są takie elementy, jak zajętość pamięci zewnętrznej i pamięci operacyjnej oraz czas wykonania operacji wejścia-wyjścia.

Zanim przejdziemy do omówienia typowych rozwiązań stosowanych w systemach ZBD zwróćmy uwagę na istniejące ograniczenia. Przestrzeń możliwych rozwiązań w zakresie metod rozmieszczania i adresowania danych przedstawia rysunek 33.



Rys. 33. Zakres rozwiązań w dziedzinie metod adresowania i rozmieszczania danych

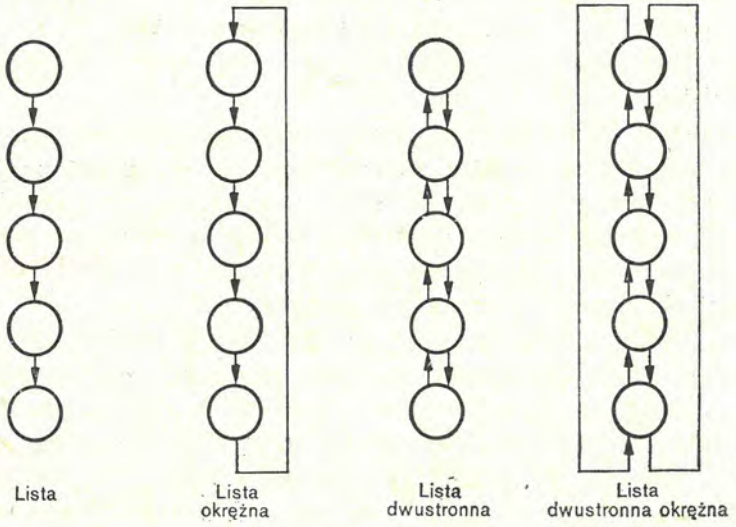
Urządzenie pamięci zewnętrznej możemy traktować jako pewien ciąg komórek o ustalonych adresach, przy czym w każdej komórce możemy przechowywać jedną wartość. Wartością tą może być wartość danej lub adres innej komórki z tego samego przedziału adresowego. Na przedstawionym schemacie takie wartości danych oznaczono literą W, a adresy komórek literą A. Przestrzeń możliwych rozwiązań jest wyznaczona przez cztery punkty połączone liniami prostymi, reprezentujące dwie podstawowe techniki adresowania: bezpośrednie i pośrednie, oraz dwie podstawowe techniki rozmieszczania danych, lista ciągła oraz lista rozproszona. Adresowanie bezpośrednie polega na pobieraniu wartości danej z komórki, w której jest ona przechowywana na podstawie adresu tej komórki.

Natomiast adresowanie pośrednie wymaga pobrania innej komórki pamięci zawierającej adres komórki, w której znajduje się poszukiwana wartość. Techniki pośredniego adresowania są szeroko wykorzystywane w mechanizmach adresowania pamięci operacyjnej, a w wypadku pamięci zewnętrznej są podstawą dla wszystkich stosowanych metod indeksowania.

Ciąg logicznie powiązanych wartości danych lub ciąg adresów komórek zawierających te wartości może być rozmieszczony w komórkach opatrzonych kolejno po sobie następującymi adresami lub w komórkach o dowolnych adresach z dostępnego przedziału adresów. W pierwszym wypadku mamy do czynienia z listą ciągłą, a w drugim — z listą rozproszoną. Podstawowe metody adresowania i rozmieszczania danych wyznaczające przestrzeń możliwych rozwiązań już przedstawiono (por. rys. 33). Wszystkie dotychczas stosowane w systemach ZBD metody adresowania i rozmieszczania danych mieszczą się w przedstawionej przestrzeni możliwych rozwiązań. W wypadku adresowania pośredniego może naturalnie istnieć hierarchia adresów o dowolnej ilości poziomów.

Istotną zmianę w tych rozwiązaniach mogą przynieść dopiero urządzenia pamięci, pozwalające na wybieranie odpowiednich komórek nie na podstawie ich adresów, lecz na przechowywanych w nich wartościach danych. Takie techniki adresowania nazywamy adresowaniem asocjacyjnym lub adresowaniem na podstawie zawartości. Powszechnie przyjętym odwzorowaniem rekordu

(lub n-tki relacji) jest ciągła lista zawartych w nim pól. W wypadku występowania powtarzalnych grup takich pól, można zastosować podział rekordu na części, będące elementami listy rozproszonej. Najczęściej posługujemy się wtedy adresem pierwszego pola rekordu jako adresem całego rekordu fizycznego. W razie podziału rekordu na elementy listy rozproszonej muszą być tworzone odpowiednie pola, zawierające adresy następnych elementów listy. Na rysunku 34 przedstawiamy podstawowe typy listy rozproszonej.

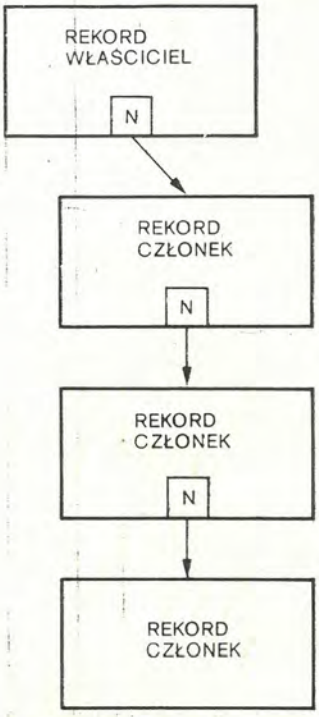


Rys. 34. Typy listy rozproszonej

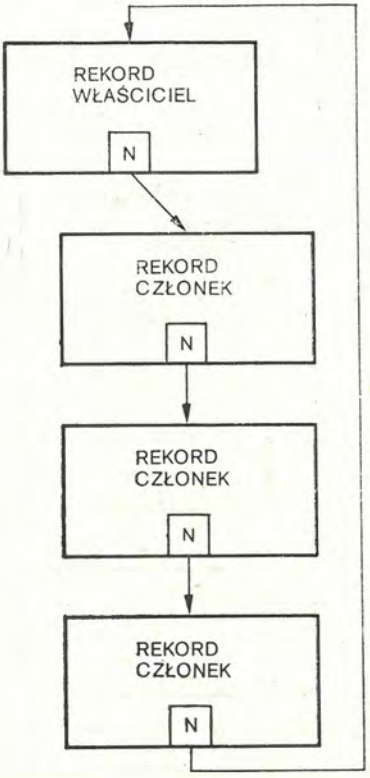
Tworzenie złożonych struktur (listy rozproszone, mechanizmy pośredniego adresowania), których węzłami są rekordy, wymaga przyjęcia systemu adresacji polegającego na utrzymaniu stałego adresu rekordu w bazie danych. Adres rekordu jest wyznaczany w momencie zapisania nowego rekordu w bazie danych i pozostaje bez zmian przez cały okres przechowywania rekordu. Taki adres rekordu będziemy nazywać kluczem bazy danych. Jedynym wypadkiem, w którym klucz bazy danych rekordu może być zmieniony, jest reorganizacja fizycznej struktury danych. Rekord fizyczny jest podstawowym elementem konstrukcyjnym, we wszystkich typach struktur występujących w ramach fizycz-

nej struktury danych. Występuje on jako odwzorowanie rekordu logicznego, jako element odwzorowania powiązania rekordów, jak również jako element mechanizmu adresowania pośredniego (indeks, tablica łączników adresowych itp.).

Podstawowym problemem odwzorowania logicznej struktury danych jest odwzorowanie powiązania rekordów typu 1 : N. Problem ten występuje zarówno w sieciowej, jak i relacyjnej strukturze danych. Sposób odwzorowania powiązań typu 1 : N ma istotne znaczenie dla parametrów czasowych, związanych z wykorzystaniem ścieżki selekcji, która zawiera te powiązania. Rozważmy kilka możliwych metod odwzorowania zbioru strukturalnego. Utrzymanie odpowiednich powiązań wymaga zastosowania



Rys. 35. Lista



Rys. 36. Lista okrężna

łączników adresowych zawierających klucze bazy danych odpowiednich rekordów. Rozważane metody odwzorowania wymagają utrzymania następujących typów łączników adresowych:

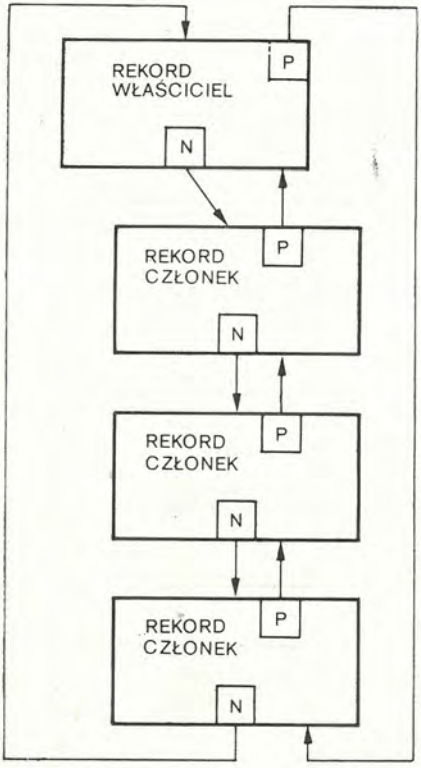
N — zawiera klucz bazy danych następnego rekordu,

P — zawiera klucz bazy danych poprzedniego rekordu,

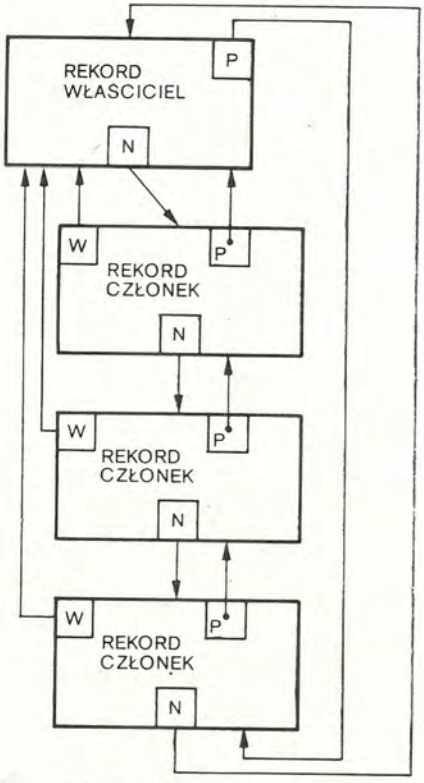
W — zawiera klucz bazy danych rekordu właściciela.

Metody odwzorowania zbioru strukturalnego obejmują następujące typy elementów fizycznej struktury danych:

Lista. Wszystkie rekordy łącznie z rekordem-właścicielem (z wyjątkiem ostatniego rekordu listy) zawierają łącznik adresowy do następnego rekordu (por. rys. 35; typy łączników adresowych — N).



Rys. 37. Lista okrężna dwustronna



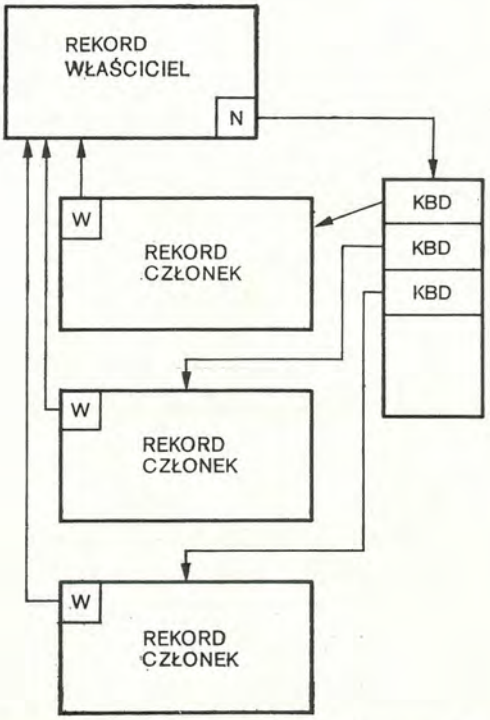
Rys 38. Lista okrężna dwustronna z łącznikiem adresowym do właściciela

Lista okrężna. Wszystkie rekordy łącznie z rekordem-właścicielem zawierają łącznik adresowy do następnego rekordu, przy czym rekordem następnym w stosunku do ostatniego rekordu listy jest właściciel (por. rys. 36; typy łączników adresowych — N).

Lista okrężna dwustronna. Wszystkie rekordy zawierają łączniki adresowe do następnego i poprzedniego rekordu (por. rys. 37; typy łączników adresowych — N, P).

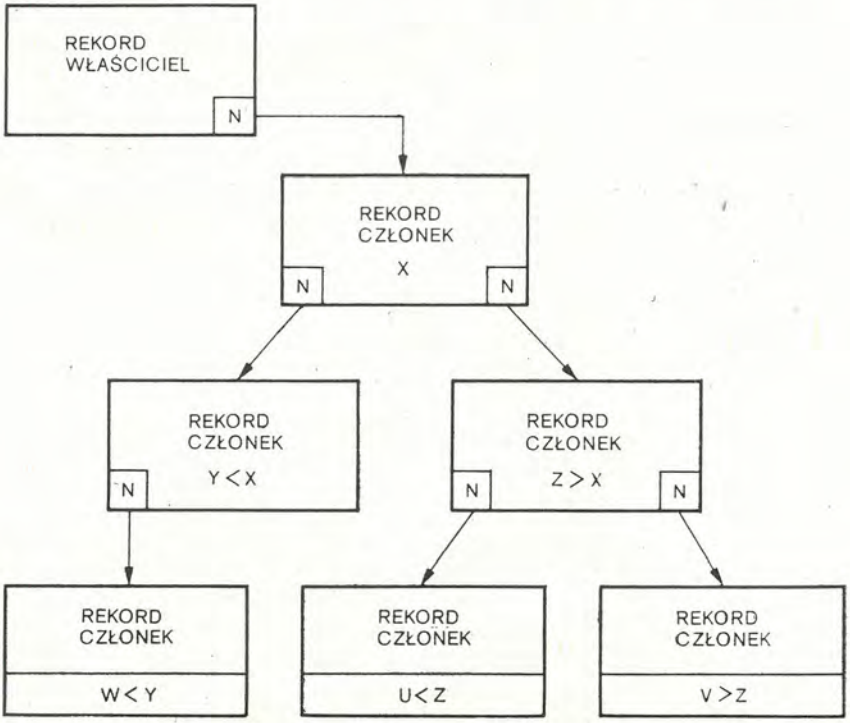
Lista okrężna dwustronna z łącznikiem adresowym do właściciela. Wszystkie rekordy członkowskie zawierają dodatkowo łącznik adresowy do rekordu-właściciela (por. rys. 38; typy łączników adresowych — N, P, W).

Tablica łączników adresowych danych. Rekord-właściciel zawiera łącznik adresowy do tablicy obejmującej klucze bazy danych wszystkich rekordów członkowskich tego zbioru strukturalnego (por. rys. 39).



Rys. 39. Tablice łączników adresowych

Drzewo binarne. Każdy rekord-członek zawiera łączniki adresowe do dwóch innych rekordów-członków tego powiązania. Lewy łącznik adresowy wskazuje na rekord zawierający niższą wartość danej elementarnej przyjętej jako kryterium logicznego uporządkowania zbioru strukturalnego (klucz sortowania), prawy na rekord zawierający wyższą wartość (por. rys. 40).



Rys. 40. Drzewo binarne

Wybór sposobu fizycznego odwzorowania zbioru strukturalnego w fizycznej strukturze danych wynika z przewidywanego sposobu wykorzystania go przez algorytmy procesów użytkowych. Rozważmy zbiór typowych operacji na zbiorze strukturalnym dzieląc go na operacje selekcji i aktualizacji.

Operacje selekcji:

Znajdź pierwszy (FF) — wybierz pierwszy rekord zbioru strukturalnego, rozpoczynając selekcję od rekordu-właściciela.

Znajdź ostatni (FL) — wybierz ostatni rekord zbioru strukturalnego, rozpoczynając selekcję od rekordu-właściciela.

Znajdź następny (FN) — wybierz następny rekord zbioru strukturalnego, rozpoczynając selekcję od dowolnego rekordu.

Znajdź poprzedni (FP) — wybierz poprzedni rekord zbioru strukturalnego, rozpoczynając selekcję od dowolnego rekordu.

Znajdź właściciela (FO) — wybierz rekord-właściciela zbioru strukturalnego, rozpoczynając selekcję od dowolnego rekordu.

Znajdź według klucza (FK) — wybierz rekord-członka zbioru strukturalnego na podstawie wartości klucza sortowania, rozpoczynając od dowolnego rekordu.

Operacje aktualizacji zbioru strukturalnego:

Włącz jako pierwszy (IF) — włącz rekord-członek jako pierwszy rekord zbioru strukturalnego.

Włącz jako ostatni (IL) — włącz rekord-członek jako ostatni rekord zbioru strukturalnego.

Włącz według klucza (IK) — włącz rekord-członek do zbioru strukturalnego, utrzymując jego porządek według przyjętego klucza sortowania.

Wyłącz rekord (R) — wyłącz rekord-członek ze zbioru strukturalnego.

Rozpatrywana operacja włączenia rekordu do zbioru strukturalnego składa się z dwóch faz:

1) wybieranie miejsca włączenia rekordu (odpowiednik operacji selekcji),

2) aktualizacja odpowiednich łączników adresowych.

Przyjęto następujące kryteria oceny efektywności wykonywanej operacji:

Bardzo dobra (BD) — nie wymaga dostępu do żadnych pośrednich rekordów zbioru strukturalnego.

Dobra (D) — wymaga dostępu do pośrednich rekordów, przy czym liczba dostępów nie jest wprost proporcjonalna do liczby rekordów, uczestniczących w zbiorze strukturalnym.

Średnia (S) — liczba wymaganych dostępów jest mniejsza od $\frac{N}{2}$, gdzie N jest liczbą rekordów uczestniczących w zbiorze strukturalnym.

Zła (Z) — liczba wymaganych dostępów do pośrednich rekordów jest większa od $\frac{N}{2}$.

Ocena efektywności operacji na poszczególnych typach odwzorowań zbioru strukturalnego jest zawarta w tablicy 9.

Tablica 9

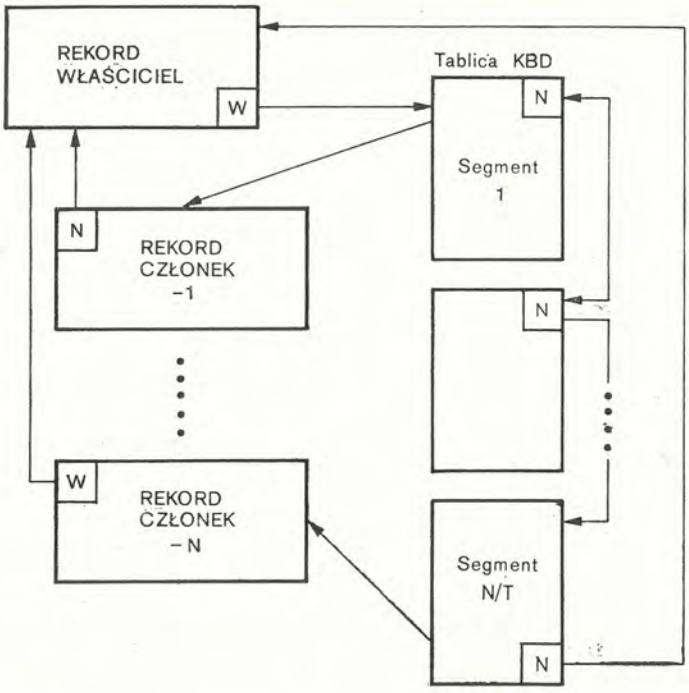
Ocena efektywności operacji na poszczególnych typach odwzorowania zbioru strukturalnego

Typ odwzorowania	Selekcja						Aktualizacja			
	FF	FL	FN	FP	FK	FO	IF	IL	IK	R
Lista	DB	Z	BD	—	—	—	BD	Z	—	BD
Lista okrężna z łącznikami adresowymi typu:										
N	BD	Z	BD	Z	Z	Z	BD	Z	Z	BD
N, P	BD	BD	BD	BD	S	Z	D	D	S	D
B, P, O	BD	BD	BD	BD	S	BD	D	D	S	D
Tablica kluczy bazy danych	D	D	D	D	D	BD	D	D	D	D
Drzewo binarne	S	S	BD	Z	D	—	Z	Z	Z	Z

BD — bardzo dobra
D — dobra
S — średnia
Z — zła

W wypadku operacji włączenia wzięto także pod uwagę liczbę aktualizowanych łączników adresowych, konieczność aktualizacji pośrednich elementów (tablica kluczy bazy danych) oraz konieczność wykorzystania specjalnych algorytmów (np. algorytm równoważenia drzewa binarnego).

Stosunkowo wysoka ocena wyszukiwana według klucza sortowania w razie odwzorowania zbioru strukturalnego jako tablicy kluczy bazy danych oraz drzewa binarnego wynika z możliwości zastosowania w obu tych wypadkach algorytmu cięcia binarnego. W wypadku drzewa binarnego ten sposób selekcji wynika bezpośrednio z organizacji struktury. Algorytm cięcia binarnego wymaga wykonania $\log_2 N + 1$ dostępów do rekordów, gdzie N jest

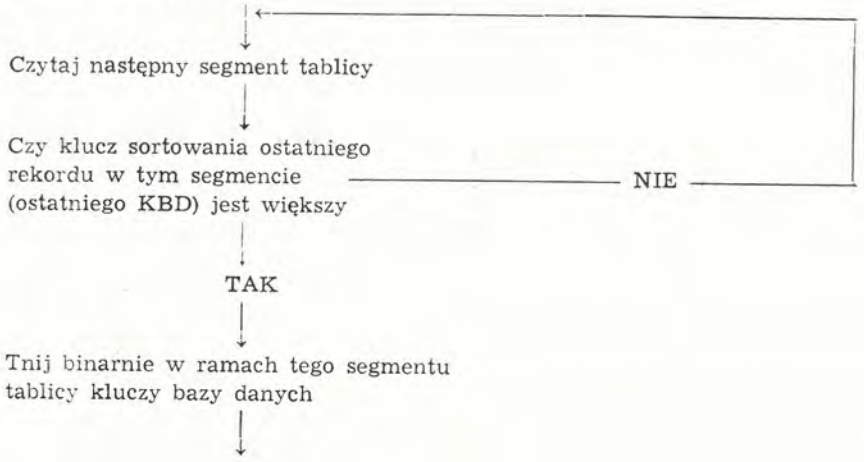


Rys. 41. Konstrukcje tablicy łączników adresowych

liczbą rekordów członków zbioru strukturalnego. W wypadku drzewa binarnego taka liczba dostępów pozwala na dokonanie selekcji pod warunkiem, że drzewo binarne jest zrównoważone.

Podobny wynik można uzyskać w wypadku tablicy łączników adresowych jedynie wtedy, gdy cała tablica zostaje pobrana do pamięci operacyjnej. Ponieważ rozmiary tablicy łączników adresowych mogą być stosunkowo duże (N×długość łącznika adresowego) jest ona zwykle konstruowana w postaci okrężnej listy segmentów (por. rys. 41). Liczba segmentów tablicy łączników adresowych bazy danych w ramach powiązania rekordów jest równa $\frac{N}{T}$, gdzie T jest liczbą kluczy bazy danych, przechowywanych w jednym segmencie tablicy. W tej sytuacji średnia ilość dostępów wymaganych dla odszukania rekordu według wartości klucza sortowania (razem z dostępami do segmentów

tablicy) wyniesie $D = \frac{N}{T} + \log_2 T + 1$. Algorytm selekcji działa w następujący sposób:



Omówione już elementy fizycznej struktury danych są podstawowym mechanizmem odwzorowania ścieżek selekcji, w ramach określonych w logicznej strukturze danych. Istotnym problemem z punktu widzenia efektywnej obsługi ścieżek selekcji, określonych w logicznej strukturze danych, jest minimalizacja liczby dostępów do rekordów w bazie danych. Ciąg elementów fizycznej struktury danych, do których jest wykonywany dostęp w ramach obsługi określonej ścieżki selekcji, nazywamy ścieżką dostępu.

Problemy adresowania rekordów w ramach ścieżki dostępu obejmują wyznaczanie pierwszego elementu ścieżki dostępu, czyli tzw. „punktu wejścia" do struktury danych oraz wybieranie następnych elementów w ramach istniejącej struktury powiązań. Adresowanie rekordu spełniającego rolę początku ścieżki selekcji jest realizowane przez adresowanie bezpośrednie lub przy wykorzystaniu różnych technik adresowania pośredniego. Podstawowym mankamentem tej pierwszej techniki adresowania jest konieczność przechowywania adresów rekordów fizycznych w oprogramowaniu użytkowym lub w pomocniczych plikach danych. Taka sytuacja w dużym stopniu utrudnia fizyczną reorganizację danych i może stać się istotnym ograniczeniem

eksploatacyjnym. Wybieranie pozostałych elementów ścieżki dostępu przebiega na podstawie uczestniczenia tych elementów w strukturach listowych. W tym wypadku istnieje możliwość skrócenia ścieżki dostępu (sekwencyjnego przeglądania listy) przez zastosowanie adresowania pośredniego. Hierarchię podstawowych technik adresowania danych przedstawiamy na rysunku 42.

Rys. 42. Podstawowe techniki adresowania

Kodowanie mieszające (*hash coding*) polega na znalezieniu takiej funkcji $f(k)$, która odwzorowałaby zbiór kluczy $K = \{k_1, k_2, ..., k_n\}$ na zbiór adresów $A = \{0, 1, 2, ..., m\}$ tak, by losowo wybrany ciąg kluczy $k_1, k_2, ..., k_1$, gdzie $1 \leqslant n$ przekształcony przez funkcję f na ciąg adresów $a_1, a_2, ..., a_1$ był mniej więcej równomiernie rozłożony w przedziale $[0, m]$.

Zastosowanie kodowania mieszającego w fizycznej strukturze danych wymaga utworzenia odpowiedniego zbioru adresów w ramach tej struktury. Wykorzystanie dotychczas omawianej struktury adresów opartej na jednoznacznej identyfikacji rekordu przez klucz bazy danych nie jest możliwe ze względu na fakt, że funkcje wykorzystywane w kodowaniu mieszającym nie są z zasady funkcjami różnowartościowymi. Oznacza to, że dla różnych wartości kluczy kodowania mieszającego zachodzi możliwość wygenerowania tego samego adresu. Rekordy, zawierające wartości kluczy powodujące wygenerowanie tego samego adresu

przez daną funkcję kodowania mieszającego (procedurę kodowania mieszającego), nazywamy synonimami.

Liczba takich synonimów zależy, dla określonego zbioru rekordów, od zakresu wartości kluczy, rozkładu tych wartości, zakresu dostępnych adresów oraz od algorytmu procedury kodowania mieszającego.

W praktyce stosuje się kilka podstawowych technik wyznaczania zbioru adresów dla procedury kodowania mieszającego i rozwiązania problemu synonimów. Jedną z takich technik jest podział całego obszaru dostępnej pamięci zewnętrznej na przedziały o stałej długości tak, aby adresy tych przedziałów pamięci stanowiły zbiór adresów dla procedury kodowania mieszającego. Ta technika może być stosowana w wypadku równomiernego i znanego *a priori* rozkładu liczby synonimów w poszczególnych przedziałach. W wypadku nierównomiernego rozkładu będziemy mieli do czynienia ze słabym wykorzystaniem jednych przedziałów oraz przepełnieniem u innych. Ponieważ trudno jest w praktyce tak dobierać algorytm procedury kodowania mieszającego, by przy danym rozkładzie wartości kluczy generować równomierny rozkład liczby synonimów. Wprowadzono nadmiarowe przedziały pamięci wykorzystywane w wypadku, gdy liczba synonimów przekracza liczbę rekordów, możliwą do zapamiętania w przedziale pamięci o adresie generalnym przez procedury kodowania mieszającego.

Inną techniką jest podział całego obszaru pamięci na przedziały o zmiennej długości tak, by adresy tych przedziałów pamięci stanowiły zbiór adresów dla procedury kodowania mieszającego. Najczęściej stosowaną implementacją takich przedziałów o zmiennej długości są listy synonimów kodowania mieszającego, przy czym adres początku takiej liczby jest adresem z przedziału adresów, wykorzystywanych przez procedurę kodowania mieszającego. Problem budowy algorytmów procedur kodowania mieszającego oraz wybór tych algorytmów z punktu widzenia charakterystyki zbioru wartości kluczy został szeroko omówiony w literaturze [2] i wykracza poza zakres pracy.

[2] Por. Y. Lum i in., *Key to Adress Transformation Techniques: a Fundamental Performance Study on Large Existing Formatted Files*, CACM, vol. 14, 4, 1971; J. Martin, *Computer Date Base Organization*, Prentice Hall Ed., New York 1975.

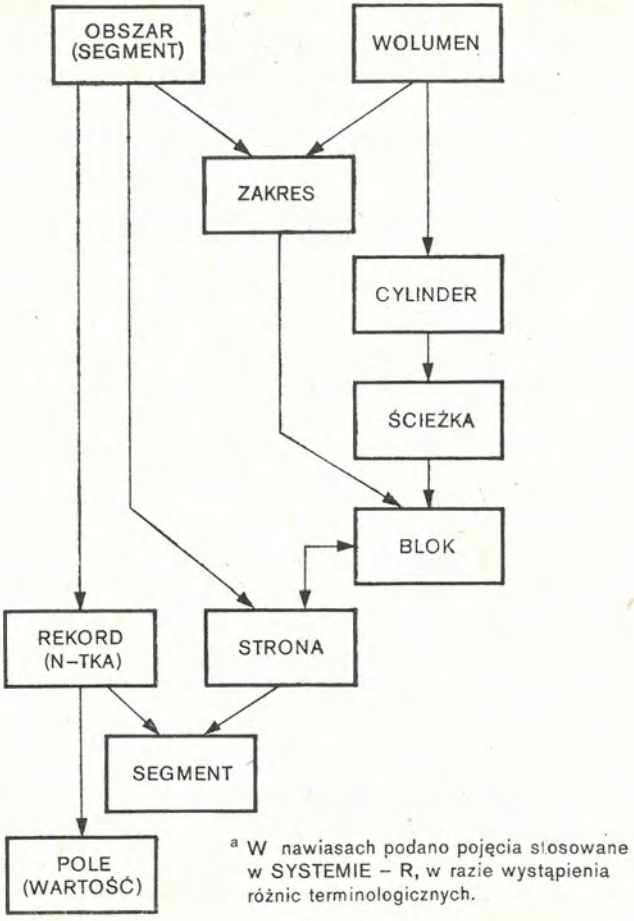
Indeksowanie jest wykorzystywane dla wybierania jednego rekordu lub zbioru rekordów na podstawie wartości danych elementarnych, występujących w tym typie rekordu. W wypadku wybierania jednego wystąpienia rekordu stosuje się techniki oparte na pojedynczym kluczu wyszukiwania (wartość danej elementarnej lub konkatenacja wartości danych elementarnych), przy założeniu jednoznacznych wartości klucza wyszukiwania lub przynajmniej jego dużej wartości identyfikacyjnej (niewielka liczba synonimów ze względu na wartość tego klucza wyszukiwania).

Podstawowymi technikami indeksowania służącego do wybierania jednego wystąpienia rekordu są metoda indeksowo-sekwencyjna oraz metoda drzewa binarnego. W wypadku wyszukiwania na podstawie wielu kluczy (zwykle o małej wartości identyfikacyjnej) stosuje się techniki oparte na zasadzie zbiorów odwróconych.

W praktyce stosowane techniki indeksowania są połączeniem obu typów rozwiązań. Dobrą ilustracją możliwych rozwiązań w zakresie fizycznej struktury danych są przykłady poszczególnych elementów fizycznego modelu danych w systemie ZBD RODAN oraz w relacyjnym systemie ZBD SYSTEM-R. SYSTEM-R był zrealizowany w ośrodku badawczym firmy IBM San Jose w Kalifornii, USA.

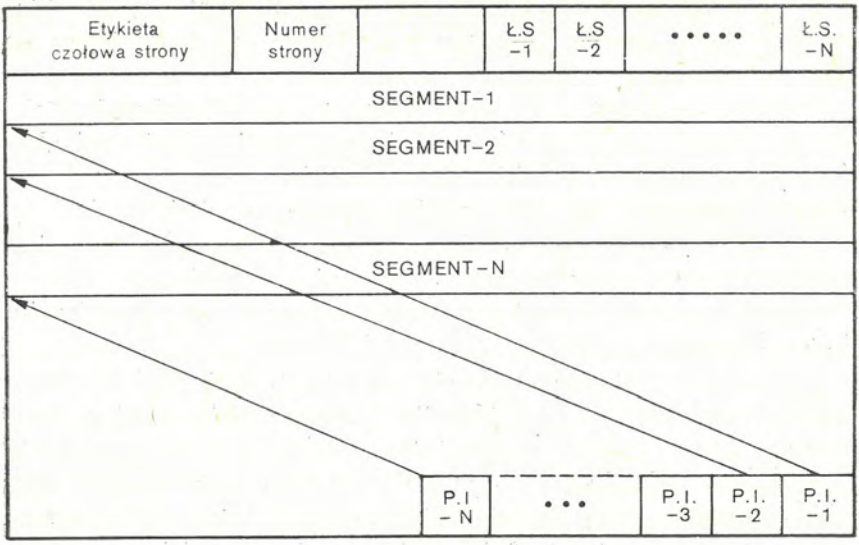
Podobieństwo rozwiązań przyjętych w obu tych systemach potwierdza tezę, że organizacja fizycznej struktury danych wynika z możliwości współczesnych urządzeń pamięci zewnętrznej, a nie z klasy logicznej struktury danych przyjętej w systemie zarządzania bazą danych. Powiązania elementów fizycznej struktury danych, występujące w obu rozpatrywanych systemach ZBD, przedstawiamy na rysunku 43.

Jedyną różnicą w stosunku do typowej dla współczesnych systemów ZBD fizycznej struktury danych jest związek typu N : M między stroną a rekordem. Oznacza to, że jeden rekord może być zapisany na więcej niż jednej stronie. Taka sytuacja występuje m.in. wtedy, kiedy aktualizacja rekordu zwiększa jego długość, a brak jest miejsca na stronie, na której ten rekord jest zapisany. Część rekordu jest przenoszona na inną stronę (jak najbliżej), a klucz bazy danych pozostaje bez zmian.



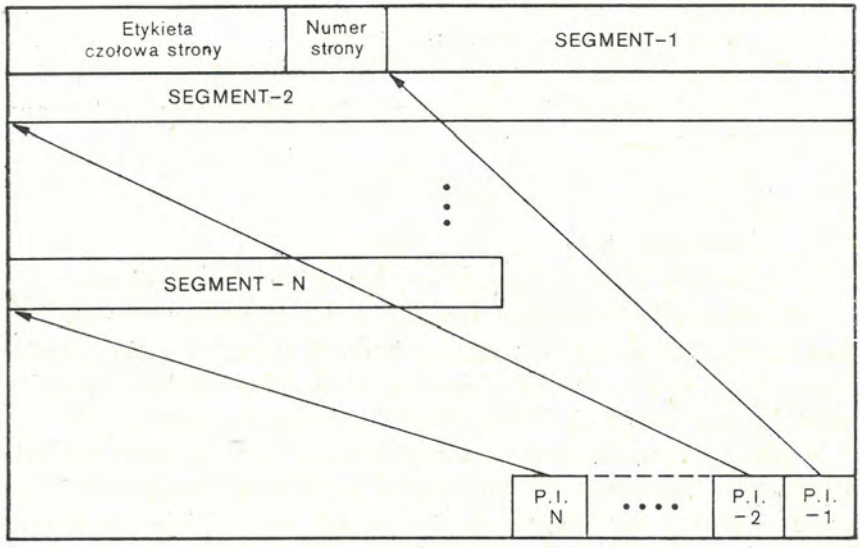
Rys. 43. Powiązania elementów fizycznej struktury danych w systemach ZBD RODAN i SYSTEM-R

Ważna, z punktu widzenia zasad rozmieszczania i adresowania rekordów, jest przyjęta w obu systemach struktura strony (por. rys. 44). Tutaj również występuje duże podobieństwo rozwiązań (jedyną poważną różnicą są adresy łańcuchów synonimów utrzymywane w systemie RODAN), a zastosowane pośrednie adresowanie rekordów na stronie (miejsca indeksowe) daje dużą swobodę w przesuwaniu rekordów w ramach strony bez zmiany klucza bazy danych.



Ł.S – adres łańcucha synonimów kodowania mieszającego
P.I. – pozycja indeksowa

SYSTEM ZBD RODAN



SYSTEM-R

Rys. 44. Struktura strony w systemach ZBD RODAN i SYSTEM-R

W obu systemach klucz bazy danych (identyfikator n-tki) składa się z kodu obszaru, numeru strony oraz numeru miejsca na stronie.

Przyjęte rozwiązanie pozwala utrzymać zasadę niezmienności klucza bazy danych rekordu przy jednoczesnej pełnej swobodzie w zakresie aktualizacji zapisanych rekordów.

W obu systemach przyjęto podział rekordu na dwie części:

1) część organizacyjną zawierającą łączniki adresowe wynikające z uczestnictwa rekordu w powiązaniu rekordów,

2) część danych, której struktura wynika z opisu w Języku Opisu Danych.

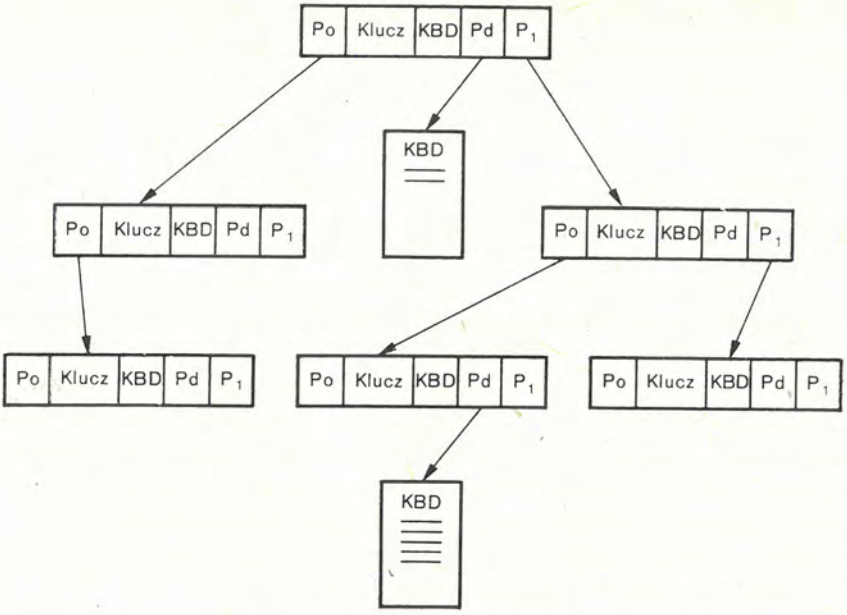
Zbiór strukturalny (powiązanie typu 1 : N) może być odwzorowany w systemie RODAN jako lista okrężna, lista okrężna dwustronna, lista okrężna z łącznikiem adresowym do właściciela oraz tablica kluczy bazy danych. W wypadku SYSTEMU-R operacja łączenia relacji (powiązanie typu 1 : N) jest wspomagana przez utrzymywanie zgodnie z deklaracją w opisie struktury danych okrężnej listy dwustronnej.

Zastosowane techniki adresowania obejmują:

— RODAN
  — adresowanie bezpośrednie,
  — kodowanie mieszające,
  — indeks zbioru strukturalnego stanowiący zmodyfikowane B-drzewo,
  — indeks w ramach typu rekordu,
— SYSTEM-R
— adresowanie bezpośrednie,
  — indeks relacji stanowiący zmodyfikowane B-drzewo.

Modyfikacja drzewa binarnego polega na utrzymywaniu list duplikatów kluczy (tj. list kluczy bazy danych rekordów zawierających te same wartości kluczy). Strukturę indeksu typu B-drzewa w systemie RODAN przedstawiamy na rysunku 45.

W obu systemach węzeł indeksu jest zapisywany jako rekord, a odpowiednie węzły są wiązane w listy dwustronne. Zastosowana technika zawiera elementy typowe dla zbiorów odwróconych (utrzymywanie listy adresów według wartości). Interesującym rozwiązaniem z punktu widzenia zarządzania fizyczną strukturą danych jest zastosowany w obu systemach algorytm



Rys. 45. Struktura indeksu typu B-drzewa w systemie RODAN

dynamicznego odzyskiwania nieużytków. W momencie wykonywania operacji skreślenia rekordu (n-tki) z bazy danych, zajmowane miejsce jest natychmiast włączane do puli wolnej pamięci, a zwolniony klucz bazy danych może być przydzielony następnemu zapisywanemu rekordowi. Dynamiczne odzyskiwanie nieużytków może mieć duże znaczenie w wypadku obszarów bazy danych o dużej aktywności operacji skreślania i dopisywania rekordów.

VI. Język dostępu do danych

1. Język Manipulacji Danymi w sieciowym modelu danych

Omawiając sieciowy model danych Komitetu CODASYL zwróciliśmy uwagę, że ta klasa logicznej struktury danych jest przede wszystkim przeznaczona do wykorzystania przez użytkownika-programistę. Podstawowym językiem dostępu do danych dla tego logicznego modelu danych jest Język Manipulacji Danymi (JMD), zaprojektowany jako rozszerzenie możliwości konwencjonalnych języków programowania (COBOL, PL/1, FORTRAN).

Język programowania, w którym mogą być wykorzystane komendy JMD nazywamy bazowym językiem programowania *(host language)*. Obecnie dostępne systemy ZBD opracowane na podstawie raportów Komitetu CODASYL[1] obejmują COBOL i PL/1 jako bazowe języki programowania. Wszystkie możliwości takich języków programowania oraz zasady budowy oprogramowania użytkowego nie ulegają istotnym zmianom.

Ważnym efektem zastosowania JMD w powiązaniu z jego bazowym językiem programowania jest uzyskanie niezależności oprogramowania użytkowego od danych. Jest to przede wszystkim niezależność od fizycznego modelu danych, ponieważ komendy JMD działają jedynie na elementach logicznej struktury danych, oraz częściowa logiczna niezależność od danych, ponieważ wizja struktury danych programu użytkowego jest zwykle podzbiorem logicznej struktury danych całej bazy danych. Te możliwości powodują istotne uproszczenie prac nad projektem i opracowaniem oprogramowania użytkowego oraz znacznie zmniejszają nakłady na jego utrzymanie.

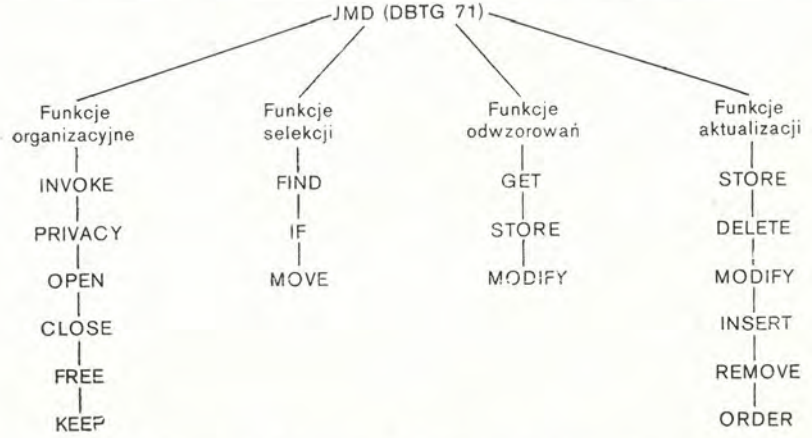
[1] Por. *CODASYL Data Base Task Group*, April 1971, *Report* ACM, New York 1972.

Czas projektowania i realizacji programu użytkowego wynosi zwykle około od 20 do 30% czasu realizacji programu o tych samych funkcjach wykonanego w technikach konwencjonalnych.

Raport DBTG 71 zawiera pełny opis składni oraz wiele zasad semantycznych Języka Manipulacji Danymi, będącego mechanizmem komunikacji między programem użytkowym napisanym w bazowym języku programowania a bazą danych. Zgodnie z zamierzeniami raportu proponowany Język Manipulacji Danymi jest niezależny od współczesnych języków programowania. Język Manipulacji Danymi jest zbiorem komend wykonujących funkcje użytkowe w czterech podstawowych grupach:

— funkcje organizacyjne,
— funkcje selekcji,
— funkcje odwzorowań,
— funkcje aktualizacji.

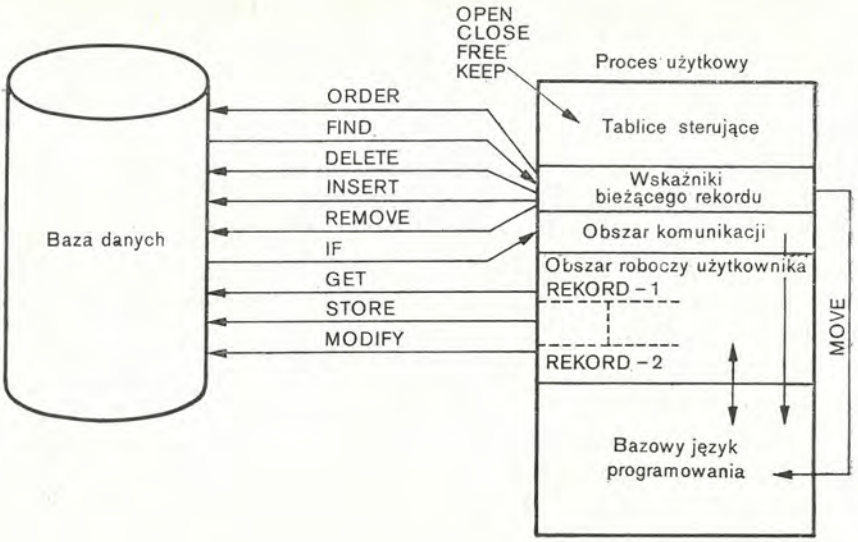
Strukturę Języka Manipulacji Danymi ilustruje rysunek 46.



