

Na prawach rękopisu

**INSTYTUT CYBERNETYKI TECHNICZNEJ
POLITECHNIKI WROCŁAWSKIEJ**

Raport serii: PREPRINTY nr 42/99

**Metody syntezy programów
obiektowych
operujących na polimorficznych
strukturach danych**

(Rozprawa doktorska)

Zofia Kruczkiewicz

PROMOTOR:

prof. dr hab. inż. Adam Sielicki

Słowa kluczowe:

- modele ORM programów
- polimorfizm
- równoważność modeli programów
- efektywne modele programów
- synteza modeli programów

**WROCŁAW
kwiecień 1999**

Spis treści

1. Wprowadzenie	5
1.1. Informacje ogólne	5
1.2. Metody tworzenia oprogramowania zorientowanego obiektowo	8
1.3. Ocena oprogramowania obiektowego	11
1.4. Główne cele i teza pracy	13
2. Schemat konceptualny ORM	14
2.1. Definicje własności syntaktycznych	14
2.1.1. Podstawowe typy obiektowe	16
2.1.2. Podstawy formalne specjalizacji i polimorfizmu	21
2.2. Definicje własności semantycznych modelu	28
2.2.1. Populacja typów obiektowych	28
2.2.2. Populacja typów obiektowych związanych relacją specjalizacji i polimorfizmu	31
2.2.3. Przykłady populacji typów obiektowych	32
2.2.4. Wyrażenia relacyjne	35
2.2.5. Ograniczenia	37
2.2.6. Identyfikacja typów obiektowych	44
2.3. Przykład modeli ORM	52
3. Równoważność schematów konceptualnych	56
3.1. Własności grafu funkcjonalnego schematu konceptualnego	56
3.2. Metody oceny równoważności schematu konceptualnego	62
3.2.1. Warunki równoważności schematów konceptualnych	62
3.2.2. Proces badania równoważności schematów konceptualnych	64
3.2.3. Projekt algorytmu sprawdzania równoważności schematów konceptualnych	67
3.3. Przykłady zastosowania metody oceny równoważności schematów konceptualnych	73
4. Algorytm podziału schematu konceptualnego na klastry jako podstawa specyfikacji klas	91
4.1. Podstawy formalne budowy klastrowej schematu konceptualnego	92
4.1.1. Syntaktyka klastrów	92

4.1.2. Semantyka klastrow.....	93
4.2. Algorytm podziału na klastry.....	96
4.3. Przykłady podziału na klastry modeli ORM	100
5. Metody oceny równoważnych schematów konceptualnych	109
5.1. Podstawy oceny oprogramowania we wszystkich fazach jego powstawania.....	110
5.2. Podstawy formalne oceny spójności grafu funkcjonalnego schematu konceptualnego	115
5.3. Metody pomiaru charakterystyk wewnętrznych modelu ORM, uwzględniające wpływ polimorfizmu ze specjalizacją.....	123
5.3.1. Metryka połączeń między typami obiektowymi	123
5.3.2. Metryki spójności modelu	124
5.4. Przykład zastosowania metryk do oceny równoważnych schematów konceptualnych ORM.....	127
6. Metoda tworzenia i wyboru efektywnego modelu opartego na polimorficznej strukturze danych	138
6.1. Podstawy formalne metody syntezy modeli opartych na polimorficznych strukturach danych.....	138
6.2. Metody syntezy modeli ORM.....	148
6.2.1. Iteracyjna metoda budowy modelu opartego na polimorficznych strukturach danych.....	150
6.2.2. Wybór najlepszego rozwiązania.....	156
7. Podsumowanie i wnioski	161
Literatura	164
Załączniki	172
Pakiet <i>ModGraf</i> zawierający klasę do badania zależności funkcyjnych	173
Pakiet <i>Strukt</i> zawierający obiektowe struktury danych typu <i>Kolekcja</i>	182
Program obiektowy <i>Grafy</i> zawierający obiekt do badania zależności funkcyjnych	186
Pakiet <i>Dodatki</i> zawierający pomocnicze podprogramy	187
Program <i>Spójne_Składowe</i> wyznaczający liczbę spójnych składowych w grafie dwudzielnym	188

Podziękowanie

Serdecznie dziękuję swojemu Promotorowi,
Panu Profesorowi dr hab. inż. Adamowi Sielickiemu za cenne
uwagi merytoryczne, które pomogły mi w realizacji badań
związanych z pracą doktorską.

1.Wprowadzenie

Podejście obiektowe charakteryzuje się *syntetycznym* tworzeniem oprogramowania wynikającym ze stopniowego rozwiązywania problemu, w przeciwieństwie do podejścia strukturalnego, które opiera się na dekompozycji problemu.

Podejście obiektowe w informatyce wypracowane zostało w związku z rozwojem systemów informatycznych na gruncie teorii i praktyki. [12, 17, 19, 48, 50, 62, 63, 90, 94, 102, 114, 116, 131]. W pracach [19, 73, 90, 116, 139] podkreśla się rolę tego podejścia w zagadnieniach sieciowego obiegu informacji. W pracy [134] przedstawiono ścisły związek między architekturą komputerów zorientowaną obiektowo i oprogramowaniem obiektowym. Podejście obiektowe charakteryzuje się ujednoliceniem pojęć stosowanych na różnych poziomach realizacji problemów - przy sformułowaniu problemów, przy ich analizie i implementacji [38, 39, 40]. Techniki stosowane na różnych poziomach realizacji aplikacji, zintegrowane podejściem obiektowym, są określane jako techniki obiektowe.

W podejściu obiektowym analiza problemu jest wykonywana na etapie modelowania, niezależnie od implementacji. Model oprogramowania obiektowego składa się z modelu struktury obiektów, czyli statycznego modelu obiektowego, oraz modelu dynamiki zachowania obiektów. Na etapie implementacji główny model klas jest uzupełniany dodatkowymi klasami reprezentującymi pomocnicze struktury danych i tworzy się program wynikowy (wykonywalny). Decyzje podjęte na etapie budowy modelu obiektowego, ocenianego za pomocą charakterystyk wewnętrznych, mają nadrzędny wpływ na charakterystyki zewnętrzne programów wykonywalnych. Pojęcie syntezy programów obiektowych w aspekcie struktur danych odpowiada syntezie modeli struktur klas.

Przedmiotem niniejszej rozprawy jest badanie wpływu polimorfizmu na charakterystyki wewnętrzne oprogramowania i opracowanie metod syntezy efektywnych modeli struktury obiektów operujących na polimorficznych strukturach danych.

W niniejszej pracy nie rozpatruje się dynamicznego modelu zachowania obiektów. W wielu metodach obiektowych model dynamiczny jest tworzony dopiero po utworzeniu modelu struktury obiektów [44, 61, 65, 114, 117, 123, 124]. Analiza własności dynamicznych modelu zachowania obiektów oraz wpływ modelu struktury obiektów na jakość modelu dynamicznego mogą być rozpatrywane po realizacji modelu struktury obiektów. Jest to odrębny problem. Problemy przedstawione w tej pracy mogą więc być rozpatrywane niezależnie od problemów dynamiki obiektów.

1.1. Informacje ogólne

Praca dotyczy problemów związanych z tworzeniem efektywnego modelu obiektowego realizowanego za pomocą statycznego modelu konceptualnego ORM (Object Role Modelling) [15, 32, 34, 43, 54, 58, 59, 60, 64, 65, 78, 79, 80, 106, 107, 108, 121, 122, 135]. Wybrana metoda spełnia wymagania dotyczące tworzenia oprogramowania obiektowego, gdyż cechy tej metody spełniają postulaty stawiane metodom modelowania oprogramowania [4, 5, 8, 13, 20, 35, 51, 52, 66, 67, 71, 88, 101, 127, 137], a w szczególności obiektowego [10, 33, 38, 48, 56, 72, 74, 83, 87, 89, 98, 109, 110, 118].

Proces tworzenia oprogramowania obiektowego składa się z dwóch głównych etapów:

- rozwiązywania problemu niezależnie od implementacji, przedstawienia wyniku w postaci modelu oraz testowania,
- implementacji modelu za pośrednictwem projektowania, programowania i testowania.

Oba etapy są realizowane za pomocą różnych elementów. Oznacza to, że język opisujący model obiektowy, język projektowania i język programowania wprowadzają różne reprezentacje tych samych pojęć. Jednak stosowane metody są objęte wspólnym paradygmatem obiektowym. Stąd główny „szkielet” oprogramowania, utworzony na etapie modelowania, jest uzupełniany o pewne szczegóły na etapie implementacji.

Na rys. 1.1 przedstawiono główne cechy poszczególnych etapów tworzenia oprogramowania obiektowego. Podczas rozwiązywania problemu, niezależnie od implementacji tworzony jest konceptualny model oprogramowania, reprezentowany za pomocą specyfikacji obiektowej. Podczas implementacji (rys 1.1 i 1.2), obejmującej projektowanie i programowanie, jest tworzony program wykonywalny w oparciu o specyfikację modelu konceptualnego [38, 39, 40, 41, 62, 63, 65, 79, 81, 89, 100, 117, 120]. Konstrukcja modelu decyduje o konstrukcji programu wykonywalnego, w którym pojawiają się jedynie szczegóły wynikające ze środowiska, w którym działa program.

Modelowanie	Projektowanie	Programowanie
Co należy zrobić?	Jak należy zrobić?	
• model konceptualny aplikacji • <u>specyfikacja obiektowa modelu</u>	• architektura sprzętu i oprogramowania • dostęp użytkownika • przechowywanie danych	• specyfikacja obiektowa programu (deklaracja klas, definicja metod) • dodatkowe struktury danych (klasy pomocnicze, pliki)

Rys. 1.1 Proces tworzenia oprogramowania

Postulaty w modelowaniu	Postulaty w projektowaniu i programowaniu
• iteracyjna realizacja trzech etapów tworzenia oprogramowania	
• opis w języku naturalnym lub w postaci diagramów • łatwość zmian i rozwoju modelu • wieloużywalność elementów modelu • elementy modelu niezależne od implementacji	• właściwy podział zadań między sprzęt i oprogramowanie • optymalna kolejność następujących działań realizujących: podział na podsystemy, przechowywanie danych, sterowanie dostępem do zasobów systemu, system sterowania oprogramowaniem, reakcję systemu na warunki pracy, dostęp użytkownika • testowanie zgodności semantyki programu i modelu

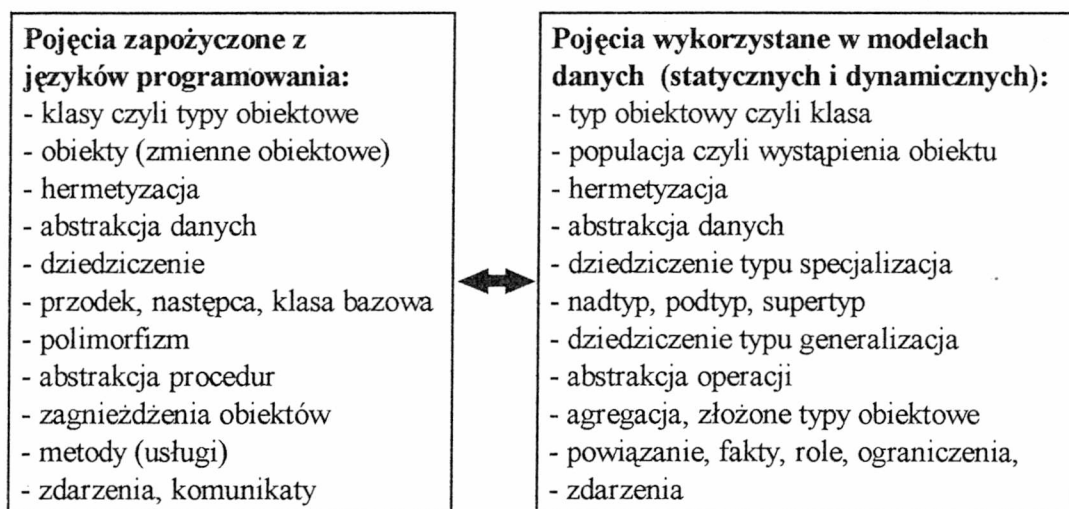
Rys. 1. 2. Postulaty spełniane w procesie tworzenia oprogramowania obiektowego

W praktyce postulat iteracyjnego tworzenia oprogramowania jest spełniany za pośrednictwem następujących czterech podejść [57]:

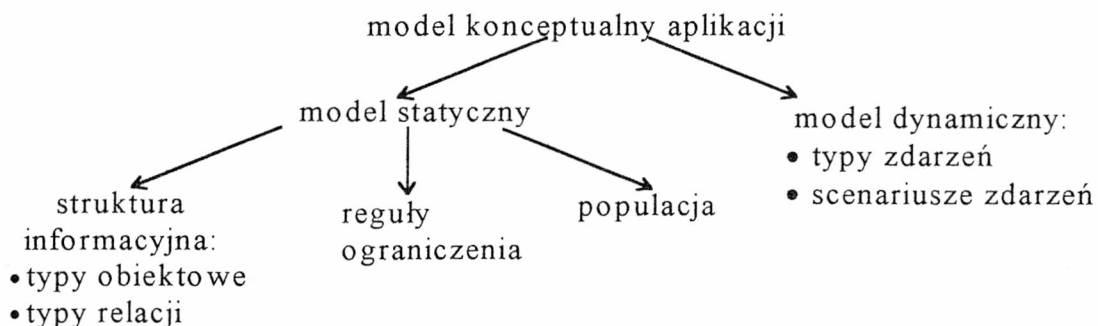
- typu *Spiral Model*, gdzie w kolejnej iteracji etapy tworzenia oprogramowania są realizowane w standardowej kolejności, począwszy od modelowania;
- typu *Fountain Model*, gdzie w kolejnej iteracji proces tworzenia oprogramowania jest realizowany od dowolnego etapu z zachowaniem ustalonej kolejności następnych etapów;
- typu *Pinball Model*, gdzie w kolejnej iteracji proces tworzenia oprogramowania jest realizowany od dowolnego etapu bez zachowania ustalonej kolejności następnych etapów;
- typu *SOMA Model*, gdzie wykorzystano podejście typu *Fountain Model*. Proces powstawania oprogramowania jest wspomagany tworzeniem stopniowo rozbudowywanych prototypów reprezentujących produkt każdego z etapów.

W toku procesu tworzenia oprogramowania operuje się pojęciami zapożyczonymi z systemów programowania i systemów modelowania danych (rys. 1.3), a model konceptualny aplikacji odgrywa nadrzędną rolę (rys. 1.4).

Model statyczny opisuje, jakie typy obiektowe istnieją, jakie wykonują operacje oraz w jaki sposób je wykonują, natomiast model dynamiczny zawiera dane o tym, kiedy i w jakiej kolejności mają być wykonane te operacje.



Rys. 1.3. Baza pojęciowa integrowana w technice obiektowej



Rys. 1.4. Role spełniane przez składowe modelu konceptualnego

Podstawą specyfikacji obiektowej oprogramowania - projektowanej aplikacji - jest

opis w terminach: typów obiektowych i typów relacji, typów zdarzeń, rodzaju, wagi i kolejności ograniczeń. Typy obiektowe oraz typy relacyjne tworzą strukturę informacyjną, która jest częścią syntaktyczną, natomiast reguły oraz ograniczenia i populacja, czyli wystąpienia obiektów, są częścią semantyczną modelu statycznego. Typy zdarzeń i scenariusze ich występowania opisują dynamikę modelu (rys. 1.4. wg [107]).

W rozdziale 1.2 i 1.3 przedstawiono dwa ważne obszary zagadnień związanych z tezą pracy i postawionymi celami. Jednym z nich jest obszar tworzenia oprogramowania obiektowego, a szczególnie etap modelowania (1.2), drugim jest obszar oceny oprogramowania (rozdz. 1.3).

W rozdziale 1.4 przedstawiono tezę i główne cele wyznaczone w tej rozprawie.

1.2. Metody tworzenia oprogramowania zorientowanego obiektowo

W pracach [12, 50, 61] zaprezentowano przegląd metod tworzenia oprogramowania zorientowanego obiektowo, gdzie wyróżniono trzy grupy metod: metody oparte na analizie struktur danych, metody oparte na analizie zdarzeń - zaliczane do metod starszej generacji oraz metody zaliczane do drugiej generacji, oparte na analizie scenariuszy. Metody te obejmują etap analizy i implementacji. Klasyfikacja metod wynika z różnych metodologii tworzenia oprogramowania obiektowego opartych na tym samym paradygmacie obiektowym, czyli: istnieniu obiektów i ich wzorców - klas, dziedziczeniu, generalizacji, hermetyzacji, abstrakcji operacji i danych.

Techniki obiektowe drugiej generacji są znacznie efektywniejsze w porównaniu z pierwszą generacją, która nie zawsze właściwie łączyła własności statyczne i dynamiczne modelu aplikacji. Było to spowodowane stosowaniem technik modelowania wywodzących się z podejścia strukturalnego. Do tych najstarszych metod należą między innymi pierwsze wersje: OMT[114], Object-Oriented Design with Applications [17], OOA [38], OOA&A [94], Objectory [12, 61]. Bardziej elastyczne podejście reprezentują metody drugiej generacji, często będące ulepszeniem pierwszej wersji. Na szczególną uwagę zasługują metody: Objectory i jej ulepszona wersja [61] oraz system CORBA [19, 90, 116, 139].

Główne różnice w metodach tworzenia oprogramowania obiektowego są uwidocznione na poziomie modelowania, realizowanego niezależnie od implementacji. Metoda OMT oraz OOA na etapie modelowania łączy trzy modele: *obiektów*, *funkcjonalny* oraz *dynamiczny*. W metodzie Object-Oriented Design with Applications stosowane są *statyczne modele: klas*, *obiektów* oraz *dynamiczne modele: przekształceń stanów*, *interakcji*.

