POLITECHNIKA ŚLĄSKA
IM. W. PSTROWSKIEGO

SKRYPTY UCZELNIANE
Nr 1123

WALDEMAR KAPUŚCIK

PROJEKTOWANIE BAZ DANYCH TYPU CODASYL

GLIWICE 1988

POLITECHNIKA ŚLĄSKA
IM. W. PSTROWSKIEGO

SKRYPTY UCZELNIANE
Nr 1123

WALDEMAR KAPUŚCIK

PROJEKTOWANIE BAZ DANYCH TYPU CODASYL

GLIWICE 1983

OPINIODAWCA

Doc. dr Zdzisław Zapolski

KOLEGIUM REDAKCYJNE

Wiesław Gabzdyl (redaktor naczelny), *Stanisław Kozielski* (redaktor działu), *Elżbieta Stinzing* (sekretarz redakcji)

REDAKCJA

Anna Błażkiewicz

REDAKCJA TECHNICZNA

Alicja Nowacka

Wydano za zgodą
Rektora Politechniki Śląskiej

Skrypt przeznaczony dla studentów Wydziału Automatyki i Informatyki (kierunek Informatyka)

PL ISSN 0434-0825

Dział Wydawnictw Politechniki Śląskiej
ul. Kujawska 3, 44-100 Gliwice

Nakł. 200+55 Ark. wyd. 13,2 Ark. druk. 13,375 Papier offset. kl. III. 70x100 70 g
Oddano do druku 15.03.1983 Podpis. do druku 18.07.1983 Druk ukończ. we wrześniu 1983
Zam 494/83 Z-23 Cena zł 106,—

Skład, fotokopie, druk i oprawę
wykonano w Zakładzie Graficznym Politechniki Śląskiej w Gliwicach

SPIS TREŚCI

Str.

1. WSTĘP

Problematyka baz danych, ich projektowanie i eksploatacja jest stosunkowo młodą dziedziną informatyki. Pierwsze prace badawczo-koncepcyjne w tym zakresie podjęła na przełomie lat sześćdziesiątych i siedemdziesiątych specjalistyczna grupa robocza Data Base Task Group międzynarodowego Komitetu CODASYL (Conference of Data Systems Language). Wypracowane przez nią koncepcje są coraz częściej stosowane przez najpoważniejszych producentów sprzętu informatycznego, aczkolwiek nie jest to jedyne możliwe rozwiązanie w tym zakresie (np. innym rozwiązaniem są relacyjne bazy danych). Prace grupy DBTG spowodowały inne spojrzenie na problem metodyki projektowania i przechowywania informacji. Nowe zastosowania informatyczne, coraz większe zbiory danych, a co za tym idzie, coraz dłuższy czas ich przetwarzania zmusiły do zmiany metod organizacji danych. Przy okazji zauważono pewien naturalny związek istniejący między pozornie niezależnymi danymi i spróbowano wykorzystać go do usprawnienia pracy programów. Do tego czasu podstawowymi jednostkami przechowywania zgrupowanych danych były zbiory. Oczywiście mogły być one umieszczane na różnych nośnikach (np. pamięć taśmowa, bębnowa, dyskowa, karty perferowane, taśmy papierowe), mogły być wykorzystane przez różne programy. Niestety jednak wiele programów korzystających w gruncie rzeczy z tych samych informacji musiało używać fizycznie różnych zbiorów. Elementy w zbiorach (rekordy) posiadały z reguły tę samę lub podobną treść informatyczną. Zbiory różniło jedynie uporządkowanie elementów, czy ich liczba. Stąd właśnie wynikała często potrzeba wykonywania na zbiorach pracochłonnych operacji typu porządkowego, takich jak: sortowanie, wybieranie czy scalanie. Stąd się wzięło niepotrzebne wielokrotne przechowywanie tej samej informacji w różnych układach i przekrojach, niestety na różnych fizycznie urządzeniach. Zbiory zostały podporządkowane programom, tylko dla nich istniały (eng. program oriented files).

Wprowadzenie baz danych, a więc centralnego przechowywania odpowiednio zorganizowanych danych gromadzonych dla potrzeb wielu programów użytkowych, zasadniczo polepszyło istniejącę sytuację. Podstawowe zalety systemów z bazami danych w porównaniu z systemami tradycyjnymi są następujące:

1) Minimalizacja redundancji informacji (aczkolwiek może istnieć celowa redundancja pewnych danych minimalizująca czas transakcji).

2) Uniezależnienie danych od programów, co znacznie przyspiesza proces przygotowywania i testowania nowych programów.

3) Logiczne powiązanie danych między sobą pod kątem konkretnych zastosowań, co kapitalnie ułatwia dostęp do danych znacznie przyspieszając pracę programów.

3) Centralne administrowanie danymi ułatwiające kontrolę danych, ich ochronę i blokowanie.

Definicja bazy danych modyfikowała się w miarę rozwoju jej zastosowań. Prześledźmy kilka bardziej popularnych definicji różnych autorów:

1) Bazą danych nazywamy duży plik lub grupę plików (Knuth D.E.).

2) Użytkowa baza danych jest to zbiór plików przechowywanych w formie bloków o bezpośrednim dostępie (Pinherio C.).

3) Baza danych to zbiór danych lub zbiór takich zbiorów, zorganizowany w sposób naturalny dla danych, które się na niego składają, nie mających powiązania z wybranym algorytmem przetwarzania (Emery G.).

4) Baza danych to zbiór przechowywanych w pamięci danych operacyjnych używanych przez systemy użytkowe dla specjalnych zastosowań (Data C.J.).

5) Bazą danych nazywamy zbiór danych zorganizowanych tak, by mogły być wykorzystywane przez różnych programistów (Martin I.).

Celowo pominięte zostały w tym miejscu niektóre definicje relacyjnych baz danych (Deheneffe, Codd), gdyż tego typu bazy nie są przedmiotem niniejszego opracowania. Z przedstawionych definicji oraz z przykładów konkretnych implementacji baz danych w różnych systemach wynikają pewne cechy wspólne, mogące służyć do określania tego pojęcia. Są one następujące:

1) Baza danych to grupa zbiorów o określonej strukturze zorganizowanych w sposób naturalny dla danych.

2) Dane tworzące bazę przechowywane są w pamięci (masowej).

3) Dane zorganizowane są w taki sposób, by mogły być wykorzystane, nawet równocześnie, przez różnych użytkowników.

4) Dane zmieniają się w czasie.

Wymaga wyjaśnienia owa "naturalna" współzależność danych. Że jest tak w istocie, rozpatrzmy na najprostszym przykładzie. W pewnym systemie informatycznym posiadamy zbiór wszystkich pracowników danego zakładu i wykaz działów (wydziałów) tego przedsiębiorstwa. Dane te pozornie niezależne od siebie łączy relacja DZIAŁ - PRACOWNIK, gdyż każdy pracownik jest w naturalny sposób przydzielony do jakiegoś działu (p. przykład 1 z rozdz. 2.3.4).

Ostatnie lata przyniosły jeszcze jeden ciekawy aspekt w tej sprawie. Jest nim zauważalny odwrót od szerokiego zastosowania baz danych w każdej dziedzinie do wybranych tylko zagadnień. Spowodował to stosunkowo wysoki koszt wdrożenia i eksploatacji baz i to zarówno koszt dosłowny (skomplikowane oprogramowanie, szkolenie programistów, organizacja działu administrującego bazę), jak również koszt w sensie informatycznym (wydłużenie

czasu pracy niektórych programów, zajętość pamięci masowej, zajętość pamięci operacyjnej przez system obsługi bazy). Nastąpiło ograniczenie stosowania baz danych do systemów pracujących w czasie rzeczywistym i takich, w których wymagany jest szybki dostęp do informacji. Oczywiście z już istniejącej i załadowanej bazy mogą korzystać różne programy (np. w trybie wsadowym), jednak projektowanie odbywać się powinno wyłącznie pod kątem transakcji umożliwiających konwersacyjny dostęp do danych. Ten właśnie element można dołączyć jeszcze do określenia bazy danych:

5) Z bazy danych korzystają głównie programy pracujące w czasie rzeczywistym lub w systemie komunikacyjnym.

Minimalizacja czasu pracy transakcji w projektowanym systemie ma podstawowe znaczenie. Na czas wykonania transakcji składają się następujące czasy: komunikacji z operatorem (obiektem) poprzez urządzenia wejścia/wyjścia, czas realizacji pewnej liczby elementarnych operacji dostępu do bazy (usunięcie, wprowadzenie, modyfikacja) oraz czas przetwarzania informacji. Organizacja bazy danych, która wpływa decydująco na czas realizacji operacji elementarnych, jest definiowana przez użytkownika, podczas gdy pozostałe czasy są zależne od konfiguracji sprzętowej oraz systemu operacyjnego i z reguły nie mogą być przez użytkowników modyfikowane. O czasie wykonywania elementarnych operacji na bazie danych decyduje liczba związanych z nimi fizycznych operacji wejścia/wyjścia (wymiana informacji między pamięcią masową a operacyjną). Liczba fizycznych dostępów jest w większości przypadków niemożliwa do określenia na etapie projektowania bazy, gdyż do określenia jej potrzebne byłoby posiadania informacji o rozmieszczeniu rekordów w pamięci zewnętrznej (mapy pamięci). Dlatego też do celów projektowania bazy (porównywania alternatywnych struktur) bardziej nadaje się metoda obliczania liczby logicznych dostępów (p. rozdz. 5).

Niniejsze opracowanie dotyczyć będzie właśnie metodyki projektowania baz danych. Od razu narzucają się pytania. Jaka jest istota projektowania baz? Co różni ten proces od tradycyjnego projektowania? Jakie są etapy projektowania? Spróbujmy pokrótce odpowiedzieć na te pytania. Najprościej, proces projektowania bazy danych można podzielić na dwa etapy: projektowanie struktury logicznej oraz projektowanie struktury fizycznej bazy. Pierwszy etap to w gruncie rzeczy odpowiedzi na generalne pytania: Jak dane powiązać między sobą? Jak je pogrupować w rekordy? Jak je najefektywniej przetwarzać? Etap drugi to z kolei powiązanie danych z konkretnymi fizycznymi urządzeniami (nośnikami); to zadecydowanie o wielkości bazy, o metodach dostępu i sposobach zabezpieczenia. Do tego zagadnienia można jeszcze podejść inaczej. Zasadniczo są dwie drogi (częściowo rozłączne) prowadzące do zdefiniowania bazy danych. Są to mianowicie:

1) Analiza istniejących danych (w kartotekach niekomputerowych, zbiorach tradycyjnych itp.), ich związku między sobą, wielkości i liczności.

2) Analiza projektowanych programów, ich potrzeb, powiązań i ograniczeń.

Obie te drogi mogą doprowadzić do optymalnego celu, jakim jest efektywna baza danych. Wydaje się jednak, że znacznie większy wpływ na ostateczny wynik ma analiza projektowanych programów. To pod kątem ich konkretnych zastosowań, metod i algorytmów powinno definiować się optymalne struktury danych. Optymalne oczywiście dla konkretnego programu czy systemu. Takie podejście modyfikuje całkowicie tryb projektowania systemu. Wymaga ono udziału specjalistów baz danych (administratora bazy danych) już w początkowych etapach projektowania, takich jak np. definiowanie funkcji systemu. Niedopuszczalną praktyką jest zaś rozdzielenie zespołu projektowego na zespół projektujący programy i zespół projektujący dane (bazę).

Niniejsze opracowanie zawiera pewną propozycję sformalizowanego podejścia do projektowania baz danych. Co prawda opiera się ono na definicjach grupy DBTG komitetu CODASYL oraz na konkretnej implementacji systemu zarządzania baz danych (DMS 1100 dla komputerów firmy SPERRY UNIVAC), nie zmniejsza to jednak istotnie ogólności rozważań i nie ogranicza możliwości zastosowania. Przedstawioną metodę można stosować do innych implementacji bazujących na ustaleniach komitetu CODASYL (np. RODAN[x)] oraz do innych baz opartych na sieciowym modelu danych, których realizacja nastąpiła za pomocą pól indeksowych (wskaźników). Większym problemem wydaje się być zastosowanie przedstawionej metody obliczania fizycznej struktury danych, gdyż jest ona w większym stopniu uzależniona od systemu komputerowego. Wydaje się jednak, że podobny tryb postępowania (ewentualnie z małymi modyfikacjami), a w najgorszym przypadku tylko pewna prawdy ogólne, można przyjąć do większości implementacji. Zaproponowane standardy dokumentacyjne w każdym przypadku ułatwią formalny zapis kolejnych etapów projektowania.

[x)] SZBD RODAN opracowany w OBRI - Warszawa, przeznaczony dla komputerów serii RIAD i IBM/360/370 pracujących z systemem operacyjnym OS/MVT.

2. PODSTAWOWE POJĘCIA

2.1. Obszar (ang. area)

Obszar jest podstawowym pojęciem związanym z codasylowską koncepcją bazy danych. Jest on odpowiednikiem zbioru w tradycyjnych systemach informatycznych, pojęciem jednak mocno uogólnionym i w związku z tym szerszym. Jest to fizyczny podzbiór bazy danych. W szczególnym przypadku może on obejmować całą bazę danych. Jako zbiór jest on identyfikowany, zarządzany i chroniony przez system operacyjny lub inny wyspecjalizowany procesor (np. TIP w systemie komputerowym UNIVAC-1100). Obszar jest przydzielony do konkretnego fizycznego urządzenia. Posiadając różne typy pamięci masowej mamy w tym miejscu możliwość przydzielania częściej wykorzystywanych obszarów do szybszych urządzeń. Co więcej, niektóre systemy operacyjne zezwalają na przydział urządzeń dyskowych w zadanym z góry zakresie ścieżek. Cechę tę można wykorzystać przydzielając często używanemu obszarowi centralne miejsce na dysku (w środkowym zakresie ścieżek), co kapitalnie zmniejsza czas dostępu do tego zbioru na skutek minimalizacji ruchu głowic.

Definicja obszaru różni się w kilku miejscach od klasycznego określenia zbioru w informatyce, a mianowicie:

1) W obszarze mogą być przechowywane różne typy rekordów o stałej i zmiennej długości.

2) Rekordy tego samego typu mogą być przechowywane w rożnych obszarach, a nawet we wskazanych ich częściach.

3) W tym samym obszarze mogą być przechowywane rekordy o różnym typie alokacji, np. direct, calc, indeksowo-sekwencyjny, swobodny (p. rozdz. 2.2).

4) Obszar może być przeglądany sekwencyjnie lub metodą swobodną (random).

5) Obszar składa się z bloków (stron) stałej długości, zawierających zmienną liczbę rekordów.

Brak tutaj cechy typu "BLOCK CONTAINS N RECORDS" występującej np. w COBOL-u.

6) Ochrona integralności bazy danych, sposoby zabezpieczenia jej przed zniszczeniem definiowane są na poziomie obszaru.

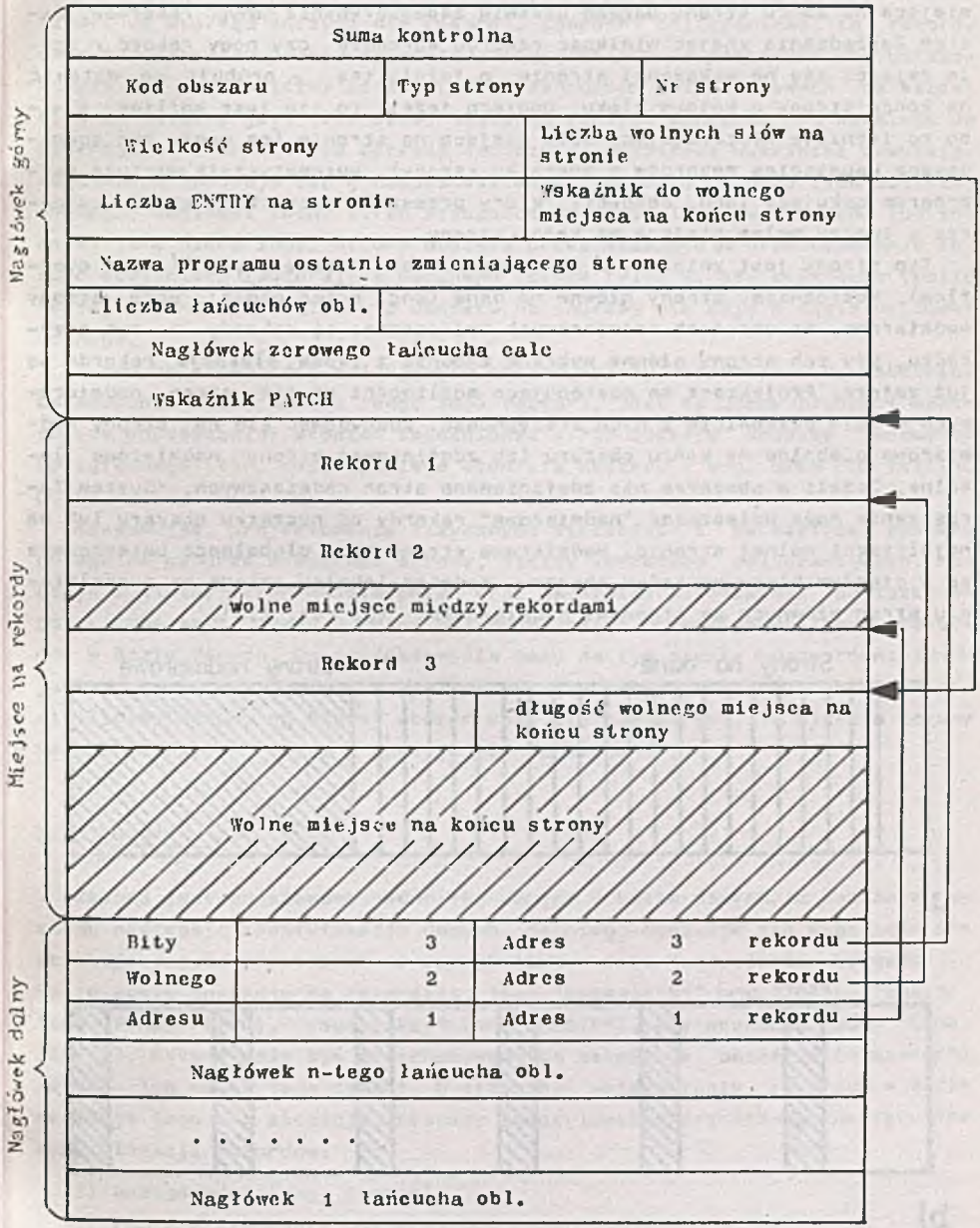
Jak widać, różnice w porównaniu z podejściem klasycznym są duże. Zwraca uwagę spora dowolność i elastyczność takiego podejścia. W gruncie rze-

czy za obszar można uważać szereg identycznych pustych pudełek, w których można przechowywać różne przedmioty. O wyborze właściwego pudełka (czytaj: strony) decyduje System Zarządzania Bazą Danych (w systemie DMS jest nim DMR - Data Management Routine). Przed ładowaniem początkowym obszaru i normalnym z niego korzystaniem niezbędne jest odpowiednie przygotowanie tego obszaru do pracy. Procedurą realizującą to jest tzw. inicjowanie obszaru. Polega ono na podziale przydzielonego zakresu pamięci masowej na bloki o zadanej stałej długości i odpowiednim wygenerowaniu nagłówków tych bloków (patrz rys. 2.1). Również rozszerzenie obszaru przez zwiększenie liczby stron wymaga zainicjowania w analogiczny sposób nowych stron tego obszaru.

Obszar jest identyfikowany przez nazwę oraz numeryczny kod (używany przez system). Oba parametry muszą być unikalne i to w ramach nie tylko jednej bazy danych, ale w ogóle w ramach wszystkich baz całego systemu komputerowego. Z uwagi na to, że System Zarządzania Bazą Danych może obsługiwać jednocześnie wiele baz, ta unikalność jest niezbędna głównie do zapewnienia integralności danych. Zabezpieczanie obszaru przez zapamiętywanie na taśmie śladowej systemu wszelkich zmian, jakie w nim zachodzą ("before looks", "after looks"), wymaga pełnej identyfikacji bloków opisujących te zmiany. To właśnie umożliwia potem poprawne odtwarzanie wybranego obszaru do dowolnego stanu z przeszłości.

System Codasyl przewiduje trzy typy obszarów. Podział implikuje rodzaj danych, jakie można w nich przechowywać. Głównym typem obszaru jest obszar przeznaczony na dane użytkownika, a więc na różnego rodzaju rekordy. Pozostałe dwa typy obszarów to obszary na dane systemowe, a mianowicie na tablice wskaźników dla specjalnego typu grupy logicznej (ang. pointer-array set) oraz na indeksy używane w metodzie indeksowo-sekwencyjnego przechowywania rekordów. Obszary każdego typu składają się z określonej liczby bloków stałej długości, zwanych stronami (ang. page). Strony są jednostkami danych biorącymi udział w operacjach WE/WY. Każda strona ma z góry określoną wielkość (dla danego obszaru) ograniczoną parametrami technicznymi urządzenia. Stąd biorą się ograniczenia typu: "wielkość strony musi być wielokrotnością sektora czy ścieżki", różne w różnych konfiguracjach sprzętu. Każda strona powinna mieć część systemową (nagłówek górny i dolny) oraz miejsce na przechowywanie danych. Części systemowe stron zależą oczywiście od Systemu Zarządzania Bazą Danych, jednak można tu napotkać podobne elementy.

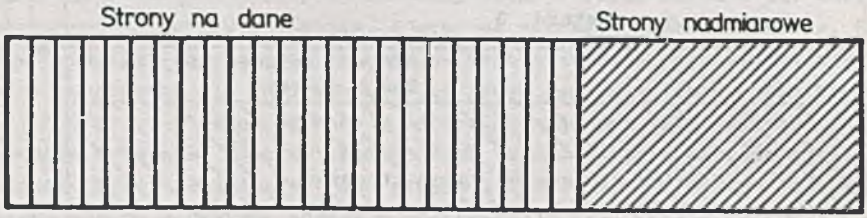
Omówmy przykładowo budowę strony w systemie DMS. Strona na dane posiada w tym systemie dziesięcio-słowowy nagłówek górny oraz nagłówek dolny o zmiennej długości (patrz rys. 2.1). Głównymi informacjami w nagłówku górnym są: kod obszaru, typ strony, numer strony, jej wielkość, całkowitą liczbę wolnych słów na stronie, wskaźnik (pointer) do słowa, od którego zaczyna się ciągły wolny blok na końcu strony oraz liczbę zapamiętanych na stronie rekordów. Nagłówek górny stron z obszaru na indeksy zawiera



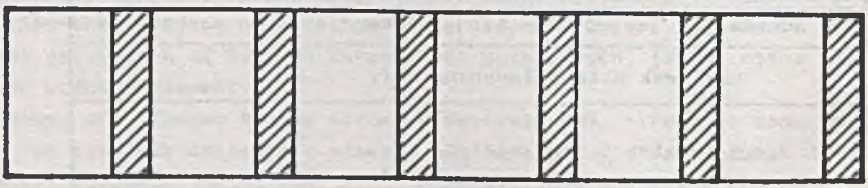
Rys. 2.1. Budowa strony bazy danych

jeszcze dodatkowe informacje. Są nimi liczba kluczy na tej stronie, wielkość klucza (w znakach) i poziom indeksów tej strony. Wskaźnik do wolnego miejsca na końcu strony bardzo ułatwia zapamiętywanie nowych rekordów. System Zarządzania znając wielkość rekordu sprawdza, czy nowy rekord w ogóle zmieści się na wskazanej stronie, a jeżeli tak - próbuje go wstawić na końcu strony w wolnym bloku. Dopiero jeżeli to nie jest możliwe, a mimo to istnieje wystarczająco dużo miejsca na stronie (co mogło być spowodowane usunięciem rekordów z początku strony), automatycznie startuje podprogram pakujący (ang. compact), który przesuwa wszystkie rekordy ku górze i tworzy wolne miejsce na końcu strony.

Typ strony jest związany z koncepcją stron nadmiarowych (ang. overflow). Rozróżniamy strony główne na dane (ang. prime pages) oraz strony nadmiarowe. Na stronach nadmiarowych umieszczane są rekordy w tym przypadku, gdy ich strony główne wybrane zgodnie z typem alokacji rekordu są już zajęte. Projektant ma następujące możliwości użycia stron nadmiarowych - może całkowicie z nich zrezygnować, zdecydować się na strony nadmiarowe globalne na końcu obszaru lub zdefiniować strony nadmiarowe lokalne. Jeżeli w obszarze nie zdefiniowano stron nadmiarowych, System Zarządzania może umieszczać "nadmiarowe" rekordy od początku obszaru lub na najbliższej wolnej stronie. Nadmiarowe strony typu globalnego umieszczane są w ciągłym bloku na końcu obszaru, nadmiar lokalny polega na przeplataniu stron głównych ze stronami nadmiarowymi (p. rys. 2.2).



a)



b)

Rys. 2.2. Strony nadmiarowe
a) globalne - na końcu obszaru, b) lokalne - przeplatane ze stronami na dane

Nagłówek dolny stron z danymi zawiera nagłówki łańcuchów obliczeniowych (ang. calc) oraz adresy kolejnych rekordów na tej stronie (numer słowa, od którego zaczyna się rekord). Łańcuchy obliczeniowe to rekordy pamiętane poprzez procedurę obliczeniową, którym ta procedura przyporządkowała ten sam logiczny adres. Liczba łańcuchów obliczeniowych na każdej stronie obszaru jest taka sama. Nagłówek takiego łańcucha to wskaźnik do pierwszego rekordu o tym adresie logicznym, pozostałe wskaźniki zamykające łańcuch znajdują się w części systemowej każdego rekordu typu obliczeniowego. Nagłówek dolny stron przeznaczony na tablice wskaźników lub indeksy jest nieco inny. Strony obszaru przeznaczonego do przechowywania tablic wskaźników zawierają w nagłówku dolnym tylko adresy kolejnych tablic na tej stronie, zaś strony z obszaru na indeksy nie mają w ogóle nagłówka dolnego.

Z innych parametrów opisujących obszar należy wspomnieć o wskaźniku załadowania początkowego (ang. load factor). Jest to liczba (procent) wskazująca dopuszczalny stopień zapełnienia stron obszaru podczas ładowania początkowego (tzn. odpowiedniego otwarcia obszaru - ang. OPEN FOR INITIAL LOAD).

Reasumując, projektowanie fizycznych wielkości i parametrów obszaru polega na wyborze wielkości strony, liczby łańcuchów obliczeniowych, rodzaju i liczby stron nadmiarowych oraz wskaźnika załadowania początkowego. Liczbę stron obszaru można obliczyć znając przewidywaną liczbę rekordów w bazie danych. Do projektowania bazy na tym etapie należy również podjęcie decyzji o wyborze rodzaju ochrony dla danych obszarów oraz określenie liczby stron, do której obszar może się rozszerzyć. Te właśnie sprawy będą przedmiotem rozdz. 7 niniejszego opracowania.

2.2. Rekord

Rekord jest podstawową jednostką danych w bazie danych. Na poziomie rekordu następuje identyfikacja danych, do niego odwołuje się większość instrukcji języka przetwarzania danych (DML - Data Manipulation Language). Są to takie operacje na rekordzie, jak: "zapamiętaj" (ang. STORE), "znajdź" (ang. FIND, FETCH), "zmodyfikuj" (ang. MODIFY), czy wreszcie "usuń" (ang. DELETE). Rekord może być przechowywany we wskazanym obszarze (obszarach) na dane lub nawet jego części. O fizycznym umieszczeniu rekordu w bazie decyduje jego typ alokacji. Systemy codasylowskie przewidują następujące typy alokacji rekordów:

1) bezpośredni (ang. direct),
2) poprzez procedurę obliczeniową (ang. calc),
3) indeksowo-sekwencyjny,
4) swobodny (ang. via set).

Trzy pierwsze z tych typów można uważać za dostęp bezpośredni do rekordu, gdyż realizowany jest on wprost na bazie znajomości adresu bądź klucza rekordu.

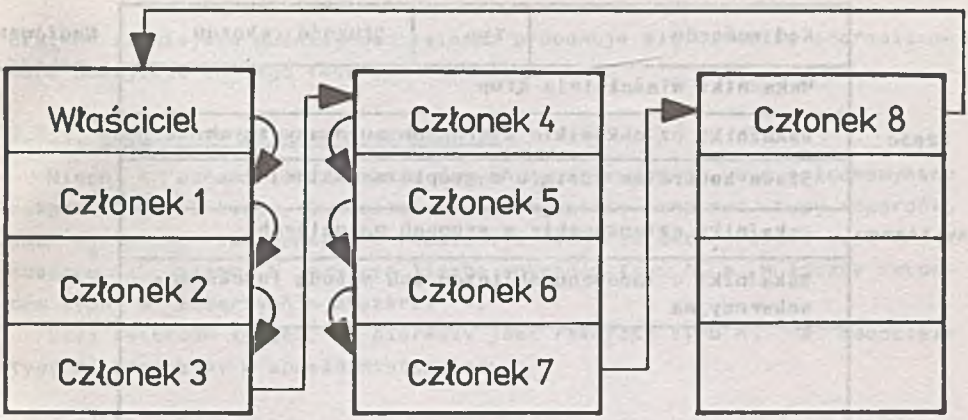
Typ "direct" polega na zapamiętaniu rekordu w ustalonych miejscach (adresach) obszaru. Muszą więc być znane w tym przypadku adresy wszystkich rekordów tego typu w bazie, a ładowanie takich rekordów może następować jedynie do pustego obszaru. Należy zaznaczyć, że często zamiast adresu fizycznego rekordu (typu: nr cylindra, ścieżki, pozycji, czy sektora) w bazach danych stosowana jest adresacja logiczna (np. system DMS - 1100). Adres rekordu w tym przypadku to numer strony obszaru i numer rekordu na stronie, zaś znajomość fizycznego adresu strony nie jest potrzebna dla programu. Wszystkie czynności związane z realizacją operacji WE/WY organizuje i wykonuje System Zarządzania Bazą Danych.

Typ alokacji "calc" opiera się na algorytmie przeliczającym dany klucz na adres w podanym obszarze adresowym. Kluczem może być dowolne pole z rekordu. Procedura przyporządkowuje rekordowi adres logiczny z dokładnością do numeru strony i numeru łańcucha obliczeniowego. Łańcuch obliczeniowy to rekordy-synonimy; rekordy, którym procedura obliczeniowa mimo różnych kluczy przydzieliła ten sam adres.

Metoda indeksowo-sekwencyjnego pamiętania rekordów jest powszechnie znana. W systemach z bazami danych stosuje się jej wersję polegającą na tym, że dane (rekordy) i indeksy są przechowywane w różnych obszarach. W obszarze przeznaczonym na dane na zwykłych stronach są przechowywane rekordy, w obszarze na indeksy przechowywane są maksymalne klucze rekordów z każdej strony z danymi. Dopuszczalne jest stosowanie wielu poziomów indeksowania. Dostęp do rekordów pamiętanych tą metodą jest stosunkowo szybki. Zaleca się stosowanie tej metody szczególnie do pamiętania dużych zbiorów danych.

Ostatnia metoda alokacji rekordów - swobodna polega na dowolnym umieszczaniu rekordów w wolnych miejscach obszaru. Może w tym miejscu obowiązywać (ale nie musi) strategia najbliższego miejsca. Polega ona na tym, że rekordy członkowskie grupy (patrz rys. 2.3) umieszczane są tak blisko rekordu właściciela, jak jest to możliwe, a więc na tej samej lub najbliższych stronach.

Powiązanie rekordów między sobą w strukturę sieciową, co dopuszcza codasylowska koncepcja bazy, wymaga zapamiętania wskaźników (ang. pointer) łączących rekordy w grupy. Jednym z możliwych rozwiązań jest podzielenie rekordu na część z danymi i część systemową, służącą właśnie do tego celu. Dla użytkownika (programu) dostępna jest tylko część rekordu z danymi, pozostałą wykorzystuje System Zarządzania Bazą Danych do realizacji specyficznego dostępu do rekordów. W części systemowej wyróżnić można: nagłówek rekordu, wskaźniki we wszelkiego rodzaju grupach, w których ten rekord jest członkiem lub właścicielem, oraz ewentualnie wskaźnik w łańcuchu obliczeniowym, jeżeli rekord jest pamiętany za pomocą procedury obli-



Rys. 2.3. Strategia najbliższego miejsca

czeniowej. W nagłówku rekordu powinny się znaleźć takie informacje, jak: kod rekordu (unikalny w ramach bazy danych), jego typ oraz długość. Ta ostatnia jest szczególnie ważna przy pamiętaniu rekordów zmiennej długości. Wskaźniki rekordu w grupie czy łańcuchu obliczeniowym to tzw. klucze bazy danych. Są one logicznymi adresami rekordów w bazie i są zbudowane z kodu obszaru, numeru strony i numeru rekordu na stronie. O wielkości klucza i jego poszczególnych części decyduje się na etapie projektowania bazy poprzez odpowiednie zdefiniowanie liczby obszarów w systemie i maksymalnej dopuszczalnej liczby stron w obszarze. Zaleca się takie definiowanie kluczy, aby zbudowane były one jednakowo dla każdego obszaru. Bardzo to ułatwia późniejszy odczyt systemowych wydruków z bazy danych w postaci binarnej (oktalnej).

Rekordy, które po zmodyfikowaniu rozszerzającym ich wielkość nie zmieściły się na pierwotnej stronie, mają jeszcze nagłówek dolny zawierający wskaźnik do głównej strony tego obszaru. Sam rekord w takim przypadku umieszczony jest na stronie nadmiarowej lub najbliższej stronie z danymi. Budowę rekordu w systemie DMS-1100 obrazuje rys. 2.4.

W rekordzie mogą być przechowywane różnego rodzaju dane. Mogą to być dane numeryczne, alfabetyczne czy alfanumeryczne. Mogą być one zapamiętane w różnych dopuszczalnych kodach, tj. ASCII, FIELDATA czy binarnie. Zależy to od klauzuli opisującej kolejne pola rekordu. Opis ten jest bardzo zbliżony do opisu rekordu w języku COBOL, m.in. występuje tu charakterystyczna struktura wielopoziomowa. W rekordzie mogą być przechowywane również pola zmiennej długości. Ściślej mówiąc, są to tablice o zmiennej liczbie powtórzeń. Jedynym ograniczeniem jest to, aby część zmiennej długości w rekordzie następowała po części stałej.

Poprawne zdefiniowanie rekordu i jego pól, czyli właściwa konsolidacja danych, jest jednym z podstawowych warunków dobrego projektu systemu.

	Kod rekordu	Typ	Długość rekordu	Nagłówek rekordu
	Wskaźniki właściciela grup			
Część	Wskaźniki członkowskie w grupach automatycznych			
	Słowo kontrolne udziału w grupie manualnej			
systemowa	Wskaźniki członkowskie w grupach manualnych			
	Wskaźniki w łańcuchu obliczen. lub metoda indeksowo-sekwencyjna			
Część użytkowa	D a n e			
	Adres strony głównej rekordu zmodyfikowanego			Nagłówek dolny

Rys. 2.4. Budowa rekordu bazy danych

O kryteriach tej konsolidacji, różnych możliwych wariantach rozwiązań i ich konkretnej ocenie będzie mowa w rozdziale 4 niniejszego opracowania.

2.3. Grupa logiczna (ang. set.)

Pojęcie grupy logicznej jest bardzo istotne w codasylowskiej koncepcji bazy. Grupa daje możliwość zapamiętania skomplikowanych nieraz związków logicznych zachodzących między rekordami. Grupy mogą się nakładać między sobą czy obejmować te same typy rekordów. Z uwagi na ważność tego pojęcia

oraz na łatwiejsze później odniesienia proponuje się bardziej sformalizowane podejście do tego tematu.

2.3.1. Podstawowe pojęcia i oznaczenia

Niech X oznacza obszar bazy danych, w którym mogą być przechowywane rekordy różnych typów. Symbolami A, B będziemy oznaczać typy rekordów, zaś A_i i B_i ich konkretne realizacje (A_i to i-ty rekord typu A z obszaru X). Niech N oznacza liczbę rekordów typu A, a M liczbę rekordów typu B zawartych w obszarze X.

Ciąg rekordów takich, że pierwszy jest rekordem typu A, a pozostałe typu B, oznaczymy w sposób następujący:

$$S_i = S(A_i) = (A_i, B_{i_1}, B_{i_2}, \ldots, B_{i_j}, \ldots, B_{i_k}). \quad 1 \leqslant i_j \leqslant M; \quad 1 \leqslant i \leqslant N. \tag{2.1}$$

Niech symbol $S(A_{i,j})$ oznacza "j+1" wyraz tego ciągu, tzn.:

$$S(A_i, j) = B_{i_j} \quad \text{dla} \quad 1 \leqslant j \leqslant k$$

$$S(A_i, 0) = A_i \tag{2.2}$$

Niech $S(A,B)$ oznacza dowolny zbiór rozłącznych ciągów S_i. Zbiór ten nazywać będziemy grupą logiczną rekordów typu A i B. Ciągi S_i należące do $S(A,B)$, nazywamy wystąpieniami tej grupy, rekordy typu A właścicielami grupy, a rekordy typu B członkami grupy (patrz (2.2)). Zwróćmy uwagę na fakt, że nie określamy precyzyjnie charakteru związku logicznego zachodzącego między rekordami A i B, poprzestając jedynie na fizycznym obrazie tego związku. Zauważmy również, że te same typy rekordów mogą być związane różnymi zależnościami, można więc definiować w tym przypadku różne grupy logiczne oznaczane kolejno $S_1(A,B)$, $S_2(A,B)$ itd. (przykład 2).

W dalszej części pracy, tam gdzie nie będzie wzbudzało to wątpliwości, wyrażenie "grupa" będzie używane dwuznacznie; raz do nazwania typu związku logicznego między rekordami A i B, innym razem do nazwania konkretnego ciągu rekordów $S(A_i)$. Liczbę k ze wzoru (2.1) nazywamy licznością realizacji grupy $S(A_i)$. Wprowadźmy jeszcze wykorzystywane dalej odwzorowanie:

$$S^{-1}(B_j) = A_i \quad \text{dla} \quad B_j \in S(A_i) \tag{2.3}$$

Odwzorowanie to przyporządkowuje dowolnemu rekordowi członkowskiemu jego rekord właściciela w danej grupie.

Wystąpienie grupy logicznej identyfikowane jest poprzez rekord właściciela. W szczególności wystąpienie grupy może składać się tylko z rekordu właś-

ciciela, tzn. $S_i = (A_i)$. Grupę taką nazywamy grupą pusta. Innymi słowy, każdy rekord zdefiniowany jako właściciel zawsze umieszczony jest w strukturze grupy.

Zasadniczo inna sytuacja zachodzi w przypadku rekordów członkowskich. Te mogą być podporządkowane pewnym rekordom typu właściciel lub nie. Jeżeli każdy rekord członkowski znajduje się w jakimś wystąpieniu grupy, grupę taką nazywamy automatyczną. Oznacza to, że w momencie wprowadzania rekordu do bazy należy podpiąć go do właściwej grupy. Formalnie można to zapisać w postaci warunku:

$$\bigwedge_{1 \leq j \leq M} \bigvee_{1 \leq i \leq N} B_j \in S(A_i) \qquad (2.4)$$

Jeżeli w grupie mogą istnieć rekordy członkowskie, które nie są przydzielone do żadnego właściciela, grupę taką nazywamy manualna. Oczywiście w przypadku tych grup warunek (2.4) nie zawsze jest spełniony. Podpinanie rekordów do grupy lub usuwanie ich z grupy wykonuje się poprzez odpowiednie instrukcje języka manipulacji danych (INSERT, REMOVE). Są to operacje wykonywane niejako "ręcznie" na bazie danych, stąd nazwa grupy w odróżnieniu od grup automatycznych. W momencie zapamiętywania rekordu członkowskiego w bazie jego podporządkowanie logiczne w grupie nie musi być jeszcze znane.

Wprowadźmy dodatkowo dwa wygodne pojęcia: klucz bazy danych i rekord bieżący.

- Klucz bazy danych rekordu to logiczny adres tego rekordu w bazie danych (obszar, numer bloku, numer rekordu w bloku).
- Rekord B_j nazywamy rekordem bieżącym dla programu P, jeżeli jest on ostatnim przeczytanym lub zapamiętanym rekordem przez odpowiednie instrukcje programu P.

Stan "bieżący w programie" ma sens oczywiście tylko do końca pracy tego programu. Definicję tę można poszerzyć na "bieżący w obszarze", "bieżący w grupie logicznej" czy "bieżący w typie rekordu". I tak kolejno:

- Rekord B_j nazywamy rekordem bieżącym w obszarze X, jeżeli jest to ostatnio przeczytany lub zapamiętany rekord tego obszaru przez którykolwiek z aktywnych obecnie programów.
- Rekord B_j nazywamy rekordem bieżącym dla typu rekordu B, jeżeli jest to ostatnio przeczytany lub zapamiętany rekord tego typu przez którykolwiek z aktywnych obecnie programów.
- Rekord nazwiemy rekordem bieżącym dla grupy S, jeżeli jest to ostatnio przeczytany lub zapamiętany rekord tej grupy przez którykolwiek z aktywnych obecnie programów. Rekordem bieżącym w grupie S(A,B) może być dowolny rekord właściciela A_i lub którykolwiek z jego rekordów członkowskich.

2.3.2. Poszerzenie grupy o nowe rekordy członkowskie

Niech $S(A_i) = (A_i, B_{i_1}, B_{i_2}, \ldots, B_{i_k})$ będzie tym wystąpieniem grupy, do którego chcemy dołączyć nowy rekord B_{i_0}. Dołączenie to powinno się odbywać w zadanym z góry porządku dla danego typu grupy.

Oczywiście nowy rekord zostanie którymś z kolei rekordem członkowskim grupy, a więc:

$$B_{i_0} = S(A_i, l), \quad 1 \leq l \leq k + 1. \tag{2.5}$$

Zdefiniujmy logiczny punkt wstawienia do grupy następująco:

- jeżeli $l = 1$, to mówimy, że w grupie określono logiczny punkt wstawiania jako "pierwszy";
- jeżeli $l = k + 1$, to mówimy, że w grupie określono logiczny punkt wstawiania jako "ostatni".

Załóżmy dodatkowo, że rekord $S(A_i, j)$ jest rekordem bieżącym grupy (w szczególności może to być rekord właściciela):

- jeżeli $l = j + 1$, to mówimy, że w grupie określono logiczny punkt wstawienia jako "następny";
- jeżeli $l = j - 1$, to mówimy, że w grupie określono logiczny punkt wstawienia jako "poprzedni".

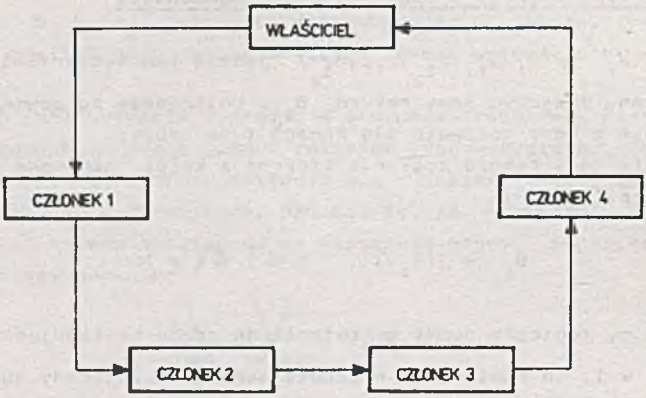
Logiczny punkt wstawienia w grupie ilustruje rys. 2.7. Ponadto wartość l może być ustalona automatycznie w zależności od wartości określonego klucza z pola danych rekordu lub w zależności od klucza bazy danych tego rekordu. Logiczny punkt wstawienia w tego typu grupach nazywamy umownie "sortowany".

2.3.3. Komputerowa realizacja grupy

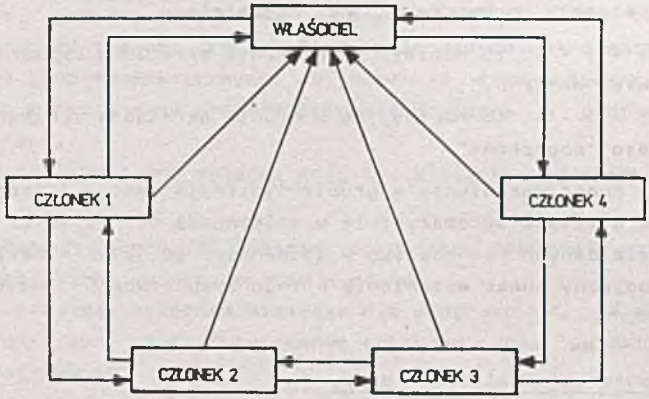
Zastanówmy się nad komputerową realizacją pojęcia grupy logicznej. W zasadzie istnieją dwie możliwości: utworzenie łańcucha rekordów powiązanych ze sobą wskaźnikami lub istnienie tablic adresów wszystkich rekordów członkowskich każdego wystąpienia grupy.

Grupy takie nazwiemy odpowiednio: "cykliczną" lub "z tablicą wskaźników". Przykładowe wystąpienia tych grup przedstawiają odpowiednio rys. 2.5 i rys. 2.8. W grupach cyklicznych oprócz wskaźników "w przód" można zdefiniować dodatkowe wskaźniki "wstecz" i "do właściciela" (rys. 2.6). Zauważmy, że wystąpienie grupy cyklicznej to lista cykliczna (ewentualnie podwójnie połączona) z zaznaczonym pierwszym rekordem tej listy (rekord właściciela).

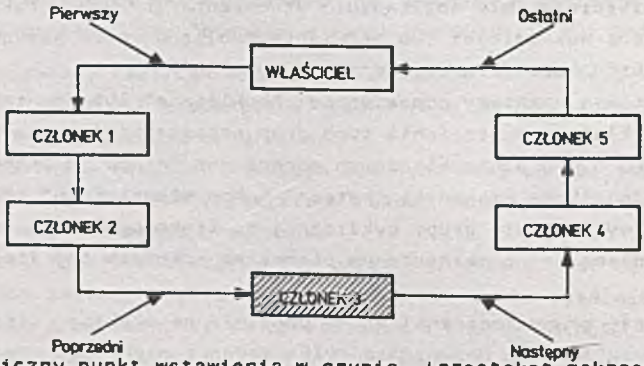
Definicję grupy logicznej można uogólnić na większą liczbę typów rekordów (przykład 4). Oczywiście tylko jeden z nich może być zdefiniowany



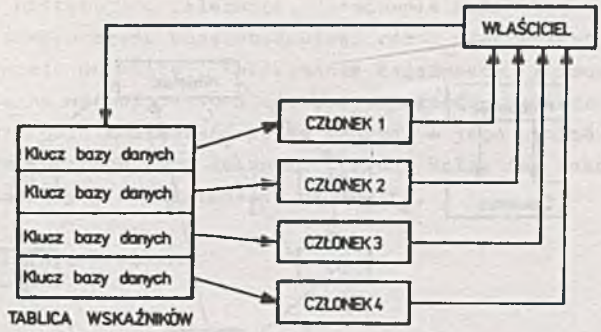
Rys. 2.5. Grupa cykliczna



Rys. 2.6. Grupa cykliczna z dodatkowymi wskaźnikami



Rys. 2.7. Logiczny punkt wstawienia w grupie (prostokąt zakreskowany symbolizuje rekord "bieżący")



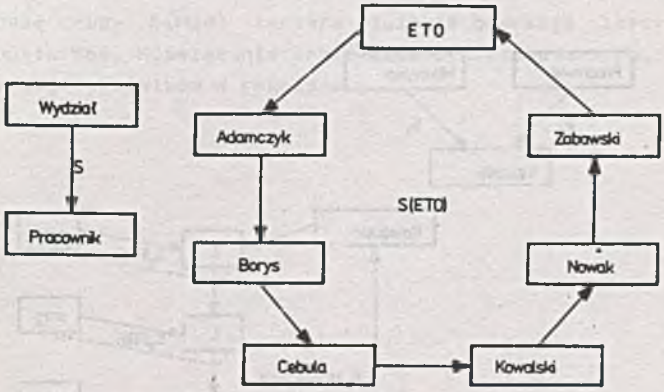
Rys. 2.8. Grupa z tablicą wskaźników

jako właściciel grupy, pozostałe typy mogą występować jedynie jako członkowie.

2.3.4. Przykłady grup logicznych

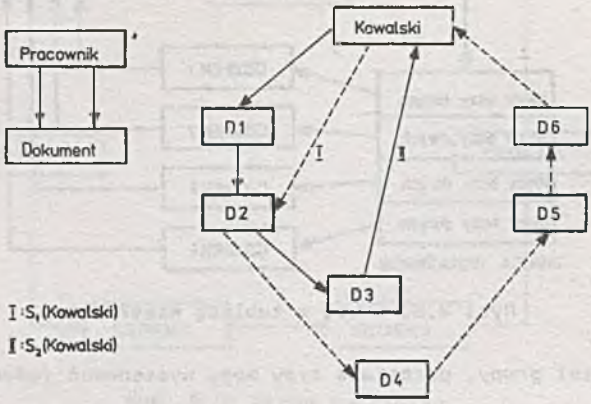
1. "Wydział - Pracownik"

Jest to jeden z najprostszych i naturalnych przykładów grupy. Wykorzystano tu oczywistą współzależność danych opisujących wydział i pracownika. Każdy pracownik jest przyporządkowany do jakiegoś wydziału. Jest to więc grupa automatyczna. Porządek w grupie (logiczny punkt wstawienia) zdefiniowano jako "sortowany". Symbol S(ETO) przedstawia wystąpienie grupy zdefiniowane właścicielem - działem ETO.



Rys. 2.9. Przykład grupy - "Wydział - Pracownik"

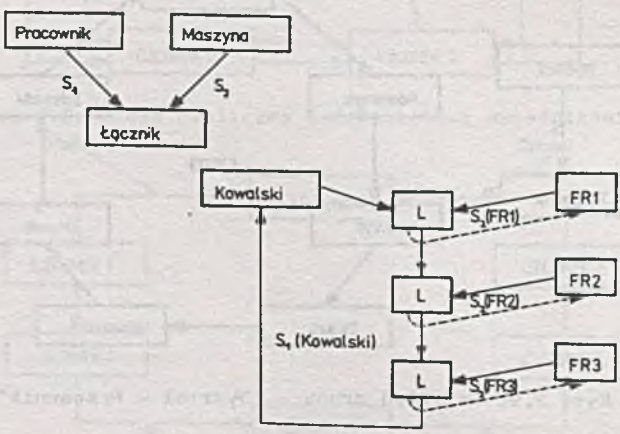
2. "Pracownik - Dokument"



Rys. 2.10. Przykład grup - "Pracownik - Dokument"

Dla tych samych typów rekordów można zdefiniować różne grupy logiczne. Niech S_1 oznacza grupę zdefiniowaną następująco: "pracownik jest jedynym autorem pewnych dokumentów", a S_2: "pracownik wypożyczył aktualnie pewne dokumenty". Załóżmy dodatkowo, że biblioteka dysponuje tylko jednym egzemplarzem każdego dokumentu. Wtedy S_1 (Kowalski) oznacza zbiór dokumentów, których autorem jest Kowalski, a S_2 (Kowalski) zbiór dokumentów, które aktualnie są u Kowalskiego.

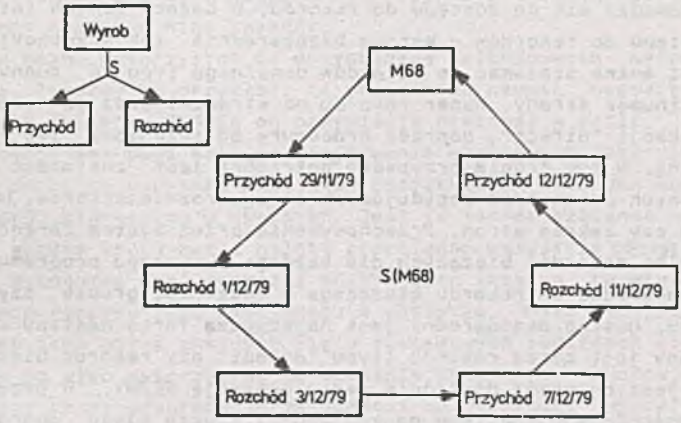
3. "Pracownik - Maszyna" i "Maszyna - Pracownik"



Rys. 2.11. Przykład grup - "Pracownik - Maszyna" i "Maszyna-Pracownik"

Zdefiniujmy następującą zależność: "pracownik może obsługiwać różne typy maszyn; tę samą maszynę mogą obsługiwać różni pracownicy". Jest to zależność typu "wielu do wielu". Rozwiązanie zależności za pomocą grup logicznych polega na wprowadzeniu dodatkowego rekordu zwanego łącznikiem. Jeżeli pracownik może obsługiwać kilka maszyn, w jego grupie S_1 będzie odpowiednia liczba rekordów typu łącznik, które z kolei są rekordami członkowskimi we właściwych wystąpieniach grupy S_2.

4. "Wyrób - przychód - rozchód"



Rys. 2.12. Przykład grupy - "Wyrób - Przychód - Rozchód"

Wystąpienie grupy S(M68) zawiera historię operacji (przychody, rozchody) produktu M68. Rozwiązanie takie może być stosowane np. celem ograniczenia liczby wskaźników w rekordach.

3. FIZYCZNE ASPEKTY ORGANIZACJI DANYCH W BAZIE

3.1. Dostęp do danych

Podstawową jednostką danych w bazie jest rekord. Dlatego też dostęp do danych sprowadza się do dostępu do rekordu. W bazach danych istnieją dwie metody dostępu do rekordów - metoda bezpośrednia lub sekwencyjna. Dostęp bezpośredni można stosować do rekordów dowolnego typu o znanym adresie logicznym (numer strony, numer rekordu na stronie) oraz do rekordów o metodzie alokacji "direct", poprzez procedurę obliczeniową lub indeksowo-sekwencyjną. W tym drugim przypadku potrzebna jest znajomość klucza rekordu i innych parametrów decydujących o jego rozmieszczeniu, jak np. nazwa obszaru czy zakres stron. Przechowywanie przez System Zarządzania Bazą Danych tablic rekordów bieżących dla każdego aktywnego programu umożliwia dostęp bezpośredni do rekordu bieżącego w obszarze, grupie czy danym typie rekordu. Dostęp bezpośredni jest najszybszą formą dostępu do rekordu. Jeżeli znany jest adres rekordu (typu "direct" czy rekordu bieżącego), realizowany jest on przez dokładnie jedną operację WE/WY. W przypadku dostępu bezpośredniego opartego na znajomości klucza ilość operacji WE/WY może być większa. W zapełnionej bazie danych należy bowiem liczyć się z tym, że rekord nie znajduje się na swojej stronie głównej, lecz na stronie nadmiarowej. Średnio można przyjąć, że w tym przypadku wykonywanych jest 1-2 dostępów fizycznych. Z kolei dostęp do rekordu pamiętanego metodą indeksowo-sekwencyjną wymaga, w zależności od liczby poziomów indeksowania, przeczytania najpierw kilku stron indeksowych. W metodzie tej powinno ograniczać się liczbę poziomów indeksowania (maksymalnie 3) poprzez odpowiedni wybór wielkości strony na indeksy. Stąd można przyjąć, że w tej metodzie należy wykonywać 2-3 fizyczne dostępy, aby dotrzeć w końcu do strony zawierającej szukany rekord.

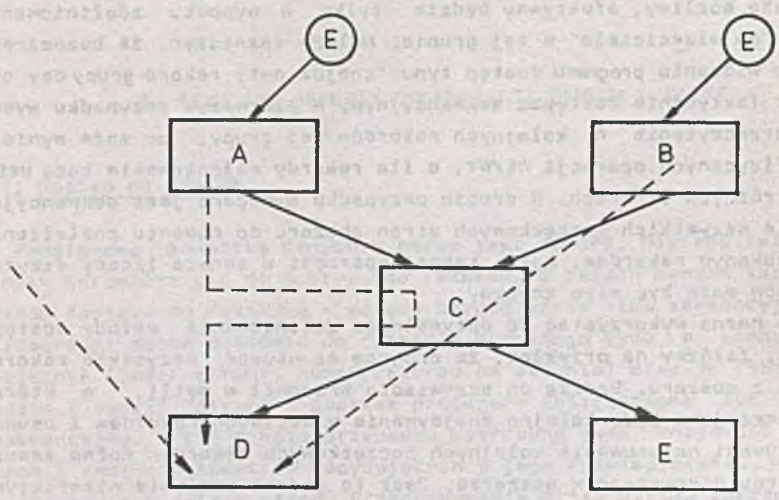
Sekwencyjny dostęp może być stosowany do przeglądania obszarów i grup. W obu przypadkach możliwy jest dostęp do pierwszego, ostatniego i "n"-tego rekordu. Można również przeczytać rekord następny lub poprzedni w stosunku do rekordu bieżącego. Przeglądanie może dotyczyć wszystkich typów rekordów (w obszarze lub grupie) lub tylko typu wybranego. Możliwe jest również szukanie duplikatów ze względu na klucz (w metodzie obliczeniowej lub w grupach posortowanych). W grupie możliwy jest również dostęp od rekordu członkowskiego do jego rekordu właściciela. Efektywność dostępu w grupie zależy od rodzaju zdefiniowanych w niej wskaźników. Na przykład, dostęp do rekordu poprzedniego może spowodować konieczność sekwencyjnego

przeczytania całej struktury łańcuchowej, o ile nie będzie w grupie zdefiniowany wskaźnik "wstecz". Podobnie dostęp do rekordu właściciela, co prawda zawsze możliwy, efektywny będzie tylko w wypadku zdefiniowania wskaźników "do właściciela" w tej grupie. Należy zaznaczyć, że bezpośredni z punktu widzenia programu dostęp typu "znajdź n-ty rekord grupy czy obszaru" jest faktycznie dostępem sekwencyjnym. W pierwszym przypadku wymaga bowiem przeczytania n kolejnych rekordów tej grupy, co może wynieść nawet n fizycznych operacji WE/WY, o ile rekordy członkowskie będą umieszczone na różnych stronach. W drugim przypadku wymagane jest sekwencyjne przeglądanie wszystkich początkowych stron obszaru do momentu znalezienia strony z szukanym rekordem. Koszt takiej operacji w sensie liczby fizycznych dostępów może być więc znaczny.

Uwagi te można wykorzystać do optymalnego definiowania metody dostępu w programie. Załóżmy na przykład, że program ma usuwać wszystkie rekordy danego typu z obszaru. Będzie on oczywiście pracował w pętli, w której głównymi operacjami będą kolejno znajdywanie właściwych rekordów i usuwanie ich. Z uwagi na usuwanie kolejnych początkowych rekordów można zawsze szukać rekordu pierwszego w obszarze. Jest to jednak szalenie nieefektywne, gdyż w każdym kroku pętli należy przeglądać wszystkie początkowe strony obszaru wykonując masę operacji WE/WY. Dużo lepszym rozwiązaniem jest szukanie tego rekordu jako następnego w obszarze. Mimo usunięcia rekordu poprzedniego jego adres znajduje się w systemowych tablicach stanu bieżącego, można go więc wykorzystać do szukania kolejnego rekordu. Skróci to znacznie czas pracy programu (w zależności od wielkości obszaru czasy te mogą się różnić nawet kilkusetkrotnie). Ten przykład wskazuje, jak istotne jest prawidłowe zdefiniowanie struktury danych dopasowanej do programu i właściwe jej wykorzystanie przez program.

Udział rekordu w strukturze sieciowej realizowanej przez różne grupy pozwala na różnoraki dostęp do tego samego rekordu. W związku z tym używa się pojęcia: ścieżka dostępu (ang. path). Jest to łańcuch typów rekordów zaczynający się w górze hierarchii w strukturze sieciowej, a kończący się danym typem rekordu (patrz rys. 3.1).

Innymi słowy, jest to wybranie ze struktury sieciowej pewnej struktury hierarchicznej (drzewa). Ma to zastosowanie między innymi przy definiowaniu podschematu wykorzystującego jedynie jedną z wielu możliwości pełnego schematu bazy danych. Wśród dopuszczalnych ścieżek dostępu można niekiedy wyróżnić ścieżkę główną oraz ścieżki alternatywne. Pojęcie ścieżki głównej ma bowiem związek z metodą alokacji rekordów poprzez wskazaną grupę (ang. via set). Taka grupa, o ile jest ich więcej, jest właśnie wybierana do ścieżki głównej dostępu. To samo dotyczy również rekordów umieszczonych wyżej w hierarchii, gdy rekord właściciela w jednej grupie jest rekordem członkowskim w drugiej. Pojęcie ścieżki dostępu jest bardzo mocno związane z wyborem właściwego wystąpienia grupy. Wystąpienie grupy jest określone poprzez właściwe wystąpienie rekordu właściciela. Wystąpienie



Rys. 3.1. Ścieżki dostępu do rekordu

grupy to przecież nic innego jak rekord właściciela i jego rekordy członkowskie; wybór właściciela wskazuje jednoznacznie, o którą grupę chodzi. Istnieje jeszcze druga metoda określania wystąpienia grupy. Wykorzystuje ona pojęcie stanu bieżącego w grupie; właściwe wystąpienie może być wskazane rekordem bieżącym, to jest takim rekordem, który ostatnio został odczytany lub zapamiętany przez program. Oczywiście rekordem tym może być zarówno rekord właściciela, jak i dowolny jego rekord członkowski. Ta ostatnia cecha może być wykorzystana do sekwencyjnego przeglądania grup. Uniknięcie konieczności ustalania rekordu właściciela przy każdym rekordzie członkowskim znacznie skraca czas tej operacji. Tak więc do sekwencyjnego przetwarzania rekordów grupy nadaje się lepiej metoda wyboru grupy poprzez określenie stanu bieżącego, zaś metoda wyboru przez rekord właściciela skuteczniejsza jest do organizacji bezpośredniego dostępu do rekordu członkowskiego w grupie. Obie te metody mogą być wykorzystane w definiowaniu różnych schematów do tej samej bazy danych, co ma duże znaczenie np. w ładowaniu początkowym bazy danych (patrz rozdział 9).

3.2. Wstawianie rekordów do bazy danych

Zapamiętywanie rekordów w bazie danych jest jedną z podstawowych operacji w każdej bazie. Nawet w dużych statycznych bazach danych, gdzie ak-

tualizacje danych następują rzadko, wymagane jest początkowe załadowanie tych baz rekordami używanymi później. Patrząc od strony fizycznej proces zapamiętania rekordu w bazie można podzielić na następujące fazy:

1) Wybór adresu rekordu.
2) W wypadku braku miejsca na wybranej stronie wybór strony nadmiarowej lub innej strony na dane.
3) Wygenerowanie części systemowej rekordu.
4) Fizyczna aktualizacja strony.

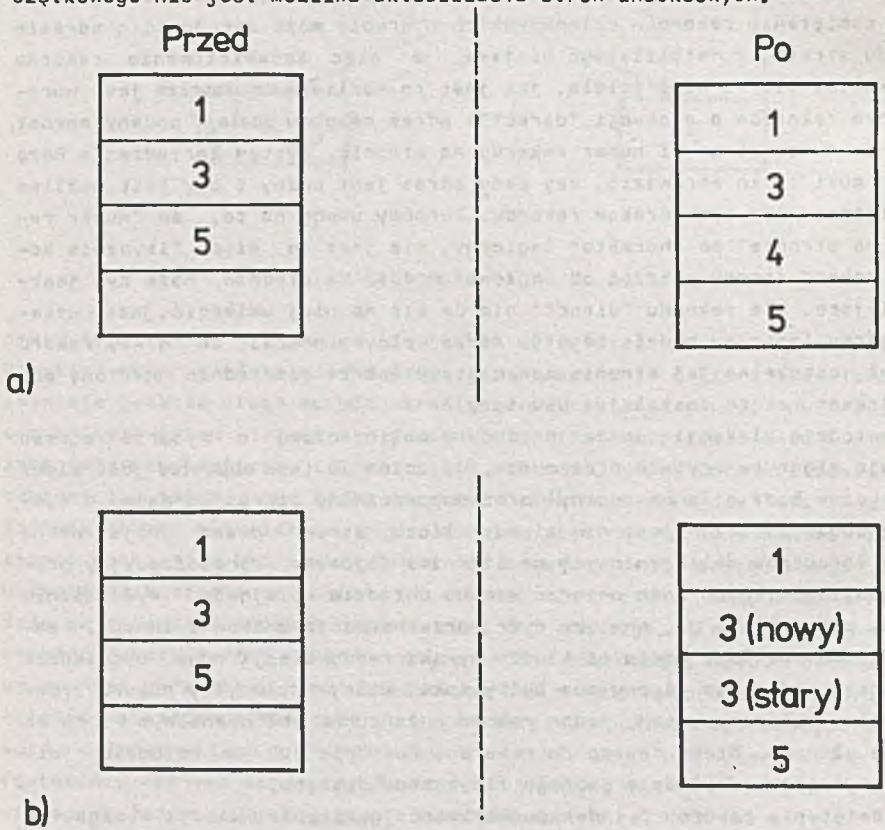
Omówmy kolejno te fazy.

Wybór adresu zależy w sposób istotny od metody alokacji rekordu oraz innych pomocniczych parametrów, takich jak stopień załadowania początkowego czy zadeklarowany interwał między rekordami tego samego typu. W przypadku pamiętania rekordów członkowskich w grupie może decydować o adresie rekordu strategia najbliższego miejsca, a więc zapamiętywanie rekordu członka tak blisko właściciela, jak jest to możliwe. Najprostsze jest wprowadzanie rekordów o alokacji "direct"; adres rekordu zostaje podany wprost. Jest nim numer strony i numer rekordu na stronie. System Zarządzania Bazą Danych musi tylko sprawdzić, czy dany adres jest wolny i czy jest możliwe zapamiętanie pod tym adresem rekordu. Zwróćmy uwagę na to, że "numer rekordu na stronie" ma charakter logiczny, nie jest to więc fizycznie kolejny rekord strony patrząc od nagłówka w dół. Na stronie może być jeszcze miejsce, ale rekordu "direct" nie da się na niej umieścić, jeżeli wskazany adres logiczny będzie zajęty. Adres wolny oznacza, że "n"-ty rekord nie był jeszcze na tej stronie zapamiętany lub że poprzednio wstawiony pod tym adresem rekord został już usunięty.

W metodzie alokacji poprzez procedurę obliczeniową o wyborze adresu decyduje algorytm użyty w procedurze. Wejściem do tego algorytmu jest klucz (numeryczny bądź alfanumeryczny) oraz dopuszczalny zakres adresów. W systemach codasylowskich jest nim iloczyn liczby stron przez projektowaną liczbę łańcuchów obliczeniowych na stronie. Algorytm randomizacyjny procedur obliczeniowych może opierać się na metodzie kolejnego wydzielania klucza, potęgowania go, analizy cyfr, przetwarzania znaków i innych. W zależności od rozkładu wartości kluczy wyniki randomizacji mogą być lepsze lub gorsze. Idealnym algorytmem byłby taki, który rozłożyłby rekordy równomiernie w obszarze, tzn. jeden rekord w łańcuchu obliczeniowym i to na stronie głównej. Wtedy dostęp do rekordu, podobnie jak w metodzie "direct", wymagałby dokładnie jednego fizycznego dostępu.

Zapamiętanie rekordu o indeksowo-sekwencyjnym typie alokacji wymaga wyraźnego rozróżnienia ładowania początkowego od późniejszego dokładania rekordów do bazy. Ładowanie początkowe tego typu rekordów jest procesem złożonym. Wymaga przede wszystkim wstępnego przygotowania danych (rekordów) poprzez odpowiednie ich posortowanie. W miarę zapełniania stron z danymi następuje tworzenie odpowiadających im stron indeksowych zawierających in-

formacje o maksymalnym kluczu rekordu na każdej stronie. Bardzo istotne na tym etapie jest zostawienie wolnego miejsca na stronie, a nie zapisywanie jej całkowicie. Można to zrobić poprzez zadeklarowanie odpowiedniego współczynnika załadowania początkowego strony. Ułatwia to znacznie proces późniejszego dokładania rekordów do obszaru już zapełnionego. Proces ten przebiega następująco. O ile na stronie głównej jest wolne miejsce i klucz nowego rekordu jest mniejszy od maksymalnego klucza tej strony, zapamiętanie takiego rekordu wymaga jedynie reorganizacji strony z danymi (uporządkowanie rekordów) (patrz rys. 3.2). Jeżeli na stronie głównej nie ma miejsca, gdyż jest ona zapełniona rekordami indeksowo-sekwencyjnymi lub innego typu, rekord zostanie wstawiony na stronę nadmiarową zgodnie z zasadami podanymi poniżej. Należy zaznaczyć, że poza okresem ładowania początkowego nie jest możliwa aktualizacja stron indeksowych.



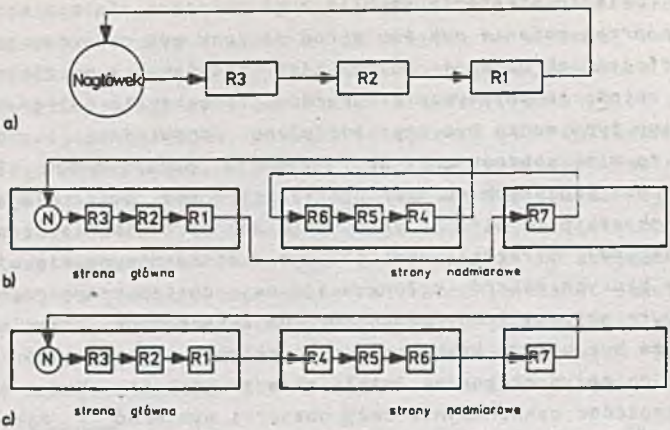
Rys. 3.2. Umieszczanie rekordu indeksowo-sekwencyjnego na stronie głównej a) zapamiętanie rekordu o kluczu nowym na stronie, b) zapamiętanie duplikatu

Metoda swobodnego zapamiętywania rekordów członkowskich w grupie polega na przydziale pierwszego wolnego adresu ze wskazanego obszaru. Może w

tym miejscu działać strategia najbliższego miejsca, ale też może być ona wstrzymana poprzez zadanie zakresu stron na dany typ rekordu, zapełnienie stron do wartości określonej wskaźnikiem załadowania początkowego oraz interwałową metodę zapamiętywania rekordów. Ta ostatnia polega na tym, że rekordy danego typu muszą być poprzedzielane odpowiednią liczbą pustych stron. Może to mieć zastosowanie do ładowania początkowego właścicieli grup, aby na następujących po nich pustych stronach było miejsce na ich rekordy członkowskie (a więc na stosowanie dla nich właśnie strategii najbliższego miejsca). Strategia najbliższego miejsca nie obowiązuje również w grupach, w których rekordy członkowskie mają dostęp bezpośredni, np. poprzez procedurę obliczeniową. Zauważamy, że sekwencyjne przeglądanie takich grup może być bardzo kosztowne, gdyż rekordy członkowskie mogą być porozrzucane po całym obszarze. Jeżeli ponadto jest ich dużo, opłaca się bardziej przeglądać sekwencyjnie cały obszar i wybierać z niego rekordy wskazanego typu. Może to wielokrotnie skrócić czas tej operacji. Grupy z rekordami członkowskimi o dostępie bezpośrednim mogą mieć inne zastosowanie (patrz usuwanie rekordów z grupy, rozdz. 5).