Rys. 46. Struktura Języka Manipulacji Danymi (DBTG 71)

Komendy Języka Manipulacji Danymi mogą być rozmieszczone w dowolny sposób w programie użytkowym.

Komunikacja między programem użytkowym a bazą danych odbywa się za pośrednictwem obszaru roboczego użytkownika, a symboliczne adresowanie tego obszaru jest zapewnione przez automatycznie włączone do programu użytkowego deklaracje od-



Rys. 47. Język Manipulacji Danymi (DBTG 71) — przepływ danych i informacji sterujących

powiednich typów rekordów. Przepływ danych i informacji sterujących w ramach procesu użytkowego przedstawia rysunek 47.

Przedstawiony proces użytkowy jest podzielony na szereg obszarów, przy czym niektóre z nich są wykorzystywane wyłącznie przez system ZBD, a inne są dostępne również dla bazowego języka programowania.

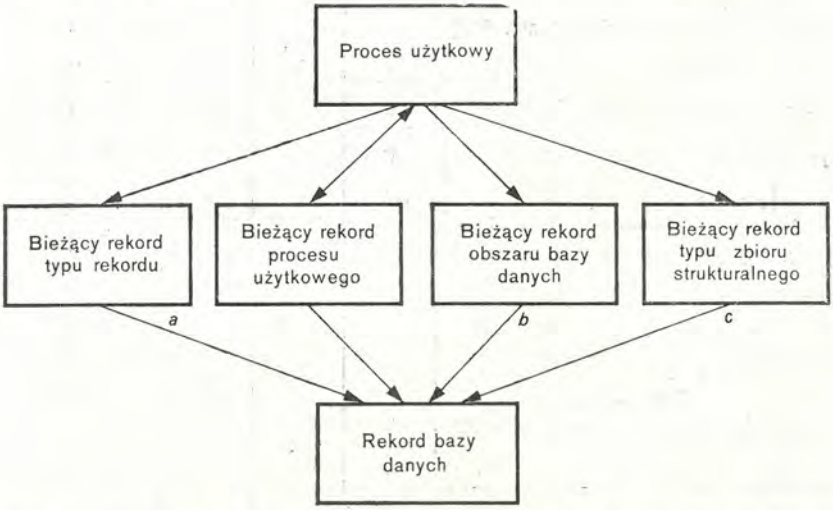
Tablice sterujące zawierają informacje o elementach logicznej i fizycznej struktury danych, objętych działaniem procesu użytkowego. Tu są m.in. określone zasady odwzorowania logicznej struktury danych w fizyczną strukturę danych. Poza tym w tablicach sterujących utrzymywane są informacje o stanach bazy danych, takich jak otwarcie obszaru bazy danych, blokada elementu logicznej struktury danych itp.

Obszar komunikacji służy do przekazywania do programu użytkowego informacji o wyniku wykonania komendy JMD; służy on także do sygnalizacji wszystkich błędów powstałych w toku wykonywania procesu użytkowego oraz przekazywania informacji o miejscach wystąpienia takich błędów.

Obszar roboczy użytkownika służy do przechowywania rekor-

dów pobieranych z bazy danych lub zapisywanych do bazy danych. Postać tych rekordów jest zgodna z przyjętą dla danego procesu strukturą danych użytkownika. Obszar roboczy użytkownika jest dostępny dla bazowego języka programowania przez adresy symboliczne.

Przedmiotem każdej funkcji użytkowej (z wyjątkiem funkcji organizacyjnych) wykonywanej przez komendy JMD jest zawsze określony rekord bazy danych, a realizacja komend jest oparta na pojęciu bieżącego rekordu. Dla każdego procesu użytkowego jest utrzymywany pewien zbiór wskaźników, określających bieżący rekord, tzn. biorący udział w przetwarzaniu dla poszczególnych typów elementów struktury danych. Wzajemne powiązania wskaźników bieżącego rekordu, utrzymywanych automatycznie dla poszczególnych typów elementów struktury danych, będącej zakresem działania procesu użytkowego a rekordem bazy danych, ilustruje schemat Bachmana (por. rys. 48).



Rys. 48. Schemat powiązań rekordu bazy danych ze wskaźnikami bieżącego rekordu poszczególnych typów elementów struktury danych

a Bieżącym rekordem określonego typu mogą być wszystkie rekordy, należące do tego typu.

b Bieżącym rekordem obszaru bazy danych mogą być wszystkie rekordy, zapisane w tym obszarze.

c Bieżącym rekordem typu zbioru strukturalnego mogą być wszystkie rekordy uczestniczące w jakimkolwiek zbiorze strukturalnym, należącym do tego zbioru.

Zasady wykorzystania wskaźników bieżącego rekordu przez komendy selekcji JMD zebrano w tablicy 10.

Tablica 10

Zasady wykorzystania wskaźników bieżącego rekordu w realizacji funkcji selekcji JMD

Funkcja selekcji	Wskaźniki bieżącego rekordu				Ścieżka selekcji
	procesu użytkowego	obszaru bazy danych	typu rekordu	typu zbioru strukturalnego	
1	2	3	4	5	6
FIND					
FORMAT 1. [nazwa-rekordu] USING identyfikator	u	u	u	u	
FORMAT 2. [OWNER IN nazwa-zbioru-str. OF] CURRENT OF nazwa-rekordu RECORD	u	u	w/u	u	
CURRENT OF nazwa-obszaru AREA	u	w/u	u	u	
CURRENT OF nazwa-zbioru-str. SET	u	u	u	w/u	
CURRENT OF proces użytkowy	w/u	u	u	u	
FORMAT 3. NEXT / PRIOR / FIRST / LAST / k. całk. / identyfikator [nazwa rekordu] RECORD OF nazwa-zbioru-str. SET	u	u	u	w/u	
FORMAT 3. NEXT / PRIOR / FIRST / LAST / k. całk. / identyfikator nazwa-obszaru AREA	u	w/u	u	u	
FORMAT 4. OWNER RECORD OF nazwa-zbioru str. SET	u	u	u	w/u	
FORMAT 5. [NEXT DUPLICATE WITHIN] nazwa-rekordu RECORD	w/u	u	u	u	

cd. tablicy 10

1	2	3	4	5	6
FORMAT 6. nazwa-rekordu VIA [CURRENT OF] nazwa-zbioru-str. [USING] wartość danej elementarnej, [wartość danej elementarnej]	u	u	u	w/u	X
FORMAT 7. NEXT DUPLICATE WITHIN nazwa-zbioru-str. USING wartość danej elementarnej [wartość danej elementarnej]	u	u	u	w/u	
IF					
nazwa-zbioru-str. SET NOT EMPTY	—	—	—	w	
RECORD NOT { MEMBER / OWNER } OF { nazwa-zbioru-str. / ANY } SET	w	—	—	—	
MOVE CURRENT OF STATUS FOR					
PROCES UŻYTKOWY	w	—	—	—	
nazwa-rekordu RECORD	—	—	w	—	
nazwa-obszaru AREA	—	w	—	—	
nazwa-zbioru-str. SET	—	—	—	w	
TO identyfikator					

u — ustawia wartość
w — wybiera według wartości

Pojęcie bieżącego rekordu jest szczególnie ważne w wypadku realizacji funkcji selekcji, ponieważ ustawienie odpowiednich wskaźników jest jedynym efektem komend JMD, wykonujących funkcje użytkowe należące do tej grupy. Bieżący rekord określonego typu elementu struktury danych jest punktem wyjścia dla wykonywanych w ramach komendy FIND wyrażeń selekcji rekordu. Takie wykorzystanie wskaźników bieżących rekordów stanowi podstawę „nawigacji" po strukturze danych. Wskaźniki bieżącego rekordu są utrzymywane automatycznie dla wszystkich

typów elementów struktury danych objętych zakresem działania (podschemat) procesu użytkowego.

Wartością wskaźnika bieżącego rekordu jest jednoznaczny identyfikator rekordu (klucz bazy danych) tworzony i utrzymywany automatycznie przez system ZBD.

Omówiliśmy mechanizm działania podstawowych komend JMD.

Podstawową zasadą selekcji (za pomocą komendy FIND) jest wyznaczenie punktu w strukturze danych, od którego ma być realizowany proces selekcji rekordu. Kolejno wykonywane komendy FIND wyznaczają ścieżkę selekcji poszukiwanego rekordu. Początkiem takiej ścieżki selekcji musi być rekord, którego selekcja nie wymaga wyznaczenia jego względnej pozycji w strukturze danych. Selekcji takiego rekordu dokonuje się przez format 1 (por. tabl. 10) komendy FIND, gdzie identyfikator, umieszczony w wyrażeniu selekcji rekordu, musi zawierać wartość klucza bazy danych szukanego rekordu, lub przez format 5 komendy FIND (bez frazy NEXT DUPLICATE WITHIN) na podstawie podanych przez proces użytkowy wartości danych elementarnych określonych jako klucze kodowania mieszającego. Oba te formaty komendy FIND wyznaczają „punkty wejścia" do struktury danych. Ciąg komend FIND wyznaczających ścieżkę selekcji może być zastąpiony deklaracją ścieżki selekcji w schemacie lub podschemacie struktury danych, przy czym pierwszy element takiej ścieżki musi być „punktem wejścia" do struktury danych. Automatyczna ścieżka selekcji jest wykorzystywana przez format 6 (bez frazy CURRENT OF) komendy FIND. Przyjęte zasady selekcji prowadzą do dużego uzależnienia algorytmu programu użytkowego od logicznej struktury danych wykorzystywanej przez ten program.

Program użytkowy zawierający algorytm „nawigacji" po strukturze danych ma metody dostępu do szukanego rekordu.

1. Może rozpocząć od początku bazy danych lub jakiegokolwiek znanego mu rekordu i sekwencyjnie wybierać kolejne „następne" rekordy aż do odszukania odpowiedniego rekordu lub osiągnięcia logicznego końca bazy danych.

2. Może wejść do bazy danych wybierając rekord na podstawie jego klucza bazy danych pozwalający na bezpośredni dostęp do fizycznego adresu rekordu.

3. Może wejść do bazy danych przez wartości danych elementarnych, będące podstawą wyznaczania fizycznego adresu rekordu (algorytm kodowania mieszającego lub indeks).

4. Może wybierać kolejno wszystkie rekordy o określonych wartościach danych elementarnych uczestniczące w dowolnym zbiorze strukturalnym.

5. Może wybierać kolejno wszystkie rekordy członkowskie, rozpoczynając od właściciela dowolnego zbioru strukturalnego.

6. Może wybierać poprzedni lub następny rekord w stosunku do rekordu członkowskiego, w ramach dowolnego zbioru strukturalnego.

7. Może wybierać właściciela zbioru strukturalnego na podstawie dowolnego rekordu członkowskiego.

Ten zbiór możliwości selekcji rekordów jest mocną podstawą wykonywania procesów użytkowych wyszukujących lub aktualizujących dowolnie złożoną strukturę danych.

W ramach dostępnych możliwości nawigacji po strukturze danych istnieją następujące typy funkcji selekcji:

B — bezpośrednia selekcja rekordu na podstawie wartości jego danych elementarnych lub wartości klucza bazy danych,

R→R — przejście od jednego rekordu do drugiego w ramach obszaru bazy danych,

C→C — przejście od jednego rekordu członkowskiego do drugiego w ramach tego samego zbioru strukturalnego,

W→C — przejście od rekordu-właściciela do rekordu członkowskiego zbioru strukturalnego,

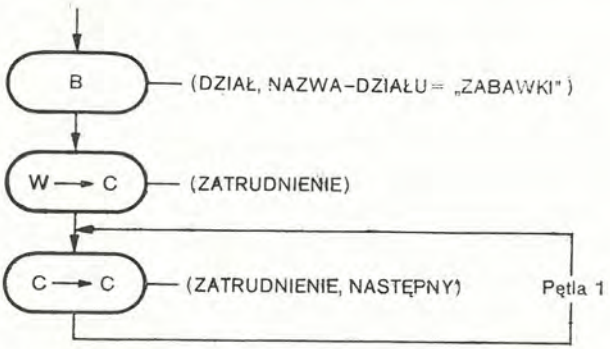
C→W — przejście od rekordu członkowskiego do rekordu-właściciela zbioru strukturalnego,

D→D — przejście od jednego rekordu zawierającego określone wartości danych elementarnych do drugiego rekordu zawierającego te same wartości (duplikat). Przejście może być wykonane w ramach typu rekordu lub typu zbioru strukturalnego.

Te funkcje selekcji mogą być kwalifikowane typem rekordu, typem zbioru strukturalnego, kierunkiem przejścia i wartościami danych elementarnych. Rozważmy kilka algorytmów nawigacyjnej selekcji rekordów, realizowanych na sieciowej strukturze danych (por. rozdz. III, baza danych domu towarowego). Konstruk-

cja przedstawionych algorytmów jest niezależna od przyjętego bazowego języka programowania.

Wynikiem działania każdej funkcji selekcji jest zawsze jeden rekord. Taka organizacja procesu selekcji wynika z możliwości bazowych języków programowania oraz zasad budowy obszaru roboczego użytkownika wykorzystywanego do przechowywania rekordów przeczytanych z bazy danych. Organizacja struktury sterującej programu użytkowego (iteracje, badania, skoki warunkowe) spoczywa na odpowiednich funkcjach bazowego języka programowania. Na rysunkach 49, 50, 51 przedstawiamy schematy trzech algorytmów wyszukiwania działających na podstawie przedstawionego w rozdziale IV (por. rys. 23) sieciowego modelu danych domu towarowego.



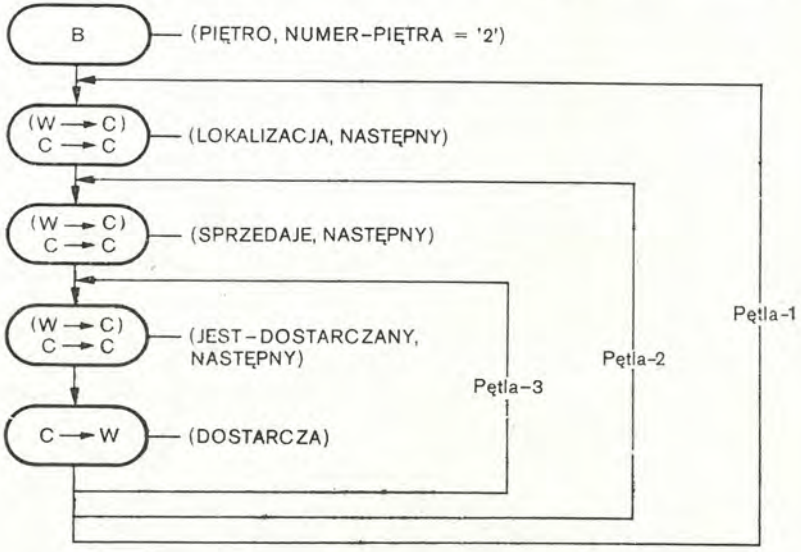
Rys. 49 Znajdź nazwiska wszystkich pracowników, którzy pracują w dziale zabawek

Przedstawione schematy algorytmów użytkowych (realizujących zapytania) zawierają podstawowe typy funkcji selekcji połączone strzałkami, które reprezentują drogę stosowania. Funkcje selekcji są kwalifikowane nazwami typów rekordów lub typów zbiorów strukturalnych, które są przedmiotem ich działania. W wypadkach, gdy funkcja selekcji wykorzystuje wartość danej elementarnej jako argumentu selekcji, wprowadzono odpowiednią kwalifikację. W każdym innym wypadku odpowiedzialne za porównanie właściwych wartości i dokonanie selekcji rekordu są fragmenty programu użytkowego, napisane w bazowym języku programowania.

Realizacja pierwszego zapytania (por. rys. 49) wymaga bezpo-

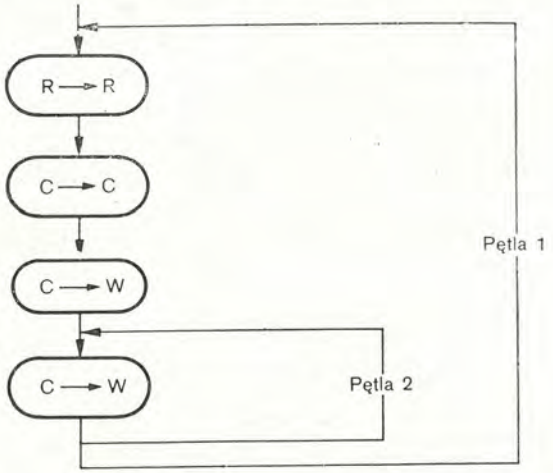
średniej selekcji odpowiedniego rekordu DZIAŁ, a następnie sekwencyjnego przejścia przez rekordy PRACOWNIK, występujące w zbiorze strukturalnym ZATRUDNIENIE, którego właścicielem jest wybrany rekord DZIAŁ. Wyszukiwane jest zakończenie w momencie dojścia do końca tego zbioru strukturalnego. Do organizacji pętli programowej, druku listy pracowników oraz badania warunku końca zbioru strukturalnego wykorzystuje się odpowiednie funkcje bazowego języka programowania. Zauważmy, że sekwencyjne przejście po zbiorze strukturalnym wymaga, w wypadku rozpoczęcia przejścia od właściciela, wykonania dwóch typów funkcji selekcji, tj. przejścia od rekordu-właściciela do rekordu-członka zbioru strukturalnego (W→C) oraz przejścia od jednego rekordu członkowskiego do drugiego w ramach tego samego zbioru strukturalnego (C→C). W wypadku sekwencyjnego przejścia po zbiorze strukturalnym oba te typy funkcji są równoważne. Fakt ten wykorzystamy w następnym przykładzie.

Drugi przykład (por. rys. 50) demonstruje dość skomplikowaną nawigację po sieciowej strukturze danych. Realizacja zapytania wymaga, po bezpośredniej selekcji rekordu PIĘTRO, sekwencyjne-



Rys. 50. Algorytm wyszukiwania. Znajdź wszystkie działy zlokalizowane na 2 piętrze, które sprzedają towar dostarczany przez dostawcę „xyz"

go przejścia po odpowiednim zbiorze strukturalnym LOKALIZACJA, a dla którego tak wybranego rekordu DZIAŁ — sekwencyjnego przejścia po odpowiednim zbiorze strukturalnym SPRZEDAJE. Dla każdego wybranego rekordu TOWAR jest przeszukiwany zbiór strukturalny JEST-DOSTARCZANY. Dla każdego rekordu DOSTAWA jest odszukiwany rekord DOSTAWCA jako właściciel zbioru strukturalnego DOSTARCZA. Realizacja tego algorytmu wymaga zorganizowania trzech pętli programowych, a warunkiem ograniczającym każdą z tych pętli jest koniec zbioru strukturalnego. Analizę schematu (por. rys. 51), przedstawiającego algorytm trzeciego zapytania, pozostawiamy Czytelnikowi.



Rys. 51. Algorytm wyszukiwania: Znajdź wszystkich dostawców dostarczających towar do działu, którego wartość sprzedaży przekracza „10 000 000"

Przedmiotem funkcji odwzorowań i aktualizacji Języka Manipulacji Danymi jest zawsze bieżący rekord procesu użytkowego (z wyjątkiem komendy ORDER), przy czym zakres oddziaływania aktualizacji może obejmować wiele rekordów i zbiorów strukturalnych. Odwzorowania między strukturą danych użytkownika a logiczną strukturą danych, tj. między obszarem danych w obszarze roboczym programu a obszarem danych w obszarach roboczych systemu ZBD są realizowane przez komendy GET, STORE i MODIFY. Komenda STORE powoduje zapisanie nowego re-

kordu do bazy danych (włącznie z przydziałem klucza bazy danych) i ustawienie odpowiednich wskaźników bieżącego rekordu. Przedmiotem komend GET i MODIFY jest zawsze bieżący rekord procesu użytkowego, przy czym komenda GET powoduje pobranie tego rekordu do obszaru roboczego programu, a komenda MODIFY — zamianę wartości danych elementarnych rekordu zapisanego w bazie danych na wartości danych elementarnych z obszaru roboczego użytkownika. W niektórych wypadkach komenda MODIFY powoduje przeniesienie rekordu z jednego zbioru strukturalnego do drugiego w ramach tego samego typu. Przeniesienie z jednego zbioru strukturalnego do drugiego w ramach tego samego typu jest wynikiem modyfikacji danych elementarnych, opisanych jako argumenty selekcji wystąpienia zbioru strukturalnego.

Włączania i wyłączania rekordu ze zbioru strukturalnego (lub wielu zbiorów strukturalnych) dokonuje się za pośrednictwem komend INSERT i REMOVE. Przedmiotem działania tych komend jest bieżący rekord procesu, a odpowiedni zbiór strukturalny jest wyznaczany przez bieżący rekord tego typu zbioru strukturalnego. Zmiana uporządkowania zbioru strukturalnego (trwała lub czasowa) jest wykonywana za pomocą komendy ORDER, przy czym odpowiedni zbiór strukturalny jest wybierany tak samo, jak w wypadku komend INSERT i REMOVE. Skreślanie rekordu z bazy danych (oraz w niektórych wypadkach powiązanych z nim innych rekordów) następuje przez komendę DELETE.

Komenda INVOKE nie wykonuje akcji na strukturze danych. Jest ona rozpisywana na odpowiednie deklaracje rekordów (zgodnie z podschematem struktury danych), umożliwiające adresowanie obszaru roboczego programu. Dostęp do obszarów bazy danych wymaga odpowiedniego otwarcia tych obszarów. Funkcję otwierania obszarów z jednoczesną kontrolą uprawnień procesu użytkowego wykonuje komenda OPEN. Po wykorzystaniu obszarów bazy danych musi nastąpić ich zwolnienie. Funkcja ta jest realizowana przez komendę CLOSE.

Komendy FREE/KEEP są mechanizmami ochrony równolegle realizowanych procesów użytkowych przed utratą integralności danych, spowodowaną interferencją tych procesów. Problemy synchronizacji procesów i ochrony integralności danych omówimy dalej.

2. Języki dostępu do danych w relacyjnym modelu danych

a. Relacyjny Język Manipulacji Danymi oparty na rachunku predykatów

Klasycznym przykładem relacyjnego języka dostępu do danych opartego na rachunku predykatów jest, opracowany w 1971 r. w ośrodku badawczym firmy IBM w San Jose, USA, język relacyjny ALPHA. Twórcą tego języka jest E. F. Codd [2]. Język ALPHA działa na znormalizowanej strukturze relacyjnej, a jego funkcje obejmują wyszukiwanie i aktualizację danych. Jest on wykorzystywany przez użytkowników bezpośrednich, jak również może być Językiem Manipulacji Danymi rozszerzającym możliwości bazowych języków programowania (COBOL, PL/1, FORTRAN).

Język ALPHA zawiera:

— wyrażenia selekcji definiowania na podstawie formuły rachunku predykatów,

— funkcje użytkowe, które obejmują:

— pobieranie wartości lub zbioru wartości,

— zmianę wartości lub zbioru wartości,

— dopisywanie elementu (n-tki) lub zbioru elementów do relacji,

— wykreślenie elementu (n-tki) lub zbioru elementów z relacji,

— deklarowanie relacji i jej dziedzin w celu włączenia do zbioru relacji bazy danych,

— skreślenie całej relacji bazy danych,

— możliwość ograniczania zakresu (zdanie RANGE) działania procesu użytkowego na strukturze danych.

Wyrażenia selekcji języka ALPHA spełniają następujące zasady syntaktyczne:

⟨wyrażenie selekcji⟩ : : = ⟨nazwa funkcji użytkowej⟩ ⟨nazwa obszaru roboczego⟩ ⟨lista docelowa⟩[⟨:⟩⟨kwalifikacja⟩]

⟨nazwa funkcji użytkowej⟩ : = GET | HOLD | OPEN GET | OPEN UPDATE | DELETE

[2] Por. E. F. Codd, *A Data Base Sublanguage Founded on the Relational Calculus*, Proc. Workshop on Date Description, Access and Control SIGFIDET 1971, ACM, New York 1971.

⟨lista docelowa⟩ : : = ⟨nazwa⟩ [⟨nazwa⟩] ...
⟨kwalifikacja⟩ : : = predykat złożony
⟨predykat złożony⟩ : : = ⟨predykat atomiczny⟩ ⟨spójnik zdaniowy⟩ ⟨predykat atomiczny⟩
⟨spójnik zdaniowy⟩ : : = $\vee$ | $\wedge$ | $\Rightarrow$ | $\Leftrightarrow$
⟨predykat atomiczny⟩ : : = ⟨nazwa⟩ ⟨symbol relacji⟩ ⟨nazwa⟩ | ⟨literał⟩
⟨nazwa⟩ : : = nazwa relacji. nazwa atrybutu | zmienna relacyjna. nazwa atrybutu | nazwa relacji | zmienna relacyjna
⟨symbol relacji⟩ : : = $=$ | $\neq$ | $>$ | $<$ | $\geqslant$ | $\leqslant$
⟨literał⟩ : : = ⟨ciąg znaków⟩
Ponadto w konstrukcji predykatów wykorzystuje się:
⟨kwantyfikator⟩ : : = $\wedge$ | $\vee$
⟨symbol negacji⟩ : : = $\sim$
⟨nawiasy⟩ : : = (|)

Lista docelowa wyrażona jako lista nazw jest definicją nowej relacji, która ma zostać utworzona we wskazanym obszarze roboczym na skutek wykonania funkcji użytkowej GET. Żadna nazwa występująca w liście docelowej nie może być związana kwantyfikatorem (n-tka tworzonej relacji odpowiada zmiennej wolnej funkcji zdaniowej). Nazwy występujące w liście docelowej mogą odnosić się do dowolnej liczby relacji bazy danych. Jeżeli w liście docelowej wystąpi jedynie nazwa relacji lub nazwa zmiennej relacyjnej, to lista obejmuje wszystkie dziedziny tej relacji. Przepływ danych między procesem użytkowym a bazą danych jest realizowany za pośrednictwem obszaru roboczego. Dane przesyłane z bazy danych do obszaru roboczego tworzą tablicę o następujących właściwościach:

— dane elementarne są tego samego typu w ramach jednej kolumny,

— różne kolumny mogą obejmować dane elementarne różnych typów,

— wszystkie wiersze tablicy są różne.

Relacja umieszczona w obszarze roboczym może uczestniczyć w relacji funkcji selekcji, będąc tym samym rozszerzeniem zbioru relacji bazy danych. Ta możliwość jest szczególnie użyteczna w wypadku konieczności rozłożenia bardzo skomplikowanego wyrażenia selekcji na ciąg prostych wyrażeń. Proces użytkowy

może równolegle korzystać z dowolnej liczby obszarów roboczych. Zmienna relacyjna jest reprezentacją n-tki określonej relacji, przy czym zakresem zmienności tej zmiennej jest cała relacja. Zmienna relacyjna jest deklarowana przez użytkownika za pomocą operatora RANGE. Ponieważ wykorzystanie zmiennej relacyjnej jest konieczne w niektórych wyrażeniach selekcji zawierających kwantyfikatory, to przyjmiemy że zmienne relacyjne muszą być zawsze używane w wypadku wystąpienia takich wyrażeń. Wykonanie zapytania języka ALPHA wymaga deklaracji obszaru roboczego, w którym jest zapamiętywany wynik selekcji. W przykładach zapytań oznaczymy taki obszar roboczy symbolem W.

Przykład 1

Znajdź nazwiska wszystkich pracowników, którzy pracują w dziale zabawek.

```
RANGE   PRACOWNIK    P
RANGE   DZIAŁ        D
GET W P. NAZWISKO: ∨ D((P. NUMER-DZIAŁU=D.
NUMER=DZIAŁU)
          (D TYP=ASORTYMENTU='ZABAWKI'))
```

Predykat elementarny P.NUMER-DZIAŁU=D.NUMER DZIAŁU nazwiemy zwrotem łączenia relacji, ponieważ odpowiada on bezpośrednio operacji łączenia relacji w algebrze relacji.

Przykład 2

Znajdź wszystkie działy zlokalizowane na 2 piętrze, które sprzedają towar dostarczany przez dostawcę 'XYZ'.

```
RANGE   DZIAŁ      D
RANGE   TOWAR      T
RANGE   DOSTAWA    S
  GET W D. NUMER-DZIAŁU: (D. NUMER-PIĘTRA = '2') ∧
      ∨ T (D. NUMER-DZIAŁU = T. NUMER-DZIAŁU ∧
∨ S (T. KOD = S. KOD ∧ S. NAZWA-DOSTAWCY = 'XYZ'))
```

Zgodnie z twierdzeniem o sprowadzaniu formuł do postaci normalnej możemy napisać ten predykat w taki sposób, aby wszystkie kwantyfikatory znajdowały się na początku.

GET W D. NUMER-DZIAŁU: $\vee$ T $\vee$ S (D. NUMER-PIĘTRA =
= '2' $\wedge$ D. NUMER-DZIAŁU = T. NUMER-DZIAŁU $\wedge$
T. KOD = S. KOD S. NAZWA-DOSTAWCY = 'XYZ')

Każdy kwantyfikator znajdujący się po lewej stronie formuły może być przeniesiony do zdania RANGE, dotyczącego zmiennej związanej tym kwantyfikatorem. Atrybut SOME oznacza kwantyfikator szczegółowy, a atrybut ALL kwantyfikator ogólny, wiążący wymienioną w zdaniu RANGE zmienną relacyjną. Zdanie GET razem z koniecznymi zdaniami RANGE mogą więc mieć następującą postać:

```
RANGE  DZIAŁ     D
RANGE  TOWAR     T  SOME
RANGE  DOSTAWA   S  SOME
   GET W D. NUMER-DZIAŁU = D. NUMER-DZIAŁU =
       = T. NUMER-DZIAŁU ∧ T. KOD = S. KOD
           S. NAZWA-DOSTAWCY = 'XYZ'
```

Ponieważ wszystkie zmienne relacyjne, które nie występują w liście docelowej muszą być związane kwantyfikatorem, to w wypadku opuszczenia kwantyfikatora dla jakiejś zmiennej przyjmuje się przez domniemanie kwantyfikator szczegółowy (atrybut SOME w zdaniu RANGE).

Przykład 3

Znajdź nazwy wszystkich tych dostawców, którzy dostarczają towary do wszystkich działów.

GET W X. NAZWA-DOSTAWCY: $\wedge$ D $\vee$ T $\vee$ S (D. NUMER-
DZIAŁU = T. NUMER-DZIAŁU $\wedge$ T. KOD =
= S. KOD $\wedge$ S. NAZWA-DOSTAWCY =
= NAZWA-DOSTAWCY)

Kolejność kwantyfikatorów w tej formule musi być zachowana, ponieważ zmiana kolejności różnokształtnych kwantyfikatorów zmienia znaczenie predykatu. Stąd też odpowiednie zdania RANGE zawierające kwantyfikatory muszą wystąpić w odpowiedniej kolejności.

```
RANGE  DOSTAWCA  X
RANGE  DZIAŁ     D  ALL
```

```
RANGE  TOWAR      T  SOME
RANGE  DOSTAWA    S  SOME
  GET W X. NAZWA-DOSTAWCY: (D. NUMER-DZIAŁU =
      = T. NUMER-DZIAŁU ∧ T. KOD = S. KOD ∧
    S. NAZWA. DOSTAWCY = X. NAZWA-DOSTAWCY)
```

Relacja umieszczona w obszarze roboczym jako wynik realizowanej selekcji może być posortowana rosnąco (operator UP) lub malejąco (operator DOWN) według dowolnych atrybutów.

Możliwość wyrażeń selekcji w znacznym stopniu rozszerza biblioteka funkcji języka ALPHA, przy czym zbiór dostępnych funkcji może być dowolnie rozbudowywany. Biblioteka zawiera m.in. następujące funkcje:

COUNT — daj liczbę elementów zbioru,
MAX — wybierz maksymalną wartość ze skończonego zbioru wartości,
MIN — wybierz minimalną wartość ze skończonego zbioru wartości,
TOTAL — daj sumę wszystkich wartości numerycznego atrybutu (duplikaty, jeżeli występują, są włączone do sumy).

Uzyskiwanie relacji jako wyniku wyrażenia selekcji może spowodować istotne problemy, wynikające z zasad działania bazowych języków programowania. Trudny do rozwiązania jest problem rezerwacji pamięci operacyjnej na tablice wynikowe (broszury robocze) w wypadku dużej i trudnej do przewidzenia liczby danych wyjściowych. Wprowadzony do języka ALPHA aparat sekwencyjnego dostępu do kolejnych n-tek relacji, będącej wynikiem wyrażenia selekcji, daje możliwość wykorzystania tego języka jako rozszerzenie możliwości bazowego języka programowania. Rozwiązanie przykładu 3, zgodnie z zasadą sekwencyjnego przetwarzania, prowadzi do następującego ciągu wyrażeń języka ALPHA:

```
RANGE  DOSTAWCA   X
RANGE  DZIAŁ      D  ALL
RANGE  TOWAR      T  SOME
RANGE  DOSTAWA    S  SOME
OPEN GET W X. NAZWA-DOSTAWCY: (D. NUMER DZIAŁU =
```

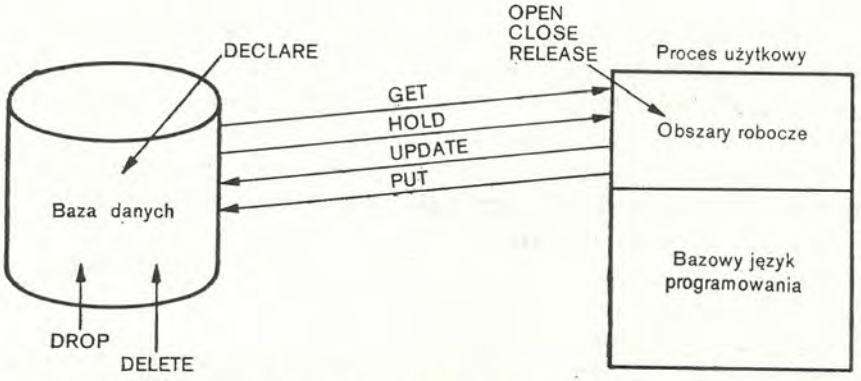
= T. NUMER-DZIAŁU ∧ T. KOD = S. KOD ∧ S. NAZWA-DOSTAWCY = X. NAZWA-DOSTAWCY)

Każde kolejne wykonanie wyrażenia:

GET W

powoduje pobranie następnej wartości X. NAZWA-DOSTAWCY do obszaru roboczego W. Sterowanie jest zwracane do programu użytkowego po pobraniu każdej kolejnej wartości. Przerwanie sekwencyjnego przetwarzania jest powodowane wykonaniem wyrażenia CLOSE W. Po wyczerpaniu wszystkich wartości wybranych w wyrażeniu selekcji system informuje użytkownika o końcu zbioru.

Cechą charakterystyczną selekcji realizowanej w języku ALPHA jest jej względna niezależność od istniejących w strukturze danych ścieżek selekcji. Ponieważ istniejące w strukturze danych powiązania są realizowane za pośrednictwem operacji łączenia relacji, możliwość wykonania wyrażenia selekcji jest determinowana przez możliwość wykonania tych operacji na relacjach objętych selekcją. Pełny repertuar funkcji użytkowych realizowanych w języku ALPHA ilustruje schemat przepływu informacji (por. rys. 52). Wykonanie funkcji użytkowej HOLD ma taki sam efekt, jak wykonanie GET, przy czym dodatkowo uprzedza system ZBD o zamiarze wykonania aktualizacji wybranych wartości. Funkcja UPDATE W prowadzi do wykonania aktualizacji odpowiednich wartości danych w bazie danych na



Rys. 52. Język ALPHA — przepływ danych i informacji sterujących

podstawie wartości danych, zawartych w obszarze roboczym W. DELETE powoduje skreślenie z bazy danych wszystkich wybranych n-tek określonej relacji, a funkcja DROP skreślenie całej relacji. Funkcja PUT umożliwia dopisanie n-tek do istniejących relacji bazy danych lub dopisanie całej nowej relacji, przy czym w takim wypadku relacja bazy danych musi zostać przedtem opisana w tym procesie użytkowym za pośrednictwem funkcji DECLARE.

b. Strukturalny język zapytań SEQUEL

Wyrażenia selekcji w strukturze relacyjnej oparte na rachunku predykatów wymagają określenia:

— dodatkowych zmiennych, których zakresem zmienności są wartości wierszy lub części wierszy tablicy reprezentującej relację,

— funkcji zdaniowej, zawierającej operatory relacyjne, spójniki zdaniowe i kwantyfikatory.

Stosowanie rachunku predykatów wymaga od użytkownika znajomości elementów logiki matematycznej i rachunku kwantyfikatorów. Jest to istotnym utrudnieniem dla wielu użytkowników. Można przyjąć, że użytkownik bezpośredni traktuje relacyjną strukturę danych jako pewien zbiór tablic, a formułowane przez niego wyrażenia selekcji są analogiczne do zwykłego przeszukiwania tablic przebiegającego według następującego schematu:

1) przejrzyj kolumnę (lub kolumny) tablicy szukając zadanej wartości (lub zbioru wartości),

2) dla każdej z tablic wartości wybierz odpowiednie elementy innych kolumn występujące w tym samym wierszu.

W wypadku stosowania rachunku predykatów konstrukcja wyrażenia selekcji wymaga przyjęcia zupełnie innego schematu:

1) wybierz wiersze tablic,

2) oblicz wartość logiczną predykatu, jeżeli prawda, to pobierz wiersze (lub części wierszy),

3) wykonaj następną iterację.

Język SEQUEL jest propozycją realizacji wymaganej przez użytkownika selekcji według naturalnego dla niego schematu

przeszukiwania tablic, co pozwala uniknąć konieczności używania kwantyfikatorów i wyrażeń łączenia relacji stosowanych w wypadku korelowania danych zawartych w różnych relacjach.

Strukturalny język zapytań SEQUEL (*Structured English Query Language* [3]) był opracowany w latach siedemdziesiątych w ośrodku badawczym firmy IBM w San Jose. Był on jednym z elementów relacyjnego systemu ZBD SYSTEM-R. Podzbiór tego języka został wykorzystany w systemie ZBD RODAN jako relacyjny język bezpośredniego użytkownika.

Funkcje języka SEQUEL obejmują dwie podstawowe grupy zdań. Zdania przeznaczone do opisu relacyjnej struktury danych i zdania działania na relacjach. Relacje opisywane w języku SEQUEL dzielą się na dwie kategorie, relacje bazowe oraz relacje pochodne. Relacje bazowe są opisywane przy użyciu zdania DEFINE TABLE, a relacje pochodne za pośrednictwem zdania DEFINE VIEW. Z punktu widzenia użytkownika, wykonującego działania na relacjach, nie ma istotnej różnicy między obu tymi typami relacji. Przez materializację relacji należy rozumieć proces kompletowania (w obszarach roboczych systemu) tablicy dwuwymiarowej, będącej aktualnym w danej chwili obrazem relacji. Materializacja relacji bazowej polega na odczytywaniu wartości danych z bazy danych, tworzeniu z tych wartości kolejnych wierszy relacji oraz przesyłaniu tych wierszy do odpowiedniego obszaru roboczego. W wypadku wykorzystania języka SEQUEL w systemie ZBD RODAN proces wybierania wartości danych z bazy danych przebiega zgodnie z regułami odwzorowań opisanymi w języku opisu odwzorowań (por. rozdz. IV, pkt 5). Sposób materializacji relacji pochodnej wynika z opisu tej relacji. Opis relacji pochodnej ma formy zapytania z dodatkowym podaniem nazwy relacji i nazw poszczególnych jej atrybutów (kolumn). Taki sposób definiowania relacji jest możliwy, ponieważ w języku SEQUEL wynikiem każdego zapytania jest relacja. Możliwości języka SEQUEL zademonstrujemy na podstawie wielu przykładów opisu relacji pochodnych oraz formułowaniu zapy-

[3] Por. D. Chamberlin, R. Boyce, *SEQUEL: A Structured English Querry Language*, Proc. ACM SIGMOD 1974, Ann Arbor 1974; R. Boyce, D. Chamberlin, *Using Structured English Query Language is a Data Definition Facility*, IBM, Research Report Ry 1318, 1973.

tań, opierają się na relacyjnym modelu domu towarowego (por. rozdz. IV, pkt 4).

Przykład 1

Zdefiniuj relację pochodną obejmującą wszystkich nisko uposażonych pracowników, tj. pracowników zarabiających poniżej 15 tys. zł.

```
DEFINE VIEW
  BIEDNI-PRACOWNICY (IMIĘ, NAZWISKO, POBORY
  FUNKCJA)
  AS SELECT IMIĘ, NAZWISKO, POBORY, FUNKCJA
    FROM PRACOWNICY
    WHERE POBORY < 15 000
```

Po trasie DEFINE VIEW następuje nazwa relacji pochodnej oraz podana w nawiasach lista jej atrybutów. Materializacja tej relacji przebiega tak, jak realizacja zapytania. Wyrażenie SELECT zawiera listę docelową zapytania, to jest listę atrybutów, które mają być pobrane, określenie źródła tych atrybutów (FROM) oraz opis warunków selekcji (WHERE). W tym wypadku opis relacji pochodnej BIEDNI-PRACOWNICY jest wynikiem zapytania obejmującego relację odpowiednich n-tek z relacji PRACOWNICY.

Opis relacji pochodnej może również odwoływać się do innej relacji pochodnej, pod warunkiem, że została ona już wcześniej opisana.

Przykład 2

Opisz relację pochodną, obejmującą wszystkich źle zarabiających pracowników, którzy pełnią funkcje kierownicze.

```
DEFINE VIEW
  BIEDNI-KIEROWNICY (IMIĘ, NAZWISKO, POBORY)
  AS SELECT IMIĘ, NAZWISKO, POBORY
  FROM BIEDNI-PRACOWNICY
  WHERE FUNKCJA KIEROWNIK
```

Jeżeli przewidujemy, że odwołania do relacji pochodnej będą wykonywane wielokrotnie, możemy zapamiętać jej opis w katalogu relacji. Opis relacji jest zapamiętywany w katalogu relacji, jeżeli zawiera opcję PERMANENT. Użycie opcji TEMPORARY

sprawia, że relacja jest dostępna jedynie w czasie trwania dialogu użytkownika z systemem, w toku którego została opisana. Użytkownik, który definiuje daną relację jest jej właścicielem. Tylko właściciel relacji może tę relację wykreślić z katalogu relacji, używając zdania DROP TABLE lub zdania DROP VIEW. Właściciel zakatalogowanej relacji może przyznać użytkownikom prawo dostępu do tej relacji, używając zdania GRANT. Używając zdania REVOKE właściciel relacji może odwołać prawo dostępu do niej dla wszystkich lub wybranych użytkowników.

Możliwość definiowania relacji pochodnych w połączeniu z możliwością przyznawania praw dostępu do relacji jest bardzo selektywnym mechanizmem do określania fragmentów bazy danych dostępnych dla różnych użytkowników. Relacje pochodne w połączeniu z możliwością ich katalogowania umożliwiają wygodne przechowywanie opisów bardziej skompletowanych zapytań. Działanie na relacjach obejmuje realizację zapytań oraz aktualizację wybranych elementów relacji. Pomijając frazę opisującą sposób edycji wyniku zapytania, każde zapytanie jest kombinacją bloków SELECT, z których każdy ma następującą postać:

```
SELECT    lista-wyrażeń,
FROM      lista-relacji,
WHERE     wyrażenia-selekcji,
GROUP BY  lista-atrybutów HAVING wyrażenie-selekcji.
```

Realizację zapytania określonego w jednym bloku SELECT można w uproszczeniu przedstawić w następujący sposób:

1) jest tworzony iloczyn kartezjański relacji określonych w liście relacji,

2) dla każdego wiersza otrzymanego iloczynu kartezjańskiego sprawdza się, czy warunki podane w wyrażeniu selekcji są spełnione,

3) jeżeli dla danego wiersza warunki wyrażenia selekcji są spełnione, to oblicza się poszczególne wyrażenia z listy wyrażeń (podanej po słowie kluczowym SELECT) i listę (wiersz) otrzymanych wyników dołącza się jako kolejny wiersz do tablicy będącej wynikiem wykonania bloku SELECT.

4) jeżeli po słowie kluczowym FROM jest wymieniona tylko

jedna relacja, to wykonanie bloku SELECT sprowadza się do punktów 2), 3) odniesionych do tej relacji.

Przykład 3

W domu towarowym (znanym z poprzednich przykładów) znajdź wszystkie działy zlokalizowane na drugim piętrze:

```
SELECT  NUMER-DZIAŁU
FROM    DZIAŁ
WHERE   NUMER-PIĘTRA = 2
```

Lista wyrażeń może być zastąpiona gwiazdką, jeżeli interesują nas wszystkie atrybuty danej relacji. Relacje wymienione w liście relacji są lokalne w ramach danego bloku SELECT. Oznacza to, że jedna i ta sama relacja może być wielokrotnie użyta w zapytaniu, występując w różnych blokach SELECT. W ramach jednego zapytania może wystąpić dowolna liczba bloków SELECT.

Przykład 4

Znajdź nazwiska wszystkich pracowników, którzy pracują w dziale zabawek.

```
SELECT  NAZWISKO
FROM    PRACOWNIK
WHERE   NUMER-DZIAŁU =
        = SELECT NUMER-DZIAŁU
        FROM DZIAŁ
        WHERE TYP-ASORTYMENTU = ZABAWKI
```

W pewnych wypadkach istnieje konieczność kwalifikowania nazw relacji użytych w liście relacji dodatkowymi nazwami dowolnie wybranymi przez użytkownika. Ma to miejsce w następujących dwóch wypadkach.