Metoda Objectory składa się z trzech faz: *analizy*, *konstruowania* i *testowania*. W fazie analizy specyfikuje się *model wymagań* i *model analizy*, natomiast w fazie *konstrukcji* *model projektowy* oraz *model implementacji*. *Model wymagań* składa się z trzech modeli: *use case*, *opisu interakcji* oraz *modelu dziedziny problemu*. Model *use case* zawiera elementy typu *actor* oraz *wymagania* czyli wyszczególnione sekwencje operacji zwane *use cases*. Model *use case* stanowi model środowiska, natomiast użytkownikami są wystąpienia aktorów (*actors*). Każda z sekwencji *use cases* odpowiada opisowi klasy realizującej te operacje oraz opisowi klasy żądającej wykonania tych operacji (czyli opis aktora) i jest prezentowana w postaci *opisu interakcji*. *Model dziedziny*

problemu klasyfikuje i przedstawia opis elementów świata rzeczywistego w postaci: klas, ich atrybutów, powiązań statycznych i dynamicznych, związków dziedziczenia. *Model analizy* powstaje na podstawie *modelu wymagań* i przedstawia schemat konceptualny systemu. Struktura informacyjna tego schematu składa się z trzech typów obiektów: encji, obiektów kontrolnych i interfejsowych. Na schemacie uwidacznia się własności funkcjonalne w postaci następujących relacji: dziedziczenia, związków statycznych, agregacji, atrybutowych, komunikacyjnych. Na etapie konstruowania tworzony jest *diagram interakcji* realizujący sekwencje *use case*. Następnie na podstawie diagramu interakcji jest tworzony *model zachowania obiektów*. *Model zachowania* jest przedstawiony w postaci *grafu stanów tranzycji* i określa, które żądanie zostało otrzymane i jak zostanie zrealizowane w danym stanie. Na podstawie modelu zachowania tworzy się *model komunikacji*.

Metoda *Data Model Objectory* jest rozszerzeniem techniki Objectory. W tej metodzie głównie rozbudowano model dynamiki. Wprowadzono modelowanie zachowania niedeterministycznego wraz z konstrukcjami decyzyjnymi, polegającymi na arbitralnym wyborze pewnych możliwości działań. Wprowadzono modelowanie wewnętrznych działań współbieżnych oraz uzupełniono modelowanie współbieżnych działań zewnętrznych. W metodzie dopracowano formalnie poszczególne modele, nieco inaczej rozkładając ich kompetencje. Wprowadzono *model danych*, zamiast *modelu analizy*, który składa się z *modelu typów* wraz z powiązaniami (podobnie jak w modelu analizy) oraz *modelu stanu*. *Model stanów* powstaje przez homomorficzne odwzorowanie grafu typów i reprezentuje semantykę modelu. Graf ten może posiadać dodatkowe gałęzie reprezentujące relacje dziedziczenia. *Model danych* przedstawia stan systemu: populacje klas, wartości atrybutów, relacje między wystąpieniami obiektów. *Model zachowania* jest ściśle powiązany z grafem stanu i przekształca wartości stanu. *Model zachowania* jest bazą do utworzenia *modelu komunikacji*. Semantyka *modelu komunikacji* jest opisana na drodze translacji *modelu zachowania* do algebry procesu. Algebra procesu charakteryzuje się siłą formalnego wyrażania i dzięki temu doskonale nadaje się do opisu *modelu komunikacji*. *Model zachowania* składa się ze zbioru zachowań obiektów, które są połączeniem zbiorów: *stanów*, *zadań*, *elementarnych komunikacji*, *decyzji*, *stanów poprzednich*, *kreatorów obiektów* oraz *zakończeń obiektów* (konstruktory i destruktory). *Model zachowania* umożliwia również modelowanie własności zachowań współbieżnych. Model ten jest przedstawiany w postaci diagramu przedstawiającego wszystkie elementy modelu. Zbiory elementarnych komunikacji dla komunikacji synchronicznej są wiadomościami, natomiast asynchronicznej są sygnałami. Wiadomości lub komunikaty są usługami żadanymi przez specjalny obiekt lub dowolny obiekt.

Technika obiektowa CORBA (Common Object Request Broker Architecture) jest systemem opracowanym przez firmę OMG (Object Management Group) do zastosowań sieciowych, opracowywanych na skalę przemysłową. System ten został oparty na architekturze OMA (Object Management Architecture), składającym się z *modelu obiektowego* oraz *modelu odwołań*. *Model obiektowy* definiuje, w jaki sposób obiekty są rozproszone w niejednorodnym środowisku, natomiast *model odwołań* charakteryzuje interakcje między tymi obiektami. W *modelu obiektowym* OMA, w celu zachowania enkapsulacji wszystkie usługi są udostępniane za pośrednictwem interfejsów. Model odwołań OMA składa się z części *komunikacyjnej ORB* (Object Request Broker) i służy do ułatwienia komunikacji między klientami i obiektami. Jest to kluczowy element architektury. Oprócz tego wyodrębniono cztery typy interfejsów umożliwiających taką komunikację:

- *usługi obiektowe (object services)*: interfejs niezależny od dziedziny aplikacji, używany przez wiele rozproszonych programów obiektowych i udostępniający następujące usługi: *usługę nazwową*, która umożliwia poszukanie obiektu wg nazwy oraz *usługę handlową*, umożliwiającą wyszukanie obiektu wg jego właściwości.
- *ułatwienia (common facilities)*: interfejsy zorientowane na użytkownika końcowego,
- *interfejsy dziedziny (domain interfaces)*: interfejsy zorientowane na specyfikację dziedziny problemu,
- *interfejsy aplikacji (application interfaces)*: interfejsy przystosowane dla danej aplikacji, lecz nie są objęte standardem.

Oparcie architektury systemu na komunikacji za pośrednictwem ORB i dalsza specjalizacja interfejsów obsługujących aplikacje pracujące w rozproszonym środowisku doprowadziła do powstania architektury CORBA. Podstawą architektury tego systemu są:

- *rdzeń ORB (ORB Core)*,
- *język definiowania interfejsów (OMG Interface Definition Language-OMG IDL)*,
- *interfejs magazynowy (Interface Repository)*,
- *język odwzorowań (Language Mappings)*,
- *dziury i szkielety (Stubs and Skeletons)*,
- *dynamiczne wezwania i wysyłki (Dynamic Invocation and Dispatch)*,
- *adaptery obiektowe (Object Adapter)*,
- *protokoły ORB (Inter-ORB Protocols)*.

Podstawowymi elementami architektury CORBA, opartych na tych elementach, są:

- *implementacja obiektów (Object Implementation)*: definiuje się operacje, które implementują interfejs CORBA IDL za pomocą takich języków jak C, C++, Java, Smalltalk, Ada;
- *klient (Client)*: jest programową encją (reprezentującą rolę serwera lub klienta) wywołującą operacje z *implementacji obiektów*;
- *Object Request Broker*: jest realizacją komunikacji między *klientem* i *implementacją obiektów*. Ułatwia się programowanie rozproszone za pomocą odizolowania klienta od szczegółów wywoływania metod, co daje taką postać żądań klienta jaką są wywoływane procedury lokalne. ORB poszukuje implementacji obiektu i zwraca jego odpowiedź;
- *interfejs ORB (ORB Interface)*: ORB jest logicznym *istnieniem*, które może być implementowane w dowolny sposób (jeden lub kilka procesów lub zbiór bibliotek). Aby uniezależnić aplikację od implementacyjnych szczegółów, zdefiniowano abstrakcyjny interfejs dla ORB. Ten interfejs posiada wiele usługowych funkcji takich jak konwersje danych, tworzenie listy argumentów żądań realizowanych przez dynamiczny interfejs wezwań;
- *IDL stubs and skeletons*: stanowi *klej* między aplikacjami klienta i serwera a ORB. Transformacja między definicjami IDL i docelowym językiem programowania jest zautomatyzowana przez kompilator IDL. Wykorzystanie kompilatora redukuje potencjalne wszelkie niekonsekwencje w realizacji klienta i serwera.
- *interfejs dynamicznych wezwań (Dynamic Invocation Interface DII)*: pozwala klientowi bezpośrednio wykorzystywać mechanizm żądań realizowanych przez ORB. Aplikacje stosujące DII do dynamicznych żądań skierowanych do obiektów bez wywoływania interfejsu IDL, są połączone z obiektami za pomocą specyficznych *dziur DII*. Podobnie jak *dziury IDL*, w *DII dziury* pozwalają klientom na odseparowanie

operacji nadawania i odbioru lub tylko nadawania.

- *dynamiczny szkieletowy interfejs (Dynamic Skeleton Interface DSI)*: pełni rolę podobną do DII dla klienta. DSI pozwala ORB uwolnić żądanie do implementacji obiektów, które jeszcze nie zostały rozpoznane w czasie tłumaczenia. Implementacja wykorzystuje albo specyficzne typy *szkieletów IDL*, albo własne *szkielety dynamiczne*;
- *obiektywny adapter (Object Adapter)*: pomaga ORB w uwalnianiu żądań do obiektów i w aktywowaniu obiektów. Adapter wiąże implementację obiektów z ORB. Obiektowe adaptory mogą specjalizować się dla ustalonych stylów implementacji obiektów.

System CORBA stanowi etap przejściowy między obiektową techniką tworzenia oprogramowania na skalę przemysłową a systemami wieloagentowymi [102].

Opisane metody charakteryzują podobne podejście do sposobu wyodrębnienia klas i obiektów czyli tworzenia diagramów relacyjnych danych wywodzących się z metody ER (Entity-Relationship) podane przez Chena [36]. Metoda ER [5, 8, 21, 47, 70, 75, 95, 96, 97, 106, 111, 112, 115, 120] jest podstawą do tworzenia statycznego modelu konceptualnego, który może być odwzorowywany w logiczne struktury danych takie jak: model relacyjny, sieciowy, hierarchiczny lub obiektowy. Metoda ER ulega ciągłemu rozwojowi [3, 21, 47] i w przypadku metod obiektowych jest w pewien charakterystyczny sposób przystosowywana do każdej z nich [85, 114, 112]. Jednak metoda ER największą rolę odgrywa przy tworzeniu oprogramowania zorientowanego obiektowo *metodami opartymi na analizie struktur danych*.

Metoda ORM [15, 34, 43, 54, 58, 59, 60, 64, 65, 78, 79, 80, 106, 107, 108, 121, 122, 135] wywodzi się z metody ER [32] i pozwala na szczegółową analizę wszystkich zależności funkcyjnych i niefunkcyjnych występujących w problemie [15]. Umożliwia wykorzystanie formalizmów i doświadczeń wypracowanych przy tworzeniu logicznego modelu relacyjnego do realizacji modelu struktur obiektów w metodach tworzenia oprogramowania obiektowego. Jednocześnie metoda ORM ułatwia proces projektowania klas, gdyż można wyodrębnić całą strukturę klasy wraz z atrybutami oraz określić związki, w jakich biorą udział poszczególne atrybuty. Takie podejście zostało zaprezentowane w [43]. Przykładem systemu typu CASE (Computer Aided System Engineering), wykorzystującego metodę ORM do tworzenia modelu struktur klas jako elementu pełnego obiektowego modelu konceptualnego aplikacji jest system OSA [65].

1.3. Ocena oprogramowania obiektowego

W niniejszej rozprawie opracowano metody badania charakterystyk wewnętrznych diagramu konceptualnego, określających złożoność modelu [1, 55, 57, 93, 112, 132, 133]. Charakterystykami wewnętrznymi modelu są: rozmiar, złożoność struktur danych, złożoność logiczna, spójność wewnętrzna, spójność semantyczna oraz złożoność połączeniowa. Pomiar tych charakterystyk odbywa się za pośrednictwem pewnych metryk wiążących obiektywnie zmierzone elementy modelu [68, 69]. W niniejszej pracy do oceny modelu wybrano charakterystykę spójności wewnętrznej oraz złożoności połączeniowej [22, 23, 26, 24, 37, 46]. Ocena modelu jest dokonywana za pośrednictwem metryki spójności i metryki połączeń. Pomiar wartości tych metryk umożliwia wybór najlepszego rozwiązania ze względu na najmniejszą liczbę typów powiązań oraz największą spójność, wynikającą z dziedziczenia i polimorfizmu oraz ograniczeń zbiorowych.

Oprogramowanie obiektowe można oceniać na podstawie charakterystyk zewnętrznych. W pracach [6, 7, 11, 14, 25, 27, 29, 30, 31, 37, 46, 49, 57, 68, 77, 82, 84, 119, 132, 133] przedstawiono różne definicje charakterystyk zewnętrznych oprogramowania, np.:

- a) zrozumiałość, modyfikowalność, testowalność;
- b) jakość: niezawodność, dostępność, konserwowalność (zrozumiałość, modyfikowalność, testowalność), użyteczność;
- c) jakość: stopień osiągniętej abstrakcji, wieloużywalność, zrozumiałość, konserwowalność, testowalność; koszt wytworzenia oprogramowania, funkcjonalność.

W niniejszej pracy przyjęto definicję charakterystyk zewnętrznych podaną w punkcie c. Możliwość oceny oprogramowania na podstawie oceny modelu struktur obiektów wynika z faktu, iż można decydować o charakterystykach zewnętrznych oprogramowania na etapie budowy modelu klas. Taka zależność między charakterystykami zewnętrznymi i wewnętrznymi wynika z prostych odwzorowań modelu w dziedzinę reprezentacji struktur danych na etapie implementacji czyli projektowania i programowania. W niniejszej pracy przyjęto metryki spójności i połączeń do oceny charakterystyk wewnętrznych, co dzięki korelacji pozwala ocenić stopień osiągniętej abstrakcji (zastosowanie polimorficznych struktur danych), wieloużywalność (rozmiar drzew polimorficznych i specjalizacji), zrozumiałość, konserwowalność i testowalność. W niniejszej pracy założono, że znany jest związek jakościowy, który określa, że poprawa charakterystyk wewnętrznych prowadzi do poprawy charakterystyk zewnętrznych.

Problematyka oceny oprogramowania stanowi obecnie kluczowe zagadnienie w inżynierii oprogramowania. Powstały specjalne grupy zajmujące się jakością oprogramowania obiektowego np.: ESEG (Experimental Software Engineering Group), ESERG (Empirical Software Engineering Research Group at Bournemouth University). W pracach cytowanych w tym rozdziale zaprezentowano obszar zagadnień związanych z działalnością tych grup. Większą część prac poświęcono ocenie oprogramowania obiektowego na etapie projektowania i programowania. Proponuje się różne zestawy charakterystyk zewnętrznych i wewnętrznych. Wynika to z faktu, że bardzo trudno określić związki jakościowe, a tym bardziej ilościowe między tymi charakterystykami. Eliminuje się wszystkie charakterystyki zewnętrzne, które są powiązane funkcyjnie. Dąży się do uzyskania ograniczonego zbioru metryk, które jednocześnie są powiązane z możliwie wieloma charakterystykami zewnętrznymi.

Głównym zagadnieniem w ocenie oprogramowania jest określenie, co należy mierzyć na poszczególnych etapach powstawania oprogramowania, aby w obiektywny sposób ocenić jakość produktu końcowego.

W niniejszej pracy oprogramowanie badano na etapie konstruowania modelu conceptualnego. Przyjęto założenie, iż tym lepszy model a więc i jakość oprogramowania, im mniejszą zawiera liczbę połączeń i charakteryzuje się większą spójnością. Zdefiniowano te pojęcia w pracy jako liczbę powiązań między typami obiektowymi, natomiast spójność jako wyrażenie wiążące liczbę ról (lub predykatów związków między klasami) występujących w modelu, wynikającą z dziedziczenia, polimorfizmu oraz ograniczeń. Definicje występujące w tej pracy, zostały podane przez autora na podstawie pewnych analogii, jakie występują pomiędzy definicjami operującymi na budowie klasy (atrybuty i metody) a budową klasy w modelu conceptualnym. Przyjęto bowiem, że powiązania między typami obiektowymi reprezentują minimalny zbiór metod realizujących te powiązania i stanowią szkielet komunikacji między wystąpieniami tych klas (populacji, obiektów).

Jednocześnie można w modelu uwzględnić wszystkie typy atrybutowe (atrybuty) związane z poszczególnymi typami obiektowymi (typami abstrakcyjnymi) i wyznaczyć spójność modelu.

Jednym z wyników osiągniętych w pracy jest zdefiniowanie metody oceny modeli struktur obiektów.

1.4. Główne cele i teza pracy

Teza pracy:

Zastosowanie polimorficznej struktury danych w modelach oprogramowania obiektowego prowadzi do poprawy jego charakterystyk zewnętrznych.

W niniejszej pracy należało opracować podstawy formalne pewnych narzędzi, niezbędnych do udowodnienia postawionej tezy, co określiło wyznaczenie celów. W związku z rozpatrywaniem struktur danych na poziomie modelu struktur obiektów należało ujednolicić definicję polimorfizmu stosowanego w modelowaniu z definicją tego mechanizmu stosowanego w językach programowania. Założono, że ocena wpływu polimorficznych struktur danych na charakterystyki zewnętrzne oprogramowania jest możliwa jedynie w modelach równoważnych. Należało więc określić kryteria równoważności modeli ORM oraz zdefiniować formalnie i opracować metodę badania równoważności modeli. Teza pracy wymagała określenia w sposób obiektywny miary charakterystyk zewnętrznych oprogramowania badanego za pomocą jego modelu struktur klas. Wymagało to zdefiniowania miar wewnętrznych oprogramowania wyznaczanych w sposób obiektywny za pomocą elementów modelu struktur klas. Badanie charakterystyk wewnętrznych równoważnych modeli jest problemem wielokryterialnego wyboru. Stąd należało przyjąć kryterium efektywności modeli w celu wyeliminowania rozwiązań nieefektywnych.

Postawione cele:

1. Sformułowanie podstaw formalnych wzorca polimorfizmu, czyli jego syntaktykę i semantykę, na poziomie modelowania ORM odpowiadającego jego implementacji na poziomie języków programowania obiektowego.
2. Opracowanie reguł specjalizacji w języku teorii mnogości.
3. Sformułowanie podstaw formalnych i algorytmu do badania równoważności modeli konceptualnych ORM.
4. Opracowanie podstaw formalnych i konstrukcja algorytmu klastrowego do budowy modeli struktur klas na podstawie modeli ORM.
5. Sformułowanie podstaw formalnych i algorytmu do pomiaru złożoności strukturalnej reprezentowanej przez pomiary spójności i liczby połączeń modelu konceptualnego ORM i jego postaci klastrowej.
6. Przyjęcie kryterium wyboru rozwiązań efektywnych i opracowanie metody wyboru zbioru rozwiązań efektywnych wg przyjętego kryterium.
7. Opracowanie algorytmu tworzenia efektywnych modeli ORM, wg przyjętego kryterium, opartych na polimorficznych strukturach danych.

Wszystkie elementy oznaczone w tekście za pomocą {*} reprezentują opis formalny wyników osiągniętych w pracy.

2. Schemat konceptualny ORM

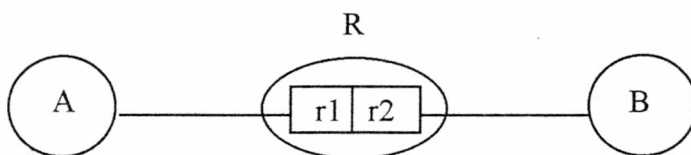
Wybrano metodę modelowania ORM (*Object Role Modelling*), ponieważ umożliwia ona wyodrębnienie struktury wszystkich klas występujących w aplikacji. Metoda przedstawia świat rzeczywisty jako kolekcję obiektów, które odgrywają swoje role i są uwidocznione w sformalizowany sposób na diagramie schematu konceptualnego ORM [15, 34, 43, 54, 58, 59, 60, 64, 65, 78, 79, 80, 106, 107, 108, 121, 122, 135].