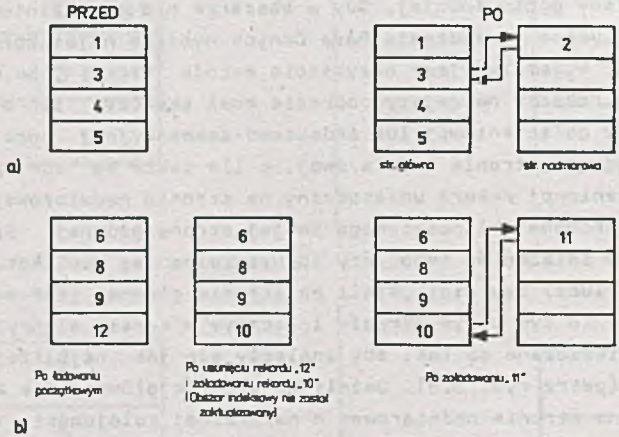
Jeżeli na wskazanej lub obliczonej stronie brak miejsca, rekord musi zostać zapamiętany gdzie indziej. Gdy w obszarze nie są zdefiniowane strony nadmiarowe, System Zarządzania Bazą Danych wybiera najbliższą niepełną stronę z danymi. Wyjątkiem jest oczywiście metoda "direct", bo wtedy próba zapamiętania rekordu na zajętym adresie musi skończyć się błędnie. W przypadku metody obliczeniowej lub indeksowo-sekwencyjnej może nastąpić zapisanie rekordu na stronie nadmiarowej, o ile takie są zdefiniowane. W metodzie obliczeniowej rekord umieszczany na stronie nadmiarowej musi być podłączony do łańcucha obliczeniowego swojej strony głównej. Sposób tego podłączenia może zależeć od tego, czy dopuszczalne są duplikaty rekordów (ze względu na klucz) czy nie. Jeżeli na stronie głównej jest miejsce, to rekordy-synonimy (o tym samym adresie logicznym w sensie algorytmu randomizacyjnego) umieszczane są tak, aby znalazły się jak najbliżej nagłówka obliczeniowego (patrz rys. 3.3). Jeżeli na stronie głównej nie ma miejsca, umieszczane są na stronie nadmiarowej w naturalnej kolejności ich pojawiania się. Kolejność łączenia rekordów w tym samym łańcuchu zapamiętywanych na następnych stronach nadmiarowych zależy od sposobu traktowania duplikatów. Jeżeli są one dozwolone następuje reorganizacja łańcucha tak, aby ze strony głównej od razu można było uzyskać ostatnią stronę nadmiarową. Może to przyspieszyć proces zapamiętania nowego rekordu, gdyż istnieje większa szansa znalezienia wolnego miejsca już na pierwszej stronie nadmiarowej. Jeżeli duplikaty nie są dozwolone, kolejność rekordów w łańcuchu nie ulega zmianie, gdyż i tak System Zarządzania Bazą Danych musi przejrzeć cały łańcuch, aby wychwycić ewentualne duplikaty.

Wstawianie rekordów indeksowo-sekwencyjnych do już zapełnionego obszaru może być zadaniem trudnym. Z uwagi na ograniczenie, że stron indeksowych nie można aktualizować, należy inaczej poinformować System o "niena-



Rys. 3.3. Wstawianie rekordów do łańcuchów obliczeniowych

a) łańcuch obliczeniowy na stronie głównej, b) łańcuch obliczeniowy przebiegający stroną główną i strony nadmiarowe (duplikaty dozwolone), c) jak w "b", z tym że duplikaty nie są dozwolone



Rys. 3.4. Umieszczanie rekordów indeksowo-sekwencyjnych na stronach nadmiarowych

a) wstawienie rekordu o kluczu "2" - połączenia rekordów "3" i "2" wskaźnikami, rekord "3" zaznaczony jako "duplikat", b) wstawienie rekordu o kluczu wyższym niż ostatni na stronie, połączenie wskaźnikami rekordów "10" i "11", rekord "10" zaznaczony jako "nie ostatni"

turalnym" adresie rekordu. Można to zrobić wykorzystując część systemową rekordu (nagłówek główny i dolny).

Rozpatrzmy kolejne przykłady zgodnie z rys. 3.4. Jeżeli na stronie głównej jest miejsce, to nowy rekord zostanie na niej zapisany, a strona odpowiednio zreorganizowana. Jeżeli na stronie głównej nie ma miejsca, a nowy rekord ma klucz niższy niż największy na stronie, to po umieszczeniu go na stronie nadmiarowej nastąpi również aktualizacja rekordu o kluczu

bezpośrednio wyższym. Rekord ten zostanie zaznaczony jako duplikat (w sensie metody indeksowo-sekwencyjnej). Ponadto oba rekordy zostaną połączone wskaźnikami. Są to adresy obu rekordów umieszczone na końcu rekordu jako tzw. nagłówek dolny. Weźmy następny przykład. Załóżmy, że strona z danymi została całkowicie załadowana, a następnie usunięto z niej rekord o najwyższym kluczu. Z kolei w to miejsce mógł zostać zapamiętany rekord o kluczu największym na stronie, ale mniejszym od maksymalnego zapisanego w obszarze indeksowym. Jeżeli teraz zajdzie potrzeba zapamiętania rekordu o kluczu większym niż ostatni rekord na stronie, ale mniejszym czy równym od klucza z obszaru indeksów, należy zaktualizować ostatni rekord na stronie. Musi on wskazywać rekord ze strony nadmiarowej o większym niż on kluczu, inaczej mówiąc, musi odzwierciedlać stan "nie ostatni". Oba te rekordy, ostatni ze strony głównej i rekord ze strony nadmiarowej zostaną połączone wskaźnikami będącymi ich adresami w bazie.

Kolejną fazą, którą należy omówić, jest wygenerowanie części systemowej rekordu. Najłatwiejsza sprawa jest z nagłówkiem rekordu, gdyż znane są jego kod, typ i wielkość. Jeżeli wstawiany rekord jest właścicielem w różnych grupach, należy odpowiednio zainicjować jego wskaźniki do rekordów członkowskich. Ponieważ rekordów członkowskich jeszcze nie ma, wskaźnikami tymi będzie adres samego siebie. Takie załadowanie wskaźników oznacza, że grupa jest pusta. Jeżeli wstawiany rekord jest rekordem członkowskim w jednej czy kilku grupach, należy wybrać ich właściwe wystąpienia i następnie je zreorganizować. Koszt tej operacji zależy oczywiście od rodzaju wskaźników używanych w grupie (patrz rozdz. 5). W rezultacie, oprócz zainicjowania właściwych wskaźników we wstawianym rekordzie (w przód, ewentualnie wstecz i do właściciela), musi nastąpić również aktualizacja tych wskaźników w rekordzie poprzednim (ewentualnie następnym i w rekordzie właścicielu). Nieco inaczej przebiega wstawianie rekordu do grupy z tablicami wskaźników. Z uwagi na to, że rekordy członkowskie nie są bezpośrednio połączone, wymagana jest jedynie aktualizacja właściwej tablicy wskaźników. W rekordzie członkowskim inicjowane są jedynie wskaźniki do właściciela. W tym miejscu należy jeszcze wspomnieć o różnicy między grupami automatycznymi a manualnymi. W pierwszym przypadku rekord członkowski już w momencie wstawiania go do bazy jest podpinany do właściwej grupy. Jeżeli grupa (ściślej mówiąc - rekord w grupie) zdefiniowana jest jako manualna, rekord członkowski może zostać zapisany w bazie bez powiązania go z jakimkolwiek innym rekordem tej grupy. Może to nastąpić później, gdyż takie podpięcie można zrealizować programowo poprzez odpowiednią instrukcję DML (INSERT).

Ostatnią fazą jest fizyczne umieszczanie nowego rekordu na stronie. Podstawowym zadaniem Systemu Zarządzania Bazą Danych jest sprawdzenie, czy na wybranej stronie jest jeszcze wolne miejsce. O ile tak, System próbuje umieścić ten rekord na końcu strony. W razie konieczności startowana jest procedura pakująca stronę, a więc tworząca ciągły wolny obszar na końcu

strony. Wtedy też rekord umieszczany jest fizycznie na końcu strony, a jego adresem logicznym jest pierwsze wolne miejsce w tablicy adresowej w nagłówku dolnym strony. Jednocześnie następuje aktualizacja licznika słów wolnych na stronie oraz wskaźnika do pierwszego słowa wolnego bloku na tej stronie. Informacja o tym, gdzie na stronie rozpoczyna się nowy rekord, umieszczana jest w tablicy adresowej. Wreszcie, jeżeli wstawiany rekord jest typu obliczeniowego, następuje aktualizacja jego łańcucha obliczeniowego rozpoczynającego się na jego stronie głównej. Polega ona na aktualizacji adresu w nagłówku łańcucha obliczeniowego oraz ewentualnie wskaźnika wstecz w rekordzie następnym. Oznacza to, że nowy rekord wstawiany jest jako pierwszy w łańcuchu. Również w tym rekordzie wygenerowane zostają wskaźniki w przód i ewentualnie wstecz.

3.3. Usuwanie rekordów z bazy danych

Usuwanie rekordów jest procesem niezbędnym w każdym systemie. Bez tej czynności baza danych (czy tradycyjne zbiory) wcześniej czy później zapełniłaby się i byłoby niemożliwe zapamiętanie w niej jakiejkolwiek informacji. Usuwanie rekordów może być wykonywane okresowo przez programy pracujące w trybie wsadowym, bądź interakcyjnie w przypadku usuwania rekordów błędnych, zarezerwowanych czy zbędnych. Usuwanie może być procesem czasochłonnym; zależne ono jest od liczby rekordów w bazie oraz od rodzaju i liczby połączeń logicznych usuwanych rekordów. Patrząc od fizycznej strony zagadnienia, proces ten można podzielić na dwie fazy. Oto one:

1) Aktualizacja powiązań logicznych i fizycznych, w których usuwany rekord partycypował.

2) Aktualizacja strony, na której znajdował się usuwany rekord.

Faza pierwsza występuje tylko dla rekordów powiązanych z innymi zależnością grupową lub dla rekordów indeksowo-sekwencyjnych i obliczeniowych. Jeżeli usuwany rekord jest członkiem automatycznym bądź manualnym w grupie, musi nastąpić reorganizacja właściwego wystąpienia tej grupy. Polega ona na aktualizacji wskaźników w rekordzie poprzednim i ewentualnie w rekordzie następnym i właścicielu. Jeżeli usuwany rekord jest właścicielem grupy, usunięciu muszą ulec wszystkie jego rekordy członkowskie ze wszystkimi tego konsekwencjami. W przypadku grup manualnych nastąpi wypięcie wszystkich rekordów członkowskich grupy, tak że pozostaną one w bazie bez wyraźnego przydziału grupy. W wypadku grup automatycznych nastąpi fizyczne usunięcie wszystkich rekordów grupy. Jeżeli usuwamy rekord członkowski z grupy posiadającej tablicę wskaźników, wystarczy jedynie zaktualizować odpowiednią tablicę adresów.

Podobny problem jak przy usuwaniu rekordu z grupy występuje przy usuwaniu rekordów obliczeniowych. Rekordy te są powiązane ze sobą wskaźnikami tworząc zamknięte łańcuchy. Usunięcie rekordu takiego typu powoduje

konieczność reorganizacji jego łańcucha obliczeniowego. Reorganizacja taka polega na aktualizacji wskaźnika (adresu) w rekordzie poprzednim w stosunku do usuwanego. Aby przyspieszyć tę operację, co jest istotne, gdy łańcuchy są długie, można zdefiniować w rekordach obliczeniowych dodatkowe łączniki wstecz. Ułatwi to dostęp do rekordu poprzedniego, gdyż uniknie się konieczności przeczytania wszystkich rekordów łańcucha przebiegając go w przód. Skróci to oczywiście czas usuwania.

Usuwanie rekordów indeksowo-sekwencyjnych opiera się na założeniu, że nie można aktualizować stron indeksowych. Tak więc nawet w wypadku usunięcia rekordu o największym kluczu na stronie nie zmodyfikuje się strony indeksowej, na której dalej będzie się ten klucz znajdować. Konsekwencją usunięcia rekordu ze strony nadmiarowej jest aktualizacja części systemowej rekordu ze strony głównej o kluczu bezpośrednio wyższym.

Drugą fazą usuwania rekordu jest aktualizacja strony, na której był on zapisany. Sam rekord nie musi być fizycznie niszczony (zerowany) i wystarczy, że będzie on odpowiednio zaznaczony. Jednym z możliwych rozwiązań jest użycie części bitów nagłówka rekordu. Rekord taki jest niedostępny dla Systemu czy programów użytkowych, lecz w szczególnych przypadkach można jeszcze wykorzystać zawarte w nim informacje. Poprzez manualne operacje na bazie programami w języku wewnętrznym komputera lub DMU (Data Management Utility, zwłaszcza jego funkcja PATCH) można zmodyfikować nagłówek rekordu przywracając go systemowi. Fizyczne usuwanie rekordów zwolnionych następuje dopiero w przypadku pakowania strony. Ma to miejsce bądź jako zlecenie zewnętrzne DMU (COMPACT) bądź jest samoczynnie inicjowane przez System Zarządzania Bazą w razie konieczności zapamiętania nowego rekordu na stronie, na której miejsce wolne co prawda jest, lecz jest ono pokawałkowane, nieciągłe. Należy zaznaczyć, że pakowanie nie usuwa wszystkich wolnych słów na stronie. Nie są zwalniane niektóre adresy z tablicy adresowej, mimo że nie są one zajęte. Jeżeli na stronie było zapamiętanych wiele rekordów, np. 20, a następnie usunięto z niej rekordy o numerach od 2 do 15 i od 17 do 20, to mimo zwolnienia wielu adresów tablica adresowa będzie w dalszym ciągu duża. W przedstawionym przykładzie obejmować będzie ona adresy od 1 do 16. Zajmuje to niepotrzebnie miejsce na stronie. Należy o tym pamiętać podczas projektowania fizycznej wielkości strony uzależniając ją od wielkości rekordów na stronie oraz sposobu i częstotliwości ich przetwarzania. W przypadku częstego usuwania rekordów małych lepiej zdefiniować strony jak najmniejsze, aby nie dopuścić do przedstawionych wyżej strat.

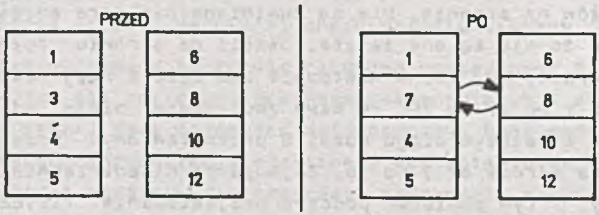
3.4. Modyfikacja rekordów

Modyfikacja (aktualizacja) rekordu może obejmować zarówno zmianę pól danych w rekordzie lub zmianę jego części systemowej. Naturalną czynnością jest aktualizacja pól w rekordzie, różnego rodzaju liczników, stanów

itp. Aktualizacja części systemowej może być spowodowana niekoniecznie poprzez bezpośrednią akcję na tym rekordzie. Podczas takich operacji, jak wstawianie czy usuwanie, może nastąpić aktualizacja rekordów sąsiednich w odpowiednich związkach logicznych czy fizycznych. Zmianie może ulec również nagłówek rekordu, dokładniej mówiąc - pole zawierające długość rekordu (zmiennej długości).

Aktualizacja pól kluczowych w rekordzie może spowodować konieczność reorganizacji grupy logicznej czy łańcucha obliczeniowego. Może to mieć miejsce w grupach posortowanych wg klucza (czy kluczy). W obu tych przypadkach modyfikowany rekord będzie miał taki sam adres jak poprzednio. Zmianie ulegnie jedynie jego część systemowa obrazująca jego nowe powiązania z innymi rekordami. Reorganizacji muszą ulec odpowiednie grupy czy łańcuchy obliczeniowe.

Również modyfikacja pola kluczowego w rekordzie indeksowo-sekwencyjnym nie zmieni adresu tego rekordu. Jeżeli zmodyfikowany klucz będzie mieścił się w zakresie kluczy dla tej strony, modyfikacja taka pociągnie za sobą jedynie konieczność reorganizacji strony. Jeżeli natomiast jest on poza zakresem - aktualizacji musi ulec rekord na innej stronie o kluczu bezpośrednio wyższym. Zostanie on zaznaczony jako duplikat, natomiast zmieniany rekord też musi być odpowiednio zaznaczony, np. poprzez ustawienie odpowiedniego bitu w nagłówku. Wstawienie tego bitu będzie oznaczało stan "klucz rekordu został zmodyfikowany", co umożliwi sekwencyjne przeglądanie strony (rekordy o zmodyfikowanym kluczu będą wtedy pomijane). Oprócz tego oba rekordy zostaną połączone wskaźnikami będącymi ich adresami (patrz rys. 3.5).

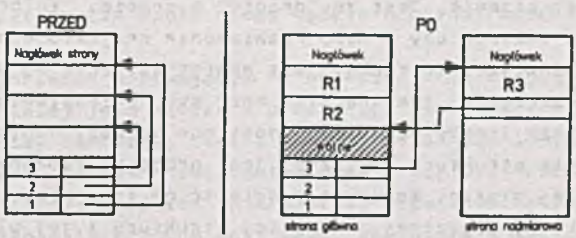


Rys. 3.5. Modyfikacja klucza w rekordzie indeksowo-sekwencyjnym. Zmiana klucza "3" na "7" spowoduje, że rekordy "7" i "8" zostaną połączone wskaźnikami, "rekord" "7" zaznaczony jako "zmodyfikowany" a rekord "8" jako "duplikat"

Istnieje jeszcze jedna forma modyfikacji rekordu będącego członkiem grupy. Bez zmiany jakiegokolwiek pola wewnątrz rekordu może on zmienić logiczny związek z właścicielem, prościej mówiąc - może zmienić właściciela. Może to nastąpić wprost poprzez zadziałanie odpowiedniej instrukcji DML (MODIFY) lub na skutek wypięcia i ponownego podpięcia rekordu członkowskiego w grupie manualnej. Wtedy to reorganizacji muszą ulec dwa różne

wystąpienia grupy. Z pierwszego z nich rekord zostanie usunięty, do drugiego wstawiony ze wszystkimi fizycznymi tego konsekwencjami. W obu grupach należy zaktualizować wskaźniki w rekordach sąsiednich.

Należy jeszcze omówić problem modyfikacji rekordów zmiennej długości. W rekordzie takim po aktualizacji może zmienić się jego długość. O ile ona się zmniejszy, następuje jedynie aktualizacja nagłówka rekordu (długość) oraz aktualizacja wielkości wolnego miejsca na stronie. Jeżeli rekord zwiększy się, może być kłopot z umieszczeniem rekordu pod tym samym adresem. Jeżeli na stronie głównej będzie dość miejsca, nastąpi tylko jej fizyczna reorganizacja - przesunięcie rekordów następnych w dół. Jeżeli na stronie będzie za mało miejsca rekord zostanie umieszczony na stronie nadmiarowej lub innej stronie z danymi. Oczywiście aktualizacji ulegnie wtedy nagłówek strony głównej - wielkość wolnego miejsca i liczba rekordów na stronie. Aktualizacji ulegnie wtedy również adres rekordu w nagłówku dolnym strony. Zamiast numeru słowa na stronie, od którego rekord rozpoczyna się, będzie tu numer nowej strony rekordu przesuniętego. Z kolei w tak zmodyfikowanym rekordzie musi być łącznik do jego strony głównej (patrz rys. 3.6). Oczywiście sytuacja taka jest niekorzystna, gdyż zwiększa czas dostępu do rekordu oraz czas trwania samej modyfikacji. Podczas projektowania bazy należy unikać definiowania rekordów zmiennej długości, jeżeli często zmieniana jest ich długość. Lepszym rozwiązaniem jest wtedy przyjęcie rekordu o stałej a większej długości (maksymalnej lub statystycznie opłacalnej) lub rozbicia takiego rekordu na dwa typy i powiązanie ich elastyczną zależnością grupową.



Rys. 3.6. Modyfikacja długości rekordu. Po modyfikacji rekord nie mieści się na stronie głównej i zostaje zapamiętany na stronie nadmiarowej. Wskaźnik w tablicy adresowej strony kieruje do strony nadmiarowej, wskaźnik w zmodyfikowanym rekordzie kieruje do strony głównej

4. PROJEKTOWANIE WSTĘPNEJ STRUKTURY LOGICZNEJ

Projektowanie struktury logicznej jest najważniejszym etapem pracy administratora przy projektowaniu nowej bazy danych. Na tym etapie można bowiem uzyskać największe wymierne efekty, jak: skrócenie czasu pracy transakcji czy zmniejszenie wielkości programów. Błędy popełnione w projekcie struktury logicznej nie dadzą się naprawić właściwym projektem struktury fizycznej. Eksploatacja użytkowych programów systemu będzie utrudniona.

Każdy system zarządzania bazami danych posiada określony zasób obiektów i ich atrybutów, którymi projektant może dowolnie dysponować. Mamy tutaj na myśli różnego rodzaju dostępy do rekordów, typy powiązań logicznych rekordów czy sposoby ich uporządkowania. Projektowanie powinno polegać na optymalnym wyborze tych obiektów dla danego konkretnego zastosowania. Zauważmy, że "obiektem" w powyższym sensie może być nawet pojedynczy łącznik - wskaźnik z jednego rekordu do drugiego. Zdefiniowanie właściwych wskaźników w grupach logicznych może znacznie ułatwić ich przetwarzanie. Z kolei sztywne definiowanie maksymalnej liczby wskaźników w grupie, rozwiązanie mające czasem swoje uzasadnienie, często w innych przypadkach jest wręcz szkodliwe. Niepotrzebnie zwiększa rekord, często opóźnia wręcz przetwarzanie. Jest za "drogie" w sensie informatycznym. Tak więc o wyborze takiego czy innego rozwiązania projektowego powinna decydować głęboka analiza. Potrzebny jest aparat służący do porównywania alternatywnych rozwiązań. Tym aparatem może być przedstawiona w rozdz. 5 metoda obliczania liczby logicznych dostępów do bazy danych.

Projektowanie struktury logicznej jest procesem złożonym i wieloetapowym. Najogólniej biorąc, można je podzielić na trzy etapy - projektowanie wstępnej struktury logicznej, ocena tej struktury i jej ulepszanie.

Projektowanie wstępnej struktury logicznej jest etapem rozpoczynającym pracę administratora nad wdrożeniem nowej bazy. Oczywiście pomijamy tutaj problemy związane z wyborem konfiguracji komputera czy wyborem i generowaniem systemu zarządzania bazą. Zakładamy, że projektant dysponuje komputerem z pamięcią masową oraz systemem operacyjnym i SZBD realizującym codasylowską koncepcję bazy. Na etapie tym następuje tworzenie ogólnego zarysu struktury logicznej nowej bazy danych. Oczywiście, aby do tego doprowadzić, należy poświęcić sporo czasu głównie na kontakt z przyszłym użytkownikiem. Należy dokładnie zdefiniować wszystkie funkcje systemu, zebrać informacje o wielkości danych i ich liczbie, o częstotliwości transakcji. To pozwoli na tworzenie najprostszych struktur danych realizują

cych pojedyncze funkcje, a następnie na scalanie ich w jedną bazę. Etap ten można więc podzielić na szereg podetapów, takich jak: zbieranie informacji od użytkownika, definiowanie i grupowanie funkcji systemu, analiza i konsolidacja danych, definiowanie rekordów i definiowanie grup logicznych. Omówmy kolejno wszystkie z nich.

4.1. Zbieranie informacji od użytkownika

Nazwa tego etapu jest oczywiście umowna. Mamy tutaj na myśli zbieranie informacji w ogóle, od zainteresowanych osób i z dokumentacji projektowej, od informatyków i przyszłych użytkowników nie będących informatykami. Celem tego etapu jest zdefiniowanie żądań pod adresem systemu oraz zebranie informacji o uwarunkowaniach liczbowych danych.

Przebieg tego etapu zależy od wielu czynników, a przede wszystkim od składu zespołu projektującego, ich doświadczenia i specjalizacji. Idealne rozwiązanie byłoby takie, że projektant systemu byłby równocześnie projektantem bazy danych. W wielu ośrodkach nie jest to jednak możliwe do zrealizowania, co wynika z braku odpowiedniej liczby specjalistów projektowania baz danych. Dlatego też stosuje się w nich inne rozwiązanie - projektant bazy nie jest przydzielony do konkretnego zespołu projektowego, nie bierze bezpośredniego udziału w rozmowach z użytkownikiem, lecz niejako z góry nadzoruje i kierunkuje pracę właściwych projektantów. W ten sposób może jednocześnie obsługiwać kilka zespołów projektowych i w rezultacie projektować jedną zintegrowaną lub kilka odrębnych baz danych. Wtedy też, we wstępnych rozmowach - wywiadach dotyczących funkcji systemu - rolę użytkownika wobec niego będzie odgrywał właściwy projektant systemu.

Innym czynnikiem wpływającym na przebieg tego etapu jest przyszły użytkownik, a ściślej jego wiedza i podejście. Są to często osoby o różnym wykształceniu czy zawodzie, często nie mający do czynienia z informatyką. W takim przypadku dobrym rozwiązaniem wydaje się być wstępne przeszkolenie tych osób z zakresu podstaw informatyki. Szkolenie takie, a zwłaszcza praktyczne pokazanie im różnych możliwości komputera i bazy danych, przybliży im znacznie nową problematykę i ułatwi kontakt z nimi. Najlepsze efekty daje zaprezentowanie im pracujących już systemów i tworzących je programów. W rozmowach z osobami - nieinformatykami należy zachować szczególną uwagę i posługiwać się prostym, zrozumiałym językiem i nie używać niezrozumiałego żargonu informatycznego pełnego wyszukanych obcojęzycznych terminów. Jest oczywiste, że nie wystarczy jedna tylko rozmowa. Należy odbyć ich wiele, do momentu całkowitego zdefiniowania problemu. Wygodnym narzędziem pracy projektanta może być magnetofon, lecz wymaga on dobrego przygotowania przeprowadzającego wywiad. Za to później projektant może spokojnie przesłuchać taśmę i wynotować najistotniejsze ustalenia. Do rozmów - wywiadów można używać specjalnych formularzy. Oczywiście - raczej

do kolejnych rozmów, nie do pierwszej. Wtedy te następne wywiady będą polegały na wspólnym sprawdzaniu i poprawianiu już wypełnionych formularzy (patrz zał. 4.1).

O przebiegu tego etapu decyduje również merytoryczna treść projektowanego systemu, jego złożoność i trudność. Projektant systemu, jak i projektant bazy powinni dobrze przygotować się do konkretnego zagadnienia, często z dziedzin sobie nieznanych. Opłaca to się jednak, gdyż łatwiej znajdą oni kontakt z przyszłymi użytkownikami, fachowcami z zakresu techniki czy ekonomii.

Głównym celem rozmów jest określenie funkcji systemu. Mówimy celowo funkcje a nie programy, gdyż o łączeniu czy rozdzielaniu programów będzie decydował projektant. Dla użytkownika jest to sprawa prawie w ogóle bez znaczenia, on ma tylko sprecyzować swoje żądania od przyszłego systemu. To sprecyzowanie żądania polega na określeniu pary "WEJŚCIE-WYJŚCIE". Należy tego dokonać dla każdej funkcji systemu; określić dane wejściowe dla każdej z nich i określić spodziewany wynik, a więc dane wyjściowe. Celowo operujemy tutaj słowem "dana", a nie np. rekord. Na tym etapie badamy związki zachodzące między pojedynczymi danymi, o ich łączeniu w rekordy będą decydowały różne czynniki, co zobaczymy później. Istotne na tym etapie jest zebranie informacji o wielkości i rodzaju danych (alfabetyk, numeryk, alfanumeryk). Informacje te mogą być zbierane w różnej formie; może to być średnia wielkość danej, wielkość minimalna i maksymalna czy wreszcie może być to dana o stałej długości. Do późniejszych obliczeń zwłaszcza struktury fizycznej istotna jest znajomość liczby wystąpień danej określonego typu. W prostej linii będzie to potem liczba rekordów określonego typu w bazie. Oczywiście istotna jest tutaj liczba maksymalna, lecz również może się przydać średnia liczba wystąpień danej.

Na tym etapie definiuje się wszystkie przyszłe funkcje systemu. O funkcjach tych zbieramy dodatkowe informacje - o trybie i częstotliwości ich przetwarzania. Mówiąc tryb przetwarzania, mamy na myśli przetwarzanie w czasie rzeczywistym, w trybie konwersacyjnym (demand) i wsadowym (batch). Dane dotyczące częstotliwości przetwarzania zbieramy w takich jednostkach, które są dogodne dla użytkownika. Czasem może to być liczba transakcji na godzinę, czasem na dobę czy nawet tydzień (miesiąc). W późniejszym czasie projektant sprowadzi te liczby do wspólnych jednostek. Ważne na tym etapie jest ewentualne wyłapanie "wąskich gardeł" systemu. Jest to często możliwe, gdyż użytkownik może znać z góry rozkład częstości transakcji w czasie zmiany czy dnia. I tak np. w systemach raportujących produkcję największy spływ transakcji następuje w godzinach kończących zmianę. Rozkład częstości transakcji nie jest równomierny, godziny kończące zmianę są właśnie tym "wąskim gardłem" systemu.

„FUNKCJE"

Nr funkcji	Nazwa lub opis funkcji	Wejście	Wyjście	Pokrewieństwo	Częstotliwość

Rys. 4.1. Formularz "FUNKCJE"

4.2. Grupowanie funkcji systemu

Etap ten, jak już to wcześniej mówiliśmy, przeplata się cały czas z etapem pierwszym - zbieraniem informacji. Te zebrane informacje zapisywane są na specjalnym formularzu i w kolejnych rozmowach - wywiadach są aktualizowane. Formularz "FUNKCJE" jest wygodnym narzędziem projektanta ułatwiającym zbieranie i precyzowanie informacji od użytkownika. Omówmy kolejne pola tego formularza:

Pierwszym z nich, poza liczbą porządkową, jest nazwa funkcji systemu. Może to być nazwa skrótowa lub dłuższy opis funkcji, np.: "Wypisz historię zatrudnienia pracownika" czy "Dostarcz informacje o zadłużeniu pracownika". Kolejnym polem jest kolumna nazwana "WEJŚCIE". Wpisuje się w niej pole (pola) wejściowe dla realizacji danej funkcji. W szczególności może to być pole kluczowe - w sensie którejś z metod alokacji rekordu lub klucz do sortowania. W obu przypadkach pole to powinno identyfikować właściwy obiekt fizyczny. Jeżeli do danej wymagany jest dostęp bezpośredni, należy to odpowiednio zaznaczyć, np. symbolem "*" przed nazwą pola. Kolejna kolumna nazwana została "WYJŚCIE". Wpisuje się tu nazwy pól, które będą wynikiem działania funkcji. W tym miejscu należy rozróżnić dane stałe i obliczane od danych uzyskanych z przyszłej bazy. Tylko te ostatnie zapisujemy w formularzu FUNKCJE. Pola powtarzające się n-krotnie zapisujemy w postaci $\{\text{nazwa - pola}\}_n$. Kolejna kolumna zatytułowana jest "POKREWIEŃSTWO". Chodzi tutaj o zaznaczenie związku logicznego, jaki zachodzi między polem wejściowym a kolejnymi polami wyjściowymi. Możemy zaznaczyć tutaj następujące związki logiczne: "1:1", "1:W" i "W:W". Związek 1:1 przedstawia zależność typu funkcyjnego, tzn. każdemu elementowi wejściowemu odpowiada dokładnie jeden element wyjściowy, określonego typu. Związek "1:W", czytany "jeden do wielu", oznacza zależność hierarchiczną, tzn. jednemu elementowi wejściowemu odpowiadać może wiele elementów wynikowych tego samego typu. Przykładem takiego związku mogą być pola "kod-pracownika" i "nr - telefonu - służbowego". Przy założeniu, że pracownik może dysponować kilkoma telefonami służbowymi, mamy taką właśnie zależność danych. Z kolei zależność typu "W:W" (czytany "wielu do wielu") odzwierciedla sytuację, gdy pola wejściowe i wyjściowe są wzajemnie powiązane zależnością typu 1:W, tzn. gdy tworzą strukturę sieciową. Przykładem takiego związku mogą być pola "nazwisko-instruktora" i "nazwa-kursu". Zauważmy, że zarówno jeden instruktor może prowadzić różne kursy, jak i jeden kurs może być prowadzony przez różnych instruktorów. Kolejną kolumną formularza FUNKCJE jest pozycja "CZĘSTOTLIWOŚĆ". Wpisujemy tutaj poznaną w wywiadzie czy projektowaną częstotliwość realizacji danej funkcji. Jednostki, w których podajemy częstotliwość, początkowo mogą być różne, dogodne dla użytkownika. Mogą one podawać liczbę transakcji na godzinę czy na dobę. Później wygodnie jest je sprowadzić do wspólnej miary, np. procentowego udziału w obciążeniu systemu. Mamy tutaj dwie możliwości - bądź

będzie to procentowy udział danej funkcji w ramach jednego projektowanego systemu, bądź procentowy udział danej funkcji w ramach wszystkich eksploatowanych w danej konfiguracji systemów. Ten drugi wariant stosuje się dla kolejno wdrażanych systemów i baz danych w ośrodku. Częstotliwość, czy może lepiej - procentowy udział funkcji w obciążeniu systemu pozwoli na ocenę obciążenia komputera i ewentualne wykrycie "wąskich gardeł" systemu.

Tak wypełniony formularz stanowi doskonały materiał do dalszej analizy. Wszystkie wymienione w nim funkcje są dokładnie sprecyzowane i opisane. Prosto nazwane funkcje, np.: "Wypisz historię zatrudnienia pracownika", można teraz dokładniej definiować dochodząc do formy: "Mając dany kod pracownika wypisz historię jego dotychczasowego zatrudnienia a mianowicie daty rozpoczęcia pracy, nazwy zakładów pracy i stanowiska".

Pozostałe informacje uzyskane z wywiadów z użytkownikiem, takie jak: wielkość pola czy liczba wystąpień danej, można zanotować na tym samym formularzu bądź też na formularzu "DANE" grupującym wszystkie pola elementarne systemu.

Jak łatwo zauważyć, na etapie tym nie operujemy pojęciem programu lecz funkcji. Oczywiście jeden program może wykonywać kilka różnych funkcji. O połączeniu funkcji w jeden program decydują przede wszystkim tryb i częstotliwość przetwarzania (muszą być analogiczne lub zbliżone) oraz konfiguracja sprzętowo-programowa w ośrodku. W systemach wieloprogramowych stosuje się zasadę pisania jak najprostszych a więc i jak najmniejszych programów. Ułatwi to ich przetwarzanie i oczywiście przysporzy. Zwiększenie liczby programów w systemie ma również swoje ujemne strony, takie jak: zwiększenie wielkości tablic kontrolnych czy zbiorów na elementy absolutne programów.

Należy zauważyć również, że analiza zebranych na tym etapie informacji może spowodować modyfikację funkcji lub nawet zupełne jej odrzucenie. Ma to miejsce wtedy, gdy ograniczenie konfiguracji (np. pojemność pamięci dyskowej) nie pozwala na przechowanie zaprojektowanej dużej liczby informacji. Należy w tym miejscu przeprojektować system; zamiast danych szczegółowych używać ich kumulacji, grupować dane elementarne w szersze typy itp. Podobny wniosek można uzyskać na drodze analizy projektowanej częstotliwości transakcji. Jeżeli przepustowość linii komunikacyjnych i procesora będzie mniejsza od projektowanego obciążenia systemu, należy również zmodyfikować czy uprościć funkcje systemu. Może też się zdarzyć, że co prawda średnie obciążenie transakcjami będzie dopuszczalne, lecz w niektórych godzinach system nie mógłby przetworzyć żądanej liczby transakcji. Są to tzw. "wąskie gardła" systemu. Również w tym przypadku wymagana jest od projektanta zmiana koncepcji systemu, polegająca często na modyfikacji czy skreśleniu niektórych funkcji.

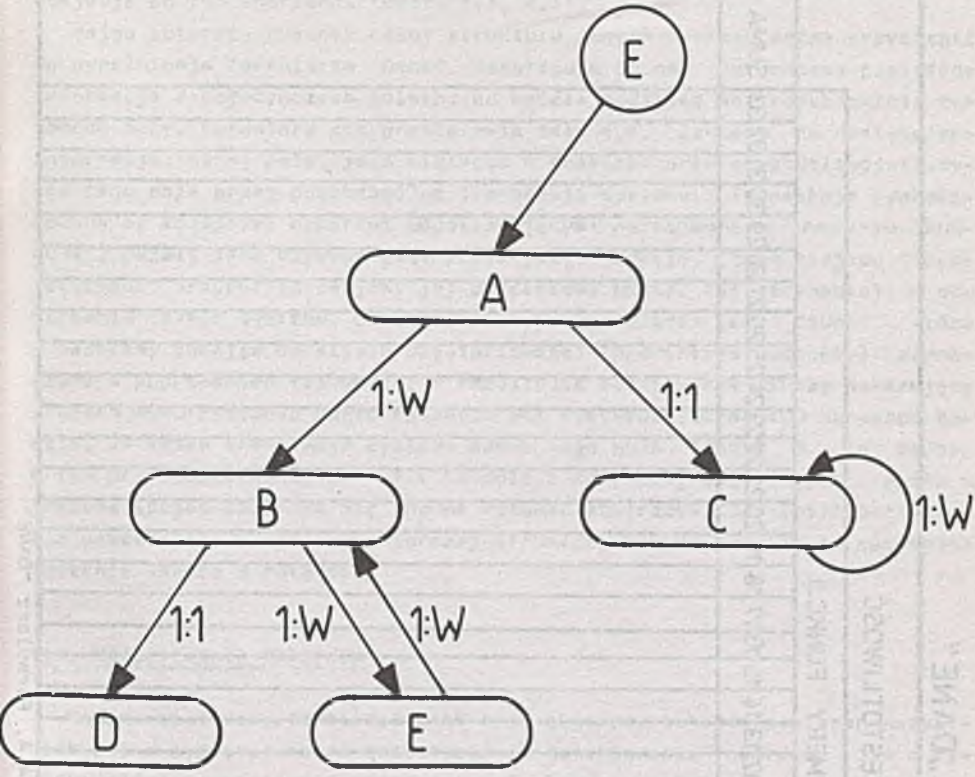
4.3. Konsolidacja danych

Celem tego etapu jest analiza wszystkich danych elementarnych poszczególnych funkcji składowych w projektowanym systemie. Rozpoczyna się no od analizy danych realizujących każdą transakcję z osobna. Funkcję oraz obsługujące ją dane warto naszkicować; można to uczynić na podstawie informacji zebranych na formularzu FUNKCJE. Przedstawienie graficzne ułatwi orientację, zwłaszcza przy późniejszym scalaniu danych w jedną bazę. Zasady rysowania prostej struktury danych realizującej daną funkcję są następujące:

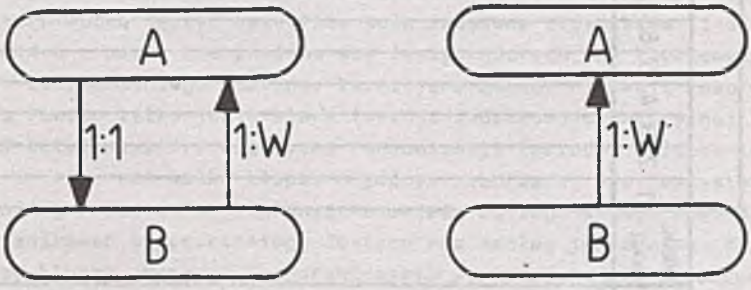
Każde pole danej reprezentowane jest przez owalny kontur z nazwą pola w środku. Jeżeli do danej wymagany jest dostęp bezpośredni, zaznaczamy to strzałką prowadzącą do konturu z literką "E" (ang. ENTRY-wejście). Literka E sygnalizuje jedynie konieczność bezpośredniego dojścia do danej, nie przesądza sprawy, jaki faktycznie dostęp będzie zrealizowany w odpowiednim rekordzie (direct, obliczeniowy, indeksowo-sekwencyjny). Jeżeli pola związane są ze sobą zależnością "1:1" czy "1:W", zaznaczamy to strzałką łączącą oba kontury. Nad strzałką zapisujemy typ zależności. Jeżeli w zależności "1:W" wymagane jest uporządkowanie danych, zaznaczamy to słowem "rosnąco" czy "malejąco" przy konturze pola podporządkowanego. Zależność "W:W" zaznaczamy przez dwie przeciwnie skierowane strzałki łączące powiązane pola. Oczywiście różne pola mogą być powiązane ze sobą poprzez wiele zależności. Może mieć miejsce przypadek, że dane pole związane jest z innymi polami tego samego typu. Symbolicznie zaznaczamy to poprzez strzałkę jednocześnie wychodzącą i wchodzącą do konturu oznaczającego daną.

W rezultacie dla każdej funkcji systemu otrzymujemy wstępną strukturę danych realizujących tę funkcję. Przykład takiej struktury to rys. 4.2, a w rozdziale 4.6 zobaczymy struktury rozwiązujące postawiony w rozdziale problem.

Wejściowe struktury danych każdej funkcji służą do konstrukcji zbiorczej dla całego systemu. Oczywiście scalanie to można wykonywać etapami, np. scalając najpierw dane w ramach podsystemów czy innych modułów przetwarzania, a następnie scalać struktury realizujące te moduły. Najłatwiej wykonuje się to graficznie, nanosząc poszczególne struktury dla transakcji na wspólny rysunek. Zasady tworzenia rysunku zbiorczego są analogiczne jak dla rysunków struktur pojedynczych. Jest jednak oczywiste, że rysunek zbiorczy wprowadzi pewne uporządkowanie elementów, czasem pewne uproszczenia czy uogólnienia. Dane pole będzie występowało na rysunku tylko raz, chyba że z jakiś powodów zdecydujemy się na celową redundancję. Uproszczeniu mogą ulec również związki logiczne zachodzące między polami. Jeżeli z analizy jednej funkcji wynika, że między polami A i B zachodzi związek typu "1:1", zaś z drugiej - że między polami B i A zwią-



Rys. 4.2. Przykład oznaczeń stosowanych na rysunku wstępnej struktury danych



Rys. 4.3. Upraszczanie związków logicznych między danymi

„DANE"

		CZĘSTOTLIWOŚĆ																															
		NUMERY FUNKCJI																															
Nazwa pola	Wielk. pola	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	SUMA

Rys. 4.4. Formularz "DANE"

zek "1:W", na zbiorczym rysunku zaznaczamy jedynie związek "1:W", gdyż obejmuje on ten poprzedni (patrz rys. 4.3).

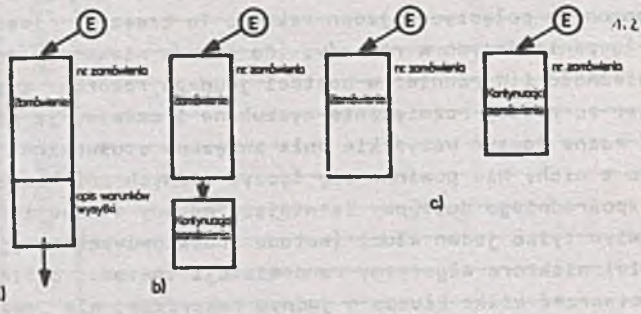
Mając zbiorczy rysunek całej struktury danych systemu można przystąpić do wypełniania formularza "DANE". Uszereguje on nam dotychczas posiadane informacje o pojedynczych polach, co będzie podstawą do projektowania rekordów bazy. Formularz ten przedstawia zał. 4.4. Zawiera on następujące informacje: nazwę pola, jego wielkość w znakach oraz częstotliwości użycia tego pola przez poszczególne transakcje systemu. Transakcje symbolizowane są kolejnymi numerami odpowiadającymi porządkowi na formularzu "FUNKCJE". Jeżeli dana używana jest przez jakąś funkcję, to w miejscu "częstotliwość" przyporządkowujemy jej procentowy udział tej transakcji w obciążeniu całego systemu. Ostatnią rubryką formularza jest "SUMA", którą wypełniamy dodając do siebie częstotliwości (procentowe udziały) zaznaczone w poprzednich kolumnach. W rezultacie otrzymujemy liczbę wskazującą stopień wykorzystania poszczególnych pól systemu. Liczba 100 oznaczać będzie, że każda transakcja systemu używa tego pola, liczba 0 - że żadna. W tym przypadku należy skreślić to pole z przyszłej bazy danych, o ile w jakichś sposób znalazło się ono na rysunku zbiorczym. Częstotliwości użycia poszczególnych pól ze zbiorczej struktury danych pozwolą na optymalne łączenie danych w rekordy.

4.4. Definiowanie rekordów

Mając wypełniony formularz DANE oraz zbiorczy schemat powiązań wszystkich pól w systemie można przystąpić do definiowania rekordów. Już z rysunku widać, że dane związane ze sobą zależnością 1:1 grupują się obok siebie i można je połączyć w jeden rekord. To zresztą jest podstawowym warunkiem łączenia danych w rekordy. (Co prawda niekiedy opłaca się realizować zależność 1:W również w postaci jednego rekordu zmiennej długości, ale jest to raczej rozwiązanie wyszukane i omówimy je później). W jeden rekord można łączyć wszystkie pola związane stosunkiem "1:1" lub tylko niektóre z nich. Nie powinno się łączyć różnych pól kluczowych używanych do bezpośredniego dostępu. Istniejące metody alokacji rekordów dopuszczają bowiem tylko jeden klucz (metoda indeksowo-sekwencyjna) w rekordzie. Z kolei niektóre algorytmy randomizacji (metoda obliczeniowa) potrafią przetwarzać kilka kluczy w jednym rekordzie, ale wszystkie z nich muszą być jednocześnie ustalone. Natomiast do tego samego rekordu nie można zorganizować bezpośredniego dostępu raz według jednego a raz według drugiego klucza. Narzuca to ograniczenie o rozdziale pól kluczowych między różne rekordy. Oczywiście można takie pola połączyć w jeden rekord, ale tylko jedno z tych pól może być zdefiniowane jako klucz. Dostęp do rekordu według drugiego pola musi być realizowany sekwencyjnie lub na drodze własnych oryginalnych metod (patrz rys. 4.13). Z uwagi na to, że do-

stęp tak zorganizowany będzie bardziej czasochłonny niż dostęp bezpośredni, powinno go się stosować w rzadziej używanych transakcjach.

O łączeniu bądź rozdzielaniu pól na rekordy decyduje również wielkość pola i częstotliwość jego użycia w systemie. Jest intuicyjnie wyczuwalne, że pola duże, a jednocześnie rzadko używane powinny być umieszczone poza głównym rekordem. Im rekord jest mniejszy, tym szybszy będzie dostęp do niego i szybsze jego przetwarzanie. Pole rzadko używane niepotrzebnie zwiększałoby ten rekord, a większość transakcji i tak z tego pola nie korzystałoby. Zwróćmy uwagę na to, że termin "rzadko używany" nie oznacza, że pole jest wykorzystywane przez małą liczbę różnych typów transakcji, lecz mamy tu na myśli niskie sumaryczne obciążenie systemu wszystkimi transakcjami stosującymi to pole. W tym więc przypadku można zdefiniować dwa typy rekordów: rekord główny i jego kontynuację zawierającą rzadziej używane pola. Rekord - kontynuacja może zawierać to samo pole kluczowe co rekord główny (celowa redundancja), o ile ma być do niego dostęp bezpośredni, lub być powiązany z rekordem głównym zależnością pseudogrupową (grupa o jednym tylko rekordzie członkowskim lub grupa pusta). W szczególności rekord-kontynuacja może mieć zmienną długość. Rozwiązanie takie stosuje się głównie do zapamiętania w takim rekordzie opisu słownego, tekstu itp. Przykładem może być rekord - kontynuacja - zamówienia zawierający opis warunków wysyłki towaru, w którym umieszczono takie informacje, jak: rodzaj opakowania, warunki transportu, adresy itp. Zakładamy oczywiście, że opis będzie jednym, niepodzielnym polem, przetwarzanym całościowo, np. do wydruku specyfikacji wysyłkowej (patrz rys. 4.5).



Rys. 4.5. Przykład rekordu - kontynuacji. Możliwe rozwiązanie
a) rekord zmiennej długości, b) pseudogrupa z 1 rekordem - kontynuacją, c) bezpośredni dostęp do rekordu - kontynuacji

Niekiedy zdarza się, że dane występują nie we wszystkich realizacjach rekordów. Jeżeli liczba takich rekordów jest duża, nie opłaca się definiować w tym typie rekordu takich pól. Lepiej przenieść je do rekordu - kontynuacji. Zaoszczędzimy w ten sposób sporo miejsca w pamięci masowej. Weźmy np. zbiór rekordów typu PRACOWNIK opracowany dla jakiegoś systemu.

Załóżmy, że rekordów tych będzie około 10000 i że dla około 1000 z nich (pracownicy awaryjno-techniczni i kadra) będzie wymagane zapamiętanie adresu domowego i nr telefonu domowego. Obie te dane powinniśmy umieścić poza rekordem głównym,gdyż w przeciwnym razie stracilibyśmy duży obszar pamięci masowej dokładnie 9000 x (wielkość pola NR-TEL-DOM + wielkość pola ADRES-DOM).

Przy podejmowaniu decyzji o łączeniu czy rozdziale pól wygodne jest wykonanie prostych obliczeń pozwalających ocenić stopień dopasowania pól do siebie. Wykonajmy tabelkę zawierającą pozycje: nazwa pola, wielkość pola (w znakach) i częstotliwość (w %). Miernikiem możliwości łączenia pól niech będzie iloraz "częstotliwość/wielkość pola". Pola, dla których iloraz ten jest dużo mniejszy od pozostałych (różnica co najmniej rzędu lub kilku rzędów), należy umieścić poza rekordem głównym (patrz rys. 4.6).

Nazwa pola	Wielkość pola (w znakach)	Częstotliwość (%)	Częst./wielkość
A	10	5	0,5
B	100	5	0,05
C	10	80	8
D	100	80	0,8

Rys. 4.6. Łączenie pól w rekordy. Obliczanie ilorazu "częstotliwość/wielkość"

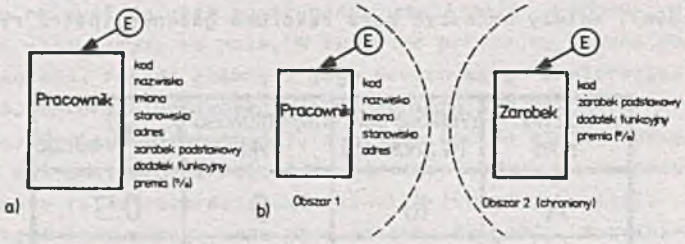
W tym przypadku pola należałoby podzielić na dwa rekordy: w rekordzie głównym pola A, C, D, w rekordzie - kontynuacji pole B. Jest to oczywiście tylko jedno z sensownych rozwiązań.

Inne z nich można symbolicznie zaznaczyć następująco:

1) A + B + C + D tzn. 1 rekord,
2) A + B, C + D tzn. 2 rekordy,
3) A + C + D, B tzn. 2 rekordy,
4) D, C + A, B tzn. 3 rekordy.

Przy projektowaniu wstępnej struktury proponuje się przyjęcie rozwiązania 2 lub 3. Dokładniejszą ocenę wszystkich alternatywnych rozwiązań dla konkretnego systemu można wykonać za pomocą metody obliczania liczby logicznych dostępów do bazy przedstawionej w rozdziale 5. Wtedy też może okazać się, że dla jakiegoś specyficznego systemu optymalny rozkład pól na rekordy będzie inny niż 2 lub 3.

O podziale pól na rekordy mogą również decydować inne względy, jak np. ochrona pól przed niepowołanym dostępem. Niektóre systemy posiadają możliwość ochrony (poprzez klucz) nawet pojedynczych pól w rekordzie, ale jest to jeszcze stosunkowo rzadkie rozwiązanie. Gdy takiej możliwości nie ma, można rekord rozbić na dwa rekordy i umieścić je w różnie chronionych obszarach. Na przykład pola dotyczące zarobków pracownika, mimo że są związane z resztą zależnością 1:1, można wydzielić tworząc z nich nowy rekord i umieścić go w innym obszarze, do którego dostęp jest tylko poprzez klucz (patrz rys. 4.7).



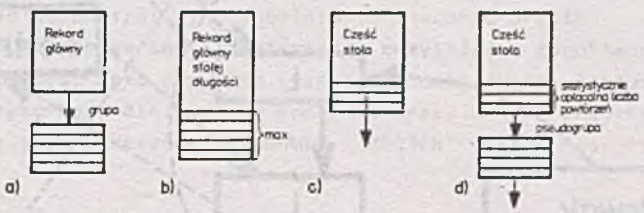
Rys. 4.7. Łączenie pól w rekordy a ochrona danych
a) dane w jednym rekordzie, b) dane podzielone na dwa rekordy przechowywane w różnych obszarach

O budowie rekordu mogą również decydować ograniczenia sprzętowe, takie jak: pojemność pamięci zewnętrznej czy operacyjnej. Jeżeli komputer posiada małą pamięć operacyjną, bufory na strony też nie będą duże. Aby je lepiej wykorzystać, we wszystkich obszarach definiuje się małe strony, w miarę możliwości o jednakowej długości. Z kolei małe strony pociągają za sobą konieczność definiowania małych rekordów, co najwyżej równych wielkości strony. Tak więc w niektórych systemach komputerowych projektant może mieć ograniczenie na maksymalną wielkość rekordów.

W przypadku przekroczenia zadanej wielkości należy rozbić rekord na dwa lub więcej mniejszych rekordów lub nawet generalnie zmienić koncepcję.

Dane związane ze sobą zależnością 1:W rozbijamy przeważnie na 2 typy rekordów i wiążemy te rekordy poprzez zależność grupową. Mogą być od tego wyjątki, ale raczej w specyficznych wypadkach. Na przykład gdy liczba powtórzeń jest niska, rekord główny może mieć stałą długość, a tablica w nim maksymalny rozmiar. Oczywiście dopuszcza się w tym rozwiązaniu to, że pewne pola tablic nie będą nigdy wykorzystane. Rozwiązanie takie nie jest elastyczne, w przypadku pojawienia się w systemie konieczności zapamiętania tablicy o większym wymiarze, należy reorganizować bazę danych. Pewniejszym rozwiązaniem jest zdefiniowanie rekordu zmiennej długości.

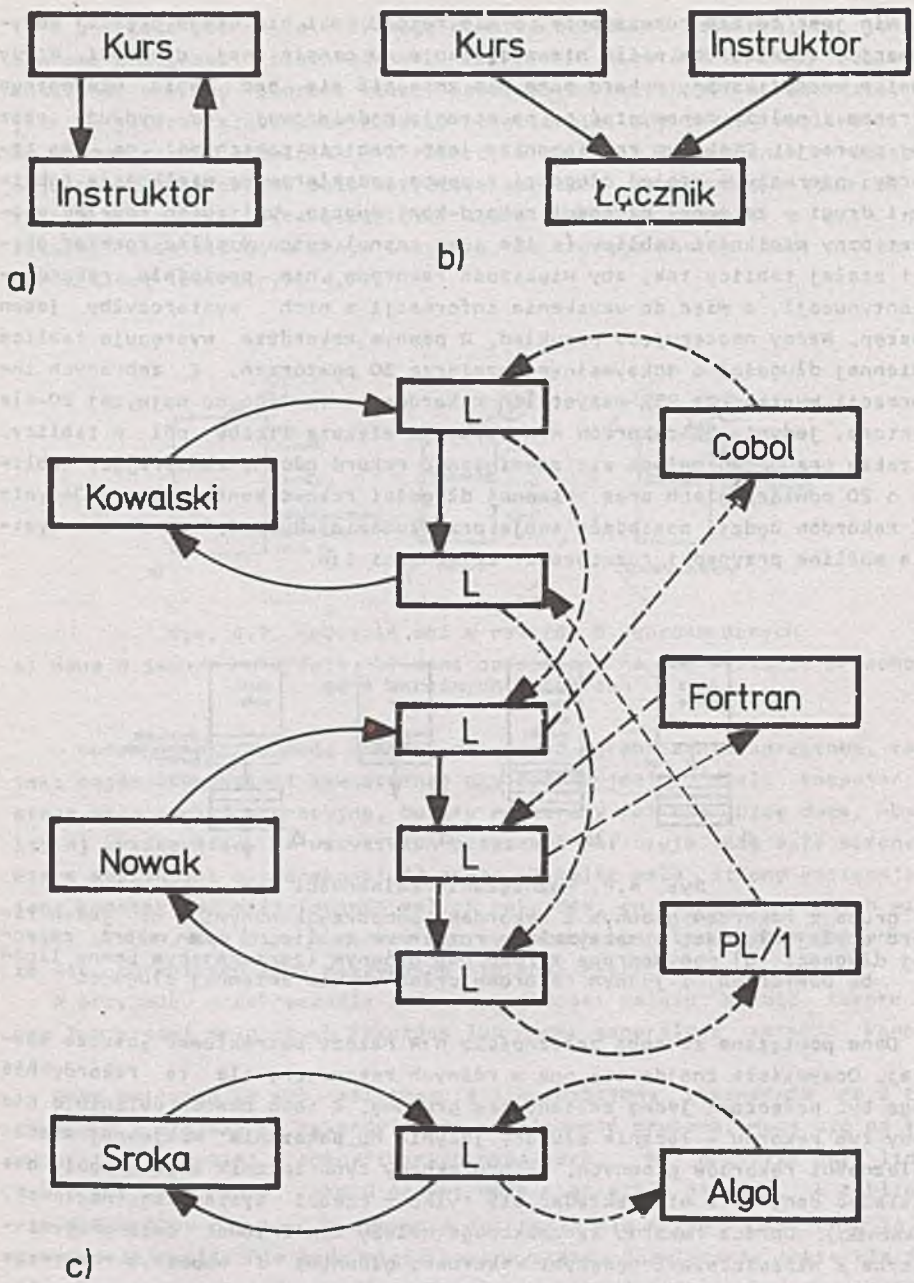
Nie jest to złe rozwiązanie, o ile rekord taki nie ulega częstej modyfikacji, a zwłaszcza o ile nie modyfikuje on często swej długości. Wtedy bowiem zmodyfikowany rekord może nie zmieścić się pod swoim pierwotnym adresem i należy zapamiętać go na stronie nadmiarowej, co wydłuża czas tej operacji. Ciekawym rozwiązaniem jest rozbicie takich pól na dwa rekordy: pierwszy - stałej długości z pewną zadeklarowaną wielkością tablicy i drugi - zmiennej długości rekord-kontynuacja. Analizując rozkład statystyczny wielkości tablicy (o ile jest znany) można określić rozmiar części stałej tablicy tak, aby większość rekordów nie posiadała rekordów-kontynuacji, a więc do uzyskania informacji z nich wystarczyłby jeden dostęp. Weźmy następujący przykład. W pewnym rekordzie występuje tablica zmiennej długości o maksymalnym rozmiarze 30 powtórzeń. Z zebranych informacji wynika, że 95% wszystkich rekordów ma tablice co najwyżej 20-elementowe, jedynie 5% rekordów wykorzystuje większą liczbę pól w tablicy. W takim przypadku opłaca się zdefiniować rekord główny zawierający tablicę o 20 powtórzeniach oraz zmiennej długości rekord-kontynuację. Jedynie 5% rekordów będzie posiadało swoje przedłużenia. Rys. 4.8 zawiera wszystkie możliwe przypadki rozwiązania zależności 1:W.



Rys. 4.8. Rozwiązanie zależności 1:W

a) grupa z rekordem głównym i rekordami podporządkowanymi, b) jeden rekord stałej długości o maksymalnym rozmiarze tablic, c) jeden rekord zmiennej długości, d) pseudogrupa z rekordem głównym (zawierającym pewną liczbę powtórzeń) i jednym rekordem członkowskim zmiennej długości

Dane powiązane ze sobą zależnością W:W należy potraktować jeszcze inaczej. Oczywiście znajdą się one w różnych rekordach, ale te rekordy nie mogą być połączone jedną zależnością grupową. Z tego powodu definiuje się nowy typ rekordu - łącznik służący jedynie do pokazania wzajemnej współzależności rekordów głównych. Pseudorekordy typu łącznik mogą w ogóle nie posiadać danych, a więc składać się tylko z części systemowej (nagłówek, wskaźniki). Oprócz rekordu łącznikowego należy zdefiniować dwie grupy logiczne z właścicielami będącymi rekordami głównymi i wspólnym rekordzie członkowskim, właśnie typu łącznik. Dostęp do pseudorekordu łącznik będzie realizowany przez jedną lub drugą grupę. Rekord-łącznik będzie zapamiętywany w bazie, gdy stwierdzimy, że pewne wystąpienia rekordów głów-

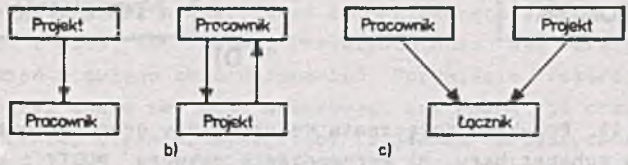


Rys. 4.9. Przykład rozwiązania zależności W:W
a) schemat ideowy, b) schemat struktury bazy, c) fizyczna realizacja sieci

nych obu typów są wzajemnie związane. Pokażmy to na przykładzie (patrz rys. 4.9).

Załóżmy, że INSTRUKTOR jest rekordem zawierającym informację o wykładowcy, a KURS rekordem opisującym tematykę i przebieg danego szkolenia. Rekordy INSTRUKTOR i KURS są powiązane ze sobą zależnością typu W:W. Dany wykładowca może bowiem przeprowadzać szkolenia z różnych dziedzin i odwrotnie, ten sam kurs może być prowadzony przez różnych instruktorów. Jeżeli konkretny instruktor, np. Kowalski, potrafi przeprowadzić kurs COBOL-u i PL1, to w jego występieniu grupy INSTRUKTOR-ŁĄCZNIK znajdą się dokładnie dwa rekordy członkowskie. Z kolei te same rekordy typu łącznik znajdą się w odpowiednich występieniach grupy KURS-ŁĄCZNIK, a mianowicie występieniach z właścicielem odpowiednio COBOL i PL1. Przeglądając grupę INSTRUKTOR-ŁĄCZNIK i pobierając za każdym razem właściciela drugiej grupy, uzyskamy wykaz kursów, które może prowadzić dany wykładowca. Z kolei przeglądanie grupy KURS-ŁĄCZNIK z jednoczesnym pobieraniem informacji o właścicielu grupy INSTRUKTOR-ŁĄCZNIK umożliwi wybranie wszystkich wykładowców przygotowanych do prowadzenia danego kursu.

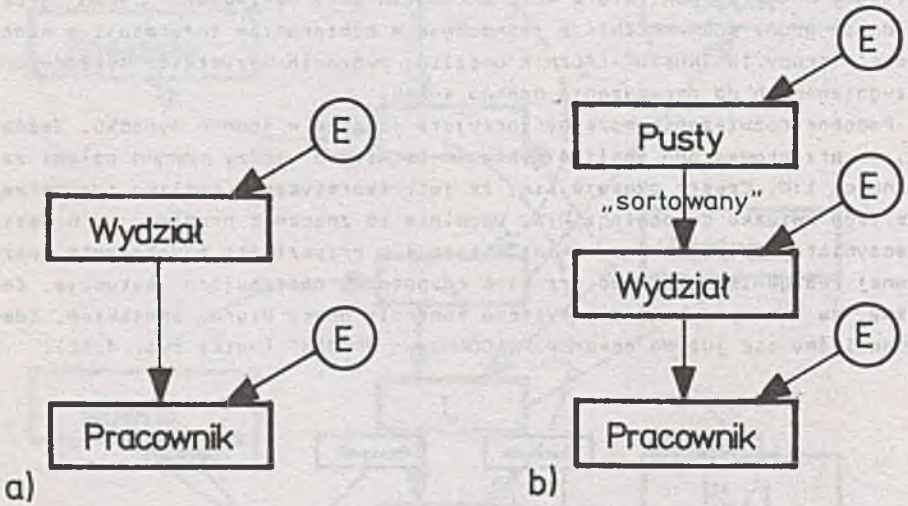
Podobne rozwiązanie może być przyjęte jeszcze w jednym wypadku. Załóżmy, że przeprowadzona analiza wykazała istnienie między pewnymi polami zależności 1:W. Często okazuje się, że jest teoretycznie możliwe rozszerzenie tego związku do postaci W:W. Uogólnia to znacznie problem, a w razie rzeczywistego wystąpienia takiej sytuacji w przyszłości zapobiegnie kosztownej reorganizacji. Jako przykład rozpatrzmy następującą sytuację. Załóżmy, że tworzymy bazę dla systemu kontroli pracy biura projektów. Zdecydowaliśmy się już na rekordy PRACOWNIK i PROJEKT (patrz rys. 4.10).



Rys. 4.10. Uogólnianie zależności 1:W na W:W
a) schemat ideowy - początkowy, b) schemat ideowy - uogólniony, c) rozwiązanie zależności W:W

Między rekordami PROJEKT i PRACOWNIK zachodzi oczywista zależność 1:W, która oznacza, że projekt wykonuje pewna grupa pracowników biura. Aktualnie mamy taki stan, że każdy pracownik jest przydzielony tylko do jednego projektu, łatwo jednak sobie wyobrazić sytuację, że pewne osoby będą jednocześnie pracowały w kilku projektach, np. konsultant, kreślarz, maszynistka. To uzasadnia przyjęcie niejako na wyrost zależności typu W:W rozwiązywanej, jak wiemy, za pomocą rekordu-łącznika.

Za pomocą rekordu-łącznika można również wychwycić pewne zależności występujące w ramach jednego typu rekordu. Załóżmy, że rozpatrujemy zbiór rekordów PRACOWNIK. Wśród tych rekordów są oczywiście takie, które reprezentują podwładnych, inne zaś przełożonych. Nie chcemy w żaden sposób rozróżnić tych rekordów. Nie byłoby to zresztą możliwe, gdyż każdy pracownik ustawiony jest jakoś w hierarchii służbowej i wielu z nich jest jednocześnie i podwładnymi i szefami. Powiązanie rekordów PRACOWNIK poprzez prostą grupę logiczną nie jest możliwe, choćby z uwagi na to, że dysponujemy jednym tylko typem rekordu. Można jednak zdefiniować rekord-łącznik i dwie dodatkowe grupy PODWŁADNI i PRZEŁOŻENI (patrz rys. 4.11). Grupa PODWŁADNI poprzez rekordy łącznik wskaże nam wszystkich pracowników podległych danemu. Z kolei grupa PRZEŁOŻENI może pokazać wszystkich szefów każdego pracownika.



Rys. 4.11. Przykład dołączania do struktury grup sortowanych
a) początkowy schemat bazy, b) wprowadzenie rekordu PUSTY i posortowanej grupy "PUSTY-WYDZIAŁ"

W bazie danych można definiować również rekordy robocze, nie mające co prawda wiele wspólnego z zebranymi danymi, ale spełniające pewne funkcje utylitarne. Rekordy tego typu mogą być różnego rodzaju tablicami konwersji, słownikami, mogą ułatwiać obliczenie (np. zmiana daty do innego formatu) czy wręcz omijać je. Umieszczenie rekordów roboczych w bazie może ułatwić przetwarzanie czy dostęp do rekordów głównych. Z uwagi na raczej małą ich liczbę koszt tego rozwiązania jest stosunkowo niski. W uzasadnionych przypadkach baza może nawet służyć do komunikowania się różnych programów. Może to się odbywać przez specjalne rekordy robocze, które pierwszy z programów zakłada, a drugi po wykorzystaniu usuwa. Czas pobytu

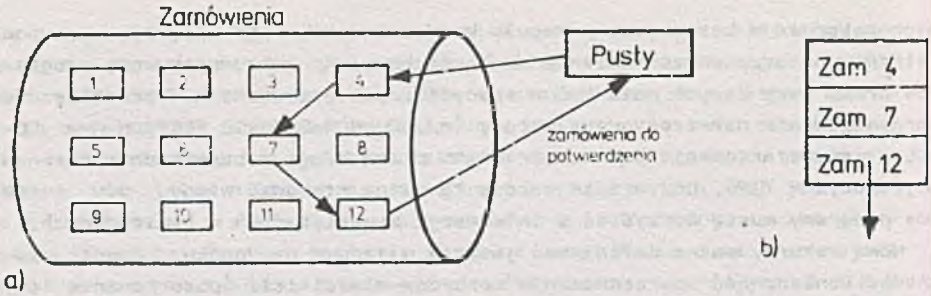
tych rekordów w bazie jest z reguły bardzo krótki, nie wpływają więc one istotnie na stopień zapełnienia obszarów bazy. Kontaktowanie się programów przez bazę danych może być w specyficznych przypadkach prostsze do zorganizowania niż tradycyjne metody (np. COMMON-STORAGE SECTION w COBOL-u, czy startowanie jednego programu przez drugi z odpowiednim ekranem wejściowym - TIP). Oczywiście metodę tę można stosować wtedy, gdy i tak oba programy muszą korzystać z informacji zapamiętanych w bazie danych.

Nowe rekordy można definiować jeszcze w jednym przypadku. Często zachodzi konieczność uporządkowania rekordów-właścicieli (posortowanie ich) lub odpowiedniego ich pogrupowania ułatwiającego pracę.