1. Gdy relacja występuje wielokrotnie w liście relacji i trzeba zapewnić jednoznaczność odwołań, występujących w wyrażeniu selekcji lub w liście wyrażeń.
2. Gdy w wyrażeniu selekcji danego bloku SELECT chcemy odwołać się do bloku SELECT na wyższym poziomie.

Przykład 5

Wybierz ten dział, który ma największą wartość sprzedaży spośród wszystkich działów sprzedających ten sam asortyment.

```
SELECT  X. NUMER-DZIAŁU, X. TYP ASORTYMENTU
FROM    DZIAŁ X, DZIAŁ Y
WHERE   X. TYP-ASORTYMENTU =
        = Y. TYP-ASORTYMENTU
AND     X. WARTOŚĆ-SPRZEDAŻY =
        = MAX (Y. WARTOŚĆ-SPRZEDAŻY)
```

W przykładzie 5 wykorzystano, poza możliwością wprowadzania własnych nazw relacji, również spójnik zdaniowy AND oraz funkcję standardową języka SEQUEL, służącą do wybierania najwyższej wartości (MAX). Innymi funkcjami standardowymi są AVG — obliczanie średniej wartości, MIN — wybieranie najmniejszej wartości czy COUNT — podanie liczby wystąpień. W bloku SELECT jest możliwe używanie operacji grupowania wierszy relacji ze względu na wartość wskazanych atrybutów.

Przykład 6

Znajdź wszystkie te działy, które zatrudniają więcej niż 10 pracowników.

```
SELECT  NUMER-DZIAŁU
FROM    PRACOWNIK GROUP BY NUMER-DZIAŁU
HAVING  COUNT (IMIĘ, NAZWISKO) > 10
```

Używając operacji grupowania wierszy relacji można dodatkowo określać w zapytaniu kryteria selekcji dla tworzonych grup. Fraza HAVING może wystąpić jedynie w połączeniu z operacją grupowania wierszy (GROUP BY). Fraza GROUP BY może być używana wyłącznie w wyrażeniach selekcji (po słowie kluczowym WHERE).

Wykorzystanie języka SEQUEL w bazowym języku programowania wymaga wprowadzenia mechanizmu dostępu do kolejnych wierszy wybranych na podstawie wyrażenia selekcji. Programista musi również określić zmienne zadeklarowane w programie, do których system ma przesyłać wybrane wartości za pomocą operatora BIND. Dostarczanie wartości z kolejnych wybranych wierszy jest realizowane na podstawie wskaźnika selekcji (*cursor*) za pośrednictwem operatora FETCH. Rozwiązanie przykładu 6 w programie PL/1 wymaga następującej sekwencji zdań języka SEQUEL.

```
CALL BIND ('X', ADDR (X));
CALL SEQUEL (C1, 'SELECT NUMER-DZIAŁU: X
                  FROM PRACOWNIK GROUP BY
                  NUMER-DZIAŁU
                  HAVING COUNT (IMIE, NAZWISKO) > 10);
```

Odwołanie do systemu (CALL SEQUEL) spowoduje ustawienie wskaźnika selekcji C1 przed pierwszym wierszem tablicy, będącej rezultatem wyrażenia selekcji. Każdorazowe wykonanie zdania CALL RETCH (C1) spowoduje przesłanie wartości pola NUMER-DZIAŁU do zmiennej X oraz przesunięcie wskaźnika selekcji C1 do następnego wiersza. Wyczerpanie wszystkich wierszy (aktywny zbiór) wybranej tablicy jest sygnalizowane przez system. Program użytkowy może równolegle utrzymywać dowolną ilość wskaźników selekcji oraz wykorzystywać je dla budowania wyrażeń selekcji. Wprowadzanie nowych n-tek do relacji bazy danych (INSERT), skreślanie n-tek (DELETE) oraz zmiana wartości (UPDATE) może być realizowane na pojedynczej n-tce lub ich zbiorze w ramach jednego odwołania do systemu. Bieżąca n-tka będąca przedmiotem aktualizacji jest wyznaczana za pomocą specjalnego predykatu CURRENT TUPLE OF CURSOR, przy czym można korzystać z dowolnego aktywnego w tym momencie wskaźnika selekcji. Nowe wartości mogą być przedstawione w postaci stałych, zmiennych programu (określonych przez operator BIND) lub wyrażeń arytmetycznych.

Przykład 7

Podnieś wynagrodzenie wszystkich pracowników działu 2 o 10%.

```
CALL SEQUEL UPDATE PRACOWNIK
                SET POBORY = POBORY * 1.1
                WHERE NUMER-DZIAŁU = 2';
```

Aktualizacji podlegają wszystkie wiersze tablicy PRACOWNIK, które spełniają klauzulę WHERE. Słowo kluczowe SET powoduje ustawienie nowej wartości wymienionego atrybutu.

Przykład 8

Podnieś wynagrodzenie określonego pracownika, oznaczonego wskaźnikiem selekcji C1 (wybrany w programie użytkowym ze zbioru rozpatrywanych pracowników) o 10%.

```
CALL SEQUEL 'UPDATE PRACOWNIK
            SET POBORY = POBORY * 1.1
            WHERE CURRENT TUPLE OF CURSOR C1';
```

Analogicznie argumentem operatora SET może być stała lub zmienna zadeklarowana w programie użytkowym. Przykłady wykorzystania języka SEQUEL oraz zasady strukturalnego programowania przy jego wykorzystaniu można znaleźć w literaturze.

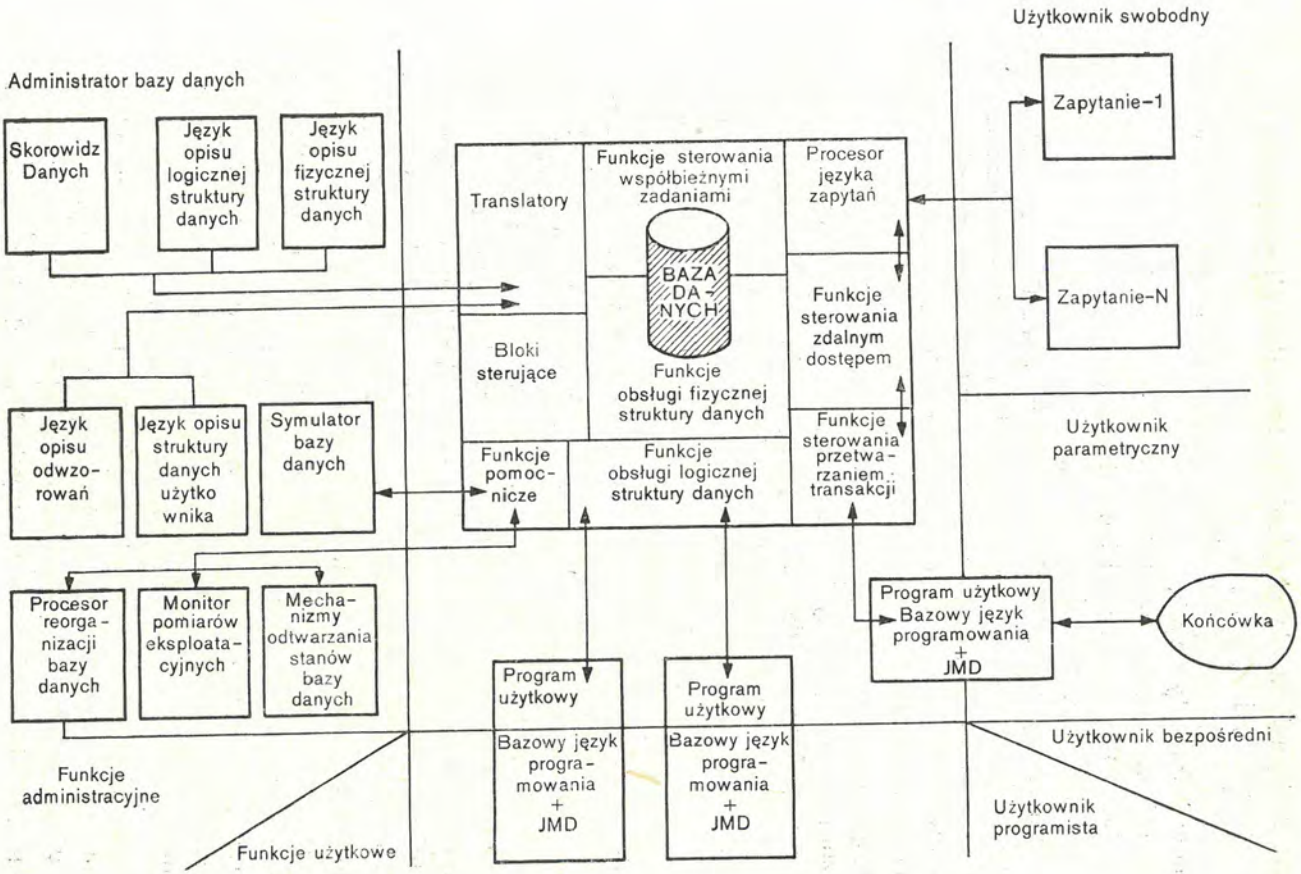
VII. Architektura Systemów Zarządzania Bazą Danych

1. Podstawowe elementy architektury SZBD

Możliwości systemu zarządzania bazą danych wynikają z dostępnego w tym systemie zbioru mechanizmów, służących do realizacji i utrzymania bazy danych oraz konstrukcji oprogramowania eksploatowanych na podstawie tej bazy danych systemów informatycznych. W otoczeniu systemu ZBD działają trzy podstawowe typy jego użytkowników: Administrator Bazy Danych, programista użytkowy oraz użytkownik finalny. Możliwości poszczególnych typów użytkowników oraz zakres ich kompetencji są zasadniczo różne, co doprowadziło do powstania trzech wyraźnie zróżnicowanych grup mechanizmów systemu ZBD. Najbardziej rozbudowaną grupą są mechanizmy systemu spełniające funkcje administracyjne, tj. funkcje obejmujące zagadnienia związane z projektowaniem, zakładaniem i utrzymaniem bazy danych. Dwie pozostałe grupy mechanizmów organizują dostęp programów użytkowych lub użytkowników finalnych do bazy danych. Schemat architektury systemu ZBD uwzględniający trzy podstawowe grupy mechanizmów przedstawiamy na rysunku 53.

Administrator Bazy Danych odpowiada za następujące grupy zagadnień.

1. Organizacja bazy danych obejmuje:
   — projektowanie i opis logicznej struktury danych,
   — projektowanie i opis fizycznej struktury danych,
   — projektowanie i opis struktury danych użytkownika dla realizowanych na podstawie baz danych systemów informatycznych,
   — przydział praw dostępu do bazy danych,
   — założenie bazy danych.



Rys. 53. Architektura Systemu Zarządzania Bazą Danych

2. Bieżąca kontrola eksploatacji bazy danych obejmuje:
— analizę jej wykorzystania,
— analizę parametrów eksploatacyjnych poszczególnych systemów informatycznych,
— ochronę integralności danych.
— kontrolę poprawności bazy danych,
— organizację procedur awaryjnych.

Postulowany zakres funkcji Administratora Bazy Danych wymaga oddania do jego dyspozycji wielu różnych mechanizmów systemu ZBD. Szczególnie ważny z takich mechanizmów jest Skorowidz Danych, wspomagający projektowanie i opis pojęciowego modelu danych.

Języki opisu danych służą do opisu elementów logicznej i fizycznej struktury danych oraz struktury danych użytkownika. Zbiór języków opisu danych dostępnych w ramach systemu ZBD jest podstawowym narzędziem Administratora Bazy Danych, umożliwiając mu projektowanie, zakładanie i utrzymanie bazy danych. Rozważając możliwości języków opisu danych podzielimy elementy tych języków na siedem podstawowych grup, obejmujących takie elementy, jak opis struktury, kontrola danych, kontrola dostępu, manipulacja danymi, wyznaczanie parametrów eksploatacyjnych, alokacja pamięci i administrowanie danymi. Opis struktury tworzą wszystkie te klauzule, które określają typy elementów występujących w logicznej i fizycznej strukturze danych, a w wypadku opisu struktury danych użytkownika znajdują się w polu widzenia procesu użytkowego. Kontrola danych daje możliwość automatycznego utrzymania zasad spójności logicznej obowiązujących w ramach opisywanego modelu danych. Elementy opisu zasad automatycznej kontroli danych pozwalają na centralną kontrolę danych, niezależnie od procesów użytkowych. Kontrola dostępu obejmuje opis zasad dopuszczalności wykonania poszczególnych funkcji użytkowych, w ramach obsługiwanych procesów użytkowych.

Elementy opisu dopuszczalności funkcji użytkowych są również aparatem synchronizacji równolegle wykonywanych procesów użytkowych i dają możliwość przyjęcia selektywnej strategii przydziału zasobów bazy danych. Manipulacja danymi obej-

muje wszystkie te elementy opisu, które są bezpośrednio rozszerzeniem możliwości Języków Manipulacji Danymi przez deklarowanie pewnych automatycznie wykonywanych funkcji (np. ścieżka selekcji w sieciowej strukturze danych). Wyznaczanie parametrów eksploatacyjnych bazy danych jest przede wszystkim związane z tymi elementami opisu, które bezpośrednio dotyczą fizycznej struktury danych. Do tej grupy zaliczamy wszystkie te elementy opisu, które stanowią informację dla systemu ZBD, umożliwiając usprawnienie obsługi funkcji użytkowych lub obniżenie kosztu przechowywania danych. Alokacja pamięci polega na przypisywaniu poszczególnych elementów struktury danych do odpowiednich jednostek organizacyjnych bazy danych (obszary, urządzenia itp.). Brak takich elementów opisu powoduje zwykle przyjęcie przez system ZBD domniemanych zasad alokacji. Środki administrowania danymi obejmują procedury przygotowane przez Administratora Bazy Danych, które są wywoływane automatycznie przez system ZBD w określonych w opisie danych punktach przetwarzania procesów użytkowych.

Możliwości języków opisu danych wynikają przede wszystkim z zasad budowy opisywanego modelu danych oraz zakresu dopuszczalnych odwzorowań między rożnymi modelami danych obsługiwanymi przez system ZBD. Te zagadnienia zostały szczegółowo omówione w poprzednich rozdziałach pracy. Mechanizmami pozwalającymi na realizację funkcji bieżącej kontroli eksploatacji bazy danych są takie elementy systemu ZBD, jak procesor reorganizacji bazy danych, monitor pomiarów eksploatacyjnych oraz mechanizmy odtwarzania stanów bazy danych. Ze względu na duże znaczenie tych mechanizmów, zostały one omówione szczegółowo w dalszych punktach tego rozdziału.

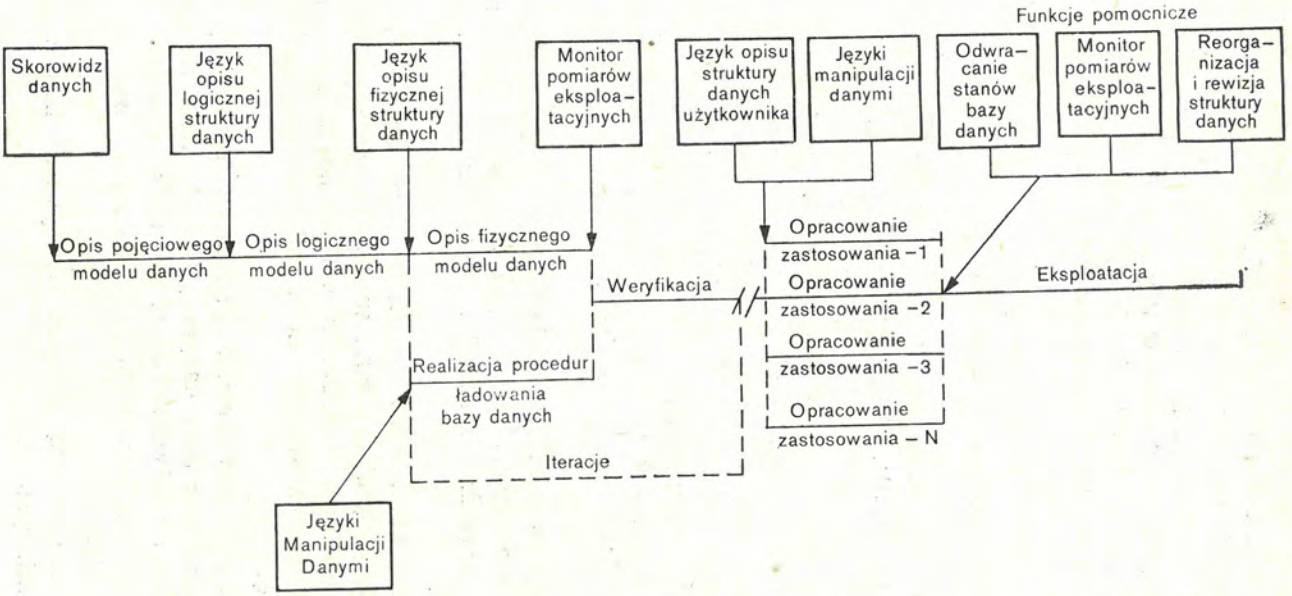
Podział na pozostałe dwie grupy użytkowników systemu ZBD przeprowadzimy na podstawie stosowanych przez nich metodach dostępu do bazy danych. Użytkownicy programiści realizują dostęp do bazy danych za pośrednictwem przygotowanych przez siebie programów użytkowych pisanych w bazowych językach programowania (PL1, COBOL i in.) z dodatkowym użyciem komend Języka Manipulacji Danymi. Język ten jest na ogół językiem proceduralnym, umożliwiającym swobodną nawigację po bazie danych. Daje to programiście możliwość stosowa-

nia efektywnych algorytmów wyszukiwania oraz aktualizacji bazy danych.

Użytkowników bezpośrednich możemy podzielić na użytkowników parametrycznych i użytkowników swobodnych. Komunikacja użytkowników parametrycznych z systemem ZBD sprowadza się zwykle do prostego dialogu, często za pośrednictwem formularzy wyświetlanych na ekranie końcówki. Wyświetlanie formularzy wypełnionych odpowiednimi informacjami, jak również wczytywanie informacji wejściowych, wprowadzonych w odpowiednie miejsca formularza, oraz ich przetwarzanie jest obsługiwane zwykle przez program użytkowy, przygotowany uprzednio przez programistę. Taki program jest wywołany przez użytkownika za pośrednictwem końcówki przed rozpoczęciem dialogu. Wizja bazy danych (model danych użytkownika) użytkowników parametrycznych ma więc charakter formularzowy. Zakres informacji dostępnych dla użytkowników parametrycznych jest z reguły bardzo ograniczony.

Użytkownicy swobodni nie mają z góry określonych potrzeb co do zakresu żądanych informacji. Ich zapytania są z natury *ad hoc*, a zakres informacji może być stosunkowo szeroki. Użytkownicy ci nie mają szczegółowych algorytmów nawigacji po bazie danych. Wymaga się więc, aby przeznaczony dla nich język do zadawania pytań (Język Zapytań) był językiem nieproceduralnym. Oznacza to, że w formułowanych pytaniach użytkownik precyzuje jedynie, co chce uzyskać, bez określania, w jaki sposób ma być wykonane jego żądanie. Algorytm realizacji takiego zapytania jest ustalony przez system ZBD. Zagadnienia związane z językami dostępu do danych zostały już szczegółowo omówione. Wykorzystanie elementów architektury w procesie tworzenia i eksploatacji bazy danych przedstawiamy na rysunku 54.

W przedstawionej sieci czynności mieszczą się trzy podstawowe fazy prac: projektowanie i tworzenie bazy danych, projektowanie i tworzenie zastosowań oraz eksploatacja. Weryfikacja bazy danych polega przede wszystkim na ocenie jej parametrów eksploatacyjnych. W wypadku osiągnięcia niezadowalających parametrów należy powtórzyć czynności obejmujące opis fizycznego modelu danych oraz ładowanie bazy danych. Takie podejście do projektowania bazy danych prowadzi do powsta-



Rys. 54. Wykorzystanie elementów architektury systemu ZBD w procesie tworzenia i eksploatacji bazy danych

wania wysokich kosztów wynikających chociażby z konieczności wielokrotnego ładowania bazy danych. Możliwość uniknięcia takich kosztów daje symulator bazy danych pozwalający na ocenę parametrów eksploatacyjnych na podstawie odpowiedniego opisu struktury danych oraz procesów użytkowych. Stosowanie symulatora baz danych może być również bardzo przydatne dla podejmowania decyzji w zakresie reorganizacji bazy danych.

2. Odtwarzanie stanów bazy danych

Utrzymywanie wspólnej bazy danych będącej scentralizowanym źródłem informacji dla wielu systemów informatycznych prowadzi do poważnych problemów zachowania integralności danych w razie awarii. Stosowane techniki odtwarzania stanów bazy danych prowadzą do odzyskania integralności danych, przy czym uzyskany stan danych powinien być najbliższym integralnym stanem struktury danych w stosunku do momentu czasowego, w którym awaria wystąpiła.

Przydatny z punktu widzenia wyboru techniki odtwarzania zbiorów bazy danych jest podział możliwych błędów na następujące klasy.

Upadek bazy danych. Wszystkie dane w całej lub części bazy danych stają się niedostępne. Przykładem takiej sytuacji może być fizyczne uszkodzenie nośnika magnetycznego (pakiet dyskowy).

Błędy użytkownika. Wprowadzone do bazy danych wartości są sprzeczne z przyjętymi zasadami spójności logicznej i formalnej niesprzeczności lub są nieprawdziwe. Występowanie takich wartości w bazie danych wpływa na wyniki następnych procesów (kontaminacja).

Uszkodzenie jednostki centralnej i pamięci operacyjnej. W tym wypadku integralność bazy danych jest zachowana, nie jest ona jednak dostępna dla użytkowników. Rozwiązanie takiego problemu wykracza zwykle poza techniki odtwarzania rozważane w ramach systemu ZBD.

Istnieją w zasadzie trzy miejsca wykrywania błędów. Są to:
— użytkownik systemu,

— operator systemu lub Administrator Bazy Danych,
— system ZBD.

Oczywiście należy dążyć do tego, aby większość błędów wykrywał sam system, a najmniej użytkownik. Stara się to osiągnąć przez rozwiązanie sprzętowe (np. sprawdzanie bitów parzystości przy przesłaniach) oraz przez doskonalenie oprogramowania (np. wykrywanie zadań, które są w stanie permanentnego oczekiwania na sterowanie lub działają w pętli programowej, sprawdzanie poprawności tablic i bloków kontrolnych). Trzeba jednak dodać, że rozbudowanie diagnostyki systemu pociąga za sobą wzrost kosztów systemu rzędu od 20 do 30% i nie zawsze jest to finansowo uzasadnione. Ponadto istnieją także takie błędne sytuacje, które mogą być wykryte przede wszystkim przez operatora. Na przykład: uszkodzenie klimatyzacji, niesprawna praca urządzeń we/wy itp. Od szybkości działania operatora zależy czy awaria może być szybko usunięta. Błędy wykryte przez Administratora Bazy Danych na skutek analizy wyników badań eksploatacyjnych i diagnostycznych są na ogół zauważane na etapie testowania systemu i od momentu oddania systemu do normalnej eksploatacji z ich występowaniem można się nie liczyć. Pozostaje jednak pewna grupa błędów, która zostanie dostrzeżona przez użytkownika w wyniku analizy otrzymanych wyników i zgłoszona do operatora. Na ogół informacja taka dotrze do operatora po dość długim czasie od wystąpienia błędu. Tego rodzaju sytuacje prowadzą zwykle do dużych trudności w odtwarzaniu prawidłowych stanów bazy danych, ponieważ błąd mógł zdążyć się rozprzestrzenić na wiele rekordów w bazie danych i na wiele procesów użytkowych.

Techniki odtwarzania stanów bazy danych obejmują dwie grupy procedur:

— procedury gromadzenia informacji obejmujących historię wykonanych procesów użytkowych,
— procedury odtwarzania stanów bazy danych.

Istnieje siedem podstawowych metod zbierania informacji.

1. Kopia zawartości bazy danych umieszczana w zewnętrznych urządzeniach pamięciowych (*dumping*).

Można tu wyróżnić dwa sposoby:

— kopia fizyczna, polegająca na kolejnym przepisywaniu ście-

żek, ścieżki, cylindry czy bloki bazy danych cylindrów czy bloków fizycznych bazy danych,

— kopia logiczna, polegająca na kolejnym przepisywaniu rekordów logicznych.

Kopiowanie fizyczne jest oczywiście znacznie szybsze, ale za to mniej wygodne. Przy kopiowaniu logicznym mamy możliwość zagęszczenia rekordów bazy danych, a także wykonania pewnych operacji diagnostycznych na bazie danych. W niektórych wypadkach baza danych jest tak duża, że nie jest możliwe wykonanie kopii całej bazy danych za jednym razem (zajęłoby to zbyt dużo czasu). Alternatywnym rozwiązaniem jest wtedy kopiowanie zawartości kolejnych obszarów bazy. Innym wyjściem jest okresowe wykonywanie tzw. szczątkowej kopii. Polega ona na przeszukiwaniu wszystkich zapamiętanych rekordów, wybrania tych, które były modyfikowane od czasu ostatniego szczątkowego kopiowania i zapisanie ich w urządzeniach pamięci zewnętrznej. Kopię szczątkową można stosować wtedy, gdy każdy rekord w bazie danych posiada pole do zaznaczenia czasu operacji modyfikacji.

2. Rejestrowanie kopii rekordów przed modyfikacją.
3. Rejestrowanie kopii rekordów po modyfikacji.
4. Rejestrowanie transakcji wejściowych.
5. Rejestrowanie komunikatów wyjściowych.
6. Umieszczenie punktów kontrolnych (*check point*).
7. Utrzymywanie zapasowej bazy danych.

Omówione tu metody gromadzenia informacji można podzielić z kolei na dwie klasy.

I. Czynności sterowane, tzn. takie, których częstotliwość wykonywania jest ustalona przez osobę decydującą o strategii odtwarzania (Administratora Bazy Danych). Do tej klasy można zaliczyć kopiowanie zawartości bazy danych na taśmę i wydawanie punktów kontrolnych.

II. Czynności stochastyczne, tzn. takie, które są wykonywane w momencie wystąpienia jakiegoś zdarzenia. Do tej klasy czynności należą wszystkie rejestrowania (inaczej logowania). Tworzą one tzw. ślad kontrolny (*audit trail*), będący historią wejść przetwarzanych przez system oraz zmian dokonywanych w bazie danych. Ślad kontrolny powinien zawierać oprócz właściwej in-

formacji (kopii rekordu przed, czy po modyfikacji, tekstu transakcji wejściowej, tekstu komunikatu wyjściowego) pewne dodatkowe informacje służące do dokładnego rozpoznania rejestrowanej czynności:

— identyfikację końcówki,
— identyfikację użytkownika,
— identyfikację transakcji,
— identyfikację typu rejestrowanej czynności,
— stempel czasowy.

Procedurą odwracania stanów nazywamy układ wcześniej przygotowanych kroków, służących do odtwarzania systemu do stanu sprzed wystąpienia błędu. Ogólnie można wyróżnić osiem różnych procedur odtwarzania związanych z różnymi rodzajami błędów. Dla błędów powodujących upadek bazy danych dostępne są trzy procedury:

1. Kopiowanie bazy z zapasowej bazy danych.
2. Odtwarzanie bazy danych z ostatniej kopii i rekonstrukcja bazy przy użyciu kopii zmodyfikowanych rekordów.
3. Odtworzenie bazy danych z ostatniej kopii i powtórne wykonanie wszystkich procesów użytkowych.

W razie błędów spowodowanych przez procesy użytkowe możemy wykonać następujące procedury:

1. Odtworzenie bazy danych z ostatniej kopii przy wykorzystaniu kopii zmodyfikowanych rekordów dotyczących wszystkich prawidłowo zrealizowanych procesów. Ponowne wykonanie procesów, w których wystąpił błąd, po usunięciu źródła powstania błędu.
2. Otworzenie bazy danych do stanu sprzed wykonania błędnego procesu. Usunięcie źródła błędu i ponowne wykonanie wszystkich procesów.

Wybór odpowiednich technik odtwarzania stanów bazy danych oraz projekt podejmowanych procedur odtwarzania jest jednym z elementów projektu bazy danych.

3. Reorganizacja bazy danych

Logiczna struktura danych jest reprezentacją struktury rzeczywistości objętej bazą danych. Wszelkie zmiany zachodzące w struk-

turze rzeczywistości lub zmiany w sposobie wykorzystania bazy danych prowadzą nieuchronnie do zmiany logicznej struktury danych, a tym samym do zmiany zawartości informacyjnej bazy danych. Modyfikację opisu logicznej struktury danych oraz wynikające z niej zmiany logicznej struktury danych będziemy nazywali logiczną reorganizacją bazy danych lub rewizją struktury danych.

Zmiany obejmujące modyfikację opisu fizycznej struktury danych oraz wynikająca z niej przebudowa elementów fizycznej struktury danych nie zmieniają zawartości informacyjnej bazy danych. Zmiany należące do tej grupy nazwiemy fizyczną reorganizacją struktury danych. Wpływ takich zmian ogranicza się jedynie do kształtowania parametrów eksploatacyjnych rozpatrywanej bazy danych.

Logiczna reorganizacja struktury danych może obejmować następujące elementy:

— usunięcie istniejących typów danych elementarnych, rekordów i związków między tymi rekordami,

— dodanie nowych typów danych elementarnych, rekordów i związków między nimi,

— dodanie nowych typów danych elementarnych,

— zmiana struktury danych wewnątrz istniejących rekordów, np. wykorzystanie istniejących typów danych elementarnych do tworzenia nowych typów rekordów,

— zmiana charakterystyki wartości danych elementarnych,

— zmiana istniejących związków (zbiór strukturalny w wypadku struktury sieciowej).

Proces reorganizacji struktury danych może obejmować zmiany takich elementów, jak:

— rozmieszczenie obszarów bazy danych na urządzeniach pamięci zewnętrznej,

— wielkości obszarów bazy danych,

— wielkości stron bazy danych,

— zasady odwzorowania rekordów w fizycznej strukturze danych,

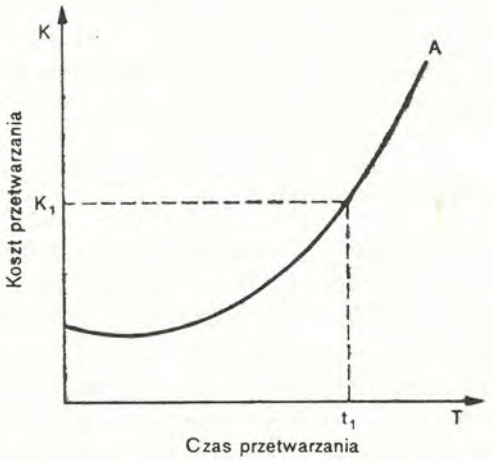
— algorytmy kodowania mieszającego,

— reprezentacja związków rekordów w fizycznej strukturze pamięci,

— strategia rozmieszczania rekordów (strategia przydziału klucza bazy danych).

Ogólnie rzecz biorąc możemy przyjąć, że możliwości w zakresie logicznej reorganizacji struktury danych powinny uwzględniać zmiany w dowolnych elementach opisu logicznej struktury danych, a fizyczna reorganizacja struktury danych powinna obejmować wszystkie elementy opisu fizycznej struktury danych.

Zmiana strategii rozmieszczenia rekordów w bazie danych wymaga określenia nowych zasad w postaci języka reorganizacji. Potrzeba takiego języka wynika z silnego uzależnienia sposobu rozmieszczenia rekordów od algorytmów procesów ładowania



Rys. 55. Krzywa zależności kosztu przetwarzania od czasu eksploatacji bazy danych

i aktualizacji bazy danych. Wysoka aktywność procesów aktualizacji powoduje powstanie istotnych różnic między zamierzonym (wynikającym z opisu strategii) a rzeczywistym rozkładem rekordów w obszarach bazy danych. Fakt ten jest jedną z podstawowych przyczyn pogarszania się parametrów eksploatacyjnych.

Przedstawiona krzywa zależności (por. rys. 55) między kosztem przetwarzania a czasem przetwarzania bazy danych ilustruje typową sytuację eksploatacyjną. Duża aktywność procesów aktualizacji (skreślenia i dopisywania rekordów) wpływa na wzrost wypełnienia obszarów bazy danych nie pozwalając tym samym

na realizację założonej strategii rozmieszczania rekordów. W tej sytuacji, czas przebiegu wszystkich procesów użytkowych, działających na tych obszarach bazy danych, ulega istotnemu wydłużeniu. Ponieważ wykonanie fizycznej reorganizacji wiąże się z poniesieniem określonego kosztu (przepisanie reorganizowanych obszarów bazy danych z koniecznym przetwarzaniem) można przyjąć, że istnieje pewien poziom kosztu przetwarzania (np. k_1), przy którym warto już wykonać taki przebieg. Przy stosunkowo równomiernej aktywności procesów aktualizujących bazę danych można wyznaczyć regularne okresy przetwarzania, po których należy wykonać fizyczną reorganizację. Innym zagadnieniem jest reorganizacja struktury danych, wynikająca ze zmiany sposobów jej wykorzystania (zmiana algorytmów procesów użytkowych, zmiana priorytetów tych procesów itp.). Praktycznie taka reorganizacja wymaga wykonania nowego projektu fizycznej struktury danych wraz z odpowiednim opisem.

4. Pomiary eksploatacyjne

Pomiary eksploatacyjne są podstawowym źródłem informacji potrzebnych Administratorowi Bazy Danych w procesie projektowania i utrzymania bazy danych. Iteracyjny charakter procesu projektowania bazy danych wymaga weryfikacji poszczególnych wersji projektu struktury danych na podstawie uzyskanych parametrów eksploatacyjnych wynikających z przebiegów kluczowych procesów użytkowych (tj. procesów użytkowych, dla których utrzymanie takich parametrów jak czas odpowiedzi, czas przebiegu jest sprawą istotną z punktu widzenia możliwości ich wykorzystania).

Ponieważ samo zbieranie informacji eksploatacyjnych może być istotnym czynnikiem obniżającym parametry eksploatacyjne systemu ZBD, zwykle istnieje możliwość włączania i wyłączania mechanizmów pomiarów. Typowymi metodami wykorzystania zbieranych informacji eksploatacyjnych są następujące procedury.

1. Analiza statystyczna uzyskanych danych pomiarowych.
2. Okresowe sprawozdania w zakresie informacji eksploatacyjnych zbieranych i przechowywanych przez system ZBD.

3. Dynamiczne informowanie operatora systemu (za pośrednictwem konsoli) o podstawowych parametrach bieżąco eksploatowanych procesów. Ten sposób wykorzystania informacji eksploatacyjnych daje możliwość optymalizacji pracy systemu przez interwencję operatora (zmiana priorytetów, przydziału zasobów itp.).

4. Informowanie o odchyleniach od przyjętych normatywnych parametrów eksploatacyjnych. Tego typu informacja może być podstawą zainicjowania fizycznej reorganizacji struktury danych.

5. Automatyczna reakcja systemu ZBD zmierzająca do poprawy uzyskiwanych parametrów eksploatacyjnych.

Pewna grupa informacji eksploatacyjnych dotyczy wyłącznie działania procesów użytkowych (np. czasy dostępu, czasy oczekiwania), a informacje należące do tej grupy mogą być zbierane wyłącznie w czasie wykonania tych procesów. Takie informacje eksploatacyjne nazwiemy dynamicznymi. Druga grupa informacji — to wyniki analizy stanu bazy danych w określonym momencie czasowym (np. stopień wypełnienia obszarów bazy danych, ilość wolnego miejsca). Analiza stanu bazy danych może być wykonana w dowolnym momencie czasowym, zupełnie niezależnie od procesów użytkowych.

Zbiór informacji eksploatacyjnych jest przetwarzany zwykle przez programy dokonujące odpowiedniej edycji (wykresy) oraz analizy statystycznej. Jest on dostępny również w swojej pierwotnej formie i może być przedmiotem przetwarzania przez odpowiednie procedury przygotowane przez Administratora Bazy Danych. Informacje eksploatacyjne możemy podzielić na trzy poziomy:

— poziom logicznej struktury danych,
— poziom fizycznej struktury danych,
— poziom struktury danych użytkownika.

Typowe informacje eksploatacyjne zbierane dla sieciowej struktury danych zawiera tablica 11.

Przykładem mechanizmów pomiarów eksploatacyjnych może być Monitor Pomiarów Eksploatacyjnych w Systemie Zarządzania Bazą Danych RODAN. Celem tego monitora jest dostarczenie Administratorowi Bazy Danych informacji eksploatacyjnych, umożliwiających podejmowanie odpowiednich decyzji w zakresie:

Tablica 11

Informacje eksploatacyjne istotne dla sieciowej struktury danych

Poziom pomiaru	Dynamiczne informacje eksploatacyjne	Statyczne informacje eksploatacyjne
1	2	3
Logiczna struktura danych	— częstotliwość wykorzystania poszczególnych funkcji użytkowych dotyczących poszczególnych typów rekordów, zbiorów strukturalnych i obszarów bazy danych, — informacje obejmują takie parametry jak: . ilość odwołań . czas odpowiedzi . ilość błędów itp., — częstotliwość wykorzystania określonych podschematów i procedur bazy danych.	— ilość rekordów danego typu w poszczególnych obszarach bazy danych, — ilość zbiorów strukturalnych danego typu, — rozkład liczności zbiorów strukturalnych danego typu, — rozkład ilości powtórzeń grup powtarzalnych w ramach danego typu rekordu.
Fizyczna struktura danych	— ilość dostępów (pisanie, czytanie) do obszarów bazy danych, — rozkład dostępów (adresy) do stron bazy danych, — czas wykonania krytycznych procesów na poziomie fizycznym (np. przeszukanie indeksu), — średnia długość kolejek, — ilość blokowanych procesów, — ilość zakleszczeń procesów.	— wielkość obszarów bazy danych, — rozkład rekordów na stronach w ramach obszaru bazy danych, — ilość wolnego miejsca w obszarze bazy danych, — rozkład wolnego miejsca na stronach w ramach obszaru bazy danych, — rozkład wielkości indeksów, — długość łańcuchów synonimów kodowania mieszającego, — rozkład populacji zbiorów strukturalnych na stronach obszaru bazy danych.
Struktura danych użytkownika	— częstotliwość wykorzystania określonej ścieżki selekcji,	

cd. tablicy 11

1	2	3
	— zmiany wykonane w bazie danych, — ilość odwołań do poszczególnych funkcji użytkowych, — czas wykonania poszczególnych funkcji użytkowych.	

— zmian zasad harmonogramowania procesów użytkowych eksploatowanych pod kontrolą Monitora Wielozadaniowego Systemu RODAN,

— logicznej i fizycznej reorganizacji struktury danych,

— rewizji struktury danych,

— modyfikacji algorytmów procesów użytkowych.

Moduł zbierania informacji eksploatacyjnych Monitora Pomiarów Eksploatacyjnych rejestruje następujące zdarzenia:

— inicjacji / zakończenia pracy modułu zbierania informacji eksploatacyjnych),

— inicjacji / zakończenia procesu użytkowego,

— inicjacji / zakończenia funkcji użytkowych STORE, DELETE, MODIFY, REMOVE, INSERT, DIND,

— pisania / skreślenia modyfikacji rekordu,

— otwarcia obszaru bazy danych,

— inicjacji / zakonczenia selekcji według zadeklarowanej w klauzuli SET SELECTION ścieżki selekcji,

— włączania / (wyłączania rekordów do / ze zbiorów strukturalnych.

Zbiór informacji eksploatacyjnych jest po odpowiednim posortowaniu poddawany obróbce przez procedury analizy pomiarów. Wynikiem działania tych procedur są następujące raporty.

1. Raport eksploatacyjny bazy danych:

— ogólne parametry eksploatacyjne,

— czas rozpoczęcia-zakończenia pracy MPE,

— czas eksploatacji bazy danych,

— czas jednostki centralnej wykorzystany przez moduł zbierania informacji eksploatacyjnych,

— czas jednostki centralnej wykorzystany przez Monitor Wielozadaniowy,
— liczba zainicjowanych procesów użytkowych,
— liczba błędnie zakończonych procesów użytkowych,
— liczba otwarć poszczególnych obszarów bazy danych,
— informacje o zmianach struktury danych,
— liczba dopisań, skreśleń i modyfikacji w ramach typu rekordu,
— liczba dopisań i skreśleń segmentów w ramach typu rekordu,
— liczba dopisań i skreśleń dzielonych rekordów (rekordów zapisanych na kilku segmentach) w ramach typu rekordu,
— średnie czasy selekcji według poszczególnych ścieżek selekcji.

2. Raport eksploatacyjny procesu użytkowego:
— kod procesu użytkowego,
— rodzaj zakończenia procesu użytkowego,
— typ błędu,
— czas zakończenia procesu użytkowego,
— czas jednostki centralnej wykorzystany przez proces użytkowy,
— liczba otwarć poszczególnych obszarów,
— informacje o zmianach dokonywanych w strukturze danych,
— średnie czasy realizacji i liczba odwołań do poszczególnych funkcji użytkowych,
— średnia liczba fizycznych transferów dla poszczególnych funkcji użytkowych,
— średnie czasy selekcji według poszczególnych ścieżek selekcji,
— średnie czasy selekcji wystąpień rekordów i liczby fizycznych transferów na selekcję według poszczególnych ścieżek selekcji,
— zmiany uczestnictwa rekordów w zbiorach strukturalnych.

Monitor Pomiarów Eksploatacyjnych jest włączany i wyłączany dynamicznie na żądanie operatora systemu. Dodatkowo istnieją w systemie ZBD RODAN funkcje pomocnicze, zbierające wiele statycznych informacji eksploatacyjnych.

VIII. Organizacja Systemów Zarządzania Bazą Danych

1. Ogólny model organizacji SZBD

Wykorzystanie SZBD w systemach informatycznych przynosi niewątpliwie korzyści, wynikające z centralizacji funkcji zarządzania zasobami danych, jak również ze znacznego skrócenia cyklu opracowania tych systemów oraz ich znacznie większej elastyczności użytkowej. Uzyskanie tych korzyści wymaga jednak poniesienia określonych kosztów wynikających z działania poszczególnych elementów systemu ZBD. Ważnym zadaniem projektantów systemu informatycznego jest minimalizacja takich kosztów przy jednoczesnym utrzymaniu wszystkich wymaganych funkcji systemu.

Świadome projektowanie bazy danych i systemów działających w jej otoczeniu zastosowań wymaga znajomości podstawowych elementów organizacji stosowanego SZBD.

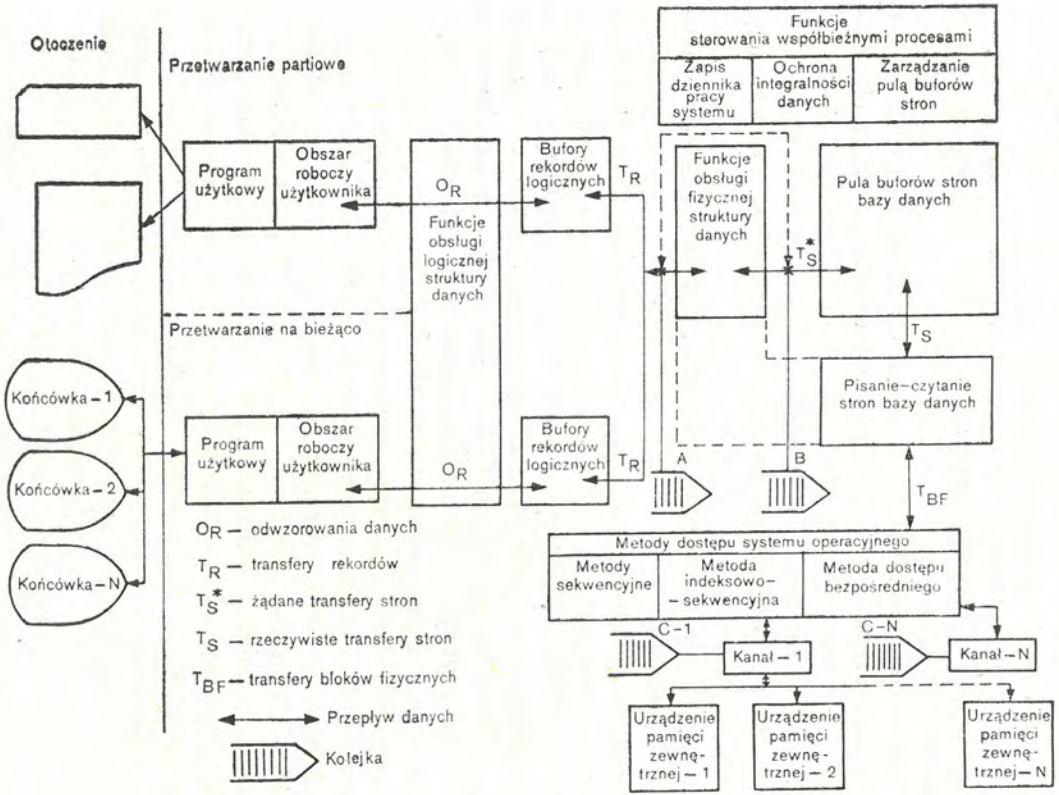
Kojarzenie decyzji projektowych z algorytmami ich wykonania oraz związanymi z nimi parametrami eksploatacyjnymi pozwala na wybór optymalnych rozwiązań. Sytuację komplikuje dodatkowo konieczność wyboru kompromisowych rozwiązań wynikających z różnych, często sprzecznych, wymagań systemów zastosowań realizowanych na podstawie wspólnej bazy danych. Obciążenie poszczególnych elementów konfiguracji komputera wynikające z działania poszczególnych grup funkcji SZBD będziemy rozważali z punktu widzenia trzech podstawowych parametrów eksploatacyjnych, a mianowicie: zajętość pamięci operacyjnej przez bufory danych i elementy oprogramowania systemu, obciążenie jednostki centralnej wynikające z wymagań algoryt-

mów poszczególnych funkcji systemu oraz obciążenie urządzeń pamięci zewnętrznej.