2.1. Definicje własności syntaktycznych

Opis metody jest oparty na pracach [15, 54, 59, 60, 135]. W niniejszej pracy zmodyfikowano istniejący mechanizm prezentowania polimorfizmu. Zmiana polega na uwzględnieniu w relacji polimorfizmu dziedziczenia typu specjalizacja. Modyfikacja ta jest podyktowana potrzebą tworzenia takich modeli, które w prosty sposób są odwzorowywane w dziedzinę implementacji zorientowaną obiektowo. W językach programowania polimorfizm jest rozszerzeniem specjalizacji (dziedziczenia „w dół”). Taki mechanizm formalnie zdefiniowano w modelu konceptualnym ORM.

Zastosowane definicje, mające charakter definicji aksjomatycznych, są podane na podstawie [15, 54, 59, 60]. Uzupełnienia zaznaczono gwiazdką. Również zmodyfikowano pewne definicje, dążąc do uzyskania takiego narzędzia, które najlepiej wyraża semantykę rozwiązywanego problemu a jednocześnie może być w prosty sposób odwzorowane w specyfikację obiektową modelu.

Podstawowym elementem opisu jest typ relacji oraz typ obiektowy. Termin relacja kojarzy się z klasycznym opisem matematycznym, określającym relację jako podzbiór produktu kartezjańskiego [86, 130]. Inne podejście do opisu relacji polega na zastosowaniu mechanizmu odwzorowania, umożliwiającego określanie własności połączeń między elementami relacji [15, 91, 127]. Na rys. 2.1 pokazano typ relacji R w taki sam sposób jak typy obiektowe A i B . Typ relacji stanowi połączenie między rolami r_1 i r_2 , jakie obiekty A i B odgrywają w relacji R . Typ relacji R jest nazwany typem faktu.

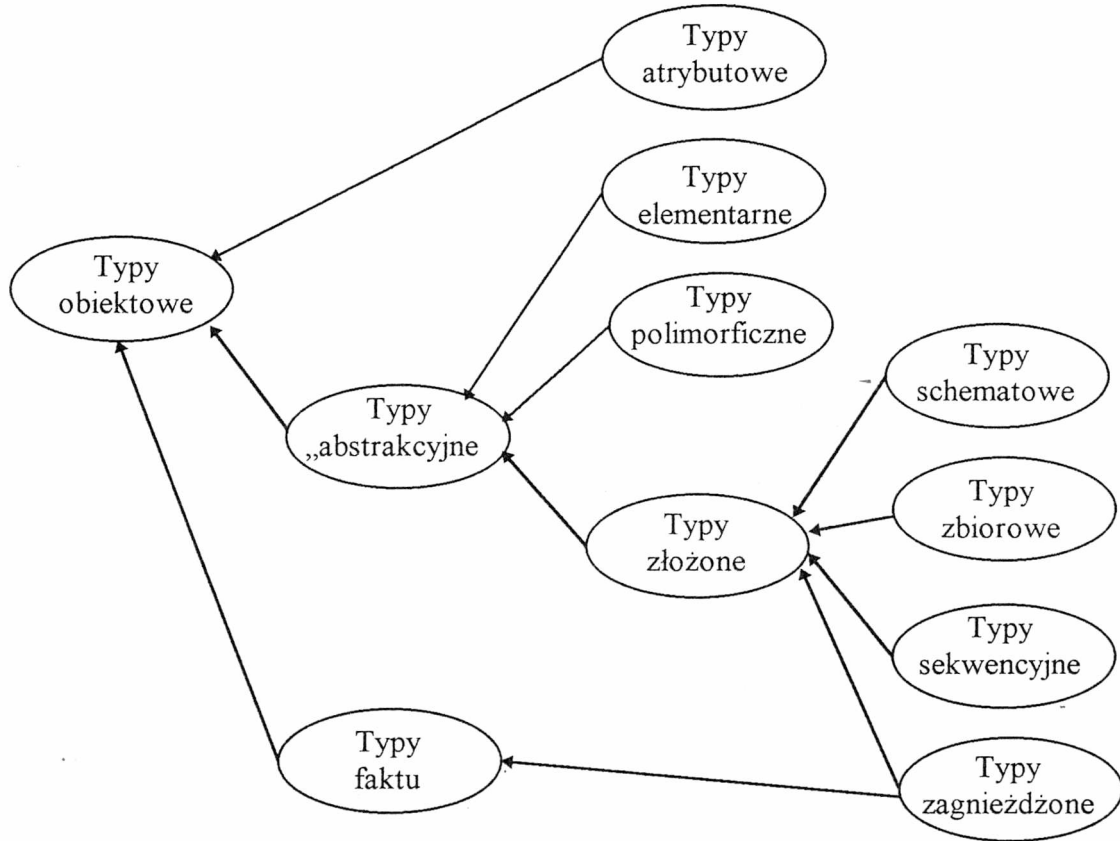


Rys. 2.1. Graficzna reprezentacja typów: relacyjnego (R) i obiektowych (A , B) w modelowaniu ORM

Struktura informacyjna jest zdefiniowana jako

$SI = (TOb, TWart, TFakt, TRol, Fakt, Baza, Spec, Pol, Korzen, Elt, Schem)$ (2.1)
wraz z definicjami aksjomatycznymi określającymi elementy struktury informacyjnej.

Na rys. 2.2 zaprezentowano diagram różnych typów obiektowych stosowanych do budowy struktury informacyjnej. Krawędzie łączące poszczególne typy obiektowe określają powiązania między typami obiektowymi. Te powiązania są wyszczególniane przy opisie każdego z typów obiektowych, podanym w dalszej części rozdziału 2.1.



Rys. 2.2. Diagram typów struktury informacyjnej modelu

- $TWart$ - typ atrybutowy reprezentowany przez konkretne wartości (liczby, łańcuchy itp),
- TE - typ obiektowy elementarny, gdzie $TWart \cap TE = \emptyset$,
- TZb - typ obiektowy zbiorowy,
- $TZag$ - typy obiektowy zagnieżdżony,
- $TSekw$ - typ obiektowy sekwencja,
- $TSch$ - typ obiektowy schematowy,
- $TZl = TSch \cup TZb \cup TZag \cup TSkw$; $TZl \cap TWart = \emptyset$ - typy obiektowe złożone,
- $TZl \cup TE = TA$ - typy abstrakcyjne, bez konkretnych wartości,
- $TPol \subset TA$ - typy polimorficzne, {*}
- $TRol$ - skończony zbiór ról,
- $TFakt$ - zbiór wynikający z podziału zbioru $TRol$. Każdy element zbioru $TFakt$ nazywa się typem faktu i spełnia zależność $TWart \cap TFakt = \emptyset$. Funkcja

$$Fakt : TRol \rightarrow TFakt$$

określa typ faktu w zależności od roli, która należy do niego i jest zdefiniowana jako

$$Fakt(p) = f \Leftrightarrow p \in f. \quad (2.2)$$

Typy faktów spełniają zależność $TFakt \supset TZag$.

- $TOb = TA \cup TPol \cup TWart \cup TFakt$ - typy obiektowe (zbiór niepusty),
- funkcja określa typ obiektowy pełniący daną rolę,

$$Baza: TRol \rightarrow TOB, \quad (2.3)$$

- funkcja wyznaczająca wszystkie typy obiektowe pełniące role należące do danego zbioru ról

$$Bazy: \varphi(TRol) \rightarrow \varphi(TOb),$$

$$Bazy(p) = \{Baza(r) \mid r \in p\}, \quad (2.4)$$

- funkcja określająca elementy typu zbiorowego i typu sekwencyjnego

$$Elt: TZb \cup TSch \rightarrow TOB, \quad (2.5)$$

- relacja opisuje dekompozycję typu schematowego

$$Schem \subseteq TSch \times TOB, \quad (2.6)$$

- relacja częściowego porządku określająca dziedziczenie typu specjalizacja („w dół”)

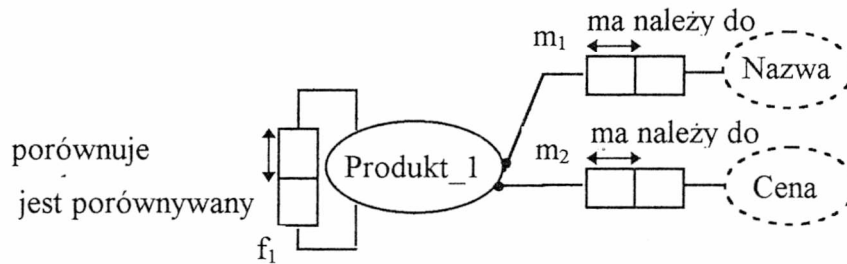
$$Spec \subseteq TA \times TA, \quad (2.7)$$

- relacja częściowego porządku określająca dziedziczenie uogólniające („do góry”)

$$Pol \subseteq TA \times TA, \quad (2.8)$$

- funkcja wyznaczająca praprzodka drzewa dziedziczenia

$$Korzen: TA \rightarrow TA, \quad (2.9)$$



Rys. 2.3. Diagram ORM

Zbiór aksjomatów opisujących własności struktury informacyjnej są systematycznie przedstawiane w kolejnych częściach rozdziału 2.1.1. Przyjęto taki sposób przedstawiania, gdyż te aksjomaty stanowią podstawową definicję własności różnych elementów modeli. Nazwy kolejnych aksjomatów składają się z litery **A** i numeru aksjomatu. Ułatwia to odwoływanie się do nich w różnych miejscach niniejszego tekstu.

Aksjomat A1 opisuje zależności pomiędzy typami obiektowymi.

A1 (separacja) -Typy *TWart*, *TFakt*, *TSekw*, *TSch*, *TZb* i *TE* tworzą podział zbioru *TOB*.

2.1.1. Podstawowe typy obiektowe

a) Typy abstrakcyjne i konkretne

W modelowaniu ORM typy obiektowe atrybutowe *TWart* są reprezentowane bezpośrednio przez konkretne wartości (liczby, łańcuchy itp). Pozostałe typy obiektowe są typami abstrakcyjnymi. Zbiór typów atrybutowych *TWart* jest rozłączny ze zbiorem ty-

pów abstrakcyjnych (2.1)

$$TWart \cap (TFakt \cup TSekw \cup TSch \cup TZag \cup TZb \cup TPol \cup TE) = \emptyset. \quad (2.10)$$

Na przykładowym diagramie ORM (rys. 2.3) przedstawiono typy abstrakcyjne za pomocą elips narysowanych linią ciągłą (*Produkt_1*), natomiast typy atrybutowe za pomocą elips narysowanych linią przerywaną (*Nazwa* jako wartość łańcuchowa, *Cena* jako wartość liczbowa). Pozostałe oznaczenia będą przedstawiane w toku wprowadzania definicji nowych pojęć.

b) Typy faktów

Głównym mechanizmem modelowania konceptualnego jest przedstawianie rozwiązania problemu w postaci struktury powiązań między typami obiektowymi. Te powiązania są typami relacji zwane typami faktów. Typ faktu stanowi połączenie między rolami jakie typy obiektowe odgrywają w relacji (rys. 2.1). Typ faktu może być traktowany jako typ obiektowy i wtedy może być powiązany rolą z innym typem faktu (typy zagnieżdżone *TZag*). Ponieważ każdy typ faktu traktowany jest jako zbiór ról, wprowadza się pojęcie funkcji

$$Inne_Role: \varphi(TRol) \rightarrow \varphi(TRol), \quad \{*\}$$

która wyznacza zbiór pozostałych ról należących do danego typu faktu na podstawie danej roli p (2.2)

$$Inne_Role(p) = \{x \mid x \in Fakt(p) \wedge x \neq p\}. \quad \{*\} (2.11)$$

Zbiór ról, posiadających ten sam typ bazowy wyznacza funkcja

$$Role_Bazy: \varphi(TOB) \rightarrow \varphi(TRol), \quad \{*\}$$

czyli

$$Role_Bazy(x) = \{y \mid Baza(y) = x\}. \quad \{*\} (2.12)$$

Binarna relacja jako typ faktu wiążący typ atrybutowy i typ abstrakcyjny nazywa się mostem. Term *Most (f)* klasyfikuje typ faktu jako typ mostu, jeśli spełnione jest wyrażenie (2.3, 2.1)

$$\exists_{p,q} [f = \{p,q\} \wedge Baza(p) \in TWart \wedge Baza(q) \notin TWart]. \quad (2.13)$$

Aksjomat A2 określa separację między typami atrybutowymi i abstrakcyjnymi (2.3, 2.2)

A2- $Baza(p) \in TWart \Rightarrow Most(Fakt(p))$.

Oznacza to, że jeśli istnieje most $m = \{p,q\}$, to za pomocą operatorów *konkr* i *abstr* można wyodrębnić rolę obiektu atrybutowego jako (2.3, 2.1)

$$konkr(m) \in m \wedge Baza(konkr(m)) \in TWart \quad (2.14)$$

i rolę obiektu abstrakcyjnego jako (2.3, 2.1)

$$abstr(m) \in m \wedge Baza(abstr(m)) \notin TWart. \quad (2.15)$$

Nazwy ról występujących w tym samym typie faktu muszą się różnić, natomiast w innych typach faktów mogą się powtórzyć.

Na diagramie ORM z rys. 2.3 typ faktu jest przedstawiany w postaci zbioru prostokątów, każdy z nich odpowiada roli należącej do typu faktu. Typy faktów $m1$, $m2$ są mostami, natomiast $f1$ jest typem faktu wiążącym role (*porównuje atrybuty, jest porównywany*) dwóch wystąpień typów obiektowych *Produkt_1*.

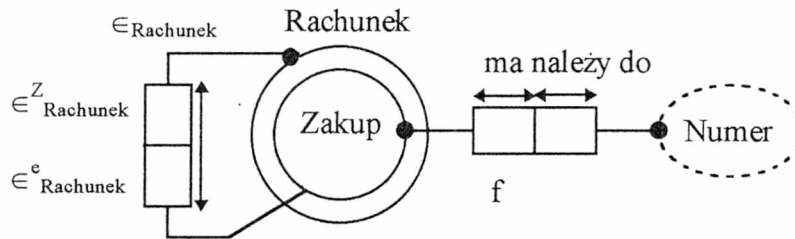
Z diagramu na rys. 2.3 mamy

$abstr(m1)=ma$,
 $konkr(m1)=należy\ do$.

c) Typy zbiorowe

Typ zbiorowy TZb jest odpowiednikiem pojęcia zbioru z teorii mnogości. Wystąpienie typu zbiorowego jest zbiorem wystąpień jego elementów danego typu obiektowego. Podobnie jak w teorii mnogości zbiór jest identyfikowany przez swoje elementy [86].

Przykładem typu zbiorowego jest diagram przedstawiony na rys. 2.4 reprezentujący typ zbiorowy *Rachunek*, zawierający abstrakcyjne elementy typu *Zakup* identyfikowane przez unikatową wartość typu atrybutowego *Numer*.



Rys. 2.4. Diagram ORM typu zbiorowego *Rachunek*

Typ faktu wiążący typ zbiorowy *Rachunek* składa się z dwóch ról

$$\in_{Rachunek} = \{ \in^Z_{Rachunek}, \in^e_{Rachunek} \}, \quad (2.16)$$

gdzie (2.3)

$$Baza(\in^Z_{Rachunek}) = Rachunek$$

oraz (2.3, 2.5)

$$Baza(\in^e_{Rachunek}) = Elt(Rachunek) \equiv Zakup.$$

Na diagramach nie uwidacznia się tej relacji, gdyż zakłada się, że jest ona domyślna. Separację typu abstrakcyjnego zbiorowego i jego elementów od typów atrybutowych zawiera aksjomat A3.

A3- $Elt(x) \notin TWart$.

d) Typy sekwencyjne

W przeciwieństwie do typu zbiorowego typ sekwencyjny $TSekw$ indeksuje swoje elementy. Wystąpienie typu sekwencyjnego jest sekwencją wystąpień elementów danego typu obiektowego. Podobnie jak typy zbiorowe sekwencja jest identyfikowana przez swoje elementy.

Diagram reprezentujący typ sekwencyjny *Rachunek* jest przedstawiony na rys. 2.5. Typ sekwencyjny *Rachunek* zawiera abstrakcyjne elementy typu *Zakup* identyfikowane przez unikatową wartość typu atrybutowego *Numer*.