Problem ten można rozwiązać definiując dodatkową grupę z rekordem pustym jako właścicielem i właściwym typem rekordu jako członkiem. Podobnie jak rekord-łącznik rekord-właściciel w tej grupie może nie posiadać w ogóle danych. Dostęp do niego jest przeważnie bezpośredni (direct). Rozwiązanie takie jest złe, gdy liczba rekordów członkowskich jest duża, a dostęp do nich jest swobodny, np. poprzez procedurę randomizacyjną. Wtedy przeczytanie całej grupy wymaga praktycznie wykonania tylu fizycznych dostępów do bazy, ile jest rekordów członkowskich. Lepszym rozwiązaniem jest wtedy sekwencyjne czytanie rekordów w obszarze i sortowanie ich poza bazą. Jednak w niektórych przypadkach rozwiązanie takie może być sensowne. Załóżmy np. że zdefiniowaliśmy już rekordy WYDZIAŁ i PRACOWNIK. Dostęp do obu tych rekordów jest bezpośredni, jednak rekordów typu WYDZIAŁ jest zaledwie kilka. Jeżeli niektóre transakcje czy nawet programy wsadowe wymagają posortowania według kodu wydziału, można zdefiniować nową grupę z pustym rekordem właścicielem i zdefiniować porządek w tej grupie jako "sortowany" (patrz rys. 4.11). Może to poważnie przyspieszyć przetwarzanie, zwłaszcza odczyt rekordów.

Podobne rozwiązanie można zastosować celem czasowego wyróżnienia rekordów jednego typu z dużej ich liczby. Przykładem może być drukowanie potwierdzeń zamówień z dużego zbioru zamówień. Oczywiście rekordy-zamówienia mogłyby poprzez pewne swe pole wskazywać, czy można je przyjęć i wystawić potwierdzenie, czy nie. Wymagałoby to jednak przeczytania wszystkich rekordów obszaru, a więc przeczytania wszystkich stron. Jeżeli liczba zamówień do wydruku będzie niska (niższa niż liczba stron w obszarze), można powiązać te rekordy zależnością grupową z rekordem pustym jako właścicielem. Oczywiście przeważająca liczba rekordów-zamówień nie będzie występowała w tak zdefiniowanej grupie (patrz rys. 4.12). Przetwarzanie zamówień do potwierdzenia będzie jednak znacznie ułatwione i szybsze. Ten problem można zresztą rozwiązać inaczej, poprzez zdefiniowanie rekordu zmiennej długości zawierającego tablicę adresów czy kluczy zamówień do dalszej obróbki.

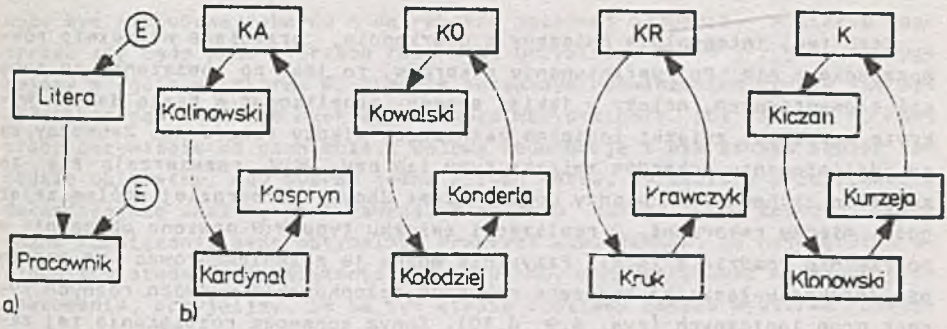
Pseudorekordy organizacyjne mogą również służyć do usprawnienia dostępu do rekordów według pól nie będących kluczami.



Rys. 4.12. Wybieranie rekordów spełniających szczególne warunki z dużego zbioru rekordów
a) przy pomocy grupy, b) poprzez tablicę adresów (kluczy)

Załóżmy, że w rekordzie PRACOWNIK polem kluczowym używanym do alokacji jest pole "kod-pracownika". Załóżmy też, że niektóre transakcje wymagają bezpośredniego dostępu do tego rekordu, przy czym wejściem do nich jest pole "nazwisko-pracownika". Oczywiście można przeczytać cały zbiór pracowników, by wybrać z niego właściwy rekord, tzn. o zadanym nazwisku. Z uwagi na dużą liczbę rekordów PRACOWNIK procedura ta może być czasochłonna. Aby ją przyspieszyć, można pogrupować wstępnie rekordy PRACOWNIK w następujący sposób. Zdefiniujmy pseudorekordy o bezpośrednim dostępie odpowiadające kolejnym literom A,B,C... Niech litery będą polami kluczowymi w tych rekordach. Zdefiniujmy grupę logiczną "LITERA-PRACOWNIK" łączącą pracowników o nazwiskach zaczynających się na tę samą literę. Aby znaleźć teraz konkretnego pracownika, wystarczy przejrzeć grupę, której właścicielem jest litera rozpoczynająca szukane nazwisko. Jeżeli w dalszym ciągu grupy odpowiadające niektórym literom będą zbyt liczne, można na podobnej zasadzie utworzyć rekordy nagłówkowe z dwóch początkowych liter nazwiska. Oczywiście wybrać należy bardziej popularne pary liter, wszystkie pozostałe rzadziej występujące można połączyć razem w jedną grupę (patrz rys. 4.13).

Przy definiowaniu rekordów należy mieć ciągle na uwadze ograniczenie ich wielkości. Mniejsze rekordy łatwiej i szybciej są przetwarzane, łatwiej też znaleźć miejsce na ich zapamiętanie. Poza formalnym rozbiciem różnych danych na różne rekordy można zastanowić się, jak zmniejszyć fizyczną wielkość rekordu o już ustalonych polach. Przede wszystkim można tego dokonać poprzez wybór kodu, w którym przechowywać będziemy dane, co w niektórych systemach jest dopuszczalne. Na przykład w systemie UNIVAC-1106 możliwe jest przechowywanie danych w kodzie FIELDATA (6 znaków na słowo) lub w kodzie ASCII (4 znaki na słowo). Oczywiście baza danych, w której dane przechowywane są w kodzie FIELDATA, będzie oszczędniejsza. Kolejna sprawa to pamiętanie danych numerycznych w postaci binarnej. Co prawda pogorszą się wtedy warunki bezpośredniego odczytu i poprawiania danych bez udziału programów (dump, patch), ale oszczędzimy w ten sposób sporo miejsca. Co więcej, jeżeli pola te są często wykorzystywane do one-



Rys. 1.13. Pseudorekordy organizacyjne LITERA - przykład
a) schemat bazy, b) kilka wystąpień grupy LITERA - PRACOWNIK

cji arytmetycznych (np. do sumowania), uzyskamy wyraźne ich przyspieszenie. Kolejną możliwością jest symboliczne kodowanie pewnych pól alfanumerycznych. W tym jednak przypadku należy postępować wyjątkowo ostrożnie. Co prawda na pewno uzyska się tą drogą zmniejszenie rekordu, ale kodowanie i dekodowanie może znacznie wydłużyć czas przetwarzania informacji z tego rekordu. Będzie to miało miejsce zwłaszcza wtedy, gdy tablice kodów będą duże i ich przeszukiwanie musi trochę potrwać. Często można zastosować metodę kombinowaną - częściowego kodowania. Otóż niekiedy zdarza się, że jakieś pole przybiera w przeważającej liczbie rekordów zaledwie kilka wartości. Na przykład w przypadku adresów zamieszkania pracowników jakiegoś zakładu w 80%-90% przypadków będziemy mieli do czynienia z kilkoma powtarzającymi się nazwami miast. Dla nich właśnie można zastosować kody zamiast dłuższych z reguły nazw. Miasta mniej popularne będą pamiętane wprost, bardziej popularne za pomocą kodu. Rekord zawierający takie pole powinien być zdefiniowany jako rekord zmiennej długości (nazwa miasta jako tablica o zmiennej liczbie znaków). To samo rozwiązanie można stosować jeszcze w innym przypadku, celem przyspieszania wprowadzania danych z monitora. Jeszcze jedną możliwość daje powtarzanie się tej samej wartości w kilku kolejnych polach. Wtedy też, zamiast pełnej nazwy, można używać jakiegoś symbolu oznaczającego powtórzenie poprzedniej zawartości, a więc o sensie "jak wyżej" (np. adres stały, adres tymczasowy). Co prawda rozwiązanie to nadaje się głównie do wprowadzania danych, ale w uzasadnionych przypadkach dzięki niemu możemy uzyskać zmniejszenie się niektórych rekordów. Oczywiście i w tym przypadku rekord musi być zdefiniowany jako zmiennej długości.

4.5. Definiowanie grup logicznych

Etap ten, integralnie związany z poprzednim, przebiega właściwie równocześnie z nim. Po zdefiniowaniu rekordów, to jest po powiązaniu ze sobą pól elementarnych, należy w jakiś sposób zrealizować w bazie danych wykryte wcześniej związki logiczne zachodzące między rekordami. Zauważmy, że po zdefiniowaniu rekordów związki typu 1:W czy W:W rozszerzają się ze związków zachodzących między pojedynczymi danymi na bardziej ogólne zależności między rekordami. O realizacji związku typu W:W mówiono obszernie w poprzednim rozdziale (4.4). Fizycznie można je zaimplementować za pomocą pseudorekordu-łącznika będącego rekordem członkowskim w dwóch różnych typach grup logicznych (rys. 4.9, 4.10). Innym sposobem rozwiązania tej zależności jest zdefiniowanie dwuwymiarowej tablicy (rys. 4.14), która odzwierciedli wzajemny związek rekordów.

Załóżmy, że A,B,C,D... będą kolejnymi wystąpieniami rekordów jednego typu, zaś a,b,c,d kolejnymi realizacjami drugiego typu rekordów. Jeżeli oba typy rekordów pozostają ze sobą w zależności W:W, tak przygotowana tablica pokaże, które rekordy są ze sobą powiązane. Wystarczy bowiem na przecięciu się odpowiedniej kolumny i wiersza zaznaczyć ten stan, np. poprzez specjalny symbol (*) lub 1. Tablicę tę można przeglądać wierszami lub kolumnami i wtedy otrzymamy żądaną grupę rekordów zależnych. Omówione rozwiązanie ma jednak istotny mankament. Jest nim rozmiar tablicy. W przypadku dużej liczby rekordów obu typów tablica ta może być tak duża, że nie zmieści się w dostępnym obszarze pamięci operacyjnej i jej przetwarzanie będzie utrudnione. Tak więc przy dużej liczbie rekordów należy zdecydować się raczej na rozwiązanie pierwsze, tzn. za pomocą rekordu-łącznika.

Rekord 1 \ Rekord 2	a	b	c	d	e	f
A		X		X	X	
B	X	X				X
C			X	X		
D	X				X	X

Rys. 4.14. Rozwiązanie zależności W:W przy pomocy tablicy dwuwymiarowej

Inną sytuację mamy w przypadku związku typu 1:W zachodzącym między rekordami dwóch typów, np. A i B. W tym przypadku mamy kilka sensownych rozwiązań. Pierwszym z nich jest zdefiniowanie grupy logicznej z rekordem A

jako właścicielem i rekordem B jako członkiem grupy. Drugim rozwiązaniem może być rozbudowa rekordu A do rekordu zmiennej długości, w którym powtarzać się będą pola tworzące rekord B. Oczywiście w tym przypadku rezygnujemy w ogóle z rekordu B. Trzecim sensownym rozwiązaniem wydaje się być dołożenie do rekordu B niektórych (może wszystkich?) pól rekordu A. Mamy wtedy oczywiście do czynienia z celową redundacją i nie zawsze zabieg ten będzie opłacalny. Aby wybrać jedną z tych metod, należałoby przeprowadzić dokładniejszą analizę porównawczą. W różnych konkretnych zastosowaniach różne rozwiązania będą optymalne. Praktyka mówi jednak, że najczęściej opłaca się stosować rozwiązanie pierwsze. Aby usystematyzować metodykę projektowania, przyjmijmy, że na tym etapie będziemy zawsze wybierać rozwiązanie za pomocą grupy logicznej. Ocenę tego rozwiązania i ewentualne jego ulepszenie przedstawimy w rozdziałach następnych.

Należy jeszcze powiedzieć o możliwości definiowania różnych grup logicznych ułatwiających przetwarzanie danych. Mimo braku formalnych przesłanek (a więc zależności 1:W) można już na tym etapie definiować grupy wiążące w jakiś sposób rekordy określonych typów. Łączy się to z koniecznością wyboru rekordu właściciela. Rekord taki może być wybrany z już istniejących rekordów jako jedno konkretne wystąpienie danego typu rekordu. Oczywiście pola tworzące ten rekord mogą mieć inne znaczenie lub być w ogóle niewykorzystane. Można również zdefiniować nowy typ rekordu, a właściwie pseudorekordu bez danych używanego jedynie jako nagłówek łańcucha rekordów z grupy (patrz rozdział poprzedni).

Należy wspomnieć jeszcze o dwóch sprawach związanych z definiowaniem grupy. Pierwszą z nich jest nazwa grupy, którą będziemy się potem posługiwać czy w opisie programów, czy na rysunkach bazy danych. Najczęściej stosowanym rozwiązaniem jest tworzenie nazwy grupy z nazw definiujących ją rekordów. Mogą to być np. pełna nazwa rekordu właściciela i nazwa rekordu członkowskiego połączone myślnikiem, mogą to być skrócone nazwy rekordów (DZIAŁ - PRACOWNIK lub DZ-PRAC). Przyjęcie takiej zasady ujednolica nazewnictwo i ułatwia orientację w bazie danych.

Ważne na tym etapie jest zdanie sobie sprawy z typu grupy logicznej. Zależy to oczywiście od charakteru związku zachodzącego między rekordami. Jeżeli nie wszystkie rekordy typu członek grupy będą podporządkowane właścicielom (tzn. jeżeli rekordy te będą mogły pozostawać poza strukturą grupy), grupę taką należy zdefiniować jako manualną. Przykładem może być sztucznie zdefiniowana grupa zbierająca zamówienia gotowe do realizacji. Tylko niektóre zamówienia charakteryzują się tym stanem obróbki, który umożliwia wysłanie potwierdzenia zamówienia. Większość zamówień jest w stanie właściwej realizacji lub dopiero na etapie obróbki wstępnej (dobór technologii, przydział materiału itd.). Znajdują się one poza strukturą, więc ww. grupa musi być zdefiniowana jako manualna. Inny przypadek będzie zachodził wtedy, gdy liczba rekordów właścicieli będzie większa od liczby rekordów - członków grupy. Wtedy to niektóre grupy będą puste, tzn.

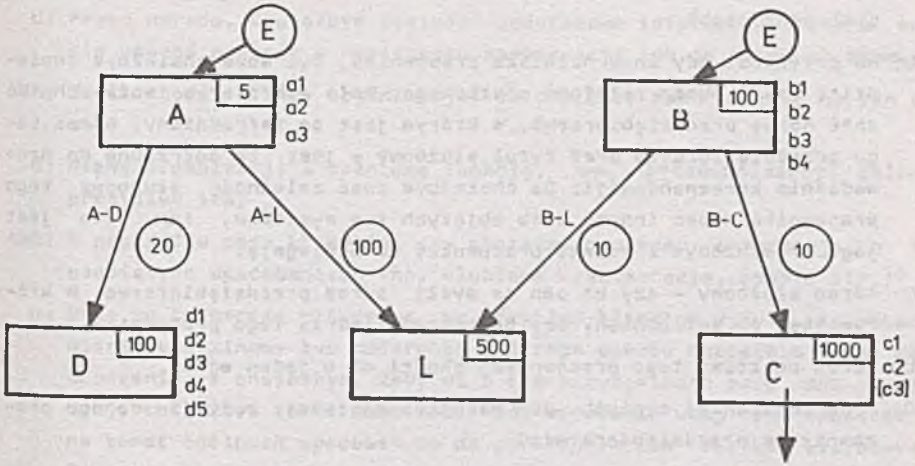
właściciel nie będzie miał przyporządkowanych żadnych rekordów członkowskich. Grupa taka może być zdefiniowana zarówno jako automatyczna, jak i manualna. Rozwiązanie takie może być nieopłacalne, gdy liczba właścicieli pustych grup będzie duża. Wtedy to w każdym takim rekordzie-właścicielu tracimy co najmniej jedno słowo na wskaźnik w grupie, który i tak nie jest wykorzystany.

Mając zdefiniowane rekordy i grupy logiczne można przystąpić do szkicowania wstępnego projektu struktury logicznej bazy. Proponuje się, aby rekordy na szkicu były przedstawione w postaci prostokątnych konturów, we wnętrzu których będzie nazwa rekordu. Nazwy pól elementarnych tworzących rekord można zaznaczać obok prostokąta. Dostęp bezpośredni do rekordu, o ile taki jest potrzebny, zaznaczamy tak samo jak dostęp bezpośredni do pól (patrz poprzednie szkice). Będzie to więc strzałka, obok której można zaznaczyć nazwę pola (pól) kluczowego, za pomocą którego ten dostęp będzie realizowany. Rekord zmiennej długości zaznaczamy strzałką wychodzącą z dolnej części prostokąta. Grupę logiczną zaznaczamy za pomocą strzałki łączącej dwa (lub więcej) rekordy. Strzałka taka biegnie od rekordu właściciela do rekordu zdefiniowanego jako członek grupy. Oczywiście rekordy mogą pozostawać w stosunku do siebie w różnych relacjach. Ten sam rekord może być zdefiniowany jako właściciel i członek, zresztą wielokrotny w różnych grupach. Rekordy tworzą więc pełną strukturę sieciową.

Do zakończenia tego etapu i w ogóle wstępnego projektu struktury logicznej należy jeszcze na szkicu zaznaczyć poznane uwarunkowania liczbowe danych. Mamy tutaj na myśli liczebność rekordów poszczególnych typów w bazie oraz liczby wystąpień rekordów członkowskich w każdej grupie logicznej. Pierwsza z tych liczb jest właściwie znana już z pierwszych wywiadów z użytkownikiem. Należy ją jeszcze sprawdzić i ewentualnie uaktualnić, np. przez przyjęcie pewnego procentowego zapasu na przyszłość. Ze średnią liczbą wystąpień rekordów w grupie może być trochę trudniejsza sprawa. W niektórych przypadkach można ją wziąć jako iloraz liczby rekordów członkowskich i liczby rekordów - właścicieli grupy. Otrzymany wynik należy jednak skonfrontować z użytkownikiem. Użytkownik może znać dokładniej rozkład liczby wystąpień rekordów członkowskich. Często zdarza się, że duża liczba rekordów typu właściciel nie posiada w ogóle rekordów członkowskich, tworząc tzw. grupy puste. Wtedy średnia liczebność pozostałych niepustych grup będzie wyższa niż uzyskany uprzednio iloraz. Tak czy inaczej, powinniśmy określić średnią liczbę rekordów w grupie. Pozwoli nas ona w późniejszych etapach na wykonanie szeregu obliczeń oceniających zaprojektowaną strukturę (np. obliczenie liczby logicznych dostępów do grupy). Może również przyczynić się do wyboru parametrów fizycznych bazy, np. wielkości strony. Zgodnie ze strategią najbliższego miejsca rekordy członkowskie umieszczone są tak blisko rekordu właściciela, jak jest to możliwe. Mając wielkości rekordów (które znamy, bo wiadomo jak duże są

tworzące je pola) i średnią liczbę rekordów w grupie, można tak dobrać wielkość strony, aby pomieściła wszystkie rekordy członkowskie jednej grupy.

Maksymalną liczbę rekordów w bazie i średnią liczbę rekordów w grupie proponuje się nanosić na szkic wstępnej struktury w następujący sposób. Liczba w kółku narysowanym przy strzałce obrazującej grupę oznacza średnią liczbę rekordów w grupie. Z kolei maksymalną liczbę wystąpień rekordów w bazie zaznaczamy w prostokącie wyciętym w konturze oznaczającym określony typ rekordu. Rys. 4.15 przedstawia przykładowy projekt wstępnej struktury logicznej tak dobrany, aby pokazać wszystkie omówione powyżej zasady szkicowania.



Rys. 4.15. Projekt wstępnej struktury logicznej - przykład oznaczeń

4.6. Przykład

Pokażmy na przykładzie, w jaki sposób stosuje się poznane w rozdziale 4 zasady projektowania. Niniejszy przykład został wzięty z kursu projektowania baz danych zorganizowanego przez firmę SPERRY UNIVAC w ośrodku szkoleniowym firmy w Birmingham w 1978 r.

Załóżmy zgodnie z tym przykładem, że zleceniodawcą i przyszłym użytkownikiem bazy danych jest dyrektor jakiegoś przedsiębiorstwa. Celem projektu jest opracowanie bazy danych dla informatycznego systemu wspomagającego pracę tegoż dyrektora. System ten można z grubsza podzielić na 3

obszary. Pierwszy z nich to zbieranie informacji o różnych osobach, pracujących bądź w tym przedsiębiorstwie bądź w przedsiębiorstwach obcych. Drugi podsystem ma na celu zbieranie informacji o przedsiębiorstwach i ich wzajemnym powiązaniu. Wreszcie trzeci moduł to zbieranie informacji o odbytych i zaplanowanych spotkaniach i konferencjach, ich uczestnikach, tematyce i rezultatach.

Projektowanie bazy rozpoczyna się od zbierania informacji od użytkownika. Pokażmy więc i tutaj przykład takiej rozmowy-wywiadu przeprowadzonej między projektantem bazy a zainteresowanym dyrektorem. Jak zauważymy, jest to jedynie fragment rozmowy dotyczący pierwszego podsystemu. Oto on:

ABD: Jakie informacje chce pan uzyskać od tego systemu?

U: Chciałbym otrzymać informacje zawodowe i prywatne o interesujących mnie osobach, a także tych, którzy potencjalnie mogą mnie interesować. Osoby te mogą być zatrudnione w moim przedsiębiorstwie lub w innych.

ABD: Na podstawie czego chciałby pan otrzymać dane na temat interesujących pana osób?

U: Na przykład, gdy znam nazwisko pracownika, być może chciałbym dowiedzieć się o numer telefonu służbowego. Moja sekretarka może chcieć znać nazwę przedsiębiorstwa, w którym jest on zatrudniony, adres tego przedsiębiorstwa oraz tytuł służbowy - jest to potrzebne do prowadzenia korespondencji. Ja chciałbym znać zależność służbową tego pracownika wobec innych osób objętych tym systemem, tzn. kto jest jego przełożonym i którzy pracownicy mu podlegają.

ABD: Adres służbowy - czy ma pan na myśli adres przedsiębiorstwa, w którym jest on zatrudniony czy bezpośredni adres tego pracownika.

U: Adres pocztowy tego pracownika, chodzi mi o jeden adres.

ABD: A co dokładniej chciałby pan wiedzieć na temat pozycji danego pracownika w przedsiębiorstwie?

U: Na pewno tytuł służbowy. Także opis zakresu jego obowiązków w przedsiębiorstwie. Informacje te będą mi potrzebne przed ewentualną naradą lub przeprowadzeniem rozmowy telefonicznej w celu uzyskania lepszego kontaktu z tą osobą.

ABD: Co dokładniej ma zawierać ten opis?

U: Różne dane. W niektórych przypadkach będzie mi wystarczało wiedzieć, że ta osoba jest klientem, w innych przypadkach będę chciał bardziej szczegółowych informacji.

ABD: Czy jedna osoba może być zatrudniona w więcej niż w jednym przedsiębiorstwie?

U: Nie, przynajmniej nie te osoby, które mnie interesują.

ABD: A jeśli chodzi o pracowników w pana przedsiębiorstwie, jakie informacje będą panu potrzebne?

U: Te same. Nie chcę ich wyróżniać w żaden sposób.

ABD: Jeśli chodzi o informacje, które określił pan jako "prywatne", co chciałby pan wiedzieć?

U: Chciałbym wiedzieć, kim ta osoba dla mnie jest (tzn. znajomym, klientem czy dostawcą).

ABD: Ile kategorii osób pan rozróżnia?

U: O ile pamiętam 10-15.

ABD: Czy chciałby pan przechowywać informacje na temat przynależności do organizacji społecznych i politycznych lub religijnych?

U: Oczywiście, tak! Przynależność do klubów, miejscowych i ogólnokrajowych, jeśli pracownicy ci zajmują się polityką.

ABD: Kiedy będzie pan potrzebował tych informacji?

U: Przed naradą. Chciałbym posiadać podstawowe informacje, które mogą się okazać pomocne w nawiązaniu konwersacji lub do celów służbowych.

ABD: Jeśli chodzi o inne organizacje, co chciałby pan wiedzieć na ten temat?

U: Nazwę organizacji i pełnioną funkcję, np. przewodniczący, członek prezydium itp.

ABD: W przypadku pana klientów, czy chciałby pan znać informacje o ich osobistych upodobaniach (np. ulubione restauracje, napoje itp.)?

U: Było by to bardzo przydatne. Na przykład klientom wysyła się pozdrowienia urodzinowe lub podarunki - z tego powodu chciałbym znać daty urodzenia. I chciałbym, żeby mi o tym przypominano parę dni wcześniej. Chciałbym także mieć możliwość wywołania uwag lub spostrzeżeń na temat ogólnych upodobań co do podarunków lub spotkań służbowych. Często, ale nie we wszystkich przypadkach kontaktuję się z ludźmi w domu, ze znajomymi, dobrymi klientami i z tego powodu będę potrzebował adres domowy i numer telefonu - dla około 50% osób.

ABD: Jak często te informacje będą zmieniane lub uzupełniane?

U: Dane te są raczej stałe. Czasami pracownik przenosi się do innego przedsiębiorstwa i wtedy będę zmieniał te dane, ale to zdarza się nieczęsto.

ABD: Wspomniał pan o wysyłaniu kart urodzinowych, tzn. chciałby pan, we właściwym czasie otrzymać wykaz osób, do których ma pan wysłać życzenia?

U: Tak, dokładnie o to mi chodzi.

Załącznik 4.16

"FUNKCJE"

Nr funkcji	Nazwa lub opis funkcji	Wejście	Wyjście	Pokrewieństwo	Częstotliwość
1	Podaj nr telefonu pracownika	NAZ-PRAC	NR-TEL-SLUZ	1:W	24%
2	Podaj dane o pracowniku	NAZ-PRAC	NAZ-ZAK ADR-SLUZ STANOWISKO	1:1 1:1 1:1	26%
3	Podaj przełożonych danego pracownika	NAZ-PRAC	NAZ-PRZEL-PRAC	1:W	3%
4	Podaj podwładnych danego pracownika	NAZ-PRAC	NAZ-POD-PRAC	1:W	3%
5	Podaj opis odpowiedzialności służbowej pracownika	NAZ-PRAC	STANOWISKO ZAKR-ODP	1:1 1:1	3%
6	Podaj typ zależności danego pracownika od nas	NAZ-PRAC	TYP-PRAC	1:1	0,43%
7	Podaj organizacje do których należy dany pracownik	NAZ-PRAC	NAZ-ORG FUN-ORG	1:W	1%
8	Wskaż, którzy pracownicy obchodzą urodziny w danym dniu	DATA-UR	NAZ-PRAC	1:W	1%
9	Podaj upodobania danego pracownika	NAZ-PRAC	UPODOBANIA	1:W	3%
10	Podaj adres i telefon domowy pracownika	NAZ-PRAC	ADR-DOM NR-TEL-DOM	1:1 1:1	0,43%
11	Podaj adres i telefon danego zakładu	NAZ-ZAK	ADR-ZAK NR-TEL-ZAK	1:1 1:1	7%
12	Podaj poznanych pracowników z danego zakładu	NAZ-ZAK	NAZ-PRAC STANOWISKO NR-TEL-SLUZ	1:W	3%
13	Podaj przedsiębiorstwa danej grupy	KOD-ZAK	NAZ-ZAK NR-TEL-ZAK TYP-ZAK	1:W	1%
14	Podaj zakłady mające specyficzny stosunek do nas i pochodzącą z określonej grupy przedsiębiorstw	KOD-ZAK TYP-ZAK	NAZ-ZAK NR-TEL-ZAK	1:W	0,43%
15	Podaj wszystkie działy danego przedsiębiorstwa	NAZ-ZAK	POD-NAZ-ZAK POD-ADR-ZAK POD-TEL-ZAK	1:W	0,43%
16	Podaj zakład nadrzędny w stosunku do danego	NAZ-ZAK	NAD-NAZ-ZAK NAD-ADR-ZAK NAD-TEL-ZAK	1:1	0.43%
17	Podaj rezultaty danego spotkania	DATA-SPOT	MIEJSCE-SPORT CEL-SPOT REZULT-SPOT	1:1 1:1 1:1	9%
18	Podaj wszystkie spotkania z danym zakładem w podanym zakresie czasu	NAZ-ZAK a) data-pocz. i data-Kon nie są danymi z bazy	DATA-SPOT MIEJSCE-SPOT CEL-SPOT TYP-SPOT	1:W	0.43%
19	Podaj wszystkie spotkania z daną osobą w podanym zakresie czasu	NAZ-PRAC a) zakres czasu jw.	DATA-SPOT MIEJSCE-SPOT CEL-SPOT TYP-SPOT	1:W	4%
20	Podaj wszystkie spotkania określonego typu w podanym zakresie czasu	TYP-SPOT a) zakres czasu jw.	DATA-SPOT MIEJSCE-SPOT CEL-SPOT NAZ-PRAC NAZ-ZAK	1:W	0,43%
21	Podaj spotkania z danego dnia/tygodnia	a) data czy zakres jw.	DATA-SPOT MIEJSCE-SPOT CEL-SPOT TYP-SPOT NAZ-PRAC NAZ-ZAK	1:W	1%
22	Wstaw informacje o nowym spotkaniu	DATA-SPOT	jw.	-	4%
23	Usuń informacje o spotkaniu	DATA-SPOT	-	-	4%

Rys. 4.16. Formularz FUNKCJE dla przykładu 4.6

Analiza ww. wywiadu pozwoli nam na przynajmniej częściowe wypełnienie formularza FUNKCJE. Przede wszystkim należy dokładnie zdefiniować funkcje użytkowe, które baza danych będzie realizować. Spróbujmy określić zadania pierwszego podsystemu na podstawie cytowanej rozmowy. Oto one:

- dostarczanie numerów telefonicznych konkretnych osób,
- dostarczanie innych danych o pracowniku (stanowisko, zakład, adres służbowy),
- pokazanie przełożonych danego pracownika,
- pokazanie podwładnych (tych zapamiętanych w bazie) danego pracownika,
- dostarczenie opisu odpowiedzialności służbowej pracownika,
- pokazanie typu stosunku danego pracownika do nas (klient, wykonawca, sprzedawca itp.),
- wylistowanie stowarzyszeń i organizacji, do których należy dany pracownik,
- informowanie o urodzinach pracowników w danym dniu,
- podanie upodobań pracownika,
- dostarczenie adresu i telefonu domowego pracownika.

Przedstawmy te funkcje na odpowiednim formularzu zaznaczając dla nich dane wejściowe (WEJŚCIE), dane wyjściowe (WYJŚCIE), typ zachodzącej między nimi zależności i częstotliwość używania danej funkcji. Na formularzu zaznaczamy częstotliwość już wyrażoną w procentach, obrazujących udział danej funkcji w obciążeniu całego systemu. Oczywiście liczby te nie zostały wzięte z cytowanego wcześniej wywiadu, zostały one uzgodnione później.

Załóżmy dalej, że podobne rozmowy pozwoliły na sprecyzowanie założeń obu pozostałych podsystemów. Wyniki tych rozmów przedstawiamy również na formularzach FUNKCJE. W rezultacie mamy zdefiniowane i opisane wszystkie żądania użytkownika pod adresem systemu. Wypełnione formularze przedstawiamy powyżej (zał. 4.16).

Zauważmy, że w funkcjach o numerach 18-21 danymi wejściowymi są daty (czy zakres dat), które jako takie nie są danymi przechowywanymi w bazie. Na formularzu FUNKCJE zaznaczamy je symbolem *. Dane te pozwolą nam na wybranie właściwych informacji z bazy. Zorganizowanie danych pod kątem przyszłego sprawnego wyboru potrzebnych informacji (w tym przypadku powiązania niektórych dni roku z konkretnymi osobami) jest właściwie celem pracy projektanta bazy.

Kolejnym etapem pracy projektanta bazy jest analiza i konsolidacja danych. W prostszych przypadkach można je wykonywać rozpoczynając od każdej pojedynczej funkcji systemu. W naszym przykładzie z uwagi na dużą liczbę typów transakcji prościej i szybciej będzie rozpocząć konsolidację na poziomie podsystemów. Po prostu na wspólnym rysunku szkicujemy dane relizujące wszystkie funkcje podsystemu. Oczywiście na tym etapie dokonujemy już pewnych uproszczeń, np. zaznaczając tylko raz pole występujące w kilku funkcjach (jak nazwisko pracownika czy nazwa zakładu). Kolejne rysunki

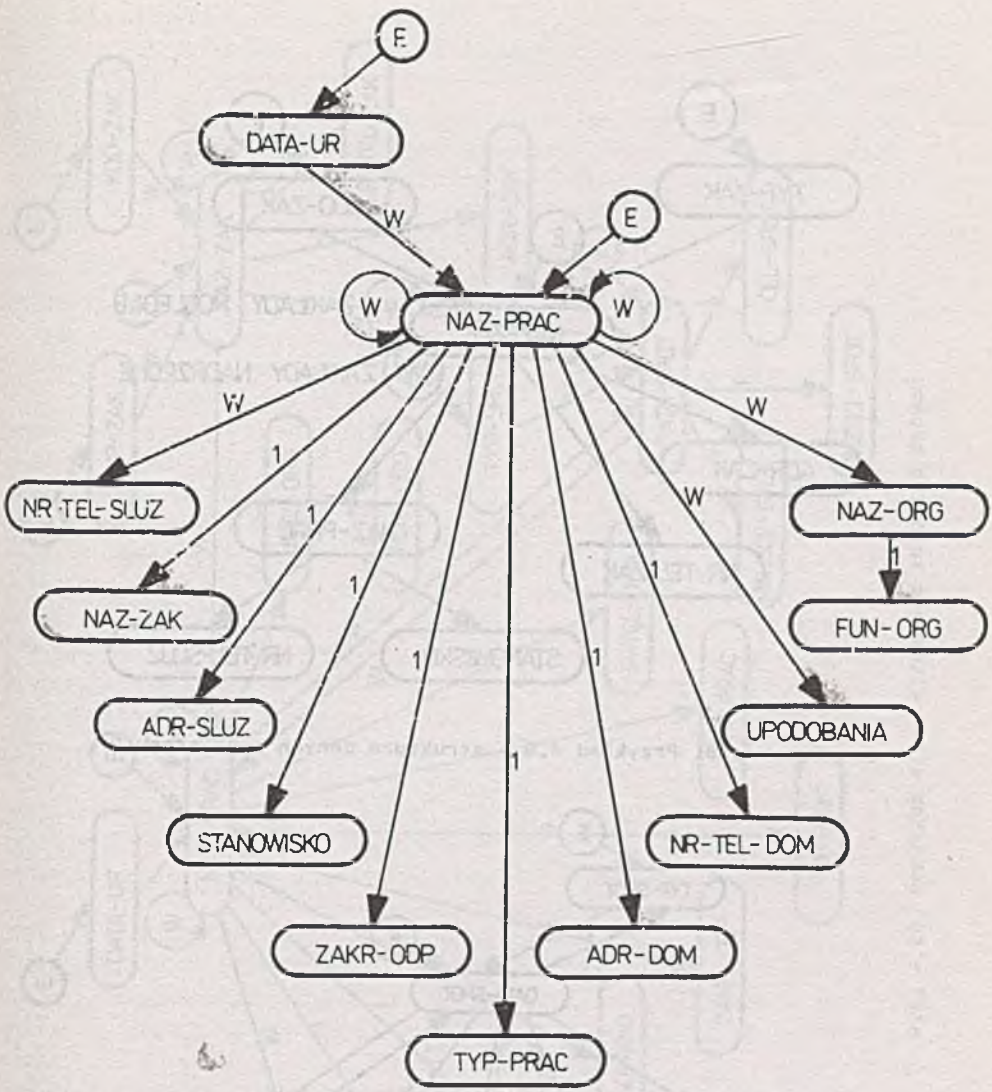
4.17, 4.18 i 4.19 przedstawiają struktury danych realizujące poszczególne podsystemy.

Celem lepszej orientacji opiszmy w tym miejscu wszystkie występujące w systemie dane oraz nazwy tych danych w bazie. Nazwy te będą pojawiać się zarówno w tekście tego rozdziału, jak i na odpowiednich rysunkach.

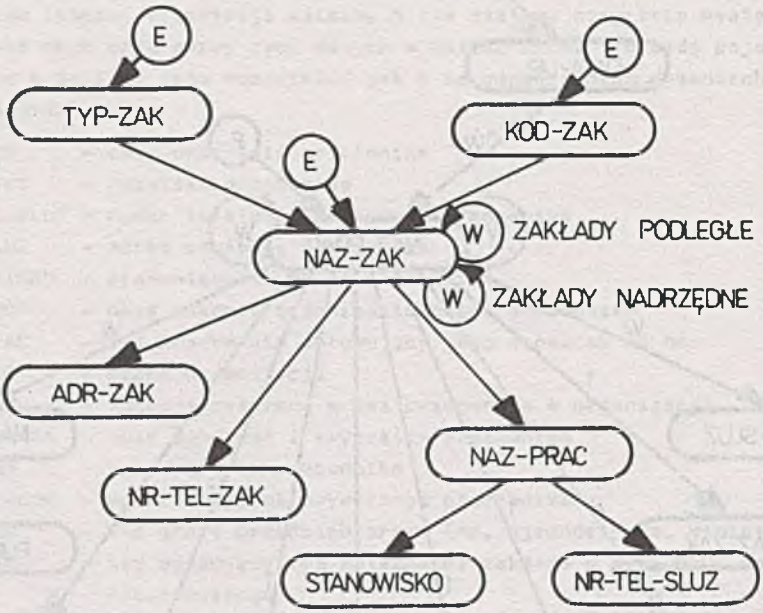
Oto one:

DATA-UR - data urodzenia pracownika
NAZ-PRAC - nazwisko pracownika
NR-TEL-SLUZ - numer telefonu służbowego pracownika
ADR-SLUZ - adres służbowy pracownika
STANOWISKO - stanowisko służbowe pracownika
ZAKR-ODP - opis zakresu odpowiedzialności pracownika
TYP-PRAC - kod pracownika obrazujący jego stosunek do nas
NAZ-ORG - nazwa organizacji
FUN-ORG - funkcja pełniona przez pracownika w organizacji
UPODOBANIA - opis upodobań i zwyczajów pracownika
ADR-DOM - adres domowy pracownika
NR-TEL-DOM - numer telefonu prywatnego pracownika
KOD-ZAK - kod grupy przedsiębiorstw (np. zjednoczenia, ministerstwa)
TYP-ZAK - kod opisujący typ zależności zakładu w stosunku do zakładu macierzystego
NAZ-ZAK - nazwa zakładu pracy
ADR-ZAK - adres zakładu pracy
NR-TEL-ZAK - numer telefonu zakładu (centrali)
TYP-SPOT - kod typu spotkania (np. konferencja, bankiet)
DATA-SPOT - data i godzina spotkania
MIEJSCE-SPOT - miejsce spotkania
CEL-SPOT - opis celu spotkania
REZULT-SPOT - opis wyników spotkania (o ile takie są).

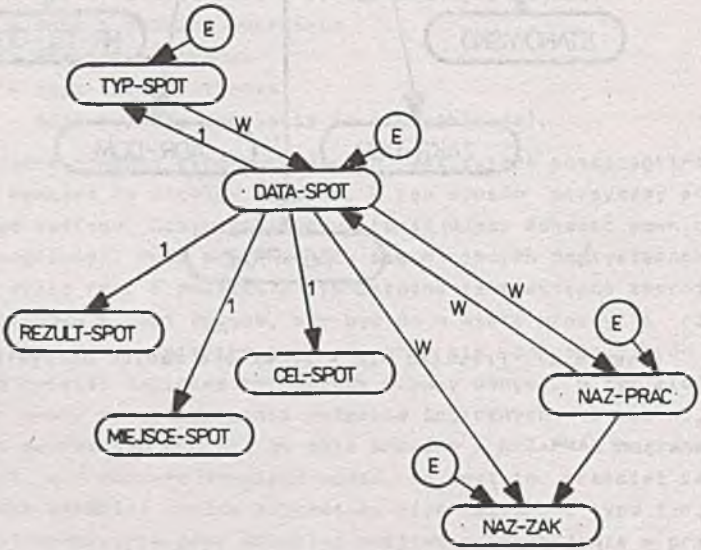
Przedstawione powyżej struktury danych realizujące poszczególne podsystemy można nanieść na wspólny rysunek. W ten sposób otrzymamy strukturę danych całego systemu. Oczywiście przy okazji należy dokonać pewnych uproszczeń czy uogólnień. Pola powtarzające się w różnych podsystemach musimy naszkicować tylko raz. W związku z tym istotne jest wstępne zaprojektowanie samego rysunku w taki sposób, aby był on w miarę prosty i czytelny. Szczególnie ważne jest uniknięcie przecinania się poszczególnych linii, obrazujących związki logiczne zachodzące między danymi. W tym miejscu należy zwrócić uwagę na upraszczanie związków logicznych. I tak np. z analizy obszaru pierwszego wynika, że pola NAZ-ZAK i NAZ-PRAC związane są zależnością 1:1, a z obszaru drugiego widać, że jest to przecież zależność typu 1:W. Jako bardziej ogólną zostawiamy więc zależność typu 1:W. W tak zdefiniowanej strukturze jest przecież możliwe przetwarzanie w przeciwnym kierunku, tj. mając nazwisko pracownika można znaleźć jego zakład pracy.



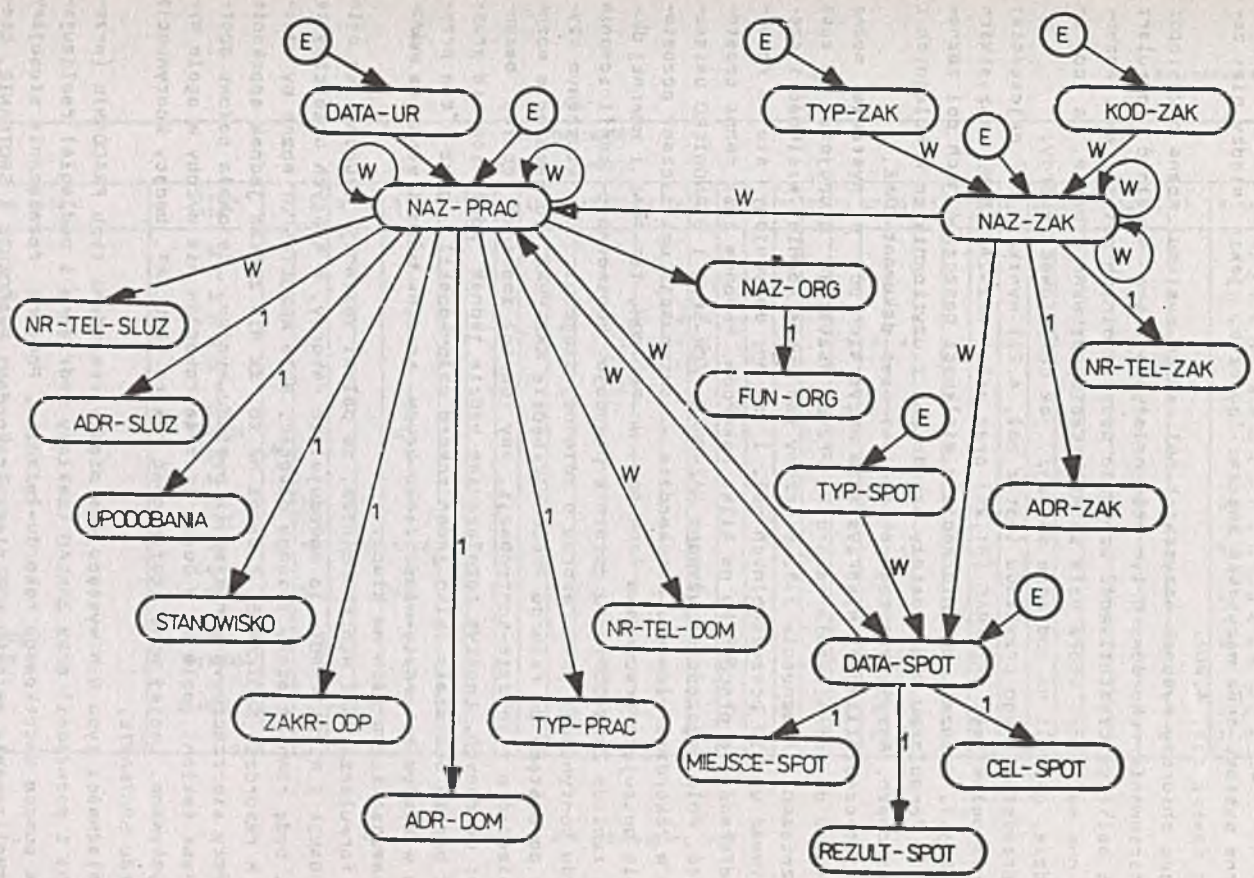
Rys. 4.17. Przykład 4.6 - struktura danych podsystemu 1



Rys. 4.18. Przykład 4.6 - struktura danych podsystemu 2



Rys. 4.19. Przykład 4.6 - struktura danych podsystemu 3



Rys. 4.20. Przykład 4.6 - struktura zbiorcza danych

Rezultatem tego etapu jest zbiorczy rysunek pokazujący wszystkie pola elementarne systemu oraz wszystkie związki logiczne, jakie między nimi zachodzą (patrz rys. 4.20).

Mając zbiorczy rysunek wszystkich pól całego systemu można przystąpić do definiowania rekordów. W tym celu należy najpierw wypełnić formularz DANE i obliczyć częstotliwość używania poszczególnych pól systemu. Pozwoli to nam na ocenę, które pola są najczęściej używane w bazie, a co za tym idzie, pozwoli na podjęcie decyzji, jak definiować rekordy.

Z przedstawionego przykładu (patrz zał. 4.21) wynika, że najczęściej używanym polem jest NAZ-PRAC (91,2% oraz pole NAZ-ZAK (48% wszystkich transakcji). Załóżmy, że informacje o wielkości poszczególnych pól zaznaczone na formularzu DANE zostały uzgodnione z użytkownikiem w kolejnych z nim rozmowach. Rezultatem tego etapu jest więc dokument DANE.

Znając częstotliwości wykorzystania wszystkich pól w systemie można przystąpić do definiowania rekordów. Przede wszystkim grupujemy ze sobą pola związane zależnością 1:1. Bierzemy pod uwagę również wielkość i częstotliwość użycia poszczególnych pól. I tak np. decydujemy się na rozbicie informacji o pracowniku na kilka rekordów. Powodem jest różna częstotliwość. Pola najczęściej używane NAZ-PRAC, ADR-SLUZ i STANOWISKO umieszczamy w rekordzie głównym. W rekordzie - kontynuacji umieszczamy pozostałe pola opisujące pracownika (ADR-DOM, NR-TEL-DOM, TYP-PRAC i zmiennej długości tablica ZAKR-ODP). To ostatnie powoduje konieczność zdefiniowania rekordu kontynuacji jako rekordu o zmiennej długości. Polem kluczowym używaźyn do dostępu do rekordu PRACOWNIK będzie NAZ-PRAC. To samo pole można by umieścić w rekordzie-kontynuacji, aby również zapewnić dostęp bezpośredni do danych. Lepszym rozwiązaniem będzie jednak zdefiniowanie grupy (choć będzie ona miała tylko jeden rekord członkowski), gdyż w tym przypadku w każdym rekordzie-właścicielu grupy tracimy tylko 1 słowo na wskaźnik zamiast kilku słów na klucz.

Z formularza DANE wynika również, że polami zmiennej długości są pola UPODOBANIA i REZULT-SPOT. To powoduje, że rekordy, w których umieścimy te pola, będą również miały zmienną długość. Pole REZULT-SPOT można by umieścić w rekordzie SPOTKANIE. Z uwagi na to, że nie zawsze jednak spotkanie zakończy się rzeczowym wynikiem (z wywiadu wynika, że około połowy spotkań jest takich), pole to w dużej liczbie rekordów nie byłoby w ogóle wykorzystywane. Lepiej więc zdefiniować rekord REZULTAT, będący kontynuacją rekordu SPOTKANIE.

Zależności typu W:W występujące między rekordami typu PRACOWNIK (przełożeni i podwładni) oraz ZAKLAD (zakłady nadrzędne i podległe) realizujemy za pomocą dodatkowego rekordu-łącznika. Podobne rozwiązania stosujemy do zrealizowania związku W:W między rekordami PRACOWNIK i SPOTKANIE. Ciekawa zależność zachodzi między rekordami ZAKLAD i SPOTKANIE. Z wywiadu wynika, że jest to zależność typu 1:W. Dokładniejsza analiza wskazuje jednak, że jest to właściwie zależność typu W:W. Chcąc uprościć ewentualne

"DANE"

CZESTOTLIWOŚĆ

NUMERY FUNKCJI

Nazwa pola	Wielk. pola	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	Suma
FUN-ORG	25							1	.																										1,0
DATA-UR	6								1																										1,0
NR-TEL-SLUZ	10	24											3																						27,0
ADR-ZAK	25											7			.4	.4																			7,8
NAZ-ZAK	36		26									7	3	1	.4	.4	.4		.4		.4	1	4	4											48,0
NR-TEL-ZAK	10											7		1	.4	.4	.4																		9,2
TYP-ZAK	2													1	.4																				1,4
DATA-SPOT	12																	9	.4	4	.4	1	4	4											22,8
ZAKR-ODP	254500					3																													3,0
ADR-DOM	25										.4																								0,4
NR-TEL-DOM	10										.4																								0,4
ADR-SLUZ	25		26																																26,0
TYP-PRAC	2						.4																												0,4
NAZ-PRAC	36	24	26	3	3	3	.4	1	1	3	.4		3							4	.4	1	4	4											81,2
UPODOBANIA	04500												3																						3,0
STANOWISKO	15		26			3							3																						32,0
KOD-ZAK	3													1	.4																				1,4
MIEJSCE-SPOT	25																	9	.4	4	.4	1	4	4											22,8
REZULT-SPOT	043500																	9																	9,0
NAZ-ORG	36							1	.																										1,0
CEL-SPOT	254500																	9	.4	4	.4	1	4	4											22,8
TYP-SPOT	2																		.4	4		1	4	4											13,4

Rys. 4.21. Przykład 4.6 - formularz DANE

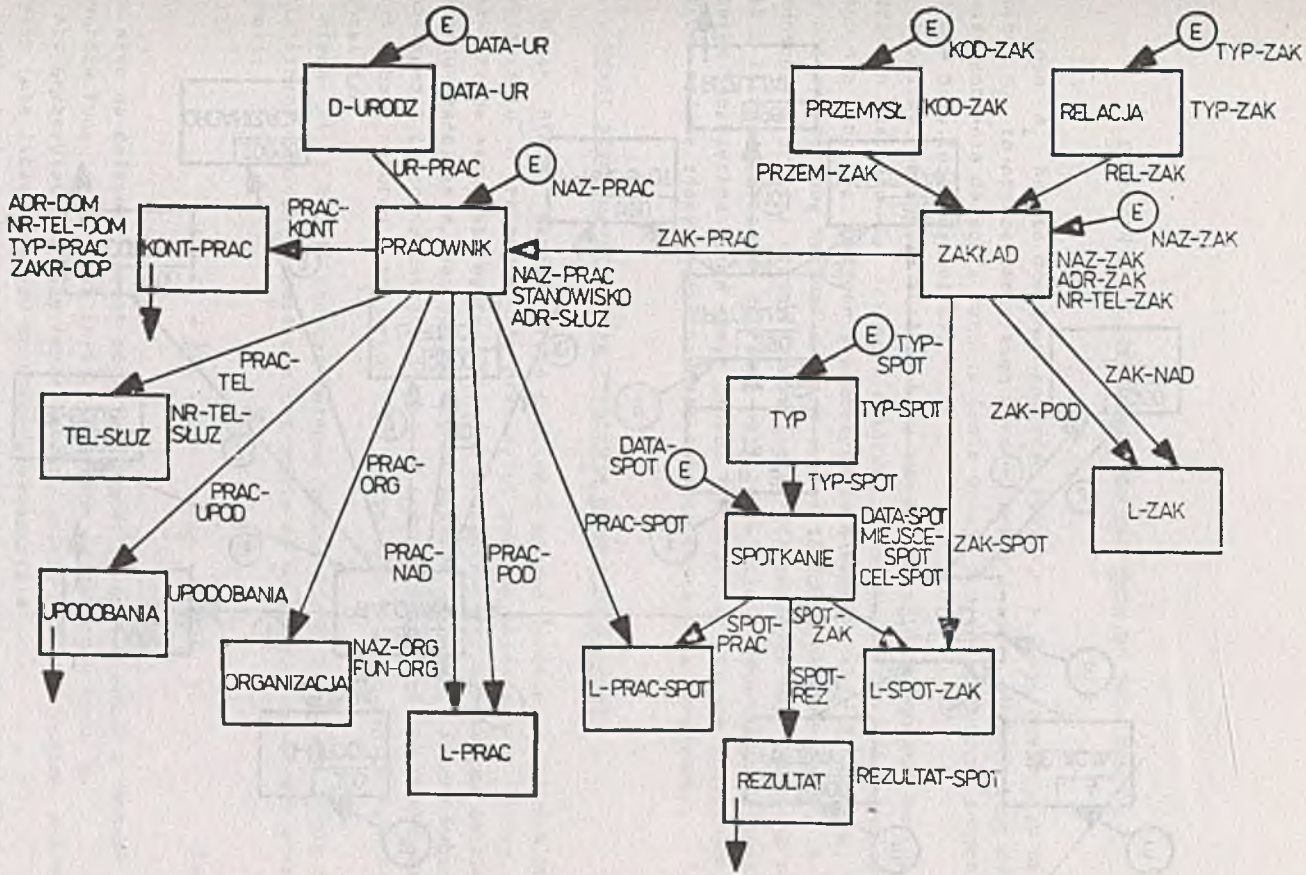
przyszłe funkcje typu: "przedstawiciele jakich zakładów brali udział w rozmowach" (którą dotychczas można by zrealizować okrężną drogą poprzez przeglądnięcie pracowników biorących udział w spotkaniu i następnie pobraniu informacji o ich zakładach pracy), lepiej od razu zdefiniować dodatkowy rekord łącznik wiążący rekordy SPOTKANIE i ZAKLAD w zależność typu W:W.

W tym miejscu pozostaje jeszcze określić typ grupy. W przypadku grupy URODZ-PRAC mamy do czynienia z grupą manualną, gdyż nie wszystkie rekordy członkowskie będą podporządkowane rekordom wyższego poziomu. Zakładamy bowiem, że nie dla wszystkich pracowników należy pamiętać ich daty urodzenia.

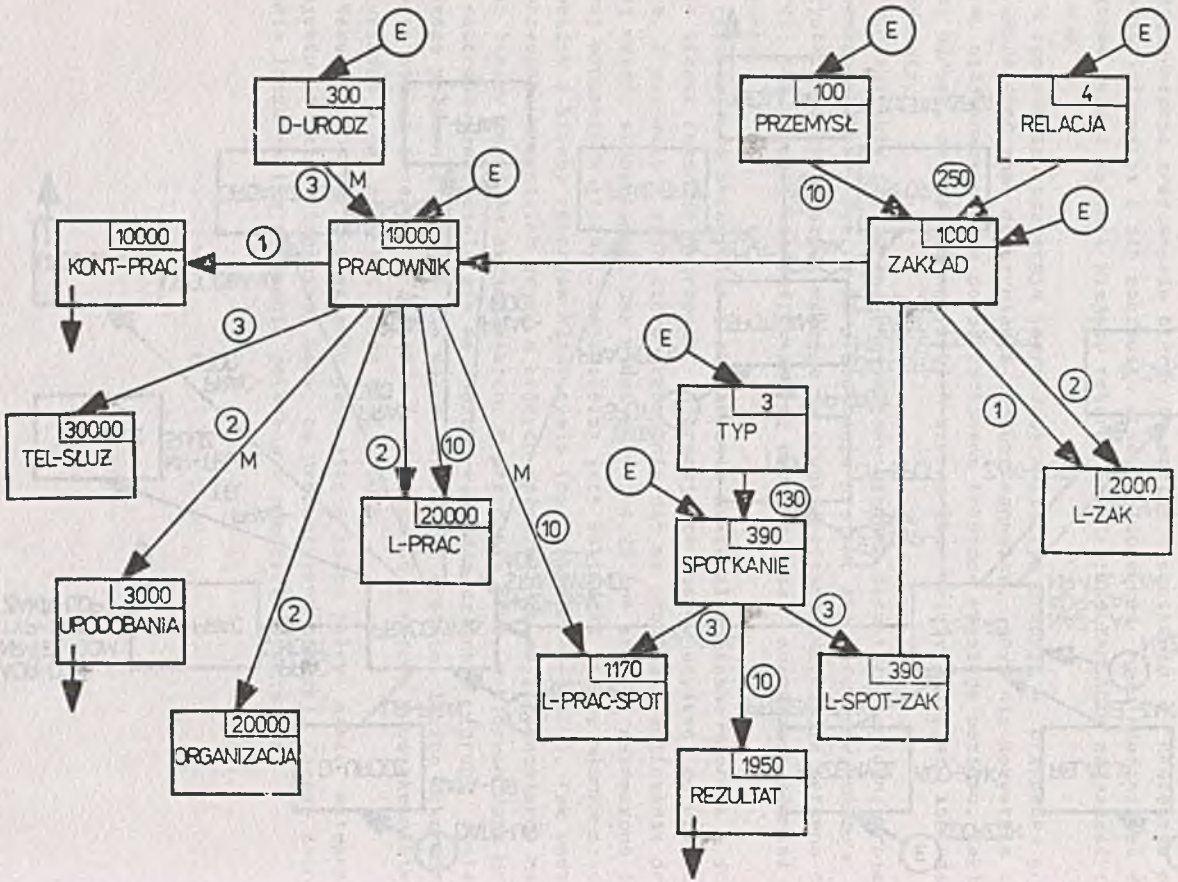
W rezultacie tego etapu otrzymujemy wstępną strukturę logiczną projektowanego systemu. Rys. 4.22 przedstawia projekt tej struktury. Pokazuje on wszystkie rekordy bazy i wiążące je grupy logiczne. Oczywiście zarówno rekordy, jak i grupy identyfikowane są poprzez nazwę zaznaczoną również na rysunku.

Pozostaje do określenia i zaznaczenia maksymalne liczba rekordów i średnia liczba rekordów w grupach. Informacje te musimy uzyskać od użytkownika. Z pierwszą z tych liczb przeważnie nie ma problemu, obliczanie drugiej należy skomentować. Jak widać z rys. 4.23 (na którym zaznaczono już wyżej wspomniane liczby), najczęściej jest to iloraz liczby rekordów członkowskich i liczby rekordów właściciela (np. grupa PRAC-TELSL). Tak będzie w przypadku wszystkich grup, o których założymy, że mają równomierny rozkład liczby rekordów członkowskich. W przypadku grup, w których duża liczba rekordów-właścicieli nie ma swoich rekordów-członków (grupy puste), takiego założenia przyjąć nie można. Wtedy też średnia liczba rekordów w grupie będzie inna, co oczywiście musimy poznać z rozmów z użytkownikiem. Przykładem u nas może być grupa PRAC-UPODOB.

Rezultatem pracy projektanta jest więc w końcu pełny szkic wstępnego projektu struktury logicznej bazy, co prezentujemy poniżej. Zasady zaznaczania liczb na tym schemacie pokazano w rozdziale poprzednim.



Rys. 4.22. Przykład 4.6 - wstępny projekt powiązań rekordów



Rys. 4.23. Przykład 4.6 - wstępny projekt struktury logicznej z zaznaczonymi charakterystykami liczbowymi bazy

5. OBLICZANIE LICZBY LOGICZNYCH DOSTĘPÓW DO BAZY DANYCH

Niech A_i będzie dowolnym rekordem obszaru X. Logiczny dostęp do rekordu A_i to przeczytanie tego rekordu przez system obsługi bazy danych celem pobrania danych z rekordu (pole danej) lub wskaźników do innych rekordów bazy (pole łącznika).

Zauważmy, że system obsługi bazy danych często musi wykonać szereg logicznych dostępów do jakiegoś rekordu, mimo że program ten bezpośrednio tego nie żąda (np. polecenie przeczytania siedemnastego rekordu danej grupy pociąga za sobą wykonanie przez system 18 logicznych dostępów, a nie jednego). Z drugiej strony liczba logicznych dostępów nie zawsze ma związek z liczbą fizycznych operacji wejścia/wyjścia. Rekordy grupy mogą być bowiem zapamiętane w tym samym bloku. Tym niemniej liczba logicznych dostępów wpływa zdecydowanie na czas trwania transakcji (programu).

5.1. Koszt odczytu/wstawienia rekordu w grupie cyklicznej

Niech S(A,B) oznacza grupę cykliczną mającą średnio w każdym swoim wystąpieniu n rekordów. Aby przeczytać dowolny rekord z tej grupy, należy przede wszystkim wybrać właściwe wystąpienie grupy poprzez ustalenie rekordu właściciela. Następnie, aby przeczytać pierwszy rekord wykonamy dodatkowo 1 logiczny dostęp, drugi - 2 itd. Załóżmy, że prawdopodobieństwo wyboru rekordu z tej grupy jest takie samo dla wszystkich rekordów i wynosi 1/n.

Niech X(1,2,3...) oznacza zmienną losową liczby dostępów. Wtedy średnia liczba logicznych dostępów potrzebna do odszukania dowolnego rekordu wynosi:

$$K = EX = \frac{1}{n}(1 + 2 + \ldots n) = \frac{n+1}{2} \qquad (5.1)$$

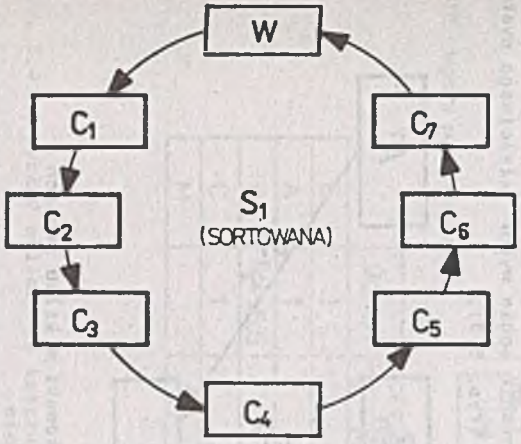
Tak więc na całkowity koszt odczytu rekordu złoży się koszt 1 dostępu dla rekordów typu A i koszt (n+1)/2 dostępów do rekordów typu B.

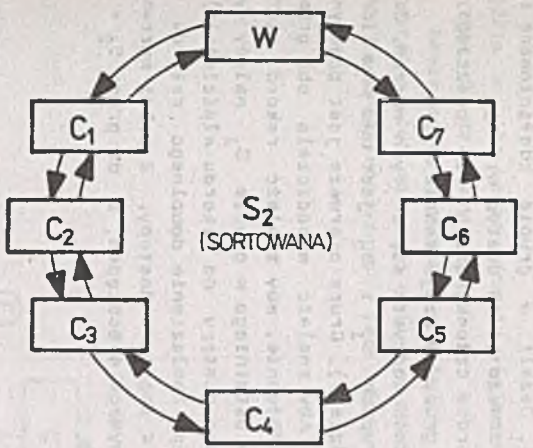
Aby przeczytać pierwszy rekord grupy, należy wykonać zawsze tylko 1 dostęp (nie licząc dostępu do rekordu właściciela).

Liczba logicznych dostępów potrzebnych do przeczytania ostatniego rekordu grupy zależy od rodzaju wskaźników używanych w danej grupie. Jeżeli mianowicie zdefiniowane są w tej grupie wskaźniki "wstecz", to wystarczy wykonać 1 dostęp, jeżeli nie - n logicznych dostępów.

Logiczny punkt wstawienia w grupie	„pierwszy"			„ostatni"			„następny/sortowany"		
Wskaźnik używany w grupie	w przód	do właści-ciela	wstecz	w przód	do właści-ciela	wstecz	w przód	do właści-ciela	wstecz
Przeczytanie żądanego rekordu z grupy	$\frac{n+1}{2}$								
Przeczytanie pierwszego rekordu z grupy	1								
Przeczytanie ostatniego rekordu z grupy	n		1	1			n		1
Zapamiętanie nowego rekordu	1		2	1		2	$\frac{n+1}{2}$		$\frac{n+3}{2}$

Rys. 5.1. Koszt odczytu i wstawienia rekordu do grupy





	GRUPA S_1	GRUPA S_2
Wybierz właściciela W	1	1
Wybierz C_1	1	1
Wybierz C_7	7	1
Wybierz M_n	$\frac{N+1}{2} = 4$	$\frac{N+1}{2} = 4$
Zapamiętaj M_n	$\frac{N+1}{2} = 4$	$\frac{N+3}{2} = 5$

Rys. 5.2. Przykład obliczania kosztu odczytu i wstawiania do grup

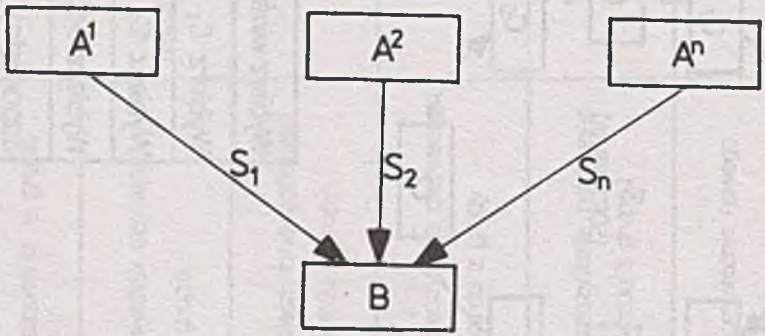
Przy zapamiętywaniu nowego rekordu należy postępować następująco: wybrać właściwe wystąpienie grupy, ustalić logiczny punkt wstawiania, zaktualizować wskaźniki rekordów sąsiednich lub tylko poprzedniego (w zależności od rodzaju wskaźników w grupie). Jeżeli w grupie zdefiniowane są wskaźniki "wstecz", to wymagany jest zawsze jeden dostęp więcej.

Liczbę logicznych dostępów do rekordów członkowskich dla poszczególnych funkcji przy różnej organizacji grupy obrazuje tabela 5.1.

Pokażmy na przykładzie, w jaki sposób korzysta się z tych wzorów. Załóżmy zgodnie z rys. 5.2, że w dwu grupach S_1 i S_2 jest taka sama liczba rekordów członkowskich (w przykładzie 7). Grupa pierwsza jest pojedynczo, a grupa druga podwójnie łączona. Aby znaleźć właściciela obu grup, wystarczy wykonać 1 dostęp logiczny. Podobnie, aby znaleźć rekord pierwszy. W przypadku znajdywania rekordu ostatniego w grupie S_1 należy wykonać 7 dostępów logicznych (nie licząc dostępu do rekordu właściciela), a w grupie S_2 wystarczy jeden dostęp. Znalezienie dowolnego rekordu z grupy S_1 i S_2 kosztuje $(7+1)/2 = 4$ logiczne dostępy. Z kolei wprowadzenie nowego rekordu do grupy S_1 wymaga 4 dostępów, a do grupy S_2 - 5 logicznych dostępów: $(7+3)/2$.

5.2. Koszt wstawiania rekordu członkowskiego do kilku grup cyklicznych

Załóżmy, że rekord typu B jest zdefiniowany jako rekord członkowski w n różnych grupach cyklicznych S1,S2...Sn. Aby zapamiętać nowy rekord typu B w tej strukturze, należy zapewnić sobie wybór właściwego wystąpienia każdej z przedstawionych grup (rys. 5.3).



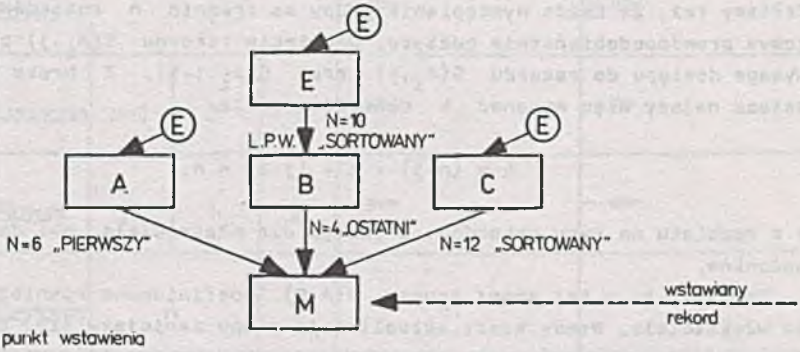
Rys. 5.3. Rekord członkowski w kilku grupach

Jest to realizowane przez wybór rekordów właściciela tych grup. Koszt całkowity zapamiętania rekordu B będzie więc równy:

$$K = \sum_{i=1}^{n} (w_i + g_i), \qquad (5.2)$$

gdzie w_i to koszt odczytu właściciela A^i, g_i to koszt wstawienia rekordu B do grupy S_i.

Koszt odczytu właściciela będzie wynosił 1, gdy rekord ma dostęp bezpośredni, lub można go policzyć metodą przedstawioną w pkt. 5.1, gdy dostęp do niego realizowany jest poprzez jakąś grupę.



L.P.W. - logiczny punkt wstawienia

	Odczyt	Wstawienie			Razem
		A - M	B - M	C - M	
E	1				1
A	1				1
B	$\frac{N+1}{2}=5.5$				5.5
C	1				1
M		1	1	$\frac{N+1}{2}=6.5$	8.5
					17

Rys. 5.4. Przykład obliczania kosztu wstawienia rekordu do struktury sieciowej

Rys. 5.4 przedstawia przykładową strukturę danych oraz tabelkę ułatwiającą obliczanie liczby logicznych dostępów do bazy. Założenie nowego

rekordu M w takiej strukturze polega na właściwym wyborze wszystkich grup logicznych A-M, B-M i C-M oraz aktualizacji tych grup. Koszt całkowity tej operacji można podzielić na koszt odczytu i koszt aktualizacji danych. Dokładniejsze wyniki przedstawia tabelka.

5.3. Koszt usunięcia rekordu członkowskiego z grupy cyklicznej

Koszt usunięcia rekordu członkowskiego z grupy cyklicznej składa się z dwóch składowych - kosztu znalezienia tego rekordu oraz kosztu aktualizacji rekordów z grupy.

Koszt znalezienia rekordu członkowskiego może wynosić 1, gdy jest do niego dostęp bezpośredni lub być obliczony zgodnie z tabelą 5.1. Usunięcie z grupy polega na aktualizacji wskaźników w rekordach sąsiednich lub tylko w rekordzie poprzednim.