Każdy z tych trzech parametrów eksploatacyjnych ma wpływ na możliwości eksploatacyjne systemu informatycznego. Zajętość pamięci operacyjnej wpływa bezpośrednio na liczbę współbieżnych zadań (praca wieloprogramowa) możliwych do realizacji na danym zestawie sprzętu komputerowego, natomiast obciążenie jednostki centralnej i urządzeń pamięci zewnętrznej wpływa na czas wykonania zadań użytkowych.

Wpływ poszczególnych parametrów eksploatacyjnych SZBD na eksploatację systemu informatycznego jest różny dla różnych konfiguracji sprzętu komputerowego. Istotnymi elementami z punktu widzenia sprzętu komputerowego są takie parametry, jak pojemność pamięci operacyjnej, cykl pamięci i cykl jednostki centralnej oraz czasy wykonania operacji wejścia-wyjścia dla urządzeń pamięci zewnętrznej. Omawiając poszczególne grupy funkcji systemu ZBD zwrócimy szczególną uwagę na ich wpływ na omówione już parametry eksploatacyjne. Schemat organizacji systemu ZBD przedstawia rysunek 56.

Przedstawiony schemat jest wystarczająco ogólny, by mógł być ilustracją rozwiązań organizacyjnych, stosowanych w większości znanych systemów ZBD. Dla uniknięcia zbędnych komplikacji pominięto elementy systemu związane z obsługą teleprzetwarzania oraz języka bezpośredniego użytkownika. Programy użytkowe napisane w bazowym języku programowania odwołują się do systemu ZBD przez komendy Języka Manipulacji Danymi. Dane wykorzystywane przez te programy są umieszczone w obszarze roboczym użytkownika, będącym integralną częścią programu użytkowego. Postać danych jest zgodna z opisem struktury danych użytkownika i są one dostępne przez automatycznie włączone do programu użytkowego deklaracje. Istotną cechą systemu ZBD są dopuszczalne różnice między postacią elementów struktury danych użytkownika a postaciami odpowiadających im elementów logicznej struktury danych. Definicja struktury danych użytkownika może obejmować takie zagadnienia, jak wybór podzbioru elementów logicznej struktury danych, transformacja elementów logicznej struktury danych oraz zmiana nazw tych elementów. Typowy dla współczesnych rozwią-



Rys. 56. Schemat organizacji Systemu Zarządzania Bazą Danych

zań jest następujący zakres możliwości selekcji podzbioru i transformacji elementów logicznej struktury danych:

Dana elementarna:

— zmiana charakterystyki wartości danej elementarnej,

— pominięcie danej elementarnej,

— zmiana kolejności danych elementarnych,

— grupowanie danych elementarnych pod wspólną nazwą.

Rekord:

— pominięcie wszystkich rekordów określonego typu,

— pominięcie wszystkich rekordów zapamiętanych w określonym obszarze bazy danych.

Zbiór strukturalny:

— pominięcie wszystkich zbiorów strukturalnych określonego typu,

— pominięcie niektórych typów rekordów członkowskich zbioru strukturalnego.

Odwzorowania między logiczną strukturą danych a strukturą danych użytkownika są jednym z podstawowych zadań funkcji obsługi logicznej struktury danych. Elementy struktury danych są przesyłane z obszaru pamięci operacyjnej przeznaczonego na bufory rekordów logicznych do obszaru roboczego użytkownika w wypadku czytania danych z bazy danych i odwrotnie — w wypadku zapisu danych do bazy danych. W toku takiego przesyłania są wykonywane wszystkie konieczne konwersje wartości danych elementarnych oraz wymagane zmiany wewnętrznej struktury rekordów. Odwzorowanie danych (OR) może powodować poważne obciążenie jednostki centralnej komputera. Stopień obciążenia jednostki centralnej jest uzależniony od ilości oraz typów konwersji wartości danych elementarnych oraz od różnic występujących w wewnętrznej strukturze rekordów. W wypadku systemów informatycznych, w których przeważa przetwarzanie zorientowane na jednostkę centralną (przewaga obliczeń numerycznych nad operacjami wejścia-wyjścia), dodatkowe obciążenie wynikające z realizacji odwzorowań danych może doprowadzić do istotnego pogorszenia parametrów eksploatacyjnych.

Przedmiotem działania funkcji obsługi logicznej struktury danych są rekordy logiczne oraz zbiory strukturalne. Rekordy lo-

giczne mogą być przedmiotem operacji czytania, zapisu, skreślania oraz modyfikacji. Również zbiory strukturalne mogą być modyfikowane przez włączanie i wyłączanie rekordów członkowskich. Wszystkie te operacje wymagają czytania wielu dodatkowych rekordów logicznych. Na przykład wyszukanie określonego rekordu członkowskiego zbioru strukturalnego może obejmować wiele czytań rekordów uczestniczących w tym zbiorze strukturalnym, koniecznych dla porównania argumentów selekcji z wartościami danych elementarnych zawartych w tych rekordach. Liczba operacji zapisu i czytania rekordów logicznych (oznaczona T_R) wynika z obsługiwanych komend Języka Manipulacji Danymi, algorytmu programu użytkowego oraz opisu logicznej i fizycznej struktury danych. Liczba transferów rekordów ma więc istotny wpływ na obciążenie urządzeń pamięci zewnętrznej. W większości wypadków właśnie obciążenie urządzeń pamięci zewnętrznej decyduje o parametrach eksploatacyjnych systemu informatycznego.

Przedmiotem działania funkcji obsługi fizycznej struktury danych są takie elementy tej struktury, jak strony bazy danych, łączniki adresowe, indeksy oraz tablice kluczy bazy danych. Zapis i czytanie rekordów powodują wykonanie odpowiedniej liczby operacji czytania i zapisu stron bazy danych, przy czym ta liczba zależy przede wszystkim od rozmieszczenia rekordów na stronach. Podobnie operacje na zbiorach strukturalnych wymagają wykonania operacji modyfikacji, tworzenia lub skreślania odpowiednich łączników adresowych. Odwołania do stron bazy danych wynikające z działania funkcji obsługi fizycznej struktury danych (oznaczona przez T_S) nie zawsze powodują operacje czytania lub zapisu na urządzeniach pamięci zewnętrznej. Wynika to z działania funkcji zarządzania buforami stron, a więc z możliwości przechowywania w puli buforów tych stron, do których są wykonywane wielokrotne odwołania.

Działanie takiego mechanizmu może wpłynąć na istotne podniesienie parametrów eksploatacyjnych przez zmniejszenie obciążenia urządzeń pamięci zewnętrznej. Stosunek liczby żądanych transferów stron bazy danych (T^*_S) do liczby rzeczywiście wykonanych transferów (T_S) jest miarą efektywności działania funkcji zarządzania pulą buforów stron. Projektując eksploatację syste-

mu informatycznego należy tak dobierać wielkość puli buforów stron, aby iloraz $\frac{T^*_S}{T_S}$ był jak największy.

W większości znanych systemów ZBD, strona bazy danych odpowiada rekordom fizycznym wykorzystywanych metod dostępu systemu operacyjnego. W tej sytuacji liczba rzeczywistych odwołań do stron (T_S) jest równa liczbie operacji wejścia-wyjścia (T_{BF}) metod dostępu systemu operacyjnego.

Funkcje sterowania współbieżnymi procesami obejmują ochronę integralności danych polegającą m.in. na synchronizacji wykonania tych procesów przez nakładanie blokad na elementy struktury danych. Blokowanie elementów struktury danych może prowadzić do powstawania kolejek odwołań do tych elementów.

Oznaczone na rysunku 56 kolejki A i B są alternatywne, przy czym pierwsza może wystąpić, jeżeli blokady są nakładane na rekordy logiczne, a druga — jeżeli blokowane są strony bazy danych. Powstawanie kolejek odwołań do elementów struktury danych może doprowadzić do zakleszczenia oczekujących programów użytkowych. Długość takich kolejek oraz liczba występujących zakleszczeń zależą przede wszystkim od stopnia interferencji wykonywanych współbieżnie procesów użytkowych. Duża liczba operacji wejścia-wyjścia może spowodować powstawanie kolejek odwołań do kanałów sterujących działaniem urządzeń pamięci zewnętrznych. Długość takich kolejek zależy przede wszystkim od parametrów technicznych tych urządzeń oraz ich konfiguracji.

Istotną cechą systemu ZBD jest wykorzystanie pamięci operacyjnej przez oprogramowanie poszczególnych grup funkcji. W znacznej większości systemów to oprogramowanie jest tak skonstruowane, by obsługa wielu współbieżnych programów użytkowych nie powodowała powielania się wielu kopii tych samych elementów oprogramowania w pamięci operacyjnej.

Zakres oraz sposób wykonania operacji na elementach logicznej i fizycznej struktury danych wynika bezpośrednio z zasad implementacji konkretnego systemu ZBD. Istotny wpływ ma również przyjęty w danym systemie logiczny oraz fizyczny model danych.

Duże podobieństwo stosowanych rozwiązań występuje w zakresie funkcji sterowania współbieżnymi procesami. Ze względu na duże znaczenie tego zagadnienia dla możliwości eksploatacyjnych Systemu Zarządzania Bazą Danych poświęcimy mu pozostałe punkty tego rozdziału.

2. Sterowanie współbieżnymi procesami

a. Zagrożenie integralności danych

Centralizacja funkcji przechowywania i wykorzystania danych wynikająca z zastosowania Systemów Zarządzania Bazą Danych doprowadziła do powstania wielu istotnych problemów związanych z ochroną integralności danych.

W wypadku tradycyjnych technik przetwarzania opartych na wielu nie powiązanych między sobą zbiorach danych, skutki utraty integralności danych są znacznie mniej poważne ze względu na ich lokalny zasięg (zwykle w ramach jednego zastosowania) oraz możliwości łatwiejszego odzyskania prawidłowych wartości danych.

Baza danych jest zwykle jedynym i podstawowym zasobem informacji, a jej zniszczenie może w wypadkach skrajnych uniemożliwić prawidłowe funkcjonowanie wielu działających na jej podstawie systemów informatycznych. Pojęcie ochrony integralności bazy danych obejmuje:

— ochronę istnienia bazy danych przez fizyczne zabezpieczenie, kopie oraz środki odtwarzania,

— utrzymanie jakości bazy danych przez kontrolę danych wejściowych, procedury diagnostyczne i sterowanie procesami aktualizującymi bazę danych,

— utrzymanie tajności danych przez regulację i kontrolę dostępu oraz szyfrowanie danych.

Ta bardzo ogólna definicja obejmuje zespół środków wykorzystywanych w istniejących i projektowanych systemach zabezpieczeń.

Istnieją trzy kategorie źródeł zagrożenia integralności bazy danych:

1. Przypadkowe:

— błąd użytkownika,
— błąd systemu,
2. Pasywna infiltracja:
— „podsłuch" na liniach transmisji,
— wyłapywanie impulsów,
3. Aktywna infiltracja:
— przeglądanie zbiorów danych w trybie przetwarzania na bieżąco,
— łamanie blokad dostępu,
— fizyczne przejęcie zbiorów danych,
— obejście systemu (wydruki zawartości pamięci, luki w systemie operacyjnym).

Pierwsza kategoria źródeł zagrożenia dotyczy przede wszystkim problemu jakości bazy danych, a dwie pozostałe są związane bezpośrednio z zagrożeniem tajności danych. Kierunki rozwoju funkcji Systemów Zarządzania Bazą Danych, a szczególnie postulowane przez wielu autorów podniesienia stopnia dostępności danych przez:
— bezpośredni dostęp,
— równoległą realizację procesów użytkowych,
— łatwość i elastyczność wyszukiwania,
prowadzą do powstania sytuacji poważnego zagrożenia integralności bazy danych.

Środki ochrony integralności bazy danych można podzielić na trzy grupy:
— zapobieganie wykonania niedopuszczalnych operacji na bazie danych,
— wykrywanie potencjalnych sytuacji naruszenia integralności,
— likwidacja skutków naruszenia integralności.

Mechanizmy ochrony jakości i tajności danych są podstawowymi elementami pierwszej grupy środków ochrony integralności bazy danych. Wykrycie naruszenia tajności danych, po uzyskaniu przez proces użytkowy dostępu do bazy danych, daje zwykle niewielkie szanse uniknięcia wynikających z takiej sytuacji strat pośrednich lub bezpośrednich. Naruszenie jakości danych może być wykryte za pośrednictwem procedur kontroli bazy danych lub na skutek uzyskania „nieprawdopodobnych"

wyników procesów użytkowych. Możliwości automatycznych procedur kontroli dotyczą przede wszystkim takich elementów, jak:

— konstrukcja stron,
— całość łańcuchów adresowych,
— zgodność indeksów z zawartością bazy danych.

Znacznie trudniejszym problemem jest kontrola spójności logicznej bazy danych, tj. zgodności struktury danych z regułami danego zastosowania. Możliwości likwidacji skutków naruszenia integralności bazy danych wynikają z tradycyjnie już stosowanych technik opartych na koncepcji punktów kontrolnych, pełnych i częściowych kopii bazy danych oraz algorytmów odwracania.

Poważnym ograniczeniem dla stosowanych powszechnie technik odtwarzania stanów bazy danych jest ich stosunkowo wysoki koszt związany bezpośrednio z wielkością eksploatowanej bazy danych.

Podstawą dalszych rozważań będą następujące definicje:

1. Ochrona tajności danych jest zabezpieczeniem przed nieupoważnionym dostępem do danych lub modyfikacją danych zawartych w bazie danych.

2. Ochrona jakości danych polega na zagwarantowaniu spójności logicznej i formalnej niesprzeczności danych zawartych w bazie danych.

Obie podane definicje implikują konieczność rozpatrywania każdej operacji inicjowanej przez procesy użytkowe z punktu widzenia przyjętych w określonym zastosowaniu reguł tajności oraz z punktu widzenia zagrożenia jakości danych. W wypadku ochrony tajności istnieje zwykle dobrze zdefiniowany zbiór reguł oceny dopuszczalności rozpatrywanej operacji. Reakcja mechanizmu ochrony odbywa się zgodnie z przyjętymi *a priori* regułami i polega na bezwarunkowym odrzuceniu niedopuszczalnych operacji. Podstawowym źródłem zagrożenia jakości bazy danych jest równoległe wykonanie procesów użytkowych na wspólnej bazie danych. Ocena wpływu operacji na jakość struktury danych polega na rozpatrywaniu wyniku tej operacji, z punktu widzenia operacji innych równolegle wykonywanych procesów.

Zapobieganie negatywnym skutkom wzajemnego oddziaływania równoległych procesów użytkowych polega na synchronizacji operacji na strukturze danych inicjowanych przez te procesy.

Mechanizm ochrony tajności danych spełnia dwie grupy funkcji:

— funkcje identyfikacji,
— funkcje kontroli dostępności.

Funkcje identyfikacji zawierają następujące elementy:

— identyfikację użytkownika (procesu użytkowego),
— identyfikację danych,
— identyfikację dopuszczalnych operacji na strukturze danych.

Kontrola dostępności polega na korelacji operacji na strukturze danych inicjowanych przez procesy użytkowe z zapamiętanymi opisami uprawnień procesów użytkowych i/lub inicjowanych przez te procesy operacji na strukturze danych. Funkcje identyfikacji użytkownika są realizowane zwykle w toku inicjacji procesu użytkowego na podstawie takich informacji, jak hasło, klucze dostępu, identyfikacja użytkownika, identyfikacja urządzenia itp. Funkcjami identyfikacji danych oraz opisu reguł dopuszczalności operacji na strukturze danych są elementy Języka Opisu Danych.

Podstawowymi parametrami oceny możliwości mechanizmu ochrony tajności są:

— poziom ochrony tajności,
— zakres ochrony tajności,
— wnikliwość ochrony tajności.

Poziom ochrony tajności wynika z pozycji chronionego elementu struktury danych w hierarchii typów elementów określonej klasy struktury danych. Zakres ochrony tajności dotyczy możliwości grupowania elementów struktury danych objętych wspólnym kryterium niedostępności. Na przykład zakres ochrony wyznaczony przez typ elementu struktury danych oznacza, że wspólnym kryterium niedostępności są objęte wszystkie te elementy struktury danych, które należą do określonego typu.

Wartość elementu struktury danych ogranicza zakres wspólnego kryterium niedostępności jedynie do tych elementów struktury danych, które należąc do wspólnego typu zawierają war-

tość lub wartości spełniające przyjęte dla kryterium niedostępności wyrażenie logiczne. Zakres ochrony, wynikający z kontekstu struktury danych, obejmuje wspólnym kryterium niedostępności wszystkie te elementy struktury danych, dla których jest prawdziwe wyrażenie logiczne zawierające wartości innych elementów struktury danych lub wartości elementów otoczenia.

Wnikliwość ochrony tajności dotyczy ograniczenia możliwości wykonania operacji na strukturze danych objętych wspólnym kryterium niedostępności wyznaczonym dla określonego poziomu i zakresu ochrony. I tak, wnikliwość ochrony tajności na poziomie procesu użytkowego oznacza, że wspólnym kryterium niedostępności są objęte wszystkie operacje na strukturze danych, inicjowane przez określony proces użytkowy. Wnikliwość ochrony na poziomie typu operacji na strukturze danych oznacza, że określonym kryterium niedostępności zostały objęte wszystkie wystąpienia operacji danego typu. Wzrost stopnia dostępności danych, a szczególnie konieczność stworzenia możliwości dostępu dla użytkowników bezpośrednich wymaga mechanizmów ochrony danych o dużej selektywności.

Umożliwienie bezpośredniego dostępu dla wielu użytkowników do wspólnej bazy danych spowodowało powstanie zupełnie nowych problemów w zakresie ochrony jakości danych. Nie kontrolowane działanie współbieżnych procesów użytkowych, z których przynajmniej dwa wykonują operacje aktualizujące bazę danych, może doprowadzić do naruszenia jakości bazy danych.

Mniej groźne, z punktu widzenia ochrony jakości bazy danych, jest naruszenie jakości procesu użytkowego, spowodowane równoległym wykonywaniem procesów użytkowych, z których przynajmniej jeden wykonuje operacje modyfikacji. Mamy tutaj do czynienia z sytuacją „pozornego naruszenia jakości danych", ponieważ obraz bazy danych przekazany użytkownikowi za pośrednictwem procesu o naruszonej jakości jest nieprawidłowy, mimo iż jakość danych w bazie nie została naruszona. Jeżeli jednak taki proces użytkowy o naruszonej jakości inicjuje operacje modyfikujące, może wystąpić błędna modyfikacja struktury danych prowadząca do naruszenia jakości samej bazy danych.

Problem jakości danych należy rozpatrywać na dwóch płaszczyznach:

— formalnej niesprzeczności danych polegającej na zgodności struktury danych z zasadami konstrukcji przyjętymi dla tej klasy struktury danych,

— spójności logicznej danych polegającej na zgodności struktury danych z zasadami przyjętymi w semantycznym modelu opisującym daną rzeczywistość.

Najprostszym rozwiązaniem problemu zagrożenia jakości danych przez równolegle wykonywane procesy użytkowe jest wprowadzenie zasady „wyłączności wykorzystania środków", stosowanej w wielu współczesnych systemach operacyjnych. Wynikające ze stosowania takiej zasady blokowanie elementów lub najczęściej grup elementów struktury danych prowadzi do wzajemnego blokowania się procesów użytkowych, tj. sytuacji zakleszczenia procesów. Konieczność wprowadzenia mechanizmów blokad, w wypadku równoległego wykonywania procesów użytkowych, doprowadziła do poważnego ograniczenia stopnia równoległości dostępu do bazy danych. Rozwiązania przyjęte w większości współcześnie eksploatowanych uniwersalnych Systemów Zarządzania Bazą Danych są tak istotnymi ograniczeniami stopnia równoległości dostępu, że niemożliwe jest obecnie, ze względu na uzyskiwany czas odpowiedzi, stosowanie tych systemów w wielu zastosowaniach transakcyjnych, takich jak systemy rezerwacji, systemy sterowania na bieżąco itp. Drugim istotnym mankamentem stosowanych obecnie rozwiązań jest przekazanie decyzji dotyczących ochrony jakości danych w ręce użytkowników (programistów), opracowujących procesy użytkowe przez umieszczenie odpowiednich mechanizmów (np. FREE/ /KEEP — raport DBTG) w Językach Manipulacji Danymi. Taki stan rzeczy jest w sposób oczywisty sprzeczny z przyjętą powszechnie zasadą pełnej odpowiedzialności Administratora Bazy Danych za eksploatację i utrzymanie wspólnie przechowywanych danych.

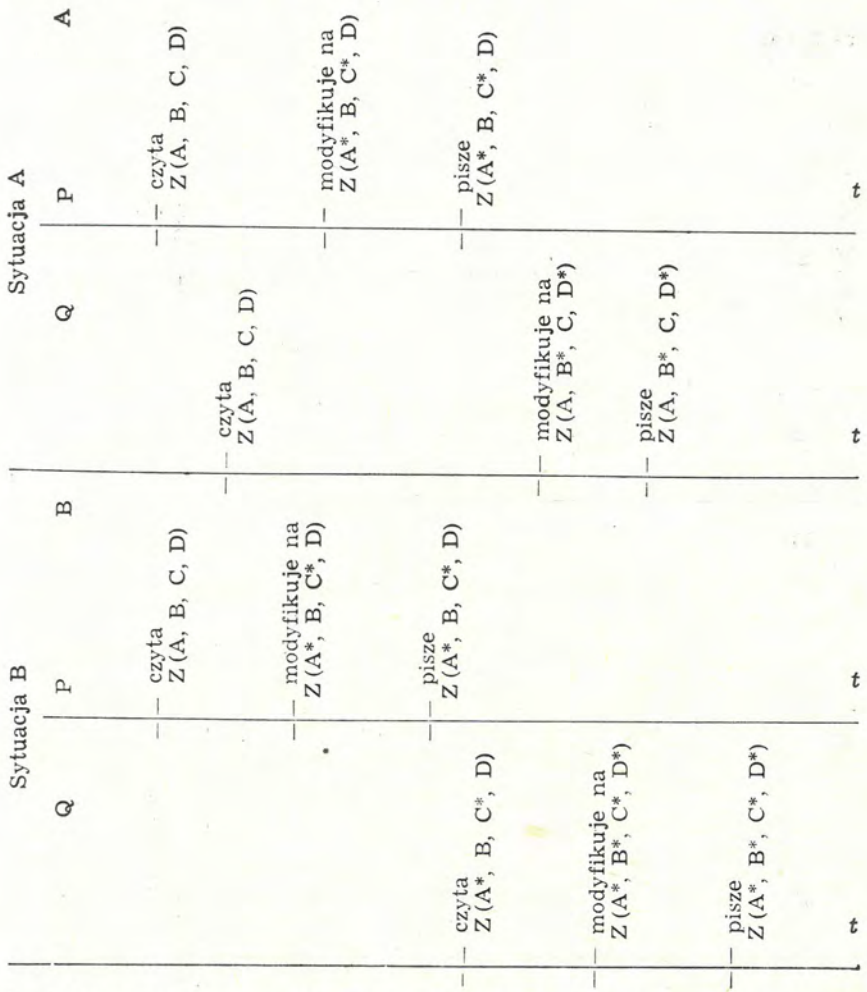
Najprostszym przykładem zagrożenia jakości danych, wynikającego z równoległego wykonywania procesów użytkowych, jest następujący wypadek. W systemie działają dwa procesy P i Q (por. tabl. 12). Oba przetwarzają (modyfikują) rekord Z zawierający dane A, B, C, D. Załóżmy następującą sekwencję zdarzeń:

— proces P czyta rekord Z (A, B, C, D);

— proces Q czyta rekord $Z(A, B, C, D)$;

— proces P aktualizuje dane A i C i żąda modyfikacji bazy danych (nowy rekord $Z(A^*, B, C^*, D)$);

Tablica 12



Przykład ilustrujący zagrożenie jakości bazy danych w wypadku dwóch procesów modyfikujących:

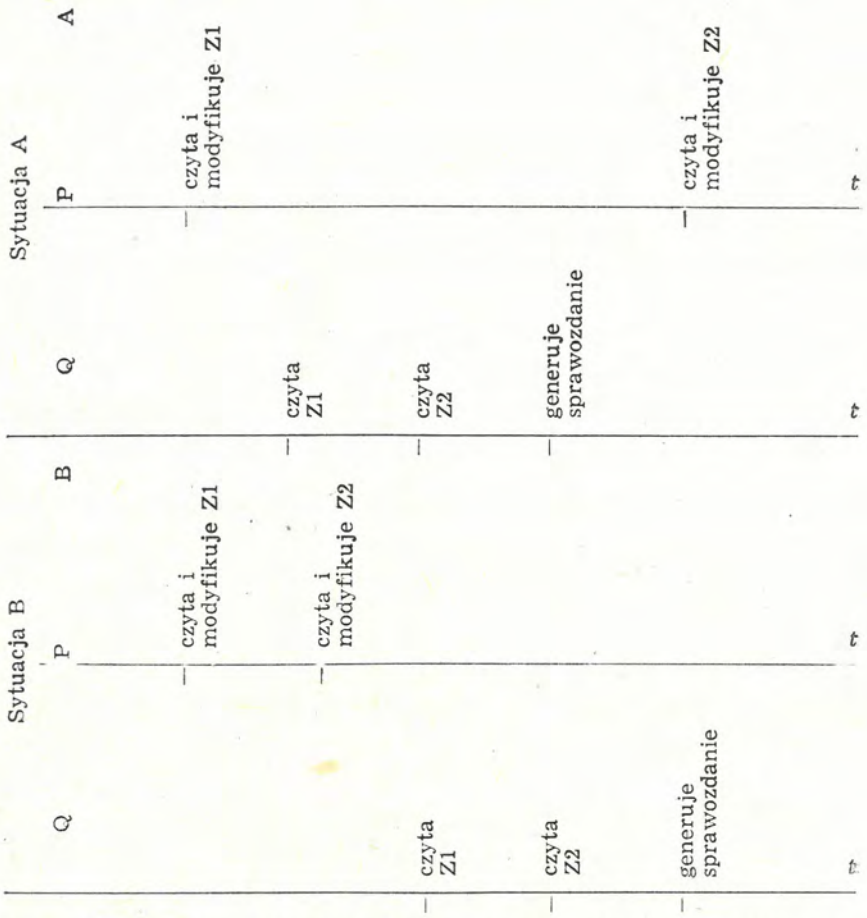
A — rekord Z został zniszczony,

B — wystarczyło zmienić kolejność zdarzeń, aby wszystko było w porządku.

— proces Q aktualizuje B i D i żąda modyfikacji bazy danych (nowy rekord — $Z(A, B^*, C, D^*)$).

W wyniku działania procesów P i Q nastąpiła zmiana w bazie danych: Z(A, B, C, D)→Z(A, B*, C, D*), a więc nie został tam

Tablica 13



Przykład ilustrujący zagrożenie jakości procesu użytkowego przez inny proces modyfikujący zawartość bazy danych:

A — sprawozdanie wygenerowane przez proces P jest błędne.

B — wystarczyło zmienić kolejność zdarzeń, aby wszystko było w porządku.

żaden ślad po działaniu procesu *P*. Jakość struktury danych została naruszona.

Inaczej wygląda problem zagrożenia integralności procesu użytkowego wynikający również z interferencji równolegle wykonywanych procesów. W systemie działają dwa procesy *P* i *Q* (por. tabl. 13). Oba działają na rekordach Z1 i Z2. Proces *P* modyfikuje zawartość tych rekordów, natomiast *Q* tylko je czyta.

Załóżmy, że rekord Z1 zawiera liczbę pracowników wydziału I, a rekord Z2 zawiera liczbę pracowników wydziału II. Proces *P* ma za zadanie zarejestrować fakt przeniesienia dwóch pracowników z wydziału I do II, a proces *Q* sporządza sprawozdanie zawierające m.in. całkowitą liczbę pracowników obu wydziałów.

Załóżmy następującą sekwencję zdarzeń:

— proces *P* czyta i modyfikuje zawartość rekordu Z1 (zmniejszenie liczby pracowników o 2),

— proces *Q* czyta rekord Z1,

— proces *Q* czyta rekord Z2 i oblicza całkowitą liczbę pracowników,

— proces *P* czyta i modyfikuje zawartość rekordu Z1 (zwiększenie liczby pracowników o 2).

W wyniku działania procesu *P* nastąpiła zmiana w bazie danych, ale jakość bazy danych została zachowana. Natomiast proces *Q* wygeneruje błędne sprawozdanie, jakość procesu użytkowego została naruszona. Sytuacja ta może prowadzić pośrednio do zagrożenia jakości danych. Jeżeli proces użytkowy o naruszonej integralności w kolejnej fazie przetwarzania modyfikuje inne elementy struktury danych, to efekty jego działania mogą być nieoczekiwane. Nieco inaczej wygląda problem ochrony jakości danych rozpatrywany z punktu widzenia zwartej sekwencji funkcji użytkowych (przejście z jednego logicznie spójnego stanu struktury danych do drugiego takiego stanu).

Przykładem zachowania formalnej niesprzeczności struktury danych przy równoczesnym złamaniu logicznej spójności jest następująca sytuacja. W bazie danych mamy dane o zatrudnieniu w wydziałach A, B, C i D. Obowiązują następujące zasady spójności logicznej:

— każdy pracownik musi pracować w jakimś wydziale,

— każdy pracownik wydziałów A i D musi mieć wyposażenie odpowiednio A' i D' (por. rys. 57).

Z punktu widzenia formalnej niesprzeczności struktury danych dopuszczalne jest wykonanie następujących akcji:

— przejście pracownika X z wydziału A do wydziału B, czyli modyfikacja związku „pracuje w" powodująca przeniesienie rekordu będącego reprezentacją pracownika X ze zbioru strukturalnego, którego właścicielem jest rekord reprezentujący wydział A do zbioru strukturalnego, którego właścicielem jest rekord reprezentujący wydział D.

Rys. 57. Struktura danych WYDZIAŁ, PRACOWNIK, WYPOSAŻENIE PRACOWNIKA

Nowy stan struktury danych jest formalnie niesprzeczny, chociaż nie jest zgodny z zasadami określonymi dla danego zastosowania (niespójny logicznie), ponieważ pracownik X pracując w wydziale D nie powinien mieć wyposażenia typu A'. Zachowanie spójności logicznej struktury danych wymaga wykonania następującej sekwencji funkcji użytkowych:

— modyfikacja zbioru strukturalnego „pracuje w" jak wyżej,

— modyfikacja rekordu „wyposażenie pracownika" (zamiana wartości „typu wyposażenia" z A' na D' oraz modyfikacja wszystkich wartości związanych z tą daną elementarną).

Przejście z jednego logicznego spójnego stanu struktury danych do drugiego wymaga wykonania kolejnych modyfikacji. Podobnie jak w wypadku naruszenia formalnej niesprzeczności struk-

tury danych, jej spójność logiczną należy rozważać zarówno pod względem naruszenia jakości danych, jak i naruszenia jakości procesu i wynikającej z tego zjawiska kontaminacji struktury danych. Podobnie ma się rzecz z ciągiem funkcji użytkowych prowadzących do spójnego logicznie wyniku i naruszenia struktury danych.

Przykładem takiego logicznie spójnego podzbioru informacji wyjściowych może być lista wyposażenia poszczególnych pracowników w ramach jednego wydziału. Jeżeli w otoczeniu bazy danych działa równoległy proces modyfikujący strukturę danych, to prawdopodobna jest następująca sytuacja.

Proces wyszukujący *W* drukuje kolejno informacje o pracownikach wydziału A wraz z informacjami o posiadanym przez nich wyposażeniu (logicznie spójny podzbiór informacji). Wymaga to wykonania kolejnych selekcji wartości z rekordów członków zbioru strukturalnego „pracuje w", którego właścicielem jest rekord będący reprezentacją wydziału A, oraz dla każdego takiego rekordu kolejnych selekcji wartości z rekordów członków, zbioru strukturalnego „posiada". Proces modyfikujący *M* przenosi rekord będący reprezentacją pracownika ze zbioru strukturalnego „pracuje w", którego właścicielem jest rekord reprezentujący wydział D do zbioru strukturalnego, którego właścicielem jest rekord reprezentujący wydział A.

Zgodnie z zasadami zdefiniowanymi dla tego zadania zwarta sekwencja zmian procesu M jest w tym wypadku następująca:

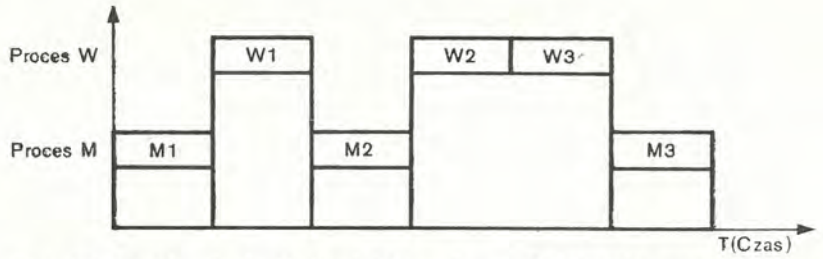
— przeniesienie rekordu „pracownik" z jednego zbioru strukturalnego „pracuje w" do drugiego,

— modyfikacja kolejnych rekordów „wyposażenie pracownika" należących do zbioru strukturalnego „posiada", którego właścicielem jest zmodyfikowany rekord „pracownik".

Rozważmy następującą kolejność realizowanych w obu procesach funkcji użytkowych (por. rys. 53).

M1 — modyfikacja rekordu „pracownik X" przenosi ten rekord ze zbioru strukturalnego „pracuje w", którego właścicielem jest rekord „wydział D" do zbioru strukturalnego „pracuje w", którego właścicielem jest rekord „wydział A".

W1 — selekcja, której przedmiotem są wartości danych elementarnych zawartych w rekordzie „pracownik X".



Rys. 58. Realizacja funkcji użytkowych współbieżnych procesów W : M

M2 — modyfikacja rekordu „wyposażenie pracownika" (np. odzież robocza) zmienia wartość danej elementarnej „typ wyposażenia" z D' na A'; modyfikacja ta ustanawia wartość pierwszego rekordu członkowskiego zbioru strukturalnego „posiada", którego właścicielem jest rekord „pracownik X".

W2 — selekcja, której przedmiotem są wartości danych elementarnych zawartych w rekordzie „wyposażenie pracownika" (np. odzież robocza).

W3 — selekcja, której przedmiotem są wartości danych elementarnych zawartych w rekordzie „wyposażenie pracownika" (np. narzędzia) będącego następnym rekordem członkowskim zbioru strukturalnego „posiada", którego właścicielem jest rekord „pracownik X".

M3 — modyfikacja rekordu „wyposażenie pracownika" (np. „narzędzia") zmienia wartość danej elementarnej „typ wyposażenia" z D' na A'; modyfikacja ta ustanawia wartości drugiego rekordu członkowskiego zbioru strukturalnego „posiada", którego właścicielem jest rekord „pracownik X".

Wynik wyszukiwania procesu *W* jest niespójny logicznie, ponieważ dane o pracowniku wydziału A zawierać będą informację, iż posiada on wyposażenie typu A' (odzież roboczą) i typu D' (narzędzia), co jest stanem niedopuszczalnym z punktu widzenia logicznej spójności bazy danych.

Łatwo zauważyć że opisane zjawisko spowodowane interferencją funkcji użytkowych W3 i M1 (selekcja wartości z rekordu członka zbioru strukturalnego „posiada" i modyfikacja właściciela tego zbioru strukturalnego, tj. rekordu „pracownik"). Gdyby proces *W* modyfikował struktury danych, mogłoby to doprowadzić do naruszenia spójności logicznej struktury danych.

b. Ochrona integralności danych

Aparat synchronizacji procesów użytkowych obejmuje mechanizm kontroli dostępu o następujących funkcjach:

— ocena dopuszczalności operacji na strukturze danych z punktu widzenia ochrony tajności danych,

— ocena bezpieczeństwa operacji na strukturze danych z punktu widzenia ochrony jakości danych,

— odrzucenie lub czasowe zablokowanie wykonania operacji na strukturze danych.

Mechanizm kontroli dostępu musi mieć następujące cechy:

— elastyczną obsługę ochrony tajności przez pełną możliwość definicji zasad dopuszczalności operacji na strukturze danych z wykorzystaniem dowolnego poziomu, zakresu i wnikliwości ochrony,

— ochronę jakości danych niezależnie od liczby i charakterystyki równoległe wykonywanych procesów użytkowych,

— niezależność synchronizacji procesów użytkowych od decyzji użytkownika,

— zapewnienie wysokiego stopnia równoległości realizacji procesów użytkowych przez selektywną strategię przydziału zasobów.

Zasady działania mechanizmu kontroli dostępu wynikają z definicji dopuszczalnej i bezpiecznej operacji na strukturze danych.

Operacja na strukturze danych jest dopuszczalna wtedy i tylko wtedy, gdy:

— przedmiot tej operacji znajduje się w strukturze danych, na której działa proces użytkowy inicjujący tę operację,

— przedmiot tej operacji znajduje się w zakresie prawa dostępu procesu użytkowego inicjującego tę operację.

Operacja na strukturze danych jest bezpieczna wtedy i tylko wtedy, gdy:

— nie istnieje równoległy proces użytkowy inicjujący taką operację, że zakres oddziaływania tej operacji zawierałby elementy struktury takie same, jak inny proces użytkowy,

— nie istnieje równoległy proces użytkowy, zawierający operację taką, że jej przedmiot znajduje się w zakresie oddziaływania operacji innego procesu użytkowego.

W wypadku złamania zasad dopuszczalności operacja na strukturze danych jest bezwarunkowo odrzucona.

Warunki określające bezpieczeństwo operacji zmieniają się dynamicznie w toku wykonywania zadania. Wobec tego inicjacja operacji nie spełniającej zasad bezpieczeństwa jest czasowo blokowana, a moment jej wykonywania zależy od zmiany sytuacji wynikającej z realizacji równoległych procesów użytkowych. Automatyczna blokada jednej z równoległych operacji modyfikujących strukturę danych ogranicza zagrożenie do poziomu zagrożenia jakości procesów użytkowych. Taka sytuacja nie eliminuje w pełni zagrożenia jakości bazy danych, ponieważ w wypadku naruszenia jakości procesu, który może modyfikować inne elementy struktury danych (np. inny typ rekordów, inny obszar bazy danych) mamy do czynienia z rozchodzeniem się błędów w bazie danych.

c. Zakleszczenia procesów użytkowych

Synchronizacja procesów na podstawie koncepcji zasobów chronionych prowadzi do wzajemnego blokowania się równolegle realizowanych procesów, tj. zakleszczeń procesów. W tej sytuacji aparat synchronizacji procesów musi zawierać mechanizmy wykrywania i rozwiązywania zakleszczeń procesów.

Rozważmy trzy możliwe podejścia do problemu zakleszczeń równolegle realizowanych procesów użytkowych.

Zapobieganie. Jest to rozwiązanie najlepsze. Polega ono na badaniu, czy kolejne zmiany stanu systemu związane z żądaniami blokowania fragmentów bazy danych mogą doprowadzić do zakleszczenia. Jeżeli takie zagrożenie istnieje, to żądanie nie zostanie spełnione w ogóle lub będzie powtórnie rozpatrzone, gdy sytuacja się zmieni.

Wykrywanie. Jeżeli system nie jest wyposażony w mechanizmy zapobiegania, można go zaopatrzyć w mechanizmy wykrywania zakleszczenia — gdy już do niego dojdzie. Po wykryciu zakleszczenia jedyną możliwą akcją korekcyjną jest cofnięcie lub wręcz usunięcie części procesów i zwolnienie zajmowanych przez nie zasobów bazy danych.

Ignorowanie. W systemie nie wyposażonym w mechanizmy

wykrywania czy zapobiegania zakleszczeniom, mogą one być wykrywane tylko z zewnątrz — przez operatora, programistę oczekującego na wykonanie programu lub przy wyłączeniu systemu. Niewiele jest jednak systemów, w których przyjęto takie pozorne w istocie rzeczy rozwiązanie problemu zakleszczeń.

W celu zapobiegania należy przede wszystkim określić warunki konieczne i wystarczające dla wystąpienia zakleszczenia. Tak więc zakleszczenie wystąpi w razie powstania następujących sytuacji:

(1) proces użytkowy ma prawo żądać wyłączności dostępu do potrzebnych zasobów,

(2) w systemie działają jednocześnie dwa (lub więcej) procesy ubiegające się o wyłączny dostęp do dwóch lub więcej zasobów,

(3) proces użytkowy może żądać wyłącznego dostępu do dodatkowych zasobów, gdy ma już przydzielone do wyłącznego użytku inne zasoby,

(4) proces nie może być zmuszony do zwolnienia przydzielonych zasobów dopóki są mu potrzebne.

Zakleszczenie procesów zachodzi, gdy istnieje zamknięty łańcuch procesów taki, że każdy proces ma przydzielone do wyłącznego użytku zasoby, jakich żąda następny w łańcuchu.

Z poprzednich rozważań wynika wniosek, że pierwszy warunek musi być spełniony dla ochrony jakości struktury danych. Natomiast zapobieganie powstaniu którejkolwiek z pozostałych sytuacji jest treścią wszystkich znanych metod zapobiegania zakleszczeniom. Metody te zostaną tu pokrótce omówione.

1. Wstępne porządkowanie procesów użytkowych. Metoda ta nie dopuszcza do spełnienia drugiego warunku wystąpienia zakleszczenia. Polega ona na tym, że Administrator Bazy Danych z zewnątrz ustala kolejność wykonywania procesów użytkowych. Jeśli którekolwiek dwa procesy zawierają konfliktowe żądania dostępu, nie zostaną one dopuszczone do jednoczesnego działania. Rozwiązanie takie może być wystarczające w systemie o przetwarzaniu wsadowym, gdzie procesy są uruchamiane z zasady w pewnej sekwencji. Nie jest to jednak dopuszczalne w systemie działającym w trybie bezpośredniego wielodostępu. Także w wypadku, gdy przetwarzanie procesu użytkowego jest sterowane przez wartości danych (*data driven*), trudne, a nawet często

wręcz niemożliwe jest określenie *a priori*, czy dane dwa procesy są konfliktowe.

2. Odbieranie przydzielonych zasobów procesowi użytkowemu, w chwili gdy system stwierdzi, że spełnienie kolejnego jego żądania przydziału doprowadziłoby do zakleszczenia, jest inną metodą zapobiegania zakleszczeniom. Po odebraniu przez system wszystkich przydzielonych poprzednio zasobów proces użytkowy powinien wystąpić z nowym kompleksowym żądaniem przydziału obejmującym zarówno już wcześniej posiadane zasoby, jak i te dodatkowe. Procedura badająca zagrożenie zakleszczeniem jest wywoływana w momencie, gdy dowolny proces żąda przydziału zasobu i proces ten ma już przydzielony co najmniej jeden zasób oraz co najmniej jeden z żądanych zasobów nie może być przydzielony natychmiast. Procedura ta ma za zadanie wykryć, czy przydział żądanych zasobów nie doprowadziłby do zakleszczenia.

3. Kolejna metoda zapobiegania zakleszczeniom polega na określaniu *a priori* kolejności, w jakiej procesy użytkowe żądać będą przydziału zasobów bazy danych. Nie dojdzie do zakleszczenia systemu, jeżeli wszystkie procesy użytkowe będą wydawały żądania przydziału zasobów bazy danych w określonej kolejności, zgodnej z pewnym globalnym uporządkowaniem elementów bazy danych. Metoda ta może być stosowana dla wąskiej klasy systemów o specyficznej strukturze bazy danych. Nie wydaje się prawdopodobne, aby zbyt często udawało się spełnić takie wymagania.

W większości wypadków metody zapobiegania zakleszczeniom nie usuwają niebezpieczeństwa permanentnego blokowania określonych procesów. Wystarczy, aby była niekorzystna sekwencja przydziałów zasobów, by pewien proces czekał praktycznie w nieskończoność na przydział żądanej kombinacji zasobów. Najczęściej spotykanym rozwiązaniem jest wprowadzenie możliwości faworyzowania niektórych procesów. Gdy system stwierdzi, że pewien proces czeka zbyt długo na przydział zasobów, załatwia jego żądanie w pierwszej kolejności.