Typ faktu wiążący typ sekwencyjny *Rachunek* składa się z dwóch ról

$$\in_{Rachunek} = \{\in^S_{Rachunek}, \in^e_{Rachunek}\}, \quad (2.17)$$

gdzie (2.3, 2.5)

$$Baza(\in^e_{Rachunek}) = Elt(Rachunek) \equiv Zakup$$

oraz (2.3)

$$Baza(\in^S_{Rachunek}) = Rachunek.$$

Typ faktu indeksujący elementy typu sekwencyjnego jest wyrażony przez

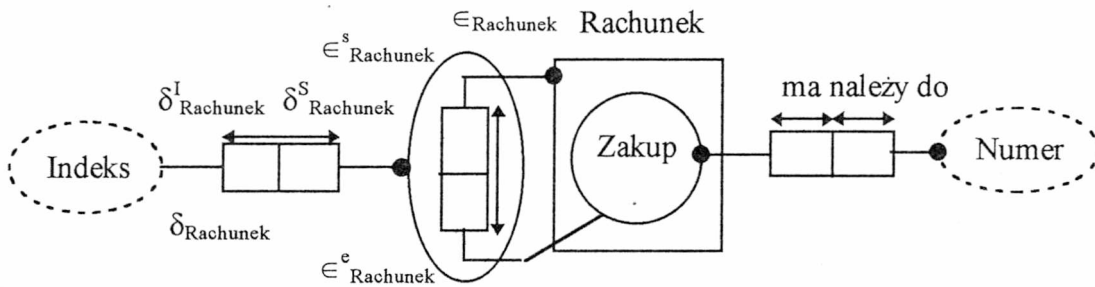
$$\delta_{Rachunek} = \{\delta^S_{Rachunek}, \delta^I_{Rachunek}\},$$

gdzie (2.3)

$$Baza(\delta^S_{Rachunek}) = \in_{Rachunek}$$

oraz (2.3)

$$Baza(\delta^I_{Rachunek}) = Indeks.$$



Rys. 2.5. Diagram ORM typu sekwencyjnego

Na diagramach nie uwidacznia się tej relacji, gdyż zakłada się, że jest ona domyślna. Separację typu abstrakcyjnego sekwencyjnego i jego elementów od typów atrybutowych zawiera aksjomat A3.

e) Typy zagnieżdżone

Zagnieżdżony typ obiektowy *TZag* o nazwie $\in_{Rachunek}$ został przekształcony z typu faktu $\in_{Rachunek}$. Stanowi on teraz abstrakcyjny typ obiektowy związany rolą $\delta^S_{Rachunek}$ indeksującą elementy *Zakup* sekwencji *Rachunek*. Obszerniej opisano własności typów zagnieżdżonych w rozdziale 2.2.4.

f) Typy schematowe

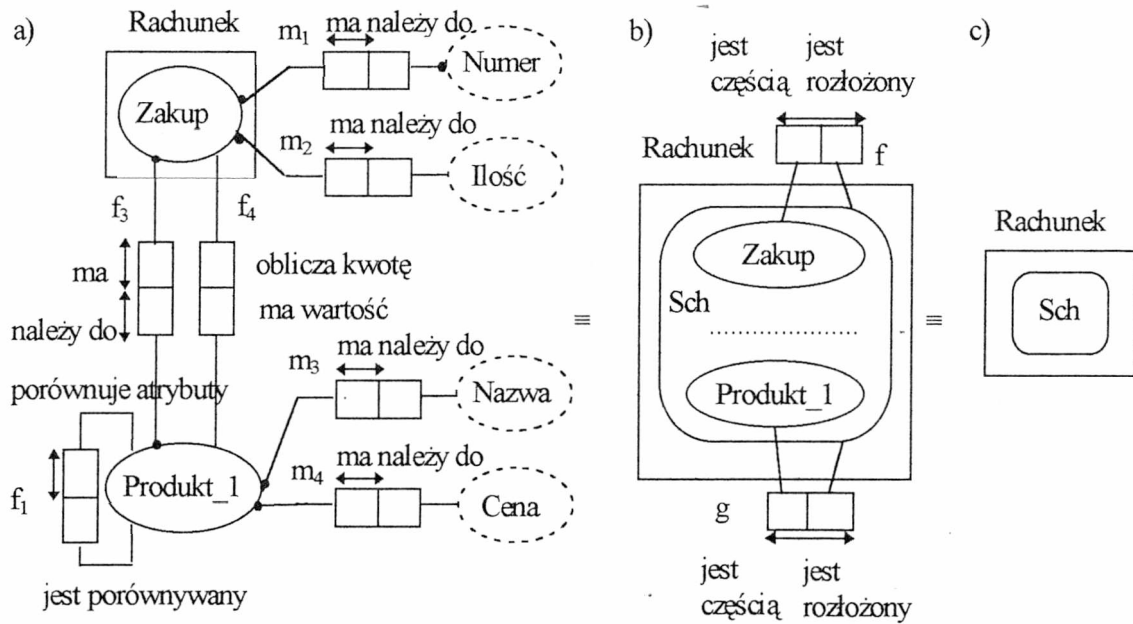
Typy schematowe *TSch* są wykorzystywane do dekompozycji obszernych schematów. Na rys. 2.6 przedstawiono przykład dekompozycji diagramu przedstawiającego sekwencję elementów typu obiektowego *Zakup* związanego z typem obiektowym *Produkt_1*. Na rys. 2.6 c zastąpiono diagram elementu typu sekwencyjnego *Zakup* z rys. 2.6 a ty-

pem schematowym $TSch$. Na rys. 2.6 b przedstawiono proces zastępowania diagramu typu obiektowego $Zakup$ wraz ze wszystkimi typami obiektowymi, z którymi jest połączony za pomocą typów faktów.

Fragment struktury informacyjnej związanej z x tzn. SI_x (2.1) może być wyprowadzona z typów obiektowych, na które x jest rozłożony (2.6)

$$TOb_x = \{y \in TOb \mid x \text{ Schem } y\}. \quad (2.18)$$

Na tej podstawie można wyprowadzić fragment diagramu struktury informacyjnej z każdego z elementów na które rozłożono x .



Rys. 2.6. Diagram ORM typu schematowego

Aksjomat A4 opisuje prawo dekompozycji schematu.

A4- (strukturalne zagnieżdzenie) $x \in TSch \Rightarrow SI_x$ jest diagramem ORM.

Każdy typ schematowy x i każdy jego składowy typ obiektowy y są związane typem faktu

$$\in_{x,y} = \{\in_{x,y}^c, \in_{x,y}^d\},$$

gdzie (2.3)

$$Baza(\in_{x,y}^c) = x$$

oraz (2.3)

$$Baza(\in_{x,y}^d) = y.$$

Ten typ faktu umożliwia przejście z typu schematowego do typu obiektowego z dekompozycji. Na rys. 2.6 b typ faktu f wiąże typ obiektowy $Zakup$ z typem schematowym $TSch$ i g z typem obiektowym $Produkt_1$.

Mamy więc (rys. 2.6 b):

$$f = \{\text{jest częścią}, \text{jest rozłożony}\},$$

$Baza(jest\ rozłożony) = Sch,$
 $Baza(jest\ częścią) = Zakup,$
 $g = \{jest\ częścią, jest\ rozłożony\},$
 $Baza(jest\ rozłożony) = Sch,$
 $Baza(jest\ częścią) = Produkt_1.$

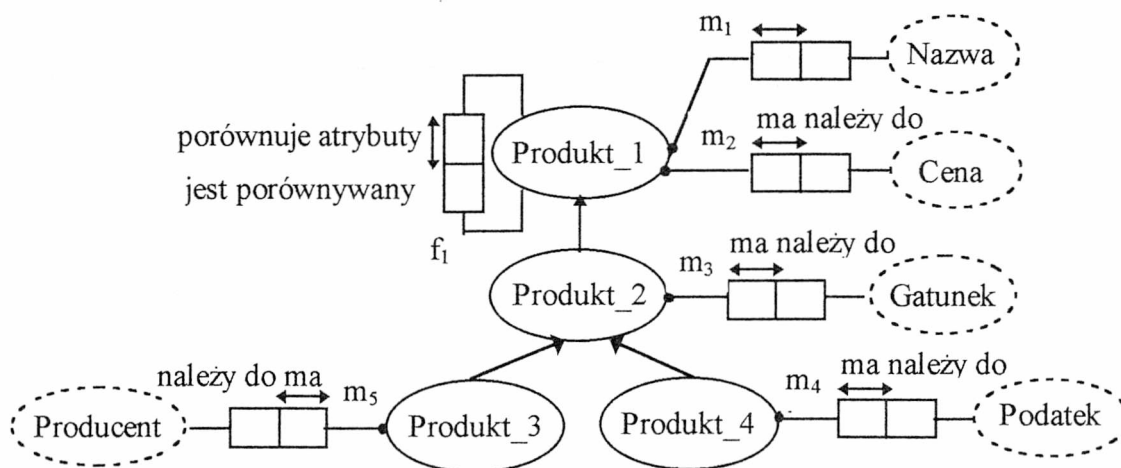
2.1.2. Podstawy formalne specjalizacji i polimorfizmu

a) Specjalizacja

Relacja $x\ Spec\ y$ (2.7) częściowego porządku określa zależność między właściwościami typów obiektowych x i y . Relacja ta porządkuje typy obiektowe abstrakcyjne według przekazywanych właściwości typowi obiektowemu należącemu do tej relacji. Oznacza to, że typ obiektowy x dziedziczy wszystkie role, jakie pełni w typach faktów typ obiektowy y i może rozwijać swoje właściwości, pełniąc dodatkowe role w innych typach faktów. Te nowe role, jakie obiekt x pełni wchodząc w relację $Spec$ (2.7) z typem obiektowym y są opisane regułą dziedziczenia $SpecRole$ (2.24). Następca z typu obiektowego x znowu dziedziczy wszystkie jego właściwości (odziedziczone po y i własne, opisane regułą dziedziczenia).

Relacja $Spec$ (2.7) wiąże typy obiektowe w strukturę drzewiastą z jednym poprzednikiem i dowolną liczbą następców. Poprzednik w drzewie y i następcy x wchodzą odpowiednio w skład relacji $x\ Spec\ y$. Korzeń (wierzchołek) drzewa dziedziczenia jest unikalny i może być wyznaczony za pomocą funkcji $Korzen$ zastosowanej do każdego elementu drzewa.

Na diagramach ORM relacja specjalizacji jest oznaczona graficznie strzałką ciągłą skierowaną do typu, od którego są dziedziczone właściwości. Na rys. 2.7 typ obiektowy $Produkt_1$ jest korzeniem drzewa dziedziczenia, a typ obiektowy $Produkt_2$ jest jego następcą, który dziedziczy udział w faktach f_1 , m_1 i m_2 oraz w wyniku reguły dziedziczenia nabywa udział w fakcie m_3 . Podobnie typ obiektowy $Produkt_3$ dziedziczy udział w faktach f_1 , m_1 , m_2 , m_3 oraz nabywa udział w fakcie m_5 , natomiast $Produkt_4$ dziedziczy te same właściwości co $Produkt_3$, lecz nabywa udział w fakcie m_4 .



Rys. 2.7. Drzewo dziedziczenia opisane relacją specjalizacji $Spec$

Relację specjalizacji opisują aksjomaty podane niżej (2.1, 2.7)).

A5 (zupełność)- $Spec \subseteq TE \times TA$,

A6 (asymetria)- $a Spec b \Rightarrow \neg b Spec a$,

A7 (zwrotność)- $a Spec b \wedge b Spec a \Rightarrow a = b$, {*}

A8 (przechodniość)- $a Spec b \wedge b Spec c \Rightarrow a Spec c$.

W każdej hierarchii specjalizacji istnieje unikatowy element zwany *Korzeniem*, wyznaczony funkcją *Korzen* (2.9), opisany następującymi aksjomatami:

A9 (spójność)- $a Spec b \Rightarrow Korzen(a) = Korzen(b)$,

A10 (przodek)- $a \neq Korzen(a) \Rightarrow a Spec Korzen(a)$.

Z tych aksjomatów opisujących korzeń w hierarchii dziedziczenia wynika ważna cecha opisana następującym lematem:

Lemat 2.1 $\neg Korzen(a) Spec b$.

Lemat ten jest wyprowadzony z aksjomatów specjalizacji i mówi, że żaden wierzchołek drzewa dziedziczenia nie może jednocześnie pełnić roli następcy w tym drzewie.

Z tego lematu wynika wniosek

Wniosek 2.1 (idempotencja) $Korzen(Korzen(a)) = Korzen(a)$.

Własności drzewa specjalizacji można opisać pewnymi funkcjami, które jakościowo opisują typy obiektowe wchodzące w relację specjalizacji. Funkcja wyznaczająca zbiór $\{y\}$ nadtypów danego typu x jest zdefiniowana jako

$$NadTypy: \varphi(TA) \rightarrow \varphi(TA), \quad \{*\}$$

czyli (2.1, 2.7, 2.9)

$$NadTypy(x) = \{y / y \in TA \wedge (x Spec y \vee (x = Korzen(x) \wedge y = x))\}. \quad \{*\} (2.19)$$

Zbiór wszystkich nadtypów określa zbiór ról, które dziedziczy dany następcza, czyli

$$NadRole: \varphi(TA) \rightarrow \varphi(TRol), \quad \{*\}$$

czyli (2.2, 2.1, 2.3, 2.19)

$$NadRole(x) = \{y / Fakt(y) \in TFakt \wedge (Baza(y) \in NadTypy(x))\}. \quad \{*\} (2.20)$$

Zbiór typów obiektowych y spełniających relację $y Spec x$ nie posiadających następców opisuje następująca funkcja *Liście_Spec* (2.1, 2.7)

$$Liście_Spec(x) = \{y / (y \in TA \wedge y Spec x \wedge (\neg \exists z \in TA [z Spec y]))\}. \quad \{*\} (2.21)$$

Na podstawie tych zależności można wyznaczyć zbiór typów abstrakcyjnych *NadTypyAbstr* i atrybutowych *NadAtrybuty*, z którymi dany obiekt uczestniczy w typach faktów zawartych w zbiorze *NadFakty* (2.1, 2.20, 2.11, 2.3):

$$NadAtrybuty(x) = \{y / y \in TWart \wedge (\exists z \in NadRole(x) \exists t \in Inne_Role(z) [Baza(t) = y])\}, \quad \{*\} (2.22)$$

$$NadTypAbstr(x) = \{y / (y \in TA) \wedge (\exists z \in NadRole(x) \exists t \in Inne_Role(z) [Baza(t) = y])\}. \quad \{*\} (2.23)$$

Reguła określająca specjalizację typu wyznacza nowe własności, jakie nabywa dany typ obiektowy w stosunku do swoich nadtypów. Funkcja wyznaczająca te role jest określona jako

$$SpecRole: \varphi(TA) \rightarrow \varphi(TRol), \quad \{*\}$$

czyli (2.2, 2.1, 2.12)

$$SpecRole(x) = \{y \mid (Fakt(y) \in TFakt) \wedge (y \in Role_Bazy(x))\}. \quad \{*\} \quad (2.24)$$

Z tych zależności można wyznaczyć nowe typy atrybutowe i abstrakcyjne, jakie w wyniku specjalizacji nabywa następca (2.1, 2.24, 2.11, 2.3):

$$SpecAtrybuty(x) = \{y \mid (y \in TWart) \wedge (\exists z \in SpecRole(x) \exists t \in Inne_Role(z) [Baza(t)=y])\}, \quad \{*\} \quad (2.25)$$

$$SpecTypyAbstr(x) = \{y \mid (y \in TA) \wedge (\exists z \in SpecRole(x) \exists t \in Inne_Role(z) [Baza(t)=y])\}. \quad \{*\} \quad (2.26)$$

Zbiór wszystkich ról, które pełni dany typ obiektowy wchodzący w hierarchię specjalizacji, są dane zależnością (2.20, 2.24)

$$WRole(x) = NadRole(x) \cup SpecRole(x). \quad \{*\} \quad (2.27)$$

Na podstawie rys. 2.7 mamy:

$$\begin{aligned} NadTypy(Produkt_3) &= \{Produkt_1, Produkt_2\}, \\ Korzen(Produkt_3) &= Produkt_1, \\ NadAtrybuty(Produkt_3) &= \{Nazwa, Cena, Gatunek\}, \\ NadTypyAbstr(Produkt_3) &= \{Produkt_1\}, \\ SpecAtrybuty(Produkt_3) &= \{Producent\}, \\ SpecTypAbstr(Produkt_3) &= \{\}, \\ Liście_Spec(Produkt_1) = Liście_Spec(Produkt_2) &= \{Produkt_3, Produkt_4\}, \\ NadTypy(Produkt_1) &= \{Produkt_1\}. \end{aligned}$$

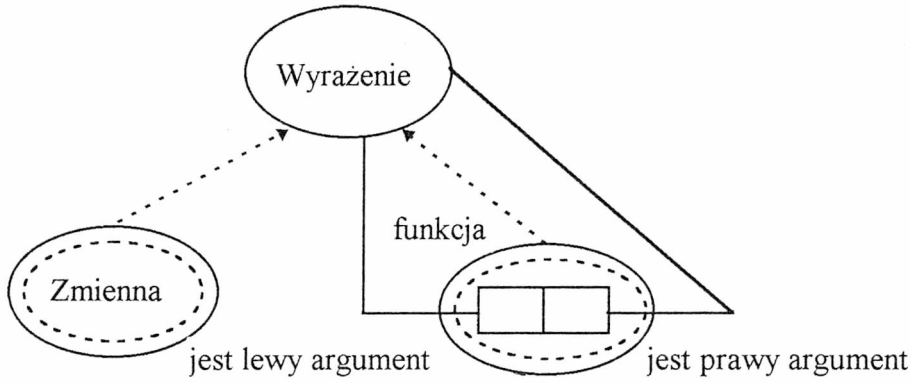
b) Polimorfizm

Mechanizm relacji *Pol* (2.8) nawiązuje do pojęcia generalizacji w metodzie ORM, która umożliwia budowę nowego typu obiektowego przez łączenie istniejących typów obiektowych. W pracy zmodyfikowano pojęcie generalizacji tak, aby pełniej wyrazić na etapie modelowania pojęcie polimorfizmu w takiej postaci, w jakiej jest zaimplementowane w obiektowych językach programowania.

Przykładem klasycznej generalizacji jest model wyrażenia przedstawiony na rys. 2.8. Wyrażenie może być zbudowane albo ze *Zmiennej* albo za pomocą *funkcji* wiążącej prostsze wyrażenia. Wystąpienia typu *Wyrażenie* swoją strukturę dziedziczą od typu *Zmienna* lub od zagnieżdżonego typu *funkcja*, czyli

$$Wyrażenie = Zmienna \vee funkcja.$$

Zmienna lub *funkcja* podstawione za *Wyrażenie* mogą więc odgrywać swoje role oraz role związane z typem *Wyrażenie*. Za pomocą polimorfizmu można zdefiniować rekurencyjnie dowolne wyrażenie. Typ *Wyrażenie* dziedziczy udział w typach faktu od swoich następców w drzewie dziedziczenia. Jest to przekład dziedziczenia „do góry”.



Rys. 2.8. Przykład diagramu ORM generalizacji

W relacjach *Pol* (2.8) wyróżnia się dwa typy: typ, który może być tylko morfizmem innych typów i jest jednocześnie typem abstrakcyjnym oraz typ, który oprócz tej własności może być własnym morfizmem czyli należy do typów polimorficznych *TPol*. Relacja *Pol* jest relacją częściowego porządku opisaną następującymi aksjomatami (2.1):

A11- (zupełność) $Pol \subseteq TE \times TA$,

A12- (asymetria) $a Pol b \Rightarrow \neg b Pol a$,

A13- (zwrotność typów polimorficznych) $a Pol b \wedge b Pol a \Rightarrow a = b \wedge a. \in TPol, \quad \{*\}$

A14- (przechodność) $a Pol b \wedge b Pol c \Rightarrow a Pol c$,

A15- (2.9) $a Pol b \wedge b Spec a \Rightarrow a, b \in TPol \wedge Korzen(a) \in TPol. \quad \{*\}$

Aksjomat **A15** określa przypadek, gdy między dwoma elementami zachodzą relacje *Spec* (2.7) i *Pol* (2.8). Wówczas elementami drzewa dziedziczenia muszą być typy obiektowe polimorficzne. Takie drzewo nazywamy drzewem polimorficznym ze specjalizacją.