Załóżmy, że w grupie S(A,B) zdefiniowane są tylko wskaźniki "w przód". Załóżmy też, że każde wystąpienie grupy ma średnio n rekordów o jednakowym prawdopodobieństwie odczytu. Usunięcia rekordu $S(A_i,j)$ z tej grupy wymaga dostępu do rekordu $S(A_i,j)$ oraz $S(A_i,j-1)$. Z braku wskaźników wstecz należy więc wykonać k dostępów, gdzie:

$$k = (n-j) + 1 + (j-1) = n, \tag{5.3}$$

a w rozbiciu na typy rekordów: 1 dostęp dla właściciela, n-1 dostępów dla członków.

Załóżmy, że w tej samej grupie S(A,B) zdefiniowano również wskaźniki do właściciela. Wtedy koszt aktualizacji grupy zmniejszy się, gdyż bezpośrednio z rekordu $S(A_i,j)$ można przejść do rekordu właściciela A_i.

Wartości zmiennej X(0,1,2...n-1) przedstawiają ilości koniecznych dostępów dla usunięcia i-tego rekordu (nie licząc dostępu do właściciela). Średnia liczba dostępów w tym przypadku wyniesie więc:

$$k = EX = \frac{1}{n}(0+1+...+n-1) = \frac{n-1}{2} \tag{5.4}$$

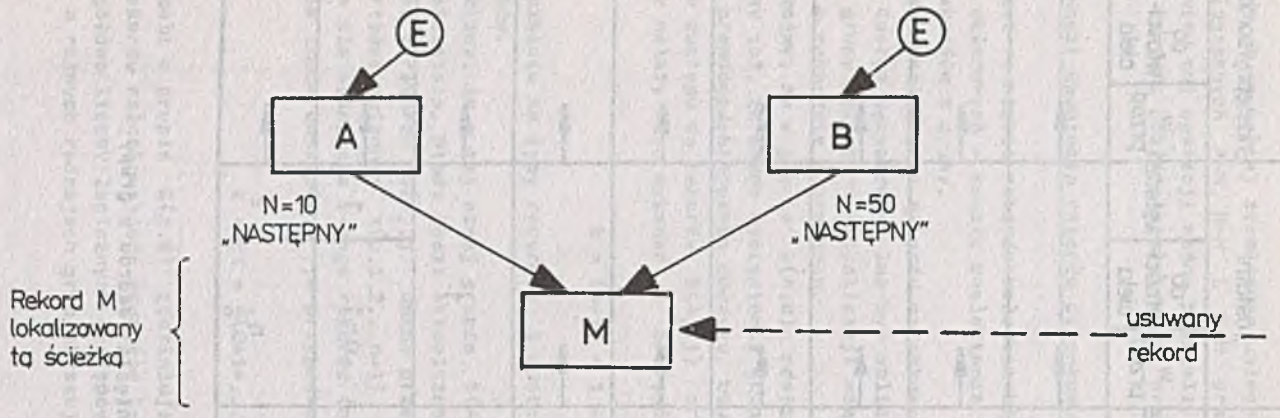
Jeżeli w grupie S(A,B) zdefiniujemy wskaźniki wstecz, to do usunięcia rekordu członkowskiego wymagane będą dokładnie 2 dostępy logiczne. Szczegółowo liczby logicznych dostępów przy usuwaniu rekordów członkowskich w różnych rodzajach grup obrazuje tabela 5.5.

5.4. Koszt usuwania rekordu członkowskiego z kilku grup cyklicznych

Załóżmy ponownie, że rekord typu B (zgodnie z rys. 5.3) występuje w n różnych grupach S1,S2,...,Sn jako rekord członkowski. Aby usunąć do-

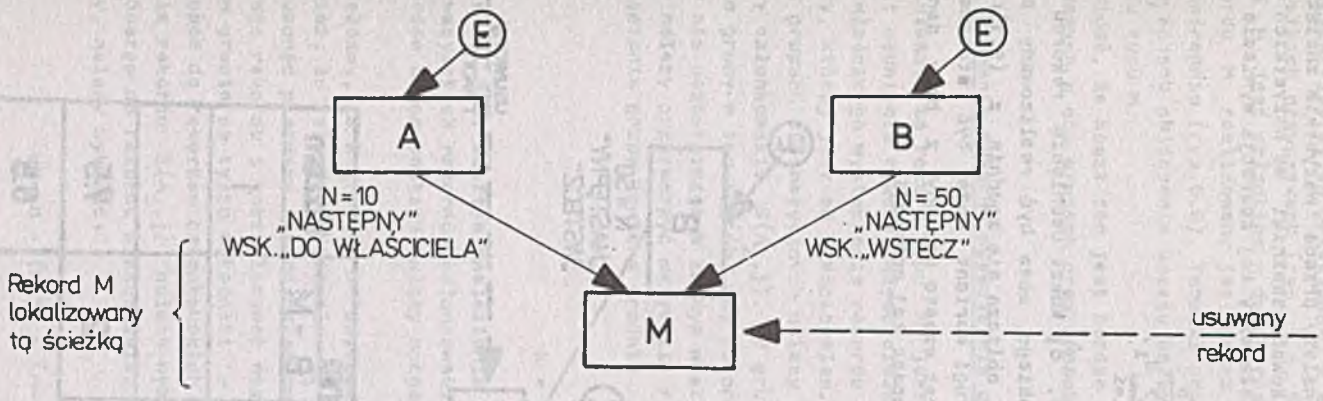
Logiczny punkt wstawienia w grupie		„pierwszy"			„ostatni"			„następny/sortowany"		
Wskaźniki używane w grupie		w przód	do właści-ciela	wstecz	w przód	do właści-ciela	wstecz	w przód	do właści-ciela	wstecz
Usuwanie zadanego rekordu z grupy	Czytanie	$\frac{n+1}{2}$			→			→		
	Usuwanie	$n-1$	$\frac{n-1}{2}$	2	→			→		
	Razem	$\frac{3n-1}{2}$	n	$\frac{n+5}{2}$	→			→		
Usuwanie pierwszego rekordu z grupy	Czytanie	1			→			→		
	Usuwanie	$n-1$	0	1	→			→		
	Razem	n	1	2	→			→		
Usuwanie ostatniego rekordu z grupy	Czytanie	n		1	1			→		
	Usuwanie	$n-1$		1	$n-1$		1	→		
	Razem	$2n-1$		2	n		2	→		

Rys. 5.5. Koszt usunięcia rekordu z grupy



	Odczyt	Usuwanie		Razem
		A - M	B - M	
A	1	1		2
B			1	1
M	$\frac{N+1}{2}=5.5$	N - 1 = 9	N - 1 = 49	63.5
				66.5

Rys. 5.6. Obliczanie kosztu usuwania rekordu ze struktury sieciowej - przykład 1



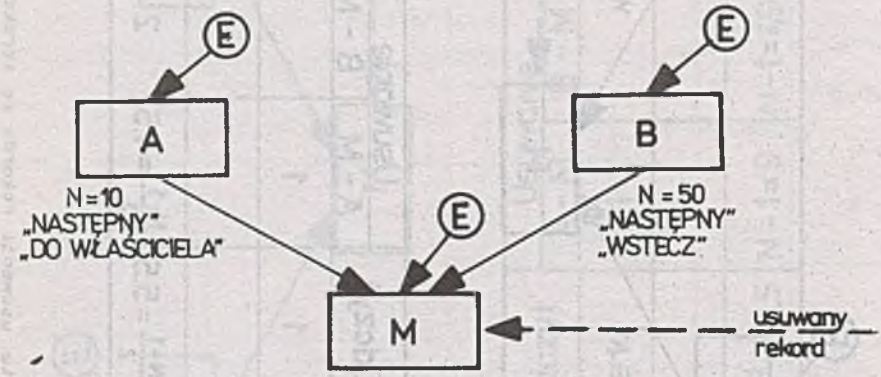
	Odczyt	Usuwanie		Razem
		A - M	B - M	
A	1	1		2
B				
M	$\frac{N+1}{2}=5.5$	$\frac{N-1}{2}=4.5$	2	12
				14

Rys. 5.7. Obliczanie kosztu usuwania rekordu ze struktury sieciowej - przykład 2

wolny rekord Bj z takiej struktury, należy przede wszystkim znaleźć go w tej strukturze, a następnie zaktualizować wskaźniki we wszystkich grupach, w których był członkiem. Koszt całkowity tej operacji wyniesie więc:

$$k = m + \sum_{i=1}^{n} d_i, \qquad (5.5)$$

gdzie m to koszt odczytu rekordu Bj, $\hat{d}_i$ koszt usunięcia rekordu Bj z grupy S_i. Dostęp do rekordu członkowskiego może być realizowany przez przeglądanie dowolnej grupy (wtedy m oblicza się zgodnie z (5.1)) lub metodą bezpośrednią (wtedy $m = 1$). Drugi wariant może być stosowany w przypadkach, gdy rekord członkowski jest często usuwany z bazy danych, gdyż zmniejsza to istotnie całkowity koszt tej operacji.



	Odczyt	Usuwanie		Razem
		A - M	B - M	
A		1		1
B				
M	1	$\frac{N-1}{2} = 4.5$	2	7.5
				8.5

Rys. 5.8. Obliczanie kosztu usuwania rekordu ze struktury sieciowej przykład 3

Rozpatrzmy kilka przykładów, w których zastosowano poznane wzory. We wszystkich przypadkach (rys. 5.6, 5.7 i 5.8) mamy podobną strukturę. Rekord M jest członkiem w dwóch różnych grupach A-M i B-M. Dostęp do rekordu M realizowany jest przez pierwszą grupę (rys. 5.6 i 5.7) lub bezpośrednio (rys.5.8). Tabelki umieszczone na kolejnych rysunkach obrazują sposób obliczania kosztu całkowitego operacji usunięcia jednego rekordu typu M.

Widać, że koszt ten jest bardzo zależny od rodzaju wskaźników zdefiniowanych w obu grupach oraz sposobu dostępu do rekordu M.

5.5. Usuwanie rekordu właściciela grupy cyklicznej

Załóżmy, że rekord typu A jest właścicielem cyklicznej grupy S(A,B). Koszt usunięcia rekordu właściciela będzie składał się z kosztu znalezienia właściwego wystąpienia rekordu A_i oraz kosztu "zlikwidowania" całej grupy, której jest on właścicielem.

W grupach automatycznych należy wtedy usunąć fizycznie wszystkie rekordy członkowskie $S(A_i,j)$. W grupach manualnych wystarczy zreorganizowanie grupy w ten sposób, aby rekordy członkowskie pozostały w bazie danych nie uczestnicząc w żadnym wystąpieniu grupy S(A,B). W obu przypadkach należy przetworzyć wszystkie rekordy grupy. Przy założeniu, że każde wystąpienie grupy zawiera średnio n rekordów, koszt tej operacji wyniesie:

$$k = w + d, \tag{5.6}$$

gdzie w jest kosztem znalezienia rekordu A_i, a d jest kosztem usunięcia wszystkich rekordów członkowskich. Liczba d logicznych dostępów do rekordów członkowskich zależy mocno od rodzaju wskaźników używanych w grupie.

Załóżmy, że w grupie S używane są tylko wskaźniki "w przód". Załóżmy również, że rekord właściciela A_i jest rekordem bieżącym (w programie). Aby usunąć pierwszy rekord członkowski $S(A_i,1)$, należy wykonać 1 dostęp do tego rekordu i zaktualizować wskaźnik w rekordzie właściciela. Ponieważ w grupie są tylko wskaźniki "w przód", należy wykonać dodatkowo "n-1" dostępów do rekordów członkowskich oraz 1 do rekordu właściciela. Do usunięcia rekordu $S(A_i,j)$ należy wykonać 1+n-j dostępów do rekordów typu B i 1 dostęp do rekordu właściciela. Aby więc usunąć wszystkie rekordy z grupy, należy wykonać:

$$d = n + (n-1) + (n-2) + \ldots + (n-n) + n$$

$$d = \frac{n(n+1)}{2} + n = \frac{n(n+3)}{2} \tag{5.7}$$

dostępów, z czego $\frac{n(n+1)}{2}$ do rekordów członkowskich i n dostępów do rekordu A_i.

Załóżmy z kolei, że w grupie S(A,B) oprócz wskaźników "w przód" zdefiniowane są również wskaźniki "do właściciela". Załóżmy też, że rekord A_i jest już rekordem bieżącym grupy. Usuwanie rekordów z grupy będzie polegało na aktualizacji wskaźników w rekordzie poprzednim, a więc będą kolejno przetwarzane rekordy:

$$S(A_i,1),\ A_i,\ S(A_i,2),\ A_i,\ldots,S(A_i,n),\ A_i.$$

Tak więc w tym przypadku należy wykonać n dostępów do rekordu właściciela A_i.

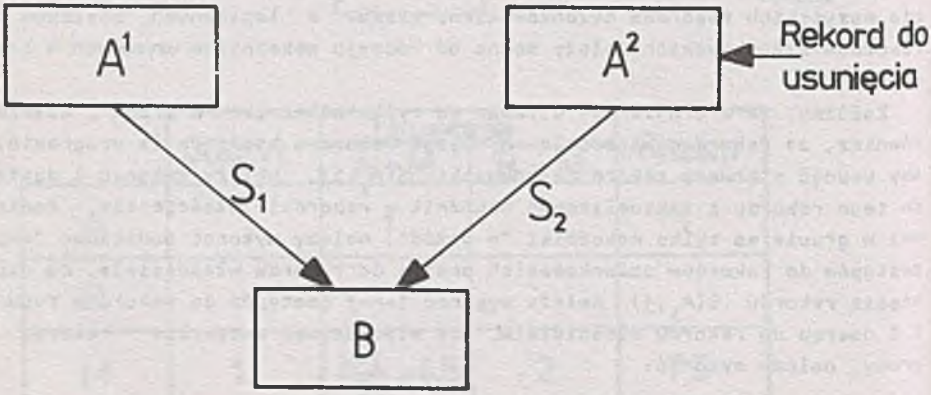
Zatem:

$$d = 2n \tag{5.8}$$

(z czego n na rekordy typu B i n na rekord A_i). Załóżmy również, że w grupie S(A,B) oprócz wskaźników "w przód" zdefiniowano wskaźniki "wstecz". Koszt usuwania wszystkich rekordów członkowskich wyniesie również 2n, gdyż w tym przypadku kolejno będą przetwarzane rekordy:

$$S(A_i,1),\ A_i,\ S(A_i,2)\ A_i,\ldots,S(A_i,n),\ A_i.$$

Wskaźnik "wstecz" kolejno usuwanych rekordów będzie zawsze wskazywał rekord właściciela grupy. W tym przypadku też należy stosować wzór (5.8).



Rys. 5.9. Usuwanie właściciela w strukturze sieciowej

Sytuację mocno skomplikuje fakt, gdy rekord typu B jest określony jako rekord członkowski w kilku różnych grupach cyklicznych. Rozpatrzmy zgo-

dnie z rys. 5.9 najprostszy taki przypadek. Rekord B jest zdefiniowany jako rekord członkowski w dwóch grupach cyklicznych:

$$S_1(A^1,B) \quad \text{i} \quad S_2(A^2,B).$$

Załóżmy, że w każdym występieniu grupy S_1 jest średnio m rekordów, a grupy S_2 - n rekordów. Celem naszym jest usunięcie rekordu A^2. Operacja ta będzie przebiegała w kilku etapach - znalezienie właściciela A_i^2, usuwanie wszystkich rekordów członkowskich z grupy $S_2(A_i^2)$, reorganizacja właściwych występień grup $S_1(A_x^1)$ zawierających usuwane rekordy $S_2(A_i^2,j)$, gdzie $A_x^1 = S_1^{-1}(S_2(A_i^2,j))$.

Koszt całkowity wyniesie więc:

$$k = w + d_1 + d_2, \tag{5.9}$$

gdzie w to koszt odczytu rekordu A_i^2, d_1 - koszt reorganizacji grupy S_1, d_2 - koszt reorganizacji grupy S_2. Liczby logicznych dostępów d_1 i d_2 zależą od rodzaju wskaźników używanych w obu grupach. Liczbę d_2 można wyznaczyć na podstawie wzorów (5.7) i (5.8) w zależności od rodzaju wskaźników używanych w grupie S_2.

Obliczmy wartość liczby d_1. Załóżmy początkowo, że w grupie S_1 używane są tylko wskaźniki "w przód". Dla każdego usuwanego rekordu $S_2(A_i^2,j)$, $j = 1,\ldots,n$ musimy zreorganizować właściwe występienie grupy $S_1(A_1,B)$. Jeżeli $S_2(A_i^2,j) = S_1(A_x^1,k)$, tzn. $A_x^1 = S_1^{-1}(S_2(A_i^2,j))$, to musimy zaktualizować wskaźnik w rekordzie:

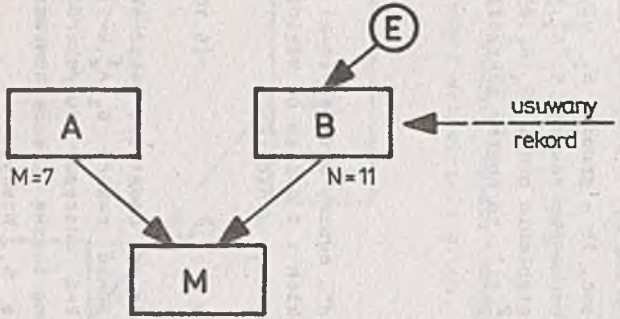
$$S_1(A_x^1,k-1).$$

Jeżeli w grupie S_1 są tylko wskaźniki "w przód", musimy wykonać $(m-k) + (k-1) = m - 1$ dostępów do rekordów członkowskich i 1 dostęp do właściciela. Tak więc łączny koszt reorganizacji grupy S_1 wyniesie:

$$d_1 = n(m-1) + n = n \cdot m \tag{5.10}$$

Załóżmy z kolei, że w grupie S_1 zdefiniowane są dodatkowo wskaźniki "do właściciela". W związku z tym, aby zaktualizować rekord $S_1(A_x^1,k-1)$, należy wykonać 1 dostęp do właściciela A_x^1 i k-1 dostępów do rekordów typu B. Niech $X(0,1,2,\ldots,m-1)$ oznacza zmienną losową liczby koniecznych dostępów do rekordów członkowskich w grupie S_1. Wtedy:

$$EX = \frac{1}{m}(0+1+2+\ldots+(m-1)) = \frac{m-1}{2}$$



a)

	Odczyt	Usuwanie		Razem
		A - M	B - M	
A		11		11
B	1		11	12
M		66[x]	66[xx]	132
				155

x) $11 \cdot (7-1) = 66$

xx) $\frac{11 \cdot (11+1)}{2} = 66$

xxx) $\frac{11 \cdot (7-1)}{2} = 33$

b)

	Odczyt	Usuwanie		Razem
		A - M	B - M	
A		11		11
B	1		11	12
M		33[xxx]	66[xx]	99
				122

c)

	Odczyt	Usuwanie		Razem
		A - M	B - M	
A		11		11
B	1		11	12
M		33[xxx]	11	44
				67

d)

	Odczyt	Usuwanie		Razem
		A - M	B - M	
A				
B	1		2	3
M		22	20	42
				45

Rys. 5.10. Przykłady obliczania kosztu usuwania właściciela w strukturze sieciowej

a) w grupie A-M wskaźniki "w przód", B-M "w przód", b) w grupie A-M wskaźniki "do właściciela", B-M "w przód", c) w grupie A-M wskaźniki "do właściciela", B-M "do właściciela", d) w grupie A-M wskaźniki "wstecz", B-M

Tak więc średnia liczba dostępów dla zreorganizowania grupy S_1 wyniesie:

$$d_1 = \frac{n(m-1)}{2} + n = \frac{n(m+1)}{2} \qquad (5.11)$$

z czego n przypada na dostęp do rekordów typu A^1.

Załóżmy teraz, że w grupie S_1 zdefiniowane są wskaźniki "wstecz". Sytuacja mocno się upraszcza. Przecież żeby zaktualizować rekord $S_1(A^1_x,k-1)$, wystarczy wykonać 1 dostęp do tego rekordu (gdyż w rekordzie $S_1(A^1_x,k)$ istnieje bezpośredni wskaźnik do niego) oraz dodatkowo 1 dostęp do rekordu $S_1(A^1_x,k+1)$, aby zaktualizować wskaźnik "wstecz" w tym rekordzie. Tak więc zawsze wymagane będą tylko 2 dostępy do rekordów członkowskich. Czyli:

$$d_1 = 2n \qquad (5.12)$$

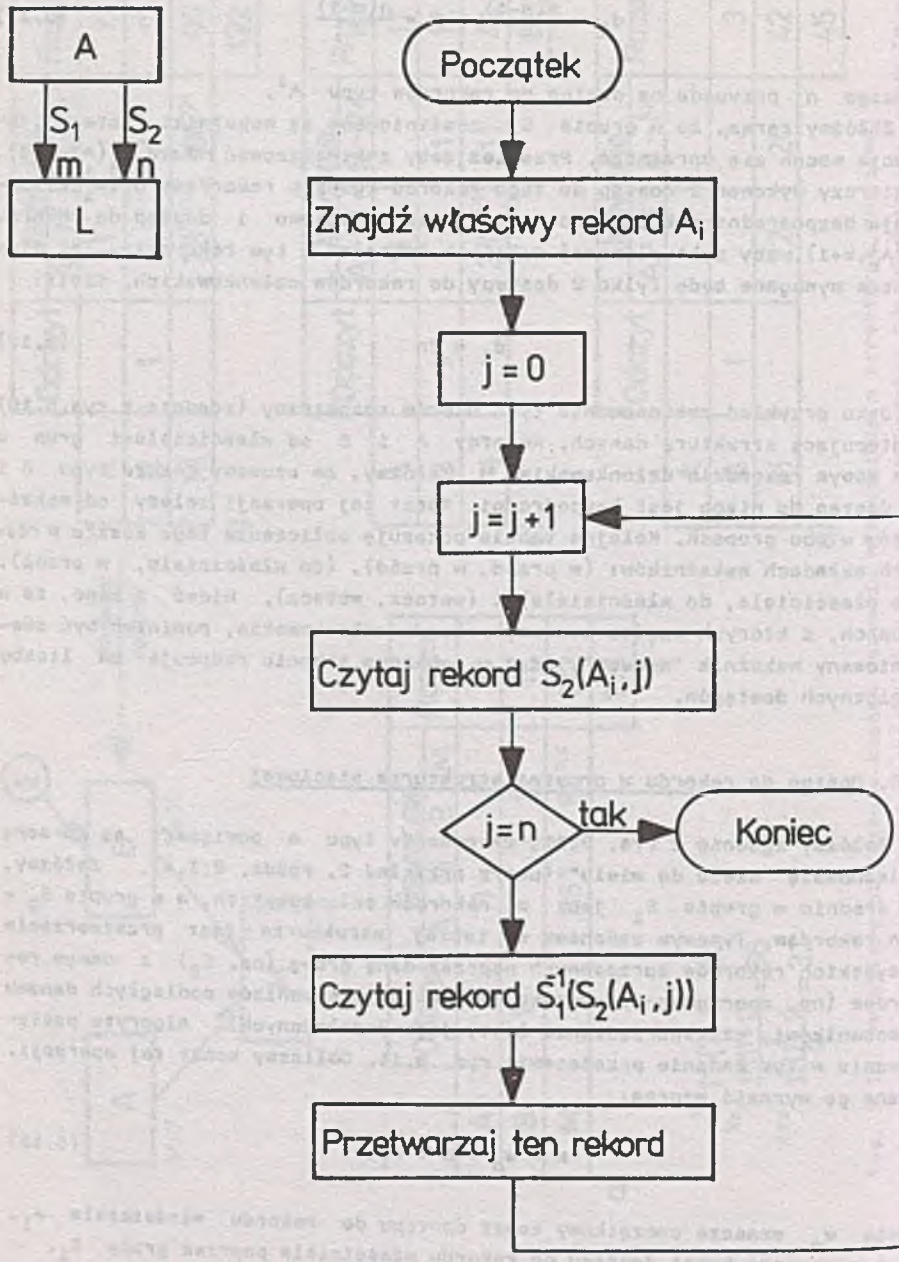
Jako przykład zastosowania tych wzorów rozpatrzmy (zgodnie z rys.5.10) następującą strukturę danych. Rekordy A i B są właścicielami grup o tym samym rekordzie członkowskim M. Załóżmy, że usuwamy rekord typu B i że dostęp do niego jest bezpośredni. Koszt tej operacji zależy od wskaźników w obu grupach. Kolejne tabele pokazują obliczenia tego kosztu w różnych układach wskaźników: (w przód, w przód), (do właściciela, w przód), (do właściciela, do właściciela) i (wstecz, wstecz). Widać z tego, że w grupach, z których często usuwamy rekordy członkowskie, powinien być zdefiniowany wskaźnik "wstecz", gdyż w poważnym stopniu redukuje on liczbę logicznych dostępów.

5.6. Dostęp do rekordu w prostej strukturze sieciowej

Załóżmy zgodnie z rys. 5.11, że rekordy typu A powiązane są ze sobą zależnością "wielu do wielu" (patrz przykład 2, rozdz. 2.3.4). Załóżmy, że średnio w grupie S_1 jest m rekordów członkowskich, a w grupie S_2 - - n rekordów. Typowym zadaniem w takiej strukturze jest przetworzenie wszystkich rekordów sprzężonych poprzez daną grupę (np. S_2) z danym rekordem (np. sporządzenie wydruku wszystkich pracowników podległych danemu pracownikowi, czy sporządzenie listy jego przełożonych). Algorytm postępowania w tym zadaniu przedstawia rys. 5.11. Obliczmy koszt tej operacji. Można go wyrazić wzorem:

$$K = w_o + n * w_1, \qquad (5.13)$$

gdzie w_o oznacza początkowy koszt dostępu do rekordu właściciela A_1, a w_1 oznacza koszt dostępu do rekordu właściciela poprzez grupę S_1.



Rys. 5.11. Algorytm przetwarzania struktury sieciowej

Liczbę w_o można wyznaczyć na podstawie (5.1) lub wynosi ona 1, gdy do rekordu A jest dostęp bezpośredni. Załóżmy, że w grupie S_1 zdefiniowane są wskaźniki "w przód" lub "w przód" i "wstecz". Problem sprowadza się do znalezienia właściciela grupy S_1 mając dany dowolny jego rekord członkowski. Ponieważ nie istnieją wskaźniki bezpośrednio do właściciela, należy kolejno czytać wszystkie rekordy L z grupy S_1. Jeżeli rekord L był pierwszym rekordem grupy S_1, należy wykonać m - 1 dodatkowych dostępów, gdy drugim - m - 2. Niech wartości zmiennej losowej X(m-1, m-2,...,0) oznaczają liczby potrzebnych dostępów dla j-tego rekordu grupy. Wtedy średnia liczba dostępów do rekordów członkowskich wyniesie:

$$EX = \frac{1}{m}((m-1) + (m-2) + \ldots + 0) = \frac{m-1}{2}$$

oraz jeden dostęp do rekordu właściciela. Ponieważ postępowanie takie należy powtórzyć dla każdego z n rekordów, łączny koszt wyniesie:

$$w_1 = n(\frac{m-1}{2}) + n = n(\frac{m+1}{2}) \tag{5.14}$$

Załóżmy teraz, że w grupie S_1 zdefiniowane są wskaźniki do właściciela. Wtedy postępowanie znacznie upraszcza się. Dla każdego z n rekordów członkowskich L wystarczy dokonać 1 dostęp bezpośrednio do właściciela grupy. Tak więc w tym przypadku będzie:

$$w_1 = n \tag{5.15}$$

Ostatni wzór w porównaniu ze wzorem (5.14) mówi, jak duże jest znaczenie właściwego zdefiniowania wskaźników w grupie w zależności od rodzaju przetwarzania.

5.7. Koszt operacji elementarnych w grupach z tablicami wskaźników

Użycie grup z tablicami wskaźników poważnie redukuje liczbę logicznych dostępów do bazy danych. Wynika to z organizacji takiej grupy, tj. z istnienia tablicy adresów wszystkich rekordów członkowskich grupy. Dostęp do rekordu członkowskiego w takiej strukturze będzie wynosił:

$$k = w + 2, \tag{5.16}$$

gdzie w jest kosztem dostępu do rekordu właściciela, a liczba 2 to 1 dostęp do tablicy adresowej grupy i drugi dostęp do szukanego rekordu członka.

Aktualizacja grupy, usunięcie rekordu członkowskiego lub wstawienie nowego, poza dostępem do rekordu członka celem wykonania odpowiedniej operacji, wymaga tylko aktualizacji tablicy adresowej grupy. Tak więc i w tym przypadku liczbę dostępów obrazuje wzór (5.16). Likwidacja całej grupy jest również prosta. Przy założeniu, że w grupie jest średnio n rekordów członkowskich, wyniesie ona:

$$k = w + 1 + n, \qquad (5.17)$$

gdzie w jest kosztem dostępu do właściciela.

Tak więc liczby logicznych dostępów w grupach z tablicami wskaźników są zdecydowanie mniejsze od odpowiednich liczb w grupach cyklicznych. Należy jednak widzieć również wady tej metody. Jest nią bardziej skomplikowany aparat programowy obsługujący obszar na tablice wskaźników czy obsługujący same tablice (wyszukiwanie adresu, sortowanie, aktualizacja tablicy). Zwiększa to system obsługi bazy danych, a przy małej liczbie rekordów członkowskich w grupie może wydłużyć czas programu (w porównaniu z grupami cyklicznymi obsługiwanymi przez analogiczny program).

Praktyka wskazuje, że tego rodzaju grupy należy stosować, gdy średnia liczba rekordów członkowskich w grupie jest duża oraz gdy transakcje wymagają porównywania co najmniej dwóch różnych występień grupy (grup) tego typu.

5.8. Podsumowanie

Przedstawioną metodę można stosować na etapie projektowania bazy danych. Dzięki niej można porównywać alternatywne rozwiązania w strukturze bazy. Mamy tu na myśli nie tylko rodzaje wskaźników w grupach, ale także inne elementy projektowe, jak redundancję danych, komasację danych w większe rekordy, wybór bezpośredniego dostępu w rekordach członkowskich, wprowadzanie nowych bądź usuwanie istniejących grup itd. W tym celu każdą transakcję systemu użytkowego należy rozbić na elementarne operacje na projektowanej bazie. Będą to operacje typu: "przeczytaj rekord", "zmodyfikuj rekord", "usuń rekord/rekordy". Znając średnią liczbę rekordów przetwarzanych przez daną transakcję można przy zadanej strukturze bazy policzyć koszt pracy tej transakcji (przy wykorzystaniu wzorów (5.1)-(5.17)). Koszt całkowity transakcji będzie sumą kosztów operacji elementarnych.

Łatwo zauważyć, że poprawienie struktury bazy dla jednej transakcji (np. usunięcie grupy) może, nawet znacznie, zwiększyć koszt pracy innej transakcji. Aby przyjąć rozwiązanie optymalne dla całego systemu użytkowego, proponuje się oszacowanie kosztu pracy całego systemu. Zdefiniujmy go wzorem:

$$k = \sum_{i=1}^{n} p_i T_i, \qquad (5.18)$$

gdzie T_i oznacza koszt i-tej transakcji systemu, a p_i oznacza częstotliwość i-tej transakcji w systemie.

Minimalizacja kosztu całkowitego systemu będzie więc polegała na minimalizacji liczby logicznych dostępów szczególnie częstych transakcji, co zresztą jest intuicyjnie wyczuwalne.

6. OCENA I ULEPSZANIE STRUKTURY LOGICZNEJ BAZY DANYCH

W rozdziale tym pokażemy, jak praktycznie wykorzystać poznane uprzednio wzory i zasady. Celem porównania alternatywnych rozwiązań w projekcie struktury logicznej będziemy obliczać dla tych rozwiązań liczby logicznych dostępów do bazy. Rozwiązanie o mniejszej liczbie dostępów będzie lepsze i dla niego system będzie pracował sprawniej. Jednakże metody tej nie będziemy stosować wybiórczo do porównywania różnych rozwiązań projektowych. Przedstawimy usystematyzowaną metodykę projektowania, w której metoda obliczania liczby logicznych dostępów znajdzie swoje miejsce.

Punktem wejścia do tego etapu jest wstępny schemat struktury logicznej bazy. Mamy w nim zdefiniowane podstawowe obiekty bazy - rekordy i grupy logiczne. Przeprowadzony wywiad pozwolił nam na zebranie informacji o stosunkach liczbowych zachodzących między tymi obiektami. W szczególności więc znamy projektowaną liczbę wszystkich rekordów poszczególnych typów i średnią liczbę wystąpień rekordów członkowskich w każdej grupie. Te informacje pozwolą nam na obliczenie kosztu pracy każdej transakcji w systemie. Kosztem tym jest iloczyn liczby logicznych dostępów i częstotliwości danej transakcji. Porównanie tego wskaźnika pozwoli na wybór najbardziej uciążliwych transakcji systemu. Na tym etapie będzie można pogrupować wszystkie transakcje ustalając ich priorytet konieczności poprawy struktury. Następnym krokiem będzie ulepszanie struktury dla kolejnych transakcji, zgodnie z ustalonym wcześniej priorytetem. Ulepszanie struktury będzie polegało na wypisaniu wszystkich alternatywnych rozwiązań poprawiających pracę danej transakcji i wykonaniu obliczeń dla każdego z nich. Chodzi tu zarówno o obliczanie liczby logicznych dostępów dla każdego rozwiązania, jak i o liczbę słów w pamięci masowej straconych lub zyskanych na wskutek jego przyjęcia. Porównanie obu tych liczb dla wszystkich alternatywnych rozwiązań pozwoli wybrać rozwiązanie najlepsze. Po ulepszeniu w ten sposób struktury dla wybranych wcześniej transakcji pozostaje jeszcze odpowiednie udokumentowanie tego etapu pracy. Przede wszystkim należy zaktualizować rysunek struktury logicznej bazy z zaznaczonymi związkami liczbowymi, które na skutek zmian logicznych mogły ulec zasadniczej zmianie. Kolejnym krokiem jest ponowne przeliczenie kosztu pracy wszystkich transakcji systemu. Wszystkich, a więc także i tych, dla których struktura nie uległa zmianie bezpośrednio. Koszt ten może bowiem ulec zmianie na skutek zmiany struktury wykonanej na potrzeby innych transakcji. W szczególności koszt ten może nawet poważnie wzrosnąć, ale jest to wkalkulowane w koszty metody. Mimo wzrostu kosztu pracy pojedynczej

transakcji o mniejszym priorytecie, sumaryczny koszt pracy całego systemu będzie na pewno mniejszy niż w rozwiązaniu początkowym. Obliczenia przedstawione powyżej są wykonywane właśnie w celu udokumentowania tego faktu. Etap ten kończy wypełnienie formularza "GRUPY", w którym opiszemy wszystkie zdefiniowane w projekcie grupy logiczne. Dla każdej grupy podamy opisujące je parametry, takie jak logiczny punkt wstawienia, typ grupy, rodzaj wskaźników używanych w tych grupach i inne. Formularz ten ułatwi nam przygotowanie postaci źródłowej schematu logicznego bazy danych.

Podsumowując, etap oceny i ulepszania logicznej struktury danych można podzielić na następujące kroki:

1) zebranie danych liczbowych o obiektach systemu (liczba rekordów, liczebność grup),
2) określenie kosztu pracy każdej transakcji systemu,
3) grupowanie transakcji ze względu na stopień obciążenia systemu,
4) przedstawienie wszystkich możliwych rozwiązań i wykonanie odpowiednich obliczeń dla transakcji, dla których należy zmienić strukturę,
5) wybór właściwego rozwiązania dla ww. transakcji,
6) aktualizacja rysunku struktury logicznej bazy danych,
7) ponowne obliczanie kosztu pracy każdej transakcji, a więc i kosztu pracy całego systemu,
8) wypełnienie formularza dokumentującego wybrane rozwiązania - opis grup logicznych zdefiniowanych w bazie.

6.1. Obliczanie kosztu pracy transakcji

Poznaną w rozdziale poprzednim metodę obliczania liczby logicznych dostępów można zastosować do oceny zaprojektowanej wstępnej struktury logicznej bazy danych. W tym celu po zebraniu wszystkich informacji dotyczących częstotliwości transakcji, wielkości pól, liczby rekordów i liczebności grup należy każdą transakcję rozłożyć na operacje elementarne. Operacjami elementarnymi na rekordach bazy są w tym przypadku dostęp, zapamiętanie i usuwanie. Modyfikację rekordu można uważać za dostęp (gdy chodzi o zmianę danych) lub za złożenie operacji usuwania i ponownego wprowadzania (w przypadku modyfikacji klucza lub zmiany właściciela). Liczba operacji elementarnych może być różna w różnych przebiegach tego samego programu, co może zależeć choćby od liczby danych wejściowych, ale i w takim przypadku można mówić o średniej liczbie tych operacji. Posługując się wzorami z rozdziału 5 można obliczyć koszt każdej pojedynczej transakcji systemu.

Obliczenia takie należy przeprowadzić dla wszystkich transakcji projektowanego systemu. Wyniki obliczeń można nanosić na formularz "LLD" (liczba logicznych dostępów), co ma również znaczenie dokumentacyjne. Omówmy poszczególne pozycje tego formularza (zał. 6.1).

REKORDY

NR PROGRAMU	TYP																			LLD DLA PROGRAMU	CZĘSTOTLIWOŚĆ PROGRAMU	WSKAŹNIK OBCIĄŻENIA

LLD - LICZBA LOGICZNYCH DOSTĘPÓW

Rys. 6.1. Formularz "LLD"

Pierwsza kolumna zawiera numer transakcji, którym posługiwaliśmy się już poprzednio. Kolumna druga opisuje typ przetwarzania programu ("R" - czas rzeczywisty, "B" - tryb wsadowy). Kolejne rubryki zawierają nazwy rekordów zdefiniowanych w bazie. Do rubryk tych wpisuje się średnią liczbę logicznych dostępów w programie dla danego typu rekordu. Zauważmy tutaj, że zaprezentowana metoda obliczania liczby logicznych dostępów pozwala nam na oszacowanie tej liczby z zachowaniem rozróżnienia typu rekordu. Innymi słowy, powie nam, że jakaś transakcja wykonuje średnio n dostępów do rekordu A, m dostępów do rekordu B, k dostępów do C itd. Kolejne 3 kolumny zawierają następujące pozycje. Pierwsza z nich to całkowita liczba logicznych dostępów całej transakcji. Druga przedstawia częstotliwość transakcji w tym samym co poprzednio sensie, a więc jako procentowy udział transakcji w obciążaniu całego systemu. Wreszcie ostatnia kolumna służy do zapisywania wskaźnika obciążenia, będącego iloczynem liczby logicznych dostępów i częstotliwości transakcji.

Wskaźnik obciążenia pokazuje nam udział danej transakcji w pracy całego systemu. Wysoki wskaźnik oznacza, że odpowiadająca mu transakcja jest szczególnie czasochłonna lub co najmniej częsta. Pokażemy na przykładzie, jak posługiwać się tym formularzem.

Wróćmy do głównego przykładu prezentowanego już w rozdziale 4. Załącznik 6.2 to wypełniony formularz "LLD" dla omawianego projektu. Aby nie powtarzać podobnych wyjaśnień i obliczeń, ograniczymy się do omówienia tylko kilku wierszy tego formularza. Każdy wiersz przedstawia oczywiście jedną transakcję; omówmy więc kilka z nich. Zakładamy przy tym, że wszystkie grupy mają porządek "NASTĘPNY" i nie posiadają dodatkowych wskaźników.

Transakcja 4

Aby pokazać wszystkich podwładnych danego pracownika, należy przede wszystkim wykonać dostęp bezpośredni do właściwego rekordu PRACOWNIK. Następnie należy przeglądać grupę PRAC-PODWL i dla każdego członka tej grupy szukać rekord właściciela w grupie drugiej, tzn. PRAC-SZEF. Z uwagi na średnią liczebność obu grup należy wykonać więc:

$$10 \times ((2+1)/2) = 15 \qquad \text{(patrz wzór (5.14))}$$

dostępów do rekordu typu L-PRAC. To pozwoli nam na dojście do 10 właściwych rekordów typu PRACOWNIK. W rezultacie otrzymujemy więc 11 dostępów do rekordu PRACOWNIK i 15 dostępów do rekordu L-PRAC.

Transakcja 14

"Podaj zakłady w ramach danego przemysłu mające specyficzny stosunek do naszego zakładu". Żądanie to może być realizowane czterema sposobami.

REKORDY

NR PROGRAMU	TYP	PRACOWNIK	TEL-SLUZ	ZAKLAD	L-PRAC	KONT-PRAC	ORGANIZACJA	D-URODZ	UPODOBANIA	PRZEMYSL	RELACJA	L-ZAK	SPOTKANIE	REZULTAT	L-SPOT-ZAK	TYP	L-PRAC-SPOT	
1		1	3															
2		5,5		1														
3		3			11													
4		11			15													
5		1				1												
6		1				1												
7		1					2											
8		3						1										
9		1							2									
10		1				1												
11				1														
12		10	30	1														
13				1255						1	10							
14a				1255						1	10							
14b				1375						250	1							

LLD - LICZBA LOGICZNYCH DOSTĘPÓW

LLD DLA PROGRAMU	CZĘSTOTLIWOŚĆ PROGRAMU	WSKAŹNIK OBCIĄŻENIA
4	24	96
6.5	26	169
14	3	42
26	3	78
2	3	6
2	.43	.86
3	1	3
4	1	4
3	3	9
2	.43	.86
1	7	7
41	3	123
1266	1	1266
1266	.43	544
1626	.43	699

REKORDY

NR PROGRAMU	TYP	PRACOWNIK	TEL-SLUZ	ZAKLAD	L-PRAC	KONT-PRAC	ORGANIZACJA	D-URODZ	UPODOBANIA	PRZEMYSL	RELACJA	L-ZAK	SPOTKANIE	REZULTAT	L-SPOT-ZAK	TYP	L-PRAC-SPOT
14c				260						1	1						
14d				10						/+1/	/+1/						
15				3								2					
16				2								1.5					
17													1	10			
18				1									982.5		30	15	
19		1											655			10	20
20		2145		390									130			1	2145
21		2145		390									130			1	2145
22		3		3									66.5		27	1	19.5
23		3		3									136		48	1	33

LLD DLA PROGRAMU	CZĘSTOTLIWOŚĆ PROGRAMU	WSKAŹNIK OBCIĄŻENIA
262	.43	113
14	.43	6
5	.43	2
3.5	.43	1.5
11	9	99
1028,5	.43	442
686	4	2744
4811	.43	2069
4811	1	4811
120	4	480
424	4	1696

LLD - LICZBA LOGICZNYCH DOSTEPÓW

Rys. 6.2. Przykład 4.6 - formularze "LLD"

1. Przeglądać sekwencyjnie grupę PRZEM-ZAK dla zadanego rekordu PRZEMYSŁ i dla każdego rekordu tej grupy znajdywać jego właściciela w grupie REL-ZAK, sprawdzając jednocześnie, czy kod relacji jest identyczny z zadanym. Aby to zrealizować, należy wykonać 1 dostęp do rekordu PRZEMYSŁ, 10 dostępów do rekordu ZAKLAD i odpowiadających im 10 dostępów do rekordu RELACJA. Aby jednak dotrzeć do rekordu RELACJA, należy średnio przejrzeć $\frac{250-1}{2}$ rekordów ZAKLAD za każdym razem. W rezultacie musimy wykonać 1 dostęp do rekordu PRZEMYSŁ, 10 - do rekordu RELACJA i 1255 do rekordu ZAKLAD, bo:

$$1255 = 10 + 10 \times ((250-1)/2).$$

2. Ustalić rekord RELACJA i sekwencyjnie przeglądać grupę REL-ZAK pobierając dla każdego członka grupy jego właściciela w grupie PRZEM-ZAK. Rozumując analogicznie jak w poprzednim sposobie musimy wykonać 1 dostęp do rekordu RELACJA, 250 dostępów do rekordu PRZEMYSŁ i 1375 dostępów do rekordu ZAKLAD. Jak widać, sposób ten jest gorszy od poprzedniego.

3. Ustalić właściwe rekordy PRZEMYSL i RELACJA. Przeczytać wszystkie rekordy obu grup PRZEM-ZAK i REL-ZAK, sprawdzając, które rekordy członkowskie należą jednocześnie do obu grup. W tym celu należy przeczytać 1 rekord PRZEMYSŁ, 1 rekord RELACJA i 260 (250+10) rekordów ZAKLAD. Oczywiście samo sprawdzanie zgodności rekordów w obu grupach jest trudne i będzie wykonane przez odpowiednią procedurę w pamięci operacyjnej. Nie pociągnie to jednak za sobą konieczności wykonywania dodatkowych dostępów do bazy danych.

4. Ten typ przetwarzania najlepiej realizowany jest przez grupy z tablicami wskaźników. Wtedy wymagane są: 1 dostęp do rekordu RELACJA i 1 do jego tablicy wskaźników, 1 dostęp do rekordu PRZEMYSŁ i jego tablicy oraz co najwyżej 10 dostępów do rekordu ZAKŁAD. W sumie najwyżej 14 logicznych dostępów do bazy danych.

Transakcja 18

"Podaj wszystkie spotkania w podanym zakresie czasu z udziałem przedstawicieli zadanego zakładu".

Przede wszystkim należy wykonać 1 dostęp do właściwego rekordu ZAKŁAD, a następnie przeczytać całą grupę ZAK-SPOT. Niepuste grupy ZAK-SPOT posiadają średnio 15 rekordów członkowskich, a dla każdego z nich należy znaleźć właściciela w grupie SPOT-ZAK. Należy więc wykonać 15 dostępów do rekordu SPOTKANIE i 15 x ((3+1)/2) = 30 dostępów do rekordu L-ZAK-SPOT. Z kolei po znalezieniu rekordu SPOTKANIE i sprawdzeniu, czy spełnia on zadane warunki (czy mieści się w podanym zakresie czasu), należy jeszcze wykonać dostęp do rekordu TYP będący właścicielem grupy TYP-SPOT. W rezultacie należy wykonać 15 dostępów do rekordu TYP i 15 x ((130+1)/2) = = 982,5 dostępów do rekordu SPOTKANIE wliczając w tę liczbę dostępy wyko-

nane przy pobieraniu rekordu SPOTKANIE jako właściciela grupy SPOT-ZAK. Należy zaznaczyć, że uzyskane wyniki obrazują wartości maksymalne (jeżeli rekord SPOTKANIE jest spoza zakresu dat, nie trzeba robić dostępu do rekordu TYP) i nie zakładają żadnego uporządkowania zainteresowanych grup (rekordy w bazie mogą być zapamiętywane w dowolny sposób, niekoniecznie w kolejności zgodnej z datami ich pojawiania się).

Transakcja 20

"Podaj wszystkie spotkania zadanego typu w podanym zakresie czasu". Aby zrealizować tę funkcję, należy rozpocząć od znalezienia rekordu TYP i następnie sekwencyjnie odczytać wszystkie rekordy SPOTKANIE z tej grupy. Z uwagi na to, że o każdym spotkaniu musimy mieć informacje o osobach i zakładach biorących w nim udział, należy przetworzyć grupy SPOT-PRAC, PRAC-SPOT i ZAK-PRAC. Przetwarzanie obu ostatnich grup polega na szukaniu rekordu właściciela, a więc odpowiednio rekordów PRACOWNIK i ZAKŁAD. Aby znaleźć właścicieli dla wszystkich rekordów członkowskich, należy wykonać do nich 3 x ((10+1)/2) dostępy (ta sama liczba w obu grupach). Z uwagi na to, że spotkań danego typu jest średnio 130 i że spotkania nie są posortowane wg daty, należy wykonać: 130 dostępów do rekordu SPOTKANIE, 130 x 3 x 11/2 = 2145 dostępów do rekordu L-SPOT-PRAC, 2145 dostępów do rekordu PRACOWNIK, 3 x 130 = 390 dostępów do rekordu ZAKŁAD. Uzyskane wyniki znów mają charakter maksymalny.

Transakcja 22

"Wprowadź do bazy informacje o nowym spotkaniu". Aby to uczynić, należy ustalić właściwy rekord TYP, stąd 1 dostęp do tego rekordu. Dla zapamiętania rekordu SPOTKANIE potrzebne jest wykonanie średnio 1+(130+1)/2 = = 66,5 dostępów do tego typu rekordu celem aktualizacji wskaźników w grupie TYP-SPOT. Następnie należy ustalić w sposób bezpośredni 3 rekordy PRACOWNIK i 3 rekordy ZAKŁAD reprezentujące osoby biorące udział w spotkaniu. Aby zapamiętać 3 rekordy łącznikowe L-SPOT-PRAC i L-SPOT-ZAK, należy wykonać odpowiednio: 3 x (1+(10+1)/2) = 19,5 dostępów w grupie PRAC-SPOT i 3 x (1+(15+1)/2) = 27 dostępów w grupie ZAK-SPOT.

Transakcja 23

"Usuń informacje o spotkaniu". Z usunięciem informacji o spotkaniu łączy się usunięcie z bazy danych rekordu SPOTKANIE, 3 rekordów L-SPOT-PRAC i L-SPOT-ZAK oraz ewentualnie 10 rekordów REZULTAT (o ile usuwamy z bazy spotkania archiwalne). Załóżmy jednak, że w tej funkcji chodzi raczej o usuwanie z bazy takich spotkań, które nie doszły do skutku.

Koszt usunięcia rekordu SPOTKANIE to 1 dostęp na znalezienie tego rekordu i aktualizacja grupy TYP-SPOT, która wymaga 129 dostępów do rekordu SPOTKANIE i 1 dostęp do rekordu TYP. Wcześniej jednak należy usunąć rekor-

dy L-SPOT-ZAK i L-SPOT-PRAC. Usuwanie rekordu L-SPOT-ZAK pociąga za sobą konieczność aktualizacji grupy ZAK-SPOT. Na mocy wzorów (5.9) i (5.10) koszt tej operacji wyniesie: 3 x (15-1) = 42 dostępy do rekordów L-SPOT-ZAK i 3 dostępy do rekordu ZAKŁAD. Koszt likwidacji grupy SPOT-ZAK wyniesie z kolei 3(3+1)/2 = 6 dostępów do rekordów L-SPOT-ZAK i dodatkowo 3 dostępy do rekordu SPOTKANIE (wzór (5.7)). Podobnie koszt likwidacji grupy SPOT-PRAC wyniesie 6 dostępów do rekordu L-SPOT-PRAC i 3 dostępy do rekordu SPOTKANIE, a koszt aktualizacji grupy PRAC-SPOT 3 x (10-1) = 27 dostępów do rekordu L-SPOT-PRAC i 3 dostępy do rekordu PRACOWNIK. W rezultacie więc musimy wykonać 1 + 129 + 3 + 3 = 136 dostępów do rekordu SPOTKANIE, 42 + 6 = 48 dostępów do rekordu L-SPOT-ZAK i 27 + 6 = 33 dostępy do rekordu L-SPOT-PRAC.

6.2. Grupowanie transakcji ze względu na obciążenie systemu

W rozdziale poprzednim pokazaliśmy, jak wykonywać oszacowanie kosztu transakcji w projektowanym systemie. Zastanówmy się teraz, jak ulepszać wstępną strukturę danych dla całego systemu. Wiadomo, że poprawienie struktury dla jednej transakcji może znacznie pogorszyć warunki pracy innej transakcji. Rozwiązanie optymalne byłoby takie, dla którego forma

$$\sum_{i=1}^{n} p_i T_i \qquad \text{(patrz wzór (5.18))}$$

osiągałaby minimum.

Minimum tego analitycznie nie można wyznaczyć, zaproponujemy więc metodę, która praktycznie doprowadza nas do rozwiązania zbliżonego do optymalnego. Metoda ta polega na poprawie struktury dla tych transakcji, których udział w systemie jest największy, a więc transakcji najczęstszych i najbardziej czasochłonnych. Mając powyższe na uwadze możemy pogrupować transakcje wyróżniając wśród nich:

1) programy w czasie rzeczywistym o dużej częstotliwości realizacji,
2) programy w czasie rzeczywistym o małej częstotliwości realizacji, lecz dużym obciążeniu całkowitym systemu (wskaźnik z "LDD"),
3) programy wsadowe o wysokiej częstotliwości realizacji,
4) inne programy.

W grupie ostatniej znajdują się te programy, których udział w obciążeniu systemu jest niski lub których w ogóle nie można optymalizować przez poprawianie struktury danych. Proponuje się przyjęcie następujących zasad decydujących o zaliczaniu programów do tej grupy:

- program przetwarza rekordy tylko jednego typu,

- wskaźnik obciążenia transakcji jest mniejszy niż 50,
- częstotliwość realizacji programu wsadowego jest niższa niż 5%.

Programy spełniające powyższe warunki zaliczać będziemy do czwartej grupy i nie będziemy ich uwzględniać przy poprawie struktury danych. Pozostają nam trzy grupy programów o coraz to niższych priorytetach. Dla wszystkich programów z tych grup należy przeprowadzić dokładną analizę struktury danych, celem zmniejszenia liczby logicznych dostępów, co spowoduje obniżenie wskaźnika obciążenia. Nie oznacza to oczywiście, że nie można optymalizować bazy dla programów grupy czwartej. Można to robić, ale po pierwsze - nie w początkowym etapie projektowania, lecz później, a po drugie - tak reorganizować strukturę, aby nie zakłócić pracy programom o wyższym priorytecie. Stosowanie tych zasad pozwoli nam na dalsze zbliżenie się do rozwiązania optymalnego.

6.3. Poprawa struktury danych dla poszczególnych transakcji

W rozdziale 6.2 wybraliśmy te transakcje, dla których należałoby poprawić strukturę danych. Każdą z tych transakcji będziemy rozpatrywać oddzielnie. W razie konfliktu o wyborze rozwiązania w całym systemie będzie decydować przydzielony transakcjom priorytet, a gdy te są równe - większy udział w obciążeniu systemu. Dla każdej transakcji będziemy postępować podobnie. Najpierw wypiszemy wszystkie nasuwające się nam rozwiązania poprawiające warunki pracy transakcji. Dla każdego z nich obliczymy ponownie liczbę logicznych dostępów do bazy danych oraz liczbę straconych czy zyskanych słów pamięci masowej po wdrożeniu takiego rozwiązania. Zysk lub strata pamięci może być związana z wprowadzeniem celowej redundancji i dodatkowych wskaźników z jednej strony, a z rezygnacją z rekordów czy grup logicznych z drugiej strony.

O wyborze właściwego rozwiązania decyduje przede wszystkim liczba logicznych dostępów. Jeżeli są one zbliżone, a jedna z metod będzie wyraźniej oszczędniej wykorzystywała pamięć masową, tę właśnie metodę należy wybrać. O wyborze rozwiązania mogą również decydować inne czynniki, takie jak: stabilność danych, częstotliwość ich użycia, wielkość pól, porządek przetwarzania rekordów i inne.

Pokażemy ulepszanie struktury w dwu najczęstszych przypadkach. Są nimi dostęp w górę struktury (ang. upward access) i dostęp w dół struktury (ang. downward access).

Dostęp w górę ma miejsce wtedy, gdy poprzez dostęp bezpośredni, przeszukiwanie sekwencyjne obszaru lub grupy pobraliśmy pewien rekord bazy i wymagane jest pobranie pewnych informacji z rekordu stojącego wyżej w strukturze. Z reguły jest to rekord-właściciel grupy (innej niż użyta do znalezienia rekordu), w której bieżący rekord jest członkiem. Zasadniczo mamy dwie możliwości poprawy takiej struktury. Możemy zdecydować się na

celową redundację pewnych pól z rekordu właściciela w rekordzie członkowskim lub na wprowadzenie dodatkowego wskaźnika "do właściciela" w omawianej grupie. O wyborze rozwiązania decydować mogą różne czynniki. I tak, istotna jest wielkość pola, dla odczytu którego wykonujemy dostęp. Gdy pole to jest mniejsze niż wskaźnik w rekordzie, redundancja jest rozwiązaniem oszczędniejszym. Wielkość zaoszczędzonego miejsca pamięci zależy oczywiście od przewidywanej liczby rekordów. Gdy liczba ta jest mała, oszczędzimy w ten sposób niewiele i tego czynnika można w ogóle nie brać pod uwagę. Innym czynnikiem decydującym o wyborze może być stabilność danej. Rozumiemy przez to częstotliwość modyfikacji tej danej. Gdy jest ona wysoka, to po zdecydowaniu się na redundancję każdorazowa modyfikacja danej spowoduje konieczność poprawy wszystkich rekordów członkowskich, w których ona występuje.

Dostęp w dół realizowany jest bardzo często w bazach danych, gdyż po to właśnie definiowane są grupy logiczne. Tym niemniej należy zastanowić się, jak usprawnić i przyspieszyć tę formę przetwarzania. Przede wszystkim istotny jest cel, w jakim musimy przetwarzać grupę, tzn. czy pobieramy informację z rekordów członkowskich, czy zapamiętujemy nowy rekord.

W rozdziale 5.1 pokazano, że koszt zapamiętywania nowego rekordu zależy od porządku w grupie (tzn. logicznego punktu wstawiania) i rodzaju wskaźników zdefiniowanych w grupie. Z kolei, aby usprawnić przetwarzanie w dół celem pobrania informacji można zdecydować się na dwa rozwiązania. Pierwsze z nich to konsolidacja danych, a więc właściwie zlikwidowanie grupy. Zamiast niej wprowadzimy rekord o maksymalnej lub zmiennej długości zawierający tablicę tych pól, które dotychczas były w rekordzie członkowskim. Liczba dostępów zostanie oczywiście ograniczona tylko do jednego, ale metoda ta ma też swoje wady, o których mówiliśmy już w rozdziale 4.4. Przy wyborze takiego rozwiązania należy wziąć pod uwagę wielkość powtarzanego pola, liczbę powtórzeń, częstotliwość użycia pól, stabilność (tzn. częstotliwość modyfikacji) oraz możliwość zmiany długości (rozszerzenia) rekordu zawierającego zmiennej długości tablicę. Konsolidacji nie można również wykonać, gdy rekord członkowski uczestniczy w strukturze sieciowej będąc równocześnie członkiem innej grupy. Wtedy rekordu członkowskiego nie można połączyć z żadnym właścicielem, gdyż stracilibyśmy w ten sposób związki logiczne zachodzące między danymi. Podobnie, gdy do rekordu członkowskiego wymagany jest dostęp bezpośredni, nie ma sensu konsolidacja tego rekordu z właścicielem. Rekord ten powinien być jednoznacznie definiowany przez klucz, gdyż wymaga tego przetwarzanie, który po konsolidacji zostałby zwykłym polem niekluczowym w rekordzie głównym.

Drugim rozwiązaniem jest prezentowana już w rozdziale 4.4. metoda pseudogrupy z jednym tylko członkiem - zmiennej długości rekordem-kontynuacją. Liczba dostępów w tym przypadku ograniczy się do dwóch, ale również to rozwiązanie posiada podobne jak poprzednie uwarunkowania (wielkość pól, liczba powtórzeń, częstotliwość przetwarzania, stabilność). Ważne jest

także, czy przetwarzane są wszystkie, czy tylko wybrane pola rekordu. W wielu przypadkach może się okazać, że jest to rozwiązanie lepsze od konsolidacji. Tym niemniej pozostawienie grupy logicznej i ewentualnie zdefiniowanie w niej dodatkowych wskaźników jest najprostsze i najbardziej elastyczne. W większości wypadków analiza przetwarzania w dół struktury nie usunie nam tego wygodnego narzędzia, jakim jest grupa.

6.4. Przykład

Pokażemy na przykładzie, jak funkcjonują przedstawione powyżej zasady. Załóżmy, że projektujemy bardzo prostą bazę danych, którą obsługiwać będzie jedynie 7 programów - 5 w czasie rzeczywistym i 2 w trybie wsadowym. Listę tych programów przedstawia formularz FUNKCJE (zał. 6.3), a formularz DANE (zał. 6.4) pokazuje pola rekordów bazy. Wreszcie zał. 6.5 przedstawia rysunek wstępnej struktury bazy dla tego projektu. Zaznaczono na nim wielkości liczbowe charakteryzujące projektowane obiekty. Kolejnym etapem jest przygotowanie formularza LLD (zał. 6.6). Dla wszystkich transakcji systemu obliczamy średnią liczbę logicznych dostępów do bazy oraz wskaźnik obciążenia systemu przez tę transakcję. Porównując te wskaźniki zauważamy, że najbardziej obciąża system transakcja 1 (wskaźnik obciążenia 31230). Pogrupujmy wszystkie programy projektowanego systemu, biorąc pod uwagę współczynnik obciążenia i częstotliwość. Otrzymujemy w rezultacie 4 grupy:

- transakcje częste (nr 1, nr 3),
- transakcje rzadkie (nr 4, nr 7),
- programy wsadowe częste (nr 2),
- inne programy (nr 5, nr 6).

Do ostatniej grupy zaliczyliśmy program 5, gdyż działa on tylko na jednym typie rekordu oraz transakcję 6 z uwagi na niski wskaźnik obciążenia systemu. Spróbujmy poprawić strukturę danych kolejno dla programów 1, 3, 4, 7 i 2.

Program 1 - składa się z przetwarzania w dół według grupy PROJ-PRAC i przetwarzania w górę w grupie WYDZ-PRAC.

Strukturę danych dla tego programu można ulepszyć na 2 sposoby. Są to: redundancja danych lub zdefiniowanie dodatkowych wskaźników w grupie WYDZ-PRAC. Porównajmy dokładniej oba rozwiązania. Pierwsze z nich spowoduje przeniesienie pola NR-WYDZ do rekordu PRACOWNIK, co pociąga za sobą utratę 10000 x 4 znaków tj. 10000 słów. (Przykład ten jest wzięty z kursu projektowania baz na komputerach UNIVAC-1100, a w tym typie maszyny słowo składa się z 36 bitów, na których można zapamiętać 4 znaki ASCII). Drugie rozwiązanie to wprowadzenie wskaźników "DO WŁAŚCICIELA" w grupie WYDZ-PRAC.

Załącznik 6.3

"FUNKCJE"

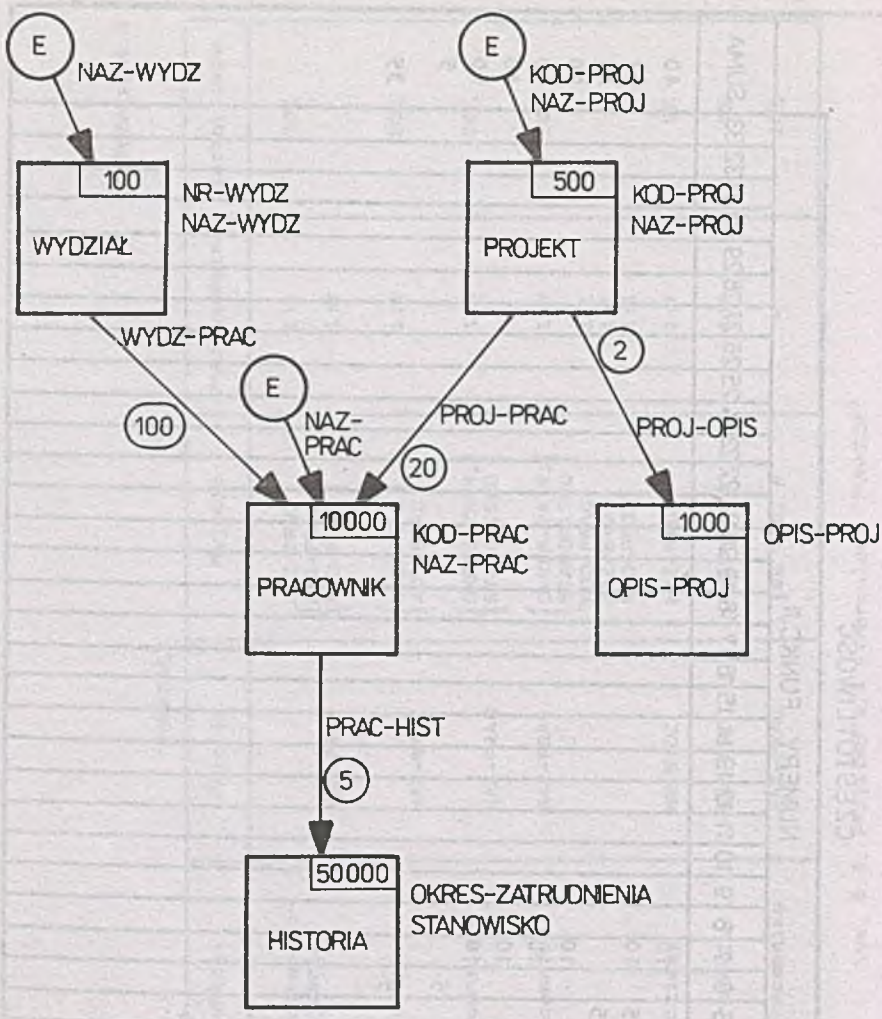
Nr funkcji	Nazwa lub opis funkcji	Wejście	Wyjście	Pokrewieństwo	Częstotliwość
1	Podaj pracowników zatrudnionych przy danym projekcie		NAZ-PROJ	1:1	30%
		KOD-PROJ	NAZ-PRAC KOD-PRAC	1:W	
2	Podaj pracowników danego wydziału	NAZ-WYDZ	NAZ-PRAC KOD-PRAC	1:W	20%
3	Podaj historię zatrudnienia danego pracownika	NAZ-PRAC	OKRES-ZATR. STANOWISKO	1:W	20%
4	Wprowadź nowego pracownika	NAZ-PRAC	OKRES-ZATR. STANOWISKO	1:W	10%
			KOD-PRAC	1:1	
			NAZ-WYDZ	1:1	
			KOD-PROJ	1:1	
5	Wypisz wszystkie wydziały	NR-WYDZ	NAZ-WYDZ	1:1	5%
6	Podaj opis danego projektu	NAZ-PROJ	OPIS-PROJ	1:1	5%
7	Usuń informacje o pracowniku	NAZ-PRAC	jak w p. 4	-	10%

Rys. 6.3. Przykład 6.4 - formularz "FUNKCJE"

„DANE"

Nazwa pola	Wielk. pola	CZĘSTOTLIWOŚĆ NUMERY FUNKCJI 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	SUMA
OKRES-ZATR	12			20	10			10																											40
NAZ-WYDZ	25		20		10	5		10																											45
NR-WYDZ	4	30				5																													35
NAZ-PRAC	36	30	10	20	10			10																											90
KOD-PRAC	6		20		10			10																											40
STANOWISKO	25			20	10			10																											40
KOD-PROJ	6	30			10			10																											50
OPIS PROJ	25-1000						5																												5
NAZ-PROJ	25	30					5																												35

Rys. 6.4. Przykład 6.4 - formularz "DANE"



Rys. 6.5. Przykład 6.4 - wstępny projekt powiązań rekordów

Koszt takiego rozwiązania rozpatrywany w kategoriach pamięci wynosi również 10.000 słów, przeznaczonych na nowe wskaźniki w rekordzie PRACOWNIK. Sprawdźmy teraz, jak zmieni się liczba logicznych dostępów w obu rozwiązaniach. W pierwszym przypadku redukuje się ona do 21, a w drugim do 41. Biorąc jeszcze pod uwagę fakt, że pole NR-WYDZ jest polem stabilnym, bardzo rzadko zmienianym, decydujemy się na przyjęcie rozwiązania pierwszego.

Program 3 - to typowy przykład przetwarzania w dół po grupie PRAC-HIST. Istnieją dwa sposoby ulepszenia takiej struktury. Są to: konsolidacja wszystkich danych w 1 rekord PRACOWNIK lub zdefiniowanie zmiennej długoś-

REKORDY

NR PROGRAMU	TYP	WYDZIAŁ	PRACOWNIK	PROJEKT	OPIS-PROJ	HISTORIA
1	R	20	1020	1		
2	B	1	100			
3	R		1			5
4	R	1	61	1		5
5	B	100				
6	R			1	2	
7	R	1	117	1		5

LLD DLA PROGRAMU	CZĘSTOTLIWOŚĆ PROGRAMU	WSKAŹNIK OBCIĄŻENIA
1041	30	31230
101	20	2020
6	20	120
68	10	680
100	5	500
3	5	15
124	10	1240

LLD - LICZBA LOGICZNYCH DOSTĘPÓW

Rys. 6.6. Przykład 6.4 - formularz "LLD"

ci rekordu członkowskiego, który zawierałby historię całego zatrudnienia pracownika. Liczba logicznych dostępów programu po zastosowaniu obu rozwiązań redukuje się odpowiednio do 1 i 2 dostępów. Obliczmy teraz, ile zyskamy (stracimy) pamięci po wdrożeniu tych metod. W pierwszym przypadku z uwagi na likwidację grupy PRAC-HIST oszczędzimy 1 wskaźnik w 50.000 rekordach HISTORIA oraz w 10.000 rekordach PRACOWNIK. Da to łącznie 60.000 słów pamięci. Jednak w tym przypadku konieczne jest zdefiniowanie w rekordzie PRACOWNIK licznika powtórzeń, gdyż jest to rekord zmiennej długości. Załóżmy, że licznik ten zmieści się na 1/4 słowa, a więc że utracimy w ten sposób 2,5 tys. słów. Reasumując, wdrożenie tej metody oszczędzi nam 57,5 tys. słów pamięci masowej.

Przeprowadzimy analogiczne obliczenia dla drugiego rozwiązania. Z uwagi na ograniczenie liczby rekordów HISTORIA z 50 tys. do 10 tys. oszczędzamy 40 tys. słów przeznaczonych na wskaźniki w grupie PRAC-HIST. Z kolei na licznik powtórzeń w zmiennej długości rekordzie HISTORIA stracimy około 2,5 tys. słów. Łączny bilans wskaże więc oszczędność 37,5 tys. słów. Porównanie liczby logicznych dostępów i zaoszczędzonej pamięci w obu rozwiązaniach przemawia za metodą pierwszą. Weźmy jednak jeszcze pod uwagę inne czynniki. Dane dotyczące historii zatrudnienia w rekordzie PRACOWNIK byłyby duże, a jednocześnie rzadko używane. Proponujemy więc rozwiązanie drugie, oszczędzające co prawda nieco mniej pamięci, ale wygodniejsze i bardziej elastyczne. Rozwiązanie to będzie niezauważalne przez inne programy, natomiast zwiększenie rozmiaru rekordu PRACOWNIK dałoby się negatywnie odczuć we wszystkich programach używających ten rekord.