Istnieje bardzo wiele systemów, gdzie na podstawie niezbyt wysokiego prawdopodobieństwa wystąpienia zakleszczenia, dopuszcza się do niego przez rezygnację z kosztownego mechanizmu zapobiegawczego. W momencie wykrycia takiego stanu jedynym

wyjściem jest cofnięcie jednego lub więcej procesów, w celu zwolnienia zasobów przez nie zajmowanych i umożliwienia dokończenia pracy pozostałym. W systemie takim muszą być wykonywane następujące funkcje:

— wykrywanie zakleszczenia,

— dokonanie wyboru, które procesy winny być cofnięte,

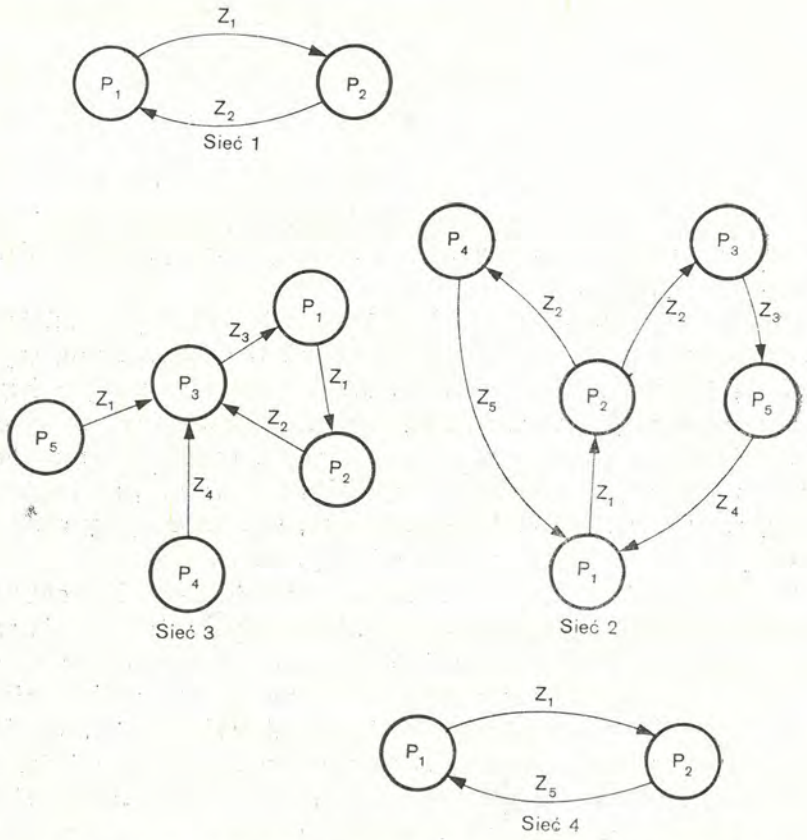
— cofnięcia niedokończonych procesów i przywrócenie bazy danych do pierwotnego stanu, tzn. stanu przed wystartowaniem cofanych procesów.

Problem wyboru procesów do usunięcia jest niezwykle istotny. Muszą to być procesy, które biorą udział w zakleszczeniu, czyli blokują i same są zablokowane. Najczęściej stosowanym kryterium wyboru jest minimalizacja kosztów systemu ponoszonych dla cofnięcia i powtórzenia przetwarzania cofanych procesów. Jedną z możliwych technik wykrywania zakleszczeń, zaproponowaną przez P. O. Marci [1], jest algorytm, wykorzystujący zorientowany graf, nazywany często siecią blokad.

Sieć blokad jest reprezentacją relacji wzajemnego blokowania równolegle realizowanych procesów. Topologiczna informacja zawarta w sieci blokad wraz z informacją dotyczącą kosztu przetwarzania poniesionego przez poszczególne procesy sterują również działaniem mechanizmu rozwiązywania zakleszczeń. Wystąpienie cyklu w sieci blokad jest warunkiem koniecznym i dostatecznym dla zakleszczenia procesów. Przykłady takich sieci przedstawiono na rysunku 59. Węzłami sieci są równolegle realizowane procesy, a łuki reprezentują relację blokowania istniejącą między dwoma procesami. Na przykład w sieci 1 proces P1 jest blokowany przez proces P2 na skutek rezerwacji przez ten proces zasobu Z1. Procesy, które nie są blokowane i nie blokują nikogo są reprezentowane przez nie połączone węzły sieci.

Na podstawie tego modelu można łatwo wykazać, że wystąpienie cyklu jest rzeczywiście warunkiem koniecznym i dostatecznym dla istnienia zakleszczonych procesów. Ponadto tylko procesy reprezentowane przez węzły zawarte w cyklu uczestniczą w rozpatrywanym zakleszczeniu. Na przykład sieć 1 jest

[1] Por. P. O. Marci, *Deadlocks Detection and Resolution in a Codasyl Based Data Management System*. W: *Proc. International Conference on Management of Data SIGMOD 76*, ACM, New York 1976.



Rys. 59. Sieci blokad procesów równolegle wykonywanych

prostą ilustracją zakleszczenia procesów P1 i P2 na skutek blokowania przez proces P1 zasobu Z2 oraz blokowania przez proces P2 zasobu Z1. Sieć 3 ilustruje bardziej skomplikowany wypadek obejmujący dwa zakleszczenia procesów (dwa cykle).

Algorytm wykrywania zakleszczenia jest wykonywany zawsze w momencie zablokowania procesu, które może spowodować zakleszczenie, tj. zablokowanie procesu, który sam blokuje inne procesy, czyli może uczestniczyć w cyklu. Zakleszczenie jest wykrywane przez przeszukanie w grafie tych ścieżek, których węzeł początkowy reprezentuje rozpatrywany proces. W tej sytuacji wszystkie istniejące zakleszczenia procesów są wykry-

wane przez identyfikację wszystkich cykli, w których uczestniczy węzeł początkowy (rozpatrywany proces).

Zastosowanie tak częstych odwołań do mechanizmu wykrywania zakleszczeń procesów jest korzystniejsze od okresowego wykonania kontroli z następujących powodów:

— wymagany jest znacznie prostszy algorytm,

— zakleszczenie jest rozwiązywane w momencie wystąpienia, co prowadzi do lepszego wykorzystania jednostki centralnej.

Rozwiązywanie zakleszczenia procesów polega na przerwaniu cyklu, reprezentującego to zakleszczenie. Przerwanie takiego cyklu polega na wycofaniu jednego z procesów do najbliższego punktu integralności lub do jego początku. Podstawowym kryterium wyboru jest w tym wypadku koszt wykonania odwrócenia procesu, obejmujący powrót do punktu, w którym nastąpiło zakleszczenie. Problemem przedstawionego algorytmu jest brak informacji o kolejności żądań zasobów zgłaszanych przez procesy użytkowe. Przyjmijmy, że zakleszczenie procesów przedstawione w sieci 2 (por. rys. 59) jest rozwiązywane przez wycofanie procesu P3 pozwalając procesom P2, P4 i P5 na kontynuację przetwarzania. Ponieważ proces P2 żąda zasobu Z5 przed zwolnieniem zasobu Z1, a zasób Z5 jest blokowany przez proces P1, nastąpi zakleszczenie przedstawione w sieci 4 (por. rys. 59). Gdyby mechanizm rozwiązywania zakleszczeń odwrócił proces P2 zamiast procesu P3, wszystkie procesy mogłyby zakończyć przetwarzanie bez wystąpienia zakleszczenia.

Informacje dotyczące sposobu i kolejności wykorzystania elementów struktury danych wynikają z algorytmów realizowanych procesów i nie są dostępne dla mechanizmu rozwiązywania zakleszczeń. Dostępność takich informacji umożliwiałoby zastosowanie algorytmów synchronizacji procesów, pozwalających na uniknięcie zakleszczeń stosowanych w systemach operacyjnych (np. algorytm Habermana) [2].

[2] Por. A. Haberman, *Prevention of System Deadlocks*, Comm. of ACM, 7, 12, 1978.

IX. Proces projektowania bazy danych

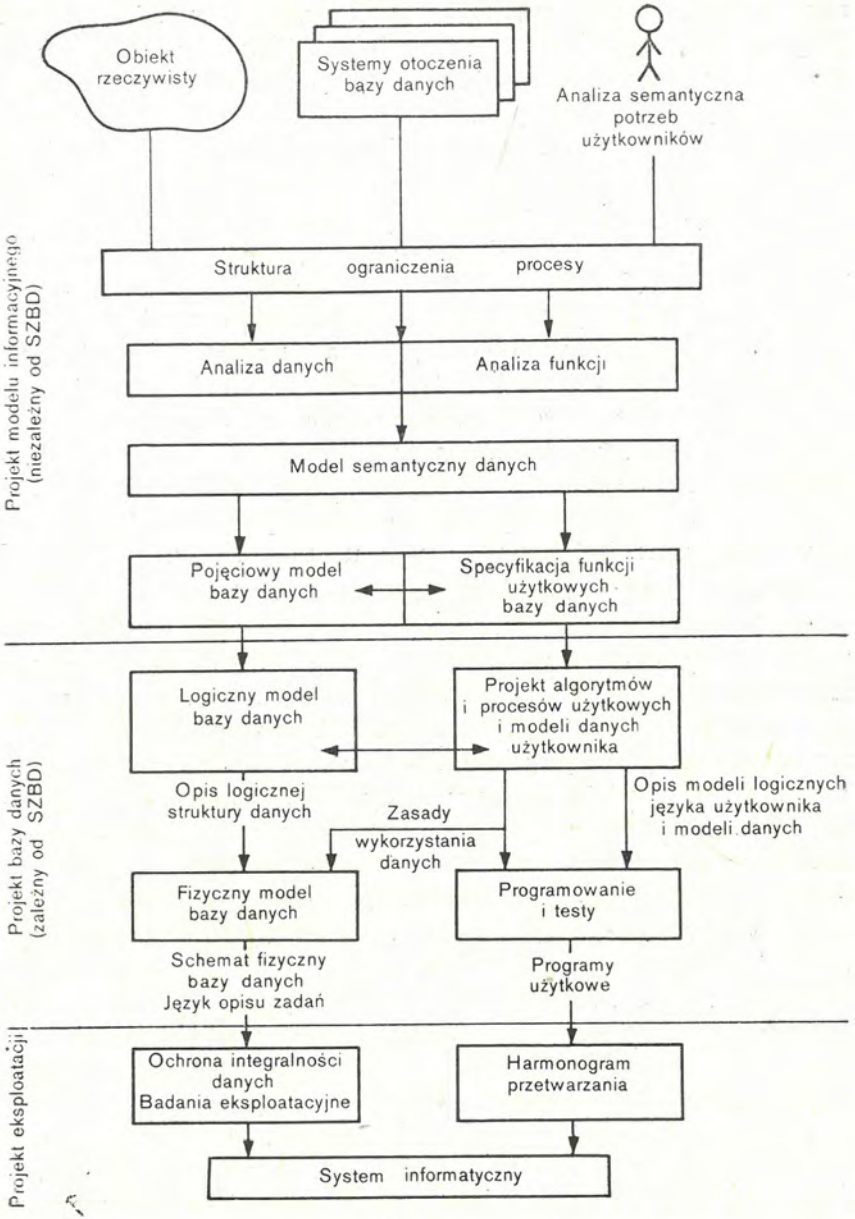
1. Ogólny schemat procesu projektowania bazy danych

Proces projektowania bazy danych jest złożonym przedsięwzięciem projektowym, obejmującym trzy „poziomy" systemu informatycznego:

— informacyjny,
— organizacyjny,
— techniczny.

Każdy z nich charakteryzuje określony stopień autonomii w toku prac projektowych. Ostatecznie jednak w kompletnym projekcie systemu bazy danych wszystkie trzy „poziomy" powinny tworzyć jedną spójną całość. Z tego względu projektant systemu bazy danych powinien wybrać jeden z tych aspektów jako wiodący, a pozostałe traktować jako ograniczenie zewnętrzne lub elementy wtórne systemu informatycznego, które powinny się dostosować do rozwiązań projektowych zastosowanych w „poziomie" uznanym za wiodący. Znaczenie tego stwierdzenia dla przebiegu procesu projektowania dalej wyjaśnimy.

Ogólny schemat procesu projektowania bazy danych przedstawia rysunek 60. Obejmuje on działania prowadzone na wszystkich trzech „poziomach" systemu bazy danych. Dosłownie interpretując ten schemat można by sądzić, że projektant bazy danych władny jest proponować rozwiązania nowe wszystkich elementów systemu informatycznego, a więc na nowo definiować i kształtować źródła danych wprowadzanych do bazy, zakres potrzeb informacyjnych użytkowników, organizację gromadzenia, przechowywania i rozpowszechniania informacji, organi-



Rys. 60. Proces projektowania systemu bazy danych

zację służb informacyjnych użytkownika finalnego, wreszcie technikę i technologię przetwarzania i przechowywania danych. W praktyce projektant bazy danych jest jednak znacznie silniej skrępowany w swoich decyzjach niż projektant tradycyjnego systemu przetwarzania danych. Jak już wspominaliśmy, z reguły nie może ingerować w systemy informatyczne zasilające bazę danych. Ingerencja ta może i powinna natomiast dokonywać się w trybie zwrotnego oddziaływania systemu bazy danych na systemy ją zasilające przez przejmowanie różnych funkcji i systemów wsadowych przez system bazy danych. Przejmowanie tych funkcji nie odbywa się w toku wdrożenia bazy danych, lecz następuje stopniowo, w procesie jej rozwoju. Stąd też projektant bazy danych musi widzieć swój projekt w aspekcie inicjatywnego, pierwszego wdrożenia systemu bazy danych, oraz w aspekcie jego rozwoju, przejmowania funkcji innych systemów informatycznych, działających w otoczeniu bazy danych, oraz wzbogacania systemu bazy danych o nowe funkcje, wynikające z rozpoznanych potrzeb informacyjnych użytkowników.

Projektant bazy danych musi więc przewidzieć, że pewne rozwiązania w obszare informacji, organizacji i technologii, niezbędne dla zainicjowania działania bazy danych, mogą okazać się niepotrzebne w dalszych etapach rozwojowych bazy danych, gdy zacznie ona oddziaływać zwrotnie zarówno na systemy ją zasilające, jak i na potrzeby użytkownika finalnego i sposoby jego obsługi. Chodzi o to, aby koszty i czas potrzebny na wprowadzanie nowych funkcji bazy danych były minimalne, aby nakłady poniesione na te funkcje bazy danych, które w dalszych etapach jej rozwoju będą zanikać, były jak najmniejsze. Chodzi też o to, by uniknąć konieczności przeprojektowywania bazy danych wówczas, gdy pojawi się potrzeba lub możliwość wprowadzenia zmian co do zakresu informacji, trybu aktualizacji, procedur przetwarzania, sposobów udostępniania wyników i organizacji obsługi informacyjnej użytkowników finalnych. Obecnie, przynajmniej w naszych warunkach, najbardziej kosztowne są rozwiązania technologiczne bazy danych. W związku z tym powinny one charakteryzować się największą elastycznością i „odpornością" na zmiany funkcji i zakresu informacji w systemie bazy danych. Doświadczenia własne oraz obserwacje „losów"

różnych baz danych uprawniają do twierdzenia, że zastosowanie uniwersalnych Systemów Zarządzania Bazą Danych jako podstawowej technologii systemu informatycznego jest bardziej opłacalne, niż rozwijanie własnego oprogramowania systemu bazy danych, uzupełnianie go stopniowo, w miarę rozwoju funkcji bazy danych. Oczywiście, o ile wykorzystujemy sprzęt informatyczny, umożliwiający efektywne eksploatowanie Systemu Zarządzania Bazą Danych.

Projektant bazy danych najczęściej nie ma wpływu na konfigurację sprzętu, na podstawie której funkcjonować będzie projektowany przez niego system. Nawet wówczas, gdy instytucja wprowadzająca bazy danych dokonuje modernizacji swojego sprzętu informatycznego lub ma możliwość wyboru ośrodka, któremu powierzy techniczną eksploatację systemu, kryteria wyboru sprzętu związane z wymaganiami bazy danych są jednymi z wielu. Najczęściej więc projektant bazy danych musi przyjmować warunki sprzętowe jako zadane. W projekcie musi zatem odpowiedzieć na pytanie: jakie rozwiązania technologiczne można zastosować w warunkach istniejącego, dostępnego sprzętu informatycznego? Nie zwalnia go to od odpowiedzi na inne pytania: jaka byłaby optymalna konfiguracja sprzętu informatycznego potrzebna do eksploatacji systemu bazy danych przy spełnianiu przezeń wszystkich zdefiniowanych funkcji użytkowych? Odpowiedź na pierwsze pytanie dotyczy najczęściej tego, jakie rozwiązania technologiczne zastosowane będą w pierwszym, inicjalnym wdrożeniu systemu bazy danych. Odpowiedź na drugie pytanie wyznacza kierunek ewolucyjnych zmian wyposażenia technicznego danej instytucji w związku z zamierzonym rozwojem Systemu Bazy Danych.

Na ogół większą swobodę decyzji pozostawia się projektantowi w odniesieniu do organizacji systemu informatycznego opartego na bazie danych. Ale i tu szybko natrafia na bariery. Pierwszą jest niezależność systemów informatycznych będących otoczeniem bazy danych od samej bazy danych. Często system bazy danych korzysta z informacji, pochodzących spoza jednostki organizacyjnej, dla której projektuje się bazę. Zwykle nie ma ona wpływu na zakres, formę, tryb i terminy przekazywania informacji przez te systemy zewnętrzne. Na przykład,

w bazie danych, której jedną z funkcji jest analiza rynku zaopatrzenia i zbytu danego przedsiębiorstwa lub grupy przedsiębiorstw może zajść potrzeba gromadzenia informacji pochodzących z systemu statystycznego państwa czy z komercjalnych zagranicznych systemów informacji ekonomicznej. Trudno spodziewać się, że systemy te zreorganizują swoją pracę na wniosek projektanta jednej z wielu baz danych, korzystających z zasilania tych systemów. Inną formą niezależności systemów otoczenia bazy danych od samej bazy jest obowiązek stosowania ogólnokrajowych, resortowych lub branżowych standardów informacyjnych. Standardy te muszą być respektowane również przez projektanta bazy danych, co najmniej w obszarze współdziałania bazy danych z otoczeniem, zarówno w zakresie wprowadzania do bazy danych informacji z innych systemów informatycznych, jak i w zakresie przekazywania przez nią informacji do innych systemów. Obowiązujące klasyfikacje, nomenklatury, kody, zasady identyfikacji obiektów w systemach ekonomicznych powinny być znane i respektowane przez projektanta bazy danych. Mogą one ułatwiać projektowanie. Dostarczają bowiem gotowych elementów metadanych. Stąd też ważne jest dokładne rozpoznanie wszystkich systemów bliższego i dalszego otoczenia bazy danych także i pod tym kątem, czy zastosowane w nich standardy informacyjne można wykorzystać w projekcie bazy danych, skoro rozwiązania optymalne z punktu widzenia projektowanej bazy danych nie są akceptowane przez systemy otoczenia. Przysłowie „Nie chciała góra przyjść do Mahometa, musiał Mahomet pójść do góry" obowiązuje także w projektowaniu systemów informatycznych. Nic nie pomoże narzekanie na „niedostosowanie" rejestrów ogólnopaństwowych, klasyfikacji i nomenklatur systemu statystycznego do indywidualnych potrzeb konkretnej bazy danych. Trzeba się do nich dostosować lub od nich uniezależnić, uruchamiając własne źródła zasilania bazy danych, opracowując własne standardy informacyjne, systemy metadanych i prowadząc samodzielnie ich aktualizację. Jest to jednak często prawnie niedozwolone, a zawsze bardzo kosztowne. Należy pamiętać, że koszt „stworzenia" własnego otoczenia informacyjnego bazy danych może okazać się znacznie wyższy od kosztu samej bazy.

Najczęściej projektant systemu bazy danych musi respektować ograniczenia organizacyjne. Ich szczególną formą jest organizacja obsługi informacyjnej i rutyn działania jednostki organizacyjnej, dla której projektowana jest baza danych. Teoretycznie projektant może zaproponować daleko idące zmiany organizacyjne procesów informacyjnych, co może wymagać zmian nawet w strukturach organizacyjnych i metodach pracy organizacji. Tu trzeba pamiętać o tym, że zaproponowanie zbyt głębokich zmian już w momencie inicjalnego wdrażania bazy danych może spotkać się z oporem użytkowników. W wypadku bazy danych, tzw. bariera psychologiczna jest zjawiskiem realnym i odczuwalnym. Użytkownik „uszczęśliwiany na siłę" bazą danych, która wymaga od niego przestawienia się na inne rutyny pracy, pozbawiony pełnej gestii nad dotychczas posiadanymi informacjami, wprowadza pozornego konkurenta w postaci Administratora Bazy Danych, może starać się udowodnić, że baza danych jest mało przydatna, zbędna czy nawet szkodliwa dla jego pracy. Każdy projektant, który choć raz wdrożył bazę danych, ma na tym polu większe lub mniejsze doświadczenia.

Projekt bazy danych musi więc uwzględniać spełnianie funkcji „inicjalnych", które związą użytkownika z bazą, przekonają go o użyteczności systemu bazy danych w wykonywaniu pewnych jego funkcji już w momencie pierwszego wdrożenia. Nie muszą to być funkcje o podstawowym znaczeniu. Co więcej, funkcje te mogą zaniknąć w dalszych etapach rozwoju bazy danych. Wskazane jest, aby były to funkcje, które są szczególnie pracochłonne i żmudne dla użytkownika, wykonującego je tradycyjnie.

Z dotychczasowych wywodów wynika, że w projekcie bazy danych powinniśmy przewidzieć (a) model bazy danych „inicjalny", przeznaczony do pierwszego wdrożenia, (b) modele baz danych rozwojowe, na co najmniej kilka faz modyfikacji danych, (c) model docelowy, scharakteryzowany ogólnie, w formie specyfikacji funkcji bazy danych.

Wreszcie, projekt bazy danych powinien wskazywać, w jaki sposób użytkownik orgnizować będzie funkcje Administratora Bazy Danych i funkcje kierowania rozwojem bazy. W wypadku systemów bazy danych nadzór autorski projektanta często przeciąga się poza fazę wdrożenia i próbnej eksploatacji. Istnieje

tendencja do stałego związania zespołu projektowego z bazą danych, pełnienie przez ten zespół funkcji Administratora Bazy Danych. Zmiana projektantów w administratorów nie zawsze daje dobre wyniki. Rozsądniejsze wydaje się, aby w toku opracowania projektu z zespołu projektantów wyłonił się zespół przyszłego Administratora Bazy Danych. Najlepiej tę funkcję mogą pełnić osoby o dobrym przygotowaniu merytorycznym, znające rutyny pracy i potrzeby użytkowników, mające z nimi dobry kontakt. Zwykle rekomenduje się, aby organizacja zlecająca zaprojektowanie systemu bazy danych wydelegowała do prac projektowych ludzi, którzy w przyszłości przejmą funkcje administrowania bazą danych.

Wynika stąd, że sama organizacja prac projektowych ma istotne znaczenie dla powodzenia i efektywności przedsięwzięcia, jakim jest system bazy danych.

2. Problemy wdrażania technologii baz danych

Problemy wdrażania technologii baz danych obejmują zagadnienia leżące w sferze zarządzania, jak również problemy czysto techniczne. Jedną z cech rozwoju zastosowań informatyki w ciągu ostatnich dwudziestu lat jest zupełnie niezrozumiała niechęć informatyków do uczenia się na cudzych błędach. Wdrożenie technologii baz danych daje możliwość osiągnięcia dużych efektów przy jednoczesnym ryzyku poniesienia dużych strat w wypadku braku powodzenia.

Wieloletnie już doświadczenia użytkowników systemów ZBD pozwalają na wskazanie wielu najbardziej typowych problemów stojących przed nowymi użytkownikami tych systemów. Omówienie tych zagadnień przed szczegółowymi rozważaniami na temat projektowania baz danych odpowiada chronologii rozwoju wydarzeń. Mamy nadzieję, że Czytelnik uniknie przynajmniej niektórych, jeżeli nie wszystkich, potencjalnych zagrożeń wynikających z podjęcia próby wdrożenia systemu Zarządzania Bazą Danych. Oczywiście omówione problemy nie wyczerpują całości zagadnienia, jednakże należą do najczęściej występujących.

Jednym z najbardziej istotnych czynników powodzenia jest moment podjęcia decyzji o zakupie i wdrożeniu systemu. Zarzą-

dzania Bazą Danych. Szanse powodzenia systemu informatycznego opartego na wspólnej bazie danych są w tym wypadku determinowane stopniem zaawansowania przedsiębiorstwa w dziedzinie informatyki. Klasycznym sposobem oceny stopnia zaawansowania w dziedzinie informatyki są fazy rozwoju zastosowań określone przez R. Nolana [1]. Rozwój zastosowań informatyki w przedsiębiorstwie przechodzi przez cztery podstawowe fazy: od inicjacji przez fazy rozwoju i sterowania do integracji.

Zwykle moment decyzji zmiany technologii przetwarzania danych następuje w trzeciej fazie, gdy mamy już do czynienia z zastosowaniami informatyki wspomagającymi procesy decyzyjne, to jest w fazie rozwoju, w której informatyka zaczyna przynosić widoczne efekty. W tej właśnie fazie ujawniają się zwykle dodatkowe potrzeby, które powodują konieczność konsolidacji istniejących zastosowań najczęściej z koniecznością ich przeprogramowania.

Integracja następuje w momencie wprowadzenia wspólnej bazy danych, obejmującej dane istniejące uprzednio w formie konwencjonalnych plików. Taki moment wprowadzenia technologii baz danych rokuje najwięcej szans powodzenia.

Istnieje bardzo wiele przykładów przedsiębiorstw i instytucji, gdzie podjęto przedwczesne decyzje wprowadzenia technologii baz danych. W niektórych wypadkach fakt ten następował w sytuacji zupełnego braku doświadczeń w dziedzinie przetwarzania danych. Takie próby kończyły się zwykle niepowodzeniem lub, w najlepszym razie, procesy projektowania i wdrażania systemu informatycznego przekraczały wielokrotnie przewidywane koszty. W nielicznych wypadkach szczęśliwego wdrożenia bazy danych i systemów zastosowań wystąpiła konieczność sprowadzenia specjalistów zewnętrznych dysponujących odpowiednim doświadczeniem. Niezależnie od różnego stopnia powodzenia osiąganego w konkretnych wypadkach, integracja zasobów danych wynikająca ze stosowania technologii baz danych wymaga dużej dyscypliny, bądź dobrej organizacji i ogólnego przygotowania przedsiębiorstwa i instytucji do stosowania informatyki.

Następnym problemem, ściśle związanym z wyborem momen-

[1] Por. R. Nolan, *Thoughts about the Fifth Stage Data Base Management Systems*, ACM, New York 1976.

tu wdrożenia systemu ZBD, jest właściwe przygotowanie informatyków do wykorzystania tego złożonego instrumentu. Szczególnie silnym ograniczeniem jest słabe przygotowanie potencjalnych administratorów bazy danych. Istotne różnice w projektowaniu i wdrażaniu oraz krytyczne znaczenie zagadnień projektowania i tworzenia bazy danych sprawiają, że wiedza w zakresie projektowania, wdrożenia i eksploatacji konwencjonalnych zastosowań jest w wypadku wykorzystania systemu ZBD absolutnie niewystarczająca. Sytuację pogarsza fakt, że szkolenia oferowane przez dostawców systemów ZBD mają bardzo ograniczony zakres, a w wypadku funkcji administrowania danymi dostarczają jedynie elementarnych wiadomości.

Innym nieco zagadnieniem jest wybór odpowiedniego systemu Zarządzania Bazą Danych. Dostępne obecnie na rynku oprogramowania systemu ZBD mają bardzo zróżnicowane możliwości, szczególnie w zakresie dopuszczalnych struktur danych, mechanizmów dostępu do danych oraz parametrów eksploatacyjnych. Dokonanie odpowiedniego wyboru wymaga dokładnego określenia celów i wymagań stawianych przed systemem informatycznym oraz odpowiednio dużego doświadczenia i wiedzy zespołu dokonującego takiej selekcji. W wielu organizacjach wprowadzających technologię bazy danych proces selekcji odpowiedniego systemu trwał co najmniej 6 miesięcy. W wypadkach szczególnych realizowano modelowe rozwiązania przewidywanego systemu informatycznego różnych dostępnych systemów ZBD przed podjęciem ostatecznej decyzji.

Brak odpowiedniej wiedzy i doświadczenia prowadzi często do niewłaściwego wykorzystania możliwości systemu ZBD. Istnieje wiele instalacji tych systemów, gdzie są one wykorzystywane jedynie jako rozwinięta metoda indeksowo-sekwencyjna. Nie jest to optymalne podejście do wykorzystania możliwości technologii baz danych i naturalnie, trudno jest w takim wypadku mówić o efektach wynikających z jej stosowania.

Taki stan rzeczy ujawniły badania przeprowadzone w krajach Wspólnego Rynku przez specjalnie powołany dla tego celu międzynarodowy zespół ekspertów [2]. Z badań wynikały istotne opóź-

[2] Por. *European Conference on Evolution and Implementation of Data-base Systems*, Bruksela 1979.

nienia w wykorzystaniu możliwości współczesnych systemów ZBD w stosunku do podobnych zastosowań zrealizowanych w USA. W momencie prowadzenia badań, żadne przedsiębiorstwo wykorzystujące system ZBD nie osiągnęło stopnia integracji danych przekraczającego 40% ogólnych zasobów danych. Praktycznie nie wykorzystywano narzędzi komputerowego wspomagania projektowania bazy danych, a badania eksploatacyjne przy wykorzystaniu odpowiednich funkcji systemu ZBD (monitor pomiarów eksploatacyjnych) były prowadzone w bardzo nielicznych wypadkach. Taka sytuacja wynika przede wszystkim ze słabego przygotowania kadry projektantów i programistów w zakresie technologii baz danych, a przede wszystkim z braku odpowiednio doświadczonej kadry administratorów bazy danych.

Równie istotnym mankamentem jest skracanie cyklu projektowania bazy danych przez pomijanie lub ograniczanie fazy analizy semantycznej i analizy danych. Wartość pojęciowego modelu danych dla właściwego zrozumienia treści informacji o objętej bazą danych rzeczywistości jest obecnie całkiem oczywista. Pomijanie tej fazy prowadzi do niepełnej oraz często sprzecznej wewnętrznie struktury danych, co w poważnym stopniu utrudnia wdrożenie i późniejszy rozwój bazy danych. Pomijanie analizy danych w cyklu projektowania bazą danych jest rezultatem przyzwyczajeń wynikających z technik projektowania konwencjonalnych zastosowań, w których analiza danych spełniała rolę marginesową i pośrednią w stosunku do projektowania procesów użytkowych. Należy więc, w wypadku wykorzystania systemu ZBD, realizować prace projektowe na podstawie odpowiedniej i dostatecznie ściśle zdefiniowanej metodyki projektowania. Stosowanie przestarzałych technik projektowania jest często źródłem kosztownych niepowodzeń.

Ostatnim z omawianych problemów wdrażania systemu ZBD jest brak zaangażowania kierownictwa przedsiębiorstwa w ten proces. Bardzo często problemy te są traktowane jako czysto techniczne, a tym samym nie wymagające ingerencji kierownictwa. Rzeczywiście w wypadku ograniczonego wykorzystania możliwości systemu ZBD taka ingerencja nie jest konieczna. Również efekty takiego podejścia ograniczania się do wprowadzenia li tylko nowej technologii przetwarzania. Jeżeli jednak baza danych ma

doprowadzić do rzeczywistej integracji zasobów danych organizacji oraz być istotnym elementem wspomagającym zarządzanie, aktywna postawa kierownictwa oraz nadanie odpowiedniego priorytetu całemu przedsięwzięciu jest nieodzowne.

Omówione już problemy wdrażania technologii baz danych wskazują na istotne znaczenie podejścia do tego zagadnienia. Przedwczesne decyzje, brak przygotowania oraz niski poziom kadry profesjonalnej są najczęstszą przyczyną niepowodzeń. Ważną sprawą jest również odpowiednia konfiguracja sprzętu i naturalnie poziom jego niezawodności. W wypadku niepowodzenia konieczna jest obiektywna diagnoza jego przyczyn. Zbyt często bowiem indolencję i brak przygotowania użytkowników próbuje się zasłonić, ferując krytyczne oceny systemu ZBD, który w wypadkach odpowiedniego podejścia do wdrażania technologii baz danych jest stosowany z całkowitym powodzeniem.

3. Cykl projektowania i wdrażania bazy danych

Omawiając problemy związane z projektowaniem i wdrożeniem systemów informatycznych opartych na wspólnej bazie danych zwróciliśmy uwagę na podstawowe znaczenie wyboru odpowiedniej metodyki projektowania. Ogólny zarys metodyki projektowania baz danych jest naturalnie niezależny od typu systemu zarządzania bazą danych (sieciowy, hierarchiczny, relacyjny), natomiast decyzje projektowe podejmowane w późniejszych fazach projektowania wynikają bezpośrednio z możliwości eksploatacyjnych systemu ZBD oraz charakterystyki sprzętu.

Pełny zakres prac nad systemem informatycznym obejmuje zagadnienia związane z projektowaniem bazy danych oraz problematykę projektowania, konstrukcji i wdrożenia oprogramowania użytkowego. Przedmiotem naszych dalszych rozważań będą przede wszystkim poszczególne fazy projektowania bazy danych i jedynie te elementy realizacji oprogramowania użytkowego, które są bezpośrednio związane ze specyficznymi cechami omawianej technologii przetwarzania.

Pełny cykl projektowania i realizacji systemu informatycznego opartego na wspólnej bazie danych przedstawiliśmy na ry-

sunku 60. Pełny cykl realizacji podzielono na trzy podstawowe części, a mianowicie projekt modelu informacyjnego, projekt bazy danych i projekt eksploatacji, pokazując jednocześnie odpowiadające poszczególnym fazom projektowania czynności związane z projektowaniem i realizacją oprogramowania użytkowego.

Pierwsze zastosowania Systemów Zarządzania Bazą Danych były zwykle dosyć proste, a wielkość baz danych stosunkowo nieduża. W takiej sytuacji projektanci baz danych koncentrowali się przede wszystkim na optymalizacji parametrów eksploatacyjnych, poświęcając najwięcej uwagi fazom logicznego i fizycznego modelu bazy danych oraz przygotowaniom oprogramowania użytkowego. Współczesne bazy danych charakteryzują się znacznie większym zakresem i stopniem skomplikowania struktur danych, spełniających wymagania wielu, często bardzo różnych, zastosowań.

Projektowanie takich baz danych jest zagadnieniem znacznie trudniejszym i praktycznie niewykonalnym w wypadku braku odpowiednich informacji o elementach działalności organizacji, objętych zakresem projektu. W tej sytuacji projektowanie modelu semantycznego nabrało istotnego znaczenia. Szczególnie, że w tej fazie możliwe jest włączenie użytkowników do prac projektowych lub przynajmniej uzgadniania dokumentacji.

Pierwszym krokiem analizy semantycznej jest określenie zakresu proponowanego systemu. Oznacza to identyfikację podstawowych grup informacji oraz związków między tymi grupami, jak również procesów bezpośrednio objętych systemem. Dodatkowym elementem mogą być również procesy zachodzące poza przyjętym zakresem systemu, lecz znajdujące się pod jego wpływem, tzn. wykorzystujące informacje objęte bazą danych lub dostarczające informacje do bazy. Podstawowymi źródłami informacji dla analizy semantycznej są użytkownicy finalni, działający w odpowiednich komórkach przedsiębiorstwa oraz przedstawiciele kierownictwa.

Najczęściej stosowaną — mimo wielu niedoskonałości — techniką są wywiady z odpowiednimi osobami prowadzone z zasady w sposób nieformalny, to jest bez wykorzystania żadnych formalnych notacji czy języków. Metainformacja dotycząca poszczególnych grup informacji obejmuje takie zagadnienia jak naz-

wa, wielkość, ilość wystąpień, zasady spójności logicznej, poufność oraz związki z innymi grupami informacji. W wypadku procesów rozważane są takie elementy, jak wykorzystanie informacji, częstość występowania, priorytet, zależność od innych procesów oraz występujące ograniczenia.

Przedmiotem analizy semantycznej danych jest uporządkowanie uzyskanych z poprzedniej fazy metainformacji, dokonanie odpowiedniej klasyfikacji elementów modelu pojęciowego w celu opisu jego struktury oraz wprowadzenie do tego modelu zasad spójności logicznej. Ogólnie rzecz biorąc, jest to proces porządkowania informacji uzyskanych w poprzedniej fazie przez formalizację opisu i usuwanie sprzeczności. Jednym z typowych problemów tej fazy projektowania jest wykrywanie homonimów i synonimów. Analizie danych odpowiada w proceduralnej części projektowanego systemu analiza funkcji. Wynikiem tej fazy projektowania jest charakterystyka funkcji użytkowych systemu. O zagadnieniu projektowania proceduralnej części systemu informatycznego nie będziemy mówić, a stosowane metody są obecnie powszechnie znane.

Końcowym produktem analizy danych jest pojęciowy model danych, obejmujący swą strukturą cały zakres projektowanego systemu informacyjnego. Elementem uzupełniającym, chociaż bardzo istotnym, jest wynik statystycznej analizy danych pozwalający na konieczną dla prawidłowego przebiegu dalszych faz projektu charakterystykę zawartości bazy danych. Rozważone dotychczas fazy projektowania są niezależne od Systemu Zarządzania Bazą Danych. Można sobie wręcz wyobrazić realizację dalszych prac przy wykorzystaniu konwencjonalnej technologii przetwarzania. Jednym z kryterium wyboru systemu ZBD powinna być łatwość przeniesienia informacji semantycznych zawartych w pojęciowym modelu danych do logicznego modelu danych dopuszczalnego w danym systemie. Najczęściej występującym ograniczeniem jest brak możliwości wyrażenia zasad spójności logicznej w większości współcześnie stosowanych Językach Opisu Danych. Zagadnienie wyboru systemu ZBD omówimy dalej.

Na projekt bazy danych składają się zwykle dwie podstawowe fazy: projekt logiczny, a następnie fizyczny projekt bazy danych.

Celem projektu logicznego jest przede wszystkim przeniesienie informacji zawartych w pojęciowym modelu danych wyrażonych jako opis jego struktury i występujących ograniczeń na poziom Języka Opisu Danych systemu ZBD. Problem podziału decyzji projektowych na logiczne i fizyczne jest obecnie dosyć trudny ze względu na fakt, że w wielu systemach ZBD obie grupy decyzji są objęte jednym Językiem Opisu Danych, a w sytuacjach, gdy istnieją osobne języki dla logicznego i fizycznego poziomu, podział nie zawsze był dokonany wystarczająco konsekwentnie. Dodatkowym utrudnieniem jest również fakt stosunkowo silnego uzależnienia parametrów eksploatacyjnych bazy danych już od decyzji podejmowanych na etapie projektu logicznego.

Podstawowym celem fizycznego projektu bazy danych jest znalezienie takiego odwzorowania elementów logicznej struktury danych na nośnikach urządzeń pamięci zewnętrznej, aby uzyskać najbardziej korzystne parametry eksploatacyjne dla danego scenariusza przetwarzania. Celowo unikamy określenia „optymalne parametry eksploatacyjne", ponieważ obecnie stosowane metody i techniki projektowania baz danych umożliwiają szukanie lepszych rozwiązań, natomiast trudno jest mówić o optymalizacji w ścisłym znaczeniu tego słowa.

Logiczne oraz fizyczne projektowanie bazy danych zależy bezpośrednio od systemu Zarządzania Bazą Danych. Omawiając dalej te zagadnienia skoncentrujmy się na możliwościach i problemach typowych dla systemów ZBD, będących implementacjami raportów Komitetu CODASYL. Jednym z powodów jest fakt, że na wszystkich głównych typach komputerów te systemy ZBD są obecnie dostępne, a drugim jest stosunkowo duża popularność tych zagadnień w naszym kraju, dzięki dość szerokiemu wykorzystaniu systemu ZBD RODAN.

Zakończenie prac nad projektem oraz próbną eksploatacją pilotowej bazy danych nie są zakończeniem prac nad systemem bazy danych. Bardzo istotnym dla powodzenia tego systemu jest projekt jego eksploatacji. Ponieważ baza danych integruje podstawowe zasoby danych przedsiębiorstwa, problem ich ochrony oraz dostępności może mieć kluczowe znaczenie dla funkcjonowania całej organizacji. Dostępność danych obejmuje również zagadnienie parametrów eksploatacyjnych, ponieważ przekroczе-

nie dopuszczalnych granic parametrów eksploatacyjnych jest bardzo często równoznaczne ze stratą dostępności danych. Projekt eksploatacji jest w poważnym stopniu determinowany harmonogramem przetwarzania, dostępną konfiguracją sprzętu oraz wymaganymi parametrami eksploatacyjnymi przyjętymi dla poszczególnych zastosowań.

Każda z przedstawionych faz projektowania bazy danych ma charakter iteracyjny, przy czym często wymagany jest powrót do fazy poprzedniej w celu modyfikacji odpowiednich decyzji projektowych. Iteracje zachodzą zwykle w ramach poszczególnych typów projektów, to jest w obrębie projektu modelu informacyjnego, projektu bazy danych czy projektu eksploatacji.

Na przykład trudno jest sobie wyobrazić konieczność modyfikacji decyzji projektowych podejmowanych w fazie tworzenia modelu pojęciowego, wynikającej z problemów związanych z parametrami eksploatacyjnymi, które wystąpiły w fazie fizycznego projektu bazy danych. Natomiast rozwój bazy danych wymaga zwykle pełnego cyklu projektowego w odniesieniu do nowo wprowadzonych elementów lub w zakresie proponowanych zmian.

4. Kryteria wyboru Systemu Zarządzania Bazą Danych

Jak wynika z poprzednich rozważań decyzja o zakupie systemu zarządzania bazą danych ma poważne znaczenie dla dalszego rozwoju informatyki w danym przedsiębiorstwie. Jest to krok, który wymaga szczególnej ostrożności, ponieważ przedwczesne wdrożenie systemu ZBD lub wybór nieodpowiedniego systemu może spowodować poważne straty, zarówno materialne, jak i czasowe. Również opóźnienie decyzji o wdrożeniu technologii baz danych na skutek obawy, że może to pociągnąć za sobą zbyt wysokie koszty lub spowodować zbyt duże zagrożenie, może na dłuższy okres okazać się niezwykle kosztownym błędem. Potrzeby standaryzacji przetwarzania oraz integracji danych mogą być na tyle pilne, że wszelkie decyzje opóźniające wdrożenie systemu zarządzania bazą danych spowodują znaczne podniesienie kosztów eksploatacji systemu informatycznego przy jednoczesnej degradacji jakości obsługi jego użytkownika.

Wybór systemu ZBD powinien być dokonany przy udziale specjalistów informatyków oraz finalnych użytkowników, których wiedza w zakresie informatyki może być bardzo niewielka. Decyzja o zakupie takiego systemu jest zwykle podejmowana przez kierownictwo przedsiębiorstwa, przy czym podstawą takiej decyzji powinien być wynik procesu selekcji systemu ZBD. Proces selekcji systemu ZBD obejmuje wiele etapów, przy czym czas trwania każdego z nich może wynosić od jednego do czterech miesięcy.

Pierwszym krokiem jest analiza potrzeb przedsiębiorstwa, której wynikiem musi być, m.in., odpowiedź na pytanie czy system ZBD powinien być stosowany na danym poziomie rozwoju zastosowań informatyki w przedsiębiorstwie. Zwykle wymagania dotyczące podniesienia jakości i użyteczności informacji uzyskiwanych przez użytkowników systemu informatycznego, poprawy możliwości dostępu do danych oraz wzmocnienia ochrony danych są wystarczającym uzasadnieniem decyzji o wdrożeniu technologii baz danych. Dodatkowym argumentem za instalacją systemu ZBD jest potrzeba przyspieszenia cyklu projektowania i wdrożenia nowych zastosowań. Istotnym elementem tego etapu prac jest przygotowanie użytkowników oraz kierownictwa przedsiębiorstwa do zmiany technologii przetwarzania. Podjęte na tym etapie szkolenie powinno ułatwić wybór właściwego systemu ZBD, a następnie jego wdrożenie.

Konsekwencją decyzji o zakupie systemu ZBD jest stworzenie zespołu odpowiedzialnego za wybór odpowiedniego systemu. Taki zespół powinien być reprezentacją wszystkich zainteresowanych służb przedsiębiorstwa, a przede wszystkim przyszłych użytkowników systemu informatycznego, analityków systemów odpowiedzialnych za projektowanie przyszłych systemów zastosowań oraz specjalistów w zakresie technologii przetwarzania. Pierwszym zadaniem zespołu dokonującego selekcji systemu ZBD jest zestawienie podstawowych możliwości takiego systemu, które odpowiadają wymaganiom technologicznym projektowanej bazy danych i przyszłego systemu informatycznego. Po ustaleniu listy wymaganych cech systemu ZBD należy ustalić priorytety jej poszczególnych elementów.

Dla uzyskania pełnej listy pożądanych cech systemu koniecz-

ne jest wykonanie analizy i klasyfikacji potrzeb użytkowników oraz odniesienie poszczególnych potrzeb do odpowiednich cech charakterystycznych systemu ZBD. Przykładem analizy i klasyfikacji potrzeb użytkowników jest metoda delficka [3] stosowana często w tego typu wypadkach. Metoda delficka jest szczególnie przydatna w sytuacjach dużego zróżnicowania wymagań użytkowników, co zdarza się często w wypadku selekcji systemu ZBD.

Przyjmijmy, że chcemy zbadać znaczenie pewnych możliwości technologicznych systemu dla poszczególnych grup przyszłych użytkowników. Podstawą tej metody jest przeprowadzenie badania ankietowego pewnej grupy ekspertów, będącej reprezentacją całej populacji użytkowników. Pytania zawarte w ankiecie powinny prowadzić do uzyskania odpowiedzi w formie oceny punktowej (powiedzmy od 1 do 10), co pozwoli ocenić zgodność lub niezgodność z poszczególnymi wymaganiami. Uzyskanie odpowiedzi w formie punktowej pozwoli na obliczenie średniej odpowiedzi oraz odchyleń od średniej. Typowymi dla takiej ankiety mogą być następujące pytania:

— Na ile istotne dla twojej pracy jest dysponowanie zintegrowaną bazą danych przedsiębiorstwa (wydziału, departamentu itd.)?

— Na ile istotna z punktu widzenia twojego zastosowania jest możliwość odtworzenia stanów bazy danych?

— Ile czasu można poświęcić na takie odtwarzanie (w minutach)?

— Jaki dodatkowy koszt przetwarzania (w procentach) jesteś gotowy ponieść na odtwarzanie?