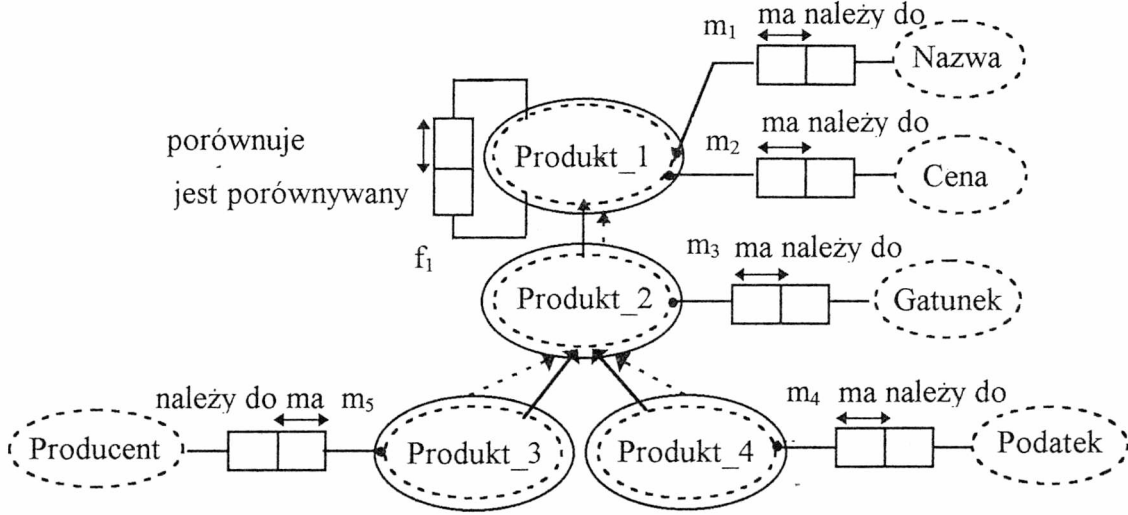
W relacji *Pol* (2.8) argument z lewej strony może wystąpić jako element równy argumentowi z prawej strony. Prawdziwe jest to dla typów polimorficznych. Takie odwzorowanie spowoduje, że wystąpienia (patrz rozdział 2.2.2) tego typu mogą zawierać elementy o jego właściwościach. Oznacza to, że w polimorfizmie węzły drzewa mogą być morfizmem swojego poprzednika i samego siebie (na diagramie zaznaczane podwójną elipsą: ciągłą i przerywaną) lub tylko morfizmem swojego poprzednika (na diagramie zaznaczane pojedynczą ciągłą elipsą). *Liście* drzewa dziedziczenia zawsze posiadają własny morfizm. Własny morfizm pozostałych elementów drzewa polimorficznego zachodzi wtedy, gdy są one powiązane relacją specjalizacji. Stąd elementy drzewa polimorficznego posiadają następujące cechy (2.1, 2.7, 2.8):

$$\forall x, y \in TPol [x Pol y \wedge y Spec x \vee x = y]. \quad \{*\} (2.28)$$

W drzewie polimorficznym nie wszystkie role są dziedziczone drogą specjalizacji. Określa to reguła specjalizacji. Na diagramach role dziedziczone występują jako role odgrywane przez typ obiektowy zaznaczony linią przerywaną. Brak własnego morfizmu jest zaznaczony pojedynczą elipsą ciągłą i wyrażane jest przez

$$\neg a Pol a.$$

Na rys. 2.8 typ *Wyrażenie* nie posiada własnego morfizmu. W modelu może również istnieć powiązanie między typami drzewa polimorficznego typu specjalizacja, co graficznie jest zaznaczone dodatkową ciągłą strzałką między przerywanymi elipsami. Takie typy obiektowe muszą być również typami polimorficznymi. Przykład takiego modelu pokazano na rys. 2.9.



Rys. 2.9. Diagram ORM reprezentujący polimorfizm typów powiązanych relacją dziedziczenia

W przykładzie na rys. 2.9 Typ *Produkt_1* może zastąpiony przez samego siebie lub przez dowolnego następcę razem z ich właściwościami (rolami), czyli

$$Produkt_1 = Produkt_1 \vee Produkt_2 \vee Produkt_3 \vee Produkt_4,$$

$$Produkt_2 = Produkt_2 \vee Produkt_3 \vee Produkt_4,$$

$$Produkt_3 = Produkt_3,$$

$$Produkt_4 = Produkt_4.$$

Niżej podano szereg funkcji charakteryzujących relację polimorfizmu w modelu ORM. Funkcja generująca zbiór wszystkich podtypów od zadanego węzła w stronę *Liści* (razem z *Liśćmi*) drzewa dziedziczenia jest zdefiniowana jako

$$PodTypy: \varphi(TA) \rightarrow \varphi(TA), \quad \{*\}$$

czyli (2.1, 2.8)

$$PodTypy(x) = \{y / (y \in TA) \wedge (x Pol y)\}. \quad \{*\}(2.29)$$

Funkcja

$$Liście: \varphi(TA) \rightarrow \varphi(TA), \quad \{*\}$$

wyznacza wszystkie *Liście* osiągalne z węzła x (2.29, 2.1, 2.8)

$$Liście(x) = \{y / (y \in PodTypy(x)) \wedge (y \in TPol) \wedge (\neg \exists z \in PodTypy(x) [y Pol z])\}. \quad \{*\}(2.30)$$

Zbiór wszystkich typów, które mogą wystąpić jako dowolny element drzewa dziedziczenia związany relacją *Pol* jest wyznaczony funkcją

$$PolTypy: \varphi(TA) \rightarrow \varphi(TA), \quad \{*\}$$

czyli (2.29)

$$PolTypy(x) = \{y \mid y \in PodTypy(x)\}. \quad \{*\} (2.31)$$

Właściwości czyli zbiór ról, jakie przydzielone są każdemu typowi zawartemu w zbiorze wygenerowanemu przez funkcję *PolTypy* są wyznaczone funkcją *PolRole*

$$PolRole: \varphi(TA) \rightarrow \varphi(TRol), \quad \{*\}$$

czyli (2.2, 2.1, 2.3, 2.31)

$$PolRole(x) = \{y \mid Fakt(y) \in TFakt \wedge (Baza(y) \in PolTypy(x) \vee Baza(y)=x)\}. \quad \{*\} (2.32)$$

Typy atrybutowe oraz typy abstrakcyjne związane ze zbiorem *PolRole* są następujące:

$$PolAtrybuty(x) = \{y \mid y \in TWart \wedge (\exists z \in PolRole(x) \exists t \in Inne_Role(z) [Baza(t)=y])\}, \quad \{*\} (2.33)$$

$$PolTypAbstr(x) = \{y \mid (y \in TA) \wedge (\exists z \in PolRole(x) \exists t \in Inne_Role(z) [Baza(t)=y])\}. \quad \{*\} (2.34)$$

W przypadku, gdy drzewo polimorficzne zawiera jednocześnie wszystkie elementy polimorficzne powiązane specjalizacją, należy stosować odpowiednio funkcje *PolTypyS* i *PolRoleS* (2.1, 2.29)

$$PolTypyS(x) = \{y \mid y \in TPol \wedge (y \in PodTypy(x) \vee y = x)\}. \quad \{*\} (2.35)$$

W drzewie polimorficznym zbiór ról generowany przez funkcję *PolRole* nie jest związany regułą specjalizacji z żadnym elementem drzewa., czyli (2.2, 2.30, 2.27)

$$y \in PolRole(x) \Rightarrow (y \in Fakt(y)) \wedge y \notin WRole(Baza(y)). \quad \{*\} (2.36)$$

Stąd (2.36) role dostępne w każdym węźle polimorficznego drzewa dziedziczenia są zdefiniowane jako 2.2, 2.1, 2.3, 2.36, 2.27, 2.32)

$$PolRoleS(x) = \{y \mid Fakt(y) \in TFakt \wedge Baza(y) \in PolTypyS(x) \wedge (y \in WRole(Baza(y)) \vee y \in PolRole(Baza(y)))\}. \quad \{*\} (2.37)$$

Typy atrybutowe oraz typy abstrakcyjne związane ze zbiorem *PolRoleS* są następujące:

$$PolAtrybutyS(x) = \{y \mid y \in TWart \wedge (\exists z \in PolRoleS(x) \exists t \in Inne_Role(z) [Baza(t)=y])\}, \quad \{*\} (2.38)$$

$$PolTypAbstrS(x) = \{y \mid (y \in TA) \wedge (\exists z \in PolRoleS(x) \exists t \in Inne_Role(z) [Baza(t)=y])\}. \quad \{*\} (2.39)$$

Na podstawie rys. 2.9 mamy:

$PodTypy(Produkt_1) = \{Produkt_2, Produkt_3, Produkt_4\},$
 $Liście(Produkt_1) = \{Produkt_3, Produkt_4\},$
 $PolTypyS(Produkt_1) = \{Produkt_1, Produkt_2, Produkt_3, Produkt_4\},$
 $PolTypyS(Produkt_3) = \{Produkt_3\},$
 $PolAtrybutyS(Produkt_1) = \{Nazwa, Cena, Gatunek, Podatek, Producent\},$
 $PolTypAbstrS(Produkt_1) = \{\},$
 $PolAtrybutyS(Produkt_2) = \{Nazwa, Cena, Gatunek, Podatek, Producent\},$
 $PolTypAbstrS(Produkt_2) = \{\},$
 $PolAtrybutyS(Produkt_3) = \{Nazwa, Cena, Gatunek, Producent\},$

$PolTypAbstrS(Produkt_3) = \{\}$,
 $PolAtrybutyS(Produkt_4) = \{Nazwa, Cena, Gatunek, Podatek\}$,
 $PolTypAbstrS(Produkt_4) = \{\}$.

Do wyznaczenia zbioru $PolRoleS$ użyto operatora projekcji π (2.62, 2.63) opisanego w rozdziale 2.2.4:

$PolRole(Produkt_1) = \{\}$,
 $PolRoleS(Produkt_1) = \{\pi_{ma}m_1, \pi_{ma}m_2, \pi_{ma}m_3, \pi_{ma}m_4, \pi_{ma}m_5, \pi_{porównuje\ atrybuty}f_1, \pi_{jest\ porównywany}f_1\}$,
 $PolRoleS(Produkt_3) = \{\pi_{ma}m_1, \pi_{ma}m_2, \pi_{ma}m_3, \pi_{ma}m_5, \pi_{porównuje\ atrybuty}f_1, \pi_{jest\ porównywany}f_1\}$.

c) Typy powiązane relacyjnie

Zależności relacyjne między typami obiektowymi a i b istnieją wtedy, gdy (2.9, 2.3, 2.31)

- a) **A16-** $\forall a \in TA [Korzen(a) = a \Rightarrow a \sim a]$,
- b) **A17-** $\exists a, b \in TA [Korzen(a) = Korzen(b) \wedge (a\ Spec\ b \vee b\ Spec\ a) \Rightarrow a \sim b \wedge b \sim a]$,
- c) **A18-** $\exists a, b \in TA [Korzen(a) = Korzen(b) \wedge (a\ Pol\ b \vee b\ Pol\ a) \Rightarrow a \sim b \wedge b \sim a]$,
- d) **A19-** $\exists q \in f, p \in t, f, t \in TFakt, x \in TA [Korzen(Baza(q) = Korzen(x) \wedge Korzen(Baza(p) = Korzen(x) \wedge (Baza(p), Baza(q) \in PolTypy(x) \Rightarrow q \sim p)]$. {*}

Na podstawie aksjomatów a - d mamy (2.1, 2.3, 2.2)

$$a, b \in TA \wedge (a \sim b) \wedge \exists p, q \in TRol [Baza(p) = a \wedge Baza(q) = b] \Rightarrow Fakt(p) \sim Fakt(q) \wedge p \sim q. \quad \{*\}(2.40)$$

Dla przypadku a) wszystkie role pełnione przez ten sam obiekt są relacyjnie powiązane. Przypadki b) i c) oznaczają, że role pełnione przez typy obiektywne powiązane zależnościami Pol (rozd. 2.1.2) i $Spec$ (rozd. 2.1.2) są również relacyjne powiązane. Przypadek d) oznacza, że role wszystkich typów obiektowych powiązanych relacją Pol i należących do każdego z poddrzew drzewa dziedziczenia są relacyjnie powiązane na poziomie wierzchołka tego poddrzewa. Przechodność zależności relacyjnych wynika z własności relacji $Spec$ i Pol .

Następujące aksjomaty opisują pozostałe przypadki relacyjnego powiązania między typami obiektowymi (2.1, 2.5):

- A20-** $x, y \in TZb \wedge Elt(x) \sim Elt(y) \models x \sim y$,
- A21-** $x, y \in TSekw \wedge Elt(x) \sim Elt(y) \models x \sim y$,
- A22-** $TOb_x \sim TOb_y \models x \sim y$.

Na rys. 2.9 mamy:

$Produkt_1 \sim Produkt_1$,
 $Produkt_2 \sim Produkt_2, Produkt_2 \sim Produkt_1$,
 $Produkt_3 \sim Produkt_3, Produkt_3 \sim Produkt_2, Produkt_3 \sim Produkt_1$,

Produkt_4 ~ Produkt_4, Produkt_4 ~ Produkt_1,
m1 ~ m2, m1 ~ f1,
m3 ~ m1, m3 ~ m2, m3 ~ f1,
m4 ~ m3, m4 ~ m1, m4 ~ m2, m4 ~ f1,
m5 ~ m4, m5 ~ m3, m5 ~ m1, m5 ~ m2, m5 ~ f1.

Na rys. 2.9 mamy:

Produkt_1 ~ Produkt_1, Produkt_1 ~ Produkt_2, Produkt_1 ~ Produkt_3, Produkt_1 ~
Produkt_4,
Produkt_2 ~ Produkt_2, Produkt_2 ~ Produkt_3, Produkt_2 ~ Produkt_4, Produkt_2 ~
Produkt_1,
Produkt_3 ~ Produkt_3, Produkt_3 ~ Produkt_2, Produkt_3 ~ Produkt_1,
Produkt_4 ~ Produkt_4, Produkt_4 ~ Produkt_2, Produkt_4 ~ Produkt_1,
m1 ~ m2, m1 ~ f1, m1 ~ m3, m1 ~ m4, m1 ~ m5,
m3 ~ m2, m3 ~ m1, m3 ~ f1, m3 ~ m4, m3 ~ m5,
m4 ~ m3, m4 ~ m1, m4 ~ m2, m4 ~ f1, m4 ~ m5,
m5 ~ m3, m5 ~ m1, m5 ~ m2, m5 ~ f1, m5 ~ m4.

2.2. Definicje własności semantycznych modelu

2.2.1. Populacja typów obiektowych

Struktura informacyjna wyznacza ramy świata rzeczywistego zwanego **Universe of Discourse (UoD)**. Stan **UoD** odpowiada wystąpieniom (populacji) typów należących do struktury informacyjnej. Zmianom stanów w **UoD** odpowiadają zmiany populacji związanej ze strukturą informacyjną. W modelowaniu konceptualnym właśnie struktura informacyjna jest podstawą do wyznaczania modelu zachowania obiektów.

Definicja

Universe of Discourse jest izomorficzne ze zbiorem możliwych populacji struktury informacyjnej i tranzytywnie zwrotne.

Populacja struktury informacyjnej i związane z nią ograniczenia są konceptualnymi właściwościami struktury informacyjnej. Populacja struktury informacyjnej jest zbiorem związanym z wystąpieniami typów obiektowych. Populacja odpowiada każdemu ze stanów **UoD** tzn. spełniona jest zależność

$$JestPop(Si, Pop). \quad (2.41)$$

Pop jest odwzorowaniem

$$Pop: TOb \rightarrow \varphi(\Omega), \quad (2.42)$$

gdzie Ω jest uniwersum wystąpień, jakie zawiera populacja struktury informacyjnej. Zbiór populacji jest równy

$$POP = TOb \rightarrow \varphi(\Omega). \quad (2.43)$$

Struktura informacyjna może posiadać populację, jeżeli istnieje połączenie między typami atrybutowymi i dziedziną konkretnych wartości. Formalnie jest to zdefiniowane

przez funkcję

$$\text{Dziedzina: } TWart \rightarrow D. \quad (2.44)$$

Zbiór D jest zbiorem dziedzin konkretnych wartości (łańcuchy, liczby itp.) czyli jest zbiorem nośników algebry niejednorodnej

$$A = \langle D, F \rangle, \quad (2.45)$$

gdzie F jest zbiorem operacji (np. +, -...) na gatunkach w D .

Definicja **UoD** jest następująca: (2.46)

1. $D \subseteq \Omega$, wystąpienia gatunków algebry są elementami UoD,
2. $\Theta \subseteq \Omega$, gdzie Θ jest policzalną dziedziną wartości amorficznych., które występują w populacji typów elementarnych (TE),
3. $x_1, \dots, x_n \in \Omega \wedge p_1, \dots, p_n \in TRol \Rightarrow (p_1: x_1, \dots, p_n: x_n) \in \Omega$. Zbiór $(p_1: x_1, \dots, p_n: x_n)$ oznacza odwzorowanie, przypisujące x_i do każdej roli. Te odwzorowania wyznaczają populację typów faktów,
4. $x_1, \dots, x_n \in \Omega \Rightarrow \{x_1, \dots, x_n\} \in \Omega$. Zbiór wystąpień $x_1, \dots, x_n$ elementów typu zbiorowego odpowiada wystąpieniom typu zbiorowego,
5. $x_1, \dots, x_n \in \Omega \Rightarrow \langle x_1, \dots, x_n \rangle \in \Omega$. Zbiór wystąpień $x_1, \dots, x_n$ elementów typu sekwencyjnego odpowiada wystąpieniom typu sekwencyjnego,
6. $X_1, \dots, X_n \in \Omega \wedge O_1, \dots, O_n \in TOB \Rightarrow (O_1: X_1, \dots, O_n: X_n) \in \Omega$. Przypisane zbiory wystąpień do typów obiektowych są także wystąpieniami czyli populacją struktury informacyjnej. Dotyczy wszystkich typów schematowych,

Aksjomaty 23 - 35 określają związek populacji z danym typem obiektowym należącym do struktury informacyjnej (2.1).

A23- (reguła typów) $x, y \notin TWart \wedge x \neq y \Rightarrow Pop(x) \cap Pop(y) = \emptyset$.

Oznacza to, że wystąpienia abstrakcyjnych typów obiektowych są tylko wtedy wspólne, jeżeli są relacyjnie powiązane.

A24- $x \in TWart \Rightarrow Pop(x) \subseteq Dziedzina(x)$.

Oznacza to, że populacja typów atrybutowych jest zbiorem wartości należących do odpowiadających dziedzin konkretnych wartości.

A25- $x \in TE \wedge JestKorzen(x) \Rightarrow Pop(x) \subseteq \Theta$.