Program 4 - wstawienie do bazy nowego rekordu PRACOWNIK można zoptymalizować przez przyjęcie logicznego punktu wstawienia w obu grupach PROJ-PRAC i WYDZ-PRAC jako "OSTATNI". W konsekwencji tego stracimy 1 słowo na wskaźnik w rekordach WYDZIAŁ i PROJEKT, co da łącznie utratę 100 + 500 = 600 słów pamięci. W zamian za to uzyskamy zmniejszenie liczby logicznych dostępów z 68 do 6 (biorąc już wprowadzone wcześniej scalenie wszystkich rekordów HISTORIA dla danego pracownika w jeden rekord członkowski). Aby zapamiętać rekord w takiej strukturze, należy bowiem wykonać 1 dostęp do rekordu WYDZIAŁ, 1 dostęp do rekordu PRACOWNIK ostatniego w grupie WYDZ-PRAC, 1 dostęp do rekordu PROJEKT, 1 dostęp do rekordu PRACOWNIK ostatniego w grupie PROJ-PRAC i wreszcie 2 dostępy na zapamiętanie właściwych rekordów PRACOWNIK i HISTORIA.

Program 7 - usuwanie rekordu PRACOWNIK można usprawnić jedynie poprzez zdefiniowanie wskaźników "WSTECZ" w obu grupach WYDZ-PRAC i PROJ-PRAC. Oczywiście stracimy w ten sposób 10.000 x 2 = 20.000 słów na wskaźniki w rekordzie PRACOWNIK. W rekordach WYDZIAŁ i PROJEKT wskaźniki WSTECZ już istnieją z uwagi na wprowadzenie w tych grupach porządku "OSTATNI". Wprowadzenie tych wskaźników znacznie przyspieszy pracę programu. Liczba logicznych dostępów redukuje się ze 124 do zaledwie 6 dostępów (1 dostęp do właściwego rekordu PRACOWNIK, 1 dostęp do rekordu HISTORIA, 2 dostępy do

rekordów w grupie WYDZ-PRAC i 2 dostępy do rekordów w grupie PROJ -PRAC).

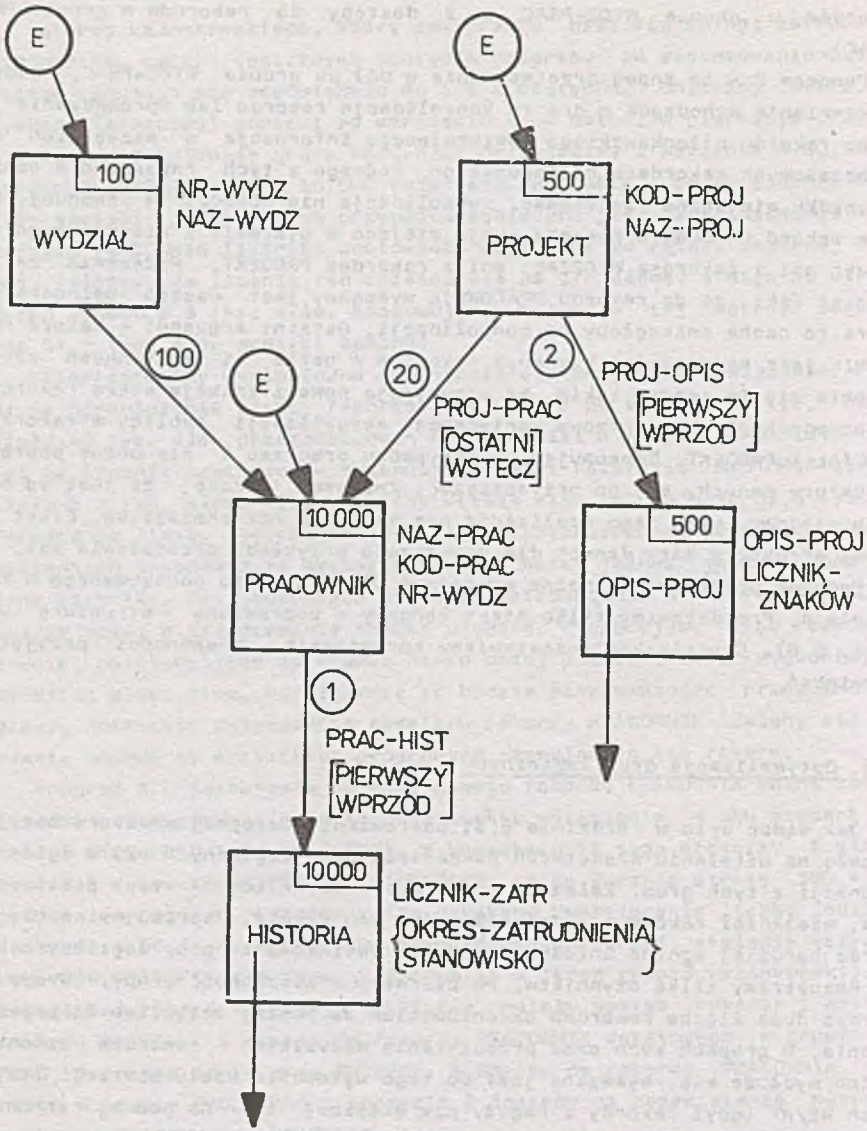
Program 2 - to znowu przetwarzanie w dół po grupie WYDZ-PRAC. Jedyne usprawnienie wchodzące w grę to konsolidacja rekordu lub wprowadzenie jednego rekordu członkowskiego zawierającego informacje o wszystkich dotychczasowych rekordach członkowskich. Żadnego z tych rozwiązań w naszym przypadku nie można zastosować. Konsolidacja nie wchodzi w rachubę, bowiem rekord PRACOWNIK posiada swoje miejsce w systemie i nie można go połączyć ani z rekordem WYDZIAŁ, ani z rekordem PROJEKT. Przemawia za tym również fakt, że do rekordu PRACOWNIK wymagany jest dostęp bezpośredni, która to cecha zniknęłaby po konsolidacji. Ostatni argument - rekord PRACOWNIK jest najczęściej używanym rekordem w bazie i stosunkowo często zmienia się (w sensie takim, że przybywają nowe i znikają stare rekordy). To powodowałoby każdorazowo konieczność aktualizacji tablicy w rekordach WYDZIAŁ i PROJEKT. Reasumując, w przypadku programu 2 nie można poprawić struktury danych, aby go przyspieszyć. Zauważmy jeszcze, że jest to program wsadowy, więc jego realizacja nie musi być natychmiastowa. Efekt poprawy struktury bazy danych dla powyższego przykładu przedstawia zał. 6.7.

Podobne postępowanie można powtórzyć dla przykładu pokazywanego w rozdziale 4. Przedstawimy tylko efekt końcowy - poprawioną strukturę bazy (zał. 6.8). Czytelnikom pozostawiamy sprawdzenie zasadności przyjętych rozwiązań.

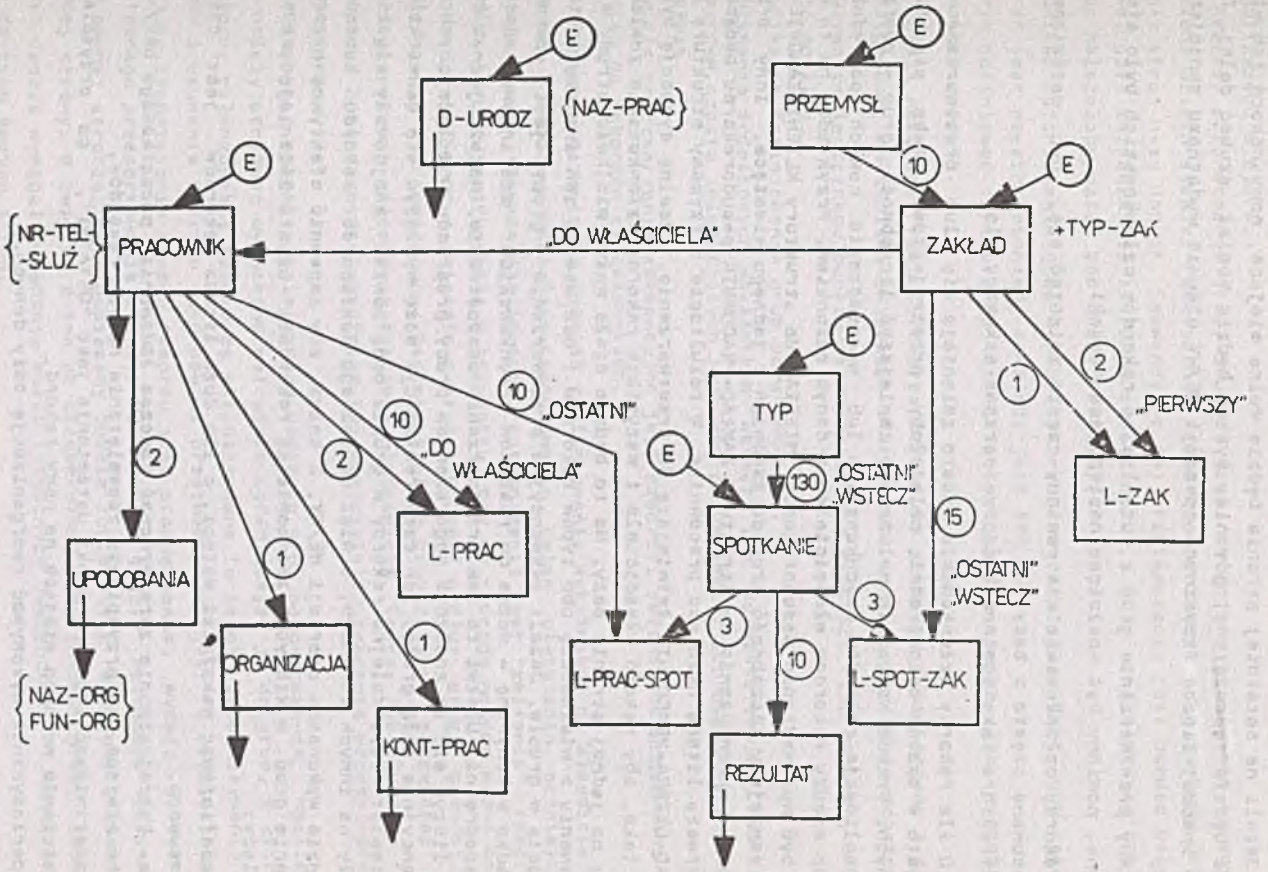
6.5. Optymalizacja grup logicznych

Jak widać było w rozdziale 6.3, poprawianie wstępnej struktury bazy polegało na ustalaniu właściwych parametrów grup logicznych czy w ogóle rezygnacji z tych grup. Zależało to od wielu czynników - typu przetwarzania, wielkości rekordów, częstotliwości ich użycia. Spróbujmy wyciągnąć teraz bardziej ogólne wnioski dotyczące definiowania grup logicznych.

Rozpatrzmy kilka czynników. Po pierwsze - liczebność grupy. Grupy z bardzo dużą liczbą rekordów członkowskich są raczej uciążliwe do przetwarzania. W grupach tych czas przeczytania wszystkich rekordów członkowskich wydłuża się, wymagane jest do tego wykonanie wielu operacji fizycznych WE/WY (gdyż rekordy z reguły nie mieszczą się na jednej stronie). Zwiększy się także koszt zapamiętania nowego rekordu. System musi przeszukiwać bowiem kolejne strony aż do znalezienia wolnego miejsca na nowy rekord. Ostatnią wadę można usunąć definiując w takich grupach logiczny punkt wstawienia jako OSTATNI. Oznacza to, że z rekordu właściciela jest wskaźnik do ostatniego rekordu członkowskiego z grupy. Aby zapamiętać nowy rekord, system wykona więc tylko 2 dostępy WE/WY, celem pobrania strony z właścicielem i strony z rekordem ostatnim.



Rys. 6.7. Przykład 6.4 - rozwiązanie końcowe



Rys. 6.8. Przykład 4.6 - rozwiązanie końcowe

Jeżeli na ostatniej stronie będzie wolne miejsce, nowy rekord tam właśnie będzie zapamiętany, gdy nie, system będzie musiał szukać dalej. Tak czy inaczej liczba fizycznych operacji WE/WY ulegnie wydatnemu zmniejszeniu.

Aby przetwarzanie grup z dużą liczbą rekordów członkowskich było efektywne, powinny być spełnione następujące warunki:

- rekordy członkowskie nie powinny często zmieniać się, być wstawiane i usuwane często z bazy,
- struktura taka powinna być przetwarzana sekwencyjnie.

O ile rekordy członkowskie często zmieniają się lub przetwarzane są często w sposób bezpośredni, należałoby przeprojektować taką strukturę danych. Przede wszystkim należałoby zmniejszyć liczebność grup - poprzez konsolidację rekordów członkowskich lub wprowadzenie nowych pośrednich grup między rekordem właścicielem a danym członkiem. Przykładem na to może być wprowadzenie pseudorekordu MIESIĄC do struktury KLIENT-ZAMÓWIENIE, co zmniejszy liczebność grup do zamówień z jednego miesiąca. Inny przykład to wprowadzenie do struktury ZAKŁAD-PRACOWNIK pseudorekordu będącego pierwszą literą nazwiska pracownika. W rezultacie otrzymamy strukturę ZAKŁAD-LITERA-PRACOWNIK łatwiejszą do przetwarzania. Idealną sytuacją byłaby taka, aby rekord właściciela i wszystkie rekordy członkowskie znalazły się na jednej stronie bazy. Na to będzie miała wpływ wielkość strony w porównaniu z wielkością obu typów rekordów i oczywiście ich średnią liczebnością w grupie. Jeżeli sekwencyjne przetwarzanie grupy jest operacją rzadko wykonywaną - można do rekordów członkowskich zdefiniować dostęp bezpośredni. Ułatwi to nam przetwarzanie bezpośrednie. Należy jednak wtedy liczyć się ze znacznym pogorszeniem pracy programów przeglądających sekwencyjnie całą grupę. Ich czas realizacji może wydłużyć się nawet kilkadziesiąt razy. Kolejne rekordy w grupie będą bowiem znajdowały się z reguły na innych stronach, a więc do każdego dostępu do rekordu konieczne będzie wykonanie operacji WE/WY. Z kolei, aby zapewnić efektywne przetwarzanie grup, w których członkowie są rekordami często zmieniającymi się, należy:

- zdefiniować wskaźniki wstecz, o ile duża liczba rekordów jest często usuwana,
- na każdej stronie zarezerwować podczas ładowania początkowego bazy wolne miejsce, co przyspieszy zapamiętanie nowych rekordów,
- zdefiniować logiczny punkt ustawienia jako "OSTATNI", co przyspieszy szukanie wolnego miejsca na nowy rekord,
- periodycznie wykonywać reorganizację bazy danych.

Kolejną sprawą, o której należałoby powiedzieć, jest definiowanie różnych grup z jednakowym rekordem członkowskim. Sytuacja taka, aczkolwiek niekiedy niezbędna, jest niekorzystna, o ile weźmiemy pod uwagę fizyczne

rozmieszczenie danych. Tylko w jednej grupie może być zachowana strategia najbliższego miejsca.

Rekordy członkowskie grupy drugiej (ang. secondary sets) nie są umieszczone blisko siebie. Praktycznie rzecz biorąc, grupa taka przebiega przez wiele stron bazy danych. Sekwencyjne jej przetwarzanie jest bardzo uciążliwe, gdyż zwiększa się liczba operacji WE/WY. Przy definiowaniu takich grup należałoby wziąć pod uwagę następujące zasady:

- unikać definiowania grup o wspólnym rekordzie członkowskim, gdy rekord ten jest często zmieniany oraz gdy jest ich dużo.
- nie definiować takich grup, które używane byłyby rzadko lub w programach wsadowych.

Jeżeli natomiast rekordy tworzą strukturę sieciową i nie możemy z niej zrezygnować, powinniśmy zwiększyć efektywność ich przetwarzania poprzez:

- używanie logicznego punktu wstawienia jako PIERWSZY lub OSTATNI, o ile to możliwe.
- unikanie definiowania grup posortowanych.
- używanie dodatkowych wskaźników "wstecz" lub "do właściciela" dla rekordów często zmieniających się, aby przyspieszyć ich usuwanie i zakładanie,
- o ile przetwarzanie sekwencyjne jest rzadkie, definiować bezpośredni dostęp do rekordów członkowskich.

Istotne znaczenie ma również zdefiniowanie wskaźników do właściciela w grupach o wspólnym rekordzie członkowskim, celem ułatwienia późniejszej reorganizacji. Zasadniczą częścią reorganizacji jest "zwijanie" rekordów wzdłuż struktury grupowej. W przypadku, gdy do jednego rekordu można dojść poprzez wiele grup - zwijanie musi się odbywać wzdłuż drogi głównej, a więc pewnej wybranej grupy. Dla pozostałych grup należy zapamiętać dla każdego rekordu członkowskiego jego właścicieli (adres lub klucz) w tych grupach. Zdefiniowanie wskaźnika "do właściciela" znacznie przyspieszy tę procedurę.

Odrębnego potraktowania wymaga używanie grup posortowanych. Możliwość tę należy stosować wyjątkowo ostrożnie ograniczając ją do grup, w których rekordy członkowskie rzadko są aktualizowane (w sensie zapamiętywania nowych i usuwania starych rekordów) i przetwarzanie jest z reguły sekwencyjne. Zapamiętanie nowego rekordu jest pracochłonne; wymaga od systemu wstępnego przesortowania kluczy. Nowy rekord będzie wstawiony fizycznie na pewnej stronie, choć powiązany będzie logicznie z rekordami z zupełnie innej strony. W związku z tym po kilkukrotnej modyfikacji grupy jej odczyt będzie wymagał wykonania wielu fizycznych dostępów WE/WY, często zresztą do tych samych stron.

Bez sensu jest definiowanie dużych grup sortowanych na potrzeby programów wsadowych oraz takich, gdzie członek ma zdefiniowany dostęp bezpośredni. Lepiej przecież wtedy sortować na zewnątrz bazy danych, za pomocą

GRUPY					
NAZWA	WŁAŚCICIEL	CZŁONKOWIE	LOGICZNY PUNKT WSTAWIENIA	TYP	WSKAŹNIKI

Rys. 6.9. Formularz "GRUPY"

własnych czy systemowych programów i procesorów. Baza danych krócej będzie blokowana przez program. Sekwencyjny odczyt wszystkich rekordów obszaru, wybór rekordów do sortowania i właściwe sortowanie zewnętrzne może być dużo szybsze niż proste na pozór odczytanie już posortowanej grupy. Zauważmy zresztą, że grupę można uporządkować niekoniecznie za pomocą zdefiniowania w niej porządku sortowanego. Często zdarza się, że określenie logicznego punktu wstawienia jako OSTATNI daje to samo, zwłaszcza gdy kluczem jest data (czas) i dane spływają równomiernie w czasie. W stabilnych grupach, tzn. w takich, w których liczba rekordów członkowskich rzadko ulega zmianie, można posortować rekordy przed zapamiętaniem ich w bazie. I znów mimo zdefiniowania logicznego punktu wstawienia np. jako NASTĘPNY grupa będzie właściwie posortowana.

Omówmy jeszcze jeden przypadek grup. Załóżmy, że grupy będą posiadały dużą liczbę rekordów członkowskich. Załóżmy też, że pod adresem takich grup są skierowane specyficzne żądania równoległego przetwarzania. Chodzi tutaj o równoległe przetwarzanie zarówno 2 różnych wystąpień tej samej grupy, jak i o przetwarzanie 2 wystąpień różnych grup (np. rys.5.11). Wtedy też najlepszym projektowym rozwiązaniem jest zdefiniowanie grup z tablicą wskaźników. Aby wybrać rekord członkowski wspólny dla różnych wystąpień grup, wystarczy porównać dwie tablice adresowe. Porównywanie to będzie wykonywane w pamięci operacyjnej, a więc będzie trwało bez porównania krócej niż pobieranie i sprawdzanie kolejnych rekordów z bazy. Jednak przy grupach o małej liczbie członków może okazać się, że grupy o strukturze łańcucha są szybsze. Wynikać to będzie z zastosowania w nich strategii najbliższego miejsca minimalizującej czas dostępu do całej grupy.

Omówione powyżej zasady pomagają w definiowaniu i we właściwym wyborze ich parametrów. Opisy wszystkich grup logicznych bazy danych można umieścić na specjalnym formularzu "GRUPY" (zał. 6.9), który ułatwi późniejsze opracowanie źródłowej wersji schematu bazy i będzie udokumentowaniem tego etapu pracy.

7. PROJEKTOWANIE STRUKTURY FIZYCZNEJ

Przystępujemy do kolejnego etapu w projektowaniu bazy danych. Jest nim powiązanie danych z fizycznymi urządzeniami - zbiorami, a więc projektowanie struktury fizycznej. Na etapie tym chodzi przede wszystkim o obliczenie wielkości obszarów, wybór wielkości strony, wybór rodzaju stron nadmiarowych i rodzaju zabezpieczania obszarów. Również na tym etapie należy pogrupować rekordy w obszary oraz wybrać metody bezpośredniego dostępu do rekordów. W przypadku zdecydowania się na dostęp poprzez procedurę obliczeniową należy wybrać jedną ze standardowych procedur dostarczonych przez producenta lub napisać własną. Ogólnie rzecz biorąc, można ten etap podzielić na następujące kroki:

- wybór metod bezpośredniego dostępu,
- podział rekordów na obszary,
- obliczenie wielkości obszarów i stron.

Prawidłowy wybór parametrów struktury fizycznej bazy zapewni jej efektywną eksploatację. Zdajemy sobie sprawę, że koszt pracy systemu zależy w głównej mierze od poprawnego projektu struktury logicznej. Jednak projektowania struktury fizycznej nie można lekceważyć, gdyż również w nim "drzemią" potencjalne możliwości skrócenia czasu programów. A to, jak już wcześniej powiedzieliśmy, jest podstawowym celem naszej pracy.

7.1. Wybór metod bezpośredniego dostępu do rekordów

Projektując logiczną strukturę bazy określiliśmy jedynie, że do pewnych rekordów wymagany jest dostęp bezpośredni, nie zastanawiając się, w jaki sposób będzie on realizowany. Na tym etapie musimy zdecydować, które ze standardowych możliwości dopuszczalnych przez system zarządzania bazą danych wybierzemy. Oczywiście różne systemy zarządzania różnią się zarówno liczbą, jak i rodzajem metod bezpośredniego dostępu. Omówimy 3 najbardziej typowe z nich:

- dostęp direct (tzn. za pomocą adresu),
- dostęp randomizacyjny (poprzez procedurę obliczeniową),
- dostęp indeksowo-sekwencyjny.

Porównamy wady i zalety tych metod, gdyż to w głównej mierze zadecyduje o wyborze właściwego dla danego typu rekordu rozwiązania. Przy porów-

Metoda	Zalecany typ przetwarzania	Ograniczenia na klucz	Wykorzystanie pamięci	Aktualizacja danych	Dostęp
DIRECT	bezpośrednie lub sekwencyjne	- wartości kluczy powinny być znane (lub chociaż ich rząd wielkości) - klucze powinny przybierać dużą część zbioru wartości	bardzo wysokie	najszybszy	najszybszy 1 operacja WE/WY
CALC	bezpośrednie	brak	niskie (konieczność stron nadmiarowych)	szybki (aktualizacja łańcuchów obliczeniowych)	szybki (przeszukiwanie łańcucha obliczeniowego) 1-2 operacje WE/WY
INDEKSOWO-SEKWENCYJNA	bezpośrednie lub sekwencyjne	brak	- bardzo wysokie - konieczność stron nadmiarowych i obszaru na indeksy	stosunkowo szybki (tworzenie łańcuchów rekordów ze stron głównych i nadmiarowych)	stosunkowo szybki (przeszukiwanie stron indeksowych) 2-3 operacje WE/WY

Porównanie metod bezpośredniego dostępu

nywaniu tych metod weźmiemy pod uwagę takie czynniki i parametry, jak: liczbę rekordów, budowę klucza i typ przetwarzania. Wyniki tego porównania są zebrane w tabeli (zał. 7.1).

7.1.1. Dostęp direct

Jest najszybszym typem dostępu, przydatnym zarówno do przetwarzania bezpośredniego, jak i sekwencyjnego. Odczyt rekordu zapamiętanego tą metodą jest zdecydowanie najszybszy. Do jego wykonania wystarczy zawsze tylko jedna operacja WE/WY. Również zapamiętanie nowego rekordu tego typu jest szybkie pod warunkiem, że adres, którego chcemy użyć, jest wolny. Gdy jest on zajęty, procedura zapamiętywania rekordu kończy się błędem. Adres w tej metodzie może mieć sens fizyczny bądź logiczny, co zależy głównie od SZBD. Adres fizyczny to fizyczne parametry lokacji rekordu w zbiorze, a więc w szczególności nr dysku, nr ścieżki (pozycji) i nr sektora. Adres logiczny, używany w bazach danych typu CODASYL, składa się z numeru strony i numeru rekordu na stronie. Właściwy adres fizyczny rekordu znajduje tu sam SZBD.

Przy stosowaniu metody direct pamięć masowa wykorzystywana jest najefektywniej. Praktycznie potrzebujemy w niej tyle miejsca, ile wprowadzamy rekordów. Nie ma stron nadmiarowych, niepotrzebny jest żaden zapas stron usprawniający metodę. Odbywa się to kosztem konieczności znajomości dokładnej mapy bazy, dokładniej mówiąc - znajomości adresów wszystkich rekordów typu direct. Jest to uciążliwe i sprawia wiele kłopotów, zwłaszcza gdy zbiór rekordów direct jest liczny i gdy jest on często aktualizowany. Oczywiście kontrolę nad rozmieszczaniem rekordów direct można wykonywać w sposób automatyczny. Może to być realizowane przez procedurę programową przetwarzającą klucz rekordu na adres (klucz w tym przypadku powinien jednoznacznie definiować rekord). Aby zastosować metodę direct w takich rekordach, należy z góry znać rozkład wartości klucza lub ich zakres (rząd) w przypadku kluczy numerycznych. Co więcej, aby zapewnić w miarę ścisłe zapełnienie obszaru rekordami direct, wymagane jest, aby przeważająca część zbioru wartości klucza faktycznie występowała w rekordach. Wtedy unikniemy pustych miejsc w bazie i uzyskamy lepsze wykorzystanie przestrzeni adresowej. Reasumując, metodę direct można stosować dla rekordów o kluczach numerycznych sekwencyjnie przebiegających pewien zbiór wartości. Przykładem mogą być rekordy, w których kluczem lub jego częścią jest data, czas, numer pracownika czy urządzenia. Z reguły też metody direct używa się do zapamiętywania pojedynczych rekordów nagłówkowych będących np. właścicielami pseudogrup lub samodzielnymi stałymi tablicami danych.

7.1.2. Dostęp randomizacyjny (calc)

Dostęp poprzez procedurę obliczeniową jest pewnym uogólnieniem metody direct. Różnica polega na tym, że nie operujemy tu adresem rekordu, a tylko jego kluczem. Procedura obliczeniowa każdorazowo przekształca ten klucz

na adres w dopuszczalnej przestrzeni adresowej. W przypadku baz danych typu CODASYL adresem logicznym rekordu calc jest numer strony i numer łańcucha obliczeniowego. Łańcuch obliczeniowy grupuje rekordy, którym algorytm przydzielił ten sam adres.

Dostęp do rekordu poprzez procedurę obliczeniową jest dostępem szybkim, do jego zrealizowania wymagany jest 1 lub 2 dostępy fizyczne (1 - gdy rekord umieszczony jest na stronie głównej, 2 - gdy na nadmiarowej lub obcej). Również zapamiętanie nowego rekordu jest szybkie, aczkolwiek z uwagi na konieczność przetwarzania łańcuchów obliczeniowych jest wyraźnie wolniejszy niż w metodzie direct. Czas przetwarzania tych łańcuchów, celem znalezienia wolnego miejsca, zależy od tego, czy dopuszczalne są rekordy - duplikaty ze względu na klucz oraz jakie wskaźniki służą do zamknięcia łańcucha obliczeniowego (mogą być zdefiniowane wskaźniki "w przód" lub "w przód" i "wstecz"). Przestrzeń adresowa jest w tej metodzie dużo gorzej wykorzystana niż przy dostępie direct. Zależne jest to zresztą od samego algorytmu randomizacyjnego. Idealny algorytm byłby taki, który równomiernie rozłożyłby rekordy w całej przestrzeni adresowej. Innymi słowy - każdy rekord byłby na stronie głównej, a w każdym łańcuchu obliczeniowym byłby dokładnie 1 rekord. Wtedy do odczytu rekordu wystarczyłby zawsze 1 dostęp fizyczny, tak jak w przypadku direct. Takich idealnych algorytmów randomizacyjnych brak. Aby zapewnić w miarę sprawną obsługę rekordów typu calc, należy przyjąć wystarczający zapas wolnego miejsca w obszarze. Z praktyki wiemy, że w przypadku małych obszarów zapas ten powinien wynosić prawie 50%, w przypadku obszarów dużych rzędu 10-15%. Dystrybucja rekordów zależy jeszcze od wskaźnika załadowania początkowego stron.

Przy ustaleniu tego wskaźnika (na pewną wartość, np. 70%) podczas późniejszego przetwarzania mamy dużą szansę, że rekord zostanie umieszczony na stronie głównej, a nie na stronie nadmiarowej.

Z tą drugą sprawą związany jest kolejny problem - rozwiązywanie konfliktów. Procedura rozwiązywania konfliktów jest integralną częścią każdego algorytmu randomizacyjnego. Zauważmy, że w naszym przypadku procedura powinna obsługiwać konflikty w każdym z trzech przypadków - brak stron nadmiarowych, strony nadmiarowe globalne lub strony nadmiarowe lokalne. Zapewniają to oczywiście procedury standardowe dostarczane przez producenta, ale w przypadku pisania własnych procedur należy o tym pamiętać.

Czy pisać własne procedury obliczeniowe? Oczywiście, system z reguły dopuszcza taką możliwość. Są powszechnie znane częściej stosowane algorytmy randomizacji polegające na kolejnym wydzielaniu, analizie znaków, podnoszeniu do kwadratu, operacji arytmetycznych na binarnej zawartości znaków i inne.

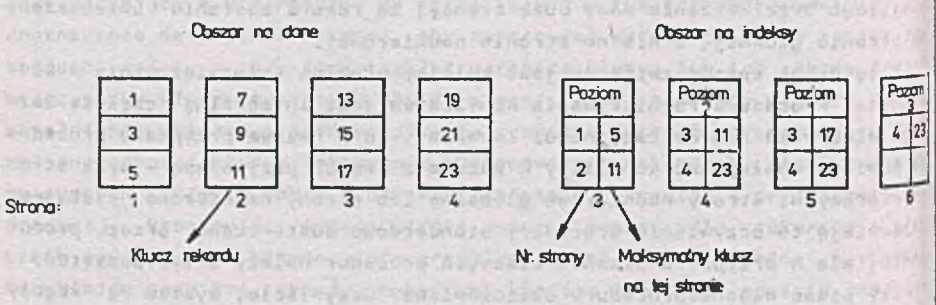
Można wymyśleć także własny algorytm. Czy to jednak opłaca się? Oczywiście, niekiedy może to się opłacać, np. jeżeli uzyskamy bardzo dobry rozkład dla jakiegoś klucza, a rekordów tego typu jest dużo. Należy pa-

miętać jednak o tym, że procedura obliczeniowa zostanie dołączona do systemu zarządzania bazy, zwiększając go w sposób istotny. Teoretycznie dla różnych rekordów można opracować różne procedury obliczeniowe, ale praktycznie takie rozwiązanie nie ma sensu. System Zarządzania Bazą Danych byłby wtedy zbyt wielki, mógłby w ogóle nie mieścić się w pamięci operacyjnej. Tak więc wydaje się, że poza uzasadnionymi przypadkami należy stosować raczej tylko jedną z procedur standardowych. Co prawda dla różnych kluczy będzie ona lepsza lub gorsza, ale różnice nie będą zbyt wielkie. Średnio można przyjąć, że do wykonania dostępu do rekordu potrzeba będzie wykonać 1,5 dostępów fizycznych.

Reasumując, metoda alokacji rekordów poprzez procedurę obliczeniową jest wygodnym narzędziem, zwłaszcza do przetwarzania bezpośredniego. Nadaje się do stosowania do wszystkich kluczy, nawet takich, dla których nie znamy wszystkich przyszłych wartości czy zakresu.

7.1.3. Metoda indeksowo-sekwencyjna

Podstawowym problemem przy zastosowaniu metody indeksowo-sekwencyjnej jest właściwe zaprojektowanie obszaru na indeksy. Obszar na indeksy z punktu widzenia SZBD jest właściwie taki jak obszar na dane. Różnica jest tylko taka, że na kolejnych stronach tego obszaru, w części przeznaczonej na dane zapamiętywane są nie rekordy, a informacje o maksymalnych kluczach na każdej stronie. Strony 1 i 2 w obszarze indeksowym są wykorzystywane przez system, np. do zapamiętania strony indeksowej najwyższego poziomu i innych.



Rys. 7.2. Organizacja danych w metodzie indeksowo-sekwencyjnej

Zauważmy, że strony w obszarze indeksowym mogą reprezentować różny poziom. Na stronach z poziomu pierwszego zapamiętane są informacje o maksymalnych kluczach rekordów ze stron z obszaru na dane, na stronach o wyższych poziomach przechowywane są informacje o maksymalnych kluczach na stronach indeksowych.

W porównaniu ze stronami na dane różny też musi być nagłówek górny tych stron. Wymagane jest w nim zapamiętanie takich informacji, jak: liczba punktów wejścia na stronie, numer poziomu, który ta strona reprezentuje i długość każdego punktu wejścia. Punkt wejścia w tym przypadku oznacza parę "numer strony" i "klucz". Jego długość może być różna dla różnych typów rekordów (głównie z uwagi na różną długość klucza). Liczba punktów wejścia zależy oczywiście od wielkości strony. Celem projektanta jest takie przyjęcie wielkości strony, aby przy zadanej liczbie rekordów minimalizować liczbę poziomów indeksowych. Zadanie to praktycznie zostało rozwiązane [1] metodą mnożników Lagrange'a i dało następujący rezultat:

$$y = v^{\frac{1}{n+1}} \tag{7.1}$$

gdzie:

y - długość listy (liczba punktów wejścia na stronie indeksowej),
v - liczba rekordów,
n+1 - liczba poziomów.

Z reguły liczbę poziomów ustala się co najwyżej na 3. To pozwala nam na zminimalizowanie liczby dostępów fizycznych przy pobieraniu rekordu. Dostęp indeksowo-sekwencyjny jest więc stosunkowo szybkim rodzajem dostępu bezpośredniego. Do jego zrealizowania potrzebne jest wykonanie (przy podanych założeniach) średnio 2-3 dostępów fizycznych. Oczywiście w miarę wzrostu nieuporządkowania obszaru warunki eksploatacji mogą pogorszyć się z uwagi na umieszczanie tych rekordów na stronach nadmiarowych (p. rozdz. 3.2). Z tego też powodu zapamiętywanie nowego rekordu będzie trwało niekiedy dłużej (w porównaniu z innymi metodami bezpośredniego dostępu). Jeżeli chodzi o wykorzystanie przydzielonej pamięci, to jest ono bardzo wysokie. Należy jednak pamiętać o pewnej systemowej "nadkładce" wymaganej przez tę metodę. Jest nią konieczność zdefiniowania stron nadmiarowych oraz zdefiniowanie dodatkowego obszaru na indeksy. Tym niemniej metodę tę opłaca się stosować głównie dla bardzo dużych zbiorów rekordów. Przetwarzanie zarówno swobodne (bezpośrednie), jak i sekwencyjne jest równie łatwo wykonywane. Metoda ta, podobnie jak w alokacji poprzez procedurę obliczeniową, nie nakłada żadnych ograniczeń na klucz. W tym miejscu wymagana jest jednak uwaga - w czasie ładowania początkowego rekordów zalecane jest zapamiętanie jak największej liczby rekordów oraz stosowanie wskaźnika ładowania początkowego stron. To usprawni znacznie późniejszą eksploatację, głównie wprowadzenie nowych rekordów.

7.2. Podział rekordów na obszary

Systemy zarządzania bazami pozwalają na dużą swobodę w łączeniu rekordów w obszary. Jest to bardzo duży postęp w porównaniu z tradycyjnymi sys-

temami przetwarzania danych, gdzie z reguły w zbiorze można było pamiętać tylko jeden typ rekordu. Co więcej, systemy ZBD dopuszczają stosowanie różnego typu alokacji w tym samym obszarze. Praktycznie oznacza to, że w jednym obszarze mogą być pamiętane zarówno rekordy direct, calc a nawet indeksowo-sekwencyjne (nie mówiąc już o rekordach zapamiętywanych poprzez grupę). Jeżeli chodzi o rekordy indeksowo-sekwencyjne, to właśnie w ich przypadku istnieje jedyne formalne ograniczenie - w jednym obszarze nie można przechowywać 2 typów rekordów pamiętanych metodą indeksowo-sekwencyjną. Łączenie ich z rekordami innych typów jest co prawda dozwolone, ale nie zaleca się zbyt często stosować tego rozwiązania z uwagi na sporą trudność w aktualizacji rekordów indeksowo-sekwencyjnych. Projektant bazy ma więc dużą swobodę w podejmowaniu decyzji o budowie obszarów. W krańcowych przypadkach może on się decydować nawet na jeden tylko obszar (zawierający wszystkie zdefiniowane rekordy) lub na zdefiniowanie tylu obszarów, ile jest typów rekordów w bazie. Co więcej, ten sam rekord może być umieszczony w różnych obszarach, co łatwo może być sterowane wewnątrz programu. Umożliwi to rozróżnienie podsystemów bazujących na podobnych rozwiązaniach (np. systemy planowania na różnych wydziałach produkcyjnych). Podejmując decyzję o przydzieleniu rekordów do odpowiednich obszarów powinno mieć się na uwadze efektywność pracy systemu. Z reguły projektujemy system dla konkretnej konfiguracji komputerowej o znanych parametrach fizycznych. Celem projektowania obszarów powinno być jak najlepsze wykorzystanie posiadanych urządzeń (pamięć zewnętrzna). Stąd powinniśmy wziąć pod uwagę szereg czynników, jak np. typ przetwarzania danych, ochronę danych przed niepowołanym dostępem, zabezpieczenie integralności danych, potrzeby przyszłej reorganizacji i inne.

Postarajmy się pokrótce je omówić.

Pierwszoplanowym problemem, jaki od razu pojawia się, jest pytanie, czy rekordy tworzące grupę logiczną lokować w jednym czy wielu obszarach. Ponieważ są argumenty zarówno za, jak i przeciw, nie odpowiemy wprost na to pytanie, lecz przedstawimy wady i zalety każdej z metod.

Umieszczenie rekordu właściciela i rekordu członkowskiego w jednym obszarze może być bardzo efektywne, szczególnie gdy rekord właściciela i jego członkowie byłyby na 1 stronie. Jest to szczególnie wskazane, gdy transakcje w czasie rzeczywistym sekwencyjnie przeglądają całą grupę. Taki rozkład rekordów zapewni strategia najbliższego miejsca. Nie ma ona jednak zastosowania, gdy rekord członkowski ma dostęp bezpośredni lub gdy ten sam rekord jest zdefiniowany jako członek w różnych grupach. Jednak nawet gdy strategia najbliższego miejsca ma zastosowanie, nie zawsze można zapewnić umieszczenie wszystkich rekordów członkowskich na jednej stronie. Zależy to oczywiście od liczby rekordów w grupie i od ich wielkości. Tak czy inaczej, zmuszeni jesteśmy do definiowania raczej większych stron w obszarze, co ma z kolei swoją ujemną stronę w przetwarzaniu wieloprogramowym. Zaletami rozwiązania umieszczenie całej grupy w jednym obszarze

są niewątpliwie przyspieszenie przydziału obszarów, łatwiejsze ładowanie początkowe i reorganizacja, gdyż tylko jeden obszar będzie przydzielany czy zamykany. Reorganizacja będzie przebiegać łatwo zwłaszcza wtedy, gdy rekordy nie będą powiązane ze sobą w strukturę sieciową. Tego jednak nie zawsze można uniknąć. Wtedy też pojawią się nowe problemy związane z blokowaniem informacji przez różne programy czy niewłaściwym wykorzystaniem przydzielonego miejsca w pamięci. Ujemnymi stronami tej metody są z kolei: mniejsza elastyczność w wykorzystaniu urządzeń (gdy mamy różne urządzenia, np. różne dyski, można przydzielać częściej używane rekordy do szybszych urządzeń) i gorsze wykorzystanie pamięci w przypadku różnic w stopniu wykorzystania danego typu rekordu. Na przykład, jeżeli właściciel grupy jest rekordem calc, to może zdarzyć się, że rzadko modyfikowane rekordy członkowskie wymuszą na systemie (poprzez zajęcie pewnych adresów w pamięci) umieszczenie często zmieniających się rekordów - właścicieli na stronach nadmiarowych.

Rozpatrzmy drugą możliwość - umieszczenie rekordu właściciela i rekordów członkowskich w dwóch odrębnych obszarach. Oczywiście w niektórych zastosowaniach przetwarzanie takich grup będzie wymagać wykonania więcej fizycznych dostępów do stron niż poprzednio. Strategia najbliższego miejsca w tym przypadku nie ma w ogóle zastosowania. Ładowanie początkowe takiej struktury będzie uciążliwe, gdyż system wprowadzając do bazy nowy rekord członkowski musi każdorazowo poszukiwać wolnego miejsca na ten rekord, a robi to zawsze od początku obszaru. Wadę tę można wyeliminować przez pamiętanie numeru ostatnio używanej strony (w programie lub w SZBD). Mimo tych wad rozwiązanie to jest racjonalne w niektórych przypadkach; zapewnia lepsze wykorzystanie urządzeń i lepsze zapełnienie strony, łatwiejsze rozwiązywanie konfliktów (blokowań), umożliwia różne zabezpieczanie różnych rekordów. Reorganizacja wymaga zamknięcia zamiast jednego - dwóch czy więcej obszarów, ale niekiedy może być to sytuacja opłacalna. Na przykład gdy reorganizujemy grupę manualną, to po odłączeniu rekordów członkowskich można swobodnie modyfikować jeden typ rekordu bez wpływu na pozostały. Nie bez znaczenia jest również możliwość definiowania mniejszych stron, co ułatwi przetwarzanie wieloprogramowe. Zasadniczymi wadami tej metody są jednak: konieczność przydzielania do programu dwóch obszarów i zwiększenie liczby fizycznych dostępów przy przetwarzaniu całej grupy.

Podobne czynniki, o których była mowa powyżej, decydują o łączeniu w obszar rekordów nie związanych ze sobą zależnością grupową. Oczywiście zwiększenie liczby typów rekordów w obszarze spowoduje gorsze wykorzystanie stron na dane, gdyż trudno będzie dobrać, optymalną w zależności od wielkości rekordów, wielkość strony. Jednak za łączeniem rekordów przemawiają często inne aspekty, jak np.: wspólna ochrona przed zniszczeniem czy przed niepowołanym dostępem. Właśnie na poziomie obszaru możemy definiować różne sposoby zabezpieczania danych (choć ostatnio wprowadza się nawet klucze kontrolne już na poziomie rekordu czy nawet jego pól). Obszar

z danymi wymagającymi specjalnej ochrony może być przydzielony na specjalnym urządzeniu z odpowiednimi kluczami do czytania i do zapisu. Również ochrona integralności danych w bazie odbywa się na poziomie obszaru. Na taśmę śladową systemu czy do odpowiadającego jej zbioru w pamięci masowej zapisuje się kopie zmienianych przez program stron obszaru.

Różne obszary możemy w różny sposób zabezpieczać. Na przykład może zdarzyć się przypadek, że dane z jakiegoś obszaru będą odczytywane w czasie rzeczywistym, a ich aktualizacja będzie wykonywana w trybie wsadowym. Dla takiego obszaru nie ma sensu definiować zapisu na taśmę śladową. Transakcje będą pracowały szybciej, a aktualizację obszaru można realizować bez zabezpieczania bezpośrednio po wykonaniu dumpu całej bazy. Szczególnie ważne i często modyfikowane obszary w niektórych systemach operacyjnych można zdefiniować jako zbiory dualne. Oznacza to, że obszar będzie przechowywany równocześnie w dwóch identycznych kopiach na różnych fizycznie urządzeniach. W razie awarii jednego z nich lub wykrycia błędu w jednej kopii można natychmiast wykorzystać do pracy drugą kopię. W metodzie tej należy jednak pamiętać o wzroście czasu przetwarzania z uwagi na podwojenie liczby fizycznych dostępów.

Bardzo ważnym czynnikiem, jaki należy wziąć jeszcze pod uwagę, jest częstość i przebieg reorganizacji danych w przyszłości. Rekordy, które częściej będziemy reorganizować (np. rekordy calc czy indeksowo-sekwencyjne), powinny być w miarę możliwości umieszczane samodzielnie w obszarach. Również jeżeli spodziewamy się, że pewna grupa będzie reorganizowana częściej niż inne, nie powinniśmy łączyć tej grupy w jednym obszarze z innymi rekordami. Przyspieszy to znacznie późniejszą reorganizację. Biorąc pod uwagę potrzeby reorganizacji powinniśmy decydować się na jak najprostszą strukturę rekordów w obszarze (co najwyżej na strukturę hierarchiczną).

Innym czynnikiem wpływającym na definiowanie obszaru jest jego planowana wielkość. To oczywiście jest podyktowane liczbą i wielkością rekordów w obszarze. Jeżeli duży obszar przetwarzany jest sekwencyjnie, zajmie to odpowiednio dużo czasu. Co prawda, sekwencyjne przetwarzanie obszarów jest domeną raczej programów wsadowych a nie transakcyjnych w czasie rzeczywistym, lecz nic nie stoi na przeszkodzie, aby przyspieszyć te właśnie programy. Wystarczy zmniejszyć obszary, co można uczynić choćby przez rozbicie jednego dużego obszaru na kilka mniejszych. W takim przypadku nie opłaca się łączyć w jednym obszarze różnych typów rekordu. Przetwarzanie sekwencyjne wszystkich rekordów wybranego typu będzie teraz szybsze, gdyż wymagać będzie przeczytania mniejszej liczby stron obszaru.

Podobnie jak przy omawianiu grup logicznych można też wziąć pod uwagę stabilność rekordów. Nie ma sensu łączyć w jednym obszarze rekordów rzadko przetwarzanych z często modyfikowanymi. Po prostu rekordy rzadko używane pogorszą warunki eksploatacji tych często używanych. Zajmą ich miejsce w pamięci, zmuszą system do wykonania większej pracy przy szukaniu

wolnego miejsca w pamięci i mogą spowodować konieczność umieszczania rekordów na stronach nadmiarowych.

7.3. Obliczanie wielkości obszaru

Celem tego rozdziału jest pokazanie procedur ułatwiających obliczanie fizycznych parametrów danych. Głównie chodzi tu o obliczenie wielkości obszaru celem zarezerwowania w pamięci masowej odpowiedniej wielkości zbioru na ten obszar. Obszar, jak wiemy, podzielony jest na strony; liczba stron obszaru w oczywisty sposób zależy od wielkości strony i przewidywanej liczby danych (rekordów). Projektant ma przeważnie pewną dowolność przy wyborze wielkości stron. Systemy zarządzania bazą dają w tym miejscu właściwie tylko jedno ograniczenie, aby strona była całkowitą wielokrotnością sektora (ścieżki). Mając to na uwadze projektant może wybrać zarówno małe, jak i bardzo duże strony. Jakie czynniki należy wziąć przy tym pod uwagę? Omówmy je kolejno.

Przede wszystkim istotna jest wielkość buforów na strony zarezerwowanych w SZBD. Ponieważ bufory te są stale w pamięci operacyjnej, ich wielkość zależy w oczywisty sposób od wielkości dostępnej pamięci operacyjnej. Z kolei, aby ułatwić pracę systemowi w obsłudze tych buforów, można zdefiniować jednakowe strony we wszystkich obszarach lub stosować co najwyżej dwa rozmiary, z których jeden jest dwukrotnie większy niż drugi. Przyjęcie takich zasad usprawni wymianę stron między pamięcią masową a operacyjną, kłopotliwą zwłaszcza w systemach o małych buforach.

Drugi czynnik, jaki należy wziąć pod uwagę, to typ pracy systemu zarządzania bazy. Jeżeli dopuszcza się w systemie pracę wieloprogramową, strony powinny być jak najmniejsze. Dzięki temu zmniejszy się prawdopodobieństwo blokowania tych samych stron przez różne programy. Przy pracy jednoprogramowej korzystniejszym rozwiązaniem jest zdefiniowanie dużej strony. Blokowanie się programów jest niemożliwe, a ograniczy się w ten sposób liczbę operacji WE/WY.

O wielkości stron może decydować również sposób ich przetwarzania. Jeżeli system przetwarza dane przeważnie sekwencyjnie - strony powinny być jak największe. Jeżeli dostęp do danych jest bezpośredni, strony powinny być małe, aby ograniczyć czas transferu potrzebnych danych między pamięcią masową a operacyjną. Należy pamiętać bowiem o tym, że podstawową jednostką biorącą udział w operacji WE/WY jest właśnie strona. Mimo że chcemy odczytać tylko jeden rekord ze strony, cała strona musi być przesłana z bazy do systemowych buforów.

Na wielkość strony ma również wpływ wielkość rekordów. Jest to oczywiste; strona musi być tak duża, żeby w jej polu na dane zmieścił się chociaż 1 rekord. Gdy pewne rekordy powiązane są zależnością grupową i przetwarzanie całej grupy jest w czasie rzeczywistym, należy definiować tak dużą stronę, aby zmieściła jak najwięcej rekordów członkowskich. Idealne

rozwiązanie byłoby takie, że rekord właściciela i jego rekordy członkowskie byłyby zapamiętane na jednej stronie. Oczywiście wielkość strony zależy w tym przypadku od wielkości obu typów rekordów oraz od liczby wystąpień członkowskich w grupie.

Kolejnym czynnikiem, o którym należy pamiętać przy wyborze wielkości strony, jest jak najlepsze jej wykorzystanie (zapełnienie). Wielkość stron powinna być tak dobrana, aby minimalizować liczbę systemowych i nie wykorzystanych słów. Jest to łatwe dla obszaru o jednym typie rekordu. Natomiast gdy w obszarze przechowywać będziemy różne rekordy, do analizy efektywności wykorzystania strony musimy wziąć średnią wielkość rekordu w obszarze. Poniżej pokażemy, jak praktycznie to realizować.

Celem naszym będzie właśnie efektywne wykorzystanie części stron przeznaczonych na dane użytkownika i zapewnienie w miarę znośnych warunków pracy procedurom lokującym rekordy w obszarach przez przyjęcie odpowiedniego zapasu przestrzeni adresowej. Aby ułatwić przeprowadzenie czasochłonnych nieraz obliczeń, a właściwie po to, aby je uporządkować i udokumentować, proponujemy używanie kolejnych formularzy (zał. 7.3-7.8). Zostały one zaprojektowane z myślą o systemie DMS-1100. Stąd wygląd tych formularzy niekiedy uwarunkowany jest rozwiązaniami przyjętymi w tym systemie, jak np. 10-słowowy nagłówek górny strony czy jednosłowowe wskaźniki między rekordami. Wydaje się jednak, że przynajmniej niektóre z tych formularzy mogą być po małej modyfikacji stosowane w innych SZBD. Obliczanie fizycznej struktury bazy będziemy wykonywać w kolejnych krokach, a każdy krok będziemy dokumentować właściwym formularzem.

7.3.1. Obliczanie wielkości części użytkowej rekordu

Procedurę tę musimy wykonać dla wszystkich projektowanych rekordów bazy. Do tego celu może służyć formularz "POLA W REKORDZIE" (zał. 7.3).

Zawiera on informacje ogólne o rekordzie, takie jak kod i nazwa, kod i nazwa obszaru, w którym ten rekord jest umieszczony oraz format rekordu, tzn. czy jest on stałej czy zmiennej długości. Z kolei dla każdego pola elementarnego (pola grupowe umieszczamy na formularzu, lecz nie wykonujemy dla nich obliczeń) wypisujemy kolejno: poziom, nazwę pola występującą w opisie systemu oraz kod tej nazwy w bazie danych, typ pola, opis pola (klauzula PICTURE w języku COBOL). Dla pól powtarzających się wypisujemy maksymalną liczbę powtórzeń, a więc rozmiar tablicy.

Jeżeli jakieś pole załadowane jest stałymi danymi lub znany jest jego zakres wartości, można to zaznaczyć w odpowiedniej rubryce formularza. Typ pola może przybierać następujące wartości: A - alfabetyczne, X - alfanumeryczne, N - numeryczne znakowe i B - numeryczne binarne.

Podstawową sprawą jest obliczenie fizycznej wielkości każdego pola. Zależy to oczywiście od kodu, w jakim przechowywane są znaki oraz od parametrów słowa maszynowego w danym typie komputera, a w szczególności od wielkości słowa, możliwości jego podziału i zasad rządzących alokacją da-

			SCHEMAT				DATA		
			NR KOLEJNY				SPORZĄDZIŁ		
Kod rekordu:		Nazwa rekordu	Obszar		Kod przechowywania d:		WIELKOŚĆ REKORDU W SŁOWACH		
			KOD	NAZWA	Stała długość	Zmienna dług	MIN	MAX	ŚREDNIO
Lp.	Poziom	Nazwa pola			Typ pola	Wielkość pola	PICTURE	Liczba powtórzeń	Zakres wartości
		w opisie systemu	w bazie danych						

Rys. 7.3. Formularz "POLA W REKORDZIE"

nych elementarnych (np. dane numeryczne mogą rozpoczynać się od granicy słowa). W systemie UNIVAC-1100 stosuje się słowo 36-bitowe, do którego można dochodzić w całości lub tylko do jego części takich jak 1/2, 1/3, 1/4 czy 1/6 słowa. W słowie tym może zapamiętać 6 znaków w kodzie FIELDATA lub 4 znaki w kodzie ASCII. Jeżeli dane jest przechowywana w postaci znakowej (typ A, X lub N), fizyczną wielkość tej danej łatwo policzyć. Wynik podajemy w postaci liczby całkowitej z odpowiednim ułamkiem. Trudniejszą sprawą jest obliczenie wielkości pola numerycznego w postaci binarnej, jednak mamy pewne określone zasady w różnych typach komputerów ułatwiające je. I znów przykład z systemu UNIVAC-1100. Podamy, jak zależy wielkość pola przeznaczonego do przechowywania liczby w postaci binarnej od ilości cyfr tej liczby (metoda USAGE COMP).

1 - 2 cyfry = 1/4 słowa = 9 bitów
3 - 5 cyfr = 1/2 słowa = 18 bitów
6 - 7 cyfr = 3/4 słowa = 27 bitów
8 - 10 cyfr = 1 słowo = 36 bitów
11 - 12 cyfr = 1 1/4 " = 45 bitów
14 - 15 cyfr = 1 1/2 " = 54 bitów
16 - 18 cyfr = 1 3/4 " = 63 bity

Podobną tabelkę można zbudować przy stosowaniu innej metody pamiętania danych numerycznych, np. USAGE COMP-4.

Reasumując, znając maksymalny zakres liczby przechowywanej w postaci binarnej w danym polu można stosunkowo łatwo obliczyć fizyczną wielkość tego pola. Do wielkości tej proponuje się dodanie tej części poprzedniego słowa, która na skutek konieczności rozpoczęcia zapisu od początku słowa nie będzie wykorzystana. Po obliczeniu wielkości pól wystarczy wpisać te wielkości do formularza oraz posumować je. W rezultacie otrzymamy wielkość rekordu w słowach, a właściwie wielkość pola danych rekordu. Zauważmy też, że gdy pole to nie zawiera całkowitej liczby słów, system z reguły i tak do całkowitej liczby słów uzupełni nam nasz rekord. Lepiej więc uzupełnić je na końcu polem pustym (np. FILLER), które może być wykorzystane w późniejszym czasie bez potrzeby reorganizacji. Wynik obliczenia nanosimy w odpowiednie miejsce formularza. Przy rekordach zmiennej długości wielkość rekordu zależy od rozmiaru tablicy. Projektując należy znać minimalną, maksymalną a przede wszystkim średnią liczbę powtórzeń. W rezultacie będziemy mogli obliczyć średnią wielkość takiego rekordu.

7.3.2. Obliczenie wielkości rekordu

Każdy rekord składa się z części użytkownika i części systemowej. Wielkość części użytkownika obliczyliśmy w punkcie wcześniejszym, teraz obliczymy część systemową. Zależy ona oczywiście od stopnia powiązań danego rekordu z innymi rekordami bazy. Powiązania grupowe są bowiem realizowane przez wskaźniki (ang. pointers), które tworzą właśnie część systemową re-

Załącznik 7.4

Format rekordu	Data	Schemat	Kod rekordu
	Nr kolejny	Sporządził	Nazwa rekordu

Liczba słów

Nagłówek rekordu _____

Wskaźniki właściciela

Właściciel - grup z łącznikami NEXT _____

Właściciel - " z łącznikami PRIOR _____

Właściciel - " z ORDER LAST z poj. łącznik. _____

Właściciel - " POINTER ARRAY _____

Wskaźnik Członka Automatycznego

Członek - grup z łącznikami NEXT _____

Członek - " z łącznikami PRIOR _____

Członek - " z łącznikami TO OWNER _____

Naturalny Znacznik /Flag/ kontrolny

1 słowo, jeśli rekord jest członkiem co najmniej jednej grupy manualnej _____

Wskaźnik członka Manualnego

ilość zarezerwowanych wskaźników _____

Wskaźniki łańcuchowe /calc lub Index Seguential/

1. słowo dla łączników pojedyńczych _____

2. słowo dla łączników podwójnych _____

Wskaźniki systemowe

1. słowo, jeśli rekord zmodyfikowano lub wpisano na stronie nadmiarowej _____

_____

Suma informacji systemowych _____

Dane użytkownika

Max. _____ słów

Min. _____ słów

Średnia wielkość rekordu _____

_____

Całkowita wielkość rekordu _____

Rys. 7.4. Formularz "FORMAT REKORDU"

kordu. Obliczenie liczby tych wskaźników oraz udokumentowanie tego faktu zapewnia nam kolejny formularz "FORMAT REKORDU" (zał. 7.4). Zawiera on informacje o związkach każdego rekordu z innymi rekordami bazy. W kolumnie po prawej stronie formularza wpisujemy odpowiednią liczbę słów przeznaczonych na wskaźniki, wynikającą z udziału rekordu w strukturze grupowej oraz rodzaju wskaźników używanych w danych grupach. W przypadku grup manualnych wystarczy zarezerwować 1 słowo kontrolne i pewną liczbę słów na wskaźniki w grupach manualnych, w których rekord może występować. Jeżeli rekord ten nie może być jednocześnie w kilku grupach manualnych, wystarczy mu zarezerwować taką liczbę wskaźników, jaka jest potrzebna dla jednej tylko grupy (o maksymalnej liczbie wskaźników). Gdy rekord może być równocześnie w kilku grupach tych wskaźników, trzeba zarezerwować więcej. Po podsumowaniu wszystkich wskaźników i nagłówka rekordu otrzymujemy całkowitą wielkość części systemowej rekordu. Dodając do tego wielkość pola danych użytkownika (obliczoną w punkcie 7.3.1) otrzymujemy całkowitą wielkość rekordu. W przypadku rekordów o zmiennej długości operować będziemy wielkością średnią.

7.3.3. Obliczanie średniej wielkości rekordu w obszarze

Gdy w obszarze zapamiętany jest tylko jeden typ rekordu, kroku tego można w ogóle nie wykonywać. W przypadku wielu typów rekordu w obszarze operować będziemy pojęciem średniej wielkości rekordu w obszarze. Będzie to oczywiście liczba różna dla różnych obszarów, zależeć będzie od wielkości rekordów oraz oczekiwanej ich liczby wystąpień (średnia ważona).

Obliczać ją będziemy wg wzoru:

$$W_{śr} = \frac{n_1 W_1 + n_2 W_2 + \ldots + n_k \cdot W_k}{N}$$

gdzie:

n_i - oczekiwana liczba rekordów typu "i",
W_i - wielkość "i" - tego rekordu,
N - sumaryczna liczba wszystkich rekordów oczekiwanych w obszarze.

Aby ułatwić i udokumentować przeprowadzone obliczenia, proponuje się użycie formularza "REKORDY W OBSZARZE" (zał. 7.5).

W kolejnych wierszach tego formularza wpisujemy zebrane wcześniej informacje o wszystkich typach rekordów w danym obszarze. Będą to kolejno informacje: kod rekordu, jego nazwa, oczekiwana liczba rekordów tego typu, średnia wielkość rekordu. Ponadto zaznaczyć możemy typ alokacji rekordu, jeżeli jest direct lub calc oraz podać szczególne warunki przechowywania rekordu (o ile takie istnieją). Należą do nich stosowanie interwałowej metody pamiętania rekordów (przedzielanie ich określoną liczbą pustych stron) czy pamiętanie rekordu nie w całym obszarze lecz określonej jego części (podanej poprzez zakres stron). Obliczenia dla wyróżnionych

Kod rekordu	Nazwa rekordu	Oczekiwana liczba rekordów	Wielkość rekordu (w słowach)	Całkowita liczba słów na dane	Direct	Calc	Interwał	Zakres stron		Przyjęta		Rekordy w obszarze			
								od	do	liczba rekordów	liczba słów na dane		Data	Schemat	Kod obszaru
													Nr kolejny	Sporządził	Nazwa obszaru
Razem															

Rys. 7.5. Formularz "REKORDY W OBSZARZE"

części obszarów powinny być wykonywane oddzielnie. W rubryce "CAŁKOWITA LICZBA SŁÓW NA DANE" wpisujemy iloczyn liczby rekordów i ich wielkości w słowach. Po wpisaniu w ten sposób informacji o wszystkich rekordach danego obszaru wystarczy podsumować kolumny "LICZBA REKORDÓW" i "CAŁKOWITA LICZBA SŁÓW NA DANE". Po podzieleniu tych sum uzyskamy średnią wielkość rekordu w obszarze. Kolumna "PRZYJĘTA LICZBA REKORDÓW" może służyć jako jedna z możliwości przyjmowania zapasu obszaru umożliwiająca rozrost bazy.

7.3.4. Projektowanie formatu strony

Po obliczeniu średniej wielkości rekordu w obszarze można przystąpić do wyboru wielkości strony. Należy tu mieć na uwadze wszystkie te zasady, o których była mowa na początku rozdziału 7.3. W tym miejscu możemy jedynie dla alternatywnych rozwiązań porównać efektywność wykorzystania stron. Idealne rozwiązanie byłoby wtedy, gdyby wszystkie słowa strony były wykorzystane (na dane systemowe lub użytkownika). Jest to jednak praktycznie nie do uzyskania, gdy w obszarze pamiętamy rekordy o różnych długościach. Operując pojęciem średniej wielkości rekordu może uda nam się tak dobrać wielkość strony, że nie stracimy w ogóle lub niewiele słów będzie nie wykorzystanych. Jednakże w praktyce na konkretnych fizycznych stronach obszaru różny będzie stopień efektywności ich wykorzystania. Tym niemniej, statystycznie rzecz biorąc, w miarę wzrostu liczby stron obszaru coraz większa ich ilość byłaby dobrze wykorzystana. Mając to na uwadze przystępujemy do projektowania struktury strony. Do udokumentowania tego etapu używać będziemy formularza "FORMAT-STRONY" (patrz zał. 7.6), aczkolwiek do początkowych obliczeń wykorzystamy również formularz "STRUKTURA OBSZARU", choć ten właściwie będzie nam dokumentował następny etap (patrz zał. 7.7).

W formularzu "STRUKTURA OBSZARU" wypełniamy następujące pozycje: oczekiwana liczba rekordów, oczekiwana liczba rekordów typu calc i średnia wielkość rekordu obliczona wcześniej. Porównanie dwóch pierwszych liczb pozwoli na ocenę stopnia udziału rekordów calc w tym obszarze. Pozwoli to nam na właściwy wybór liczby łańcuchów obliczeniowych na stronie.

Z kolei średnia wielkość rekordu pozwoli na obliczenie średniej liczby rekordów na stronie. Oczywiście otrzymaną liczbę musimy zaokrąglić w dół do liczby całkowitej, gdyż rekord musi być całkowicie zapamiętany na stronie, nie można go podzielić. Mając średnią liczbę rekordów na stronie oraz procentowy udział rekordów calc w tym obszarze można obliczyć, ile średnio rekordów calc znajduje się na każdej stronie. To pozwoli nam na przyjęcie właściwej liczby łańcuchów obliczeniowych. Jak wiemy, dostęp do rekordów calc będzie najszybszy, gdy w łańcuchu obliczeniowym będzie tylko jeden rekord. Gdy więc liczba rekordów calc na stronie jest niewielka (np. <10) można zdefiniować tyle łańcuchów ile rekordów. Gdy liczba rekordów jest większa, musimy zgodzić się na to, że w łańcuchu będzie więcej niż 1 rekord. W przeciwnym przypadku zbyt dużo słów stracilibyśmy na dane systemowe, tzn. nagłówki łańcuchów obliczeniowych.

Załącznik 7.6

Format strony	Data	Schemat	Kod obszaru
	Nr kolejny	Sporządził	Nazwa obszaru

Liczba słów

Nagłówek strony _____

Obszar dla rekordów

_____ x _____ _____

liczba rekordów x średnia wielkość rekordu

Słowo Indeksowe Rekordu

/1 słowo dla każdego rekordu/ _____

Nagłówki Łańcuchów Obliczeniowych

/1 słowo dla każdego łańcucha obliczeniowego minus 1/ _____

_____

Całkowita liczba danych na stronie _____

Przyjęta wielkość strony _____

Rys. 7.6. Formularz "FORMAT STRONY"

Załącznik 7.7

Struktura obszaru	Data	Schemat	Kod obszaru
	Nr kolejny	Sporządził	Nazwa obszaru

Procedury obliczeniowe dla elementów strukturalnych obszaru:

Oczekiwana liczba rekordów _____

W tym liczba rekordów calc _____

Stosunek liczby rekordów calc do ogólnej liczby rekordów /%/ _____

Średnia wielkość rekordu _____

Przyjęta wielkość strony _____

Liczba rekordów na stronie:

_____ - 10 = _____ : _____ : / _____ + 1/ = _____
wielkość strony — wielkość rekordu

W tym rekordów calc:

_____ x _____ = _____
liczba rekordów na stronie — % rekordów calc

liczba stron w obszarze

_____ : _____ = _____
liczba rekordów — liczba rekordów na stronie — liczba stron

Przyjęty zapas w %

Wielkość obszaru z zapasem

Wartości przyjęte:

Liczba stron

Wielkość strony

Liczba łańcuchów calc na stronie

Fizyczna wielkość obszaru:

_____ x _____ = _____
liczba stron — wielkość strony

Rys. 7.7. Formularz "STRUKTURA OBSZARU"

Mając przyjętą liczbę łańcuchów obliczeniowych można przystąpić do udokumentowania i sprawdzenia struktury strony. Na formularzu "FORMAT STRONY" (zał. 7.6) wpisujemy średnią wielkość rekordu, średnią liczbę rekordów na stronie oraz liczbę łańcuchów obliczeniowych. To pozwoli nam na obliczenie całkowitej liczby danych na stronie. Przy okazji może okazać się, że zaprojektowana strona nie pomieści tylu informacji, ile chcielibyśmy. Musimy wtedy powtórzyć obliczenie zmniejszając projektowaną liczbę rekordów na stronie lub zmienić liczbę łańcuchów obliczeniowych na tej stronie. Innym rozwiązaniem jest przyjęcie nowej wielkości strony i powtórzenie wszystkich obliczeń w tym kroku (łącznie z udokumentowaniem tego na formularzach "FORMAT STRONY" i "STRUKTURA OBSZARU").

7.3.5. Obliczanie liczby stron obszaru

Uzyskane w poprzednim kroku informacje pozwolą nam na łatwe obliczenie liczby stron obszaru. W tym celu posłużmy się częściowo już wypełnionym w poprzednim kroku formularzem "STRUKTURA OBSZARU" (zał. 7.7).

Mając obliczoną liczbę rekordów na stronie oraz planowaną liczbę rekordów łatwo policzyć niezbędną liczbę stron na te rekordy. Oczywiście przy przewidywaniu liczby rekordów projektant powinien wziąć pewną poprawkę w górę umożliwiającą ekspansję systemu. Tym niemniej projektant bazy danych niezależnie od tego powinien przyjąć pewien zapas ułatwiający pracę procedurom alokacji. Szczególnie istotne jest to w przypadku dostępu calc zwłaszcza wtedy, gdy nie zdefiniowano stron nadmiarowych w tym obszarze. Wtedy sprawność procedur obliczeniowych zależy od stopnia załadowania obszaru.

Przy dużym stopniu załadowania dostęp znacznie pogarsza się, gdyż coraz więcej rekordów znajdzie się nie na swoich głównych stronach. Coraz trudniej zapamiętać w takiej strukturze nowy rekord. Sytuację taką opisuje tabela (zał. 7.8) pokazująca (zgodnie z założonym rozkładem Poissona), ile % rekordów będzie musiało być umieszczonych na stronach obcych w zależności od stopnia załadowania obszaru oraz od liczby rekordów na stronie. Widać z niej, że przy całkowitym załadowaniu obszaru znaczna część rekordów jest umieszczana na stronach obcych, np.: gdy na stronie mieści się 10 rekordów, to 12,51% wszystkich rekordów nie będzie pod swoim głównym adresem. Zmniejszenie stopnia załadowania obszaru polepszy nam tę sytuację; przy wskaźniku załadowania 70% tylko niespełna 3% rekordów będzie na stronach obcych. To wszystko upoważnia projektanta bazy do przyjęcia zgodnie z wymaganiem odpowiedniego zapasu stron. Przyjęcie zapasu rzędu 15%-30% praktycznie zapewni nam efektywność ponad 90%.