Po uzyskaniu od respondentów ankiety odpowiedzi na wszystkie pytania należy przygotować nową ankietę zawierającą te same pytania co poprzednia oraz rozesłać ją do tych samych respondentów wraz ze średnimi wartościami odpowiedzi na poprzednią ankietę. Czas między dwoma ankietami powinien być wystarczająco długi, by respondent nie pamiętał odpowiedzi na pytania poprzedniej ankiety. Zwykle za trzecim razem odpowiedzi wszy-

[3] Por. M. Dalkey, *Studies in the Quality of Life Delphi and Decision Making*, Lexington Books, Lexington, MA., 1972.

stkich respondentów wykazują dużą zgodność. Jeżeli kolejne ankiety dają wyraźnie rozbieżne odpowiedzi na pewne pytania, to każda z nich musi zostać rozpatrzona dodatkowo z następujących punktów widzenia:

— Czy koszt zastosowania (lub zastosowań) odpowiadającego rozpatrywanej ankiecie ma wystarczająco duży udział w koszcie całkowitym systemu informatycznego, aby uwzględnienie specyficznych wymagań w istotny sposób zmieniało obraz całości?

— Czy reprezentowane zastosowanie zawiera jakieś „czynniki kluczowe", których pominięcie uniemożliwia jego realizację (np. specyficzne wymagania ochrony tajności danych).

Rezultatem tych rozważań może być pominięcie specyficznych wymagań określonej grupy użytkowników lub wprowadzenie takich wymagań jako obowiązujących dla selekcji systemu ZBD z punktu widzenia wymagań całości.

Następnym krokiem jest „przetłumaczenie" listy wymagań użytkowników na listę wymaganych cech systemu ZBD. Dodatkowym utrudnieniem jest fakt, że wymagania użytkowników nie zawsze korespondują w „relacji 1 : 1" z wymaganymi cechami systemu ZBD. Może się również zdarzyć, że poziom rozwiązań dostępnych we współczesnych systemach ZBD nie pozwala na spełnienie niektórych wymagań użytkowników. Ogólnie rzecz biorąc, ustalenie listy wymaganych cech systemu ZBD jest procesem iteracyjnym, a uzyskana lista jest kompromisem między wymaganiami użytkowników a możliwościami dostępnych systemów Zarządzania Bazą Danych.

Po ustaleniu odpowiednio spójnej i kompletnej listy wymaganych cech systemu ZBD można przystąpić do analizy dostępnych na danym rynku systemów. Zwykle, jeżeli lista wymagań jest wystarczająco kompletna, to lista rozpatrywanych systemów ZBD zostaje szybko ograniczona do kilku pozycji. Należy wówczas przystąpić do szczegółowej analizy technicznej rozpatrywanych systemów łącznie z wykonaniem odpowiednich pomiarów i prób eksploatacyjnych. Ostatecznym krokiem jest naturalnie instalacja i przekazanie systemu Zarządzania Bazą Danych do eksploatacji.

X. Projekt semantyczny bazy danych

1. Zakres projektu semantycznego

W wypadku projektowania bazy danych określenie treści informacji, jakie mają być gromadzone w bazie, jest integralną częścią zadania projektanta. W tradycyjnych systemach przetwarzania danych czynność ta była zaliczana do tzw. prac przedprojektowych. Projektant takiego systemu otrzymywał określenie zakresu informacji wejściowych, wyjściowych, procedur przetwarzania, źródeł danych, od użytkownika, w formie założeń systemu informatycznego. W wypadku bazy danych, która z założenia obsługiwać powinna wielu różnych użytkowników finalnych, założenia takie wymagają specjalnych analiz, badań i prac projektowych. Ich wynikiem jest określenie treści informacji gromadzonych w bazie.

Ta faza prac projektowych jest szczególnie trudna i odpowiedzialna. Po pierwsze dlatego, że na ogół projektanci systemów informatycznych nie mają szerszych doświadczeń w jej prowadzeniu. Chętnie stosują metody intuicyjne, „zdroworozsądkowe", lub nawet bagatelizują znaczenie wnikliwego opracowania projektu semantycznego bazy danych, uważając, że użytkownik jako zamawiający powinien określić sam, jakie informacje chce mieć w bazie. Słyszy się narzekania, że użytkownik „nie wie, czego chce", że stale zmienia przyjęte ustalenia. Nie ma na to innej rady, jak stwierdzić, że nieostre zdefiniowanie wymagań użytkownika jest jedną z cech systemów bazy danych. Przecież m.in. po to użytkownik decyduje się na korzystanie z bazy danych, by mógł sprawniej zaspokajać te potrzeby, które nie dają się zdefiniować *ex ante*. Dla rutynowych, stabilnych wymagań,

dobrze zdefiniowanych, wystarczyłby przecież tradycyjny system przetwarzania danych. Po drugie, faza ta decyduje o użyteczności bazy danych dla jej finalnych użytkowników. Często wysuwanym zarzutem wobec baz danych jest to, że nie można otrzymać z nich tych informacji, które w danej chwili użytkownik potrzebuje. Częściowo odczucie takie może wynikać z przekonania użytkownika o tym, że baza danych jest konstruowana po to, aby mógł on uzyskiwać wszelkie potrzebne informacje z jednego źródła. Takie uproszczone widzenie funkcji bazy danych, wpojone użytkownikom systemów informatycznych na przełomie lat sześćdziesiątych i siedemdziesiątych przez producentów sprzętu komputerowego, jeszcze dzisiaj wcale nie należy do rzadkości. Trzeci aspekt dotyczy kosztu projektowania i wdrażania bazy danych. Faza projektowania semantycznego jest stosunkowo mało kosztowna. Nawet kilkukrotne zwiększenie kosztu tej fazy zostanie ledwie zauważone w ogólnych kosztach systemu bazy danych. Faza ta realizowana jest przez niewielki liczebnie zespół specjalistów analityków, którzy znając specyficzne cechy wymagań użytkownika powinni jednocześnie bardzo dobrze znać ograniczenia i możliwości technologii baz danych, w tym systemów zarządzania bazą danych. O poprawnych wynikach tej fazy decyduje dobór zespołu o odpowiednich kwalifikacjach i umiejętnościach porozumienia się z przyszłymi użytkownikami. W tej fazie prac członkiem zespołu analityków powinien być przyszły Administrator Bazy Danych.

Projektowanie semantyczne bazy danych obejmuje następujące etapy:

- analizy potrzeb użytkowników,
- analizy źródeł danych,
- opracowania semantycznego modelu danych,
- opracowania pojęciowego modelu danych.

2. Analiza potrzeb użytkowników

W złożonych systemach informatycznych, operujących dużym zakresem informacji dla wielu użytkowników, trzeba dopuścić pewien stopień nieokreśloności potrzeb informacyjnych. Metody

analizy potrzeb użytkowników, stosowane w tradycyjnych systemach przetwarzania danych, przestają wystarczać. W systemach tych, zgodnie z powszechnie przyjętymi metodykami projektowania, punktem wyjściowym procesu projektowego było zdefiniowanie treści i formy tabulogramów wynikowych, dla których opracowywano programy wyjścia, korzystając w mniejszym lub większym stopniu z oprogramowania standardowego. Jednoznaczne zdefiniowanie postaci tabulogramu wynikowego było i jest jednym z podstawowych założeń metodycznych projektowania tradycyjnych systemów przetwarzania danych. Wzory tabulogramów były akceptowane przez użytkownika. Ich produkcja zgodnie z określonym harmonogramem jest głównym celem systemu epd.

W wypadku bazy danych założenie, że potrafimy w sposób jednoznaczny i kompletny określić zakres informacyjny i formę danych wynikowych, jest odrzucane. Wymagany stopień precyzji określenia potrzeb informacyjnych użytkowników finalnych można scharakteryzować następująco.

1. Istnieje możliwość ogólnego określenia treści informacji, o które może zapytywać użytkownik, w tym daje się określić listę danych, które powinny znaleźć się w bazie lub które można by otrzymać jako dane pochodne z bazy.

2. Można określić listę podstawowych systemów, zaliczonych do otoczenia bazy danych, w szczególności systemów zasilających bazę danych.

3. Istnieje możliwość określenia typowych form udostępniania danych wyjściowych oraz sprowadzenia różnorodnych zapytań użytkowników do ograniczonej listy typowych struktur zapytań.

4. Pewna część zapytań użytkowników daje się sprowadzić do profili obsługiwanych w trybie selektywnej dystrybucji informacji.

5. W toku użytkowania bazy danych mogą pojawić się nowe potrzeby informacyjne, które można sprowadzić do istniejących lub typowych form strukturalnych danych wynikowych i typowych zapytań; dla części tych potrzeb może okazać się konieczne wprowadzenie nowych typów zapytań lub nowych form strukturalnych danych wynikowych.

6. Zapytania wykraczające poza formy typowe są na tyle

rzadkie, że — o ile okażą się ważne — opłaca się je obsłużyć w trybie indywidualnym za pomocą indywidualnie przygotowanych programów użytkowych lub wykorzystania innych, mniej efektywnych środków programowych.

Analiza potrzeb użytkowników obejmuje dwa aspekty.

1. Określenie potencjalnego zakresu informacji, o które może zapytywać użytkownik, i potencjalnego zakresu niezbędnych procedur wyszukiwania, przetwarzania i redagowania danych wynikowych.

2. Określenie możliwej i prawdopodobnej skali zmian tych potrzeb.

Z wyników analizy w tym pierwszym aspekcie można określić zakres informacji i procedur przetwarzania, jakie powinny się znaleźć w systemie bazy danych. Drugi aspekt — umożliwia prawidłowy wybór rozwiązań technologicznych i dobór odpowiednio elastycznych środków programowych, umożliwiających aktualizację bazy danych.

Założenia te wyznaczają niezbędne granice elastyczności oprogramowania modułu wyjścia bazy danych. Jak widać, wymagają one, mimo znacznej elastyczności, dobrej znajomości nie tyle konkretnych potrzeb informacyjnych użytkowników, co zakresu informacji wynikowych, jakimi potencjalnie może się interesować którykolwiek z użytkowników bazy danych, obecnych lub przyszłych, typowych form prezentacji danych i typowych zapytań. Pożądana jest także znajomość częstotliwości pojawiania się różnych typów zapytań i różnych form prezentacji danych wynikowych. Ma to znaczenie nie tylko dla określenia funkcji oprogramowania wyjścia systemu bazy danych, ale również spełnia istotną rolę przy określaniu struktury bazy danych na niższych poziomach, zwłaszcza strategii przejścia z poziomu logicznego bazy danych na poziom fizyczny.

Prowadząc analizę potrzeb informacyjnych użytkowników bazy danych powinniśmy w pierwszym kroku dokonać identyfikacji tych potencjalnych użytkowników. Powinna to być identyfikacja konkretna, ze wskazaniem, jakie zespoły będą odbiorcami informacji emitowanymi z bazy danych bądź nawet konkretni ludzie. Niekiedy okazuje się to trudne. Wówczas trzeba sformułować możliwie konkretne hipotezy dotyczące potrzeb przewidy-

wanych użytkowników. Na przykład jeżeli projektujemy bazę danych, której zadaniem jest obsługa analityków i badaczy zajmujących się analizą rynku pracy w skali makroekonomicznej oraz w układach regionalnych, to nawet bez możliwości konkretnego wskazania osób będących użytkownikami, możemy określić, jakie dane, jak identyfikowanie w bazie, a także, jakie procedury ich przetwarzania powinny być opracowane dla obsługi ich potrzeb. Można to zrobić analizując metody i zakres do tej pory wykonywanych analiz i opracowań z tej dziedziny, uwzględniając zmiany w warsztacie pracy analityka, wynikające np. z możliwości szerszego stosowania złożonych procedur statystyki matematycznej czy programowania matematycznego, tradycyjnie rzadko stosowanych z powodu zbyt dużej pracochłonności. Uwzględnić też należy zmiany wynikające z interakcyjnego trybu pracy analityka (np. wariantowania i procedury oceny wariantów). Niedopuszczalne jest natomiast „pseudokonkretne" definiowanie użytkownika, które wprawdzie imiennie go wymienia, ale nic nie mówi o jego potrzebach. Na przykład nie jest zdefiniowaniem użytkownika bazy danych stwierdzenie, że użytkownikiem jakiejś bazy jest Komisja Planowania przy Radzie Ministrów, resort X, kierownictwo przedsiębiorstwa Z, komitet Y itp. Pamiętajmy przy tym, że we wszystkich podręcznikach projektowania systemów informatycznych mówi się o konieczności współuczestniczenia użytkownika w projektowaniu systemu. Wątpliwe, czy tak ogólnie zdefiniowany użytkownik, jak komisja, resort, kierownictwo, może współdziałać w charakterze współprojektanta. Tak określony użytkownik jest wewnętrznie niejednolity, składają się nań zespoły użytkowników indywidualnych, które mogą mieć różne zarówno nakładające się, jak i wykluczające się potrzeby co do zakresu informacji, częstotliwości aktualizacji, sposobu identyfikacji, szczegółowości. Niebezpieczne jest przyjęcie założenia, że suma potrzeb informacyjnych użytkowników indywidualnych składających się na takiego użytkownika zbiorowego da nam określenie potrzeb tegoż użytkownika zbiorowego. Oznaczałoby to bowiem przeniesienie całej redundancji wynikającej z różnicy indywidualnych potrzeb informacyjnych do bazy danych. Stąd też, analizując potrzeby użytkowników w organizacjach o dużej „złożoności informacyjnej", powinniśmy

w fazie analizy dążyć do stypizowania i ujednolicenia potrzeb informacyjnych.

Żądanie to może się wydawać wygórowane. Podkreślaliśmy przecież, że system informatyczny powinien w pierwszej kolejności odwzorowywać, a dopiero w drugiej kształtować potrzeby informacyjne i formy ich zaspokajania. Jest to sprzeczność pozorna. Prace nad ujednoliceniem i typizacją potrzeb informacyjnych polegać powinny bowiem nie na zmuszaniu użytkownika do redefiniowania ich, określania na nowo, czy zmiany. Chodzi o proste i żmudne zabiegi, polegające na tym, aby np. nazwy tych samych wskaźników ekonomicznych były rzeczywiście takie same i nie różniły się nieistotnymi detalami, aby tam gdzie zgodnie z wymaganiami powinny być stosowane jednolite nomenklatury, klasyfikacje czy systematyki, były one rzeczywiście używane, aby przestrzegać jednolitych zasad kodowania, symboliki mnemonicznej, aby ujawnić ewidentne dublowanie informacji w różnych systemach należących do przyszłego otoczenia bazy danych i zdecydować się na jedno tylko źródło danych dla projektowanej bazy. Ta ostatnia forma redundancji wynika zwykle z tego, że każdy z systemów informatycznych tworzących otoczenie przyszłej bazy danych jako system autonomiczny musi dysponować kompletem informacji wejściowych, gromadzonych we własnym zakresie. W wypadku scalania zbiorów pochodzących z różnych systemów otoczenia w jednej bazie danych, istnieje najczęściej możliwość rezygnacji z tej formy redundancji. Do eliminowania redundancji należy podchodzić rozważnie. Chodzi o to, by usuwając ją nie zniszczyć możliwości kontroli i korekty danych.

Innym, trudniejszym do wyeliminowania zjawiskiem, w fazie analizy i syntezy potrzeb informacyjnych, jest usunięcie tzw. zbędnej różnorodności informacyjnej. Przez zbędną różnorodność informacyjną rozumiemy występowanie w różnych systemach informatycznych tworzących otoczenie bazy danych informacji, które z punktu widzenia swojego pola znaczeniowego różnią się bardzo niewiele, w każdym razie nie na tyle, aby różnice te czymkolwiek uzasadnić, np. potrzebami analizy, specyficznymi cechami opisywanych obiektów itp. Zbędna różnorodność pojawia się wtedy, gdy systemy autonomiczne otoczenia bazy

są aktualizowane, zwłaszcza w obszarze metadanych. Aktualizacja ta dotyczy prawie zawsze tych elementów, które wiążą się z potrzebami użytkowników konkretnego systemu. Zapomina się wtedy o wcześniejszych ustaleniach ujednolicających. W imię wąsko pojętego „optymalnego dostosowania" systemu informatycznego do potrzeb użytkownika usuwa się elementy identyfikacyjne zapewniające spójność z innymi systemami, wprowadza nieuzgodnione z innymi zmiany rejestrów i słowników, metod identyfikacji. Po pewnym czasie dwa systemy informatyczne, które w momencie swych „narodzin" były spójne, tę spójność tracą.

Ponieważ do projektowania wspólnej bazy danych organizacja dojrzewa na ogół po kilku, a nawet po kilkunastu latach eksploatacji prostych systemów, dezintegracja odcinkowych systemów przetwarzania danych jest zwykle dobrze zaawansowana. Nie znaleziono do tej pory żadnych skutecznych instrumentów przekonania projektantów i użytkowników systemów przetwarzania danych, że należy przestrzegać określonych ogólnych zasad, z ich punktu widzenia zbędnych lub niewygodnych, w imię dość odległej integracji w przyszłej bazie danych. Tak więc przystępując do projektowania bazy danych, musimy uwzględniać w analizie wyłowienie wszystkich niespójności narosłych w czasie eksploatacji systemów odcinkowych, które nazwaliśmy zbędną różnorodnością informacyjną. Występuje ona zarówno w obszarze danych, jak i metadanych. Szczególnie niebezpieczną formą zbędnej różnorodności jest zróżnicowanie zakresów treściowych poszczególnych danych lub metadanych, które nie znajdują odzwierciedlenia w nazwach wskaźników lub obiektów ani też w zasadach ich identyfikacji. Na przykład zmiana w nomenklaturze wyrobów może polegać na zmianie sposobu zaliczania wyrobów do konkretnej pozycji, a nie na zmianie samej pozycji w nomenklaturze. Zmiana zakresu treściowego nazwy „Liczba pracowników przedsiębiorstwa X" może polegać na zmianie definicji pojęcia „pracownik", np. przez włączenie lub wyłączenie pracowników związanych umowami o pracę nakładczą czy chałupniczą, włączenie lub wyłączenie tzw. zorganizowanej siły roboczej, pracowników dorywczych i sezonowych itp., jak i pojęcia „przedsiębiorstwo", np. wskutek przyłączenia czy oddzie-

lenia się pewnych zakładów. Zmiany te nie znajdują odzwierciedlenia *explicite*. Dla integralności semantycznej bazy danych mogą mieć za to znaczenie wręcz zasadnicze.

W opracowaniach na temat projektowania systemów informatycznych wymienia się dwie metody analizy: wywiad i analiza istniejącego systemu informacyjnego. Zalecana jest najczęściej metoda wywiadu w różnej formie: wywiadu bezpośredniego, ankiety prowadzonej przez ankietera lub wypełnianej przez badanego użytkownika. W analizie dla potrzeb projektowania wspólnej bazy danych metoda ta ma jedną zaletę: umożliwia zrzucenie odpowiedzialności na chybioną zawartość i funkcje bazy danych na użytkownika, który przecież sam określa, co potrzebuje; i jedną wadę: nie pozwala na określenie rzeczywistych potrzeb informacyjnych użytkownika, rejestrując część jego potrzeb w danej chwili. Nie odrzucamy jednak metody wywiadu jako bezużytecznej. Jest ona naszym zdaniem mało przydatna do udokumentowania zapotrzebowania na informacje. Natomiast może okazać się niezastąpiona jako źródło wiedzy projektanta o zainteresowaniach użytkownika, metodach jego pracy, kierunkach rozwoju potrzeb informacyjnych, ewentualnych ograniczeniach stosowalności narzędzi informatycznych. Zatem wywiad uważamy za metodę poszerzenia wiedzy projektanta o użytkowniku, dla którego projektuje bazę danych, okazję do nawiązania z nim kontaktu i wciągnięcia do współpracy tych osób, które mogą okazać się kompetentnymi i wyrozumiałymi współprojektantami, reprezentującymi interesy użytkownika. Tacy użytkownicy są cenni w późniejszych fazach projektowania bazy danych jako opiniodawcy dokumentacji projektowej. Dotyczy to zwłaszcza faz opracowania modelu semantycznego i schematu pojęciowego bazy danych oraz leksyki i gramatyki języka użytkownika finalnego, a także organizacji wdrażania i eksploatacji bazy danych.

Jako podstawę analizy potrzeb informacyjnych użytkowników przy projektowaniu bazy danych widzimy — mimo wszystko — analizę istniejących systemów informacyjnych, zwłaszcza informatycznych systemów, będących potencjalnym otoczeniem przyszłej bazy danych. Analiza ta dotyczyć powinna przede wszystkim obszaru metadanych oraz zasad funkcjonowania systemów otoczenia. Chodzi tu o porównanie rejestrów stosowanych

w poszczególnych systemach informatycznych, zbadanie spójności i rozbieżności między nimi w zakresie:
— sposobu definiowania obiektów, ich zakresu,
— zasad formułowania nazw obiektów,
— zasad identyfikowania obiektów, w tym ich klasyfikowania,
— formalnej budowy identyfikatorów.

Na przykład mogą występować różnice w wymienionych aspektach w rejestrze wyrobów sprzedawanych przez przedsiębiorstwo, nabywanych przez przedsiębiorstwo i produkowanych dla potrzeb wewnętrznych produkcji jako części zamienne. Tworząc wspólną bazę danych należałoby stworzyć jednolity rejestr materiałów, wyrobów i części zamiennych, wprowadzając jedynie wyróżnienie funkcji takiego obiektu w systemie produkcyjnym przedsiębiorstwa. W takim wypadku analiza powinna polegać na porównaniu zasad tworzenia rejestrów, zaproponowaniu jednolitych zasad uwzględniających niezbędne potrzeby trzech obszarów, które teraz ma obsługiwać wspólna baza danych.

Niezbędne jest także porównanie słowników poszczególnych systemów informatycznych. W tym wypadku niezbędne jest utworzenie jednego wspólnego słownika dla całej bazy danych. Stąd też analiza ta powinna być szczególnie wnikliwa, koncentrując się na wskazaniu ewidentnego dublowania informacji oraz tzw. zbędnej różnorodności, ujednoliceniu zasad identyfikacji wskaźników, budowy ich identyfikatorów, w tym klasyfikacji. Formy strukturalne informacji, opisane w rozdziale I, wyczerpują w zasadzie wszystkie możliwości określania struktur nazw w słownikach.

Po takiej analizie i próbach ujednolicenia metadanych systemów otoczenia bazy danych projektant musi określić potencjalny zakres podmiotowy i przedmiotowy bazy danych, możliwy do przyjęcia przy aktualnym stopniu integracji podmiotowej i przedmiotowej systemów otoczenia bazy danych. Wówczas wspólnie z użytkownikiem lub jego kompetentnym reprezentantem należy zdecydować, jaki zakres bazy danych jest możliwy, sprawdzić, czy dla tak ograniczonego zakresu jest sens wdrażać kosztowny system bazy danych, wreszcie określić, jakie zmiany należałoby wprowadzić do systemów potencjalnego otoczenia bazy danych, aby można było je włączyć rzeczywiście

jako systemy tworzące otoczenie (np. zmiana słownika, rejestrów, funkcji — jakich). Nie należy również zapominać o określeniu w tej fazie, jakie funkcje konkretnych systemów otoczenia bazy danych można będzie wyeliminować dzięki wdrożeniu bazy, jakie obiekty lub wskaźniki usunąć z systemu otoczenia jako redundantne z występującymi w innym systemie odcinkowym.

Wreszcie, integralną częścią analizy jest konkretne przedstawienie zmian, jakie przechodzić będą poszczególne systemy otoczenia bazy danych w procesie jej wdrażania, a następnie rozwoju, jakie nowe systemy otoczenia trzeba uruchomić, a jakie całkowicie znikną.

Trzeba także pamiętać o przesunięciach funkcji kontroli i korekty danych między systemami otoczenia a systemem bazy danych. Na ogół projektanci baz danych chętnie przyjmują, że kontrola i korekta danych zostanie wykonana przez systemy otoczenia bazy danych. Systemy te, jako autonomiczne, mają zawsze moduł kontroli i korekty. Zwracamy uwagę, że kontrola danych przeprowadzona w ramach systemu otoczenia może okazać się niewystarczająca, jeżeli poprzestaniemy na procedurach kontroli i korekty zastanych, sprzed włączenia tego systemu do otoczenia bazy danych. Po prostu wymagania kontroli i korekty stawiane przez bazę danych mogą być znacznie szersze niż systemu autonomicznego. Jeżeli chodzi o kontrolę wartości wskaźników, wymagania te są najczęściej identyczne, natomiast rozszerzone wymagania dotyczą zwykle obszaru metadanych. Na przykład system informatyczny opracowujący dane o przedsiębiorstwach w układzie przestrzennym będzie kontrolował identyfikator przedsiębiorstwa, pomijając kontrolę innych pól: gminy, miasta, dzielnicy, województwa. System informatyczny opracowujący informacje o tychże przedsiębiorstwach, ale w układzie organizacyjnym lub ekonomicznym może pominąć kontrolę pola identyfikacji przestrzennej. Scalenie obu systemów jako otoczenia wspólnej bazy danych wymagać będzie rozszerzenia zakresu kontroli sposobu identyfikacji obu tych systemów o brakujące wzajemnie elementy.

Wynikiem analizy potrzeb informacyjnych użytkownika i źródeł danych, prowadzonych przede wszystkim przez analizę sys-

temów potencjalnego otoczenia bazy danych, są wstępnie uporządkowane wykazy obiektów i wskaźników, które stanowią potencjalną zawartość metainformacji projektowanej bazy danych oraz szczegółowy „protokół rozbieżności" identyfikacyjnych, informacyjnych, metodycznych, występujących między systemami naszego potencjalnego otoczenia. Wywiady oraz analiza dokumentów wyjściowych służą natomiast do opracowania specyfikacji wymagań języka użytkownika finalnego. Materiał ten służy za podstawę następnej fazy prac projektowych, mianowicie — opracowaniu semantycznego modelu danych.

3. Projektowanie semantycznego modelu bazy danych

Na podstawie analizy potrzeb użytkowników oraz analizy źródeł danych opracowuje się model semantyczny bazy danych, który jest podstawą pojęciowego modelu bazy danych. W opracowaniach dotyczących metod projektowania bazy danych niezbyt często wydziela się fazę modelowania semantycznego jako oddzielny etap procesu projektowania. Charakterystyczne jest jednak to, że autorzy zajmujący się metodami projektowania wielkich baz danych [1] wyróżniają modelowanie semantyczne jako oddzielny etap rac, podczas gdy specjaliści w zakresie technologii baz danych, zabierający głos w sprawie analizy potrzeb użytkowników i projektowania treści bazy danych, skłonni są uważać, że wystarczy pojęciowy model danych jako jedyny opis bazy danych czytelny dla użytkownika finalnego i odwzorowującego jego potrzeby.

Podzielamy pogląd tych pierwszych. Uzasadniają to różnice między metodami opracowania, funkcjami modelu semantycznego w procesie projektowania i użytkowania bazy danych a modelem pojęciowym oraz zawartością obu opisów bazy danych.

Wyniki analizy potrzeb informacyjnych użytkowników uzyskane przez wywiady z kadrą kierowniczą przedsiębiorstwa i jej personelem dają orientacyjny pogląd na oczekiwany przez przyszłego użytkownika zakres i formy informacji, jakie chciałby uzyskiwać z bazy danych. Na ich podstawie opracowuje się

[1] Por. Bl Nilsson, *On Models and Mapping in Data Basa Environment*, Urval 9, Stockholm 1980; R. Graves, *Semantic Modelling in Statistics Canada*, ISIS 80, Seminar, Bratislava 1980.

specyfikację funkcji bazy danych, z punktu widzenia jej użytkowników finalnych. Każdej z funkcji przypisać należy wykazy: (a) typowych zapytań użytkowników, (b) typowych form odpowiedzi na te pytania, (c) danych, które mogą być zażądane jako odpowiedź na zapytanie, (d) kryteriów selekcji i agregacji informacji oraz procedur ich przetwarzania, wykorzystywanych przy formułowaniu zapytań. Listy te tworzą leksykę języka użytkownika finalnego.

Zakres i stopień szczegółowości uzyskanych w ten sposób informacji zależą od zakresu kompetencji osób, będących ich źródłem. Zwykle informacje uzyskiwane od kierownictwa danej organizacji dotyczą szerszego zakresu potrzeb, ale uzyskany dzięki nim obraz jest dość ogólny. Natomiast informacje uzyskiwane z poszczególnych komórek funkcyjnych dotyczą poszczególnych wycinków obsługi informacyjnej, są za to dokładniejsze i bardziej precyzyjne. Nie można poprzestać na przeprowadzeniu serii odcinkowych analiz na szczeblu komórek funkcyjnych. Dopiero bowiem zestawienie obu punktów widzenia: całościowego i ogólnego spojrzenia kierownictwa i zbioru odcinkowych obrazów potrzeb przedstawionych przez oddzielne komórki wykonawcze i sztabowe, daje podstawę specyfikacji elementów leksyki języka użytkownika finalnego. W tej fazie są to wyrażenia języka fachowego, z reguły niesformalizowane.

Drugim źródłem wiedzy o zawartości przyszłej bazy danych są systemy informacyjne, tworzące otoczenie bazy danych. Są to zarówno systemy skomputeryzowane, jak i prowadzone w formie tradycyjnych kartotek ręcznych lub innych form zapisu informacji. W procesie analizy, który omówiliśmy w poprzednim punkcie, dokonujemy selekcji systemów, które potencjalnie mogą zasilać bazę danych. W procesie modelowania semantycznego opracowuje się bilans informacji. Polega on na zestawieniu z jednej strony informacji, które zostały określone w toku analizy potrzeb informacyjnych użytkownika, a z drugiej — informacji, jakie można uzyskać z systemów będących potencjalnym otoczeniem informacyjnej bazy danych.

Najczęściej zalecanym podejściem jest przyjęcie za punkt wyjścia wykazu informacji, jakich oczekuje użytkownik na wyjściu systemu informatycznego. Do każdej informacji (wskaźnika)

trzeba dobrać informacje w systemach źródłowych i ewentualnie podać procedurę przetworzenia informacji z systemów źródłowych na informację wynikową. Sposób ten pozwala na jednoznaczne określenie zakresu informacji systemów będących otoczeniem bazy danych, który powinien znaleźć się w bazie. Bilans ten pozwala również na specyfikację procedur przetwarzania danych gromadzonych w bazie na dane wynikowe. Jednak podejście takie wystarcza tylko wtedy, gdy potrzeby użytkowników dają się przedstawić w postaci wykazu konkretnych wskaźników i gdy udostępnianie informacji ma charakter rutynowy.

W wypadku systemów bazy danych zwykle potrzeby użytkowników, przynajmniej w istotnej części, nie są precyzyjnie określone. Zadaniem projektanta jest więc zminimalizowanie prawdopodobieństwa tego, że użytkownik nie otrzyma odpowiedzi na pytanie skierowane do bazy danych (BRAK DANYCH). Projektant ma do wyboru dwie wersje.

Zdarza się, choć rzadko, że system bazy danych kreuje własne otoczenie, czyli że tworzy się systemy zasilające bazę danych pod wyłączne potrzeby konkretnej bazy. Wówczas omawiane tu problemy stają się bezprzedmiotowe.

Najczęściej projektant bazy danych dysponuje ogólną informacją o potrzebach użytkowników finalnych, za to dość precyzyjnie może określić wykazy informacji, jakie można pozyskać z systemów otoczenia bazy danych.

Na podstawie takiego rozpoznania potrzeb użytkowników i systemów otoczenia bazy danych projektant, niezależnie od tego, jak dalece ogólne jest to rozpoznanie, opracowuje bilans informacji. Po stronie potrzeb użytkownika w tym bilansie trzeba się często zadowolić listą hipotetycznych typów zapytań, jakie potencjalny użytkownik może kierować do bazy danych. Informacje znajdujące się w systemach potencjalnego otoczenia bazy danych mogą okazać się niewystarczające lub występować w formie nie odpowiadającej zapytaniom użytkowników. Jeżeli w wyniku bilansu informacji stwierdzi się, że występują luki w informacjach istniejących w systemach przyszłego otoczenia bazy danych, należy w pierwszej kolejności spowodować uzupełnienie zakresu informacji gromadzonych w istniejących już systemach o dane wynikające z bilansu informacji. Dopiero w drugiej kolejności,

gdy pierwsza wersja zawiedzie, uruchamia się systemy zbierania danych specjalnie dla potrzeb projektowanej bazy.

Informacja o systemach otoczenia bazy danych występuje często w formie nie odpowiadającej potrzebom bazy danych. Rozbieżności między formą istniejącą danych a formą pożądaną w bazie mogą przyjmować następujące sytuacje:

— identyfikacja danych w systemach otoczenia jest inna niż potrzeby bazy: nadmiarowa, niepełna,

— użytkownik bazy danych potrzebuje tylko informacji przetworzonej zgodnie z określonymi sformalizowanymi procedurami, np. potrzebuje wskaźników pochodnych, wielkości uśrednionych,

— informacje w systemach otoczenia są zbyt szczegółowe, zdezagregowane w stosunku do stopnia ich agregacji, jaki wystarczyłby w bazie danych.

Pierwsza — wprowadzać do bazy danych wszystkie informacje z systemów tworzących otoczenie bazy danych, jeżeli można przypuszczać, że informacja będzie kiedykolwiek potrzebna. Praktycznie oznacza to, że w bazie lokowane będą informacje i takie, które nie będą nigdy wykorzystane. Natomiast prawdopodobieństwo udzielenia odpowiedzi na zapytania użytkowników jest duże. Druga — wprowadzać do bazy danych tylko te informacje z systemów jej otoczenia, co do których istnieje dostatecznie duże prawdopodobieństwo, że będą one potrzebne użytkownikom, zapewniając możliwość szybkiego uzupełnienia bazy danych o inne informacje występujące w systemach będących jej otoczeniem, gdy powstanie taka potrzeba. Wymaga to zastosowania rozwiązań technologicznych, zwłaszcza oprogramowania ładowania bazy danych i jej modyfikacji, aby włączenie nowych danych i nowych typów danych mogło odbywać się jako rutynowe zadanie Administratora Bazy Danych. Innym słowem, zbiory systemów należących do otoczenia bazy danych są potencjalnym archiwum systemu bazy danych. Rozwiązania organizacyjne i technologiczne powinny więc umożliwiać zarówno stałe, jak i doraźne wprowadzanie do bazy danych informacji ze wszystkich systemów zaliczonych do jej otoczenia. Jak więc widać, zdefiniowanie systemów tworzących owo otoczenie determinuje zdolności adaptacyjne bazy danych do konkretnych potrzeb użytkowników.

Jest dość oczywiste, że pierwsza wersja daje dobre wyniki

wtedy, gdy potrafimy względnie dobrze określić potrzeby użytkowników i gdy mają one w przeważającej mierze charakter rutynowy. Druga wersja sprawdza się wówczas, gdy systemy potencjalnego otoczenia bazy danych są skomputeryzowane i względnie stabilne.

Na podstawie rozpoznania, jakie relacje zachodzą między danymi w systemach otoczenia a potrzebami użytkowników opisanymi za pomocą (najczęściej) wykazu typowych zapytań, projektant definiuje model semantyczny danych, jaki będzie zastosowany w bazie. Model zależy od zastosowanej metody. Jeżeli zdecydujemy się zastosować metodę infologiczną, definiujemy obiekty, relacje między obiektami, atrybuty, relacje między atrybutami. Na poziomie modelu semantycznego definiuje się zwykle w języku niesformalizowanym — fachowym języku użytkownika, kryteria wydzielenia obiektu bazy danych, jego granic przestrzennych i czasowych. Mimo że formułowane w sposób niesformalizowany kryteria te powinny być precyzyjne, pozwalające jednoznacznie odpowiedzieć w wypadku każdej danej pojawiającej się na wejściu do bazy, czy jest to nazwa obiektu, atrybutu konkretnego obiektu, relacji itd. Kryteria te powinny być na tyle precyzyjne, aby dały się opisać w sposób sformalizowany w kolejnym etapie projektowania, jakim jest budowa modelu pojęciowego bazy danych.

Jeżeli projektant zdecyduje się zastosować metodę semiotyczną, powinien sporządzić strukturalny model semantyczny wskaźników gromadzonych w bazie danych. Podstawy obu metod omówiliśmy w rozdziale II.

Określenie obiektu bazy danych i jego atrybutów oraz relacji wymaga uwzględnienia kryteriów efektywności systemu. Dotyczą one: (a) wielkości zbiorów danych gromadzonych w bazie, (b) złożoności zapytań użytkowników, (c) czasu dostępu do danych i przetwarzania, (d) utrzymania aktualności i integralności bazy danych.

Ad (a). Często najprostszym rozwiązaniem z punktu widzenia projektanta jest takie zdefiniowanie obiektów bazy danych, by informacje o nich przed załadowaniem do bazy z systemów otoczenia nie wymagały specjalnego przetwarzania. Ułatwia to także akualizację bazy i utrzymanie jej integralności. W prak-

tyce takie definiowanie obiektów bazy danych jest sprzeczne z kryterium minimalizowania wielkości zbiorów danych przechowywanych w bazie oraz z zapewnienia łatwości formułowania zapytań przez użytkownika finalnego. Na przykład użytkownik zainteresowany uzyskaniem informacji do analiz makroekonomicznych z zakresu rolnictwa indywidualnego mógłby — w krańcowym wypadku — oczekiwać dostępu do bazy danych zawierającej jednakowe informacje o poszczególnych indywidualnych gospodarstwach rolnych, co w warunkach polskich oznacza około 3,5 miliona obiektów opisanych w bazie danych kilkunastoma informacjami. Z takiej bazy danych można by otrzymać wszystkie niezbędne dla badań makroekonomicznych grupowania danych i układy. Wystarczające jednak być może dla wielu zapytań zapewnienie dostępu do agregatów, zagregowanie informacji do poziomu jednostek administracyjnego podziału kraju: gmin. Obiektem bazy danych byłaby wówczas gmina, ściślej „rolnictwo indywidualne w gminie", a nie „indywidualne gospodarstwo rolne". 3,5 miliona obiektów możemy zredukować do 2200 obiektów (gmin), dostosowując obiekt bazy danych do potrzeb użytkownika. Założenie bazy danych liczącej wielką liczbę obiektów może okazać się technicznie niewykonalne. Wówczas zastąpienie obiektu, wynikającego z organizacji systemów otoczenia bazy danych, przez obiekt wynikający z potrzeb identyfikacji danych przez użytkownika finalnego jest skuteczną metodą. Pociąga to jednak za sobą straty informacji. Bronimy się przed nimi zwiększając liczbę cech, jakimi opisujemy „zastępczy" obiekt bazy danych. Przykładowo dla opisu indywidualnego gospodarstwa rolnego wystarczy od 20 do 30 atrybutów (wskaźników i cech identyfikacyjno-klasyfikacyjnych), ale, aby zbliżone informacje udostępnić użytkownikom na podstawie obiektu „gmina", jest niezbędne opisanie tego ostatniego za pomocą około 1000 atrybutów. Ostateczna kalkulacja w tym wypadku przemawia na korzyść wprowadzenia obiektu zagregowanego, gdyż liczba danych będzie w drugim wypadku i tak wielokrotnie mniejsza. Reguła postępowania w razie wyboru obiektu bazy danych elementarnych w formie takiej, w jakiej występują w systemach otoczenia jest następująca: powinien być to obiekt minimalny, na jaki pozwalają możliwości sprzętowe, ale po-

winien być mniejszym agregatem od tego, o który pyta użytkownik. Innymi słowy, obiekt bazy danych powinien być tak dobrany, aby typowe pytania analityczne wymagały operacji agregowania danych. Z doświadczeń wynika, że gromadzenie w bazie danych informacji zagregowanych do poziomu bezpośrednio wykorzystywanego przez użytkowników powoduje, że nie ma praktycznej możliwości nadążenia za uzupełnieniem bazy danych. Jeżeli bowiem użytkownik nie znajdzie oczekiwanej informacji w gotowej formie, to nie ma możliwości uzyskania informacji zbliżonej przez przetworzenie danych elementarnych gromadzonych w bazie.

Ad (b). Im bardziej złożona jest procedura przetwarzania informacji przechowywanych w bazie danych na formę oczekiwaną przez użytkownika, tym trudniej jest użytkownikowi finalnemu formułować poprawne zapytania do systemu bazy danych, a język oferowany użytkownikowi staje się w większym stopniu zorientowany na procedury, zamiast na problemy. Wówczas trzeba organizować w grupie Administratora Bazy Danych oddzielną służbę obsługi użytkowników finalnych. Tracimy w ten sposób ważny efekt użytkowy systemu bazy danych. Stąd też obiekt bazy danych, a konsekwentnie jego atrybuty i relacje należy definiować tak, aby język użytkownika finalnego był względnie łatwy do nauczenia się. Zależy to zarówno od wymagań użytkownika, jak i od jego przygotowania informatycznego.

Ad (c). Czas dostępu i czas przetwarzania danych jest tym dłuższy, im większa jest baza danych, a więc im bardziej elementarne gromadzi dane. Czynnik ten przemawia więc za definiowaniem obiektów bazy danych z punktu widzenia wymagań użytkownika.

Ad (d). Z reguły łatwiej utrzymać aktualność i integralność bazy danych, jeżeli gromadzi się informacje w formie identycznej lub maksymalnie zbliżonej do tych, w jakich występują w systemach otoczenia. Przyczyny są tu następujące:

— systemy otoczenia bazy danych są na ogół dość stabilne, mniej podlegają wpływowi zmiennych żądań użytkowników,

— systemy otoczenia bazy danych nie są narażone na ty-

powe zagrożenia integralności bazy danych, semantycznej, logicznej,

— gromadząc informacje w formie pierwotnej zachowujemy możliwość utrzymania integralności bazy danych dla użytkownika finalnego dzięki oprogramowaniu (procedurom) pośredniczącemu między programami użytkowymi a bazą danych (np. procedury sprowadzania danych do porównywalności),

— obiekty opisywane w systemach otoczenia, często obiekty elementarne w sensie jednostki informacji, podlegają zmianom często nieistotnym dla potrzeb użytkowników. Na przykład użytkownik żądający informacji zagregowanych dla poziomu działu gospodarki narodowej nie odczuje faktu zniknięcia lub zmiany branży pewnej liczby przedsiębiorstw w tym dziale jako dezintegrację bazy danych, gdyż jest to integralny element procedury przetwarzania danych i wynika z opisu obiektywnych procesów.

Wprowadzenie obiektów złożonych, których opis wymaga zaprojektowania oddzielnego systemu transformacji informacji z systemu otoczenia bazy danych na postać nadającą się do wprowadzenia do bazy danych, implikuje większe zagrożenie dla integralności bazy. Trzeba pamiętać o tym, że w wypadku definiowania obiektu bazy danych w sposób inny, niż zostało to zrobione w systemie otoczenia bazy danych, obowiązkiem projektanta systemu bazy danych jest uzupełnienie danego systemu otoczenia o programowanie transformujące dane na postać akceptowaną przez bazę. Częściej jednak wchodzi to zadanie do modułu ładowania bazy danych. Projektowo jest to jedno z bardziej pracochłonnych zadań projektanta bazy danych.

Jak widać, wynikiem procesu semantycznego modelowania bazy danych jest:

— wykaz typów zapytań użytkownika, w formie zdań parametrycznych sformułowanych w fachowym języku użytkownika, oraz wstępne listy parametrów, które można podstawić do tych zdań,

— słownictwo otwartego, fachowego języka użytkownika bazy danych,

— precyzyjne definicje obiektów bazy danych, ich atrybutów, relacji, w języku fachowym użytkownika,

— określenie relacji między obiektami bazy danych a obiektami występującymi w systemach otoczenia, wykaz procedur transformacji danych w systemach otoczenia na dane identyfikowane w odniesieniu do obiektów bazy danych,

— reguły ujednolicenia nazewnictwa, zasad budowy kodów itp. dla całej bazy danych i reguły aktualizacji.

Dla prostych obiektów bazy danych, posiadających niewielką liczbę atrybutów i relacji, można stosować dokumentowanie w formie grafów, których węzły są obiektami, łuki — relacjami, a węzły zakończenia grafu — atrybutami. Dla obiektów złożonych posiadających wiele atrybutów bądź wiele relacji wygodniej jest stosować dokumentowanie w formie oddzielnych wykazów (tablic). W wypadku stosowania metody semiotycznej ta ostatnia forma dokumentowania jest najwygodniejsza.

Na zakończenie pragniemy podkreślić, że z powyższych rozważań jasno wynika, iż baza danych nie może być widziana jako „odwzorowanie rzeczywistości". Obiekt bazy danych nie musi i nie jest zazwyczaj obiektem „rzeczywistym", lecz obiektem umownym, wynikającym z potrzeb semantycznych użytkownika i standardów semantycznych zastosowanych w systemach otoczenia bazy danych. Niemniej spotykana często w podręcznikach podstaw baz danych definicja, określająca bazę danych jako odwzorowanie rzeczywistości, znajduje pewne uzasadnienie dla względnie prostych semantycznie baz danych w zarządzaniu na szczeblu operacyjnym przedsiębiorstw. Wydaje się jednak, że spotykane nierzadko zalecenia, że baza danych powinna w swojej strukturze odwzorowywać maksymalnie dokładnie wybrany wycinek „rzeczywistości" może być tylko stwierdzeniem technologa systemów informatycznych, który nigdy nie miał okazji prowadzić prac analitycznych nad zdefiniowaniem optymalnego obiektu złożonej bazy danych.