Typy obiektowe występujące jako *Korzeń* nie mogą być podtypem w hierarchii specjalizacji i polimorficznej. Jest to sformalizowane za pomocą wyrażenia (predykatu)

$$JestKorzen(x) \equiv \neg x \text{ Spec } y \wedge \neg x \text{ Pol } y.$$

a populacja typów będących *Korzeniami* jest zbiorem wartości z dziedziny wartości abstrakcyjnych.

A26- (reguła zgodności)

$$x \in TFakt \wedge y \in Pop(x) \Rightarrow y: x \rightarrow \Omega \wedge \forall p \in x [y(p) \in Pop(Baza(p))].$$

Oznacza to, że populacja typów faktu jest krotką. Krotka t w populacji typu faktu jest f jest odwzorowaniem wszystkich ról na wartości związanych z nimi typów bazowych.

A27- (reguła typów zbiorowych)

$$x \in TZb \wedge y \in Pop(x) \Rightarrow y \in \varphi(Pop(Elt(x))) - \{\emptyset\}.$$

A28- (reguła bazy zbiorów)

$$x \in TZb \Rightarrow Pop(\in_x) = \{\{\in_x^z: u, \in_x^e: v\} / u \in Pop(x) \wedge v \in u\}.$$

Ta reguła jest regułą wyprowadzania populacji typu faktu $\in_x$ (tworzącego zbiór). Wyprowadzenie populacji typów bazowych roli $\in_x^e$, czyli $v \in Pop(Elt(x))$ jest opisane za pomocą reguły zgodności.

A29- (reguła typu sekwencyjnego)

$$x \in TSekw \wedge y \in Pop(x) \Rightarrow y \in Pop(Elt(x))^+.$$

Populacja sekwencji (niepustej) jest sekwencją wystąpień odpowiadających typów obiektowych.

A30- (reguła dekompozycji sekwencji)

$$x \in TSekw \Rightarrow Pop(\in_x) = \{\{\in_x^s: u, \in_x^e: v\} / u \in Pop(x) \wedge \exists i \in I [u[i] = v]\}.$$

A31- (reguła dekompozycji sekwencji)

$$x \in TSekw \Rightarrow Pop(@_x) = \{\{@_x^s: u, @_x^i: v\} / u \in Pop(\in_x) \wedge u(\in_x^s)[v] = u(\in_x^e)\}.$$

Obie reguły dekompozycji sekwencji są regułami wyprowadzania populacji dla typów faktów tworzących sekwencję $\in_x$ i indeksujących sekwencję $@_x$.

A32- (reguła dekompozycji)

$$x \in TSch \wedge y \in Pop(x) \Rightarrow JestPop(Si_x, y).$$

Populacja typu schematowego składa się populacji struktury informacyjnej związanej z danym typem schematowym.

A33- (reguła dekompozycji)

$$x Schem y \Rightarrow Pop(\in_{x,y}) = \{\{\in_{x,y}^C: u, \in_{x,y}^d: v\} / u \in Pop(x) \wedge v \in u(y)\}.$$

Relacja między populacją typu schematowego, będącego dekompozycją typu x na typ y , a populacją jego składowych obiektów jest określona populacją jego typu faktu $\in_{x,y}$. Podany lemat wyznacza populację typu schematowego jako sumę populacji jego typów składowych.

Lemat 2.2 $x Schem y \Rightarrow \forall u \in Pop(x) [u(y) \subseteq Pop(y)]$.

Dowód- Załóżmy, że $u \in Pop(x)$ i $v \in u(y)$. Stosując regułę dekompozycji mamy, że $\{\in_{x,y}^C: u, \in_{x,y}^d: v\} \in Pop(\in_{x,y})$. Z reguły zgodności i z faktu, że $Baza(\in_{x,y}^d) = y$ mamy, że $v \in Pop(y)$.

2.2.2. Populacja typów obiektowych związanych relacją specjalizacji i polimorfizmu

a) Populacja w związku specjalizacji

A34- (reguła specjalizacji) $x \text{ Spec } y \Rightarrow \text{Pop}(x) \subseteq \text{Pop}(y)$.

Ta reguła oznacza, że wystąpienia typów potomnych są również populacją nadtypu, która jednak spełnia regułę specjalizacji. Oznacza to, że populacja typu potomnego x jest równa (2.46, 2.9, 2.27, 2.3, A16, A17)

$$\text{Pop}(x) = \{z \mid z \in \text{Pop}(\text{Korzen}(x)) \wedge \forall q \in \text{WRole}(x) [z = \text{Pop}(\text{Baza}(q))]\}. \quad \{*\} \quad (2.47)$$

Stąd populacja każdego podtypu jest podzbiorem populacji nadtypu spełniającą regułę specjalizacji *SpecRole* (2.24).

Pojęcie specjalizacji jest analogiczne do pojęcia podzbioru w ramach teorii aksjomatycznej zbiorów Zermelo - Fraenkla - Skolema [60, 86]. Aksjomat podzbiorów jest następujący:

$$\forall y [y \in b \Leftrightarrow (y \in a \wedge \Phi(x_1 \dots x_n y))], \quad (2.48)$$

który pozwala na konstruowanie zbioru b , zgodnie z wybraną formułą zdaniową Φ działającą nad zbiorem a w kontekście elementów $x_1 \dots x_n$. Każdy element zbioru b występuje jako podtyp każdego elementu zbioru a , stąd wystąpienia populacji typu b są podzbiorem populacji typu a i pozostają w zależności Φ do wystąpień x populacji typu a

$$\text{Pop}(b) = \{x \mid x \in \text{Pop}(a) \wedge \Phi(x)\}. \quad (2.49)$$

Formuła $\Phi(x)$ odpowiada formule specjalizacji *SpecRole*. Zgodnie z tym założeniem $\text{Pop}(b)$ może być wyrażona jako (2.24, 2.46, 2.3)

$$\text{Pop}(b) = \{x \mid x \in \text{Pop}(a) \wedge \forall y \in \text{SpecRole}(b) [\text{Pop}(\text{Baza}(y)) = x]\}. \quad \{*\} \quad (2.50)$$

Stąd populacja nadtypu jest zbiorem wygenerowanym w następujący sposób:

$$\text{Pop}(a) = \{x \mid b \text{ Spec } a \wedge \neg (\forall y \in \text{SpecRole}(b) [\text{Pop}(\text{Baza}(y)) = x])\}. \quad \{*\} \quad (2.51)$$

W celu podkreślenia, że wszystkie wystąpienia typów a są widziane bez reguł specjalizacji, wprowadza się operator konwersji *Typ(Pop)*, oznaczający wystąpienie typu obiektowego wyznaczonego przez *Typ*. Operator konwersji jest zdefiniowany następująco (2.46, 2.1, 2.7)

$$\text{Pop}(a) = \{x \mid (\forall y \in (\text{Pop}(b), a, b \in \text{TA}, b \text{ Spec } a [x = a(y)]))\}, \quad \{*\} \quad (2.52)$$

gdzie $a(y)$ oznacza wystąpienie typu a .

b) Populacja w związku polimorfizmu

A35- (reguła polimorfizmu) $x \text{ Pol } y \Rightarrow \text{Pop}(x) = \cup_{x \text{ Pol } y} \text{Pop}(y)$.

Reguła polimorfizmu określa, że populacja nadtypu jest sumą populacji każdego z podtypów. W drzewie dziedziczenia te podtypy mogą należeć do zbiorów *PolTypy* (2.31) lub *PolTypyS* (2.35) - w przypadku, gdy elementy drzewa polimorficznego są

związane relacją specjalizacji. Stąd populacja dowolnego nadtypu x z drzewa dziedziczenia jest równa (A16, A18, A19, 2.31)

$$Pop(x) = \{y \mid (u \in PolTypy(x)) \wedge (y \in Pop(u))\}. \quad \{*\} (2.53)$$

Dla typów polimorficznych związanych relacją specjalizacji populacja dowolnego nadtypu jest równa (2.35, 2.46)

$$Pop(x) = \{y \mid (u \in PolTypyS(x)) \wedge (y \in Pop(u))\}. \quad \{*\} (2.54)$$

Pojęcie polimorfizmu jest analogiczne do pojęcia sumy podzbiorów w ramach teorii aksjomatycznej zbiorów *Zermelo - Fraenkla - Skolema* [60, 86]. Aksjomat sumy podzbiorów jest następujący:

$$\cup a = \{z \mid \exists y [y \in a \wedge z \in y]\}. \quad (2.55)$$

Pozwala on na konstruowanie i dołączanie nowych typów, poszerzających cechy istniejącego nadtypu. Nowy zbiór $a \cup b$ można skonstruować na podstawie zbioru a , przez łączenie wszystkich elementów zbiorów a i b , stąd populacja supertypu $\cup \{a, b\}$ jest sumą populacji typów a i b

$$Pop(\cup \{a, b\}) = Pop(a) \cup Pop(b). \quad (2.56)$$

Populacja dla przedstawionych powyżej typów obiektowych jest pokazana w rozdziale 2.2.3 na przykładach opartych na rys. 2.4 - 2.11.

2.2.3. Przykłady populacji typów obiektowych

a) Z rys. 2.4 dla typu zbiorowego mamy (2.46, A24, A27, A28, A26, 2.16):

$$Pop(Zakup) = \{zakup1, zakup2\},$$

$$Pop(Numer) = \{1, 2\},$$

$$Pop(Rachunek) = \{\{zakup1\}, \{zakup1, zakup2\}\},$$

$$Pop(\in_{Rachunek}) = \{\{\in^Z_{Rachunek} : \{zakup1\}, \in^e_{Rachunek} : zakup1\}, \\ \{\in^Z_{Rachunek} : \{zakup1, zakup2\}, \in^e_{Rachunek} : zakup1\}, \\ \{\in^Z_{Rachunek} : \{zakup1, zakup2\}, \in^e_{Rachunek} : zakup2\}\},$$

$$Pop(f) = \{\{ma : zakup1, należy do: 1\}, \{ma : zakup2, należy do: 2\}\};$$

b) z rys. 2.5 dla sekwencji mamy (2.46, A24, A29, A30, A31, 2.17):

$$Pop(Zakup) = \{zakup1, zakup2\},$$

$$Pop(Numer) = \{1, 2\},$$

$$Pop(Indeks) = \{0, 1\},$$

$$Pop(Rachunek) = \{\{zakup1\}, \{zakup1, zakup2\}\},$$

$$Pop(\in_{Rachunek}) = \{\{\in^S_{Rachunek} : \{zakup1\}, \in^e_{Rachunek} : zakup1\}, \\ \{\in^Z_{Rachunek} : \{zakup1, zakup2\}, \in^e_{Rachunek} : zakup1\}, \\ \{\in^Z_{Rachunek} : \{zakup1, zakup2\}, \in^e_{Rachunek} : zakup2\}\},$$

$$Pop(\delta_{Rachunek}) = \{\{\delta^S_{Rachunek} : \{\in^S_{Rachunek} : \{zakup1\}, \in^e_{Rachunek} : zakup1\}, \delta^I_{Rachunek} : 0\}, \\ \{\delta^S_{Rachunek} : \{\in^S_{Rachunek} : \{zakup1, zakup2\}, \in^e_{Rachunek} : zakup1\}, \delta^I_{Rachunek} : 0\}, \\ \{\delta^S_{Rachunek} : \{\in^S_{Rachunek} : \{zakup1, zakup2\}, \in^e_{Rachunek} : zakup2\}, \delta^I_{Rachunek} : 1\}\},$$

$$Pop(f) = \{\{ma : zakup1, należy do: 1\}, \{ma : zakup2, należy do: 2\}\};$$

c) populację typu schematowego z rys. 2.6 przedstawiono zastępując następujące nazwy (2.6, 2.18, A4, A24, A32, A33):

$$\begin{aligned}
 f &- \in_{Sch, Zakup}; \text{ jest rozłożony } - \in_{Sch, Zakup}^{Sch}, \text{ jest częścią } - \in_{Sch, Zakup}^{Zakup}, \\
 g &- \in_{Sch, Produkt_1}; \text{ jest rozłożony } - \in_{Sch, Produkt_1}^{Sch}, \text{ jest częścią } - \in_{Sch, Produkt_1}^{Produkt_1}, \\
 Pop(Sch) &= \{schemat1, schemat2\}, \\
 Pop(Zakup) &= \{zakup1, zakup2\}, \\
 Pop(Produkt_1) &= \{produkt1, produkt2\}, \\
 Pop(\in_{Sch, Zakup}) &= \{ \{ \in_{Sch, Zakup}^{Sch} : schemat1, \in_{Sch, Zakup}^{Zakup} : zakup1 \}, \\
 &\quad \{ \in_{Sch, Zakup}^{Sch} : schemat2, \in_{Sch, Zakup}^{Zakup} : zakup2 \} \}, \\
 Pop(\in_{Sch, Produkt_1}) &= \{ \{ \in_{Sch, Produkt_1}^{Sch} : schemat1, \in_{Sch, Produkt_1}^{Produkt_1} : produkt1 \}, \\
 &\quad \{ \in_{Sch, Produkt_1}^{Sch} : schemat2, \in_{Sch, Produkt_1}^{Produkt_1} : produkt2 \} \};
 \end{aligned}$$

d) reszta populacji z rys. 2.6 jest przedstawiona poniżej oprócz typów faktów wiążących elementy sekwencji (które są domyślne na diagramie ORM)-(2.46, A24, A26):

$$\begin{aligned}
 Pop(Numer) &= \{1, 2\}, \\
 Pop(Ilość) &= (10, 20\}, \\
 Pop(Nazwa) &= \{„produkt_a”, „produkt_b”\}, \\
 Pop(Cena) &= \{9, 17\}, \\
 Pop(m_1) &= \{ \{ ma : zakup1, należy do : 1 \}, \{ ma : zakup2, należy do : 2 \} \}, \\
 Pop(m_2) &= \{ \{ ma : zakup1, należy do : 10 \}, \{ ma : zakup2, należy do : 20 \} \}, \\
 Pop(m_3) &= \{ \{ ma : produkt1, należy do : „produkt_a” \}, \\
 &\quad \{ ma : produkt2, należy do : „produkt_b” \} \}, \\
 Pop(m_4) &= \{ \{ ma : produkt1, należy do : 9 \}, \{ ma : produkt2, należy do : 17 \} \}, \\
 Pop(f_1) &= \{ porównuje : produkt2, jest porównywany : produkt1 \}, \\
 Pop(f_2) &= \{ \{ ma : zakup1, należy do : produkt1 \}, \{ ma : zakup2, należy do : produkt2 \} \}, \\
 Pop(f_3) &= \{ \{ oblicza : zakup1, ma wartość : produkt1 \}, \\
 &\quad \{ oblicza : zakup2, ma wartość : produkt2 \} \};
 \end{aligned}$$

e) populacja z rys. 2.7 związana z typami powiązanymi relacją specjalizacji (A24, 2.47, 2.51, A26):

$$\begin{aligned}
 Pop(Nazwa) &= \{„produkt_a”, „produkt_b”, „produkt_c”, „produkt_d”\}, \\
 Pop(Cena) &= \{9, 17\}, \\
 Pop\{Gatunek\} &= \{1, 2\}, \\
 Pop(Podatek) &= \{22, 7\}, \\
 Pop(Producent) &= \{„Producent_A”, „Producent_B”\}, \\
 Pop(Produkt_1) &= \{produkt1, Produkt_1(produkt2), Produkt_1(produkt3), \\
 &\quad Produkt_1(produkt4)\}, \\
 Pop(Produkt_2) &= \{produkt2, Produkt_2(produkt3), Produkt_2(produkt4)\}, \\
 Pop(Produkt_3) &= \{produkt3\}, \\
 Pop(Produkt_4) &= \{produkt4\}, \\
 Pop(f_1) &= \{ \{ porównuje : produkt2, jest porównywany : produkt1 \}, \\
 &\quad \{ porównuje : produkt3, jest porównywany : produkt1 \}, \\
 &\quad \{ porównuje : produkt3, jest porównywany : produkt2 \}, \\
 &\quad \{ porównuje : produkt4, jest porównywany : produkt1 \}, \\
 &\quad \{ porównuje : produkt4, jest porównywany : produkt2 \}, \\
 &\quad \{ porównuje : produkt4, jest porównywany : produkt3 \} \},
 \end{aligned}$$

$Pop(m_1) =$
 $\{\{ma : produkt1, nalezy\ do : „produkt_a”\}, \{ma : produkt2, nalezy\ do : „produkt_b”\},$
 $\{ma : produkt3, nalezy\ do : „produkt_c”\}, \{ma : produkt4, nalezy\ do : „produkt_d”\}\},$
 $Pop(m_2) = \{\{ma : produkt1, nalezy\ do : 9\}, \{ma : produkt2, nalezy\ do : 17\},$
 $\{ma : produkt3, nalezy\ do : 17\}, \{ma : produkt4, nalezy\ do : 9\}\},$
 $Pop(m_3) = \{\{ma : produkt2, nalezy\ do : 2\}, \{ma : produkt3, nalezy\ do : 1\},$
 $\{ma : produkt4, nalezy\ do : 2\}\},$
 $Pop(m_4) = \{ma : produkt4, nalezy\ do : 22\},$
 $Pop(m_5) = (ma : produkt3, nalezy\ do : „Producent_A”);$

f) populacja z rys. 2.9 związana z typami polimorficznymi związanymi relacją specjalizacji (2.46, A24, 2.53, 2.54, A26):

$Pop(Nazwa) = \{„produkt_a”, „produkt_b”, „produkt_c”, „produkt_d”\},$
 $Pop(Cena) = \{9, 17\},$
 $Pop\{Gatunek\} = \{1, 2\},$
 $Pop(Podatek) = \{22, 7\},$
 $Pop(Producent) = \{„Producent_A”, „Producent_B”\},$
 $Pop(Produkt_1) = \{produkt1, produkt2, produkt3, produkt4\},$
 $Pop(Produkt_2) = \{produkt2, produkt3, produkt4\},$
 $Pop(Produkt_3) = \{produkt3\},$
 $Pop(Produkt_4) = \{produkt4\},$
 $Pop(f_i) = \{\{porownuje : produkt2, jest\ porownywany : produkt1\},$
 $\{porownuje : produkt3, jest\ porownywany : produkt1\},$
 $\{porownuje : produkt3, jest\ porownywany : produkt2\},$
 $\{porownuje : produkt4, jest\ porownywany : produkt1\},$
 $\{porownuje : produkt4, jest\ porownywany : produkt2\},$
 $\{porownuje : produkt4, jest\ porownywany : produkt3\}\},$
 $Pop(m_1) =$
 $\{\{ma : produkt1, nalezy\ do : „produkt_a”\}, \{ma : produkt2, nalezy\ do : „produkt_b”\},$
 $\{ma : produkt3, nalezy\ do : „produkt_c”\}, \{ma : produkt4, nalezy\ do : „produkt_d”\}\},$
 $Pop(m_2) = \{\{ma : produkt1, nalezy\ do : 9\}, \{ma : produkt2, nalezy\ do : 17\},$
 $\{ma : produkt3, nalezy\ do : 17\}, \{ma : produkt4, nalezy\ do : 9\}\},$
 $Pop(m_3) = \{\{ma : produkt2, nalezy\ do : 2\}, \{ma : produkt3, nalezy\ do : 1\},$
 $\{ma : produkt4, nalezy\ do : 2\}\},$
 $Pop(m_4) = \{\{ma : produkt4, nalezy\ do : 22\}\},$
 $Pop(m_5) = \{(ma : produkt3, nalezy\ do : „Producent_A”)\}.$

2.2.4. Wyrażenia relacyjne

Ograniczenia określają, które ze wszystkich możliwych populacji spełniających strukturę informacyjną są zawarte w **UoD** (2.46). Opisanie zależności między populacją w **UoD** a strukturą informacyjną wymaga wprowadzenia odpowiedniego aparatu matematycznego z dziedziny rachunku relacyjnego i rachunku krotek. Przedstawiony opis aparatu matematycznego wykonano na podstawie [15].