Przyjęty % zapasu wpisujemy do formularza "STRUKTURA OBSZARU" i obliczamy wielkość obszaru z zapasem. Czasem system operacyjny ma dodatkowe ograniczenie co do wielkości zbiorów, np. aby był on całkowitą wielokrotnością ścieżki czy pozycji. Wtedy też otrzymaną liczbę zaokrąglamy w gó-

Tabela 7.8

ROZKŁAD POISSONA

Procent rekordów umieszczanych na stronach obcych w metodzie alokacji randomizacyjnej

Rekordów na str.	Wskaźnik załadowania 0.5	0.6	0.7	0.8	0.9	1.0	1.1	1.2
1	21.31	24.80	28.08	31.17	34.06	36.79	39.25	41.77
2	10.36	13.65	17.03	20.43	23.79	27.07	30.24	33.30
3	5.99	8.82	11.99	15.37	18.87	22.40	25.91	29.33
4	3.76	6.15	9.05	12.32	15.86	19.54	23.25	26.93
5	2.48	4.49	7.11	10.26	13.78	17.55	21.42	25.30
6	1.69	3.38	5.75	8.75	12.24	16.06	20.06	24.11
7	1.18	2.60	4.74	7.60	11.04	14.90	19.00	23.19
8	0.84	2.03	3.97	6.68	10.07	13.96	18.15	22.46
9	0.61	1.61	3.36	5.94	9.27	13.18	17.44	21.86
10	0.44	1.29	2.88	5.32	8.59	12.51	16.85	21.36
11	0.33	1.04	2.48	4.80	8.01	11.94	16.34	20.94
12	0.24	0.85	2.15	4.36	7.51	11.44	15.89	20.58
14	0.14	0.57	1.65	3.64	6.67	10.60	15.15	19.99
16	0.08	0.39	1.28	3.09	6.00	9.92	14.56	19.53
18	0.05	0.28	1.01	2.65	5.45	9.36	14.97	19.16
20	0.03	0.20	0.81	2.30	4.99	8.88	13.66	18.86
25	0.01	0.09	0.48	1.65	4.10	7.95	12.87	18.31
30		0.04	0.29	1.23	3.47	7.26	12.31	17.93
35		0.02	0.18	0.94	2.98	6.73	11.87	17.66
40		0.01	0.12	0.73	2.60	6.29	11.53	17.47
50			0.05	0.45	2.04	5.63	11.03	17.20
60			0.02	0.30	1.65	5.14	10.68	17.03
70			0.01	0.20	1.37	4.76	10.41	16.93
80			0.01	0.13	1.14	4.46	10.21	16.86
90				0.09	0.97	4.20	10.05	16.80
100				0.06	0.83	3.99	9.92	16.77
	Procent wykorzystania obszaru							
	50	60	70	80	90	100	110	120

rę, tak aby nie tracić i tak przydzielonego przez system miejsca w pamięci masowej.

7.3.6. Obliczanie wielkości bazy danych

Krok ten jest zamknięciem etapu projektowania struktury fizycznej bazy. Jest bardzo istotny, gdyż pozwoli na porównanie projektowanych rozwiązań z fizycznymi zasobami, jakimi dysponuje system komputerowy.

Niekiedy zdarza się, zwłaszcza początkującym projektantom, że zaprojektowany system przekracza możliwości komputera - projektowana baza danych nie mieści się w posiadanej pamięci masowej. Jedynym rozwiązaniem, poza oczywiście rezygnacją z systemu lub zakupem dodatkowej pamięci, jest całkowite przeprojektowanie systemu mające na celu zmniejszenie liczby typów rekordów i ich wystąpień. Należy wtedy uprościć system (choć logicz-

Załącznik 7.9

STRUKTURA BAZY DANYCH			Data	Schemat	
			Nr kol.	Sporządził	
Kod obszaru	Nazwa obszaru	Wielkość strony	Liczba stron	Wymagana pamięć	
				w słowach	w ścieżkach
RAZEM					

Rys. 7.9. Formularz "STRUKTURA BAZY DANYCH"

nie można go w ten sposób skomplikować), zmniejszyć liczbę funkcji, używać zagregowanych wielkości zamiast danych elementarnych.

Dane dotyczące każdego obszaru bazy nanosimy na formularz "STRUKTURA BAZY DANYCH" (zał. 7.9).

Kolejno wpisujemy tu kod obszaru, nazwę obszaru, przyjętą liczbę stron i wielkość strony. To pozwoli nam na obliczenie wymaganej przez obszar pamięci najpierw w słowach, a potem ewentualnie w ścieżkach czy pozycjach (co zależy zresztą w znacznej mierze od struktury fizycznej systemu komputerowego). Dodając wielkości pamięci wymaganych przez poszczególne obszary możemy obliczyć wielkość całej bazy danych. Wielkości obszarów obliczane w tym kroku pozwalają projektantowi na przydzielenie fizycznych zbiorów pamięci masowej przeznaczonych na te obszary. W różnych systemach komputerowych może to być różnie realizowane. W niektórych z nich każdy zbiór zawiera tylko jeden obszar, w innych jeden zbiór może zawierać kilka obszarów bazy. Istotny w tym ostatnim przypadku jest w miarę równomierny podział obszarów na zbiory. Można tutaj również brać pod uwagę rozdział zbiorów na różne urządzenia fizyczne.

Umieszczenie dwóch nie związanych ze sobą obszarów na różnych dyskach umożliwi faktycznie równoległą pracę dwóm wykorzystującym je programom. Z kolei zbiór zawierający obszary częściej wykorzystywane można przydzielić do szybszego urządzenia (o ile takie istnieją w konfiguracji) lub nawet zapewnić centryczne położenie tego zbioru na dysku, aby zminimalizować ruch głowic przy odczycie tego zbioru.

7.4. Strony nadmiarowe

Jak już mówiliśmy poprzednio, niektóre typy alokacji wymagają definiowania stron nadmiarowych (metoda indeksowo-sekwencyjna), dla innych jest to opcjonalne (metoda calc lub poprzez grupę logiczną).

Zastanówmy się nad wadami i zaletami różnych rozwiązań w tym zakresie. Z uwagi na możliwość definiowania stron nadmiarowych globalnych lub lokalnych (patrz rys. 2.2) mamy właściwie trzy dopuszczalne rozwiązania. Porównajmy je kolejno.

Pierwszym rozwiązaniem będzie oczywiście nieużywanie stron nadmiarowych w ogóle. Wtedy procedury alokacji rekordów muszą być do takiej sytuacji odpowiednio przygotowane i w razie braku miejsca na stronie głównej muszą umieszczać rekord na najbliższej stronie z danymi. Oczywiście w tym przypadku musimy zdefiniować wielkość obszaru z pewnym zapasem, aby obniżyć stopień załadowania poszczególnych stron obszaru (patrz tab. 7.8). Nawet wtedy nie unikniemy jednak rekordów, które nie zmieściły się na swoich stronach głównych. Co więcej, jeżeli te rekordy umieszczone zostaną nie na stronach nadmiarowych, lecz na innych stronach z danymi, pogorszą się znacznie warunki obsługi rekordów z tych właśnie stron. Przy próbie zapisu nowego rekordu na takie strony może okazać się, że nie ma na nich

już miejsca. I w ten sposób rekord, który normalnie trafiłby na swoją stronę główną, zostanie z niej przeniesiony na inną. Przy większym stopniu załadowania bazy proces ten przebiega lawinowo i rekordy są w znacznej mierze nieuporządkowane. Sytuację może poprawić jedynie reorganizacja takiego obszaru. Tak więc wydaje się, że rozwiązanie takie jest zdecydowanie złe, co oczywiście odbije się na czasie pracy programów na tym obszarze.

Porównajmy obie metody definiowania stron nadmiarowych. Weźmy najpierw prostszą możliwość - strony nadmiarowe globalne zdefiniowane na końcu obszaru. Wielkość tej części obszaru wyznaczamy na podstawie przewidywanej liczby rekordów, które muszą trafić na strony obce przy założonym stopniu załadowania. Dane w tab. 7.8 nieco zaokrąglamy w górę, uzyskując przez to większą elastyczność przy alokacji naszych rekordów. Wielkość obszaru na strony nadmiarowe zależy również od innych czynników. Załóżmy, że w grupie logicznej właściciel ma zdefiniowany dostęp poprzez procedurę obliczeniową. Załóżmy też, że konkretny rekord właściciela został umieszczony na stronie nadmiarowej. Pytanie - jak umieszczać teraz rekordy członkowskie? Zgodnie ze strategią najbliższego miejsca rekordy takie powinny być umieszczane jak najbliżej właściciela, a więc w tym przypadku również na stronach nadmiarowych. Innym rozwiązaniem jest umieszczanie tych rekordów członkowskich jak najbliżej strony głównej, na której powinien być zapamiętany właściciel. Systemy zarządzania bazą danych przeważnie dopuszczają obydwa rozwiązania, a właściwe należy zdefiniować przy generowaniu systemu.

Podobnie może przebiegać aktualizacja grupy o nowe rekordy. Załóżmy, że w grupie takiej rekord członkowski musiał być umieszczony na stronie nadmiarowej. Rekord następny może być umieszczony również na stronie nadmiarowej lub na najbliższej stronie z danymi, na której przebywa przedostatni rekord grupy. Tak więc logiczny punkt wstawienia wskazuje nam bądź stronę nadmiarową, bądź stronę z danymi. Jeżeli zdecydujemy się, że w obszarze nadmiarowym mogą przebywać rekordy członkowskie w grupach, aby zadośćuczynić strategii najbliższego miejsca - obszar ten powinien być odpowiednio duży. Obsługa obszaru, który ma nadmiar globalny, może być uciążliwa. Szczególnie trudne jest rozszerzenie obszaru, gdy jego stopień zapełnienia zbliży się do 100%. Praktycznie nie może w tym przypadku obejść się bez reorganizacji obszaru, gdyż jedyne co można zrobić to zwiększenie liczby stron nadmiarowych w obszarze, ale absolutnie nie przyczyni się to do uporządkowania rekordów. Jeżeli mamy zdefiniowany w obszarze nadmiar lokalny, łatwiej wykonać rozszerzenie obszaru o nowe strony na dane i nadmiarowe. Tym niemniej używanie stron nadmiarowych globalnych może dawać bardzo dobre efekty, szczególnie zaś gdy w obszarze zapamiętywane są rekordy właściciela grupy poprzez procedurę obliczeniową i rekordy członkowskie poprzez tę grupę. Jeżeli obszar nadmiarowy jest odpowiednio duży, przetwarzanie sekwencyjne tych grup może być bardzo sprawne. Kolejną za-

letą tej metody jest prostota, co jest szczególnie istotne w procedurach obliczeniowych. Jak wiemy, muszą one zawierać jako swoją integralną część algorytm rozwiązywania konfliktów, który w tym przypadku jest prosty i szybki. Praktyka wykazuje, że metodę tę opłaca się stosować, jeżeli strony w obszarze są duże i mają więcej niż 20 rekordów. Stosując tę metodę należy również pamiętać o możliwości stosowania wskaźnika początkowego załadowania stron w obszarze. Jeżeli przyjmiemy, że będzie on niski, to na stronach z danymi zostaje miejsce na nowe rekordy (głównie na rekordy pamiętane przez grupę), zaś rekordy-właściciele grup mogą już podczas wstępnego ładowania znaleźć się na stronach nadmiarowych, gdzie jest dostatecznie dużo wolnego miejsca na ich rekordy członkowskie. Tak więc rozkład rekordów w obszarze będzie w miarę prawidłowy, a ich przetwarzanie szybkie.

Strony nadmiarowe lokalne szczególnie nadają się dla obszarów z małymi stronami (lub takich, w których są duże rekordy) zawierającymi co najwyżej 10 rekordów na stronie. Są one traktowane jako naturalne "przedłużenie" strony, której z innych powodów nie można zdefiniować większej. W szczególności strony nadmiarowe mogą być poprzeplatane ze stronami z danymi w stosunku 1:1; po każdej stronie z danymi będzie wtedy jej strona nadmiarowa. W innych przypadkach można definiować strony nadmiarowe po kilku/kilkunastu stronach na dane. Zależy to oczywiście od typu rekordów w obszarze, ich wielkości i sposobu przetwarzania. Metoda taka głównie nadaje się do sekwencyjnego przetwarzania grup o małej liczbie rekordów członkowskich. Rekordy członkowskie, które nie mieszczą się na stronach głównych, zostaną zapamiętane na najbliższych fizycznie stronach. Czas przetwarzania takiej grupy będzie wtedy minimalny.

Tak więc stosowanie nadmiaru lokalnego jest bardziej naturalne, łatwiej przewidzieć stan i strukturę załadowania poszczególnych części obszaru. Z kolei utrudnieniem jest większa komplikacja algorytmu obliczeniowego. Z tego też powodu nie wszystkie standardowe procedury calc dopuszczają możliwość stosowania stron nadmiarowych lokalnych. I tak np. w systemie DMS-1100 na komputery serii UNIVAC-1100 stosowane są 2 standardowe procedury obliczeniowe. Pierwsza z nich - DMSCALC bazująca na algorytmie kolejnych dzieleń pozwala na użycie tylko nadmiaru globalnego. Jest jednak znacznie szybsza od drugiej standardowej procedury - RANDENTIAL. Procedura ta bazuje na metodzie analizy znaków i pozwala na jej stosowanie w obszarach z nadmiarem lokalnym.

Reasumując, stosowanie stron nadmiarowych jest w większości przypadków konieczne, zwłaszcza gdy w obszarze są rekordy calc lub indeksowo-sekwencyjne. Wybór rodzaju stron nadmiarowych zależy od posiadanych procedur obliczeniowych oraz od rodzaju i liczby rekordów w obszarze i sposobu ich przetwarzania.

8. SCHEMAT BAZY DANYCH

Opracowanie projektu bazy zarówno struktury logicznej, jak i fizycznej jest pierwszym krokiem w implementacji nowej bazy. Kolejnym krokiem jest utworzenie schematu bazy danych. Tak nazywany jest zbiór definicji wszystkich obiektów bazy używany zarówno do kompilacji programów korzystających z bazy danych, jak i do ich realizacji. Narzędziem projektanta jest kompilator JĘZYKA OPISU DANYCH (ang. DDL - Data Definition Language). Rozróżniamy dwie wersje schematu bazy - wersję źródłową napisaną w języku DDL oraz wersję wynikową w postaci wygenerowanych zakodowanych tablic umieszczonych w specjalnym zbiorze pamięci masowej. W rozdziale tym pokażemy składnię języka DDL, wewnętrzną budowę schematu oraz przedstawimy miejsce schematu w systemie.

8.1. Język Opisu Danych - DDL

Jest to procesor językowy zbliżony zresztą do procesora COBOL. Wszystkie obiekty bazy danych, takie jak obszary, rekordy, grupy oraz ich atrybuty, są opisane w tym właśnie języku, tworząc tzw. schemat źródłowy. Składnia tego języka może być różna w różnych systemach komputerowych, tym niemniej grupa CODASYL opracowała pewne generalne zasady i kształt procesora. Strukturę syntaktyczną tego procesora pokażemy na przykładzie systemu DMS-1100 (patrz zał. 8.1).

Nie wydaje się jednak celowe dokładne omawianie poszczególnych formatów instrukcji DDL. Przecież Czytelnik z reguły będzie miał do czynienia z innym systemem komputerowym, w którym również Język Opisu Danych będzie posiadał swój własny charakter. Dlatego też omówimy jedynie pewne grupy instrukcji, podając raczej typ wprowadzanych informacji niż dokładną składnię języka z wszystkimi jej uwarunkowaniami i ograniczeniami.

Program definiujący strukturę bazy danych napisany w języku DDL jest zbliżony kształtem do programu w języku COBOL. Podobnie jak on składa się z działów (ang. DIVISION), z tym że tutaj występują jedynie dwa ich rodzaje. Są to: dział identyfikacji schematu (IDENTIFICATION DIVISION) i dział opisu danych (DATA DIVISION). W dziale identyfikacji zawarta jest właściwie tylko jedna informacja. Jest nią nazwa schematu, która zresztą utożsamiana jest z nazwą bazy danych. Wybór właściwej bazy danych przez program w systemach o wielu bazach danych odbywa się właśnie przez tę nazwę (klauzula INVOKE). Dział danych składa się z sekcji zawierających opisy kolejno obszarów (AREA SECTION), rekordów (RECORD SECTION) i grup

APPENDIX B. DDL SYNTAX SKELETON

B.1. GENERAL

Following is a complete DDL Syntax Skeleton.

B.2. SYNTAX SKELETON

IDENTIFICATION DIVISION.

SCHEMA NAME IS *schema-name*.

[; TIP FILE CODE IS *integer-0*].

DATA DIVISION.

AREA SECTION.

[AREA CONTROL IS *integer-1* AREAS.]

[AREA LOOKS INCLUDE { QUICK-BEFORE-LOOKS | BEFORE-LOOKS | AFTER-LOOKS | NO-LOOKS }]

[RECOVERY-POINTS ARE EVERY *integer-2* BLOCKS.]

AREA NAME IS *area name-1*;

AREA CODE IS *integer-1*;

[AREA MAPS TO TIP FILE;]

ALLOCATE *integer-2* [PRE-INITIALIZED] PAGES

[, *integer-10* OVERFLOW PAGES AT END]

[, *integer-3* OVERFLOW PAGES EVERY *integer-4* DATA PAGES]

[, EXPANDABLE TO *integer-11* PAGES];

PAGES ARE *integer-5* WORDS

```
[ ; LOOKS INCLUDE { QUICK-BEFORE-LOOKS | BEFORE-LOOKS | AFTER-LOOKS | NO-LOOKS } ]

[ ; LOAD IS { integer-6 WORDS | integer-7 PERCENT } ]

[ ; CALC USES integer-8 CHAINS [LINKED PRIOR] ] .

RECORD SECTION .

RECORD NAME IS record-name-1 ;

    RECORD CODE IS integer-1 ;

                    { DIRECT data-base-data-name-1 , data-base-data-name-2
                    { CALC data-base-procedure-name-1
                    {      IN data-base-data-name-3
                    {      USING data-base-identifier-1
                    {          [ , data-base-identifier-2 ] ...
                    {      DUPLICATES ARE [NOT] ALLOWED
                    { VIA set-name-1 SET
                    {     [INTERVAL IS integer-2 PAGES]
    LOCATION MODE IS

    [ { ; WITHIN area-name-1 [ { integer-3 | data-base-data-name-4 } { THRU | THROUGH } { integer-4 | data-base-data-name-5 } ] } ... ]

    [ ; RESERVE integer-5 POINTERS]

    [ ; RECORD MODE IS { FIELDATA | ASCII } ]

[ { ; { PIC | PICTURE } IS character-string-1 } ]
[ { ; [ { "Item description" } ... ] } ]
```

- **Item Description Syntax Skeleton**

```
Level-number-1 { data-base-identifier-3 }
               { FILLER                 }

[; { PIC     } IS character string-2]
   { PICTURE }

[                 LOCK                 ]
[                 DISPLAY              ]
[                 DISP                 ]
[                 DISPLAY-1            ]
[                 DISP-1               ]
[; USAGE IS       COMPUTATIONAL        ]
[                 COMP                 ]
[                 COMPUTATIONAL-4      ]
[                 COMP-4               ]
[                 AREA-KEY             ]
[                 AREA-NAME            ]
[                 DATABASE-KEY         ]

[; OCCURS { integer-6 TIMES                                  }]
[         { integer-7 TO integer-8 TIMES                     }]
[         {           DEPENDING ON data-base-identifier-4    }]

SET SECTION.

SET NAME IS set-name-1;

    SET CODE IS integer-1;

    MODE IS CHAIN [LINKED PRIOR];

                 { FIRST                          }
                 { LAST                           }
    ORDER IS     { NEXT                           };
                 { PRIOR                          }
                 { SORTED [WITHIN RECORD-NAME]    }

    OWNER IS record-name-1;

    { "Member sub-entry" } ...
```

- **Syntax Format of Member Sub-entry**

```
MEMBER IS record-name-2 { AUTOMATIC } [LINKED TO OWNER]
                        { MANUAL    }
[ { ASCENDING  } [RANGE] KEY IS || RECORD-NAME                  || } ...
  { DESCENDING }                || data-base-identifier-1       ||
                                || [,data-base-identifier-2] ...  ||

                  DUPLICATES ARE { FIRST  }
                                 { LAST   }
                                 { NOT ALLOWED } ] ;
```

{*"Set occurrence selection sub-entry"*}

- **Syntax Format of Set Occurrence Selection Sub-entry**

Format-1:

```
;SET OCCURRENCE SELECTION IS THRU
{ CURRENT OF SET          }
{ LOCATION MODE OF OWNER  }
  [ { USING data-base-identifier-3[,data-base-identifier-4]...                    }
    { ALIAS { FOR data-base-identifier-5 } IS data-base-data-name-2 }...          } ]
            { OF data-base-data-name-1   }
```

Format-2:

```
;SET OCCURRENCE SELECTION IS THRU set-name-2 USING
{ CURRENT OF SET          }
{ LOCATION MODE OF OWNER  }
  [ { USING data-base-identifier-6[,data-base-identifier-7]...                    }
    { ALIAS { FOR data-base-identifier-8 } IS data-base-data-name-2}...           } ]
            { OF data-base-data-name-1   }
{ set-name-3                                                                      }
{   { USING data-base-identifier-9 [,data-base-identifier-10]...                  }
{   { ALIAS FOR data-base-identifier-11 IS data-base-data-name-3}...              } ...
```

Rys. 8.1. Składnia DDL (system DMS-1100)

Załączni. 8.2

```
DMS*SZOPWT(1).SCHEMA
                IDENTIFICATION DIVISION.
                SCHEMA SZOPWT
                    TIP 19
                DATA DIVISION
                AREA SECTION
                AREA LOOKS QUICK-BEFORE-LOOKS AFTER-LOOKS
                AREA DA-OLZAM
                    CODE 4
                    MAPS TIP
                    ALLOCATE 556.
                    EXPANDABLE 4000
                    PAGES 448
                    CALC 4
                AREA DA-ZNZAM
                    CODE 11
                    MAPS TIP
                    ALLOCATE 936.
                    EXPANDABLE 4000
                    PAGES 448
                    CALC 3
                AREA DA-ZNPROD
                    CODE 12
                    MAPS TIP
                    ALLOCATE 156.
                    EXPANDABLE 4000
                    PAGES 448
                    CALC 8
                AREA DA-ZNWYS
                    CODE 13
                    MAPS TIP
                    ALLOCATE 248.
                    EXPANDABLE 4000
                    PAGES 448
                    CALC 8
                AREA DA-WTZAM
                    CODE 16
                    MAPS TIP
                    ALLOCATE 328.
                    EXPANDABLE 4000
                    PAGES 448
                    CALC 6
                AREA DA-WTPROD
                    CODE 17
                    MAPS TIP
                    ALLOCATE 320.
                    EXPANDABLE 4000
                    PAGES 448
                    CALC 4
                AREA DA-WTWYS
                    CODE 18
                    MAPS TIP
                    ALLOCATE 100.
                    EXPANDABLE 4000
                    PAGES 448
                    CALC 1
                AREA DA-OLPROD
                    CODE 23
                    MAPS TIP
                    ALLOCATE 120.
                    EXPANDABLE 4000
                    PAGES 448
                    CALC 6
                AREA DA-OLWYS
                    CODE 33
                    MAPS TIP
                    ALLOCATE 56.
                    EXPANDABLE 4000
                    PAGES 448.
                    CALC 3
              ..............
                RECORD SECTION
                RECORD DR-KLIENT
                    CODE 1
                    LOCATION CALC DMSCALC
                     IN DAN-WTZAM
                     USING DOI-KOD-STAT-KL
                     DUPLICATES NOT
                    WITHIN DA-WTZAM
                    WITHIN DA-OLZAM
                    WITHIN DA-ZNZAM
                    MODE ASCII
                    05          DOI-KOD-STAT-KL        PIC  X(8)
                     10         DOI-NR-STATYS          PIC  9(7)
                     10         DOI-FILIA              PIC  X
                    05          DOI-NAZ-KLI            PIC  X(20)
                    05          DOI-MIASTO             PIC  X(30)
                    05          DOI-P-NAZ-KLI          PIC  X(74)
```

cd. zał. 8.2

```
88          05        DO1-ADRES-KLI        PIC   X(74)
89          05        DO1-NAZW-BANK        PIC   X(25)
90          05        DO1-NR-KONTA         PIC   X(18)
91          05        DO1-SYMB-ORG         PIC   9(5)
92          05        DO1-FILLER           PIC   X
93      RECORD DR-NAGŁOWEK
94          CODE 2
95          LOCATION CALC DMSCALC
96           IN DAN-WTZAM
97           USING DO2-NR-ZAM
98           DUPLICATES NOT
99          WITHIN DA-WTZAM
100         WITHIN DA-OLZAM
101         WITHIN DA-ZNZAM
102         MODE ASCII
103         05        DO2-NR-ZAM           PIC   X(6)
104         05        DO2-TYP-KLI          PIC   X
105         05        DO2-KWARTAL          PIC   9
106         05        DO2-NR-ZAM-KLI       PIC   X(20)
107         05        DO2-DATA-ZAM-K       PIC   9(6)
108         05        DO2-KOD-WYS          PIC   X(3)
109         05        DO2-ATEST            PIC   X
110         05        DO2-ODB-TECH         PIC   X
111         05        DO2-IL-POZ           PIC   99
112         05        DO2-WSK-ZAK          PIC   X
113         05        DO2-WSK-DRUK-POTW    PIC   X
114         05        DO2-FILLER           PIC   X(9)
115     RECORD DR-UWZAM
116         CODE 3
117         LOCATION VIA NAG-UWZ
118         WITHIN DA-WTZAM
119         WITHIN DA-OLZAM
120         WITHIN DA-ZNZAM
121         MODE ASCII
122         05        DO3-KLUCZ-UW
123          10       DO3-ZAK-UW           PIC   99
124          10       DO3-KOD-UW           PIC   99
125         05        DO3-SYMB-UW          PIC   X(4)
126         05        DO3-TRESC-UW         PIC   X(65) OCCURS 2
127         05        DO3-FILLER           PIC   X
128     RECORD DR-POZAM
129         CODE 4
130         LOCATION CALC DMSCALC
131          IN DAN-WTZAM
132          USING DO4-NR-ZAM
133          DUPLICATES NOT
134         WITHIN DA-WTZAM
135         WITHIN DA-OLZAM
136         WITHIN DA-ZNZAM
137         MODE ASCII
138         05        DO4-NR-ZAM           PIC   X(8)
139      *   10       DO4-ROK-ZAM          PIC   99
140      *   10       DO4-NR-NAG           PIC   9(4)
141      *   10       DO4-NR-POZ           PIC   99
142         05        DO4-SYMB-PROD
143          10       DO4-GAT              PIC   X(6)
144          10       DO4-SREDN.           PIC   9(4)
145          10       DO4-RSREDN           PIC   9
146          10       DO4-GRUB             PIC   99
147          10       DO4-RGRUB            PIC   99
148         05        DO4-DLUG             PIC   9(5)
149         05        DO4-WSK-GAT          PIC   X
150         05        DO4-TYP-KLI          PIC   X
151         05        DO4-ST-ZAM-W         PIC   X
152         05        DO4-ZAM-SPEC         PIC   X
153         05        DO4-CIEZ-ZAM         PIC   9(6) USAGE COMP
154          05       DO4-WYK
155           10      DO4-CIEZ-PMG         PIC   9(6) USAGE COMP
156           10      DO4-CIEZ-NAG         PIC   9(6) USAGE COMP
157           10      DO4-CIEZ-WYS         PIC   9(6) USAGE COMP
158         05        DO4-NAZ-KLI          PIC   X(20)
159         05        DO4-FILLER           PIC   X(6)
160     RECORD DR-PRODUKT
161         CODE 5
162         LOCATION CALC DMSCALC
163         IN DAN-WTPROD
164         USING DO5-SYMB-PROD
165         DUPLICATES NOT
166         WITHIN DA-WTPROD
167         WITHIN DA-OLPROD
168         WITHIN DA-ZNPROD
169         MODE ASCII
170         05        DO5-SYMB-PROD        PIC   X(15)
171      *   10       DO5-GAT              PIC   X(6)
172      *   10       DO5-SREDN.           PIC   9(4)
173      *   10       DO5-RSREDN           PIC   9
174      *   10       DO5-GRUB             PIC   99
175      *   10       DO5-RGRUB            PIC   99
```

cd. zał. 8,2

```
176                 05          D05-WSK-GAT          PIC  X
177                 05          D05-STAN-AKT         PIC  9(7) USAGE COMP
178                 05          D05-STAN-PMAG        PIC  9(7) USAGE COMP
179                 05          D05-NORMA            PIC  X(20)
180                 05          D05-KTM              PIC  X(13)
181                 05          D05-PROD-KR          PIC  9(7) USAGE COMP
182                 05          D05-WDJN             PIC  9(5)
183                 05          D05-FILLER           PIC  X(13)
184             RECORD DR-BILPROD
185                 CODE 6
186                 LOCATION CALC DMSCALC
187                 IN DAN-WTPROD
188                 USING D06-SYMB-BILPROD
189                 DUPLICATES NOT
190                 WITHIN DA-WTPROD
191                 WITHIN DA-OLPROD
192                 WITHIN DA-ZNPROD
193                 RESERVE 2
194                 MODE ASCII
195                 05          D06-SYMB-BILPROD     PIC  X(19)
196             *    10         D06-GAT              PIC  X(6)
197             *    10         D06-SREDN            PIC  9(4)
198             *    10         D06-RSREDN           PIC  9
199             *    10         D06-GRUB             PIC  99
200             *    10         D06-RGRUB            PIC  99
201             *    10         D06-ROK              PIC  99
202             *    10         D06-MC               PIC  99
203                 05          D06-WSK-GAT          PIC  X
204                 05          D06-CZAS             PIC  9(7) USAGE COMP
205                 05          D06-CIEZ-ODL         PIC  9(7) USAGE COMP OCCURS
206                 05          D06-CIEZ-WYS         PIC  9(7) USAGE COMP
207                 05          D06-STAN-POCZ        PIC  9(7) USAGE COMP
208                 05          D06-CIEZ-STOR        PIC  9(5) USAGE COMP
209                 05          D06-IL-KRYST-P       PIC  9(3) USAGE COMP
210                 05          D06-CIEZAR-P         PIC  9(7) USAGE COMP
211                 05          D06-IL-KRYST-N       PIC  9(3) USAGE COMP
212                 05          D06-CIEZAR-N         PIC  9(7) USAGE COMP
213                 05          D06-PROD-ROK         PIC  9(7) USAGE COMP
214                 05          D06-FILLER           PIC  X(13)
215             RECORD DR-ODLEW
216                 CODE 7
217                 LOCATION CALC DMSCALC
218                  IN DAN-WTWYS
219                  USING D07-NR-ODLEWU
220                 DUPLICATES NOT
221                 WITHIN DA-WTWYS
222                 WITHIN DA-OLWYS
223                 WITHIN DA-ZNWYS
224                 MODE ASCII
225                 05          D07-NR-ODLEWU        PIC  X(6)
226             *    10         D07-PIEC             PIC  9
227             *    10         D07-NR-ODL           PIC  9(4)
228             *    10         D07-WER              PIC  X
229                 05          D07-GAT              PIC  X(6)
230                 05          D07-SREDN            PIC  9(3)
231                 05          D07-GRUB             PIC  99
232                 05          D07-DLUG             PIC  9(4)
233                 05          D07-DATA-ODL         PIC  9(6)
234                 05          D07-CIEZAR           PIC  9(4) USAGE COMP
235                 05          D07-KOD-SKLAD        PIC  99
236                 05          D07-IL-SZIUK         PIC  9(3) USAGE COMP
237                 05          D07-WSK-ODL          PIC  X
238                 05          D07-FILLER           PIC  X(5)
239             RECORD DR-OPMAG
240                 CODE 8
241                 LOCATION CALC DMSCALC
242                  IN DAN-WTWYS
243                 USING D08-DATA
244                 DUPLICATES ALLOWED
245                 WITHIN DA-WTWYS
246                 WITHIN DA-OLWYS
247                 WITHIN DA-ZNWYS
248                 MODE ASCII
249                 05          D08-DATA             PIC  9(4)
250                 05          D08-KOD-OP           PIC  X
251                 05          D08-NR-ZAM           PIC  X(8)
252                 05          D08-GAT              PIC  X(6)
253                 05          D08-SREDN            PIC  9(4)
254                 05          D08-RSREDN           PIC  9
255                 05          D08-GRUB             PIC  99
256                 05          D08-RGRUB            PIC  99
257                 05          D08-DLUG             PIC  9(5) USAGE COMP
258                 05          D08-CIEZ             PIC  9(6) USAGE COMP
259                 05          D08-NR-SPEC          PIC  9(10) USAGE COMP
260                 05          D08-NR-PKP-SAM       PIC  X(10)
261                 05          D08-FILLER           PIC  X(5)
262             RECORD DR-PLANT
263                 CODE 9
```

cd. zał. 8.2

```
264          LOCATION VIA POZ-PLANT
265          WITHIN DA-WTZAM
266          WITHIN DA-OLZAM
267          WITHIN DA-ZNZAM
268          MODE ASCII
269          05        DO9-ROK-MC          PIC  9(4)
270          05        DO9-ST-ZAM-W        PIC  X
271          05        DO9-CIEZ-KL         PIC  9(6) USAGE COMP
272          05        DO9-CIEZ-PL         PIC  9(6) USAGE COMP
273           05       DO9-WYK
274           10       DO9-CIEZ-WYK        PIC  9(6) USAGE COMP
275           10       DO9-CIEZ-WYS        PIC  9(6) USAGE COMP
276           10       DO9-CIEZ-STO        PIC  9(6) USAGE COMP
277          05        DO9-FILLER          PIC  X(6)
278      RECORD DR-MIESIAC
279          CODE 10
280          LOCATION CALC DMSCALC
281           IN DAN-WTPROD
282          USING DIO-MS
283          DUPLICATES NOT
284          WITHIN DA-WTPROD
285          WITHIN DA-OLPROD
286          WITHIN DA-ZNPROD
287          MODE ASCII
288          05        DIO-MS              PIC  99
289      ************
290      SET SECTION
291      SET KLI-NAG
292          CODE 1
293          MODE CHAIN
294          ORDER LAST
295          OWNER DR-KLIENT
296          MEMBER DR-NAGLOWEK..AUTOMATIC
297          SELECTION CURRENT SET
298      SET NAG-UWZ
299          CODE 2
300          MODE CHAIN
301          ORDER SORTED.
302          OWNER DR-NAGLOWEK -
303          MEMBER DR-UWZAM AUTOMATIC
304          ASCENDING DO3-KLUCZ-UW
305           DUPLICATES NOT
306          SELECTION CURRENT SET
307      SET NAG-POZ
308          CODE 3
309          MODE CHAIN
310          ORDER LAST
311          OWNER DR-NAGLOWEK -
312          MEMBER DR-POZAM AUTOMATIC OWNER
313          SELECTION CURRENT SET
314      SET PROD-POZ
315          CODE 4
316          MODE CHAIN
317          ORDER LAST
318          OWNER DR-PRODUKT -
319          MEMBER DR-POZAM AUTOMATIC OWNER
320          SELECTION CURRENT SET
321      SET PROD-BIL
322          CODE 5
323          MODE CHAIN
324          ORDER SORTED-
325          OWNER DR-PRODUKT -
326          MEMBER DR-BILPROD MANUAL
327          ASCENDING DO4-SYMB-BILPROD
328          DUPLICATES NOT
329          SELECTION CURRENT SET
330      SET POZ-PLANT
331          CODE 6
332          MODE CHAIN
333          ORDER SORTED.
334          OWNER DR-POZAM -
335          MEMBER DR-PLANT AUTOMATIC OWNER
336          ASCENDING DO9-ROK-MC
337          DUPLICATES NOT
338          SELECTION CURRENT SET
339      SET MIESC-BIL
340          CODE 7
341          MODE CHAIN
342          ORDER LAST
343          OWNER DR-MIESIAC
344          MEMBER DR-BILPROD MANUAL
345          SELECTION CURRENT SET

DBRKPT PRINTS
```

Rys. 8.2. Schemat w postaci źródłowej

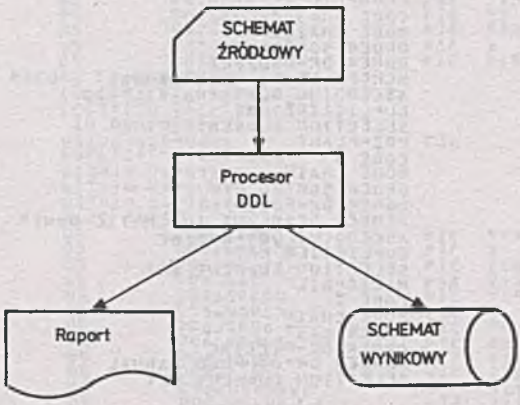
(SET SECTION). Opis każdego obszaru zawiera następujące informacje: kod obszaru, jego nazwę, wielkość (tzn. liczbę stron i wielkość strony), rodzaj i liczbę stron nadmiarowych, rodzaj ochrony obszaru, wskaźnik załadowania początkowego stron i liczbę łańcuchów obliczeniowych na stronie. Z kolei w sekcji opisującej rekordy znajdują się wszystkie dane charakteryzujące poszczególne rekordy w bazie. Są to przede wszystkim kod i nazwa rekordu oraz sposób jego alokacji. Do opisu rekordu dołączony jest opis wszystkich pól tworzących ten rekord przygotowany analogicznie jak w języku COBOL, a więc z charakterystyczną strukturą poziomów i opisem znakowym pola (klauzula PICTURE).

Kolejna sekcja opisuje wszystkie zdefiniowane grupy w bazie. Poza nazwą i kodem grupy zawarte są tutaj takie informacje, jak: typ grupy, logiczny punkt wstawienia i oczywiście nazwy rekordów właściciela i członka tej grupy. Ponadto przy opisie rekordu członkowskiego w grupie podaje się rodzaj powiązania (automatyczne lub manualne) oraz rodzaj wskaźników, jakimi jest on powiązany z innymi rekordami grupy. W tym również miejscu zdefiniować można główną i pomocnicze ścieżki dostępu do tego rekordu w bazie.

Prosty przykład programu definiującego bazę danych pokazuje załącznik 8.2. Jest to baza danych systemu planowania i kontroli produkcji dla Wydziału Wałków i Tulei w HMN "Szopienice".

8.2. Kompilacja schematu

Po napisaniu programu opisującego bazę danych można przystąpić do wygenerowania schematu wynikowego. Jest to wykonywane oczywiście przez procesor - kompilator języka DDL. Przebieg kompilacji schematu pokazuje rys. 8.3.



Rys. 8.3. Przebieg kompilacji schematu

Wejściem jest oczywiście schemat źródłowy przygotowany w języku DDL. Procesor DDL wykonuje analizę formalnej poprawności syntaktycznej i zamienia opisy obiektów na postać zakodowanych tablic wynikowych. Ponadto konstruuje tablice krzyżowe wszystkich nazw i odniesień do nich, co pozwala wykryć błędy merytoryczne w opisie danych.

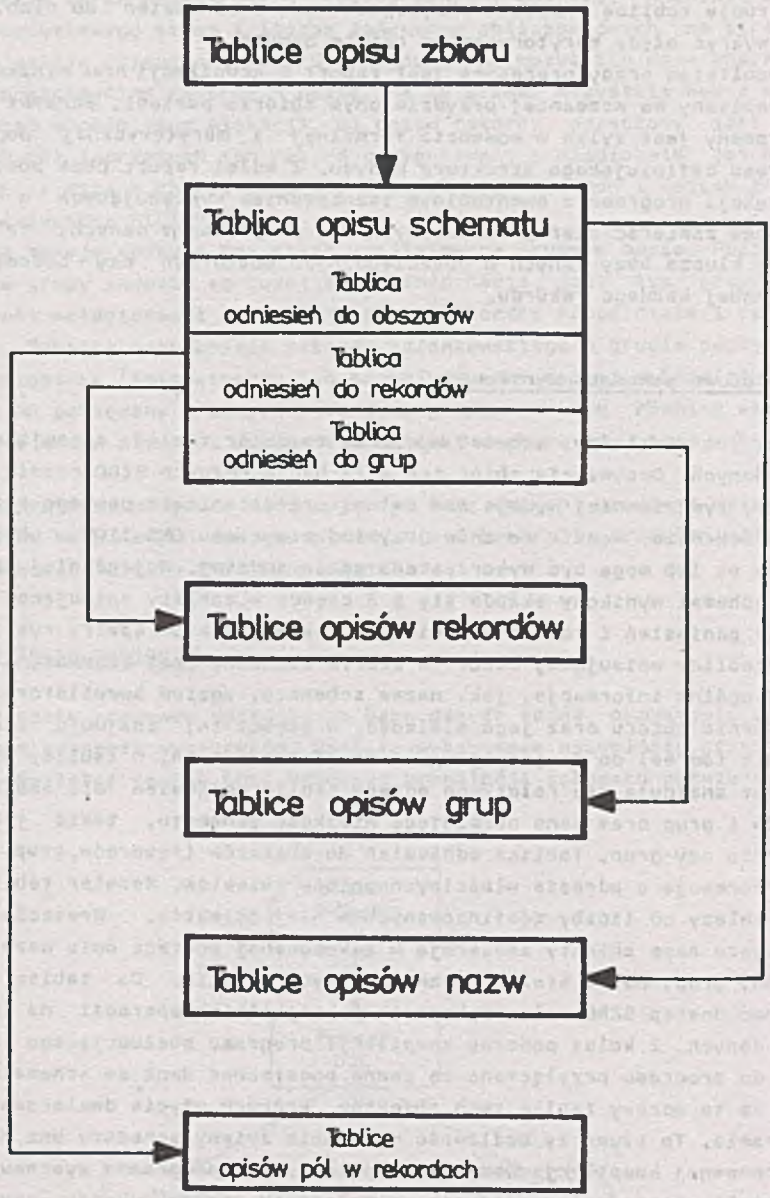
Rezultatem pracy procesora jest raport z kompilacji oraz wynikowy schemat zapisany we wcześniej przydzielonym zbiorze pamięci. Schemat wynikowy generowany jest tylko w momencie formalnej i merytorycznej poprawności programu definiującego strukturę danych. Z kolei raport poza powtórzeniem instrukcji programu z ewentualnym zaznaczeniem występujących w nich błędów może zawierać szereg użytecznych dla projektanta danych, takich jak: budowę klucza bazy danych w poszczególnych obszarach czy budowę części systemowej każdego rekordu.

8.3. Budowa schematu wynikowego

Jak już mówiliśmy, schemat wynikowy to zbiór tablic opisujących pewną bazę danych. Oczywiście zbiór ten może być w różnych SZBD różnie zorganizowany, tym niemniej wydaje się celowe przedstawienie pewnego rozwiązania w tym zakresie. Będzie to znów przykład z systemu DMS-1100, choć podobne zasady są lub mogą być wykorzystane gdzie indziej. Najogólniej rzecz biorąc, schemat wynikowy składa się z 3 części - tablicy opisującej zbiór, tablic odniesień i tablic właściwych opisów obiektów (patrz rys. 8.4).

W tablicy opisującej zbiór, w którym zapisany jest schemat, są najbardziej ogólne informacje, jak: nazwa schematu, poziom kompilatora DDL, czas utworzenia zbioru oraz jego wielkość. W części tej znajduje się również łącznik (adres) do drugiej grupy - tablic odniesień. W tablicy opisującej schemat znajdują się relatywne adresy tablic odniesień dla obszarów, rekordów i grup oraz dane obrazujące wielkość schematu, takie jak: liczba rekordów czy grup. Tablice odniesień do obszarów (rekordów, grup) zawierają informacje o adresie właściwych opisów obiektów. Rozmiar tablic odniesień zależy od liczby zdefiniowanych w niej obiektów. Wreszcie tablice opisujące same obiekty zawierają w zakodowanej postaci opis wszystkich rekordów, grup, pól i stałych nazw używanych w bazie. Do tablic tych musi wykonać dostęp SZBD celem wykonania jakiejkolwiek operacji na rekordach bazy danych. Z kolei podczas kompilacji programu obsługującego bazę danych do programu przyłączane są pewne podstawowe dane ze schematu. Z reguły są to adresy tablic tych obiektów, których użycie deklarowane jest w programie. To tłumaczy możliwość wykonania zmiany schematu bez konieczności ponownej kompilacji wszystkich pracujących programów systemu.

Jeżeli zmiana polega jedynie na dołożeniu nowego rekordu czy grupy w taki sposób, aby zostały one umieszczone na końcu tablic odniesień (poprzez odpowiedni kod), adresy opisów wszystkich dotychczasowych obiektów



Rys. 8.4. Wewnętrzna struktura schematu

będą takie same jak poprzednio. Programy dotychczasowe będą więc pracować z nowym schematem bazy.

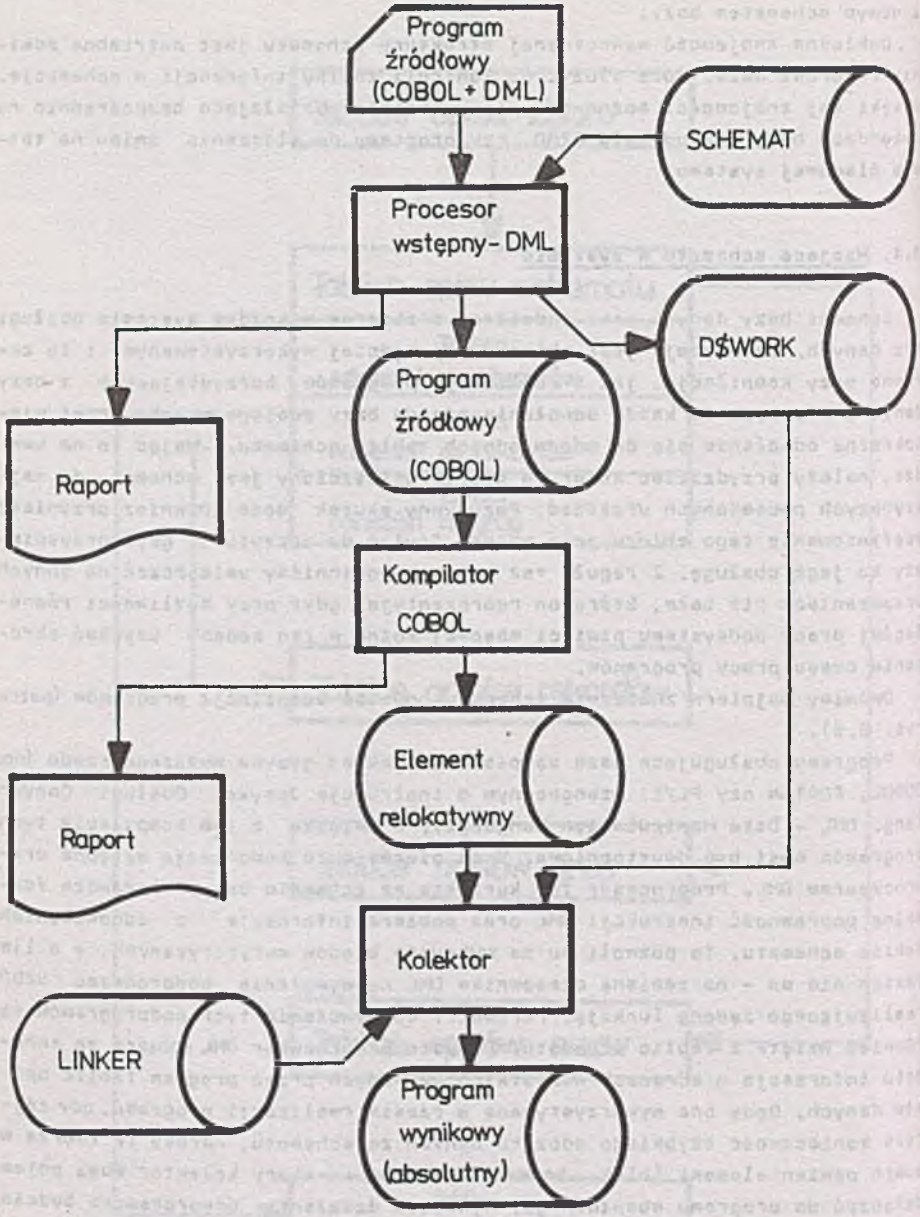
Dokładna znajomość wewnętrznej struktury schematu jest potrzebna administratorowi bazy. Może służyć do kontroli zapisu informacji w schemacie. Dzięki tej znajomości można też pisać programy działające bezpośrednio na rekordach bazy bez udziału SZBD, czy programy do śledzenia zmian na taśmie śladowej systemu.

8.4. Miejsce schematu w systemie

Schemat bazy danych jest podstawowym zbiorem w każdym systemie obsługi baz danych. Co więcej, jest zbiorem najczęściej wykorzystywanym i to zarówno przy kompilacji, jak i realizacji programów korzystających z bazy danych. Praktycznie każde odwołanie się do bazy pociąga za sobą nawet wielokrotne odwołanie się do odpowiednich tablic schematu. Mając to na uwadze, należy przydzielać zbiór, w którym umieszczony jest schemat, do najszybszych posiadanych urządzeń. Pozytywny skutek może również przynieść zdefiniowanie tego zbioru jako zbioru "tylko do odczytu", gdyż przyspieszy to jego obsługę. Z reguły też schemat powinniśmy umieszczać na innych urządzeniach niż baza, którą on reprezentuje, gdyż przy możliwości równoległej pracy podsystemu pamięci masowej można w ten sposób uzyskać skrócenie czasu pracy programów.

Omówimy najpierw znaczenie schematu podczas kompilacji programów (patrz rys. 8.5).

Programy obsługujące bazę są pisane w jakimś języku wyższego rzędu (np. COBOL, FORTAN czy PL/1) wzbogaconym o instrukcje Języka Obsługi Danych (ang. DML - Data Manipulation Language). W związku z tym kompilacja tych programów musi być dwustopniowa. Krok pierwszy to kompilacja wstępna preprocesorem DML. Preprocesor ten korzysta ze schematu bazy. Sprawdza formalną poprawność instrukcji DML oraz pobiera informacje z odpowiednich tablic schematu. To pozwoli mu na wykrycie błędów merytorycznych, a o ile takich nie ma - na zamianę czasownika DML na wywołanie podprogramu SZBD realizującego żądaną funkcję. Parametry do wywołania tych podprogramów są również wzięte z tablic schematu. Ponadto preprocesor DML pobiera ze schematu informacje o adresach wszystkich używanych przez program tablic opisów danych. Będą one wykorzystywane w czasie realizacji programu, gdy zajdzie konieczność szybkiego odczytu danych ze schematu. Adresy te tworzą w sumie pewien element (blok nazwany D$WORK), który kolektor musi potem dołączyć do programu absolutnego. Wynikiem działania preprocesora będzie oczywiście raport błędów i program źródłowy, w którym nie będzie już instrukcji DML. To pozwoli nam na kompilację programu właściwym procesorem językowym (np. COBOL). Kolejnym etapem jest praca kolektora, który z elementu relokatywnego programu (adresacja względna), tablicy adresów D$WORK i standardowego modułu LINKER tworzy wynikową postać programu. Moduł



Rys. 8.5. Przebieg kompilacji programu

LINKER zapewnia łączność programu z SZBD, zapewnia odpowiednie sterowanie wejściem do SZBD i powrotem z niego (np. na skutek błędu).

Realizacja programu już skompilowanego przebiega następująco. W momencie zgłoszenia się programu do SZBD (instrukcja IMPART) następuje skopiowanie tablic odniesień ze schematu bazy do buforów systemowych w pamięci operacyjnej. Oczywiście też zostaną załadowane wstępnie wszystkie systemowe tablice związane z wieloprogramową obsługą bazy, jak: tablice kontrolne dostępu, tablice żądań programów czy tablice związane z zabezpieczeniem informacji w bazie.

Jeżeli teraz w programie pojawi się odniesienie do bazy danych (np. instrukcja FIND RECORD - znajdź rekord), LINKER oddaje sterowanie do odpowiedniego podprogramu SZBD.

Jeżeli system do wykonania instrukcji potrzebuje dokładniejszych informacji ze schematu, ściąga do swojej części roboczej odpowiednie tablice. Po wykonaniu odwołania, np. po znalezieniu właściwego rekordu w bazie, SZBD oddaje sterowanie z powrotem do LINKERA. Ten sprawdza poprawność wykonania instrukcji i w zależności od niej przekazuje sterowanie do następnej instrukcji programu lub do procedury obsługującej błędy.

8.5. Podschemat bazy danych

W miarę rozwoju zastosowań baz danych powstało wygodne pojęcie modelu zewnętrznego. Opis modelu zewnętrznego tworzy schemat zewnętrzny, zwany też podschematem. Model zewnętrzny powszechnie porównywany jest do okna, przez które użytkownik widzi interesującą go część bazy danych. Ograniczenie zakresu "widzenia" bazy, a więc ograniczenie stopnia dostępności do niej, może mieć znaczenie przy ochronie danych przed niepowołanym dostępem.

Rozpatrzmy prosty przykład. Baza danych w pewnym przedsiębiorstwie zawiera informacje o wszystkich pracownikach. Dane te wykorzystywane są przez różne systemy informatyczne, np. kadrowy i finansowy. Jest oczywiste, że użytkownicy jednego systemu nie korzystają ze wszystkich informacji rekordu PRACOWNIK. Często jest to wręcz zabronione, np. dane opisujące zarobek pracownika mogą mieć charakter poufny, a więc powinny być niedostępne dla użytkowników systemu kadrowego. Jednym z rozwiązań takiego problemu będzie wprowadzenie podschematu, w którym zdefiniujemy nowy rekord pracownik wykrojony z oryginalnego rekordu bazy.

Użycie podschematu bazy może również wpływać na poprawę efektywności przetwarzania. Będzie to miało miejsce w systemach wyposażonych w języki manipulacji danymi osadzonymi w językach programowania, np. w COBOL-u. Programy użytkowe używają z reguły jedynie części bazy danych. Dla programów korzystających z tych samych obiektów można zdefiniować podschemat zawierający opisy tych obiektów. Opisy te są z reguły kopiowane do bufo-

rów programu czy tablic SZBD. Ograniczenie ich liczby zmniejszy zużycie pamięci operacyjnej.

Podobnie jak w przypadku schematu rozróżniamy podschemat źródłowy i wynikowy. Pierwszy z nich napisany jest w języku opisu danych zbliżonym do języka, w którym opisujemy całą bazę danych. Podschemat wynikowy powstaje po przetłumaczeniu podschematu źródłowego kompilatorem tego języka. Jest on przechowywany w specjalnym zbiorze (najczęściej dyskowym) w postaci odpowiednio przygotowanych tablic. Wszystkie uwagi dotyczące roli schematu w czasie kompilacji i realizacji programów użytkowych można przenieść na podschemat. W trywialnym zresztą przypadku podschemat może obejmować całą bazę danych i wtedy jest on prawie identyczny ze schematem.

Język opisu danych w podschemacie musi umożliwiać zdefiniowanie wszystkich obiektów używanych przez grupę programów (podsystem). Wiadomo, że obiektami tymi są obszary, rekordy i grupy logiczne. Weźmy za przykład język SDDL (Subschema Data Definition Language) wchodzący w skład systemu DMS-1100. Nie omawiając dokładnej składni tego języka (patrz zał. 8.6), zwróćmy tylko uwagę na najbardziej charakterystyczne jego cechy. Najistotniejszą sprawą jest wskazanie oryginalnego schematu bazy oraz identyfikowanie tego podschematu przez SZBD. Deklaracja ta jest jedną z pierwszych instrukcji podschematu źródłowego i ma ona postać:

SUBSCHEMA NAME IS ..... IN FILE ..... OF SCHEMA .....

Przy deklaracjach wstępnych definiuje się również główny język programowania, tzn. język w jakim będą pracować programy korzystające z podschematu. Następnie definiowane są wszystkie obiekty podschematu, a więc obszary, grupy, rekordy i nawet specjalne pola robocze zdefiniowane w schemacie przeznaczone do przenoszenia nazw obszarów, rekordów itp. Prosta składnia pozwala na skopiowanie do podschematu wszystkich, wszystkich poza wskazanymi lub tylko wskazanych obiektów bazy. Na szczególną uwagę zasługuje opis rekordu w podschemacie. Poza formalną zmianą jego nazwy dysponujemy możliwością wyboru pól tworzących ten rekord. Co więcej, możemy zmieniać opisy tych pól stosując zasady redefiniowania zgodne z założeniami głównego języka programowania, na który zdecydowano się w podschemacie.

Współpraca SZBD z podschematem zawiera pewne ciekawe elementy. Każda instrukcja DML działa dwufazowo, właściwie na dwóch różnych obiektach bazy. Załóżmy na przykład, że program korzystający z podschematu czyta rekord bazy danych. Wtedy SZBD znajduje stronę z tym rekordem i przenosi ją w oryginalnej postaci do buforów systemowych. Dopiero teraz następuje odszukanie oryginalnego rekordu na stronie i wycięcie z niego części opisanej w podschemacie. W rezultacie do programu przenoszony jest już nowy, okrojony rekord zawierający tylko niektóre pola z wyjściowego rekordu. Program może czytać czy modyfikować ten rekord, ale tylko w tej części, do której ma dostęp.

Appendix B. Complete Syntax Skeleton

```
IDENTIFICATION DIVISION

  SUBSCHEMA NAME IS subschema-name IN { FILE file-name-1 }
                                      { TIP FILE TIP-file-code-1 }

  OF SCHEMA schema-name

  [ SUBSCHEMA QUALIFIER IS qualifier-name-1 ]

                        { ASCII COBOL }
  HOST LANGUAGE IS      { FORTRAN     }
                        { PL1         }
                        { QLP         }

  [ KEY FOR COPY IS literal-1 ]

  [ LOCK FOR INVOKE IS literal-2 ]

DATA DIVISION

    [ DATA NAME SECTION ]

FORTRAN SUBENTRY:

  [ DATA NAMES ARE { ALL [ EXCEPT data-base-data-name-1 [,data-base-data-name-2 ]...] }
                   { data-base-data-name-3 [ ,data-base-data-name-4 ] . . . }
  [ WITHOUT MAPPING ]
    DATA NAME IS data-base-data-name-5 { FROM data-base-data-name-6 } ...
                                       { WITHOUT MAPPING            }
  DATA { data-base-data-name-7 [,data-base-data-name-8 ] .../literal-1 [,literal-2 ] .../ }... ]
```

PL/1 SUBENTRY:

```
DATA NAMES ARE { ALL [ EXCEPT data-base-data-name-1 [,data-base-data-name-2]...] }
               { data-base-data-name-3[ ,data-base-data-name-4 ]...
[ WITHOUT MAPPING ]
DATA NAME IS data-base-data-name-5 [ FROM data-base-data-name-6 ] ...
                                   [ WITHOUT MAPPING ]
```

ASCII COBOL SUBENTRY:

```
DATA NAMES ARE { ALL [ EXCEPT data-base-data-name-1 [,data-base-data-name-2]...] }
               { data-base-data-name-3 [ ,data-base-data-name-4 ]...

DATA NAME IS data-base-data-name-5[ FROM data-base-data-name-6 ]
  ITEM data-base-data-name-7
    VALUE IS [ ALL ] literal-1
    88 identifier-1 { VALUE IS } literal-2 [ THRU      ] literal-3 ... ...
                    { VALUES ARE }         [ THROUGH   ]
      [ literal-4 [ THRU      ] literal-5 ] ... ...
                  [ THROUGH   ]
```

OLP SUBENTRY:

```
DATA NAMES ARE { ALL [ EXCEPT data-base-data-name-1 [,data-base-data-name-2]...] }
               { data-base-data-name-3 [,data-base-data-name-4]...
[ DATA NAME IS data-base-data-name-5 FROM data-base-data-name-6]...
```

AREA SECTION

```
AREAS ARE { ALL [ EXCEPT area-name-1 [ , area-name-2 ]... ] }
          { area-name-3 [ , area-name-4 ]...

[ AREA NAME IS area-name-5 FROM area-name-6 ]...
```

RECORD SECTION

FORTRAN SUBENTRY:

```
RECORDS ARE { ALL [ EXCEPT record-name-1 [ ,record-name-2 ]... ] }
            { record-name-3 [ ,record-name-4 ]...

[ WITHOUT MAPPING ]
```

```
[ RECORD NAME IS record-name-5 [ FROM record-name-6 ]
  [ ITEMS ARE { ALL | EXCEPT data-base-identifier-1 [ ,data-base-identifier-2 ]...
                WITHOUT MAPPING
              { data-base-identifier-3 [ ,data-base-identifier-4 ] ...
  [ WITH data-base-identifier-5 FROM data-base-identifier-6 ] ...
  [ INTEGER variable-7 [ , variable-8 ] ...
  [ REAL { *4 } variable-9 [ , variable-10 ] ...
         { *8 }
  [ CHARACTER [ *n ] variable-11 [ , variable-12 ] ...
  [ EQUIVALENCE {( data-base-identifier-13 , variable-14 [ , variable-15 ] ... )} ...
  [ DATA{ data-base-identifier-16[,data-base-identifier-17]../literal-1[,literal-2]../}... ]
```

PL/1 SUBENTRY

```
RECORDS ARE { ALL [ EXCEPT record-name-1 [ ,record-name-2 ] ... ] }
            { record-name-3 [ ,record-name-4 ] ...                }

[ WITHOUT MAPPING ]

[ RECORD NAME IS record-name-5 [ FROM record-name-6 ]
  [ ITEMS ARE { ALL | EXCEPT data-base-identifier-1 [ ,data-base-identifier-2 ] ...
                WITHOUT MAPPING
              { data-base-identifier-3 [ ,data-base-identifier-4 ] ...
  [ WITH data-base-identifier-5 FROM data-base-identifier-6 ] ...
```

ASCII COBOL SUBENTRY

```
RECORDS ARE { ALL [ EXCEPT record-name-1 [ , record-name-2 ] ... ] }
            { record-name-3 [ , record-name-4 ] ...                }

[ RECORD NAME IS record-name-5 [ FROM record-name-6 ]
  ITEMS ARE { ALL [ EXCEPT data-base-identifier-1 [ ,data-base-identifier-2 ] ... ] }
            { data-base-identifier-3 [ ,data-base-identifier-4 ] ...               }
  [ WITH { data-base-identifier-5 } { data-base-identifier-entry }
         { FILLER                 } { redefines-entry-1         }
                                   { redefines-entry-2           }
  [ 66 identifier-1 RENAMES data-base-identifier-6 [ { THRU } data-base-identifier-7 ] ]
                                                     { THROUGH }
```

■ Where *data-base-identifier-entry* has the form:

```
[ FROM data-base-identifier-8
  VALUE IS [ ALL ] literal-1
  [ 88 identifier-2 { VALUE IS } literal-2 [ { THRU } literal-3 ]
                    { VALUES ARE }           { THROUGH }
    { literal-4 [ { THRU } literal-5 ] ... }
                  { THROUGH }
```

■ where *redefines-entry-1* has the form:

```
REDEFINES data-base-identifier-9

{ PIC     } IS character-string
{ PICTURE }

OCCURS integer-1 TIMES

USAGE IS { COMP   }
         { COMP-1 }
         { COMP-2 }
         { COMP-4 }
         { DISP   }
         { DISP-1 }

88 identifier-3 { VALUE IS  } literal-6 [ { THRU    } literal-7 ]
                { VALUES ARE }            { THROUGH }

    literal-8 [ { THRU    } literal-9 ] [ ... ] [ ... ]
              { THROUGH }
```

■ Where *redefines-entry-2* has the form:

```
REDEFINES data-base-identifier-10

[ level-number { data-base-identifier-11 }
               { FILLER                  }

  { PIC     } IS character-string
  { PICTURE }

  OCCURS integer-1 TIMES

  USAGE IS { COMP   }
           { COMP-1 }
           { COMP-2 }
           { COMP-4 }
           { DISP   }
           { DISP-1 }

  88 identifier-3 { VALUE IS  } literal-6 [ { THRU    } literal-7 ]
                  { VALUES ARE }            { THROUGH }

      literal-8 [ { THRU    } literal-9 ] [ ... ] [ ... ]
                { THROUGH }
]
```

OLP SUBENTRY

```
RECORDS ARE { ALL { EXCEPT record-name-1 [, record-name-2] ... } }
            { record-name-3 [, record-name-4] ...                }
```

cd. zał. 8.6

```
[ RECORD NAME IS record-name-5 [ FROM record-name-6 ]
  ITEMS ARE { ALL [ EXCEPT data-base-identifier-1 [,data-base-identifier-2 ] . . . ] }
            { data-base-identifier-3 [, data-base-identifier-4 ] . . . }
  | WITH { data-base-identifier-5 } FROM data-base-identifier-6 | . . .
    {    FILLER                  }
]
```

```
[ SET SECTION
    SETS ARE { ALL [ EXCEPT set-name-1 [ , set-name-2 ] . . . ] }
             { set-name-3 [ , set-name-4 ] . . . }
    [ SET NAME IS set-name-5 FROM set-name-6 ] . . .
]
```

OLP SECTION

```
[ SEGMENT NAME IS segment-name-1
  { THRU SET set-name-1 TO RECORD record-name-3 }
  | DATA QUALIFICATION REQUIRED | . . .
  | NO DATA RETRIEVAL ALLOWED |
]
```

```
{ PATH NAME IS path-name-1 [ PRIVATE ]
  ROOT IS RECORD record-name-4 | DATA QUALIFICATION REQUIRED |
                               | NO DATA RETRIEVAL ALLOWED |
    root-specification-clause
  [ { TO segment-name-2                                    }
    { THRU SET set-name-2 TO RECORD record-name-5          } . . . ] . . .
    { DATA QUALIFICATION REQUIRED                          }
    { NO DATA RETRIEVAL ALLOWED                            }
}
```

where *root-specification-clause* is:

```
[ DIRECT CURRENCY ASSUMED { DATABASE-KEY = literal- 1                          }
                          { AREA-NAME = area-name-1 PAGE-NUM = integer-1 } . . .
                          {     RECORD-NUM = integer-2                           }
  FETCH NEXT CURRENCY ALLOWED [ WITHIN area-name-2 [ [,] area-name-3 ] . . . ]
]
```

Rys. 8.6. Składnia SDDL (system DMS-1100)

Dostęp a zwłaszcza modyfikacja bazy przez programy korzystające z podschematu narzucają szereg oczywistych ograniczeń. Generalną zasadę można sformułować następująco: w podschemacie powinny być zdefiniowane wszystkie te obiekty bazy, z którymi może spotkać się SZBD przy wykonywaniu dowolnej instrukcji DML programu. Z tej zasady wynikają bardziej szczegółowe ograniczenia. Oto niektóre z nich:

- do podschematu muszą być włączone wszystkie rekordy typu "calc" z jednego obszaru, chociaż faktycznie w podschemacie potrzebne są tylko niektóre z nich,
- do podschematu muszą być włączone wszystkie typy rekordów członkowskich w grupach z większą liczbą typów członków,
- do podschematu muszą być dołączone specjalne obszary na indeksy czy tablice wskaźników, o ile rekord używany w podschemacie pamiętany jest metodą indeksowo-sekwencyjną lub uczestniczy w grupie z tablicą wskaźników,
- dla poprawnego wykonania instrukcji usuwania rekordu (DELETE) do podschematu powinien być dołączony cały rekord oraz cała związana z nim struktura grupowa,
- dla poprawnego wykonania instrukcji przeszukiwania obszaru do podschematu powinny być włączone wszystkie typy rekordów, które można pamiętać w tym obszarze (to samo dotyczy grupy),
- dla zapamiętania nowego rekordu konieczne jest skopiowanie do podschematu wszystkich kluczowych pól rekordu.

Mimo tych ograniczeń wprowadzenie podschematu ułatwia i upraszcza korzystanie z bazy danych. Najistotniejsze jest to, że użytkownik (programista) widzi bazę taką, jak jest mu to potrzebne, a nie taką, jaka jest ona faktycznie. Oczywiście zakres jego "spojrzenia" na bazę wynika z konkretnych potrzeb systemu, a formalną zgodę na korzystanie z części bazy użytkownik uzyskuje od administratora bazy danych.

9. ŁADOWANIE POCZĄTKOWE BAZY DANYCH

Po wygenerowaniu schematu bazy danych można przystąpić do pierwszego kroku jej eksploatacji, to jest do jej załadowania rekordami. Oczywiście musi to być zrealizowane już za pomocą programów współpracujących z bazą. Z reguły są to programy pracujące w trybie wsadowym, ale spotyka się też systemy użytkowe, w których przeważająca większość rekordów jest ładowana transakcjami pracującymi w czasie rzeczywistym. Nawet w takich systemach zachodzi jednak potrzeba wprowadzenia do bazy pewnych stałych czy nagłówkowych rekordów (np. tablic). Podstawowymi problemami, z jakimi spotkać się można przy ładowaniu bazy, są prawidłowe rozmieszczenie rekordów oraz minimalizowanie czasu pracy programów ładujących. W przypadku dużych baz danych o dużej liczbie wprowadzanych od początku rekordów czas pracy programów może być bardzo długi. W niniejszym rozdziale zaprezentujemy pewne techniki, dzięki którym przyspieszyć można te programy.

9.1. Przydział i inicjowanie obszarów

W rozdziale 7 przedstawiliśmy metodę obliczania wielkości bazy, a ściślej mówiąc - wielkości obszarów tworzących bazę. Przedstawiliśmy również pewne zasady kierujące łączeniem obszarów w fizyczne zbiory na konkretnych urządzeniach. Mając to na uwadze należy przystąpić do przydzielenia tych zbiorów, tzn. zgłoszenia ich do biblioteki głównej zbiorów systemu operacyjnego. Wielkość tych zbiorów wyznaczają wielkości tworzących je obszarów. Należy zauważyć, że w odróżnieniu od innych zbiorów systemu operacyjnego zbiory tworzące bazę nie mogą być automatycznie przez system operacyjny rozszerzane. Do tego celu służą specjalne metody, co zostanie omówione w rozdziale 10 dotyczącym reorganizacji bazy.

Po przydzieleniu obszarom fizycznego miejsca w pamięci masowej należy przystąpić do inicjowania obszarów. Jest to procedura przygotowująca obszar do pracy, polegająca na podziale przydzielonej pamięci na bloki jednakowej długości, zwane stronami i wygenerowaniu nagłówków tych stron. Procedura ta może mieć charakter zewnętrzny w stosunku do systemu użytkowego. Realizowana jest wtedy poprzez funkcję INITIALIZE procesora DMU. Procesor DMU jest specjalnie uprzywilejowanym programem w każdej bazie i jako taki musi być wygenerowany i skompilowany dla każdego schematu. Jeżeli używamy DMU, to zazwyczaj jest to pierwszy program, który pracuje w danej bazie.