4. Projektowanie modelu pojęciowego bazy danych

Pojęciowy model danych jest oparty na modelu semantycznym. Model semantyczny odwzorowuje relacje, obiekty i atrybuty nie uwzględniając ograniczeń i wymagań technologii bazy da-

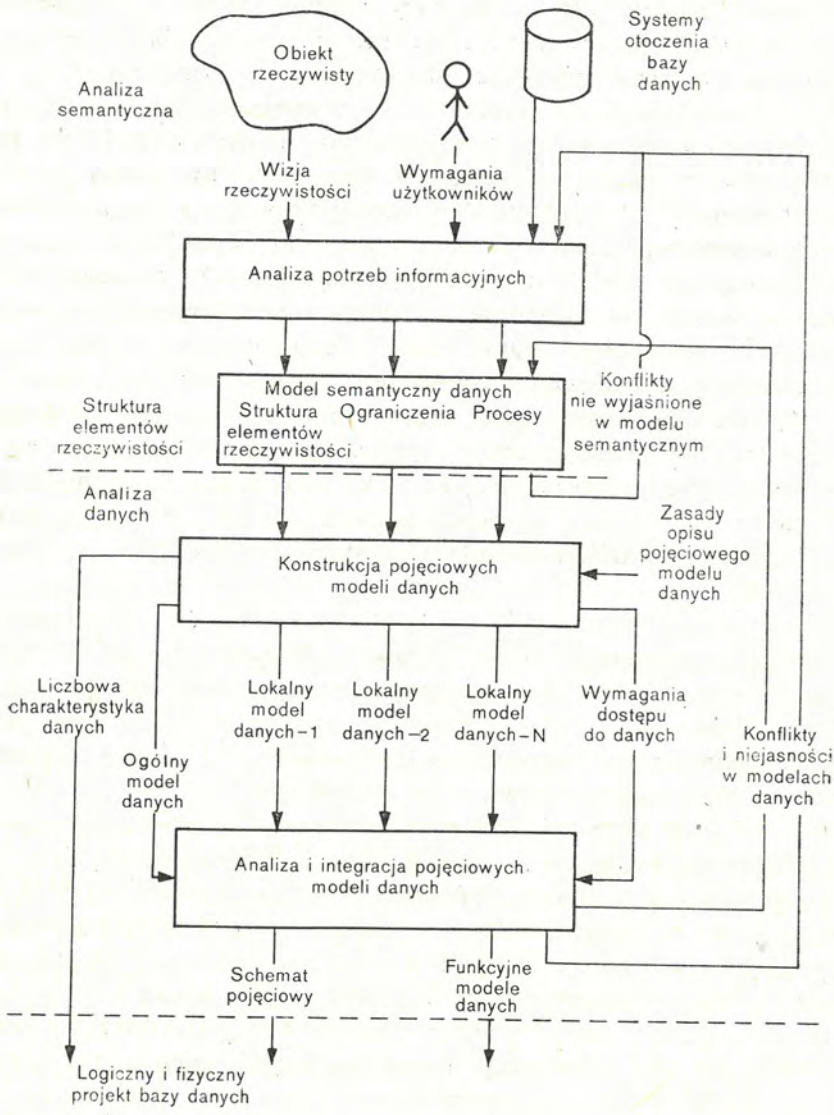
nych. Występować w nim mogą elementy trudne do odwzorowania w warunkach posiadanych środków językowych SZBD na poziomie modelu logicznego bazy danych. Model semantyczny formułowany jest w języku fachowym użytkownika finalnego, precyzuje i rozstrzyga wszelkie problemy semantyczne i definiuje treść informacji gromadzonych w bazie danych, ich relacje semantyczne z systemami otoczenia bazy danych oraz formy leksykalne, w jakich informacje z bazy mają być udostępnione użytkownikowi finalnemu. Aby zaprojektować opis tego modelu w komputerze, trzeba poddać go uprzednio formalizacji. Zadanie to należy do modelu pojęciowego bazy danych, zwanego dość często jeszcze w niektórych opracowaniach modelem konceptualnym bazy danych. Opis modelu pojęciowego w języku sformalizowanym nazywamy *schematem pojęciowym bazy danych*.

Schemat pojęciowy bazy danych nie jest prostą translacją modelu pojęciowego na opis w języku formalnym. Proces budowy modelu pojęciowego polega na selekcji i redukcji klas obiektów, relacji i atrybutów występujących w modelu semantycznym, a trudnych do odwzorowania w systemie informatycznym bazy danych.

Istotną cechą schematu pojęciowego jest jego niezależność od sposobu dalszej realizacji systemu informatycznego. W wypadku realizacji systemu informatycznego na podstawie Systemu Zarządzania Bazą Danych wynik analizy danych jest niezależny od możliwości wykorzystywanego systemu. Strukturę procesu analizy danych przedstawiono na rysunku 61.

Pierwszym etapem tej fazy projektowania bazy danych jest konstrukcja pojęciowych modeli danych, odpowiadających wizjom rzeczywistości poszczególnych grup użytkowników oraz kierownictwa organizacji. Wyniki analizy potrzeb informacyjnych służą do sformułowania ogólnego modelu danych obejmującego cały zakres prowadzonej analizy oraz wielu lokalnych modeli danych. Ogólny model danych odpowiada zwykle wizji kierownictwa badanej organizacji, natomiast lokalne modele reprezentują punkt widzenia poszczególnych grup użytkowników.

Jak już mówiliśmy wcześniej, potrzeby informacyjne analizowanych grup użytkowników wynikają z zakresu informacji wykorzystywanych przez procesy zachodzące w ich bezpośred-



Rys. 61. Proces analizy danych

nim otoczeniu. Tak więc analiza danych jest ściśle związana z analizą funkcji projektowanego systemu informatycznego, a szczególnie z występującymi w ramach lokalnych modeli danych wymagań dostępu do danych. Duży wpływ na tworzony pojęciowy model danych mają przyjęte zasady opisu.

Omawiając pojęciowy model danych (por. rozdz. III) wprowadziliśmy notację reprezentującą takie elementy struktury pojęciowego modelu danych, jak klasa obiektów, klasa powiązań obiektów oraz zbiór wartości własności. Proponowana graficzna metoda opisu pojęciowego modelu danych może być stosunkowo łatwo zastąpiona formalnym językiem opisu pojęciowego modelu danych. W praktyce projektowania baz danych wykorzystuje się wiele różnych notacji opisu pojęciowego modelu danych. Przedstawienie nawet najbardziej znanych notacji wykracza poza zakres tej pracy. Warto natomiast omówić podstawowe kryteria oceny pojęciowego modelu danych.

Podstawowym kryterium oceny jest kompletność opisu pojęciowego modelu danych z punktu widzenia reprezentacji modelu semantycznego badanej rzeczywistości. Niekompletny pojęciowy model danych może być istotnym utrudnieniem w realizacji dalszych faz projektowania oraz we wdrożeniu tworzonego systemu informatycznego. Brak kompletności może wynikać z błędów projektowych lub ograniczeń notacji opisu pojęciowego modelu danych. Drugim, równie istotnym, kryterium oceny jest redundancja opisu, polegająca na umieszczeniu w nim takich elementów, które mogą być uzyskane z innych elementów struktury pojęciowego modelu danych. Rozwój systemów informatycznych, wynikający ze zmian zachodzących w ich otoczeniu, wymaga odpowiedniej adaptacji pojęciowego modelu danych. W tej sytuacji podatność opisu na zmiany i łatwość ich wprowadzania może w niektórych sytuacjach mieć pierwszorzędne znaczenie. W tym wypadku przewagę uzyskują techniki opisu pojęciowego modelu danych umożliwiające komputerowe wspomaganie procesu tworzenia i utrzymania schematu pojęciowego oraz jego aktualizacji.

Do pożądanych cech przyjętej notacji opisu należą łatwość jego weryfikacji, niezależność od niższych poziomów projektowania bazy danych, tj. od logicznego i fizycznego modelu da-

nych, łatwość implementacji, a szczególnie możliwość komputerowego wspomagania. Przedstawiona w rozdziale III graficzna notacja spełnia — naszym zdaniem — podstawowe warunki konieczne do opisu pojęciowego modelu danych. Praktyczne wykorzystanie tej notacji wymaga uzupełnienia jej o mechanizm opisu zasad spójności logicznej (ograniczeń nakładanych na strukturę modelu). Jednym z możliwych rozwiązań jest uzupełnienie proponowanej notacji o formuły rachunku predykatów. Omawiając problemy związane z konstrukcją pojęciowych modeli danych oraz integracją tych modeli wykorzystamy tę notację do przedstawienia przykładów.

Informacjami wejściowymi dla procesu konstrukcji pojęciowych modeli danych są potrzeby informacyjne, procesy rozpatrywane z punktu widzenia całej organizacji oraz jako wymagania poszczególnych grup użytkowników opisane w modelu semantycznym. Uzupełnieniem tych informacji jest zbiór zasad spójności logicznej danych lub ograniczeń obowiązujących w modelu danych. Ograniczenia te wynikają zwykle z decyzji organizacyjnych kierownictwa lub tzw. „warunków obiektywnych", występujących w rozpatrywanej „rzeczywistości". Przykładem takiej sytuacji może być ograniczenie liczby studentów na seminariach, liczby seminariów, na które może uczęszczać student, lub ilości godzin wykładowych w miesiącu dla jednego wykładowcy itp.

Tego typu ograniczenia muszą być odwzorowane bezpośrednio w strukturze modelu, np. obiekt = student = może być powiązany co najwyżej z pięcioma obiektami = seminarium =. Podobnym ograniczeniem może być reprezentacja zdania „każde seminarium musi być powiązane z osobą prowadzącą", czyli dopuszczalne są jedynie takie stany modelu danych, w których każdy obiekt = seminarium = jest powiązany z jednym obiektem = = wykładowca =.

Wszystkie ograniczenia dotyczące stanów modelu danych będziemy nazywali statycznymi zasadami spójności logicznej.

Innym ograniczeniem może być zasada, że pracownik nie może modyfikować wysokości własnych ani cudzych poborów. W tym wypadku ograniczenie jest bezpośrednio związane z uprawnieniem do inicjacji procesu użytkowego, działającego na określo-

nych elementach struktury modelu danych. Tego typu ograniczenia będziemy nazywali dynamicznymi zasadami spójności logicznej. Zupełnie innym typem ograniczeń są ograniczenia projektowe, obejmujące zasady przypisywania nazw do poszczególnych elementów modelu danych oraz zasady zakładania i utrzymania słownika danych.

W dużych organizacjach, gdzie stan rozwoju zastosowań informatyki można uznać za zaawansowany, obowiązują standardowe procedury przypisywania nazw i wykorzystania skorowidza danych.

Podstawową przyczyną standaryzacji procedur przypisywania nazw do poszczególnych elementów modelu danych jest konieczność zapobiegania występowaniu homonimów i synonimów. Przy czym za homonimy uważamy nazwy, które są identyczne z punktu widzenia składni, lecz mają różne znaczenie semantyczne. Synonimami natomiast są nazwy identyczne z punktu widzenia semantyki i różne składniowo. Rozwiązanie problemu homonimów i synonimów wymaga zwykle uzupełniającej analizy potrzeb informacyjnych zmierzających do wyjaśnienia konfliktu nazw w miejscach ich powstawania i wykorzystania.

Posłużmy się teraz przykładem pojęciowego modelu danych domu towarowego przedstawionym w rozdziale III (por. rys. 18) w celu wyjaśnienia zasad tworzenia ogólnego i lokalnych pojęciowych modeli danych.

Prezentowany za pomocą notacji graficznej pojęciowy model danych domu towarowego może odpowiadać ogólnej wizji kierownictwa tej organizacji wyrażonej w sposób następujący. Dom towarowy jest podzielony na działy sprzedaży, z których każdy mieści się w całości na jednym piętrze. Natomiast jedno piętro może obejmować wiele działów. Określony towar może być sprzedawany wyłącznie przez jeden dział oraz równocześnie być dostarczany przez kilku dostawców. Istnieją również dostawcy domu towarowego, którzy przez pewien okres nie dostarczają żadnych towarów. Każdy dział zatrudnia pewną liczbę pracowników.

Nietrudno zauważyć, że przedstawiony na rysunku 18 semantyczny model danych w pełni oddaje treść tego scenariusza. W praktyce jednak mamy do czynienia ze znacznie bogatszą

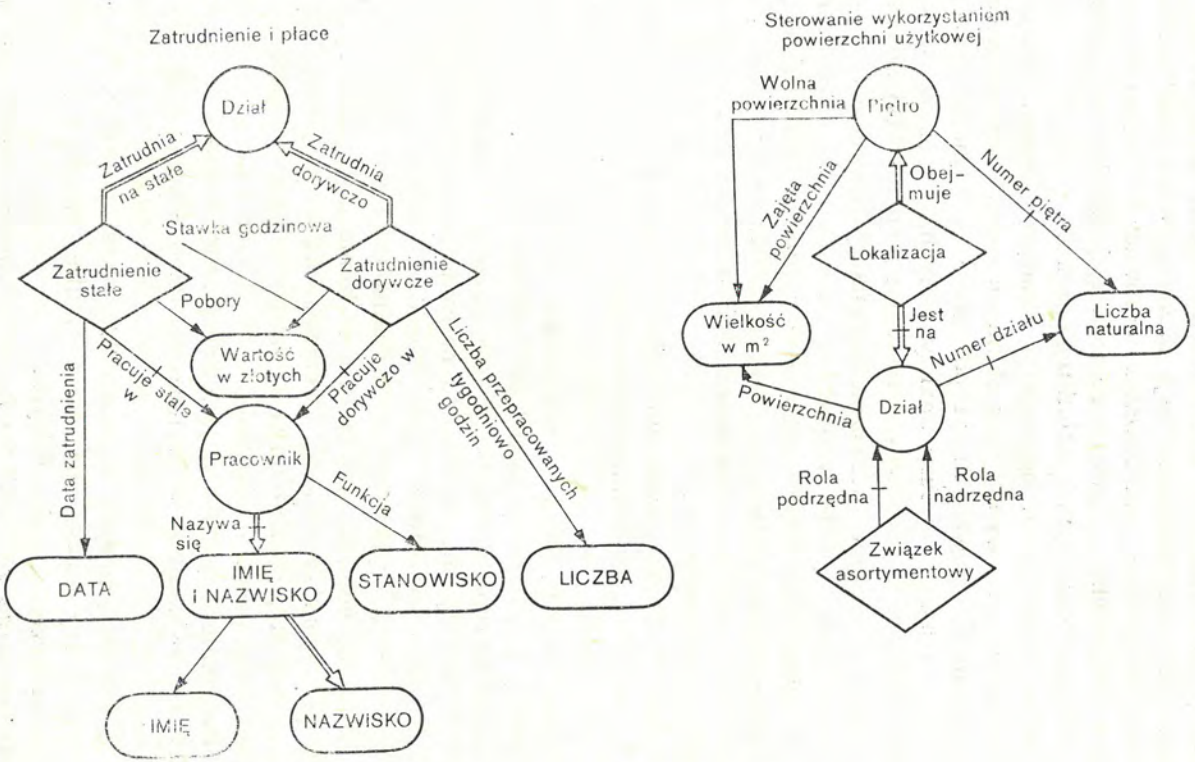
informacją o semantyce analizowanej rzeczywistości, a więc i modele znacznie bardziej złożone.

Przyjmijmy, że przedmiotem analizy potrzeb informacyjnych są takie zastosowania, jak zatrudnienie i płace oraz sterowanie wykorzystaniem powierzchni użytkowej. W wypadku systemu zatrudnienia i płac możemy mieć do czynienia z następującą wizją.

Dom towarowy zatrudnia stałych pracowników, których pobory są wypłacane miesięcznie, oraz pracowników dorywczych opłacanych według stawki godzinowej. W wypadku stałych pracowników wypłaca się również dodatek za wysługę lat, którego wysokość zależy od ilości lat pracy.

Sterowanie wykorzystaniem powierzchni użytkowej wymaga naturalnie zupełnie innego zakresu struktury ogólnego modelu danych. Kryterium rozmieszczenia działów sprzedaży na poszczególnych piętrach jest możliwie najlepsze wykorzystanie powierzchni użytkowej. Ograniczeniem w rozmieszczeniu działów jest konieczność ich grupowania według związków asortymentowych. Graficzna reprezentacja odpowiednich (lokalnych) pojęciowych modeli danych jest przedstawiona na rysunku 62.

W obu wypadkach występują istotne różnice w stosunku do zdefiniowanego ogólnego modelu danych. W lokalnym modelu danych systemu zatrudnienia i płac występują dwa rodzaje zatrudnienia, którym odpowiadają dwa typy powiązań. Dodatkowo, aby uniknąć konfliktu z zasadą semantyczną, z której wynika, że każdy pracownik musi być zatrudniony w jakimś dziale (i tylko jednym), zawartą w ogólnym modelu danych (zatrudnienie jest silnym powiązaniem typu 1 : N), należy określić dwa dodatkowe ograniczenia: pierwsze mówi, że każdy pracownik może być zatrudniony albo na stałe, albo dorywczo, a drugie — że każdy pracownik musi być zatrudniony w jakimś dziale. Konieczność określenia dodatkowych ograniczeń wynika z braku możliwości przedstawienia odpowiednich zasad semantycznych w przyjętej notacji. Dodatkowo okazuje się, przy bardziej szczegółowej analizie rozpatrywanego modelu, że wynagrodzenie (lub stawka godzinowa) jest atrybutem zatrudnienia a nie pracownika. Jest to logiczne, ponieważ ten sam pracownik może w różnych okresach pobierać różne wynagrodzenie.



Rys. 62. Lokalne pojęciowe modele danych

W wypadku systemu sterowania wykorzystaniem powierzchni użytkowej konieczne było wprowadzenie dodatkowego atrybutu (powierzchnia) działu oraz nowego powiązania (związek asortymentowy). Oczywiste jest również wprowadzenie ograniczenia, że suma powierzchni działów umieszczonych na dolnym piętrze nie może przekroczyć powierzchni tego piętra.

Po zakończeniu opisu ogólnego i lokalnych modeli danych oraz sformułowaniu wymagań dostępu do danych wynikających z zastosowań realizowanych na podstawie tych modeli można przystąpić do następnej fazy analizy danych, a mianowicie, do analizy i integracji pojęciowych modeli danych. W tej właśnie fazie powinna nastąpić ostateczna weryfikacja pojęciowych modeli danych, rozwiązanie wszystkich występujących konfliktów oraz sformułowanie pojęciowego modelu danych obejmującego całość analizowanej organizacji. Taki model stanowi zwykle kompromis między różnymi zastosowaniami odcinkowymi opartymi na wspólnej bazie danych.

Niezależnie od zastosowanych metod, integracja pojęciowych modeli danych musi prowadzić do osiągnięcia następujących wyników:

— wykrycie i rozwiązanie wszystkich konfliktów występujących między pojęciowymi modelami danych,

— dokładna reprezentacja wszystkich lokalnych modeli danych w globalnym pojęciowym modelu danych (schemat pojęciowy),

— stworzenie możliwości spełnienia wszystkich wymagań dostępu do danych wynikających z analizowanych zastosowań,

— uzyskanie pełnego — z punktu widzenia użytkownika bazy danych — opisu omawianej rzeczywistości przy jednoczesnej minimalizacji stopnia złożoności struktury rozumianej jako ilość obiektów, powiązań i atrybutów,

— stworzenie schematu pojęciowego, który może być zrealizowany na podstawie wybranego systemu Zarządzania Bazą Danych.

Integracja pojęciowych modeli danych jest procesem heurystycznym, który zwykle przebiega według omówionych ogólnych zasad łączenia modeli danych.

Ustalenie zasad zapewnienia kompatybilności modeli danych. Łączenie dwóch modeli danych wymaga ujednolicenia nazw

obiektów, atrybutów i powiązań w celu wybrania tych samych nazw dla tych samych obiektów bądź ustalenia jednoznacznych odpowiedników nazw. Występowanie tych samych nazw nie zawsze dowodzi identyczności odpowiednich elementów struktury modelu pojęciowego. Należy ustalić zasady kontroli i modyfikacji nazw elementów struktury, tak by po rozwiązaniu występujących konfliktów pojęciowe modele danych były kompatybilne. Konflikty występujące na tym poziomie dotyczą błędów w przypisywaniu nazw i identyfikacji elementów struktury pojęciowego modelu danych.

Określenie podstawowych operacji łączenia struktur danych. Do łączenia struktur pojęciowych modeli danych konieczny jest zbiór jednoznacznie określonych operacji. Operacje powinny obejmować nakładanie się identycznych struktur, podział, a także nakładanie się częściowo zgodnych struktur, przydział „kluczy pomocniczych" w wypadku, gdy ten sam obiekt jest identyfikowany przez różne atrybuty w różnych strukturach danych.

Ustalenie strategii łączenia pojęciowych modeli danych. Możliwe jest przyjęcie dwóch podstawowych typów rozwiązań problemu integracji pojęciowych modeli danych. Pierwszym z nich jest łączenie jednofazowe, polegające na równoczesnym łączeniu wszystkich analizowanych odcinkowych pojęciowych modeli danych. Proces łączenia może być w tym wypadku sterowany układem wag wyznaczających priorytety poszczególnych modeli danych. Na przykład ogólnemu modelowi danych jako wizji kierownictwa może być przypisany najwyższy priorytet. Priorytety poszczególnych wizji odcinkowych służą następnie do ustalenia zasad rozwiązywania konfliktów między modelami danych.

Drugą strategią jest łączenie wielofazowe polegające na wybraniu jednego z modeli danych jako dominującej wizji. Zwykle wybierany jest ogólny model danych, rzadziej model danych, reprezentujący wizję jednego z zastosowań. Dominujący model danych jest traktowany jako jądro przyszłego globalnego modelu danych i jest on kolejno łączony z pozostałymi pojęciowymi modelami danych.

Interpretacja wymagań dostępu do danych w kontekście globalnego modelu danych (schematu pojęciowego). Każde wymaganie dostępu do danych implikuje konieczność stworzenia od-

powiedniej logicznej ścieżki dostępu w strukturze modelu danych. Interpretacja tych wymagań w kontekście tworzonego schematu pojęciowego ma dwa podstawowe cele:

— sprawdzenie, jak globalny model danych spełnia wymagania poszczególnych zastosowań,

— stworzenie zbioru wymagań dostępu do danych wyrażonych w kategoriach schematu pojęciowego, będącego podstawą dalszych prac projektowanych na „niższych" etapach realizacji projektu (projekt logiczny i fizyczny).

Projektowanie schematu pojęciowego jest procesem skomplikowanym i wymagającym bieżącej współpracy z użytkownikiem końcowym. Forma opisu tego schematu powinna umożliwić jego weryfikację przez przedstawicieli kierownictwa organizacji oraz użytkowników działających w ramach poszczególnych zastosowań. Zagadnienie to nabiera szczególnego znaczenia ze względu na silny rozwój zastosowań eksploatowanych w trybie przetwarzania „na bieżąco". W wypadku tych bowiem zastosowań użytkownik ma bezpośredni kontakt z bazą danych, a znajomość struktury danych na poziomie schematu pojęciowego jest w takiej sytuacji konieczna. Sam proces analizy danych może być realizowany przy wykorzystaniu wielu różnych technik i metod projektowania.

Zaprezentowana notacja graficzna służy jedynie jako środek prezentacji przykładowych rozwiązań, a nie jako w pełni rozwinięta i kompletna technika tworzenia opisu pojęciowego modelu danych.

XI. Logiczny projekt bazy danych

1. Podstawowe elementy opisu logicznej struktury danych

Dotychczas omawiane etapy projektowania bazy danych nie były bezpośrednio związane z technologią przetwarzania danych, wynikającą z zasad działania i możliwości systemów zarządzania bazą danych. Prace nad projektem modelu informacyjnego są zorientowane przede wszystkim na użytkowe aspekty systemu informatycznego. Podstawowym wymaganiem w stosunku do pojęciowego modelu danych jest kompletność opisu struktury i odwzorowanie treści analizowanej informacji. Możliwości wykorzystania takiego opisu rzeczywistości na dalszych etapach prac nad systemem informatycznym są już bezpośrednio związane z przyjętą technologią przetwarzania danych.

We współcześnie opracowywanych systemach mamy zwykle do czynienia z technologią przetwarzania danych wynikającą z zastosowania systemu zarządzania bazą danych dla tworzenia i utrzymania wspólnej bazy danych. Pierwszym krokiem w tym kierunku jest selekcja i zakup systemu ZBD. Kryteria wyboru systemu zależą od celu realizowanego systemu informatycznego, a szczególnie od sprzętu, trybu przetwarzania (partiowe lub „na bieżąco") oraz stopnia złożoności algorytmów procesów użytkowych, o czym już mówiliśmy. Istotna, z punktu widzenia projektowania struktury danych, jest możliwość przeniesienia informacji zawartej w pojęciowym modelu danych na poziom modelu logicznego, tzw. sformułowanie modelu pojęciowego w języku opisu danych systemu ZBD. Takie możliwości zależą przede wszystkim od klasy struktury danych przyjętej w rozpatrywanym systemie ZBD oraz od zakresu funkcji języka opisu danych.

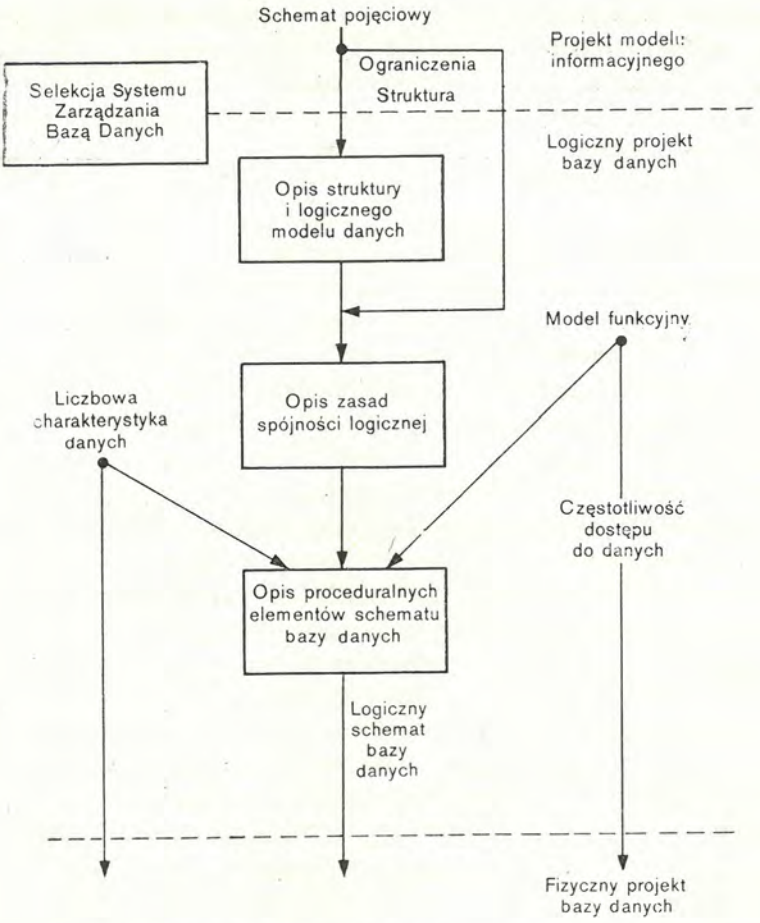
Rozważając problemy logicznego projektu bazy danych skoncentrujemy się na zagadnieniu opisu sieciowej struktury danych. Wynika to ze stosunkowo wysokiego stopnia uniwersalności tej klasy struktury danych oraz jej szerokiego wykorzystania w systemach ZBD. Klasycznym, a zarazem najbardziej kompletnym, językiem opisu sieciowej struktury danych jest język opisu danych zaproponowany przez Komitet CODASYL. Systemy ZBD, będące implementacją raportów tego Komitetu, są obecnie dostępne we wszystkich podstawowych typach komputerów. Również w Polsce najbardziej rozpowszechnionym systemem ZBD jest system RODAN, opracowany zgodnie ze specyfikacją języków zawartą w raportach Komitetu CODASYL.

W tej sytuacji omawiane przez nas zasady projektowania logicznej struktury danych (i dalej fizycznej struktury danych) zostaną zaprezentowane na podstawie elementów języków opisu danych Komitetu CODASYL. Należy tutaj podkreślić, że sekwencja czynności wykonywanych w procesie projektowania logicznej struktury danych oraz obowiązujące w ramach tego procesu ogólne zasady obowiązują również w razie wykorzystania systemów ZBD innego typu (np. hierarchicznych lub relacyjnych).

Elementy procesu projektowania logicznej struktury danych przedstawiamy na rysunku 63.

Opis struktury logicznego modelu danych obejmuje definicję typów rekordów oraz zawartych w nich grup powtarzalnych i danych elementarnych, jak również typów zbiorów strukturalnych z odpowiednimi opisami rekordów członkowskich. Opis rekordu członkowskiego zawiera zasady uczestnictwa rekordu w zbiorze strukturalnym.

Zasady tworzenia opisu logicznego modelu danych wynikają z przyjętych reguł odwzorowań elementów struktury pojęciowego modelu danych w logiczny model danych. Zbiór takich reguł wynika bezpośrednio z przyjętej notacji opisu pojęciowego modelu danych oraz klasy logicznej struktury danych. Dowolny zbiór reguł odwzorowania może być uznany za dostateczny, jeżeli umożliwia odwzorowanie jednego modelu danych w drugi bez straty informacji.



Rys. 63. Proces projektowania logicznej struktury danych

Omawiając bardziej szczegółowo problemy opisu sieciowej struktury danych zaproponowanej przez Komitet CODASYL przedstawimy przykład reguł odwzorowań, pozwalający na uzyskanie takiego opisu z pojęciowego modelu danych wyrażonego za pomocą omówionej w poprzednich rozdziałach notacji graficznej.

Powiedzieliśmy wcześniej, że pewien zakres informacji jest wprowadzany do schematu pojęciowego w postaci ograniczeń lub inaczej zasad spójności logicznej. Następnym więc etapem

tworzenia logicznego projektu bazy danych jest sformułowanie, przy wykorzystaniu dostępnych w danym systemie ZBD środków językowych, zbioru zasad spójności logicznej obowiązujących w bazie danych. Podobnie jak w wypadku modelu pojęciowego, tak i w definiowaniu struktury logicznego modelu danych podstawowych wymaganiem jest kompletność opisu zasad i spójności logicznej.

Jeżeli możliwości używanego Języka Opisu Danych nie są wystarczające, to odpowiednie zasady spójności logicznej muszą być opisane jako elementy algorytmów oprogramowania użytkowego. W wypadku systemów tradycyjnych wszystkie zasady spójności logicznej były opisane i utrzymywane w ramach oprogramowania użytkowego (programy kontroli danych).

Możliwości Języka Opisu Danych Komitetu CODASYL obejmują elementy wspomagające realizację procesów użytkowych. Do takich elementów opisu logicznej struktury danych należy opis ścieżki selekcji, opis zasad umieszczania rekordu w bazie danych czy też opis porządku zbioru strukturalnego. Projektowanie tej części logicznego schematu bazy danych jest ściśle związane z projektowaniem algorytmów programów użytkowych. Istotną cechą tej części opisu jest, aby jakakolwiek wprowadzana do niego zmiana nie niosła za sobą zagrożenia utraty informacji. Jedynym wyjątkiem może być utrzymywanie porządków zbioru strukturalnego wyrażających chronologię dopisania rekordów w procesie aktualizacji w bazach danych statystycznych. Również i w tym wypadku można uniknąć straty informacji przez uwzględnienie takiego zagrożenia (i jego likwidacji) na etapie logicznego projektu bazy danych.

Informacjami wejściowymi do tej fazy projektu logicznej struktury danych są poza opisem struktury logicznego modelu danych wraz z obowiązującymi w nim zasadami spójności logicznej, opis algorytmów procesów użytkowych (model funkcyjny) oraz liczbowa charakterystyka danych. Wynikiem logicznego projektu bazy danych jest jej schemat zapisany w Języku Opisu Danych, przy czym elementy języka opisu danych dotyczące fizycznej struktury danych są na tym etapie projektu pomijane. Odpowiednie uzupełnienia logicznego schematu bazy danych są tworzone w toku opracowania fizycznego projektu bazy danych.

Brak czystego podziału językowego między logicznym a fizycznym projektem bazy danych jest podstawowym mankamentem wykorzystywanych obecnie Języków Opisu Danych.

2. Opis struktury logicznego modelu danych

Pierwszą fazą logicznego projektu bazy danych jest „przeniesienie" informacji o strukturze pojęciowego modelu danych na poziom logiczny. Jak już powiedzieliśmy, możliwości opisu logicznego modelu danych wynikają bezpośrednio z klasy logicznej struktury danych dostępnej w wykorzystywanym systemie ZBD lub inaczej z możliwości Języka Opisu Danych. Niemniej ważne są przyjęte reguły odwzorowań elementów pojęciowego modelu danych w logicznym modelu danych. Dobrym, naszym zdaniem, przykładem języka dla logicznego projektu bazy danych jest proponowany przez Komitet CODASYL Język Opisu Danych [1]. Wykorzystamy więc ten język, również jako ilustrację procesu opisu struktury logicznego modelu danych.

Podstawowymi elementami opisu struktury logicznego modelu danych są takie pojęcia, jak rekord, grupa powtarzalna, dana elementarna oraz zbiór strukturalny. Definicja tych pojęć jako elementów struktury logicznego modelu danych wymaga dosyć ograniczonego zakresu Języka Opisu Danych. Mianowicie w wypadku rekordu wystarcza klauzula RECORD, w wypadku grupy powtarzalnej i danej elementarnej klauzula DATA-BASE--DATA-NAME i OCCURS-, a opis zbioru strukturalnego wymaga klauzul: SET, OWNER i MEMBER.

Naturalnie, uzyskanie kompletnego opisu na poziomie logicznego schematu bazy danych wymaga wykorzystania znacznie większego zakresu Języka Opisu Danych. Wynika to z konieczności uzupełnienia opisu struktury logicznego modelu danych takimi informacjami, jak opis zasad spójności logicznej oraz opis proceduralnych elementów schematu bazy danych.

Istotne, z punktu widzenia oceny możliwości przeniesienia informacji z poziomu pojęciowego na logiczny, są przyjęte za-

[1] Por. *CODASYL Data Base Task Group 1971 Report*, ACM, New York 1972; *CODASYL Data Description Language Committee DDL*, Journal of Development, NBS, New York 1973; *CODASYL Data Description Language Committee 1978 Report*, *Information Systems*, vol. 3, 1978, nr 4.

sady odwzorowania elementów tych modeli. Przykładem zbioru takich zasad są przedstawione w tablicy 14 reguły odwzorowania elementów wykorzystywanej przez nas graficznej notacji opisu pojęciowego modelu danych w logiczny model danych Komitetu CODASYL. Zestawienie odpowiadających sobie elementów opisu struktury obu modeli wykazuje wyraźnie istniejące niebezpieczeństwo straty informacji w procesie transformacji modelu pojęciowego na model logiczny. Przede wszystkim nie istnieje możliwość bezpośredniego opisu w logicznym modelu danych takich elementów pojęciowego modelu danych, jak silne oraz prawostronne słabe powiązanie obiektów typu 1 : 1 i 1 : N, jak również silnego, prawostronnie słabego i lewostronnie słabego powiązania obiektów typu N : M. Takie ograniczenie wynika z braku odpowiednich wyrażeń Języka Opisu Danych umożliwiających opisanie tak zwanego „obowiązującego właściciela" to znaczy, takiego zbioru strukturalnego, który nigdy nie będzie pusty.

Przykładem może być baza danych = *ojcowie* = gdzie zapisano jedynie mężczyzn mających dzieci, przy czym rekord = = *ojciec* = jest powiązany z odpowiednimi rekordami = *dziecko* = przez zbiór strukturalny = *rodzicielstwo* = (por. rys. 64).

Rys. 64. Schemat struktury bazy danych = OJCOWIE =

Zgodnie z podanymi zasadami budowy struktury modelu danych żaden z występujących w bazie danych *ojcowie* zbiorów strukturalnych *rodzicielstwo* nie może być tzw. pustym zbiorem strukturalnym. Inaczej mówiąc, każdemu zapisanemu rekordowi *ojciec* musi odpowiadać co najmniej jeden rekord *dziecko* powiązany z nim przez zbiór strukturalny *rodzicielstwo*.

Tablica 14

Reguły odwzorowania elementów struktury modelu pojęciowego w logicznym modelu danych

Logiczny model danych	Pojęciowy model danych
Rekord Grupa powtarzalna	obiekt
Dana elementarna	atrybut obiektu
Grupa powtarzalna Liczba wystąpień ϕ lub 1 Zbiór strukturalny *Manual-Mandatory,* *Automatic-Optional* *Manual-Optional*	powiązanie obiektów 1 : 1 słabe
Grupa powtarzalna Liczba wystąpień ϕ lub więcej Zbiór strukturalny *Manual-Mandatory,* *Automatic-Optional* *Manual-Optional*	powiązanie obiektów 1 : N słabe
Grupa powtarzalna Liczba wystąpień 1 Zbiór strukturalny *Automatic-Mandatory*	powiązanie obiektów 1 : 1 lewostronnie słabe
Grupa powtarzalna Liczba wystąpień 1 lub więcej Zbiór strukturalny *Automatic-Mandatory*	powiązanie obiektów 1 : N lewostronnie słabe
—	powiązanie obiektów 1 : 1, 1 : N silne prawostronnie słabe
Rekord relacji	powiązanie obiektów N : M słabe
—	powiązanie obiektów N : M, silne, prawostronnie słabe, lewostronnie słabe
Dana elementarna zawarta w rekordzie właściciela zbioru strukturalnego	atrybut powiązania obiektów 1 : 1, 1 : N
Dana elementarna zawarta w rekordzie relacji	atrybut powiązania obiektów N : M
Typ danej elementarnej	dziedzina

Zapis takiego powiązania jest stosunkowo prosty w wypadku omawianej graficznej notacji opisu pojęciowego modelu danych, natomiast nie jest możliwy przy wykorzystaniu Języka Opisu Danych Komitetu CODASYL. Odpowiedni opis logicznego modelu danych można uzyskać jedynie przez uzupełnienie opisu struktury tego modelu opisem odpowiednich zasad spójności logicznej.

Decyzje wyboru odpowiednich elementów struktury logicznego modelu danych wynikają często z przewidywanych wymagań dostępu do danych oraz z pożądanych parametrów eksploatacyjnych. Przykładem takiej decyzji może być wybór pomiędzy odwzorowaniem powiązania obiektów typu 1 : N (słabego) jako zbioru strukturalnego lub jako grupy powtarzalnej. W pierwszym wypadku mamy możliwość przeniesienia nazwy tego powiązania na poziom logicznego modelu danych oraz bezpośredniego opisu zasad uczestnictwa w zbiorze strukturalnym. Rozważając ten problem z punktu widzenia parametrów eksploatacyjnych zauważymy natychmiast, że przeczytanie danych dotyczących wszystkich obiektów uczestniczących w powiązaniu wymaga, w pierwszym wypadku, wydania całej serii komend Języka Manipulacji Danymi FIND/GET, natomiast w drugim wypadku wystarczy zawsze tylko jedna para takich komend. Z drugiej strony wystąpienie dużej liczby podporządkowanych obiektów może, w wypadku zastosowania grupy powtarzalnej, doprowadzić do konieczności zastosowania bardzo dużych buforów na rekordy bazy danych, a tym samym do nadmiernego wykorzystania pamięci operacyjnej.

Wyrażenie powiązania typu N : M przez rekord relacji jest praktycznie jedyną możliwością odwzorowania tego elementu struktury pojęciowego modelu danych w logicznym modelu danych Komitetu CODASYL. Zastosowanie tych reguł odwzorowania umożliwia przejście z pojęciowego na logiczny poziom opisu struktury w modelu danych, przy czym zachowanie kompletności opisu wymaga uzupełnienia definicji struktury logicznego modelu danych opisem zasad spójności logicznej. Z tego przede wszystkim powodu automatyczne generowanie opisu struktury logicznego modelu danych na podstawie opisu pojęciowego modelu danych jest w znacznym stopniu utrudnione.

3. Opis zasad spójności logicznej

Omawiając pojęciowy model danych zwróciliśmy uwagę na konieczność uzupełnienia opisu jego struktury dodatkowym opisem obejmującym zbiór zasad spójności logicznej. Struktura pojęciowego modelu danych oraz zbiór obowiązujący w tym modelu zasad spójności logicznej są, a przynajmniej powinny być, kompletnym opisem reprezentowanej rzeczywistości. Zagadnienie spójności logicznej jest związane bezpośrednio z pojęciem jakości danych. Stanowi ona jeden z jego elementów. Problem jakości danych należy bowiem rozpatrywać na dwóch płaszczyznach, to jest jako:

— niesprzeczność danych, polegająca na zgodności struktury danych z jej opisem oraz zasadami konstrukcji przyjętymi dla tej klasy struktury danych,

— spójność logiczna danych, polegająca na zgodności struktury danych z regułami przyjętymi dla reprezentowanej przez tę strukturę danych rzeczywistości.

Utrzymanie niesprzeczności danych w modelu logicznym jest jedną z podstawowych funkcji systemu ZBD realizowaną automatycznie na podstawie odpowiednich opisów struktury danych (logiczny i fizyczny schemat bazy danych). Natomiast zagadnienie spójności logicznej musi być rozpatrywane zarówno jako element statyczny opisu bazy danych definiujący zbiór dopuszczalnych stanów jej struktury, jak i element dynamiczny opisu bazy danych określający zbiór operacji dopuszczalnych do wykonania na tej bazie danych.

Możliwości Języka Opisu Danych Komitetu CODASYL obejmują swoim zakresem oba te zagadnienia. Sam zapis zasad spójności logicznej może być zrealizowany w formie deklaratywnej, to jest przy wykorzystaniu odpowiednich klauzul języka, lub w sposób proceduralny w formie procedur bazy danych.

Procedury bazy danych są zwykle realizowane w konwencjonalnych językach programowania (COBOL, PL/1). W niektórych systemach ZBD dopuszczalne jest wykorzystanie w ich ramach Języka Manipulacji Danymi.

W wypadku statycznym zasad spójności logicznej deklaratywny opis jest ograniczony wyłącznie do kontroli wartości da-

nych elementarnych lub ciągów będących produktem konkatenacji danych elementarnych. Do podstawowych elementów takiego opisu należą: kontrola duplikatów wartości danych elementarnych, kontrola wartości pustych w kluczach sortowania oraz kontrola na zakres wartości. Wszystkie te elementy kontroli danych występowały (i są) w tradycyjnych systemach informatycznych jako elementy oprogramowania użytkowego.

Łatwo sobie jednak wyobrazić wiele zasad spójności logicznej, których nie można wyrazić za pomocą tak prymitywnego aparatu językowego. Na przykład zasada =żaden zbiór strukturalny nie może być pusty= obowiązująca w bazie danych =*ojcowie*= (por. rys. 64) nie może być w ten sposób wyrażona.

Dobrym więc uzupełnieniem możliwości deklaratywnego opisu zasad spójności logicznej jest mechanizm proceduralnej realizacji kontroli tych zasad. Takim mechanizmem są procedury bazy danych automatycznie wykonywane w toku realizacji określonych funkcji Języka Manipulacji Danymi na danym typie elementu struktury danych. Jeżeli procedury bazy danych pozwalają na wykorzystanie komend Języka Manipulacji Danymi, a tym samym swobodną nawigację po bazie danych, to istnieje możliwość kontroli dowolnego predykatu wyrażającego zasadę spójności logicznej.

Opiniując zasady spójności logicznej należy jednak zawsze pamiętać, że ich kontrola może być poważnym obciążeniem systemu ZBD, a tym samym wpływać ujemnie na parametry eksploatacyjne zastosowań.

Dynamiczne zasady spójności logicznej dotyczą, jak już mówiliśmy, dopuszczalności wykonania operacji na bazie danych. Przykładem może być zablokowanie możliwości modyfikowania zawartości rekordów określonego typu przez niektóre programy zastosowań. Deklarowanie takich zasad jest realizowane za pomocą aparatu ochrony tajności danych (PRIVACY) dostępnego w Języku Opisu Danych. Również i w tym wypadku znacznie rozszerzono możliwość tego mechanizmu przez dopuszczenie procedur bazy danych.

Na zakończenie tych rozważań należy z przykrością stwierdzić, że mechanizmy opisu zasad spójności logicznej są albo całkowicie pomijane, albo stosowane w bardzo ograniczonym stopniu we

współczesnych systemach ZBD. Tak więc problem kontroli zasad spójności logicznej spoczywa nadal, tak jak w systemach tradycyjnych, w poważnym stopniu na oprogramowaniu użytkowym. Taki stan rzeczy jest poważnym ograniczeniem technologicznym we wdrażaniu systemów informatycznych, opartych na wspólnej bazie danych, a szczególnie systemów pracy „na bieżąco".

4. Opis proceduralnych elementów schematu logicznego bazy danych

Język Manipulacji Danymi Komitetu CODASYL obejmuje komendy umożliwiające nawigację po strukturze danych, odwzorowujące postacie rekordów i danych oraz modyfikujące bazę danych. Jednocześnie do Języka Opisu Danych wprowadzono wiele elementów wspomagających wykonanie komend Języka Manipulacji Danymi. Wspomaganie polega na automatycznej realizacji określonych funkcji systemu ZBD zastępujących sekwencję komend Języka Manipulacji Danymi. Prowadzi to do istotnego obniżenia pracochłonności oprogramowania użytkowego oraz do podniesienia jego parametrów eksploatacyjnych. (Każde odwołanie do komendy Języka Manipulacji Danymi jest związane z określonym kosztem inicjacji (zakończenia takiej komendy)).

Tak więc wykorzystanie proceduralnych elementów języka opisu danych przynosi największe efekty w wypadku zastępowania nimi najbardziej typowych algorytmów. Na przykład definicja ścieżki selekcji dla zbioru strukturalnego zgodnej z najczęściej stosowanym algorytmem selekcji.