Relacyjna algebra $R(SI)$ (2.1) związana ze strukturą informacyjną składa się ze zbioru wyrażeń relacyjnych na tej strukturze.

Wyrażenie relacyjne r (wyprowadzone z typów faktów) w najprostszym przypadku jest typem faktu lub związane jest z relacyjnym operatorem zastosowanym do jednego lub wielu wyrażeń:

- jeżeli jest to typ faktu, np. $f = \{p, q\}$, to $Schemat(r)$ tego wyrażenia jest zbiorem ról należących do f

$$Schemat(f) = f = \{p, q\}, \quad (2.57)$$

- w przeciwnym przypadku $Schemat(r)$ jest wyprowadzany z typów faktów należących do wyrażenia.

Wartość wyrażenia jest obliczana przez operator Pop , który zawiera zbiór krotek (populację wyrażenia) spełniających $Schemat(r)$.

a) Podstawowe operatory relacyjne i ich semantyka

Przy założeniu, że $Schemat(r) = Schemat(s)$ mamy (2.41, 2.42, 2.46):

- suma $Pop(r \cup s) = Pop(r) \cup Pop(s)$, (2.58)

- różnica $Pop(r \setminus s) = Pop(r) \setminus Pop(s)$. (2.59)

Przy założeniu, że r, s są wyrażeniami relacyjnymi mamy:

- złączenia $\bowtie$ (2.57, 2.42, 2.46)

$$Schemat(r \bowtie s) = Schemat(r) \cup Schemat(s), \quad (2.60)$$

$$Pop((r \bowtie s)) = \{t \mid t[r] \in Pop(r) \wedge t[s] \in Pop(s)\}, \quad (2.61)$$

- projekcja π (2.57, 2.42, 2.46)

Oznacza to, że r, s są wyrażeniami, $p_1, \dots, p_n$ i $q_1, \dots, q_n$ są zbiorami ról takimi, że $p_i \sim q_i$, to wyrażenie relacyjne $\pi_{p_1:q_1 \dots p_n:q_n} r$ jest zdefiniowane jako

$$Schemat(\pi_{p_1:q_1 \dots p_n:q_n} r) = \{q_1, \dots, q_n\}, \quad (2.62)$$

$$Pop(\pi_{p_1:q_1 \dots p_n:q_n} r) = \{t \mid \exists t \in Pop(r) \forall 1 \leq i \leq n [t(q_i) = s(p_i)]\}. \quad (2.63)$$

Projekcja może służyć: do zastąpienia roli inną, do poszerzenia wyrażeń o nową rolę. Można zastąpić zapis $\pi_{p_1:p_1 \dots p_n:p_n} r$ zapisem $\pi_{p_1 \dots p_n} r$ lub jeżeli $\tau = q_1, \dots, q_n$ to $\pi_\tau r$;

- selekcja δ_F , gdzie F jest formułą selekcji (2.57, 2.42, 2.46)

$$Schemat(\delta_F r) = Schemat(r), \quad (2.64)$$

$$Pop((\delta_F r) = \{t \mid t \in Pop(r) \wedge F(t)\}, \quad (2.65)$$

- operator ekstensjonalny χ rozszerza wyrażenie r o nową rolę a , która wyznacza liczbę krotek, które posiadają równą wartość populacji należących do populacji zbioru τ (2.57, 2.42, 2.46)

$$Schemat(\chi(r, \tau, a)) = Schemat(r) \cup \{a\}, \quad (2.66)$$

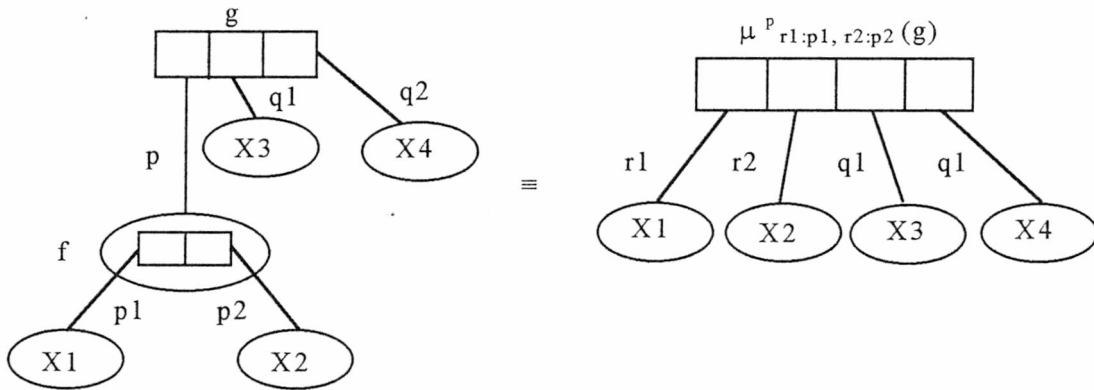
$$Pop(\chi(r, \tau, a)) = \{t \mid (t[Schemat(r)] \in Pop(r)) \wedge (t(a) = |\{s \mid s \in Pop(r) \wedge t[\tau] = s[\tau]\}|)\}, \quad (2.67)$$

- operator $\mu^P_{r1:p1, m:pn}(g)$ zastępujący zagnieżdżone typy faktów (typy obiektowe zagnieżdżone $TZag$) (2.1, 2.57, 2.42, 2.46, 2.3).

Niech f i g są wyrażeniami relacyjnymi i $p \in Schemat(g)$, $Baza(p) = Schemat(f)$. Niech $Schemat(f) = \{p_1, \dots, p_n\}$ i $\{r_1, \dots, r_n\}$ jest zbiorem ról nie występujących w $Schemat(g)$ oraz $Schemat(f)$. Niech $S = Schemat(g) \setminus \{p\}$. Wtedy mamy (rys. 2.10)

$$Schemat \mu^P_{r1:p1, m:pn}(g) = \{r_1, \dots, r_n\} \cup S, \quad (2.68)$$

$$Pop(\mu^P_{r1:p1, m:pn}(g)) = \{t \cup s[S] \mid t \in Pop(f) \wedge u \in Pop(g) \wedge u(p) = t\}. \quad (2.69)$$



Rys. 2.10. Przekształcanie zagnieżdżonego typu obiektowego f

b) Wyrażenia boolowskie

- wyrażenie wyznaczające pusty zbiór populacji danego wyrażenia relacyjnego

$$JestPusty(r) \equiv Pop(r) = \emptyset, \quad (2.70)$$

- różne wyrażenia relacyjne zawierające role pokrywające się mogą być ze sobą porównane. Oznacza to, że dla wyrażeń relacyjnych r i s funkcja ϕ jest pokryciem między $Schemat(r)$ i $Schemat(s)$, jeżeli ϕ jest bijekcją między $Schemat(r)$ i $Schemat(s)$ takie, że $\forall p \in Schemat(r) [p \sim \phi(p)]$,

- operator podzbioru (2.57, 2.42, 2.46)

$$r \subseteq_{\phi} s \Rightarrow \forall t \in Pop(r) \exists u \in Pop(s) \forall p \in Schemat(r) [t(p) = u(\phi(p))], \quad (2.71)$$

- operator równości (2.66)

$$r = \phi S \Leftrightarrow r \subseteq \phi S \wedge S \subseteq \phi^{-1} r, \quad (2.72)$$

- operator wykluczenia (2.57, 2.42, 2.46)

$$r \otimes \phi \Rightarrow \forall t \in Pop(r) \neg \exists u \in Pop(s) \forall p \in Schemat(r) [t(p) = u(\Phi(p))]. \quad (2.73)$$

c) Specjalne operatory

- operator liczby krotek wyrażenia relacyjnego r

$$n(r) = |Pop(r)|, \quad (2.74)$$

- operator liczby krotek o różnych wartościach na zbiorze ról τ

$$n(r, \tau) = n(\pi_\tau, r), \quad (2.75)$$

- operator zwracający ekstremalne wartości uporządkowanych wystąpień typów bazowych danej roli p

$$\min(r, p) = \min(Pop(\pi_p, r)), \max(r, p) = \max(Pop(\pi_p, r)), \quad (2.76)$$

- operator zakresu wartości wystąpień typów bazowych związanych z daną rolą p w wyrażeniu relacyjnym $r = \{p\}$

$$Pop(Zakres_p(r)) = \{x \mid \exists t \in Pop(r) [t(p) = x]\}. \quad (2.77)$$

2.2.5. Ograniczenia

Model może być opisany za pomocą:

- 1) zbioru stanów pokrywających strukturę informacyjną (populacja struktury informacyjnej),
- 2) zbioru tranzycji (zbiór akcji, które pochodzą od tych tranzycji).

Struktura informacyjna jest opisana jako zbiór elementarnych typów faktu nad zbiorem typów obiektowych. Stan odpowiada populacji czyli zbiorowi wystąpień tych typów faktów. Nie zawsze wszystkie elementy populacji odpowiadają każdemu ze stanów **UoD** (2.46) i wtedy są wykluczone za pomocą tzw. ograniczeń statycznych. Definicje ograniczeń są wykonywane za pośrednictwem wyrażeń algebry relacyjnej. Wyrażenia relacyjne budowane są z tych fragmentów struktury informacyjnej, które stawiają specjalne wymagania populacji. Stąd każde ograniczenie jest opisane:

- schematem wyrażenia relacyjnego zbudowanego z typów faktów należących do danego fragmentu struktury informacyjnej i odpowiednich operatorów relacyjnych,
- odpowiadającej temu wyrażeniu populacji spełniającej jednocześnie globalnie strukturę informacyjną SI (2.1) modelu.

Model konceptualny jest opisany jako

$$Model\ ORM = (SI, POP, Ograniczenia), \quad (2.78),$$

gdzie SI reprezentuje syntaktykę modelu (struktura informacyjna), natomiast $Ograniczenia$ semantykę modelu. Wyrażenie opisujące tę semantykę jest zdefiniowane jako (2.1, 2.42, 2.46)

$$JestPop(SI, Ograniczenia, Pop) \equiv JestPop(SI, Pop) \wedge \forall C \in Ograniczenia [Pop \models C]. \quad (2.79)$$

a) *Ograniczenie totalne*

Ograniczenie totalne oznacza, że populacja typów obiektowych związanych ze zbiorem τ danych ról jest równa całej populacji związanej funkcją *Baza* z każdą z tych ról. Zbiór ról posiadający taką własność należy więc do zbioru $total(\tau)$. Predykat opisujący to ograniczenie jest zdefiniowany jako

$$total \subseteq \phi(TRol). \quad (2.80)$$

Stąd populacja spełniająca to ograniczenie jest opisana jako

$$Pop \models total(\tau). \quad (2.81)$$

Ograniczenie to zawierające zbiór ról ma wtedy sens, jeżeli te role są relacyjne związane, czyli

$$Pop \models total(\tau) \Rightarrow \forall p, q \in \tau [p \sim q]. \quad (2.82)$$

Na podstawie (2.2, 2.3) mamy

$$\cup_{q \in \tau} Baza(q) = \cup_{q \in \tau} \pi_q (Fakt(q)), \quad (2.83)$$

$$\cup_{q \in \tau} Pop(Baza(q)) = \cup_{q \in \tau} Pop(\pi_q (Fakt(q))). \quad (2.84)$$

Lemat 2.3 (2.9, 2.3)

$$Połączona(Pop) \Leftrightarrow \forall a \in TA, Korzen(a) = a [Pop \models total(\{p \in TRol \mid Baza(p) = a\})].$$

Zbiór wszystkich i-zbiorów zawierających role spełniające ograniczenie *total* jest rodziną i-indeksowanych zbiorów $(\tau_i)_{i \in N}$ nad zbiorem wszystkich ról $TRol$ struktury informacyjnej i jest wyznaczony na podstawie zależności

$$RoleTotal = ((\tau_i)_{i \in N}) = \{\tau_i \mid i \in N\}. \quad \{*\} (2.85)$$

Przykłady (rys. 2.11):

$$a) Pop(Zakup) = \{zakup1, zakup2\},$$

$$\tau_1 = \{\pi_{ma} m_1, \pi_{ma} m_2\},$$

$$Zakup \cup Zakup = \pi_{ma} m_1 \cup \pi_{ma} m_2,$$

$$\cup_{q \in \tau_1} Pop(q) = \{zakup1, zakup2\} \cup \{zakup1, zakup2\} = \{zakup1, zakup2\},$$

$$\cup_{q \in \tau_1} Pop(\pi_q Fakt(q)) = Pop(\pi_{ma} m_1) \cup Pop(\pi_{ma} m_2) = \{zakup1, zakup2\} \cup \{zakup1, zakup2\} = \{zakup1, zakup2\};$$

$$b) \tau_2 = \{\in^s_{Rachunek}\},$$

$$\cup_{q \in \tau} Pop(q) = Pop(Elb(Baza(\in^e_{Rachunek}))) = \{zakup1, zakup2\},$$

$$\cup_{q \in \tau_1} Pop(\pi_q Fakt(q)) = Pop(\in^e_{Rachunek}) = \{zakup1, zakup2\};$$

$$\cup_{q \in tl} Pop(\pi_q Fakt(q)) = \{produkt2_1, produkt2_2\} \cup \{produkt3_1, produkt3_2\},$$

1.3 Częstotliwość wystąpień



b) Ograniczenie unikatowe

$$unikat \subseteq \phi(TRol). \quad (2.86)$$
$$Pop/ = \text{unikat}(\tau). \quad (2.87)$$

Ograniczenie unikatowe odgrywa dużą rolę integrującą strukturę informacyjną z **UoD** (2.46), zapewniając niepowtarzalne wartości populacji pewnych ról. Z ograniczeniem tym są związane takie podstawowe pojęcia jak: funkcjonalne zależności, klucze kandydujące oraz klucz główny. Pojęcia te odgrywają podstawową rolę przy identyfikacji typów obiektowych, która zostanie przedstawiona w dalszej części pracy (rozdział 2.2.6).

- *funkcjonalne zależności*

Oznaczają one powiązania funkcjonalne między populacją ról spełniających te zależności. Niech r jest wyrażeniem relacyjnym i niech zbiory ról: σ, τ należą do wyrażenia r , czyli (2.57)

$$\sigma, \tau \subseteq \text{Schemat}(r), \quad (2.88)$$

wtedy populacja Pop (2.42, 2.46, A26) ról ze zbioru τ jest funkcjonalnie zależna od populacji ról ze zbioru σ , czyli

$$\text{Pop} \models \sigma \rightarrow^r \tau. \quad (2.89)$$

W przypadku istnienia zależności funkcyjnych między zbiorami σ i τ , zbiór σ wyznacza populację różnowartościową, gdyż stanowi ona dziedzinę tej zależności funkcyjnej. Prawdziwy jest więc lematy 2.4 i 2.5.

Lemat 2.4 (2.89, 2.76, 266, 2.67) $\sigma \rightarrow^r \tau \Leftrightarrow \max(\chi(r, \sigma, a), a) = 1$.

Lemat 2.5 (2.57) $\text{Schemat}(r) \rightarrow^r \text{Schemat}(r)$.

- *klucze kandydujące*

Zbiór ról σ , który należy do wyrażenia r , czyli (2.57)

$$\sigma \subseteq \text{Schemat}(r) \quad (2.90)$$

nazywamy kluczami kandydującymi, jeżeli (2.57, 2.89)

$$\sigma \rightarrow^r \text{Schemat}(r) \quad (2.91)$$

i σ jest minimalnym zbiorem posiadającym następującą własność (2.89, 2.57)

$$\forall \tau [\sigma \rightarrow^r \text{Schemat}(r)] \Rightarrow \tau \not\subset \sigma]. \quad (2.92)$$

Klucze kandydujące są również łączone z pojęciem identyfikatora, który jest zdefiniowany jako *identyfikator*(r, σ). Identyfikator jednoznacznie określa populację wyrażenia relacyjnego r . Lemat 2.6 opisuje semantykę identyfikatora.

Lemat 2.6 (2.1) $\forall f \in \text{TFakt} \exists \sigma \subseteq f[\text{Pop} \models \text{identyfikator}(f, \sigma)]$.

- *unikatowe ograniczenie unikat*(σ)

Zbiór unikatowych ról należy do niepustego zbioru σ

$$\sigma \subseteq \text{TRol}. \quad (2.93)$$

Semantykę tego ograniczenia wyraża tzw. identyfikator zdefiniowany jako

$$\text{identyfikator}(\xi(\sigma), \sigma). \quad (2.94)$$

Wyrażenie relacyjne ξ jest zbudowane z elementów struktury informacyjnej i zbiór

ról σ jest identyfikatorem (lub też kluczem kandydującym) tego wyrażenia. Wyrażenie relacyjne ξ jest zdefiniowane jako (2.60, 2.61, 2.96, 2.97)

$$\xi(\sigma) = \delta_{C(\sigma)}(\text{>}</f \in \text{Fakty}(\sigma) f). \quad (2.95)$$

Funkcja *Fakty* wyznacza zbiór typów faktów związanych ze zbiorem unikatowych ról σ . Jest ona zdefiniowana jako (2.2)

$$\text{Fakty}(\sigma) = \{\text{Fakt}(p) / p \in \sigma\}. \quad (2.96)$$

Formuła selekcji $C(\sigma)$ określa sposób wyboru ról łączących role ze zbioru unikatowego, jeżeli nie należą one do jednego typu faktu. Jest ona zdefiniowana jako (A16-22, rozdz. 2.1.2))

$$C(\sigma) = \exists_{p,q \in \tau(\sigma), p \sim q} [p=q], \quad (2.97)$$

Zbiór $\tau(\sigma)$ jest zbiorem zdefiniowanym jako (2.96)

$$\tau(\sigma) = \cup \text{Fakty}(\sigma) \setminus \sigma \quad (2.98)$$

i oznacza zbiór ról łączących role ze zbioru ról unikatowych.