Inną metodą inicjowania obszarów jest używanie odpowiedniego formatu czasownika DML - "OPEN USAGE INITIAL LOAD". Działa on tak samo jak INITIALIZE, jednak wydłuża nam jeszcze bardziej program ładujący. Należy zwrócić jeszcze uwagę na możliwość inicjowania jedynie części obszaru. Ma to miejsce przy rozszerzaniu obszaru o nowe strony. Wtedy też należy używać specjalnych operacji wskazujących systemowi, że strony początkowe są już wygenerowane i z reguły zapełnione danymi i że inicjowanie dotyczy jedynie pewnej liczby stron końcowych.

Po operacji zainicjowania obszarów można przystąpić do wprowadzenia rekordów do przygotowanych już stron bazy.

9.2. Kolejność ładowania rekordów

O prawidłowym rozmieszczeniu rekordów w bazie decydują w zasadzie dwa czynniki - wskaźnik załadowania początkowego oraz kolejność zapamiętywania rekordów w obszarze.

O wskaźniku ładowania początkowego stron mówiliśmy już obszernie w poprzednich rozdziałach.

Pomówmy teraz o kolejności ładowania rekordów. Przede wszystkim najpierw powinny być ładowane do bazy rekordy direct. Jest to oczywiste, gdyż w ten właśnie sposób zabezpieczymy sobie przydzielenie rekordom założonych z góry adresów. Następnie należy ładować rekordy pamiętane poprzez procedurę obliczeniową. I tu pojawia się pierwszy problem. Jeżeli w danym obszarze zdefiniowana jest grupa logiczna z właścicielem o dostępie calc i członkiem zapamiętywanym poprzez tę grupę, można ładować obszar na dwa sposoby. Pierwszy z nich to załadowanie najpierw wszystkich właścicieli calc, a następnie kolejno grupami wszystkich rekordów członkowskich. Będzie to dłużej trwało, lecz rekordy - właściciele grup znajdą się z reguły na stronach głównych i dostęp do nich będzie szybszy. Metodę tę można stosować w tych obszarach, w których nie ma zdefiniowanych stron nadmiarowych. Drugą metodą jest ładowanie obszaru grupami, tzn. zapamiętanie najpierw właściciela, a następnie wszystkich jego członków. Jest to metoda szybsza od poprzedniej. Może ona jednak spowodować pewne kłopoty przy dość dużym stopniu załadowania obszaru. Rekordy członkowskie mogą bowiem zająć całkowicie niektóre strony, co skutecznie uniemożliwi kolejnym rekordom - właścicielom znalezienie się na stronach głównych. Metodę tę należy więc stosować w obszarach o zdefiniowanych stronach nadmiarowych. Dobre efekty daje w tym przypadku umieszczenie rekordów członkowskich również na stronach nadmiarowych.

W szczególny sposób przebiega ładowanie początkowe obszaru, w którym są rekordy indeksowo-sekwencyjne. Jeżeli w obszarze oprócz tych rekordów mogą być przechowywane inne (np. pamiętane poprzez grupę), ładowanie rozpocząć należy od rekordów indeksowo-sekwencyjnych. Jak wiemy, dla każdego typu rekordu pamiętanego metodą indeksowo-sekwencyjną musi być wygenero-

wany i zainicjowany oddzielny obszar na indeksy. W czasie ładowania początkowego obszar na dane może być otwarty do inicjowania lub do aktualizacji, a obszar na indeksy do aktualizacji.

Zauważmy, że jest to jedyny moment, w którym aktualizować można strony w obszarze na indeksy. W późniejszych realizacjach, mimo faktycznej modyfikacji danych, obszar na indeksy jest zawsze otwierany tylko do czytania. Usuwanie czy wprowadzanie nowych rekordów nie zmniejsza nam stron indeksowych, co najwyżej komplikuje strukturę przez potrzebę łączenia rekordów w złożone łańcuchy (patrz rozdział 3).

Przy ładowaniu rekordów tworzących skomplikowaną strukturę sieciową należy mieć na uwadze fakt, aby ładować najpierw rekordy od góry struktury. Jedynym odstępstwem od tej zasady są grupy manualne, w których rekordy członkowskie mogą być umieszczane w bazie bez powiązań logicznych z rekordami właścicieli. Jednak nawet taką strukturę ładuje się "z góry na dół", gdyż sekwencyjne przetwarzanie tych grup będzie dużo szybsze niż późniejsze "podpinanie" rekordów członkowskich do odpowiednich właścicieli.

9.3. Techniki przyspieszające ładowanie

Jak już mówiliśmy, ładowanie dużej liczby rekordów do bazy może być procesem bardzo czasochłonnym. Zastanówmy się, w jaki sposób można skrócić czas programów ładujących. W zasadzie istnieją dwie techniki służące do tego celu - wstępne przetworzenie rekordów poza programami ładującymi i użycie specjalnego roboczego schematu do ładowania.

Wstępne przetwarzanie rekordów jest niekiedy niezbędne, jak np. w metodzie indeksowo-sekwencyjnej. Zbiór rekordów, które mają być zapamiętane w bazie, musi być posortowany według klucza zdefiniowanego w metodzie indeksowo-sekwencyjnej.

Jednak również w innych przypadkach wstępne przetwarzanie rekordów może być bardzo korzystne. Mamy tu na myśli posortowanie rekordów w grupach, w których logiczny punkt wstawienia jest ustalony jako "SORTOWANY". Wtedy też grupę można zdefiniować z logicznym punktem wstawienia, np. "NASTĘPNY", co znacznie przyspieszy zapamiętywanie nowych rekordów, a rekordy członkowskie i tak będą posortowane. Inną sprawą jest wstępne wybranie rekordów-duplikatów na zewnątrz programu ładującego. Mamy tu na myśli duplikaty zarówno w metodzie alokacji rekordów poprzez procedurę obliczeniową czy metodę indeksowo-sekwencyjną oraz duplikaty w grupach posortowanych. Jeżeli duplikaty w tych przypadkach nie są dozwolone, wymaga to przy ładowaniu nowego rekordu sprawdzenia całego łańcucha rekordów (łańcuch obliczeniowy czy grupa logiczna). Oczywiście będzie to wymagało wykonania szeregu dostępów fizycznych, co będzie trwało odpowiednio długo. Zdefiniowanie w roboczym schemacie klauzuli, że duplikaty są dozwolone,

znacznie przyspieszy tę procedurę. Nowe rekordy będą umieszczone w pierwszym napotkanym wolnym miejscu obszaru bez sprawdzania całego łańcucha rekordów.

Oba ostatnie przypadki dotyczą właściwie metody kombinowanej, a mianowicie równoczesnego zastosowania wstępnego przetwarzania rekordów i zmiany schematu bazy.

Pomówmy dokładniej o innych możliwościach, jakie daje nam zmiana schematu. Należy zaznaczyć w tym miejscu, że zmiana musi mieć charakter logiczny, który nie ma wpływu na fizyczną budowę rekordu. Na przykład nie można na czas ładowania wprowadzić do grupy dodatkowych wskaźników, gdyż zostaną one fizycznie wygenerowane w rekordzie i nowy schemat bez tych wskaźników w grupie nie będzie odpowiadał stanowi faktycznemu danych. Jedyna możliwość, jaką tu posiadamy, to zmiana porządku w grupach (a więc zmiana logicznego punktu wstawiania), zmiana sposobu alokacji czy zmiana wyboru właściwego wystąpienia grupy.

O zmianie porządku w grupach już mówiliśmy. Zauważmy, że najbardziej ekonomiczny jest logiczny punkt wstawienia "PIERWSZY", a więc - o ile można - należy go stosować w schematach roboczych do ładowania. Zmiana sposobu alokacji dotyczy właściwie jednej sprawy - zdefiniowania alokacji typu direct w miejsce metody pamiętania rekordów poprzez grupę. W ładowaniu początkowym, gdy dokładnie znamy, jakie i ile rekordów wprowadzamy do bazy, można pokusić się o próbę rozmieszczenia rekordów członkowskich pod ustalonymi (obliczonymi) adresami. Wprowadzanie rekordów direct do bazy jest zdecydowanie szybsze niż wprowadzanie rekordów poprzez grupę, gdyż omija się systemową "nadkładkę" poszukującą miejsce na nowy rekord. Natomiast zmiana alokacji rekordów calc lub indeksowo-sekwencyjnych w ogóle nie wchodzi w rachubę z uwagi na to, że metody te mają wyraźny "fizyczny" charakter (fizyczne łańcuchy obliczeniowe na stronach czy fizycznie istniejący obszar na indeksy).

Pomówmy jeszcze o jednej "przyspieszającej" metodzie polegającej na zmianie metody wyboru właściwego wystąpienia grupy. Jest to istotne w momencie wprowadzenia do bazy rekordów pamiętanych przez grupę. Należy wtedy wskazać, do którego wystąpienia grupy rekordy te mają być dołączone. Mamy dwie możliwości wyboru - przez określenie rekordu właściciela lub przez ustalenie stanu bieżącego w grupie. Sposób pierwszy nadaje się zwłaszcza przy przetwarzaniu swobodnym (w czasie rzeczywistym). Do ładowania początkowego bazy w takich przypadkach lepiej jednak użyć wyboru grupy przez określenie stanu bieżącego. Przy sekwencyjnym wprowadzaniu rekordów, które ma miejsce podczas ładowania początkowego, kolejno wprowadzane rekordy stają się rekordami bieżącymi w grupie. Nowe rekordy mogą być umieszczane bezpośrednio po nich bez potrzeby ponownego znajdywania właściciela. Będzie to metoda dużo szybsza od poprzedniej.

Kolejną techniką, która może nawet znacznie przyspieszyć ładowanie, jest rezygnacja z zabezpieczenia obszarów podczas pracy programów ładujących.

Odbywa się to znów na poziomie schematu, a więc i w tym przypadku konieczne jest użycie schematu roboczego tylko do ładowania. Rezygnacja z zabezpieczenia dotyczy wszystkich lub tylko niektórych obszarów. Można zrezygnować z taśmy śladowej systemu a nawet z zabezpieczenia obszarów w pamięci masowej. Pociąga to za sobą określone ryzyko konieczności powtarzania w razie awarii całego ładowania od początku. Pewnym przed tym zabezpieczeniem może być pobieranie "dumpów" bazy (lub tylko niektórych obszarów) po zakończeniu pracy kolejnych programów ładujących. Po załadowaniu obszarów należy zdefiniować i wygenerować nowy schemat, który używany będzie w eksploatacji. W schemacie tym wszystkie obiekty będą posiadały już docelowy opis oraz we właściwy sposób zostanie określona ochrona obszarów danych. To faktycznie kończy proces implementacji bazy. Można przystąpić do jej eksploatacji.

10. REORGANIZACJA BAZY DANYCH

Reorganizacja bazy danych jest najtrudniejszym problemem, z jakim styka się administrator bazy. Trudność wynika z konieczności automatycznego przeniesienia do nowej zreorganizowanej bazy wszystkich rekordów umieszczonych dotychczas w starej bazie danych. W razie dużej liczby rekordów oraz skomplikowanego ich powiązania między sobą w strukturze sieciowej proces ten jest pracochłonny i trudny do oprogramowania. Zastanówmy się nad przyczynami reorganizacji bazy. Najczęstszym przypadkiem jest pogorszenie się warunków eksploatacji istniejących programów. Chodzi tutaj o wydłużenie się czasu ich pracy, czego przyczyną jest wzrost nieuporządkowania danych w bazie. Przyczyną wzrostu nieuporządkowania jest z kolei najczęściej wzrost stopnia załadowania obszarów, który pociąga za sobą konieczność umieszczenia rekordów na stronach obcych (nadmiarowych), powoduje zwiększenie systemowej "nadkładki" w pamięci masowej na dodatkowe wskaźniki czy niewykorzystane punkty wejścia na stronie. Reorganizacja fizyczna, tzn. zmiana struktury fizycznej danych bez zmiany ich zawartości informatycznej, może znacznie poprawić warunki eksploatacji na skutek lepszego wykorzystania przydzielonej pamięci masowej. Jeżeli reorganizacja ta połączona będzie z powiększeniem obszarów, skutki będą jeszcze lepsze.

Inne przyczyny powodują reorganizację logiczną, tzn. zmianę bądź samych rekordów czy ich pól, bądź zmianę istniejących między rekordami powiązań poprzez grupy logiczne. Mamy tutaj na myśli dołączanie lub skreślanie grup, jak i zmianę wskaźników używanych w grupach. Przyczyny reorganizacji logicznej są na ogół związane ze zmianą koncepcji projektu. Wynika to z niemożności przewidzenia wszystkiego na etapie projektowania, co jest szczególnie trudne w przypadku projektowania systemu dla nowego, jeszcze nie pracującego zakładu (wydziału). Innym powodem mogą być błędy popełniane przez projektanta wynikające z małego doświadczenia, błędnej analizy danych czy błędnych informacji uzyskanych od użytkownika. Pewne ograniczenie narzuca też eksploatacja systemu. Programy czasochłonne a mniej potrzebne z biegiem czasu mogą stać się całkowicie bezużyteczne, zwłaszcza jeżeli te same co w nich informacje można uzyskać łatwiejszą i szybszą drogą.

Pojawić się wreszcie mogą nowe żądania użytkownika pod adresem systemu, a co za tym idzie - bazy danych. Jest to bardzo częsty przypadek. Użytkownik dopiero po kilkumiesięcznej czy dłuższej eksploatacji lepiej poznaje możliwości i zalety systemu. Jest więc rzeczą naturalną, że będzie on chciał sobie w dalszym ciągu ułatwić czy usprawnić pracę. To wszyst-

ko powoduje konieczność rozładowania bazy danych, tzn. przeniesienie istniejących w bazie rekordów na pewien nośnik i ponownego załadowania ich do bazy już w nowej postaci i do znanej struktury.

Podstawowym problemem reorganizacji jest czas jej trwania. Mamy tu na myśli zarówno czas potrzebny na przygotowanie i przetestowanie programów rozładowujących bazę i ładujących ją, jak i czas maszynowy trwania samej reorganizacji. W przypadku dużych baz danych czas ten może być długi i podstawowym zadaniem administratora jest jego skrócenie, aby do minimum ograniczyć zablokowanie bazy przed normalnymi użytkownikami.

Pewne techniki czy metody pozwalające na skrócenie pracy programów rozładowujących i ładujących zostaną przedstawione poniżej. Zastanówmy się najpierw nad problemem, jak zapobiegać reorganizacji, jak przewidzieć przyszły rozrost systemu już na etapie projektowania bazy. Możliwości jest kilka. Pierwszą z nich jest zdefiniowanie w rekordzie pola FILLER. Oczywiście w początkowym okresie nie będzie ono wykorzystane, ale w razie potrzeby rozszerzenia rekordu o nowe pole można użyć dla tego celu pola FILLER. Najistotniejsze jest to, że fizycznie rekord nie ulegnie zmianie (tj. zwiększeniu) i w związku z tym nie ma potrzeby go przeładowywać. Należy zmienić jedynie opis tego rekordu w schemacie, co pociągnie za sobą konieczność kompilacji schematu, jak i używających tego pola programów. Aby tego uniknąć, zamiast pola FILLER można wprowadzić do rekordu normalnie nazwane pole, które można później wykorzystać do dowolnego celu. Najbardziej elastycznym rozwiązaniem jest zdefiniowanie w rekordzie tablicy na określoną liczbę znaków (tzn. elementem tablicy jest pojedynczy znak). Pozwoli to na dowolne wykorzystanie pojedynczych znaków na właściwe wskaźniki, a w razie potrzeby na odpowiednie ich łączenie, pakowanie czy konwersję.

Inną z metod zabezpieczenia się przed rozrostem systemu jest definiowanie grup logicznych, które, choć na razie nie są wykorzystywane, w przyszłości mogą się przydać. Oczywiście grupy takie należy wtedy definiować jako grupy manualne. Pozwoli to na przechowywanie rekordów nie związanych ze sobą logicznie, a z drugiej strony daje możliwość łatwego później wykorzystania zarezerwowanych w nagłówku rekordu słów na wskaźniki w grupach. Pewne grupy definiowane niejako na wyrost mogą zresztą mieć swoje projektowe uzasadnienie, np. przez przewidywane rozszerzenie systemu o nowe zadania, funkcje czy nawet podsystemy.

Łatwiejsze jest zabezpieczenie się przed koniecznością reorganizacji fizycznej. Można tego dokonać przez przyjęcie właściwego zapasu stron na dane, zdefiniowanie stron nadmiarowych czy wreszcie zostawienie na każdej stronie miejsca poprzez ustawienie wskaźnika załadowania początkowego.

Można również zapewnić sobie równomierny rozkład rekordów w obszarze poprzez zastosowanie metody interwałowej. Polega ona na rozdzieleniu rekordów tego samego typu określoną liczbą stron pustych (co może mieć zna-

czenie przy wstępnym zapamiętywaniu rekordów-właścicieli grup) i pozostawieniu miejsca na ich rekordy członkowskie.

Niekiedy jednak przyjęty zapas stron jest za mały i obszar mimo wszystko trzeba poszerzyć. Aby ułatwić tę operację, należy już w momencie definiowania obszaru przewidzieć, do jakiej liczby stron obszar ten może się rozszerzyć. Wtedy też SZBD definiuje odpowiedni rozpis klucza bazy danych dla tego obszaru biorąc pod uwagę nie faktycznie zgłoszoną liczbę stron, lecz liczbę docelową, do której kiedyś system może się rozszerzyć.

W przypadku faktycznego rozszerzenia obszaru wszystkie wskaźniki w grupach pozostają bez zmian. Dobrym rozwiązaniem jest przyjęcie wspólnej maksymalnej liczby dla wszystkich obszarów bazy. Spowoduje to zdefiniowanie jednakowych kluczy bazy danych we wszystkich obszarach, co znacznie ułatwi czytanie "dumpów" czy kopii stron na taśmie śladowej systemu.

10.1. Rozszerzanie obszarów

Istnieją dwie metody powiększania obszaru: poprzez zwiększenie strony lub zwiększenie liczby stron w obszarze. W obu przypadkach należy przede wszystkim poszerzyć fizycznie zbiór, w którym znajduje się modyfikowany obszar. Łatwiejszą z obu metod jest zwiększenie wielkości strony. Oczywiście należy przy tym wziąć pod uwagę wszystkie uwagi dotyczące wielkości strony przedstawione w rozdziale 7.3. Gdy zdecydujemy się na tę metodę i obliczymy w jakiś sposób nowy rozmiar strony, należy fizycznie rozszerzyć każdą stronę powiększając ją o określoną liczbę słów. Zauważmy, że nowe słowa muszą wejść przed nagłówek dolny strony, przez co powiększą one liczbę wolnych słów na końcu strony. Równocześnie modyfikacji muszą ulec pewne słowa w nagłówku strony, takie jak: wielkość strony czy wielkość wolnego miejsca na stronie. Czynności te można wykonać zewnętrznie w stosunku do systemu (tzn. przez własne programy w języku wewnętrznym) lub przez standardową procedurę SZBD o ile taka istnieje.

Na przykład w przedstawionym już systemie DMS-1100 funkcję tę spełnia komenda EXPAND standardowego, specjalnie uprzywilejowanego programu DMU.

Zwiększenie liczby stron w obszarze jest już bardziej skomplikowane. W odróżnieniu od poprzedniej metody jest to operacja wpływająca na alokację rekordów metodą calc. W większości przypadków procedury obliczeniowe w istotny sposób uwzględniają liczbę dostępnych stron w obszarze. W związku z tym rozszerzenie obszaru wymaga wcześniejszego rozładowania rekordów calc z tego obszaru. Po rozszerzeniu obszaru należy ponownie załadować rozładowane rekordy. Procedury obliczeniowe z reguły przydzielą tym rekordom calc zupełnie inne niż poprzednio adresy.

Drugim kłopotliwym problemem są strony nadmiarowe, o ile takie były zdefiniowane w obszarze. Generalna zasada jest taka - po rozszerzeniu strony z danymi muszą one pozostać stronami na dane, a strony nadmiarowe - nadmiarowymi. Jeżeli w obszarze zdefiniowane były strony nadmiarowe lo-

kalne, kłopotu nie ma. Wystarczy powiększyć obszar o pewną liczbę stron, a system sam określi, które z nich są na dane, a które nadmiarowe. Jeżeli w obszarze zdefiniowano nadmiar globalny na końcu obszaru, najłatwiej jest powiększyć liczbę stron nadmiarowych. Jednak często zachodzi potrzeba zdefiniowania nowych stron na dane. O ile można powiększyć ten obszar co najmniej dwukrotnie wystarczy zmienić rodzaj stron nadmiarowych w obszarze z globalnego na lokalne w taki sposób, aby wszystkie dotychczasowe strony z danymi pozostały bez zmian, a istniejący dotychczas nadmiar globalny był pierwszą częścią podzielonego nadmiaru lokalnego. Jest to skuteczne rozwiązanie i dzięki niemu można niekiedy uniknąć konieczności reorganizacji obszaru.

Zauważmy również, że można poszerzać tylko ten obszar, który ma z góry zadeklarowaną zdolność rozszerzania się do pewnej maksymalnej liczby stron. Wiąże się to z koniecznością zachowania klucza bazy danych tego obszaru (zależnie od liczby stron obszaru) we wszystkich zapamiętanych w nim rekordach. Klucz bazy danych jest wykorzystywany jako wskaźnik w grupach logicznych i łańcuchach obliczeniowych. Innymi słowy, oznacza to, że rekordy po rozszerzeniu nie zmienią swojego adresu w bazie.

Fizycznie rzecz biorąc, rozszerzanie obszaru o nowe strony polega na zainicjowaniu tych nowych stron.

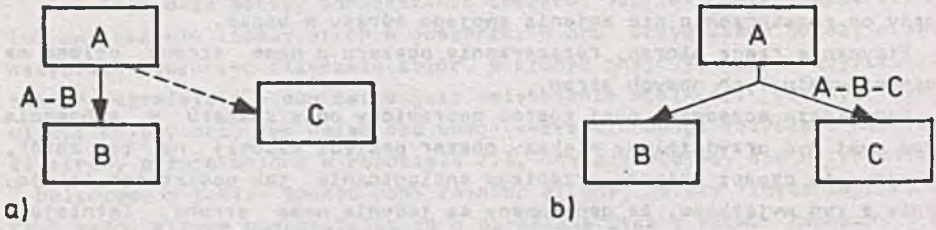
Oczywiście wcześniej musi zostać poprawiony opis obszaru w schemacie oraz musi być przydzielony większy obszar pamięci masowej na ten zbiór. Inicjowanie części obszaru przebiega analogicznie jak początkowe inicjowanie z tym wyjątkiem, że generowane są jedynie nowe strony. Istniejące strony z danymi nie są zmieniane, gdyż system w schemacie bazy ma zapisaną liczbę stron już zainicjowanych (ang. PRE-INITIALIZED PAGES).

Należy zwrócić uwagę jeszcze na jeden fakt. Po rozszerzeniu obszaru z reguły nie można odtworzyć bazy danych do wcześniejszego momentu. Wiąże się to ze zmianą wielkości strony lub zmianą adresów rekordów calc. W związku z tym po rozszerzeniu i ewentualnym załadowaniu bazy nowymi rekordami należy pobrać i zabezpieczyć "dump" całej bazy danych. Będzie to najwcześniejszy moment, do którego można będzie później doprowadzić odtwarzanie bazy.

10.2. Reorganizacja ... bez reorganizacji

Niekiedy zdarza się, że można wykonać faktyczną reorganizację logiczną bazy bez konieczności rozładowania i ponownego ładowania rekordów. Może to mieć miejsce wtedy, gdy reorganizacja polega na dołożeniu do bazy danych nowego obszaru, rekordu czy grupy łączącej nowo zdefiniowane rekordy. Wystarczy wtedy zaktualizować schemat bazy i ewentualnie zainicjować nowy obszar. Przy dopisywaniu do schematu nowych obiektów powinniśmy nadawać im kody bezpośrednio wyższe od już zdefiniowanych, nawet gdy w międzyczasie zwolniły się kody (numery) wcześniejsze.

Stosowanie tej zasady zwalnia od konieczności ponownej kompilacji wszystkich programów bazy. Skompilować należy jedynie schemat i te programy, które będą używały nowych obszarów (rekordów, grup). Trudniejszą sprawą jest zdefiniowanie nowej grupy logicznej między istniejącymi już rekordami. Wiąże się to bowiem z koniecznością fizycznego poszerzenia części systemowej rekordu o nowe wskaźniki. Jeżeli rekord ma zarezerwowane wskaźniki na grupy manualne i jeżeli tych wskaźników jest wystarczająca liczba (tzn. rekord nie występuje jednocześnie we wszystkich grupach manualnych, w których może), to sprawę rozwiązuje zdefiniowanie nowej grupy manualnej, która wykorzysta już istniejące wskaźniki. Oczywiście kod tej nowej grupy powinien być znów wyższy niż jakikolwiek inny w schemacie (patrz rozdział 8.3). Często jednak zdarza się, że rozwiązania tego z przyczyn formalnych nie można zastosować. Wtedy można zastosować inną metodę. Załóżmy zgodnie z rys. 10.1, że w pewnym obszarze zapamiętane są



Rys. 10.1. Przykład reorganizacji - dokładanie nowej grupy

dwa rekordy A i B powiązane grupą A-B. Załóżmy też, że do takiej struktury chcemy dołączyć nowy rekord C, który byłby powiązany z rekordem A grupą A-C. Bez faktycznej reorganizacji nie jest to możliwe do wykonania, gdyż łączy się ze zmianą wielkości rekordu A (dołożenie w nim wskaźników w nowej grupie A-C). Do tego celu można wykorzystać istniejącą grupę A-B. Wystarczy rozszerzyć tę grupę w ten sposób, aby dopuścić rekord C do udziału w charakterze członka tej grupy. W rezultacie otrzymamy grupę A-B-C, w której właścicielem jest rekord A, a członkami rekordy B i C. Rekord A nie ulegnie fizycznej modyfikacji, gdyż istniejący w nim wskaźnik w grupie A-B będzie teraz wskaźnikiem w grupie A-B-C. Jedyną niedogodnością może być wymieszanie rekordów typu B i C w konkretnym występieniu grupy. Można się przed tym zabezpieczyć programowo przy zapamiętaniu rekordów typu C (umieszczając je w grupie A-B-C po wszystkich rekordach B). Sekwencyjne przetwarzanie takich grup będzie łatwe, a zwłaszcza że DML z reguły dopuszcza wykonywanie operacji tylko na wskazanym typie rekordu z takiej grupy. Na przykład można szukać pierwszego rekordu typu C w grupie, czy następnego rekordu typu B itp. Aby wdrożyć takie rozwiązanie, należy skompilować schemat i wszystkie progra-

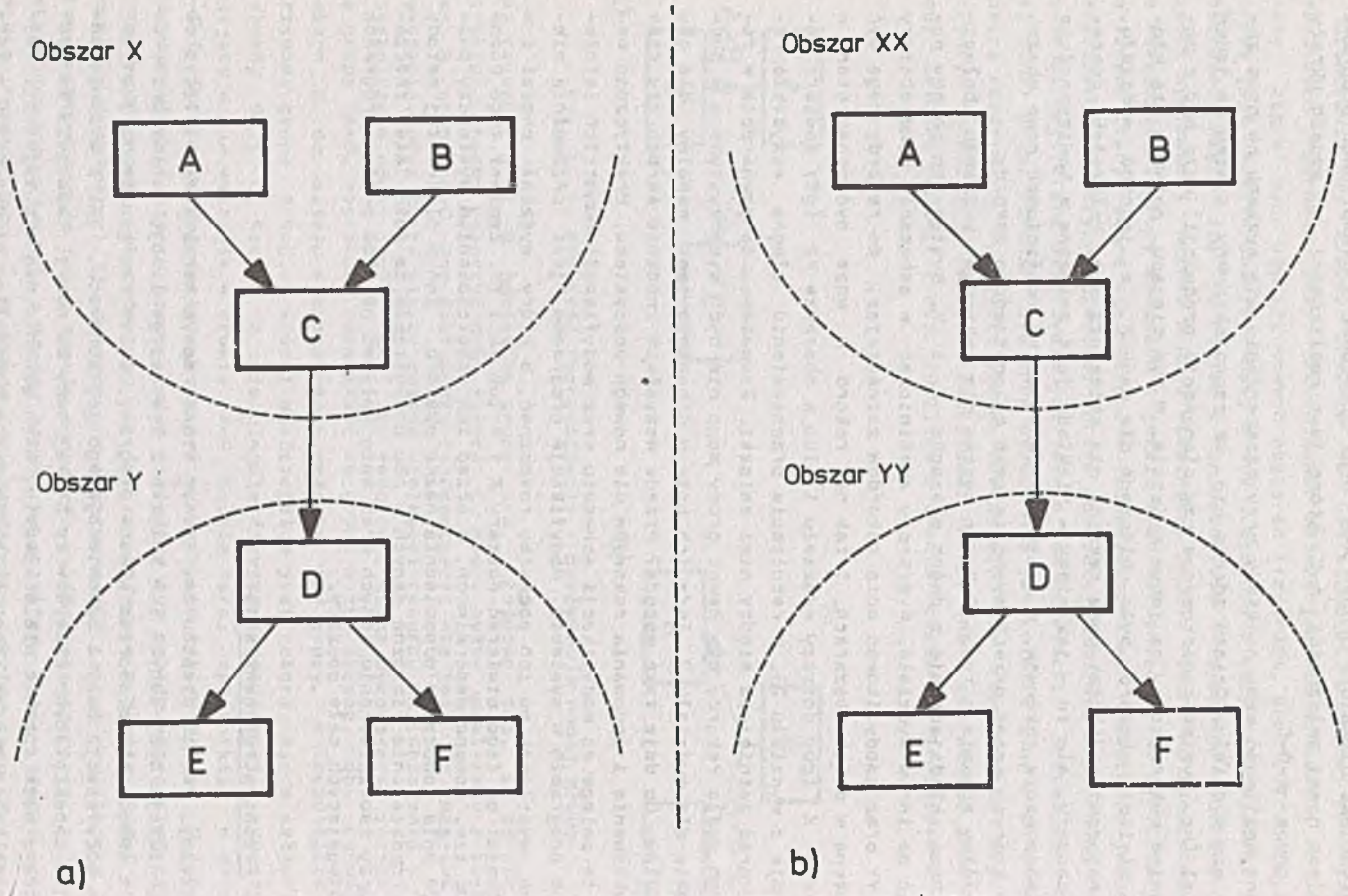
sy dotychczas używające grupy A-B oraz opracować i skompilować programy używające nowej zależności A-C, która jest realizowana w sposób ukryty przez grupę A-B-C.

Inną możliwość mamy niekiedy przy rozszerzaniu się systemu na nowe zakłady czy wydziały. Często zdarza się, że zaprojektowany system na jeden wydział (np. system raportowania czy planowania produkcji) może być z powodzeniem wykorzystany na innym wydziale. W tym miejscu pojawia się kłopot - wielkość obszarów była obliczona dla danych z jednego wydziału, więc najprawdopodobniej nowe rekordy nie zmieszczą się. Należałoby poszerzyć obszary, ale to - jak wiemy - niekiedy jest związane z koniecznością przeładowywania rekordów. W takim przypadku można zdefiniować nowe obszary, w których można przechowywać te same co poprzednio rekordy.

Załóżmy zgodnie z rys. 10.2, ze system dla jakiegoś wydziału obsługiwała baza składająca się z dwóch obszarów X i Y. System tan chcemy powielić na innym wydziale. Wystarczy zdefiniować w schemacie nowe obszary XX i YY oraz zmodyfikować opis rekordów zaznaczając, że rekordy mogą być pamiętane w dwóch obszarach. I tak np. rekord A może być zapamiętany w obszarze X (gdy dotyczy wydziału 1) lub w obszarze XX (gdy zawiera informacje z wydziału 2). W rezultacie przeniesieniu ulegną wszystkie rekordy oraz istniejące między nimi związki. Zauważmy, że pewne pola w rekordach, całe rekordy czy nawet grupy mogą nie być wykorzystywane w podsystemie dla wydziału 2, lecz ich opis w schemacie jest wspólny dla obu wydziałów. Co daje taka metoda? Przede wszystkim znacznie skraca się czas projektowania i ładowania rekordów dla nowego podsystemu. Praktycznie metoda ta polega na modyfikacji schematu oraz modyfikacji wszystkich istniejących programów w systemie. Modyfikacja programów jest przeważnie niewielka, wystarczy na ich początku rozpoznać, o który wydział chodzi i w zależności od tego otwierać obszary X i Y lub XX i YY. Zamiast tego można używać tzw. komend generalnych, a przed ich zastosowaniem wystarczy posłać do pola roboczego odpowiednią nazwę obszaru. Innym plusem tej metody jest rozdzielenie fizyczne danych z obu podsystemów, co może mieć znaczenie przy zabezpieczaniu danych oraz może wpływać na czas pracy programów przebiegających całe obszary.

10.3. Przebieg reorganizacji

Załóżmy, że nie dysponujemy żadnym standardowym narzędziem służącym do reorganizacji bazy danych i w związku z tym reorganizację musimy przeprowadzić samodzielnie poprzez własne programy. Celem reorganizacji jest zarówno modyfikacja bazy i odpowiadającego jej schematu, jak i przeniesienie już zapamiętanych rekordów ze starej bazy do nowej. Zwłaszcza ta druga sprawa może sprawić wiele kłopotu, więc głównie nią się zajmiemy. Najogólniej rzecz biorąc, reorganizację można podzielić na dwa etapy - rozładowanie bazy i ponowne jej ładowanie już w zmodyfikowanej formie.



Rys. 10.2. Przykład powielania systemu na nowe obszary
a) podsystem oryginalny dla wydziału 1, b) duplikat podsystemu dla wydziału 2

Faza pierwsza - rozładowanie bazy polega na skopiowaniu wszystkich rekordów bazy do specjalnego obszaru roboczego. W zależności od posiadanych urządzeń obszar roboczy może być zdefiniowany na dysku, w pamięci operacyjnej (dla małych baz danych) czy nawet na taśmie magnetycznej. W szczególnym przypadku, co podajemy poniżej, może to być nawet robocza baza danych.

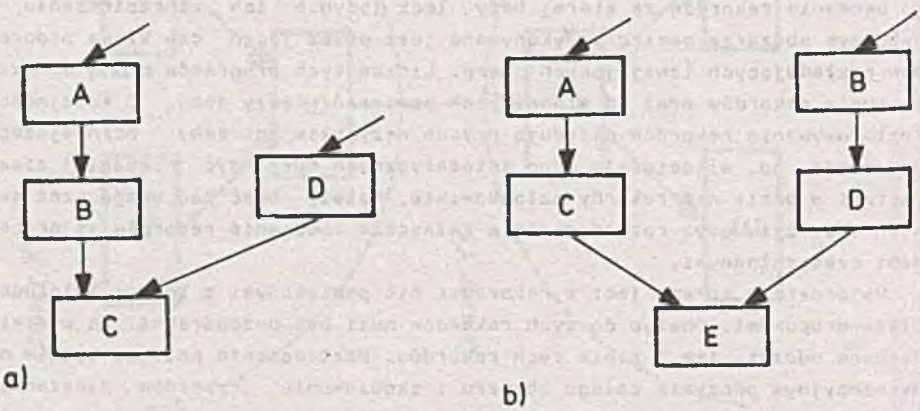
Rozładowanie obszarów jest terminem umownym; nie oznacza ono fizycznego usuwania rekordów ze starej bazy, lecz jedynie ich zabezpieczenie w roboczym obszarze pamięci. Wykonywane jest przez jeden czy kilka programów rozładujących (zwijających) bazę. Liczba tych programów zależy od liczby typów rekordów oraz od stopnia ich powiązań między sobą. O kolejności rozładowywania rekordów decydują przede wszystkim potrzeby późniejszego ładowania, np. właściciele grup automatycznych muszą być wcześniej zapamiętani w bazie niż rekordy członkowskie. Należy brać pod uwagę czas obu tych operacji, gdyż rozładowanie a zwłaszcza ładowanie rekordów są procesami czasochłonnymi.

Najprostsza sprawa jest z rekordami nie powiązanymi z innymi zależnościami grupowymi. Dostęp do tych rekordów musi być bezpośredni, co ułatwia zarówno odczyt, jak i zapis tych rekordów. Rozładowanie polegać będzie na sekwencyjnym odczycie całego obszaru i skopiowaniu rekordów wskazanego typu do obszaru roboczego. W obszarze roboczym rekordy mogą być nieco inaczej rozpisane niż w wyjściowej bazie danych (jeżeli taka jest potrzeba reorganizacji), więc kopiowania nie należy rozumieć dosłownie. Jeżeli w obszarze występuje więcej typów rekordów nie powiązanych ze sobą, to rozładowanie ich powinno być wykonane w jednym przebiegu. Wystarczy w tym celu czytać wszystkie rekordy obszaru (bez względu na typ) i wybierać te typy, które muszą być skopiowane. W obszarze roboczym rekordy różnych typów będą co prawda wymieszane, ale w niczym to nie przeszkodzi ich załadowaniu (z uwagi na dostęp bezpośredni do nich).

Rekordy powiązane ze sobą zależnością grupową najlepiej rozładowywać i ładować całymi grupami. Oznacza to, że w obszarze będziemy znajdywać kolejne rekordy właściciela, a następnie dla każdego z nich zapamiętywać jego rekordy członkowskie. Jest to sposób szczególnie efektywny, gdy dostęp do członka realizowany jest poprzez grupę, a logiczny punkt wstawiania w grupie zdefiniowano jako "NASTĘPNY". Obszar roboczy z reguły ma organizację sekwencyjną. Nawet gdy obszar roboczy jest właściwie roboczą bazą danych i przy kopiowaniu zachowujemy strukturę grupy, to rekord członkowski zawsze lokujemy poprzez tę grupę, nawet wtedy, gdy w oryginale jest to dostęp bezpośredni. Pozwoli to nam na lepsze wykorzystanie pamięci w obszarze roboczym i przyspieszenie zapisu (odczytu rekordów z tego obszaru).

Trudniejsza sprawa jest, gdy rekordy powiązane są ze sobą tworząc strukturę sieciową. Oczywiście i wtedy przy kopiowaniu czy ładowaniu należy posuwać się z góry na dół hierarchii (stąd nazwa "zwijanie"). Często jednak

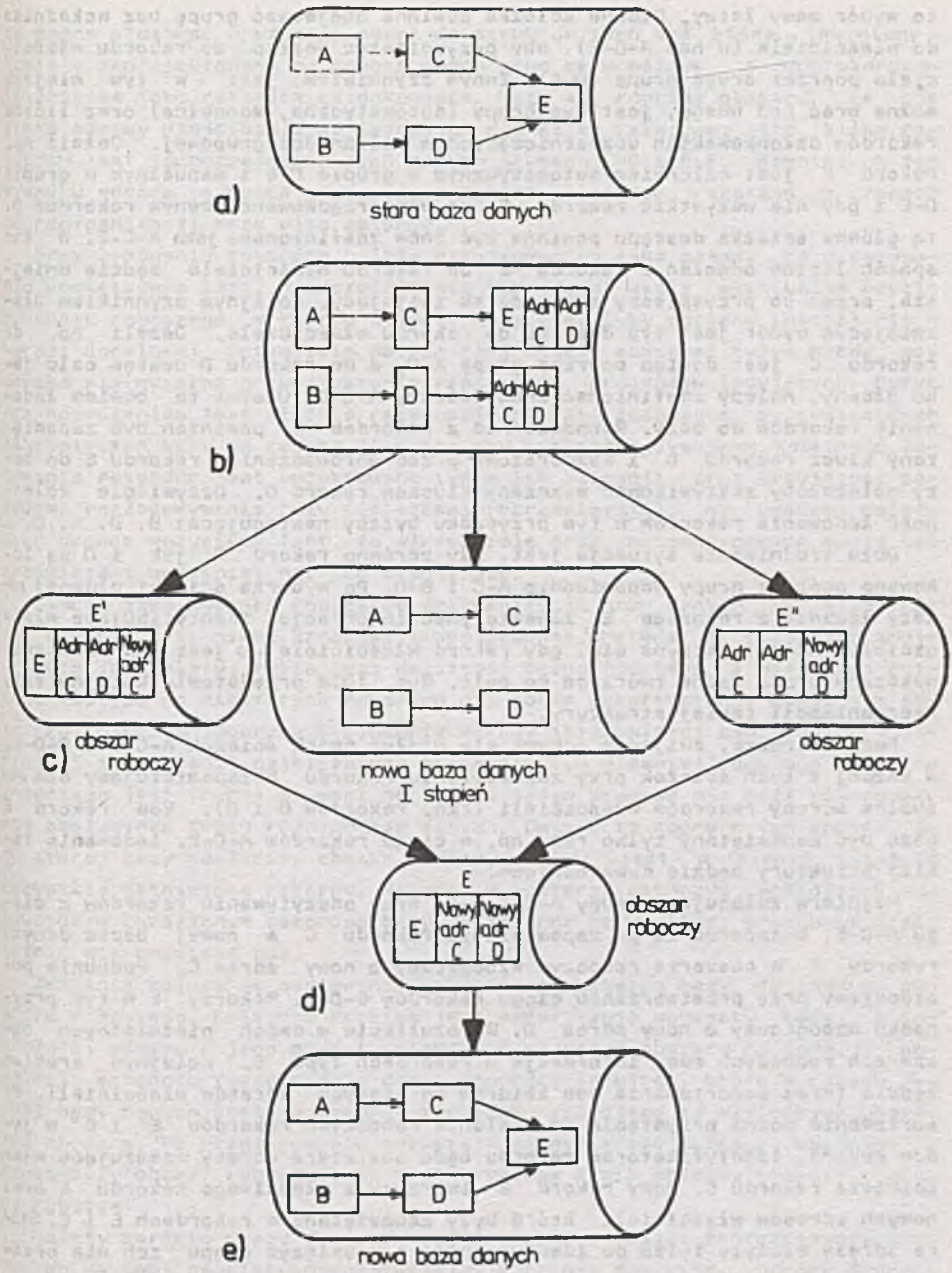
do jednego rekordu można dojść na wiele różnych sposobów. Na potrzeby reorganizacji musimy wtedy zdefiniować główne ścieżki przetwarzania wzdłuż których będziemy zwijać rekordy. Wybór głównej ścieżki przetwarzania zależy przede wszystkim od miejsca rekordu w strukturze. Spójrzmy na przykładowe struktury rekordów przedstawione na rys. 10.3.



Rys. 10.3. Przykłady reorganizowanych struktur

Naturalny sposób rozładowania czy ładowania struktury (a) jest wzdłuż ścieżki A-B-C. Przy rekordach C powinno się zapamiętywać informacje o właścicielu grupy D-C (a więc o konkretnym rekordzie D). Rekordy D w tej strukturze byłyby zapamiętywane i ładowane oddzielnie, z tym że musiałyby być wprowadzone do bazy wcześniej niż rekordy C. Jeżeli grupa D-C jest manualna, a ponadto pola rekordu C wskazują, czy i do której grupy D-C przypisany jest dany rekord C, to w czasie rozładowania rekordu C można pominąć informacje o jego właścicielu D. W przeciwnym razie łącznie z rekordem C musimy zapamiętać klucz, adres lub dowolne inne pole identyfikujące rekord D. W tym celu należy przeczytać właściwy rekord D. Jest to łatwe, gdy w części systemowej rekordu C istnieje wskaźnik do właściciela w grupie D-C. Gdy ten wskaźnik w grupie nie jest zdefiniowany, znalezienie właściciela będzie się wiązało z koniecznością odczytu wszystkich członków grupy D-C począwszy od wyjściowego rekordu C. Pochłonie to znaczną ilość czasu. W tym miejscu widać, jak potrzeby przyszłej reorganizacji mogą wpływać na projektowanie zupełnie nowej bazy danych.

Struktura (b) z rys. 10.3 jest nieco bardziej skomplikowana. Już na pierwszy rzut oka widać, że główną ścieżką reorganizacji może być ciąg A-C-E lub B-D-E. Z uwagi na podobne umiejscowienie rekordu E w obu ścieżkach o wyborze właściwej z nich decydują inne czynniki. Pierwszym z nich jest rodzaj wskaźników w grupach C-E i D-E. Jeżeli w którejś z tych grup nie jest zdefiniowany wskaźnik do właściciela (grupa C-E), a w drugiej jest,



Rys. 10.4. Reorganizacja z dwustopniowym ładowaniem
a) struktura początkowa, b) obszar roboczy, c) I faza ładowania, d) sortowanie i scalanie, e) ładowanie rekordów E

to wybór mamy łatwy. Główna ścieżka powinna obejmować grupę bez wskaźnika do właściciela (u nas A-C-E), aby przyspieszyć dostęp do rekordu właściciela poprzez drugą grupę (D-E). Innym czynnikiem, jaki w tym miejscu można brać pod uwagę, jest typ grupy (automatyczna, manualna) oraz liczba rekordów członkowskich uczestniczących w zależności grupowej. Jeżeli np. rekord E jest członkiem automatycznym w grupie C-E i manualnym w grupie D-E i gdy nie wszystkie rekordy E są podporządkowane pewnym rekordom D, to główna ścieżka dostępu powinna być znów zdefiniowana jako A-C-E. W ten sposób liczba odwołań z rekordu E do rekordu właściciela będzie mniejsza, przez co przyspieszy się program zwijający. Kolejnym czynnikiem ułatwiającym wybór jest typ dostępu do rekordu właściciela. Jeżeli np. do rekordu C jest dostęp poprzez grupę A-C, a do rekordu D dostęp calc jako główny, należy zdefiniować znów ścieżkę A-C-E. Ułatwi to bowiem ładowanie rekordów do bazy. Równocześnie z rekordem E powinien być zapamiętany klucz rekordu D i każdorazowo przed wprowadzeniem rekordu E do bazy należałoby zaktywizować wskazany kluczem rekord D. Oczywiście kolejność ładowania rekordów w tym przypadku byłaby następująca: B, D, A, C, E.

Dużo trudniejsza sytuacja jest, gdy zarówno rekord C jak i D są lokowane poprzez grupy odpowiednio A-C i B-D. Po wyborze ścieżki głównej należy łącznie z rekordem E zapamiętywać informacje identyfikujące właścicieli. Kłopot zaczyna się, gdy rekord właściciela nie jest jednoznacznie wskazany przez żadne tworzące go pole. Rys. 10.4 przedstawia kolejne fazy reorganizacji takiej struktury.

Faza pierwsza, zwijanie odbywa się wzdłuż dwóch ścieżek A-C-E i B-D-E. W każdej z tych ścieżek przy zapamiętaniu rekordu E zapamiętujemy oba aktualne adresy rekordów właścicieli (tzn. rekordów C i D). Sam rekord E może być zapamiętany tylko raz, np. w ciągu rekordów A-C-E. Ładowanie takiej struktury będzie dwustopniowe.

Najpierw załadujemy grupy A-C i B-D. Przy odczytywaniu rekordów z ciągu A-C-E, bezpośrednio po zapamiętaniu rekordu C w nowej bazie danych rekordy E w obszarze roboczym wzbogacamy o nowy adres C. Podobnie postępujemy przy przetwarzaniu ciągu rekordów B-D-E. Rekordy E w tym przypadku wzbogacamy o nowy adres D. W rezultacie w dwóch niezależnych obszarach roboczych mamy informacje o rekordach typu E. Kolejnym krokiem będzie teraz posortowanie obu zbiorów wg starych adresów właścicieli. Po sortowaniu można przystąpić do scalania roboczych rekordów E' i E" w jeden rekord. Identyfikatorem rekordu będą oba stare adresy wskazujące właścicieli rekordu E. Nowy rekord E tworzymy z właściwego rekordu E oraz nowych adresów właścicieli, które były zapamiętane w rekordach E'i E". Stare adresy służyły tylko do identyfikacji i w dalszym ciągu ich nie przechowujemy.

Mając tak przygotowany zbiór rekordów E można przystąpić do ładowania rekordów E do bazy danych. Bezpośrednio przed wstawieniem rekordu E do bazy należy zaktywizować wskazane adresami w E rekordy C i D. Metodę

tę można stosować w szczególności do struktur typu W:W, które - jak wiemy - zostały zaprojektowane za pomocą dwóch grup ze wspólnym pseudorekordem-łącznikiem jako rekordem członkowskim. Może się również okazać, że oba stare adresy właścicieli są jednakowe dla kilku rekordów, tzn. kilka rekordów jest jednocześnie w tych samych grupach C-E i D-E. Również w tym wypadku metodę tę można stosować, z tym że kolejność rekordów w grupach po reorganizacji może ulec zmianie.

Przy ładowaniu rekordów należy przyjmować te same zasady co w ładowaniu początkowym bazy. W szczególności mamy tu na myśli ewentualne użycie schematu roboczego, w którym obiekty bazy zostałyby opisane inaczej niż w wersji docelowej. Oczywiście metodę taką należy stosować tylko wtedy, gdy uzyska się wyraźne przyspieszenie realizacji programów ładujących. Dużym przyspieszeniem jest właśnie rezygnacja na czas ładowania z systemowych zabezpieczeń bazy na taśmie śladowej czy w zbiorze dyskowym. Kolejność ładowania rekordów jest podyktowana typem ich alokacji oraz przyjętymi zasadami rozładowywania bazy (ścieżkami reorganizacji). Na uwadze należy mieć przede wszystkim fakt, że właściciele grup automatycznych muszą być zapamiętani wcześniej niż ich członkowie.

Użycie jako obszaru roboczego reorganizacji bazy danych (a właściwie jej części) jest nieco bardziej skomplikowane. Metoda ta może mieć swoje niewątpliwe zalety, takie jak: możliwość bezpośredniego a nie sekwencyjnego dostępu do niektórych rekordów czy dobre wykorzystanie pamięci w obszarze roboczym (poprzez stosowanie metody interwałowej pamiętania właścicieli i strategii najbliższego miejsca). Tym niemniej obsługa obszaru roboczego jest trudna z uwagi na to, że jeden program nie może równocześnie obsługiwać dwóch różnych baz danych. Ominąć to można w ten sposób, że do starej bazy dokładamy obszar roboczy, który jest w stanie zmieścić wszystkie istniejące rekordy. Rekordy w obszarze roboczym, aczkolwiek odpowiadają określonym rekordom bazy, mają inną nazwę (kod) oraz mogą nieco różnić się rozpisem pól.

Zwijanie polega na kopiowaniu rekordów z obszarów bazy do jednego obszaru roboczego. Kolejnym krokiem jest modyfikacja schematu bazy, a dokładniej mówiąc - jego części opisującej normalne obszary na dane. Po kompilacji schematu rozpoczynają pracę programy ładujące, które w ramach jednej bazy danych kopiują rekordy z obszaru roboczego do właściwych obszarów na dane. Po przeładowaniu pozostaje zmodyfikować schemat usuwając z niego cały obszar roboczy oraz stowarzyszone z nim obiekty (rekordy, grupy robocze).

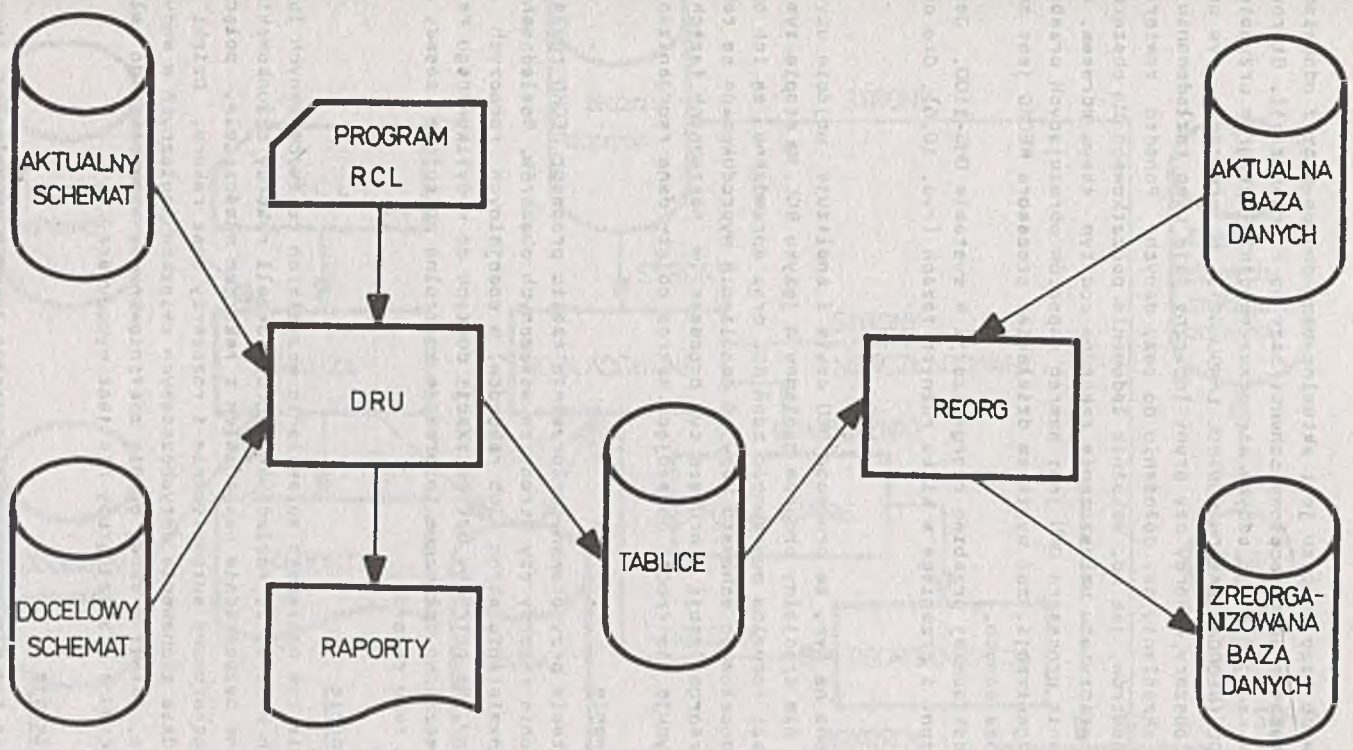
Należy zwrócić uwagę jeszcze na jedną rzecz. Każda reorganizacja bazy danych ma swój osobisty niepowtarzalny z reguły charakter. Metody dobre w jednej z nich mogą okazać się niewygodne w drugiej. Do każdego nowego problemu reorganizacji należy podejść oddzielnie. Bardzo dobre efekty może dać całkowicie niestandardowe spojrzenie na problem. Weźmy taki przykład. W pracującej bazie danych należy dołożyć jedną grupę manualną. W rekor-

dzie członkowskim nie ma dostatecznej liczby zarezerwowanych na wskaźniki manualne słów. Oczywiście problem ten można rozwiązać tradycyjnie poprzez rozładowanie i ładowanie zainteresowanych rekordów. Można podejść do tego zagadnienia zupełnie inaczej. Wystarczy napisać program w języku wewnętrznym, który będzie przetwarzał obszary bazy poza systemem zarządzania bazą. Praca programu polegałaby na przeglądaniu kolejnych stron obszaru, znajdywaniu rekordów żądanego typu i poszerzeniu ich o nowe słowa. Poszerzenie pociągałoby za sobą rozsunięcie innych danych na stronie oraz konieczność aktualizacji słów w nagłówku strony (ilość wolnego miejsca). Jeżeli strona wypełniona jest całkowicie, rekord, którego nie można rozszerzyć, należy gdzieś zabezpieczyć oraz usunąć go z bazy danych. Po przetworzeniu w ten sposób całego obszaru należy zmienić schemat bazy tak, aby opis bazy zgadzał się ze zmienionym już stanem fizycznym rekordu. Kolejną czynnością jest podpięcie rozszerzonych rekordów do właściwych wystąpień grup, co może być zrealizowane przez prosty program oraz wprowadzenie uprzednio zabezpieczonych rekordów do bazy. W przypadku małej ich liczby zabezpieczenie może polegać jedynie na wydrukowaniu zawartości rekordu, co umożliwi ponowne zapamiętanie tych rekordów przez programy użytkowe lub systemowe. Metoda przedstawiona powyżej okazała się w konkretnym zastosowaniu kilkakrotnie szybsza od pozostałych.

10.4. Reorganizacja przy użyciu oprogramowania standardowego na przykładzie DMS-1100

Jak zauważyliśmy powyżej, reorganizacja jest problemem trudnym i czasochłonnym. Wychodząc naprzeciw użytkownikom baz danych producent sprzętu i oprogramowania standardowego z reguły stara się dostarczyć gotowe oprogramowanie realizujące reorganizację w pełnym lub chociaż częściowym zakresie. Przykładem może być firma SPERRY-UNIVAC, która od kilku lat rozbudowuje system zarządzania bazy danych DMS-1100 o oprogramowanie służące do reorganizacji. Rozwój tego oprogramowania odbywał się stopniowo. Najpierw realizowana była jedynie reorganizacja fizyczna, a więc zmiana umiejscowienia rekordów w bazie bez zmiany ich zawartości. Następnie rozbudowano oprogramowanie o częściową reorganizację logiczną obejmującą między innymi dodanie lub usuwanie obszarów, zmianę atrybutów obszarów i rekordów. Prace nad kompleksowym rozwiązaniem problemu reorganizacji jeszcze trwają.

W systemie DMS-1100 reorganizację wykonują dwa niezależne procesory. Pierwszy z nich, DRU (DATA REORGANIZATION UTILITY), służy do analizy zakresu reorganizacji i przygotowania tablic kontrolnych do reorganizacji. Drugi procesor o nazwie REORG wykonuje fizyczną reorganizację bazy na podstawie przygotowanych przez DRU tablic. Najbardziej ogólny schemat reorganizacji przedstawia rys. 10.5.



Rys. 10.5. Schemat reorganizacji bazy w systemie DMS-1100

Wejściem do tego procesu jest aktualna baza danych wraz z odpowiadającym jej schematem oraz docelowy schemat bazy po reorganizacji. Sterowanie przebiegiem reorganizacji odbywa się przez specjalny program przygotowany w języku RCL (REORGANIZATION CONTROL LANGUAGE). W programie tym podaje się, które obszary, rekordy czy grupy logiczne mają ulec rozładowaniu, modyfikacji, skreśleniu czy dołożeniu do bazy danych. Ponadto zawiera on szereg parametrów, jak np. wskaźnik ładowania początkowego dla obszaru czy zasad, np. priorytetu umieszczenia rekordów pod tym samym adresem. Wynikiem działania procesora DRU jest szereg raportów obrazujących przebieg i zakres reorganizacji, zaś wynikiem działania procesora REORG jest zmodyfikowana baza danych.

Omówmy dokładniej przebieg reorganizacji w systemie DMS-1100. Jest to proces złożony i przebiega w kilku różnych fazach (rys. 10.6). Oto one:

1. Analiza

Polega ona na tym, że procesor DRU czyta i analizuje wejście użytkownika. Jest nim specjalny program napisany w języku RCL. Na etapie tym analizowana jest formalna poprawność zdań RCL oraz sprawdzane są ich odniesienia do docelowego schematu bazy. W rezultacie wyprodukowane są tablice kontrolne reorganizacji sterujące tym procesem w następnych fazach. Ponadto otrzymuje się raport obrazujący zakres objęty daną reorganizacją.

2. Rozładowanie

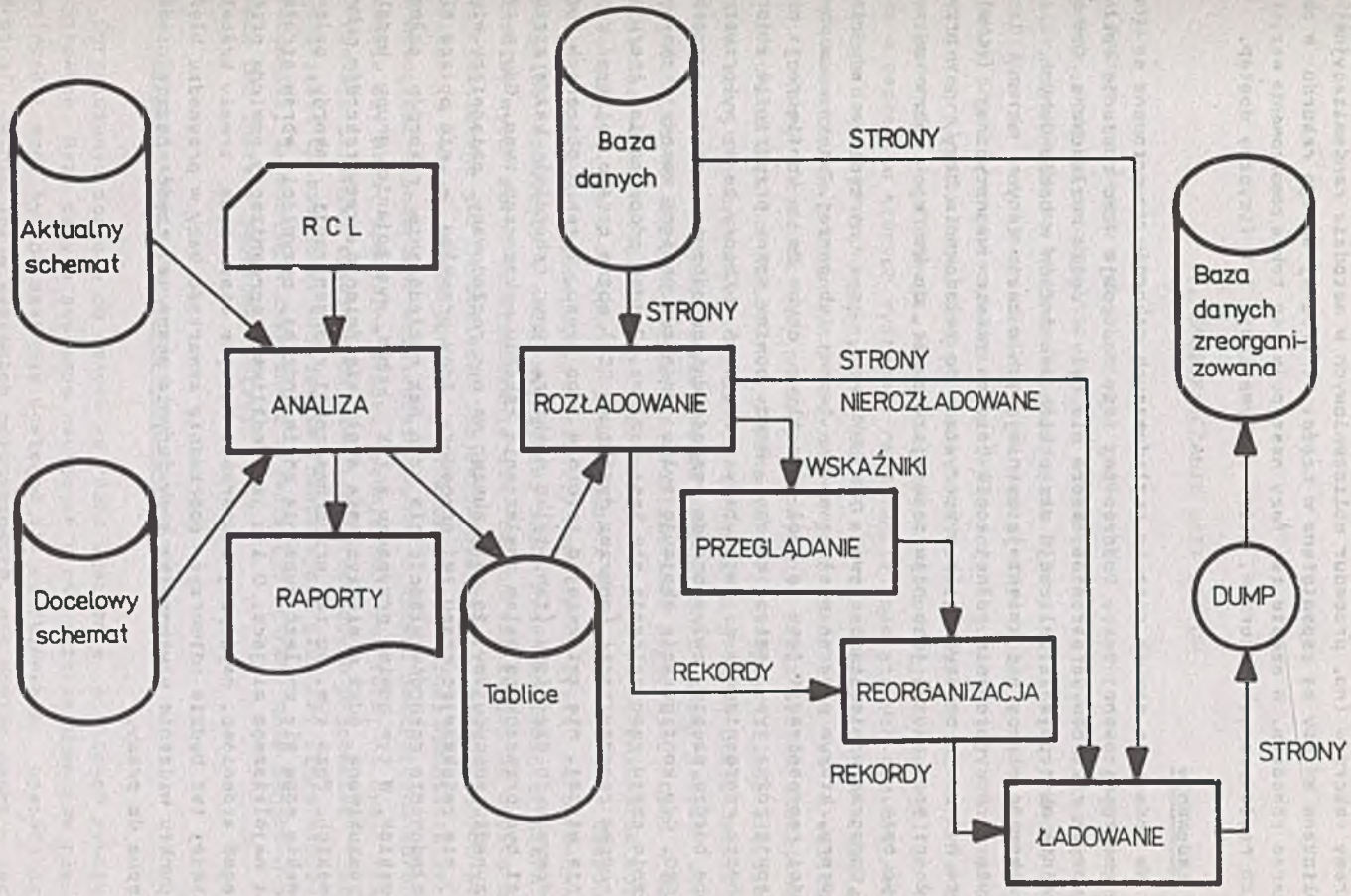
Na podstawie przygotowanych uprzednio tablic procesor REORG rozładowuje odpowiednie rekordy czy strony ze wskazanych obszarów. Rozładowanie polega na zapamiętaniu stron (lub rekordów) w specjalnych roboczych zbiorach systemu. Na potrzeby optymalizacji dostępu do modyfikowanego rekordu zostaną tymczasowo dołączone informacje kontrolne opisujące zasady zapamiętywania tego rekordu.

3. Przeglądanie

Na etapie tym następuje sprawdzenie wszystkich rozładowywanych lub zawierających rekordy do rozładowania grup. Jeżeli rekordy członkowskie nie są połączone bezpośrednim wskaźnikiem z rekordem właściciela, połączenie to będzie generowane automatycznie i rozszerzy nasz rekord. Dzięki niemu możliwe będzie zachowanie dotychczasowych związków logicznych w modyfikowanej bazie. Jeżeli w danej grupie zdefiniowane są wskaźniki do właściciela, procedura dla tej grupy nie jest wykonywana.

4. Reorganizacja

Na etapie tym dla każdego rozładowanego rekordu symulowane jest jego zapamiętanie. Wykonywane jest ono zgodnie z opisem tego rekordu w nowym schemacie bazy. Symulowanie polega na wywołaniu procedur obliczających



Rys. 10.6. Przebieg reorganizacji i przepływ informacji w tym procesie

adresy rekordów (np. procedur obliczeniowych w metodzie randomizacyjnej). Obliczone adresy są zapamiętane w części kontrolnej samego rekordu w obszarze roboczym. W czasie tej fazy następuje też takie posortowanie wszystkich rozładowanych rekordów, które zoptymalizuje ich fizyczny dostęp.

5. Ładowanie

Na podstawie nowych adresów rozładowanych rekordów przygotowane są strony zreorganizowanej bazy. Podczas tej fazy następuje więc scalenie rozładowanych rekordów ze stronami, które nie były w ogóle rozładowane. Jednocześnie następuje aktualizacja wszystkich wskaźników w bazie danych. Zmodyfikowane strony nie zmieniają istniejących w bazie danych stron, lecz służą do przygotowania pełnej kopii bazy na taśmie magnetycznej (DUMP). Może być ona w prosty sposób wykorzystana do załadowania bazy, a w przyszłości może służyć jako najwcześniejszy punkt, do którego można odtworzyć bazę.

Oprogramowanie standardowe reorganizacji jest narzędziem administratora, którym może on posługiwać się lepiej lub gorzej. Zwiększenie prędkości reorganizacji jest na ogół niemożliwe, chyba że w konfiguracji występują różne typy pamięci masowej. Wtedy bowiem można przydzielić zbiory robocze reorganizacji do najszybszych urządzeń. Zbiory te są wykorzystywane bardzo często, zwłaszcza do bezpośredniego odczytu przez procesor REORG. Gdy konfiguracja obejmuje tylko jeden typ pamięci masowej, zmniejszenie czasu reorganizacji nie jest możliwe. Jednak odpowiednio sterując procesem reorganizacji (poprzez dyrektywy RCL) można często osiągnąć znacznie więcej, nie zwiększając istotnie tego czasu. I tak w obszarach rozładowywanych częściowo (tzn. tylko niektóre typy rekordów) każda strona musi być przeszukana celem znalezienia rekordu wskazanego typu. Gdy w tym przypadku zdecydujemy, że cały obszar ma być rozładowany, osiągniemy więcej nie zwiększając czasu tej procedury. Inny przykład - nie opłaca się rozładowanie rekordów właściciela grup bez rozładowania rekordów członkowskich. W tym bowiem przypadku i tak każde wystąpienie grupy będzie przeszukiwane, gdyż w międzyczasie mogły się zmienić adresy rekordów członkowskich. Poza tym, po tak przeprowadzonej reorganizacji rekordy członkowskie mogą się znaleźć zupełnie gdzie indziej, oczywiście wbrew strategii najbliższego miejsca. O ile jest możliwe, reorganizacja powinna przebiegać stopniowo, najlepiej obszarami. Kolejne etapy będą trwały krócej, łatwiej też będzie odtworzyć poprzednią zawartość bazy w przypadku błędu. Z punktu widzenia użytkownika bazy jedynie pewna jej część będzie niedostępna do pracy.

11. ZABEZPIECZANIE BAZY DANYCH

Ważnym problemem w eksploatacji każdej bazy danych jest jej zabezpieczenie. Jest to szczególnie ważne w systemach pracujących wieloprogramowo i gdy użytkownik ma bezpośredni dostęp do bazy z terminali. Mówiąc "zabezpieczenie bazy" mamy właściwie na uwadze dwie różne sprawy - zabezpieczenie przed niepowołanym dostępem (zwłaszcza aktualizacją) oraz zachowanie integralności bazy. Integralność bazy oznacza poprawność danych w każdym momencie eksploatacji. Zachowanie integralności jest szczególnie trudne w bazach, w których występuje redundancja pamiętanych danych. Aktualizacja danych przechowywanych w różnych miejscach bazy powinna nastąpić jednocześnie, co z przyczyn praktycznych jest niemożliwe do zrealizowania. Podobny przypadek mamy podczas wykonywania przez system skomplikowanych komend DML, np. DELETE. Na wykonanie tej komendy składa się szereg aktualizacji wykonywanych na różnych rekordach i grupach bazy. Jeżeli nastąpi przerwanie w trakcie wykonywania komendy DML, baza danych utraci spójność, część łańcuchów będzie zaktualizowana, część nie. Dane w bazie przestaną odzwierciedlać stan rzeczywisty. Zarówno do ograniczenia dostępu do danych, jak i do ochrony integralności bazy administrator dysponuje szerokim zestawem standardowym narzędzi. Mamy tu na myśli oprogramowanie specjalne, wchodzące w skład większości SZBD. Zadaniem administratora jest właściwy wybór metod zabezpieczenia dla różnych aplikacji, co często musi być realizowane już na poziomie projektowania bazy. W uzasadnionych przypadkach (w bazach, gdzie ochrona jest szczególnie ważna) administrator może wzbogacać standardowe możliwości ochrony o własne specyficzne oprogramowanie czy nawet specjalne środki techniczne. Mamy tutaj na myśli różnego rodzaju czytniki specjalnych kart identyfikujących pracownika lub w późniejszym czasie nawet urządzenia rozpoznające głos czy linie papilarne pracownika. Zagadnienia związane z organizacją ochrony bazy danych w systemie DMS zostaną omówione poniżej.

11.1. Zabezpieczanie bazy przed niepowołanym dostępem

Ograniczenie dostępu do danych ma różne znaczenie w różnych systemach użytkowych. Bazy danych systemów naukowych o charakterze słowników czy encyklopedii mogą być powszechnie dostępne dla użytkowników posiadających terminal. Jednak już w systemach przemysłowych, nie mówiąc nawet o systemach z dziedzin wojskowości czy kontroli ludności, wymagane jest ograni-

czenie powszechnego dostępu. Co więcej, chodzi w nich o określenie zakresu dostępności dla poszczególnych grup osób (pracowników).

Pierwszym problemem projektanta nowego systemu i jednocześnie administratora bazy jest więc sklasyfikowanie wszystkich potencjalnych użytkowników. Dla każdej grupy należy określić dokładnie zakres dostępności do danych. Począwszy od grupy osób o nieograniczonym dostępie, poprzez osoby o dostępie częściowym (np. wybrane rekordy, grupy, obszary czy nawet tylko pola w rekordach) dojdziemy w ten sposób do osób, które w ogóle nie powinny kontaktować się z bazą danych. Zadaniem SZBD lub niezależnego od niego oprogramowania zabezpieczającego powinna być kontrola autentyczności użytkownika i sterowanie dostępem do danych w bazie.