Podstawową grupą proceduralnych elementów schematu bazy danych są funkcje wspomagające ścieżkę selekcji oraz komendy nawigacji po strukturach (komenda FIND). Do tej grupy należą takie klauzule języka opisu danych, jak LOCATION MODE, WITHIN, SET, ORDER, RANGE KEY oraz SELECTION. Niektóre z nich mają charakter czysto deklaratywny (SET SELECTION). Inne powodują istotne różnice w konstrukcji fizycznej struktury danych (LOCATION MODE). Klauzula LOCATION MODE określa początek ścieżki selekcji dla określonej komendy

Języka Manipulacji Danymi (tzw. punktu wejścia). Określony w tej klauzuli sposób odwzorowania rekordu ma istotny wpływ na wybór formatu i sposobu realizacji odpowiednich funkcji systemu ZBD. Klauzula WITHIN pozwala na automatyczny wybór przez system ZBD odpowiedniego obszaru bazy danych dla zapisania lub przeczytania rekordu.

Ścieżkę selekcji oraz elementy wspomagające selekcję rekordów w ramach wystąpień zbiorów strukturalnych wspomagają klauzule SET, SET SELECTION oraz definicja porządku zbioru strukturalnego zawarte w klauzuli ORDER i odpowiednich klauzulach (RANGE KEY). Istotną sprawą dla określenia wspomagania ścieżki selekcji jest całkowita deklaratywność. Fakt, że różne ścieżki selekcji mogą być definiowane na tych samych elementach struktury danych nie powoduje żadnych zmian w ich fizycznej postaci, umożliwia opis ścieżek selekcji odpowiadających różnym algorytmom wyszukiwania. Różne ścieżki selekcji są zwykle opisywane w podschematach zdefiniowanych dla różnych zastosowań. Wybór standardowych ścieżek selekcji powinien być oparty m.in. na ocenie liczebności danych.

Innym typem proceduralnych elementów schematu bazy danych jest wspomaganie „materializacji" wartości danych elementarnych. Taka „materializacja" polega na kopiowaniu wartości danej elementarnej z innego rekordu (klauzula SOURCE) lub obliczaniu wartości danej elementarnej na podstawie wartości innych danych elementarnych, zapisanych w tym samym lub innych rekordach (klauzula RESULT). Również w tym wypadku mamy do czynienia z obniżeniem pracochłonności oprogramowania użytkowego, dzięki standardyzacji procedur oraz podniesienia efektywności przez oszczędność miejsca w bazie danych.

Pozostałe elementy proceduralne mogą być wprowadzone do schematu bazy danych jako procedury bazy danych wywołane automatycznie w wypadku wykonania lub błędu wykonania określonych komend Języka Manipulacji Danymi.

Omówiony aparat wspomagania oprogramowania użytkowego został zastosowany jedynie w niektórych systemach ZBD, opartych na raportach Komitetu CODASYL. Procedury bazy danych będące istotną częścią omawianego aparatu mogą być również wykorzystane jako mechanizm wspomagający procedury admi-

nistrowania bazy danych. Typowym zastosowaniem z tego zakresu jest wykorzystanie procedur bazy danych do archiwowania skreślonych rekordów lub do zbierania informacji o przebiegu eksploatacji. Również i te elementy logicznego projektu bazy danych mogą mieć bardzo istotny wpływ na parametry eksploatacyjne bazy danych.

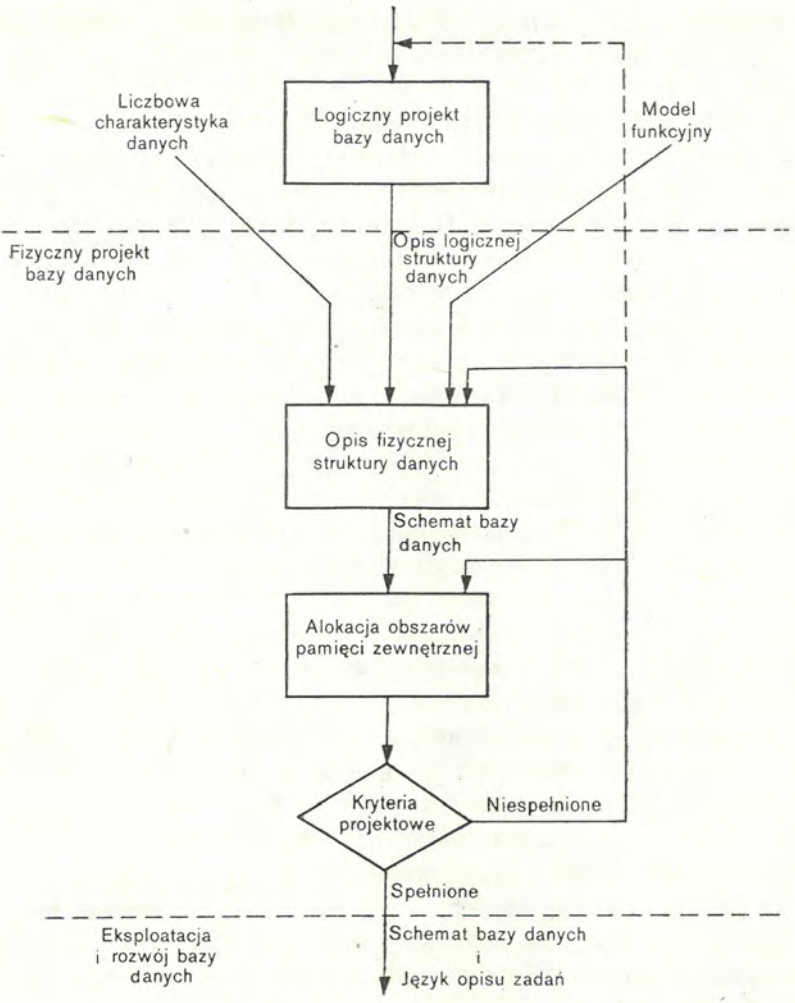
XII. Fizyczny projekt bazy danych

1. Podstawowe elementy opisu fizycznej struktury danych

Omawiając fazy procesu projektowania bazy danych zwracaliśmy już uwagę na zasadnicze znaczenie parametrów eksploatacyjnych w procesie projektowania fizycznej struktury danych. Podstawowym celem projektu fizycznej struktury danych jest znalezienie takiego odwzorowania logicznego modelu danych w model fizyczny, by parametry eksploatacyjne wszystkich zastosowań realizowanych na podstawie wspólnej bazy danych zawierały się w ograniczeniach przyjętych dla dopuszczalnych rozwiązań. Projektowanie fizycznej struktury danych prowadzi do kompromisu między różnymi wymaganiami dostępu do danych (np. sekwencyjny lub bezpośredni) występującymi w ramach różnych zastosowań.

Duża ilość powiązanych ze sobą warunków i ograniczeń praktycznie uniemożliwia opracowanie jednoznacznej metody uzyskiwania optymalnych modeli fizycznej struktury danych. Stąd też i tu mówimy zwykle o zakresie dopuszczalnych rozwiązań oraz o poprawieniu projektu, a nie o jego optymalizacji. Elementy procesu projektowania fizycznej struktury danych przedstawiamy na rysunku 65.

Wynikiem fazy logicznego projektu bazy danych jest jej schemat wyrażony jako ciąg zdań Języka Opisu Danych. Mówiąc o schemacie bazy danych zgodnym z wymaganiami Komitetu CODASYL zwróciliśmy również uwagę, że niektóre elementy wykorzystywanego Języka Opisu Danych mają znaczenie czysto fizyczne w tym sensie, że reprezentują decyzje projektowe z zakresu fizycznej struktury danych.



Rys. 65. Proces projektowania fizycznej struktury danych

Takimi elementami języka są przede wszystkim klauzule opisujące obszary bazy danych, zasady umieszczania rekordu w bazie danych oraz zasady fizycznej reprezentacji danych elementarnych. Podstawowym jednak środkiem wyrazu decyzji projektowych w zakresie fizycznej struktury danych jest język opisu fizycznej struktury danych lub inaczej Język Opisu Pamięci.

W systemach ZBD, w których nie występuje osobny język opisu pamięci, odpowiedni aparat opisu został umieszczony w Języku Opisu Danych.

Podstawowym warunkiem poprawności opisu fizycznej struktury danych są odpowiednie informacje projektowe obejmujące schemat logicznej struktury danych wraz z opisem jego proceduralnych elementów, opis funkcji realizowanych na bazie danych (tzw. model funkcyjny) oraz charakterystyka wielkości bazy danych. Szczególnie ten ostatni element, niestety często zupełnie pomijany, ma podstawowe znaczenie dla poprawności oraz wiarygodnej weryfikacji projektu fizycznej struktury danych.

Charakterystyka modelu funkcyjnego powinna obejmować opis algorytmów, wymaganie eksploatacyjne, zakres wykorzystywanej struktury danych (może być podany w formie podschematu bazy danych), zasady zachowania logicznej spójności danych oraz częstotliwość wykonania poszczególnych komend Języka Manipulacji Danymi. Opis każdego algorytmu powinien być przedstawiony jako szkielet programu użytkowego obejmujący wszystkie wykorzystywane w nim komendy Języka Manipulacji Danymi oraz podstawowe zdania sterujące bazowego języka programowania (pętle Do; skoki; zdania warunkowe if, then, else itp.).

Zasady zachowania spójności logicznej obejmują takie elementy jak opis zwartej sekwencji komend Języka Manipulacji Danymi, warunków wykonania poszczególnych komend, relacji na błędy danych oraz własne elementy obsługi sytuacji awaryjnych.

Częstotliwość wykonania poszczególnych komend Języka Manipulacji Danymi oraz całych zwartych sekwencji komend może być wyrażona np. jako ilość wykonań na jedną transakcję lub w wypadku przetwarzania partiowego jako i ilość wykonań na jeden logiczny element danych wejściowych. Jeżeli algorytm zawiera alternatywne ścieżki wykonania, należy podać rozkład częstotliwości realizacji poszczególnych sekwencji algorytmu.

Ważną charakterystyką procesu użytkowego są jego stany przed wykonaniem poszczególnych komend JMD. Jeżeli stan procesu nie jest zdeterminowany przez poprzednio wykonywaną komendę (np. komenda FIND ustawia bieżący rekord zadania oraz inne wskaźniki bieżącego rekordu), to powinien on zostać opisany. Informacje o stanach procesu użytkowego dotyczą takich

jego elementów, jak wskaźniki bieżących rekordów, obszar roboczy użytkownika, bufory rekordów itp. Podanie takich informacji pozwala na szacunek pracochłonności wykonania poszczególnych komend Języka Manipulacji Danymi. Na przykład włączenie przesortowanych wystąpień rekordów do uporządkowanego zbioru strukturalnego jest znacznie bardziej efektywne niż w wypadku tego samego zdania przy losowym rozkładzie wystąpień rekordów.

Ponieważ różnice w parametrach eksploatacyjnych mogą być znaczne, nawet warunkowe ich określenie może być bardzo przydatne w procesie projektowania fizycznej struktury danych. Wymagania eksploatacyjne określają stopień wykorzystania zasobów sprzętu i oprogramowania, dostępnych w ramach projektowanego systemu informatycznego. Opóźnienia wynikające z oczekiwania w kolejkach do poszczególnych zasobów systemu mają bardzo poważny wpływ na takie parametry eksploatacyjne, jak czas odpowiedzi transakcji lub „przepustowość" całego systemu komputerowego.

Istotnym obciążeniem systemu może być również pamięć operacyjna konieczna dla zarządzania kolejkami zadań. Do podstawowych informacji z zakresu wymagań eksploatacyjnych należą liczby typów transakcji, zasady przetwarzania współbieżnego oraz rozkład dostępu do danych. Liczba typów transakcji mówi praktycznie o liczbie różnych programów aplikacyjnych eksploatowanych w ramach systemu informatycznego. Istotnymi informacjami są w tym wypadku również ogólna liczba transakcji nadchodzących w jednostce czasu, minimalne i maksymalne obciążenia oraz częstotliwość występowania transakcji poszczególnych typów.

Równie istotnym elementem informacji projektowych jest przewidywana liczba współbieżnych transakcji oraz ich potencjalne zakresy konfliktów. Potencjalnym zakresem konfliktu nazywamy zbiór elementów struktury danych (rekordów, danych elementarnych, zbiorów strukturalnych), który jest iloczynem logicznym zbiorów elementów struktury danych objętych podschematem bazy danych. Możliwość interferencji dwóch rozpatrywanych procesów użytkowych wynika z istnienia potencjalnego zakresu konfliktu. Niektóre systemy ZBD dokonują wstępnej analizy

procesów użytkowych z punktu widzenia potencjalnych konfliktów z innymi współbieżnymi procesami. Jeżeli taka analiza nie wykazuje możliwości takich konfliktów, to następuje automatyczne wyłączenie kosztownych, z punktu widzenia zasobów systemu liczącego, mechanizmów synchronizacji współbieżnych procesów.

Bardzo często prawdopodobieństwo wystąpienia zjawiska interferencji współbieżnych procesów użytkowych jest określane przez odwrotność liczby wystąpień rozpatrywanych elementów struktury danych (rekord, strona, obszar bazy danych). W rzeczywistości sytuacja może kształtować się całkiem odmiennie. Zwykle bywa tak, że rozkład dostępu do danych w konkretnym systemie informatycznym w znaczny sposób zwiększa prawdopodobieństwo interferencji współbieżnych procesów w stosunku do teoretycznie zakładanego współczynnika (odwrotność liczby elementów).

Łatwo możemy wyobrazić sobie kraj, w którym większość podróżujących samolotem dokonuje rezerwacji w okresie mniej więcej dwóch tygodni przed odlotem (robi tak około 90% podróżnych). W tej sytuacji współczynnik prawdopodobieństwa wystąpienia zjawiska interferencji współbieżnych procesów użytkowych wykonywanych w ramach systemu rezerwacji miejsc lotniczych jest odwrotnością liczby lotów wykonywanych za dwa tygodnie (lub w przybliżeniu tej liczby), a nie odwrotnością liczby lotów zapamiętanych w bazie danych. Z punktu widzenia sytuacji eksploatacyjnej taka różnica może mieć istotne znaczenie.

W wypadku systemów pracujących „na bieżąco" ważnym parametrem może być opóźnienie powodowane czasem reakcji użytkownika końcówki systemu. Takie opóźnienie może w niektórych systemach spowodować zablokowanie na dłuższy czas istotnych zasobów systemu (pamięć operacyjna, obszar bazy danych, element struktury danych itp.). Na przykład interakcja z użytkownikiem końcówki dialogowej, która następuje w toku realizacji zwartej sekwencji komend Języka Manipulacji Danymi, może spowodować zablokowanie na długi czas elementów struktury danych, tym samym tworząc potencjalne zagrożenie nadmiernego opóźnienia współbieżnie obsługiwanej innej transakcji.

Charakterystykę wielkości bazy danych formułuje się w toku

analizy danych, a następnie uwzględnia się w logicznej strukturze danych, opracowując logiczny model bazy danych. Minimalna informacja projektowa z tego zakresu powinna obejmować następujące elementy:

— liczbę wystąpień rekordów danego typu,

— rozkład wystąpień rekordów według długości w ramach danego typu rekordu zmiennej długości,

— uczestnictwo w zbiorach strukturalnych w ramach poszczególnych typów rekordów,

— rozkład wartości kluczowych danych elementarnych, przy czym za kluczowe dane elementarne uważamy klucze, argumenty indeksów, klucze sortowania itp.,

— rozkład wartości wybranych danych elementarnych; niektóre dane elementarne są szczególnie podatne na kompresję: np. długie ciągi znaków o stosunkowo małej liczbie różnych wystąpień (nazwa i adres dostawcy w wypadku, gdy przedsiębiorstwo ma tylko 10 dostawców),

— rozkład liczności wystąpień zbiorów strukturalnych; ta informacja może być szczególnie użyteczna dla dokonania wyboru ścieżki dostępu, jak również dla wyboru odpowiedniej strategii rozmieszczenia wystąpień rekordów w bazie danych.

Wszystkie te informacje stanowią statyczną charakterystykę struktury danych. Dynamiczna charakterystyka obejmująca aktywność i zmienność poszczególnych elementów struktury danych, wyrażona zwykle jako liczba dostępów lub modyfikacji elementów określonego typu w jednostce czasu, wynika z analizy informacji zawartych w opisie modelu funkcyjnego.

Omówione już informacje projektowe są wykorzystywane w kolejnych krokach procesu projektowania fizycznej struktury danych. Kolejne kroki tego procesu to: opis fizycznej struktury danych, alokacja obszarów pamięci zewnętrznej oraz wyznaczanie i weryfikacja kryteriów projektowych.

Ważną cechą charakterystyczną projektowania fizycznej struktury danych jest iteracyjny charakter tego procesu. Ponieważ logiczny projekt bazy danych, jak również algorytmy procesów użytkowych mogą mieć istotny wpływ na parametry eksploatacyjne zachodzi czasami konieczność powrotu do tych faz projektowych bazy danych. Ze względu na dużą pracochłonność,

a tym samym wysoki koszt procesu projektowania fizycznej struktury danych, prowadzi się intensywne badania w dziedzinie komputerowego wspomagania tej fazy prac nad projektem bazy danych.

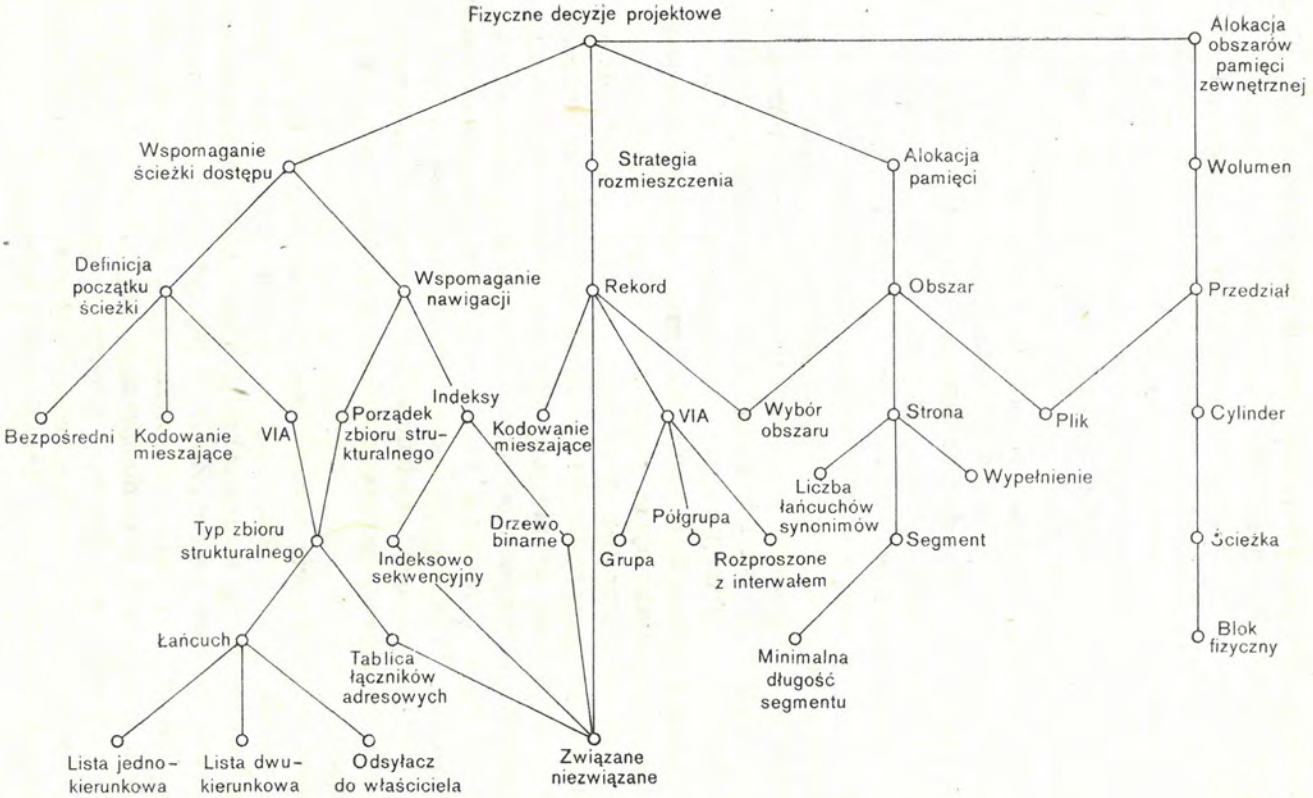
2. Opis fizycznej struktury danych

Powiedzieliśmy już poprzednio, że podstawowym celem fizycznego projektu bazy danych jest znalezienie możliwie najlepszej, z punktu widzenia wymagań przetwarzania, fizycznej postaci bazy danych. Repertuar decyzji w tym zakresie zależy przede wszystkim od możliwości wykorzystywanego Systemu Zarządzania Bazą Danych. W tej sytuacji przyjęliśmy za podstawę naszych rozwiązań raporty Komitetu CODASYL [1]. Wiele systemów ZBD spełnia już większość funkcji zawartych w tych raportach, tak więc można je uznać za powszechnie dostępne. Przyjęty opis fizycznej struktury danych zależy naturalnie od podejmowanych fizycznych decyzji projektowych, a razem z opisem logicznej struktury danych stanowi kompletny schemat bazy danych.

Ze względu na stosunkowo dużą elastyczność fizycznej struktury danych zaproponowanej w raportach Komitetu CODASYL, zakres i wzajemne powiązania decyzji projektowych dotyczących fizycznej struktury danych tworzą dosyć skomplikowany układ. Ta sytuacja jest wyraźnie widoczna na grafie fizycznych decyzji projektowych, przedstawionym na rysunku 66. Widać tam również, że różne decyzje projektowe wyrażone są jako ten sam element opisu (np. kodowanie mieszające CALC), będąc naturalnie różnymi węzłami grafu fizycznych decyzji projektowych. Taka sytuacja wynika przede wszystkim z niedoskonałości obecnie stosowanych języków opisu i powinna zostać wyeliminowana w przyszłości.

Wszystkie fizyczne decyzje projektowe dzielimy na trzy podstawowe kategorie: na decyzje wspomagające ścieżki dostępu danych, decyzje o strategii rozmieszczania rekordów zapisywanych do bazy danych oraz decyzje związane z alokacją pamięci. Wszystkie trzy kategorie decyzji projektowych są ściśle związane

[1] Por. *CODASYL Data Description Language Committee DDL, 1973 Report*, wyd. cyt.



Rys. 66. Graf fizycznych decyzji projektowych

z wymaganiami przetwarzania i mają bezpośredni wpływ na osiągane parametry eksploatacyjne.

Mówiliśmy już o nawigacyjnym charakterze wyszukiwania danych na podstawie komend wyszukujących (FIND) Języka Manipulacji Danymi. Tak więc wspomaganie ścieżki dostępu wymaga określenia jej początku, czyli tzw. punktu wejścia do struktury danych oraz zasad przejścia przez opisywaną ścieżkę, czyli elementów wspomagania nawigacji. Podstawową cechą rekordu będącego początkiem ścieżki dostępu jest możliwość jednoznacznej selekcji na podstawie jego adresu (klucz bazy danych) lub wartości zawartych w nim danych elementarnych. W wypadku wybierania rekordu przez adres mamy do czynienia z odwzorowaniem bezpośrednim (przez klucz bazy danych). Ten sposób adresowania należy stosować bardzo ostrożnie, by nie naruszyć fizycznej niezależności programów użytkowych od danych. W przeciwnym wypadku nawet tak standardowe działanie Administratora Bazy Danych jak fizyczna reorganizacja może spowodować konieczność modyfikacji programów użytkowych.

Znacznie lepszą, z tego punktu widzenia, techniką adresowania jest kodowanie mieszające, gdzie argumentem procedury przydziału, oraz — naturalnie — selekcji, adresu są wartości danych elementarnych zawartych w rekordzie (tzw. klucze randomizacji). Kodowanie mieszające nie jest więc zagrożeniem dla fizycznej niezależności programów od danych. Nie jest to jednak tak efektywne z punktu widzenia czasów realizacji, jak adresowanie bezpośrednie. Podstawowym powodem opóźnień jest tu konieczność obsługi synonimów randomizacji oraz każdorazowe wykonanie obliczeń adresu zgodnie z algorytmem kodowania mieszającego.

Ostatnią możliwością wyznaczania punktu wejścia do struktury danych jest wykorzystanie uczestnictwa rekordu w tzw. pojedynczym zbiorze strukturalnym. Właścicielem takiego zbioru strukturalnego jest system ZBD. W tej sytuacji zawsze mamy do czynienia wyłącznie z jednym jego wystąpieniem. Zastosowanie opcji VIA jest dopuszczalne jedynie wtedy, gdy istnieje możliwość jednoznacznej selekcji rekordu na podstawie wartości jego danych elementarnych spośród innych rekordów należących do tego samego zbioru strukturalnego.

Nawigacja po zbiorach strukturalnych polega albo na sekwencyjnym przejściu przez odpowiednie wystąpienia zbiorów, albo na wykorzystaniu indeksów. W wypadku, gdy mamy do czynienia z tablicą łączników adresowych, możemy zastosować cięcie binarne dla uporządkowanych zbiorów strukturalnych. Sposób uporządkowania zbioru strukturalnego ma istotny wpływ na parametry czasowe selekcji rekordów. Do najczęściej stosowanych indeksów zbiorów strukturalnych należą indeksy oparte na zasadzie zbiorów indeksowo-sekwencyjnych lub indeksów skonstruowanych jako drzewo binarne. W wypadku pierwszego typu indeksowania możemy stosunkowo efektywnie wybierać rekordy na podstawie zakresów kluczy (od — do). Natomiast drugi wypadek jest lepszy dla jednoznacznej selekcji wystąpienia rekordu na podstawie wartości kluczy selekcji.

Istotną decyzją projektową, z punktu widzenia wspomagania nawigacji, jest wybór typu zbioru strukturalnego. W wypadku zbiorów strukturalnych typu listowego odpowiednie czasy ich aktualizacji zależą od wybranych opcji łączników adresowych. Tablica łączników adresowych jest preferowanym typem zbioru strukturalnego wówczas, gdy mamy do czynienia z uporządkowanym zbiorem strukturalnym. Selekcji dokonuje się najczęściej na podstawie wartości kluczy sortowania. Możliwe do zastosowania w takim wypadku przeszukiwanie, polegające na cięciu dwójkowym, w istotny sposób podnosi czasową efektywność selekcji.

Strategia rozmieszczenia dotyczy bezpośrednio rekordu, jednakże również wywiera pośrednio wpływ na wszystkie pozostałe elementy struktur danych. Efekt równomiernego (lub bliskiego równomiernemu) uzyskujemy przez zastosowanie kodowania mieszającego. Często ten efekt jest traktowany jako szkodliwy efekt uboczny zastosowania kodowania mieszającego do adresowania.

W wypadku stosowania przetwarzania sekwencyjnego korzystne może być odpowiednie grupowanie rekordów. Tworzenie hierarchicznych układów rekordów wynikających z ich uczestnictwa w zbiorach strukturalnych można uzyskać stosując opcję VIA. Stosując opcję VIA wraz z odpowiednio dobranymi technikami ładowania bazy danych możemy uzyskać efekt pełnego lub częściowego grupowania rekordów lub efekt kontrolowanego

rozproszenia rekordów z zachowaniem wymaganych odstępów. Istotnym elementem wykorzystania obszarów bazy danych jest rozmieszczenie tablic łączników adresowych oraz indeksów. Mamy tutaj zwykle możliwość umieszczenia tych elementów fizycznej struktury danych w bezpośrednim sąsiedztwie wystąpień rekordów-właścicieli odpowiednich zbiorów strukturalnych lub w dowolnie wskazanych obszarach bazy danych.

Wykorzystanie urządzeń pamięci zewnętrznej wynika bezpośrednio z decyzji projektowych w zakresie alokacji pamięci. Obszar bazy danych jest zbiorem stron, a jego wielkość zależy od długości i liczby stron. W każdym obszarze bazy danych można umieszczać dowolną liczbę typów rekordów, a jeden typ rekordu może być przypisany do wielu obszarów. W takim wypadku wybór konkretnego obszaru bazy danych następuje w momencie zapisu wystąpienia określonego typu rekordu.

Istotnymi cechami charakterystycznymi strony bazy danych jest przyjęta dla danego obszaru liczba łańcuchów synonimów randomizacji oraz założony stopień wypełnienia stron. Pierwszy parametr ma bezpośredni wpływ na długość list synonimów randomizacji (odwrotnie proporcjonalna do liczby łańcuchów synonimów), a drugi ma istotne znaczenie dla obszarów o dużej zmienności. W wypadku dużej ilości modyfikacji, polegających na zmianie długości rekordów, istnienie rezerwy miejsca na stronach może mieć istotne znaczenie dla czasów realizacji operacji aktualizujących bazę danych. Strona bazy danych jest podzielona na segmenty, a przyjęta dla rekordu danego typu minimalna długość segmentu ma znaczenie dla rozmieszczenia wystąpień rekordów w bazie danych.

Przedstawiony zbiór fizycznych decyzji projektowych determinuje uzyskane parametry eksploatacyjne projektowanej bazy danych. Stopień wzajemnego uzależnienia podejmowanych decyzji projektowych znacznie utrudnia proces projektowania fizycznej struktury danych. Najczęściej uzyskanie wymaganych parametrów eksploatacyjnych jest możliwe dopiero po wykonaniu szeregu iteracji w ramach realizowanego projektu. Wysoki koszt każdej z takich iteracji jest jednym z podstawowych bodźców dla doskonalenia technik projektowania fizycznej struktury danych.

W większości systemów ZBD opracowanych zgodnie z raportami Komitetu CODASYL zakłada się, że obszar bazy danych odpowiada plikowi systemu operacyjnego. Powiązanie między tymi dwoma elementami jest realizowane za pośrednictwem Języka Opisu Zadań (*Job Control Language*) dostępnego w ramach systemu operacyjnego. W wypadku rodziny systemów operacyjnych OS firmy IBM (eksploatowanym również na komputerach Jednolitego Systemu) powiązanie między obszarem bazy danych a plikiem jest opisywane w zdaniu (*Data Description*) Języka Opisu Zadań. Język Opisu Zadań jest więc podstawowym narzędziem opisu przyjętej w danym systemie informatycznym strategii rozmieszczania obszarów bazy danych na nośnikach urządzeń pamięci zewnętrznej.

Projekt fizycznej struktury danych ma bezpośredni wpływ na liczbę fizycznych transferów danych koniecznych do wykonania jednej operacji logicznej (np. „włącz do zbioru strukturalnego"). Natomiast czas wykonania jednego transferu fizycznego zależy od charakterystyki technicznej urządzeń pamięci zewnętrznej, rozmieszczenia obszarów bazy danych na tych urządzeniach oraz obciążenia poszczególnych elementów konfiguracji komputera. Właśnie elementy konfiguracji urządzeń pamięci zewnętrznej (kanały, jednostki sterujące, urządzenia pamięci zewnętrznej) są najczęściej „wąskimi gardłami" konfiguracji komputerów. Stąd też alokacja obszarów pamięci zewnętrznej jest bardzo istotnym elementem fizycznego projektu bazy danych. Dopiero po zakończeniu prac nad projektem fizycznej struktury danych oraz alokacją obszarów urządzeń pamięci zewnętrznej można podać próbę szacunku takich parametrów eksploatacyjnych systemu informatycznego jak czas odpowiedzi, przepustowość systemu oraz zajętość pamięci operacyjnej.

3. Kryteria oceny projektu fizycznej struktury danych

Kryteria oceny projektu fizycznej struktury danych, a właściwie parametrów eksploatacyjnych systemu informatycznego wynikających z tego projektu, można rozpatrywać jako dwie podstawowe grupy ograniczeń: ograniczenia czasowe i ograniczenia pamięci. W wypadku drugiej grupy ograniczeń musimy brać

pod uwagę zajętość pamięci operacyjnej, jak i zajętość pamięci zewnętrznej.

Ograniczenia czasowe są związane z czasem odpowiedzi rozpatrywanym na poziomie komendy Języka Manipulacji Danymi, zwartej sekwencji komend oraz transakcji. Czas odpowiedzi komendy Języka Manipulacji Danymi jest zawarty między odwołaniem do tej komendy a momentem zwrotu sterowania do programu użytkowego. Czas odpowiedzi zwartej sekwencji akcji JMD jest równy sumie czasów odpowiedzi zawartych w niej komend Języka Manipulacji Danymi plus czas potrzebny na przetwarzanie wykonane przez zdania bazowego języka programowania.

Określenie czasu zwartej sekwencji komend ma istotne znaczenie dla wyznaczenia możliwych efektów interferencji współbieżnych procesów aplikacyjnych. Ochrona integralności danych wymaga blokowania elementów struktury danych na cały okres trwania zwartej sekwencji akcji, co z kolei może doprowadzić do zablokowania na ten okres współbieżnego procesu użytkowego. Czas odpowiedzi transakcji jest sumą czasów odpowiedzi zawartych w niej zwartych sekwencji komend. Jeżeli przewiduje się wysoki stopień interferencji współbieżnych procesów, to czas odpowiedzi może być wyznaczony jedynie eksperymentalnie lub za pomocą odpowiednich narzędzi modelujących eksploatację bazy danych (np. symulator bazy danych).

Dodatkowymi ograniczeniami czasowymi, jakie mogą być istotne dla projektowanego systemu informatycznego, to maksymalny okres wyłączenia bazy danych oraz minimalny czas potrzebny do odtworzenia bazy danych.

Wymaganie dotyczące maksymalnego czasu wyłączenia bazy danych nie musi być bezpośrednio związane z minimalnym czasem koniecznym do jej odtworzenia. W wypadku, gdy czas odtwarzania bazy danych jest dłuższy niż dopuszczalny czas jej wyłączenia, należy stosować specjalne rozwiązania projektowe. Jednym z możliwych rozwiązań jest równoległa aktualizacja drugiej kopii bazy danych i odpowiednie przełączanie w razie awarii, konieczność stosowania rozwiązań tego typu występuje jedynie w systemach pracy „na bieżąco". Ma to znaczenie dla zastosowań bazy danych do sterowania w systemach rezerwacji itp.

Jeżeli zmiany projektu fizycznego nie przyniosą oczekiwanej poprawy parametrów czasowych, należy rozważyć odpowiednie zmiany w algorytmach procesów użytkowych oraz logicznej strukturze danych.

Ograniczenia pamięciowe obejmują zwykle takie grupy zagadnień, jak wymagania systemu ZBD, wymagania procesów użytkowych oraz zajętość pamięci zewnętrznej. Wymagania systemu ZBD dotyczą przydziału pamięci operacyjnej na moduły systemu, bloki sterujące, pulę buforów stron, bufory dla zapisu dzienniczka systemu i danych pomiarowych oraz pamięci przydzielonej na utrzymanie kolejek.

Zwykle, jeżeli wystąpi blok odpowiedniego obszaru pamięci operacyjnej, system ZBD wykorzystuje pamięć zewnętrzną dla przechowania takich elementów, jak kolejki systemowe, bloki sterujące itp. Naturalnie wykorzystanie pamięci zewnętrznej wpływa w tym wypadku na istotne pogorszenie parametrów eksploatacyjnych systemu.

W wypadku procesu użytkowego rozpatrujemy jedynie zajętość pamięci, wynikającą z funkcji manipulacji danymi. Podstawowymi elementami procesu użytkowego wpływającymi na zajętość pamięci są: obszar roboczy użytkownika, bloki sterujące (podschemat, wskaźniki bieżących rekordów), bufory pośrednie oraz oprogramowanie łączące z systemem ZBD. Rozpatrując zapotrzebowanie na obszary pamięci zewnętrznej bierzemy pod uwagę wszystkie elementy struktury danych oraz związane z nimi dane sterujące, takie jak łączniki adresowe, indeksy, tablice łączników adresowych oraz dane sterujące stron. Wszystkie te czynniki pozwalają na określenie dopuszczalnego rozwiązania, będącego wynikiem projektu bazy danych.

Ważnym problemem jest jeszcze weryfikacja uzyskanego rozwiązania z punktu widzenia przyjętych kryteriów projektowych. Ocena parametrów eksploatacyjnych wymaga dodatkowo uwzględnienia takich elementów normalnej eksploatacji bazy danych, jak prowadzenie dziennika systemu, pomiarów eksploatacyjnych oraz okresowych reorganizacji bazy danych. Wszystkie te elementy są przedmiotem projektu eksploatacji bazy danych. Ponieważ mogą one być istotnym obciążeniem dla eksploatowanego systemu informatycznego; przybliżone choćby określenie

czasowych i pamięciowych skutków wykonywania tych funkcji jest konieczne dla właściwej weryfikacji projektu z punktu widzenia przyjętych kryteriów użytkowych.

Proces weryfikacji projektu bazy danych z punktu widzenia parametrów eksploatacyjnych powinien być sformalizowaną procedurą. Istnieje przecież silne podobieństwo między tym procesem a wykonywaniem obliczeń konstrukcyjnych w dowolnej dziedzinie technicznej. Istotnym mankamentem jest brak sformalizowanych metod i narzędzi dla właściwej realizacji procesu weryfikacji projektu bazy danych. Na to uzupełnienie metodyki projektowania baz danych oczekują projektanci systemów.

Literatura

Anderson M., Staniszkis W., *Systems Development Facility*, Long Range Design Report wyd. Logica Ltd, London 1979.

ANSI/SPARC Interim-Report. *Study Group on Date Base Management Systems*, FDT. Bull. off ACM SIGMOD, vol. 7, 1975, nr 2.

Astrahan M. M., Chamberlin D. D., *Implementation of a Structured English Query Language*, Proc. SIGMOND, ACM, New York 1975.

Bachman C., *The Evolution of Storage Structures*, CACM, vol. 15, 1972, nr 7.

Bańkowski J., Fiałkowski K., *Wprowadzenie do informatyki*, PWN, Warszawa 1978.

Bayer R., *On Intergrity of Data Bases and Resource Locking*. W: *Data Base Systems*, pod red. Hasselmeina, Springer Verlag, Berlin 1976.

Belke W., Graichen D., Starrus M., *Nichtmetrische Klassifizierung von Informationen*, Akademic Verlag, Berlin 1979.

Cold E. F., *A Relational Model of Date for Large Shared Data* Banks, CACM, vol. 13, nr 13.

Codd E. F., *Relational Completeness of Data Base Sublanguages*, Proc. Courant Computer Science, Symposium 6, Prentice Hall, 1971.

CODASYL Data Base Task Group, April 1971, ACM, New York 1972.

CODASYL Data Base Task Group, 1971 Report, ACM, New York 1972.

CODASYL Data Description Language Committee DDL, Journal of Development, NBS, New York 1973.

CODASYL Data Description Language Committee 1978 Report, Information Systems, vol. 3, 1978, nr 4.

Dalkey N., *Studies in the Quality of Life Delphi and Decision Making*, Lexington Books, Lexington, M. A., 1972.

Data Dictionary Systems Working Party Report, The British Computer Society, 1977.

Date C. J., *Wprowadzenie do baz danych*, WNT, Warszawa 1981.

European Conference of Evolution and Implementation of Data Base Systems, Bruksela 1979.

Graves R., *Semantic Modelling in Statistics Canada*, Proceedings of ISS 80 Seminar, VVS, Bratislava 1980.

Haberman A., *Prevention of System Deadloks*, Comm. of ACM, 7, 1978, nr 12.

Jasin E., Sterninson D., *Informacjonnyj jazyk ekonomiczeskich pokazatielej*. W: *Osnownyje princypy sozdanija informacjonnogo obiespieczenija ASPR*, Moskwa 1972.

Langefors B., *Infological Models and Information Users View, Information System*, Pergamon Press, vol. 5, 1980.

Levnowitz P., *File Structures for On-line Systems*, Spartan Books, Hayden Book Comp. Inc., London 1969.

Linguistic Structures Processing, pod red. F. Zampolii, North Holland, Amsterdam 1977.

Lum Y. i in., *Key to Hidress Transformation Techniques: A Fundamental Performance Study on Large Existing Formatted Files*, CACM, vol. 14, 1971, nr 4.

Marci P. O. *Deadlocks Detection and Resolutions in a Codasyl Based Data Management of Data SIGMOD 76*, ACM, New York 1976.

Martin J., *Computer Data Base Organization*, Prentice Hall Ed., New York 1975.

Mądrzycki J., *Problemy projektowania języków wyszukiwawczych w faktograficznych systemach informacyjnych*, Aktualne Problemy Informacji i Dokumentacji, 1981, nr 4.

Meyer B., Schneider H., *Tools for Information Systems Design and Realization, Proccedings of the IFIP TC 8 Working Conference: Formal Models and Practical Tools of Information System Design*, Oxford 1979.

Nilsson B., *On Models and Mapping in Date Base Environment*, Urval 9, Stockholm 1980.

Nolan R., *Thoughts about the Fifth Stage Date Base*, Fall 1975, vol. 7, ACM, New York 1975, nr 2.

Oleński J., *Języki problemowo-zorientowane — podstawy projektowania*. W: *Informacyjne problemy planowania*, pod red. W. Maciejewskiego, PWE, Warszawa 1982.

Prażenka D., *Systemy riadenia bazy dat-dvadsať rokov vyvoja*. „Informacne Systemy" 1981, nr 1.

Schussel G., *The Role of the Date Dictionary*, Datamation, vol. 23, 1979, nr 6.

Skuce D., *An Approach to Defining and Communicating the Conceptual Structure of Data*, University of Ottava, Ottava 1979.

Staniszkis W., *Mechanizmy ochrony danych w systemach zarządzania bazą danych*, OBRI, Warszawa 1978.

Sundgren B., *An Infological Approach to Data Bases*, Urval 7, Stockholm 1974.

Szymański A., *SEQUEL — Język zapytań dla systemu RODAN*, CPiZI Warszawa 1980.

SZBD, RODAN, dokumentacja użytkowa, CPiZI, Warszawa 1980.

Turski W., *Struktury danych*, WNT, Warszawa 1970.

Wasiliauskas A. i in., *Osnovy awtomatizacji uprawlenija narodnym choziajstwom respubliki*, Wilinus 1975.

Wilson T., *Iskustwiennyj Intiellekt*, wyd. Nauka, Moskwa 1974.

Won Kim, *Relational Database Systems*, ACM, Computing Surveys, vol. 11, 1979, nr 3.

W serii „INFORMATYKA W PRAKTYCE" ukazały się dotychczas następujące pozycje:

MARTIN ZSCHOCKE
Elektroniczne przetwarzanie danych w gospodarce materiałowej
Warszawa 1975, s. 118, cena zł 18,—

AGATA ROJEK-GROSZEWSKA, ANDRZEJ ZALESKI
Gromadzenie danych do elektronicznego przetwarzania
Warszawa 1976, s. 350, cena zł 40,—

KIT GRINDLEY, JOHN HUMBLE
Skuteczność wykorzystania komputera
Warszawa 1976, s. 245, cena zł 45,—

STANISŁAW ZADROŻNY
Organizacja zbiorów w małej informatyce
Warszawa 1977, s. 159, cena zł 25,—

EDWARD KOLBUSZ, EDWARD KRAM
Wdrażanie systemów informatycznych w przedsiębiorstwach przemysłowych
Warszawa 1977, s. 268, cena zł 36,—

ANDRZEJ JORDAN
Organizacja zbiorów w pamięciach dyskowych
Warszawa 1977, s. 129, cena zł 19,—

Praca zbiorowa
Przechowywanie danych
(tłum. z jęz. niem.)
Warszawa 1977, s. 154, cena zł 23,—

ZYGMUNT RYZNAR
Bank danych w przedsiębiorstwach przemysłowych
Warszawa 1978, s. 172, cena zł 26,—

ANTONI NOWAKOWSKI, WOJCIECH OLEJNICZAK
Minikomputery biurowe
Warszawa 1978, s. 168, cena zł 25,—

BRONISŁAW OBIREK
Przygotowanie przedsiębiorstwa do zastosowania informatyki w zarządzaniu
Warszawa 1978, s. 151, cena zł 23, —

IGNACY DZIEDZICZAK
Model księgowości informatycznej w przedsiębiorstwie
Warszawa 1979, s. 211, cena zł 38,—

JANUSZ ILCZUK, MARIA JERCZYŃSKA
Efektywność systemów informatycznych zarządzania
Warszawa 1979, s. 208, cena zł 38,—

ANDRZEJ Z. IDŹKIEWICZ
Ochrona informacji w procesie przetwarzania
Warszawa 1979, s. 147, cena zł 26,—

ANDRZEJ JAKUBOWICZ
Rachunek kosztów w hierarchicznym systemie informacji
Warszawa 1979, s. 147, cena zł 30,—

ADAM NOWICKI
Modernizacja systemu informacyjnego w przedsiębiorstwie przemysłowym
Warszawa 1979, s. 235, cena zł 42,—

ROMUALD JAGIELSKI
Komputery Jednolitego Systemu
Warszawa 1980, s. 362, cena zł 53,—

ZOFIA MENET
Mała informatyka. Środki i zastosowanie
Warszawa 1980, s. 170, cena zł 30,—

JERZY KISIELNICKI
Ekonomiczne problemy zautomatyzowanych systemów zarządzania
Warszawa 1981, s. 288, cena zł 50,—

B. BUŚKO, H. FILIPEK, J. ŚLIWIEŃSKI
Wiarygodność informacji ekonomicznej w systemach informatycznych
Warszawa 1982, s. 288, cena zł 55,—

W. N. LEBIEDIEW, A. P. SOKOŁOW
System operacyjny OS JS. Podstawy użytkowania
Warszawa 1982, s. 226, cena zł 60,—

IGNACY DZIEDZICZAK
Organizacja bazy danych księgowych
Warszawa 1983, s. 172, cena zł 100,—

Cena zł 160,—

ISBN 83-208-0350-0