Najprostszymi formami zabezpieczania przed niepowołanym dostępem jest rejonizacja terminali. Rozumiemy przez nią takie ograniczenie, że z danego terminala można realizować tylko niektóre transakcje. Tablice dopuszczalnych kodów transakcji na poszczególnych terminalach w takim przypadku stanowią integralną część systemu komunikacyjnego i są przez niego obsługiwane. Metody takiej nie można zastosować w tym przypadku, gdy użytkownik ma możliwość bezpośredniej pracy z bazą (bez udziału transakcji i programów je obsługujących) za pomocą jakiegoś języka konwersacyjnego. Pierwszym problemem, jaki się nasuwa, jest w tym przypadku identyfikacja użytkownika. Oczywiście system musi być wyposażony w tablice opisujące dopuszczalnych użytkowników. Tablice te mogą zawierać nazwiska, kody czy inne atrybuty identyfikujące pracownika. Identyfikacja pracownika może mieć charakter czysto techniczny, jak np. przeczytanie specjalnej karty identyfikującej. Często jednak systemy zabezpieczajace sprawdzają autentyczność zgłoszonej osoby. Może to być realizowane za pomocą podania hasła czy udzielenia systemowi odpowiedzi na zadane pytanie.

Bardzo często mamy do czynienia z hasłem algorytmicznym używającym aktualne w chwili pracy parametry, np. datę lub czas. System sprawdza zgodność podanego hasła wykonując analogiczne co użytkownik operacje arytmetyczno-logiczne.

Po formalnym zaakceptowaniu użytkownika przez system może rozpocząć on pracę z bazą danych. Jednak i teraz system zabezpieczeń musi sterować dostępem do poszczególnych części bazy. Może to być realizowane bądź na poziomie komend, do których użytkownik jest uprawniony lub nie, bądź na poziomie danych w bazie. Zauważmy, że aby sprawdzić dopuszczalność danej operacji, czasem system musi już wykonać dostęp do niezbędnych informacji w bazie opisujących użytkownika. I tak np. przełożony może być uprawniony do odczytu kartoteki pracownika, a pracownik nie może mieć wglądu w dokumenty kierownika. Oznacza to, że system musi sprawdzić relację zachodzącą między osobą żądającą informacji a podmiotem operacji. Sterowanie dopuszczalnością poszczególnych komend może być realizowane również poprzez hasło oraz tablicę dopuszczalnych dla danego użytkownika operacji. Podobnie jest ze sterowaniem dostępem do poszczególnych obiektów bazy. Obiekty te

powinny mieć zaznaczone w schemacie logicznym, że dostęp do nich odbywa się przez hasło oraz zawierać wzór czy algorytm tego hasła. Przed wykonaniem operacji na tym obiekcie wystarczy przesłać w programie treść hasła do odpowiednio zadeklarowanej zmiennej programu. W systemie DMS-1100 jest to realizowane poprzez klauzule ACCESS CONTROL LOCK i ACCESS CONTROL KEY.

Zdanie: "ACCESS CONTROL LOCK FOR funkcja IS ....." może pojawić się w opisie obszaru, rekordu czy grupy w schemacie danych. Precyzuje ono. o jaką operację na obiekcie chodzi (komenda DML lub w przypadku otwierania obszaru charakter tego otwarcia). Podaje również samo hasło lub nazwę procedury, która to hasło przygotowuje. Związane z tym zdaniem jest zdanie następne: "ACCESS CONTROL KEY IS nazwa", które definiuje stałą nazwę pola sterującego dostępem do tego obiektu. Jeżeli program chce wykonać pewna operację na obiekcie, np. usunąć rekord, musi mieć zdefiniowane w swoim obszarze roboczym pole o nazwie, jak zadeklarowano to w zdaniu "ACCESS CONTROL KEY", a przed samą operacją przesłać do tego pola poprawną wartość klucza (hasła). Dopiero wtedy system umożliwi wykonanie tej operacji.

Oprócz sprawdzenia dopuszczalności pewnych operacji na bazie przez potencjalnych jej użytkowników system zapewnia również automatyczne blokowanie informacji między pracującymi programami (użytkownikami). Blokowanie może następować na poziomie obszaru lub rekordu. Sterowanie blokowaniem obszaru odbywa się poprzez odpowiednie formaty komendy OPEN. W systemie DMS-1100 wyróżnia się 6 różnych formatów tego czasownika, o różnym priorytacie między sobą. Są to otwarcia: RETRIEVAL (odczyt), UPDATE (aktualizacja), PROTECTED RETRIEVAL i UPDATE (odczyt lub aktualizacja uprzywilejowana) oraz EXCLUSIVE RETRIEVAL i UPDATE (odczyt lub aktualizacja wyłączna). Blokowanie rekordów, a ściślej mówiąc - stron, na których znajdują się te rekordy, ma znaczenie dla zabezpieczenia integralności bazy danych. Z tego powodu system blokuje strony zmieniane przez jakiś program, aż do momentu zakończenia się tego programu. Inne programy, które chcą również aktualizować zmienione strony, są kolejkowane. To zresztą jest często przyczyną wzajemnego zablokowania się dwóch programów (ang. deadlock), co system musi dostrzec i rozwiązać. Do rozwiązania wzajemnego zablokowania służy "procedura cofania" zastosowana do jednego z tych programów. Wybór programu, który trzeba cofnąć, leży w gestii systemu - może to być program o mniejszym priorytecie, program, który do tej pory wprowadził mniej poprawek do bazy lub po prostu program powodujący pętlę. Podobne znaczenie ma blokowanie przed aktualizacją rekordów bieżących w jakimś programie. Chodzi o to, aby rekord pobrany przez jakiś program i przebywający najprawdopodobniej w jego buforach roboczych nie uległ zmianie.

System DMS-1100 daje jeszcze jedną możliwość ochrony danych i to na poziomie nawet elementarnego pola. Realizowane jest to poprzez podschemat dostępny dla programu (programisty), a zawierający jedynie część informacji pamiętanych rzeczywiście w bazie. W szczególności z punktu widzenia programu (poprzez podschemat właśnie) rekord może zawierać tylko niektóre

pola, gdy w istocie zawiera ich o wiele więcej. Podobnie jest z grupami logicznymi, które reprezentują relacje widoczne lub niewidoczne dla programu. Należy zaznaczyć, że w tym przypadku SZBD pobiera z bazy pełną informację do swoich buforów (np. rekord fizyczny) i wycina z nich obiekt zdefiniowany w podschemacie (w tym przykładzie rekord logiczny w podschemacie). W ten sposób do części danych program nie może wykonać dostępu, gdyż nie wie nawet o ich istnieniu.

Wszystkie techniki przedstawione w tym rozdziale służą jednemu celowi - zapobieganiu przed niewłaściwym dostępem. Klucz czy hasło mogą być jednak poznane przez niepowołaną osobę. Pozostaje w takim przypadku jeszcze jedna możliwość - szyfrowanie informacji pamiętanych w bazie. Istnieje wiele sposobów szyfrowania danych. Najprostszymi są: zmiana kolejności znaków czy zastąpienie jednych znaków innymi. Najpewniejsza jest metoda addytywna, polegająca na tworzeniu kodu wyjściowego z kodu wejściowego i specjalnego klucza ochrony przez operacje arytmetyczne (np. dodawanie). Tylko znajomość klucza ochrony pozwoli na szyfrowanie i rozszyfrowanie tekstu stanowiącego informację w bazie.

11.2. Zabezpieczanie integralności bazy danych

Problem integralności bazy jest bardzo ważny w każdym typie bazy danych. Szczególnie ostro występuje on jednak w bazach danych eksploatowanych w czasie rzeczywistym. Z uwagi na ciągłą eksploatację rodzi się potrzeba pełnej dyspozycyjności bazy w każdym momencie czasu. To z kolei pociąga za sobą istnienie procedur kontrolujących stan bazy oraz procedur odtwarzających bazę. Procedury te są częścią składową systemu operacyjnego lub SZBD, choć niekiedy mogą stanowić oprogramowanie własne użytkownika. Oprócz procedur systemy zabezpieczania bazy wymagają innych "fizycznych" narzędzi. Narzędziami tymi są "dumpy" i taśmy śladowe systemu.

11.2.1. Systemowe narzędzia ochrony bazy

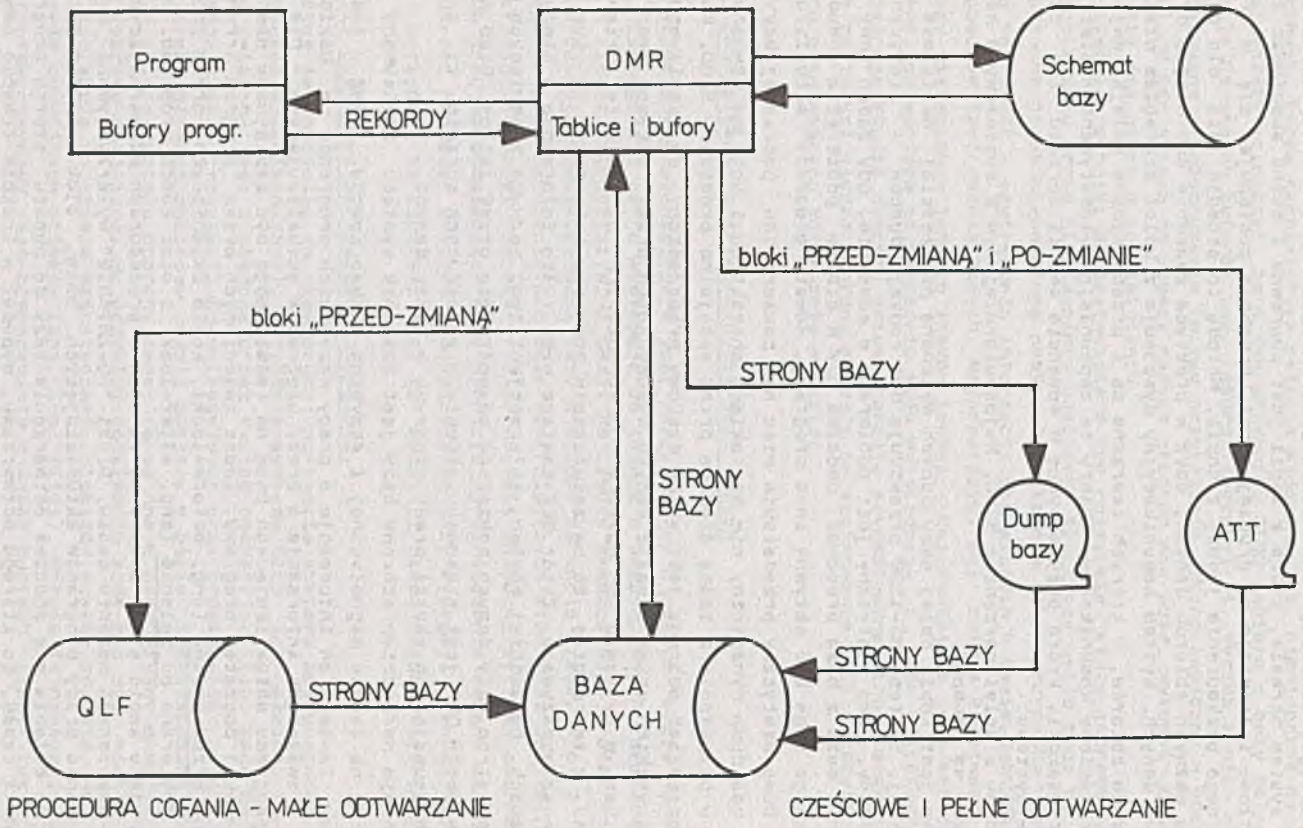
Można je podzielić na trzy grupy: narzędzia systemu operacyjnego (wraz z systemem komunikacyjnym), narzędzia systemu SZBD lub własne. Pierwsza grupa związana jest z obsługą przez system operacyjny (lub jego wyspecyfikowaną część) transakcji komunikacyjnych oraz prowadzenie przez system biblioteki głównej zbiorów. Jeżeli obszary w bazie zadeklarowane są jako zwykłe zbiory systemu operacyjnego, to są one w standardowy sposób dla danego systemu chronione. Na przykład codziennie może być dla nich wykonywana ich kopia na taśmie magnetycznej. Niekiedy system pozwala na definiowanie podwójnych zbiorów (ang. dual files). Metoda ta oznacza, że dany zbiór przechowywany jest równocześnie w dwóch kopiach (ang. two legs). Każda aktualizacja jest wykonywana jednocześnie na obu kopiach. System operacyjny (lub jego część jak w przypadku UNIVAC-1106 procesor TIP) za-

pewnia ciągłą kontrolę równoważności obu kopii. W razie wykrycia niezgodności system określa, która z kopii jest poprawna i można skopiować z niej prawidłowe informacje do drugiej kopii. Podobnie postępuje się w razie fizycznego uszkodzenia jednej z kopii. Metodę tę stosuje się dla szczególnie ważnych zbiorów danych, gdyż w praktyce zapewnia ona ciągłą dyspozyjność danych. System komunikacyjny dysponuje z kolei szeregiem własnych roboczych zbiorów, w których tworzone są i przechowywane kolejki wejściowe i wyjściowe komunikatów. Zbiory te zapewniają automatyczne wystartowanie transakcji, które były aktywne w momencie awarii sprzętu lub zawieszenia systemu.

Omówmy z kolei narzędzia SZBD. Najpopularniejszym z systemowych narzędzi bazy są "dumpy".

Dump jest kopią całej bazy danych wykonaną najczęściej na taśmie magnetycznej. System DMS-1100 przewiduje dwa rodzaje dumpów - statyczny i dynamiczny. Dump statyczny jest pobierany w momencie, gdy żaden inny program nie może z bazą pracować, podczas gdy w czasie pobierania dumpu dynamicznego mogą być aktywne inne programy, nawet programy aktualizujące bazę. Dump statyczny przedstawia więc nam pewien stan poprawnej bazy danych, zaś dump dynamiczny nie ma takiego charakteru i może być używany jedynie w połączeniu z taśmą śladową przez specjalne procedury SZBD. Kolejnym narzędziem ochrony jest zbiór dyskowy przeznaczony do zapisywania w nim wszystkich zmian w bazie danych wykonywanych przez aktywne programy. Zbiór ten, w systemie DMS nazwany QUICK-LOOK-FILE, ma charakter dziennika systemu, z tą różnicą, że po zakończeniu programu informacje o tym programie są wymazywane. Zbiór QLF zawiera więc tylko informacje o bieżących programach. Dokładniej mówiąc, informacjami tymi są kopie zmienianych przez program stron bazy danych pobierane bezpośrednio przed zmianą. Stąd wynika, że zbiór QLF jest stosowany głównie w procedurach cofania, tj. procedurach usuwających skutki pracy programu w bazie danych.

Typowym narzędziem ochrony bazy jest dziennik systemu zapisywany najczęściej na taśmie magnetycznej i nazywany z tego powodu taśmą śladową systemu. Zawiera on informacje o pracy wszystkich programów w bazie danych, a zwłaszcza informacje o pracy programów aktualizujących. Dla każdego programu aktualizującego bazę na taśmie mogą być zapisywane następujące bloki: początek programu, kopie zmienionych przez program stron pobierane przed zmianą (ang. before-look), kopie zmienianych przez program stron pobierane po zmianie (ang. after-look) i blok końca programu. Bloki zawierające kopie stron z bazy są używane w procedurach odtwarzających bazę do wskazanego momentu czasu. Bloki "PO-ZMIANIE" wykorzystywane są do symulowania pracy programów aktualizujących bazę, bloki "PRZED-ZMIANĄ" służą do cofania się podczas odtwarzania bazy do punktu startu programu, gdy punkt czasu, do którego odtwarzamy, wypadał w trakcie trwania pewnych programów. Taśma śladowa zawiera jeszcze informacje typu kontrolnego. Są to punkty czasowe zawierające informacje o dacie i czasie, służące do



Rys. 11.1. Ochrona danych w systemie DMS-1100

wskazywania zakresu odtwarzania. Punkty kontrolne (ang. checkpoints) mogą mieć charakter dynamiczny lub statyczny w zależności od tego, czy w czasie ich zapisywanie inne programy mogą być aktywne w bazie.

W systemach bazy współpracujących z systemami komunikacyjnymi taśma śladowa może jeszcze zawierać informacje o transakcjach wejściowych. Mogą one służyć do badania stopnia obciążenia systemu komunikacyjnego lub służyć do automatycznego powtarzania transakcji. W tym celu na taśmie śladowej muszą być zapisywane ekrany wejściowe transakcji aktualizujących bazę danych. Takie rozwiązanie zostało przyjęte w najnowszych wersjach systemu DMS-1100 zawierających tzw. INTEGRATED RECOVERY SYSTEM. Na taśmie śladowej w tym systemie wymieszane są komunikaty wejściowe z blokami będącymi kopiami stron bazy danych. Wyższy poziom ochrony zapewni rozdzielenie obu typu danych na dwie fizycznie taśmy. Pewną możliwość w tym zakresie dają systemy operacyjny lub komunikacyjny, dopuszczające kronikowanie transakcji komunikacyjnych. O ile nie ma takiej możliwości, warto samemu wykonać potrzebne oprogramowanie dla tej funkcji. Opłaca się to, gdyż w razie uszkodzenia taśmy śladowej SZBD zostaje jeszcze możliwość automatycznego powtórzenia transakcji za żądany okres czasu i doprowadzenie w ten sposób bazy do poprawnego stanu (rys. 11.1).

11.2.2. Podstawowe procedury odtwarzania bazy

Procedury zebrane tutaj można podzielić na trzy grupy - procedury cofania, odtwarzania i ponownego startu transakcji. Procedura cofania (ang. rollback) usuwa skutki pracy programu, który zmodyfikował bazę. Może być wykonana automatycznie przez system lub na polecenie programu. System zarządzania bazą decyduje się na cofnięcie programu z uwagi na wzajemne zablokowanie się dwóch programów lub na zlecenie specjalnie uprzywilejowanych programów żądających natychmiastowego zamknięcia obszaru (ang. DOWN IMMEDIATE). Najczęstszym przypadkiem jest jednak błędne wykonanie instrukcji DML programu, który zmodyfikował bazę danych. Po rozpoznaniu błędu program najczęściej decyduje się na rozłączenie z bazą i zakończenie, co powinno poprzedzić odtworzenie bazy do stanu sprzed początku programu.

Kolejna grupa procedur to procedury odtwarzające bazę. Ze względu na zakres działania, jak i narzędzia, na jakich te procedury bazują, można je podzielić na - odtwarzanie małe (ang. quick - recovery), częściowe (ang. selective-recowery) i pełne (ang. long-recowery). Odtwarzanie małe wykonywane jest najczęściej po zawieszeniu systemu lub błędzie systemowym. Służy do odtworzenia bazy do stanu sprzed awarii, czyli usuwa efekt pracy wszystkich programów pracujących w bazie w momencie awarii. W związku z tym, podobnie jak procedura cofania używa ona zapisów w zbiorze QLF, kopiując bloki "PRZED-ZMIANĄ" w miejsce zmodyfikowanych przez program stron. Oczywiście w czasie odtwarzania żaden inny program nie może pracować z bazą danych. Z kolei odtwarzanie częściowe lub pełne wykonywane jest w przypadku błędnego zakończenia małego odtwarzania bądź znalezienia formal-

nego błędu w bazie. Bazuje ono na informacjach zapisanych w dzienniku systemu, czyli w przypadku DMS-1100 na taśmie śladowej (ang. AUDIT-TRAIL-TAPE). Odtwarzanie odbywa się w dwóch etapach. Pierwszym z nich jest ładowanie bazy z odpowiedniego dumpu, drugim odtwarzanie programów od momentu wykonania dumpu do wyspecyfikowanego punktu czasowego (w szczególności wyspecyfikowany może być wprost punkt kontrolny). Jeżeli pobieranie punktów kontrolnych miało charakter statyczny, to po osiągnięciu tego punktu na taśmie śladowej odtwarzanie kończy się. Gdy miało charakter dynamiczny, system musi cofnąć bazę do stanu sprzed rozpoczęcia pierwszego programu aktywnego w momencie zapisu punktu kontrolnego. Jak już wiemy, do tego celu służą bloki "PRZED-ZMIANĄ" zapisane na taśmie śladowej. Samo odtwarzanie kolejnych programów polega na kopiowaniu tych stron, które program zmodyfikował. Są to bloki "PO-ZMIANIE" z taśmy śladowej systemu. Odtwarzanie częściowe od pełnego różni jedynie zakres odtwarzania. W przypadku odtwarzania częściowego zmieniamy jedynie niektóre obszary bazy. Oczywiście muszą ona być w tym momencie niedostępne dla innych programów, podczas gdy dostęp do pozostałych obszarów może przebiegać równocześnie w sposób normalny.

Pozostaje do omówienia procedura ponownego startu. Procedura ta wykorzystuje informacje zawarte na specjalnej taśmie śladowej systemu komunikacyjnego lub w przypadku prowadzenia tylko jednej taśmy śladowej część informacji zapisanych na niej.

Dokładniej mówiąc, procedura pobiera z taśmy kopie ekranów wejściowych wszystkich lub wybranych transakcji z jakiegoś okresu czasu i przekazuje je do systemu komunikacyjnego (lub operacyjnego) celem wykonania. W rezultacie procedura powtarza pracę wykonaną przez system wcześniej, podczas gdy procedury odtwarzania jedynie tę pracę symulowały znając końcowe jej efekty. Procedura ponownego startu trwa oczywiście o wiele dłużej niż procedury odtwarzania, jednak zapewnia mimo wszystko większy stopień zabezpieczenia danych. Startuje się ją w momencie błędnego wykonania pełnego odtwarzania spowodowanego np. fizycznym błędem na taśmie magnetycznej. Można ją również używać po poprawnym przebiegu pełnego odtwarzania celem doprowadzenia stanu bazy od ostatniego punktu kontrolnego do samego momentu awarii. Procedura zapewnia automatyczne powtórzenie w tej samej kolejności wszystkich transakcji w wybranym odcinku czasu.

11.2.3. Kontrola poprawności bazy

Aby baza danych była wiarygodna, należy systematycznie kontrolować jej zawartość. Do tego celu najlepiej nadają się własne programy użytkownika opracowane pod kątem konkretnych zastosowań. Pokazują one tę samą część bazy w różnych przekrojach, co często pozwoli na wykrycie formalnego błędu w bazie. Pewien rodzaj kontroli zapewnia również oprogramowanie systemowe. I tak np. funkcja VERIFY specjalnego uprzywilejowanego programu DMU w systemie DMS-1100 służy do kontroli poprawności łańcuchów obliczeniowych i łańcuchów rekordów w grupach. Dzięki tej funkcji może zostać wy-

kryte przerwanie bądź zapętlenie takiego łańcucha. Po wykryciu i zlokalizowaniu błędu można przystąpić do jego poprawienia. Można tego dokonać poprzez własne programy, procedury odtwarzania lub w sposób bezpośredni na bazie (funkcja PATCH programu DMU). Częsta i systematyczna kontrola bazy danych zapewnia więc wcześniejsze wykrycie błędów w bazie, a co za tym idzie - wpływa na dyspozycyjność bazy.

11.2.4. Systemowa kontrola danych przed wprowadzaniem

Aczkolwiek kontrolowanie przez system wprowadzanych informacji jest trudne, to pewne rozwiązanie w tym zakresie można jednak spotkać. Stosunkowo najczęściej spotykane jest systemowe zabezpieczenie przed wprowadzaniem duplikatów do bazy. Chodzi tutaj o rekordy - duplikaty ze względu na klucz w metodzie alokacji (indeksowo-sekwencyjna lub algorytmiczna) oraz ze względu na porządek określony w jakiejś grupie. Projektant może na poziomie schematu logicznego określić, w jakich przypadkach duplikaty nie są dozwolone. Wtedy też za wprowadzanie rekordów odpowiedzialny będzie system. Próba wprowadzenia do bazy duplikatu zakończy się błędem.

Innym rozwiązaniem jest zastosowanie w systemie DMS-1100 specjalnych procedur kontrolujących rekord przed zapisem. Procedury te, różne dla różnych rekordów, są zdefiniowane podobnie jak procedury sterujące dostępem. W opisie rekordu w schemacie logicznym występuje klauzula "CHECK IS PROCEDURE nazwa - paragrafu". Kiedy system ma wprowadzić nowy rekord do bazy, a więc przy czasownikach STORE lub MODIFY, sterowanie zostanie odesłane do specjalnej procedury użytkownika kontrolującej dane. Podobne procedury można definiować na poziomie pól elementarnych w rekordzie i będą one wywoływane w momencie aktualizacji tych pól. W tym zresztą przypadku klauzula "CHECK IS ..." może mieć zupełnie inny format, pozwalający na sprawdzanie, czy pole mieści się w podanym zakresie wyspecyfikowanym wartością największą lub najmniejszą pola lub jego cechy szczególne, np. wartość zero lub różna od zera.

12. PRACA PROGRAMÓW W BAZIE DANYCH

Po zdefiniowaniu i wygenerowaniu schematu opisującego zawartość logiczną bazy danych oraz po zainicjowaniu i wstępnym załadowaniu bazy rekordami użytkownik może przystąpić do eksploatacji bazy. Eksploatacja bazy polega na wykorzystaniu informacji zawartych w bazie oraz jej modyfikowaniu poprzez wprowadzanie nowych rekordów czy usuwanie starych. Eksploatacja bazy z reguły odbywa się poprzez oryginalne programy użytkownika, aczkolwiek pewne czynności na bazie, głównie kontrolne, można również wykonać przez oprogramowanie standardowe (np. weryfikację łańcuchów, dump stron itp.). Narzędziem pracy programistów są języki manipulowania danymi (DML), które są integralną częścią każdego SZBD. Rozróżniamy dwa typy języków manipulacji danymi. Pierwszy z nich, zwany podjęzykiem, jest osadzony w języku głównym, którym może być COBOL, FORTRAN czy PL/1. Programy pisane w języku głównym wzbogacane są o instrukcje DML. Wymagają one dwukrotnej kompilacji; pierwsza z nich to zmiana wszystkich komend DML na odpowiednie instrukcje języka głównego (przeważnie przesłania i wywołania podprogramów), druga to kompilacja z języka głównego na język wewnętrzny komputera. Drugim typem języków manipulacji danych są tzw. języki zapytań (ang. interactive query language). Języki zapytań umożliwiają użytkownikowi dostęp i manipulowanie danymi w bazie bez pośrednictwa języka głównego, a tylko przez interpretator tego języka pracujący w trybie konwersacyjnym. Umożliwiają więc one pracę z bazą danych bez konieczności pisania programów, a więc potencjalnym ich użytkownikiem nie muszą być informatycy. Jest to poważne udogodnienie, gdyż umożliwia pracę z bazą danych szerszemu gronu ludzi. Jednak z uwagi na konieczne w takim przypadku uogólnienia projektowe języki zapytań nie są efektywne w każdym projekcie bazy danych. Programy w języku głównym wraz z osadzonym w nim DML lepiej wykorzystują specyfikę przetwarzania danych z bazy. W związku z powyższym w niniejszym rozdziale ograniczymy się do języków manipulacji osadzonych w programie głównym. Pokażemy budowę takiego programu, zasady korzystania z systemowych tablic i buforów oraz pewne zasady przy pisaniu takich programów. Podobnie jak wcześniejsze odniesienia również w tym miejscu podamy jako przykład system DMS-1100. Ściślej mówiąc, przedstawimy język DML współpracujący z językiem COBOL jako językiem głównym.

12.1. Budowa programu w DML

Jak mówiliśmy poprzednio, DML jest podjęzykiem osadzonym w języku głównym, np. COBOL-u. W związku z tym program współpracujący z bazą danych wygląda tak samo jak każdy inny program w tym języku, a więc w wypadku COBOL-u składa się z tych samych co on części (działów, sekcji i paragrafów). Jedyna różnica tkwi w dwóch miejscach - nowej sekcji o nazwie INVOKE oraz w dziale procedur (PROCEDURE DIVISION), w której pojawiły się komendy nie będące czasownikami języka COBOL. Sekcja INVOKE zawiera informacje o bazie danych (a właściwie nawet o jej części), z jaką program chce pracować. Realizowane jest to przez podanie nazwy schematu bazy danych oraz zadeklarowanie, które rekordy, obszary i grupy należy do programu skopiować.

W sekcji tej zdefiniowane są również nazwy paragrafów w programie, do których należy oddać sterowanie w przypadku natrafienia na błąd w bazie danych lub w momencie zablokowania się dwóch programów (ang. deadlock). Ponadto zdefiniowane są tutaj bufory robocze programu do współpracy z SZBD. Mamy tutaj na myśli zarówno bufor pośredniczący w transmisji rekordów z bazy danych (ang. record delivery area), jak i tablicę komunikacyjną, w której SZBD zapisuje informacje o stanie pracy programu w bazie (ang. data management communication area). W tablicy tej zawarte są między innymi informacje o błędach w pracy z bazą i o stanie bieżącym rekordów w programie. Elementy tej tablicy mogą być wykorzystywane (zwłaszcza kontrolowane) w dziale procedur programu. Ponadto w INVOKE można zadeklarować jeszcze jeden bufor do automatycznego obliczania parametrów statystycznych pracy programu, takich jak: liczba wykonywanych instrukcji, średnie czasy oczekiwania w różnego typu kolejkach i inne. Należy zaznaczyć, że po wstępnej kompilacji opisy tych buforów zostaną automatycznie wygenerowane w żądanej części działu danych (tzn. w WORKING-STORAGE SECTION, COMMON-STORAGE SECTION czy LINKAGE-STORAGE SECTION).

Podobnie jest z opisami rekordów, które zostaną skopiowane ze schematu do odpowiedniej sekcji danych. Programista w dziale procedur może więc używać wszystkich nazw zdefiniowanych w schemacie (w szczególności wszystkich nazw pól w rekordzie) oraz nazw standardowych składających się na opis buforów roboczych programu.

Omówmy z kolei dział procedur programu zawierający w porównaniu z COBOL-em szereg nowych komend. Nie chodzi nam w tym miejscu o przedstawienie dokładnej składni DML (która zresztą zostanie zaprezentowana jako załącznik 12.1), lecz o pokazanie pewnych zasad i możliwości tego języka. Wszystkie czasowniki DML (których jest raptem kilkanaście) można pogrupować, mając na uwadze, co jest obiektem ich działania. I tak: pierwsza grupa obejmuje czasowniki współpracujące z SZBD. Są to komendy IMPART (zgłoszenie programu do SZBD) i DEPART (rozłączenie programu z SZBD). Druga grupa to czasowniki sterujące dostępem do obszaru. Mamy tu na myśli cza-

sowniki OPEN (otwarcie obszaru do określonego typu przetwarzania), CLOSE (zamknięcie obszaru) i FREE (częściowe zamknięcie obszaru, tzn. odesłanie stron z buforów systemowych do obszaru i wyzerowanie zbioru zawierającego kopie zmienianych przez program stron). Pozostałe czasowniki dotyczą rekordu, który jest podstawową jednostką informacyjną z bazy danych. Rozróżnić tu można czasowniki, które ustalają stan bieżący rekordu (STORE, FIND, FETCH) oraz czasowniki, które dotyczą rekordu bieżącego w programie. Omówmy kolejno funkcje tych czasowników. STORE powoduje przemieszczenie rekordu z buforów programu do odpowiedniego obszaru bazy i to zgodnie z zasadami alokacji tego typu rekordu. Czasowniki FIND i FETCH służą do pobierania wskazanego poprzez odpowiednią specyfikację rekordu. Różnica między nim jest taka, że FETCH dostarcza rekord do samego programu (do odpowiedniego bufora), natomiast FIND kopiuje jedynie stronę zawierającą szukany rekord z bazy do buforów systemu. Oczywiście oba te czasowniki zmieniają czy ustalają stan bieżący rekordu. O różnych możliwościach specyfikowania szukanego rekordu powiemy obszerniej w rozdziale następnym. Omówmy czasowniki ostatniej grupy. Są wśród nich czasowniki zorientowane na rekordy oraz na grupy logiczne, jednak ich cechą wspólną jest to, że zawsze działają na rekordzie uprzednio ustalonym jako bieżący. Oto one: Czasownik MODIFY powoduje zmianę zawartości rekordu oraz w niektórych przypadkach zmianę jego położenia w stosunku do innych rekordów bazy (np. zmiana klucza do sortowania w grupie czy zmiana rekordu właściciela). Czasownik DELETE służy do usuwania rekordu bieżącego z bazy danych wraz ze wszystkimi tego konsekwencjami, jak: aktualizacja łańcuchów rekordów czy usunięcie rekordów podporządkowanych automatycznie. Czasowniki INSERT i REMOVE służą do czasowego wstawienia lub usunięcia rekordu z grupy zdefiniowanej jako manualna. Czasownik ACQUIRE ma znaczenie jedynie dla grup z tablicami wskaźników. Służy on do skopiowania tablicy wskaźników podporządkowanej rekordowi bieżącemu w programie do wskazanego bufora programu, co umożliwi dalszą kontrolę czy przetwarzanie tablicy. Pozostałe czasowniki, takie jak: KEEP, LOG czy IF, mają mniejsze znaczenie, a w każdym razie nie są bezpośrednio związane z działaniem na bazie danych i dlatego nie będą dokładniej zaprezentowane.

12.2. Uwagi dotyczące programowania w DML

Cechą większości języków manipulacji danymi w różnych systemach zarządzania bazą danych jest odciążenie programisty i projektanta od potrzeby pamiętania i stosowania adresów fizycznych rekordów w bazie. Funkcję tę wykonuje moduł "METODY DOSTĘPU", który jest usytuowany na zewnątrz systemu lub jest jego integralną częścią.

W podjęzyku DML systemu DMS-1100 również mamy taką sytuację. Programista (projektant) może używać logicznego adresu rekordu (tzn. nr strony i nr rekordu na stronie) nie zdając sobie sprawy z faktycznego umiejsco-

wienia tego rekordu w przydzielonej pamięci. Dostęp do rekordu można organizować w bazie danych na szereg różnych sposobów. Najszybsza jest adresacja bezpośrednia, a więc wprost podanie adresu logicznego rekordu. Metodę tę stosujemy w przypadku rekordów o dostępie "direct" oraz w przypadku innych typów rekordów, o ile tylko znamy ich adres. Stąd pierwsza uwaga, jeżeli do tego samego rekordu musimy dochodzić kilkakrotnie, a bufory programu (systemu) są za małe na przechowanie tego rekordu, przez cały czas pracy programu po pierwszym odszukaniu rekordu w bazie należy zapamiętać jego adres. Można to zrealizować poprzez przeniesienie adresu rekordu bieżącego z DMCA do dowolnego pola roboczego programu. Metoda ta jest skuteczna nawet w przypadku rekordów "calc", gdyż liczba wymaganych operacji WE/WY (w tym przypadku 1) będzie mniejsza od średniej liczby dostępów w tej metodzie (1-2). Do rekordów o dostępie algorytmicznym lub indeksowo-sekwencyjnym, o ile tylko znamy klucz, powinniśmy dochodzić w sposób bezpośredni, a więc zgodny z metodą alokacji. Oczywiście zastrzeżenie dotyczące wielokrotnego dostępu do tego samego rekordu jest obowiązujące. Jeżeli nie znamy klucza tych rekordów oraz do rekordów o dostępie przypadkowym (poprzez grupę), to jedynym sposobem odszukania tych rekordów jest sekwencyjne przeszukiwanie całego obszaru czy grup. Jeżeli do danego rekordu można dojść poprzez różne grupy, to o wyborze właściwej z nich, poza oczywiście danymi wejściowymi określającymi niekiedy jednoznacznie główną ścieżkę dostępu, powinna decydować efektywność przetwarzania. Można ją policzyć biorąc za jednostkę dostęp logiczny do rekordu. Z tak przyjętego kryterium nasuwa się momentalnie wniosek, że w razie możliwości dojścia do tego samego rekordu przez różne grupy powinniśmy wybrać tę, która zawiera najmniej rekordów członkowskich w każdym swoim wystąpieniu. Podobny wniosek można wysnuć dla grup, w których rekord członkowski jest pamiętany metodą algorytmiczną "calc". Przeglądanie przez program pojedynczego wystąpienia takiej grupy może być sensowne, aczkolwiek wymagać będzie praktycznie wykonania tylu dostępów WE/WY, ile jest rekordów członkowskich w tej grupie. Natomiast przetwarzanie w ten sposób wszystkich wystąpień grupy jest zdecydowanie niewskazane. Przecież przy takiej metodzie może okazać się, że z bazy danych należy wielokrotnie ściągnąć tę samą stronę. Może więc dojść do tego, że liczba operacji WE/WY przekroczy liczbę stron w tym obszarze. Lepszą metodą w takim przypadku wydaje się więc sekwencyjne czytanie wszystkich stron obszaru i wybieranie do dalszego przetwarzania (np. sortowania) rekordów określonego typu. Do tego celu można używać wprost instrukcji: "FIND NEXT nazwa-rekordu RECORD WITHIN..." lub w przypadku jednoczesnego wybierania kilku typów rekordów "FIND NEXT RECORD WITHIN...". Stosując ten drugi sposób, po odczycie każdego rekordu należy oczywiście sprawdzić, jakiego typu był ten rekord. Można tego dokonać, gdyż w DMCA są informacje o nazwie (kodzie) rekordu bieżącego. Sprawdzenie typu rekordu jest wskazane przed komendami aktualizującymi bazę danych. Może się bowiem zdarzyć, że na skutek jakiegoś

błędu czy nieprzewidzianego wypadku rekordy są rozmieszczone w bazie inaczej, niż się tego spodziewamy. Sprawdzanie typu rekordu przed usunięciem czy modyfikacją rekordu uniemożliwi pogłębienie się błędnej sytuacji w bazie. Sprawdzanie typu rekordu może następować programowo (przez porównanie właściwej nazwy z nazwą aktualną z DMCA) lub automatycznie przez system, o ile zastosujemy składnię typu:

"DELETE nazwa - rekordu RECORD".

Czasownik DELETE działa w tym przypadku na rekordzie bieżącym programu, ale gdy rekord bieżący jest innego typu, niż podano, system rozpozna ten fakt i uniemożliwi wykonanie instrukcji sygnalizując błąd.

Jeżeli chodzi o sterowanie programem po natrafieniu na błąd w bazie (czy błędnie wykonaną instrukcję), to mamy do wyboru kilka możliwości. Pierwszą z nich jest całkowite zignorowanie błędu, a więc oddanie sterowania do następnej instrukcji programu. Inną możliwością jest zdefiniowanie w sekcji INVOKE generalnego paragrafu błędów. Jest to miejsce w programie, do którego zostanie przekazane sterowanie w przypadku błędnego wykonania jakiejkolwiek instrukcji DML programu. Generalny paragraf błędów najczęściej zawiera tylko wydruk statusów błędu z DMCA i oddaje sterowanie do procedur rozłączania się z bazą danych i kończących program. Oprócz generalnego paragrafu błędów można definiować szczególne paragrafy błędów. Są one definiowane na poziomie każdej instrukcji DML. W przypadku błędnego wykonania wskazanej instrukcji sterowanie zostanie przekazane do odpowiedniego miejsca programu. Tam po sprawdzeniu statusów błędu może okazać się, że pozornie błędna sytuacja jest właściwie niegroźna (np. dwukrotne otwarcie obszaru) lub w przypadku poważniejszym - można oddać sterowanie do generalnego paragrafu błędów. Podobne znaczenie ma paragraf, do którego przekażemy sterowanie w wypadku wykonania "rollback-u" naszego programu.

Każdy program w bazie - na skutek interwencji operatora czy systemu zarządzania bazą danych w momencie usuwania konfliktu dwu programów - może zostać cofnięty do swego punktu startu. Sterowanie w tym przypadku zostaje oddane do wskazanego w sekcji INVOKE paragrafu, który ma charakter generalny dla całego programu.

Ze specyfiki większości komend DML wynika konieczność zapewnienia i ochrony stanu bieżącego rekordów w programie. Przeważająca część komend działa bowiem na rekordzie bieżącym w programie. Przypomnijmy, że rekord bieżący to taki rekord, który ostatnio został odczytany lub zapamiętany przez program w bazie danych. Kontrola stanu bieżącego rekordów ma szczególne znaczenie w instrukcjach modyfikujących bazę. Chodzi o to, aby modyfikować czy usuwać z bazy ten rekord, który chcieliśmy, a nie inny rekord, który akurat jest rekordem bieżącym w programie. To pociąga niekiedy za sobą konieczność wielokrotnego odczytu tego samego rekordu. Na przykład, po zmodyfikowaniu rekordu (który oczywiście najpierw należało znaleźć) należy ten rekord wstawić do pewnej grupy manualnej. W tym celu mu-

simy określić występienie grupy, co najczęściej robimy przez znalezienie rekordu właściciela. Aby wykonać komendę INSERT, należy wcześniej wyjściowy rekord uczynić ponownie rekordem bieżącym w programie. W przeciwnym razie instrukcja INSERT zakończy się błędem (rekord bieżący będzie właścicielem a nie członkiem podanej grupy). Ponowne określanie rekordu bieżącego realizowane jest przez komendę FIND. Najszybciej jest ona wykonywana, gdy znany jest logiczny adres szukanego rekordu. Zauważmy, że dla każdego programu SZBD konstruuje tablicę bieżących rekordów odpowiednio w typie rekordu, obszaru czy grupy. Elementami tej tablicy są klucze bazy danych, a więc właśnie adresy logiczne rekordów. Stąd wynika wniosek, że powrót do stanu bieżącego rekordu można bardzo szybko uzyskać, wykonując instrukcję: "FIND CURRENT RECORD WITHIN...". Oddzielny problem to zachowanie stanu bieżącego rekordów przy sekwencyjnym przeglądaniu obszaru czy grupy poprzez instrukcję "FIND NEXT RECORD WITHIN...".

Często algorytm przetwarzania jest tak dobrany, że po znalezieniu kolejnego rekordu, np. z obszaru, należy dla tego rekordu wykonać szereg innych operacji na innych rekordach bazy. Jeżeli rekordy te są z tego samego co rekord główny obszaru, zmieni się oczywiście stan bieżący w obszarze. Kolejne wykonanie instrukcji "szukaj dalej" rozpocznie się z innego niż powinno być miejsca, w związku z czym przy sekwencyjnym przeglądnięciu całego obszaru szereg rekordów zginie (zostanie pominiętych). Aby temu zapobiec (a jest to dość typowy błąd u początkujących programistów), wystarczy zabezpieczyć stan bieżący po każdej instrukcji czytania lub zastosować klauzulę zabraniającą zmianę tego stanu w typie obszaru czy grupy. Klauzula ta, o postaci "SUPPRESS AREA" lub "SUPPRES SET", powinna być dodana do wszystkich instrukcji, które mogą zmienić stan bieżący między dwoma kolejnymi szukaniami (a więc do wszystkich instrukcji FIND, FETCH lub STORE umieszczonych w tej pętli programu).

Jeszcze jedną wygodną właściwością języka DML jest możliwość używania tzw. generalnych komend. Chodzi o to, że często ta sama instrukcja może być wykonywana dla różnego typu rekordów, grup czy obszarów. Aby nie powtarzać wielokrotnie tej samej instrukcji w programie z oryginalnymi nazwami rekordów, grup czy obszarów, można zamiast nich używać pól roboczych zawierających właściwe nazwy. Pola te powinny być specjalnie opisane z charakterystyczną klauzulą "USAGE IS AREA-NAME lub SET-NAME lub RECORD-NAME", którą DML zamienia na rozpis pola alfanumerycznego odpowiedniej długości. Komendy generalne mają szczególne zastosowanie w podprogramach oraz w ogólnych programach systemu działających na dużej liczbie obiektów bazy.

Reasumując, DML jest dla programisty bardzo wygodnym narzędziem dostępu do bazy danych. Jest prosty w obsłudze, daje szereg możliwości odczytania czy zmodyfikowania informacji w bazie. Zawiera pewne uogólnienia i pozwala na wybór metody postępowania w przypadku błędu. I choć jego komputerowa implementacja jest kosztowna w porównaniu z prostszymi na przy-

APPENDIX B. DML SYNTAX SKELETON

B.1. GENERAL

The following is the complete syntax skeleton of a DML program.

B.2. SYNTAX SKELETON

IDENTIFICATION DIVISION.

PROGRAM-ID. *program-id-name*.

⋮

ENVIRONMENT DIVISION.

⋮

DATA DIVISION.

SCHEMA SECTION.

INVOKE SCHEMA *schema-name*

[COPYING [WORKING / COMMON / LINKAGE] {ALL / *record-name-1* [, *record-name-2*] ...}]

[COPYING AREA {ALL / *area-name-1* [, *area-name-2*] ...}]

[COPYING SET {ALL / *set-name-1* [, *set-name-2*] ...}]

[RUN-UNIT-ID IS *run-unit-identification*]

[PRIORITY IS *integer-1*]

RECORD DELIVERY-AREA IS *record-delivery-area* [WORKING / COMMON / LINKAGE]

[LENGTH IS *integer-2* WORDS]

[OVERLAY {*record-delivery-area* / *record-name-3*} WITH {ALL / *record-name-4* [,*record-name-5*] ...}]

[OVERLAY *record-name-6* WITH *record-name-7* [,*record-name-8*] ...] ...

[SAVE DATA INCLUDES {COMMAND / RUN-UNIT} QUICK-BEFORE-LOOKS]

[[POINTER {AREA / AREAS} FOR INITIAL LOAD {IS / ARE} *integer-3*]]

[DMCA [AND RUN-UNIT-STATISTICS] {IS / ARE} {WORKING / COMMON / LINKAGE}]

[ERROR RECOVERY IS *general-error-paragraph*]

ROLLBACK IS *rollback-error-paragraph*.

[FILE SECTION.]

⋮

[COMMUNICATION SECTION]

⋮

[WORKING-STORAGE SECTION.

77-level programmer entered descriptions, including descriptions for any data base data names, defined in the schema, except those with a usage of area key.

01-level programmer entered descriptions for those data base data names, defined in the schema and required by run unit, including those which have a usage of area key.

01-level programmer entered descriptions.]

[COMMON-STORAGE SECTION.

77-level programmer entered descriptions, including descriptions for any data base data names, defined in the schema, except those with a usage of area key.

01-level programmer entered descriptions for those data base data names, defined in the schema and required by the run unit, including those which have a usage of area key.

01-level programmer entered descriptions.]

cd. zał. 12.1

```
[LINKAGE SECTION]
      :
[REPORT SECTION.]
PROCEDURE DIVISION.
      :
"Run Unit Oriented Commands"

    IMPART [ON ERROR GO TO error-paragraph].

    DEPART [WITH ROLLBACK] [ON ERROR GO TO error-paragraph].

    FREE [ON ERROR GO TO error-paragraph].

"Area Oriented Commands"

    OPEN ALL [USAGE-MODE IS {[EXCLUSIVE] [PROTECTED]} {RETRIEVAL} {UPDATE}]
                            {INITIAL     LOAD}
    [ON ERROR GO TO error-paragraph].

    OPEN {area-name-1} [USAGE-MODE IS {[EXCLUSIVE] [PROTECTED]} {RETRIEVAL} {UPDATE}]
         {identifier-1}                {INITIAL     LOAD}
    [, {area-name-2} [USAGE-MODE IS {[EXCLUSIVE] [PROTECTED]} {RETRIEVAL} {UPDATE}] ...
       {identifier-2}               {INITIAL     LOAD}
    [ON ERROR GO TO error-paragraph].

    CLOSE {ALL                                                  } [ON ERROR GO TO error-paragraph].
          {area-name-1 [, area-name-2] ...                      }
          {identifier-1 [, identifier-2] ...                    }

"Record Oriented Commands"

    STORE {record-name} [SUPPRESS {ALL                                  } CURRENCY UPDATES]
          {identifier-1}          {RECORD                               }
                                  {AREA                                 }
                                  {SETS                                 }
                                  {set-name-1 [, set-name-2] ...        }
                                  {identifier-2 [, identifier-3] ...    }
    [ON ERROR GO TO error-paragraph].
```

cd. zał. 12.1

```
{FIND  }
{FETCH}rse  [ SUPPRESS { ALL                                    }  CURRENCY UPDATES ]
                       { RECORD                                 }
                       { AREA                                   }
                       { SETS                                   }
                       { set-name-1 [, set-name-2] ...          }
                       { identifier-1 [, identifier-2] ...      }

    [AT END GO TO end-paragraph]  [ON ERROR GO TO error-paragraph].
```

"Where the record selection expression (rse) is one of the following"

```
Identifier-3 [, identifier-4]

CURRENT RECORD WITHIN {record-name-1} RECORD
                      {identifier-5}

{OWNER     }
{CURRENT   }
{NEXT      }
{PRIOR     }   RECORD WITHIN {set-name-3   SET  }
{FIRST     }                 {identifier-7 SET  }
{LAST      }                 {area-name-1  AREA }
{identifier-6}               {identifier-8 AREA }

{NEXT      }
{PRIOR     }
{FIRST     }  {record-name-2} WITHIN {set-name-4   SET  }
{LAST      }  {identifier-10}        {identifier-11 SET  }
{identifier-9}                       {area-name-2  AREA }
                                     {identifier-12 AREA }

[{NEXT  [DUPLICATE]}                                    ]
 {PRIOR            }                                    ]
 {FIRST            }   WITHIN {record-name-3} RECORD
 {LAST             }          {identifier-13}
[                                                       ]

{record-name-4} VIA {set-name-5} [USING data-base-identifier-1 [data-base-identifier-2]...]
{identifier-14}     {identifier-15}

NEXT DUPLICATE WITHIN {set-name-6} USING data-base-identifier-3 [, data-base-identifier-4]...
                      {identifier-16}

GET [ON ERROR GO TO error-paragraph].

MODIFY [data-base-identifier-1 [, data-base-identifier-2] ...]

    [ON ERROR GO TO error-paragraph].

DELETE [ONLY] [ON ERROR GO TO error-paragraph].
       [ALL]

KEEP [ON ERROR GO TO error-paragraph].
```

"Set Oriented Commands"

```
INSERT  INTO  { set-name-1 }  [ , INTO { set-name-2 } ] ... [ON ERROR GO TO error-paragraph].
              { identifier-1 }         { identifier-2 }

INSERT  INTO  ALL SETS  [ON ERROR GO TO error-paragraph].

REMOVE FROM { set-name-1 }  [ , FROM { set-name-2 } ] ... [ON ERROR GO TO error-paragraph].
            { identifier-1 }         { identifier-2 }

REMOVE  FROM  ALL SETS  [ON ERROR GO TO error-paragraph].
```

"Conditional Commands"

```
IF { set-name-1 }  SET [NOT] EMPTY;
   { identifier-1 }

   { statement-1 }  [ ELSE  { statement-2 } ].
   { NEXT SENTENCE }        { NEXT SENTENCE }

IF RECORD [NOT]  { OWNER }  OF { set-name-2 }  SET;
                 { MEMBER }    { identifier-2 }
                               { ANY        }

   { statement-3 }  [ ELSE  { statement-4 } ].
   { NEXT SENTENCE }        { NEXT SENTENCE }
```

"Supporting Commands"

```
LOG  identifier-1  WORDS [FOR RECOVERY POINT] [ON ERROR GO TO error-paragraph].

                                  { RUN-UNIT          }
                                  { record-name RECORD }
                                  { identifier-1 RECORD }
MOVE CURRENCY STATUS FOR          { area-name AREA     }
                                  { identifier-2 AREA  }
                                  { set-name SET       }
                                  { identifier-3 SET   }

    TO identifier-4  [ON ERROR GO TO error-paragraph].

                                  { RUN-UNIT          }
                                  { record-name RECORD }
                                  { identifier-5 RECORD }
MOVE  || AREA-KEY ||  FOR         { area-name AREA     }
      || AREA-NAME ||             { identifier-6 AREA  }
                                  { set-name SET       }
                                  { identifier-7 SET   }
                                  { identifier-8       }

    TO || identifier-9 ||  [ON ERROR GO TO error-paragraph].
       || identifier-10 ||
```

Rys. 12.1. Składnia ADMLP (system DMS-1100)

kład językami w relacyjnych bazach danych, to i tak w porównaniu do tradycyjnych systemów przetwarzania jest on poważnym udogodnieniem pracy programistów i przetwarzania danych w ogóle.

13. UWAGI KOŃCOWE

W jednym z początkowych rozdziałów wprowadziliśmy pojęcie administratora bazy danych, a więc osoby bądź grupy osób odpowiedzialnych za prawidłową eksploatację baz danych. Z uwagi na kapitalne znaczenie tej postaci w projektowaniu i eksploatacji systemów informatycznych opartych na bazach danych wróćmy do dokładniejszego omówienia roli administratora w systemie. Jest to teraz tym łatwiejsze, gdyż dotychczasowe rozdziały podręcznika pozwoliły na poznanie głównych problemów, z jakimi styka się użytkownik bazy. Administrator odgrywa istotną rolę w każdym z dotychczas przedstawionych problemów, od projektowania po reorganizację bazy i nadzór nad jej eksploatacją. Omówmy szczegółowo zadania administratora:

Generowanie SZBD

Jest to pierwszy problem, z jakim styka się administrator w momencie podjęcia decyzji o zastosowaniu baz danych. Z grubsza biorąc, polega on na dopasowaniu standardowych możliwości oferowanych przez autorów SZBD (producenta) do realnych potrzeb i możliwości użytkownika. Pomijamy w tym miejscu zarówno wybór sprzętu komputerowego, jak i wybór oprogramowania standardowego, które zresztą sprzęt niejako wymusza. Generowanie SZBD polega na wyborze procedur (programów) tworzących system i na określeniu wielkości systemowych tablic i buforów. Dla podjęcia prawidłowej decyzji o parametrach SZBD potrzebna jest znajomość sprzętu komputerowego, systemu operacyjnego, samego SZBD oraz charakterystyki eksploatowanych w przyszłości systemów użytkowych.

Projektowanie struktury logicznej i fizycznej bazy

Zagadnienie to zostało omówione dokładnie w początkowych rozdziałach podręcznika. Podkreślmy jedynie, że administrator decyduje o zawartości informacyjnej bazy, definiuje obiekty bazy oraz określa powiązania między nimi. Określa jednocześnie parametry fizyczne bazy, a więc wielkości obszarów oraz metody dostępu do danych w bazie.

Zabezpieczenie bazy danych przed niepowołanym dostępem

Odbywa się to poprzez definiowanie przez administratora procedur legalności dostępu do różnych obiektów bazy (obszar, rekord czy nawet pole w rekordzie). Problem ten może zostać również rozwiązany poprzez wprowa-

dzenie podschematu, który określi sposób, w jakim dane z bazy są widziane przez poszczególne programy użytkowe.

Zabezpieczanie integralności bazy danych

Specyfika eksploatacji baz danych polega na konieczności utrzymania ciągłej funkcjonalnej gotowości bazy oraz na zapewnieniu formalnej poprawności danych w każdym momencie eksploatacji. W razie powstania błędu w bazie eksploatacja programów jest ograniczona lub w ogóle niemożliwa aż do momentu powrotu bazy do stanu poprawnego. Stąd wynika wielkie znaczenie ochrony bazy przed zniszczeniem, co realizowane jest przez różnorodne narzędzia, takie jak: dump czy taśmy śladowe systemu. O wyborze metod ochrony i procedur odtwarzających bazę decyduje administrator. On również powinien być inicjatorem i realizatorem odtwarzania zniszczonej bazy do stanu poprawnego.

Weryfikacja danych w bazie

Administrator jako osoba odpowiedzialna za poprawność danych powinien mieć możliwość nadzoru nad danymi już w samym momencie ich wprowadzania. Może to być realizowane przez definiowanie procedur badania poprawności danych. Oprócz tego administrator powinien wykonywać wyrywkową lub pełną kontrolę danych w bazie, kontrolę wszystkich lub tylko wybranych obiektów, w różnych oczywiście przekrojach. Chodzi tu zarówno o sprawdzenie ciągłości wszystkich łańcuchów powiązań, jak i analizę różnego rodzaju sum kontrolnych. Okresowa kontrola całej bazy umożliwi wczesne wykrycie ewentualnego błędu w bazie i przyspieszy odtworzenie bazy do stanu poprawnego.

Badanie obciążenia systemu

Administrator odpowiada za poprawne, a więc i efektywne wykorzystanie bazy; on więc powinien kontrolować z reguły zmienne w czasie obciążenie systemu transakcjami. Kontrola ta pozwoli na wykrycie "wąskich gardeł" systemu oraz wybranie transakcji najbardziej obciążających system. Dokładna analiza tych programów może pozwolić na ich zoptymalizowanie lub wymusić przeprowadzenie zmian w samej bazie czy nawet SZBD.

Reorganizacja bazy

Jest to jeden z najtrudniejszych etapów pracy administratora. Potrzeba zmian bazy, wymuszona często przez życie, zderza się tu z koniecznością zachowania w systemie już raz wprowadzonych informacji. Inicjatorem reorganizacji powinien być administrator; on również powinien decydować o sposobie jej przeprowadzenia.

Współpraca z użytkownikami

Za użytkowników uważamy w tym miejscu bądź programistów opracowujących nowe programy bądź osoby bezpośrednio korzystające z bazy (np. za pomocą języków zapytań). Administrator zapewnia użytkownikom dostęp do określonych obiektów bazy (lub ich części), definiując jednocześnie procedury badania legalności tego dostępu. W specyficznych sytuacjach decyduje o zastosowaniu określonej strategii dostępu, co jest istotne zwłaszcza w sieciowych bazach danych, w których dostęp do rekordu może być realizowany różnymi metodami.

Do realizacji tych celów administrator powinien posiadać określone narzędzia. Narzędziami tymi będą głównie programy, standardowe i własne, które można podzielić na następujące grupy:

- programy ładujące bazę danymi początkowymi,
- programy reorganizacji bazy,
- programy odtwarzające bazę do stanu poprawnego,
- programy zabezpieczające bazę na nośnikach zewnętrznych,
- programy analiz statycznych do badania obciążenia systemu,
- programy kontrolno-sprawozdawcze do weryfikacji danych,
- programy do szybkiej kontroli i modyfikacji danych w bazie.

Niektóre z tych programów są właściwie częścią samego SZBD, co oznacza, że one są dostarczane przez producenta. Niekiedy jest to po prostu uprzywilejowany program pozwalający na realizację specjalnych funkcji niedostępnych zwykłym programom. Przykładem może być program DMU (Data Management Utility) wchodzący w skład systemu DMS-1100 [12],[6]. Z kolei programy kontrolne, sprawozdawcze czy analiz obciążenia winny być z reguły opracowane przez administratora, gdyż korzystają one z określonej specyfiki systemów użytkowych.

Na uwagę zasługuje grupa programów przeznaczonych do szybkiej kontroli i modyfikacji bazy. Są to programy ogólne, które zastępują pracę pewnych szczególnych programów użytkowych napisanych często tylko do zrealizowania jednej funkcji w ściśle określonej sytuacji (np. usunięcie jakiegoś błędu w bazie). Liczba takich "jednorazowych" programów w eksploatowanych systemach jest często duża. Zastąpienie ich ogólnym programem pozwalającym na sprawdzenie czy modyfikację dowolnego obiektu bazy (z dokładnością do pola rekordu) jest bardzo przydatne. W systemach wyposażonych w języki zapytań problem ten nie istnieje, gdyż zarówno odczyt, jak i modyfikację dowolnych danych z bazy można zrealizować stosując określone zdania tego języka.

Zastanówmy się nad powodem coraz szerszego stosowania baz danych w różnych ośrodkach informatycznych. Z analizy rynku amerykańskiego [18] wynika, że z początkiem 1978 r. prawie 10% wszystkich ośrodków w USA zastosowało już SZBD. Prognozy ekspertów mówią, że liczba użytkowników baz danych w USA wzrośnie w najbliższych latach do 50-75%. Podobne proporcje

można odnaleźć w Europie. Dlaczego tak się dzieje? Czy jest to moda czy konieczność?

Potencjalne korzyści ze stosowania systemów z bazami danych to zwiększenie efektywności działania przedsiębiorstwa oraz redukcja kosztów przetwarzania. Baza danych umożliwia przedsiębiorstwu centralne sterowanie danymi operacyjnymi. Dokładna i szybka informacja z bazy ułatwia kierowanie przedsiębiorstwem. Przykłady można mnożyć. Przez uzyskiwanie bieżącej informacji można usprawnić obsługę klientów, wykrywać opóźnienia w produkcji i ich źródła, śledzić realizację zamówień czy zmniejszać zapasy produkcyjne. Wprowadzenie bazy może również wpływać na lepsze wykorzystanie personelu oraz lepsze informowanie kierownictwa ułatwiające zarządzanie przedsiębiorstwem.

Bazy danych mogą również wpłynąć na obniżenie kosztów przetwarzania systemów informatycznych. Interesujące jest zestawienie dotyczące systemu RODAN [23]. Otóż przewiduje się, że wprowadzenie SZBD RODAN skróci czas projektowania o 25%, czas testowania o 40%. Oszczędność dysków na skutek uniknięcia redundancji danych wyniesie prawie 50%. Oczywiście liczby te odnoszą się do ośrodków, w których pracownicy mają już pewne doświadczenie w projektowaniu baz danych. W początkowym etapie (szkolenie pracowników!) czasy projektowania czy wdrażania baz danych mogą być dłuższe niż w przypadku systemów tradycyjnych. Jednak później dzięki bazom danych można wyraźnie skrócić cykl wdrożenia nowych systemów informatycznych, a niektóre nowe funkcje użytkowe mogą być wykonane niemal natychmiast z istniejących w bazie informacji.

Poza efektami wymiernymi bazy danych wpływają na poprawę warunków eksploatacji systemów informatycznych. Pozwalają na dzielenie tych samych danych między wielu użytkowników, co ułatwia równoważenie sprzecznych żądań różnych użytkowników. Zwiększona jest kontrola legalności dostępu do danych, ułatwiona i ujednolicona jest ochrona danych przed zniszczeniem. Z kolei wprowadzanie standardów dokumentacyjnych i eksploatacyjnych optymalizuje samo przetwarzanie i wymusza uporządkowanie otoczenia sterowanego systemu. O zmniejszeniu czy nawet całkowitym uniknięciu redundancji już wspomnieliśmy. Można jeszcze powiedzieć o wpływie bazy danych na szybkość uzyskiwania informacji. Właściwie zaprojektowana baza zapewnia szybką pracę programom najbardziej obciążającym system, a więc w sumie wpływa na efektywność przetwarzania.

Wprowadzenie baz danych pociąga za sobą pewne koszty, które związane są głównie z oprogramowaniem bazy, sprzętem, konwersją zbiorów i szkoleniem pracowników. Główny udział ma tutaj koszt samego SZBD i ewentualnie koszt programów pomocniczych wspomagających jego pracę. Zastosowanie baz danych stawia również pewne wymagania sprzętowe. Chodzi tu głównie o zwiększenie pojemności pamięci zewnętrznej i pamięci operacyjnej (wymuszone z reguły przez wielkość SZBD) oraz o ewentualną rozbudowę sieci komunikacyjnej. Przeniesienie posiadanych w przedsiębiorstwie zbiorów danych do

bazy może być trudnym problemem, zwłaszcza w przypadku wielkich i zróżnicowanych zbiorów danych. Może być tu wymagane przeformatowanie danych czy ich konwersja. Również samo ładowanie już spreparowanych danych może być trudnym problemem, o czym wspominaliśmy już w rozdziale 9. Na koszt bazy danych wpłynie również zmiana struktury zatrudnienia w ośrodku. Przede wszystkim niezbędne jest zatrudnienie administratora bazy, osoby lub osób o dużym doświadczeniu informatycznym, znających zarówno techniczną stronę komputera, jak i oprogramowanie, a zwłaszcza sam SZBD. Należy również przeprowadzić szkolenia projektantów i programistów z zakresu obsługi baz danych lub wymienić pracowników tych zespołów, co może również wpłynąć na koszt wdrożenia bazy.

Reasumując, można stwierdzić, że wdrożenie baz danych podlega pewnym ekonomicznym prawidłowościom. Oto one:

- instalacja bazy danych wymaga dużych środków inwestycyjnych (zakup sprzętu i oprogramowania),
- inwestycje nie zwracają się natychmiast (główne wydatki przypadają na początkowy etap wdrożenia, główne efekty zastosowania bazy występują dopiero po 3-5 latach od rozpoczęcia wdrażania),
- efektywność inwestycji jest potencjalnie duża.

Zastanówmy się jeszcze nad kierunkami zastosowań baz danych. Wydaje się że bazy mogą być stosowane w najróżniejszych systemach, specjalistycznych i ogólnych, w dużych i małych przedsiębiorstwach i organizacjach. Typowymi jednak przykładami zastosowania baz są systemy:

- planowanie i kontrola produkcji (np. HMN Szopienice),
- rezerwacja (np. system rezerwacji miejsc lotniczych w LUFTHANSA),
- banki,
- ewidencja ludności,
- szpitale,
- zakłady ubezpieczeń,
- biblioteki i wyższe uczelnie.

Mimo sporej tutaj różnorodności większość tych systemów łączą wspólne cechy - duża liczba danych i konieczność szybkiego dostępu do nich. Właśnie w systemach wymagających szybkiej reakcji, a więc w systemach konwersacyjnych lub czasu rzeczywistego, zastosowanie tego nowego narzędzia informatyki, jakim są bazy danych, jest szczególnie uzasadnione. Jest to zgodne z tendencjami zaobserwowanymi w całym świecie (p. [37], [38], [39]). Prognozy na najbliższe lata zakładają rozszerzenie zakresu stosowania baz danych oraz rozwój SZBD. Te ostatnie, wspólnie z oprogramowaniem służącym do obsługi transakcji komunikacyjnych mają wejść w skład systemów operacyjnych dużych komputerów. To najlepiej potwierdza wagę i znaczenie baz danych w procesie szybkiego przetwarzania informacji.

LITERATURA

1. Węgrzyn S., Gille J.C., Videl P.: Analiza doboru skorowidzowej organizacji zbiorów. Podstawy Sterowania. T. 9, 1979, z. 4.
2. PAN, ZSAK Gliwice: Wprowadzenie do zagadnień komputerowych metod organizacji zbiorów danych systemów pracujących w czasie rzeczywistym. 1978.
3. Data C.J.: Wprowadzenie do baz danych. Wydawnictwa Naukowo-Techniczne. Warszawa 1981.
4. Kapuścik W.: Liczba logicznych dostępów w bazach danych typu CODASYL. Podstawy Sterowania. T. 10 1980, z. 2.
5. ZOI HMN "Szopienice": System zabezpieczania i odzyskiwania informacji - dokumentacja eksploatacyjna. 1979.
6. SPERRY-UNIVAC, UP 7909.1. DMS 1100 Level 8R1 - System Supoort Functions. 1979.
7. SPERRY-UNIVAC, UP 7907.2. DMS 1100 Schema Definition. 1979.
8. SPERRY-UNIVAC, UP 7992.3. DMS 1100 Cobol Data Manipulation Language. 1979.
9. SPERRY-UNIVAC, UP 8601. Integrated Recovery - User Guide. 1979.
10. ZOI HMN "Szopienice": System kontroli i modyfikacji baz danych BADACZ. Dokumentacja eksploatacyjna. 1980.
11. SPERRY-UNIVAC. UP - 8505. DMS 1100 Subschema Definition. 1978.
12. Kapuścik W., Mrowiec Z.: DMU narzędziem pracy administratora baz danych. Problemy Postępu Technicznego, 2/80.
13. Wiederhold G.: Data base design. New York Mc Graw-Hill Book Company. 1977.
14. Tsichritzis D.C., Lochowsky F.H.: Data base management systems. Academic Press, New York 1977.
15. Raport Grupy Roboczej ds. Baz Danych CODASYL - OBRI, Warszawa 1977.
16. "System Zarządzania Bazą Danych RODAN", OBRI, Warazawa.
17. "System Zarządzania Bazą Danych RODAN". Problemy zastosowań" - CPiZI. Warszawa 1980.
18. Mc Fadden F.R., Suver J.D.: Costs and benefits of a data base systems, Harvard Business Review, I/II 1978.
19. Tsichritzis D.C., Lochowsky F.M.: Designing the data base. "Datamation", 1978, nr 8.
20. Szafrański B.: Projektowanie bazy danych. "Wojskowy Przegląd Organizacji i Informatyki", 1975, nr 3.
21. Lyon J.K.: An introduction to data base design. Honeywell Information System, Inc. Schenectady, New York 1971.
22. Palmer I., Gradwell D.: The key to effective use of data. "Data Processing", 1976, nr 5.
23. Głownia G.: Prace badawczo-rozwojowe w Zjednoczeniu Informatyki. "Informatyka", nr 1/80.
24. Palmer I.: Database Systems. A Practical Reference. CACI International, 1975.

25. De Blasis J., Johnson T.: Data base administration. Classical pattern, some experiences and trends. AFIPS Press, 1978.

26. Maciaszek L.: Reorganizacja baz danych. Informatyka nr 2/1980.

27. Mc Ge W.C.: On the evaluation of data models. Trans. on Database Systems. 1976, nr 4.

28. Nijssen G.M.: Set and GODASYL set or coset. "Special Working Conference on Data Base Description", North - Holland 1975.

29. Robinson K.A.: An analysis of the uses of the CODASYL set concept. "Special Working Conference on Data Base Description", North - Holland 1975.

30. Coffman E.G., Elphick M.J., Shoshani A.: System deadlocks. Computing Surveys, 1971, 3 nr 2.

31. Engels R.W.: Currency and concurrency in the COBOL data base facility. "Working Conference on Modelling in Data Base Management Systems" North - Holland 1976.

32. Marci P.P.: Deadlock detection and resolution in a CODASYL - based data management systems. "International Conf. on the Management of Data", ACM, 1976.

33. Hawley D.A., Knowles J.S., Tozer E.E.: Database consistency and the CODASYL DBTG proposals. Comp. J., 1975, 18, nr 3.

34. Bachman C.W.: The Programmer as navigator. CACM, 1973, 16, nr 11.

35. Kay M.H.: An assessment of the CODASYL DDL for use with a relational subschema. "Working Conf. On Data Base Description", North - Holland 1975.

36. Taylor R.W., Frank R.L.: CODASYL data base management systems. ACM Computing Surveys, 1976, 8, nr 1.

37. Europejski Program Badawczy Diebolda: Postępy w dziedzinie systemów zarządzania bazą danych. ZETO, OBRI, nr 99, 1979.

38. Europejski Program Badawczy Diebolda: Wytyczne do długofalowej strategii realizacji baz danych. ZETO, CPIZI, nr 104, 1979.

39. Europejski Program Badawczy Diebolda.: Prognoza na lata 1980-1989. Przegląd wyników badań. ZETO, OBRI, nr 95, 1978.