Całe wyrażenie relacyjne ξ wyznacza więc zbiór ról związanych ze zbiorem unikatowym. Jeżeli zbiór ten zawiera role objęte ograniczeniem *total* i *unikat*, wtedy zależności funkcyjne są różnowartościowe i identyfikator jest kluczem (2.95, 2.80, 2.86)

$$\text{identyfikator}(\xi(\sigma), \sigma) \equiv \text{klucz}((\xi(\sigma), \sigma), \text{jeżeli } \text{unikat}(\tau(\sigma)) \wedge \text{total}(\tau(\sigma)). \{*\} \quad (2.99)$$

Jeżeli istnieje kilka identyfikatorów czyli kilka zbiorów ról o własności klucza kandydującego dla danego wyrażenia relacyjnego r , wtedy wybrany zbiór pełni rolę klucza głównego. Taki zbiór ról określa się jako *gunikat*. Identyfikuje on jednoznacznie populację z odpowiadającym jej typem obiektowym. Między ograniczeniami *unikat* i *gunikat* zachodzi zależność (2.86)

$$\text{gunikat}(\sigma) \Rightarrow \text{unikat}(\sigma). \quad (2.100)$$

Rozróżnia się dwa przypadki wyznaczania ścieżek łączących role należące do zbioru ról unikatowych (2.95, 2.86):

- jeżeli zbiór ról unikatowych spełnia $\sigma \subseteq f$, wtedy $\xi(\sigma) = f$, co oznacza, że zbiór ról łączących należy do jednego typu faktu f ,
- jeżeli zbiór ról unikatowych należy do więcej niż jednego typu faktu, należy wyznaczyć ścieżkę przebiegającą przez wszystkie typy obiektowe połączone przez typy faktów, zawierających role unikatowe. Definicja tego połączenia ma postać (2.96, 2.2)

$$\text{Połączenie}(\sigma) \equiv |\text{Fakty}(\sigma)| > 1 \Rightarrow \exists_{p \in \sigma} [\exists_{q \in \sigma \setminus \text{Fakt}(p)} [(\text{Fakt}(p) \setminus \sigma) \sim (\text{Fakt}(q) \setminus \sigma)] \wedge \text{Połączenie}(\sigma \setminus \text{Fakt}(p))]. \quad (2.101)$$

Przedstawiony zbiór ról unikatowych związanych z identyfikatorem służy do jednoznacznej identyfikacji populacji spełniającej zbiór ról unikatowych.

Zbiór wszystkich i -zbiorów zawierających role typu *unikat* jest rodziną i -indeksowanych zbiorów $\text{unikat}(\sigma_i)_{i \in TA}$ nad zbiorem wszystkich ról *TRol* struktury informacyjnej i jest wyznaczony na podstawie zależności

$$RoleUnikat = ((\sigma_i)_{i \in TA}) = \{\sigma_i / i \in N\}. \quad \{*\}(2.102)$$

Przykłady (rys. 2.11 - 2.96, 2.60, 2.61, 2.98, 2.95, 2.99, 2.102):

- a) $Fakty(\sigma_{Produkt_1}) = \{m_3, m_4\}$,
 $\sigma_{Produkt_1} = \{\pi \text{ należy do } m_3, \pi \text{ należy do } m_4\}$,
 $\tau(\sigma_{Produkt_1}) = \{\pi \text{ ma } m_3, \pi \text{ ma } m_4\}$,
 $\xi(\sigma_{Zakup}) = \delta_{\pi \text{ ma } m_3 = \pi \text{ ma } m_4} (m_3 /></ m_4)$,
 $identyfikator(\delta_{\pi \text{ ma } m_3 = \pi \text{ ma } m_4} (m_3 /></ m_4), \{\pi \text{ należy do } m_3, \pi \text{ należy do } m_4\})$,
- b) $Fakty(\sigma_{Zakup}) = \{m_1\}$,
 $\sigma_{Zakup} = \{\pi \text{ należy do } m_1\}$,
 $\tau(\sigma_{Zakup}) = \{\pi \text{ ma } m_1\}$,
 $\xi(\sigma_{Zakup}) = m_1$,
 $identyfikator(m_1, \{\pi \text{ ma, należy do}\})$,
- c) $RoleUnikat = ((\sigma_i)_{i \in TA}) \equiv \{\{\pi \text{ należy do } m_3, \pi \text{ należy do } m_4\}, \{\pi \text{ należy do } m_1\}\}$.

c) Ograniczenie określające częstotliwość wystąpień

Ograniczenie *Częstotliwość* określa maksymalną m i minimalną n liczbę wystąpień danego typu obiektowego w danym zbiorze ról σ (dla ograniczenia *unikat* dany obiekt wystąpi w danej roli tylko raz). Definicja tego ograniczenia jest następująca: *Częstotliwość* (σ, n, m) . Semantyka tego ograniczenia jest określona jako (2.70, 2.66, 2.67, 2.76, 2.95)

$$\neg \text{Jest Pusta}(\xi(\sigma)) \Rightarrow \begin{cases} \min(\chi(\xi(\sigma), \sigma, a), a) \geq n, \\ \max(\chi(\xi(\sigma), \sigma, a), a) \leq m. \end{cases} \quad (2.103)$$

Jeżeli dany zbiór ról pokrywa się ze zbiorem ról unikatowych, wtedy mamy zależność (2.86)

$$unikat(\sigma) \Leftrightarrow \text{częstotliwość}(\sigma, 0, 1) \Leftrightarrow \text{częstotliwość}(\sigma, 1, 1). \quad (2.104)$$

Zbiór wszystkich i -zbiorów zawierających role typu *częstotliwość* jest rodziną i -indeksowanych zbiorów $(\sigma_i)_{i \in N}$ nad zbiorem wszystkich ról $TRol$ struktury informacyjnej i jest wyznaczony na podstawie zależności

$$RoleCzęstotl = ((\sigma_i)_{i \in N}) = \{\sigma_i / i \in N\}. \quad \{*\}(2.105)$$

Przykład (z rys. 2.11; 2.103, 2.105):

- a) $\sigma_1 = \{\in^{Rachunek}_{Rachunek}\}$,
b) $\text{częstotliwość}(\sigma, 1, 3)$,
c) $\sigma_2 = \{\pi \text{ należy do } m_3, \pi \text{ należy do } m_4\}$,
d) $\text{częstotliwość}(\sigma, 1, 1)$,
e) $RoleCzęstotl = ((\sigma_i)_{i \in 2}) = \{\{\in^{Rachunek}_{Rachunek}\}, \{\pi \text{ należy do } m_3, \pi \text{ należy do } m_4\}\}$.

d) Ograniczenia zbiorowe

Ograniczenia zbiorowe określają zależności między zbiorami ról. Podkreślanie związku między rolami na schemacie konceptualnym daje ważną informację o cechach populacji spełniających strukturę informacyjną.

- *Sekwencja ról*

Przydatnym ograniczeniem przekształcającym zbiór ról σ w indeksowaną sekwencję z ról jest następujący operator *Zbiór*

$$Zbiór(z) = \{x \mid x \in \sigma \wedge (\exists_i [z[i] = x])\}, \quad (2.106)$$

- *Numer roli*

Operator ten zwraca element znajdujący się na i -tej pozycji sekwencji ról jest zdefiniowany jako

$$Poz(i, z) = z[i] \mid z[i] \in \sigma. \quad (2.107)$$

Innym typem ograniczeń określających częstość wystąpień są tzw. ograniczenia zbiorowe:

- *Podzbiór _{ϕ} (σ, τ)*

Ograniczenie to oznacza, że populacja jednej (σ) z dwóch sekwencji ról jest podzbiorem populacji drugiej sekwencji ról (τ)

$$Podzbiór \subseteq TRol \times TRol. \quad (2.108)$$

Interpretacja tego ograniczenia jest zdefiniowana jako (2.62, 2.63, 2.95, 2.72)

$$\pi_\sigma(\xi(\sigma)) \subseteq_\phi \pi_\tau(\xi(\tau)), \quad (2.109)$$

- *Równość _{ϕ} (σ, τ)*

Ograniczenie to oznacza, że istnieje równość populacji dwóch sekwencji ról (σ, τ)

$$Równość \subseteq TRol \times TRol. \quad (2.110)$$

Interpretacja tego ograniczenia jest zdefiniowana jako (2.62, 2.63, 2.72, 2.95)

$$\pi_\sigma(\xi(\sigma)) =_\phi \pi_\tau(\xi(\tau)), \quad (2.111)$$

- *Rozłączność _{ϕ} (σ, τ)*

Ograniczenie to oznacza, że populacja danej sekwencji ról (σ) nie zawiera nawet jednej krotki z populacji drugiej sekwencji ról (τ)

$$Rozłączność \subseteq Rol \times Rol. \quad (2.112)$$

Interpretacja tego ograniczenia jest zdefiniowana jako (2.62, 2.63, 2.73, 2.95)

$$\pi_\sigma(\xi(\sigma)) \otimes_\phi \pi_\tau(\xi(\tau)). \quad (2.113)$$

Ograniczenia zbiorowe można przedstawić w postaci i -indeksowanych rodzin zbiorów nad zbiorem *TRol*:

- zbiór wszystkich i -zbiorów zawierających role typu *Podzbiór* jest rodziną

i -indeksowanych zbiorów $Podzbiór_{\phi}((\sigma, \tau)_{i \in N})$ nad zbiorem wszystkich ról $TRol$ struktury informacyjnej i jest wyznaczony na podstawie zależności

$$RolePodzbiór((\sigma, \tau)_{i \in N}) = \{(\sigma, \tau)_i / i \in N\}, \quad \{*\} (2.114)$$

- zbiór wszystkich i -zbiorów zawierających role typu *Równość* jest rodziną i -indeksowanych zbiorów $Równość_{\phi}((\sigma, \tau)_{i \in N})$ nad zbiorem wszystkich ról $TRol$ struktury informacyjnej i jest wyznaczony na podstawie zależności

$$RoleRówność((\sigma, \tau)_{i \in N}) = \{(\sigma, \tau)_i / i \in N\}, \quad \{*\} (2.115)$$

- zbiór wszystkich i -zbiorów zawierających role typu *Rozłączność* jest rodziną i -indeksowanych zbiorów $Rozłączność_{\phi}((\sigma, \tau)_{i \in N})$ nad zbiorem wszystkich ról $TRol$ struktury informacyjnej i jest wyznaczony na podstawie zależności

$$RoleRozłączność((\sigma, \tau)_{i \in N}) = \{(\sigma, \tau)_i / i \in N\}. \quad \{*\} (2.116)$$

Przykład (z rys. 2.11):

$$\sigma = \{\pi_{oblicza} f_4\},$$

$$\tau = \{\pi_{ma} f_3\},$$

$$Równość(\{\pi_{oblicza} f_4\}, \{\pi_{ma} f_3\}),$$

$$Pop(Baza(\pi_{oblicza} f_4)) = Pop(Baza(\pi_{ma} f_3)).$$

2.2.6. Identyfikacja typów obiektowych

Jest to proces łączenia populacji z odpowiadającymi jej elementami struktury informacyjnej. Rozróżnia się identyfikację niestrukturalną, którą realizuje się na poziomie wystąpień i identyfikację strukturalną opartą na własnościach struktury informacyjnej modelu.

a) Identyfikacja niestrukturalna

Identyfikacja niestrukturalna zakłada, że każdy typ obiektowy jest dostępny w sposób jednoznaczny. Proces identyfikowania wystąpień poszczególnych typów obiektowych jest związany z ich właściwościami, czyli „możliwościami” występowania w określonych rolach.

Identyfikacja typów atrybutowych jest bezpośrednia tzn. te typy obiektowe identyfikują same siebie. Identyczne wartości typów atrybutowych są określane za pomocą operatora równości.

Typy abstrakcyjne są reprezentowane przez swoje właściwości tzn. jeżeli pewne wystąpienia mają te same właściwości, to uznaje się je za identyczne, gdyż nie można ich rozróżnić. Tym samym identyfikują ten sam typ abstrakcyjny. Właściwości typu obiektowego wynikają z typów faktów, w których dany obiekt odgrywa rolę. Stąd identyfikacja wystąpień typów obiektowych jest opisana wyrażeniem (2.1, 2.2, 2.3, 2.118, A26)

$$\forall_{p \in TRol, x, y \in Pop(Baza(p))} \forall_{q \in Fakt(p)} [Identyczne(x, y, p, q)] \Rightarrow x = y, \quad (2.117)$$

gdzie (2.119, A26)

$$Identyczne(x, y, p, q) \equiv \forall_{z \in Pop(Baza(q))} [Rel_{Pop}(x, p, q, z) \Leftrightarrow Rel_{Pop}(y, p, q, z)] \quad (2.118)$$

oraz (2.2, 2.3, A26)

$$Rel_{Pop}(x, p, q, z) \equiv \exists t \in Pop(Fakt(p)) [t(p) = x \wedge t(q) = z]. \quad (2.119)$$

Wystąpienia typów obiektowych zagnieżdżonych są reprezentowane przez krotki, stąd są przez nie identyfikowane. W rezultacie w każdej populacji istnieją takie wystąpienia t i s typu faktu f , że prawdziwe jest wyrażenie

$$\forall p \in f [s(p) = t(p)] \Rightarrow s = t. \quad (2.120)$$

Identyfikacja niestrukturalna nie gwarantuje połączenia struktury informacyjnej z populacją, gdyż opiera się na zgodności nazw wystąpień, co nie zawsze może okazać się prawdziwe.

b) Identyfikacja strukturalna

W identyfikacji strukturalnej zakłada się, że każdy typ obiektowy jest dostępny w sposób jednoznaczny dopiero na podstawie właściwości struktury informacyjnej.

Właściwości struktury informacyjnej, które nie dopuszczają do tego, aby istniał taki typ obiektowy, który jest nierozpoznawalny („wiszący”), są następujące:

- każdy typ atrybutowy musi wystąpić w ograniczeniach typu *total* (2.1, 2.80, 2.3)

$$\forall x \in TWart \exists p \in TRol \exists total(\tau) \in Ograniczenia [Baza(p) = x \wedge p \in \tau], \quad (2.121)$$

- wszystkie typy obiektowe są identyfikowane, jeżeli są możliwe do zidentyfikowania

$$\forall x \in TA [JestIdF(x)]. \quad (2.122)$$

Identyczność wystąpień jest orzekana wtedy, gdy nie można ich wzajemnie odróżnić.

Predykat *JestIdF* jest zdefiniowany jako:

1. Jeżeli x jest typem atrybutowym, zawsze jest prawdziwa zależność (2.122)

$$JestIdF(x). \quad (2.123)$$

2. Jeżeli x jest typem zagnieżdżonym (lub zbiorem ról), wtedy spełnia *JestIdF(x)*, jeżeli wszystkie komponenty spełniają *JestIdF*, czyli (2.122, 2.3)

$$\forall p \in x [JestIdF(Baza(p))]. \quad (2.124)$$

3. Jeżeli jest typem zbiorowym lub sekwencyjnym, to mamy (2.122, 2.5)

$$JestIdF(Elit(x)). \quad (2.125)$$

4. Jeżeli x jest typem schematowym, to mamy (2.6, 2.122)

$$\forall x \in Schem y [JestIdF(y)]. \quad (2.126)$$

5. Jeżeli x jest typem abstrakcyjnym, wtedy zachodzą następujące przypadki:

- 5.1. Jeżeli x jest podtypem i spełnia relację $x \text{ Spec } y$ - wtedy x spełnia *JestIdF(x)*, gdy spełnione są zależności: $Korzen(x) \neq x$ oraz *JestIdF(Korzen(x))*, oraz istnieje unikatowa reguła specjalizacji. Identyfikacja typów *Korzen(x)* przebiega tak, jak dla typów elementarnych według punktu 5.4 (2.7, 2.122, 2.9).

- 5.2. Jeżeli x jest w relacji polimorfizmu, czyli $x \text{ Pol } y$ i nie jest w relacji Spec , wtedy x jest identyfikowalne przez typy obiektowe spełniające warunki

$$\text{JestIdF}(x) \Leftrightarrow \forall y \in \text{Liście}(x) [\text{JestIdF}(y)]. \quad \{*\} (2.127)$$

Identyfikacja typów y przebiega tak, jak dla typów elementarnych według punktu 5.4.

- 5.3. Jeżeli x jest typem polimorficznym, $x \in \text{TPol}$, o strukturze wynikającej ze specjalizacji, czyli $x \text{ Pol } y$ oraz $y \text{ Spec } x$, wtedy proces identyfikacji jest złożeniem procesu opisanego w punkcie 5.1 i 5.2. Wtedy mamy (2.30, 2.122, 2.9)

$$\text{JestIdF}(x) \Leftrightarrow \forall y \in \text{Liście}(x) [\text{JestIdF}(y) \Leftrightarrow \text{JestIdF}(\text{Korzeń}(y))] \wedge (x \in \text{TPol}). \quad \{*\} (2.128)$$

Identyfikacja typów $\text{Korzeń}(y)$ przebiega tak, jak dla typów elementarnych według punktu 5.4.

- 5.4. Jeżeli x jest typem elementarnym $x \in \text{TE}$, wtedy x jest identyfikowalny, jeżeli istnieje zbiór ról τ , który może być wykorzystany do identyfikacji. Zbiór ten jest identyfikatorem typu obiektowego, czyli mamy (2.122)

$$\text{JestIdF}(\tau),$$

a więc istnieje możliwość złożonej identyfikacji (2.122, 2.3, 2.86, 2.96, 2.80, 2.2)

$$\bullet \quad \forall p \in \tau [\text{JestIdF}(\text{Baza}(p))], \quad (2.129)$$

$$\bullet \quad \text{unikat}(\tau),$$

$$\bullet \quad \forall f \in \text{Fakty}(\tau) \exists p \in \text{KoRole}(\tau, x) \cap f [\text{unikat}(p) \wedge \text{total}(p)], \quad (2.130)$$

gdzie (2.96, 2.3)

$$\text{KoRole}((\tau, x) = \{p \in \cup (\text{Fakty}(\tau) \setminus \tau) / \text{Baza}(p) = x\}, \quad (2.131)$$

czyli wszystkie role łączące identyfikowany typ obiektowy z rolami identyfikującymi ze zbioru τ , objęte ograniczeniami typu *total* i *unikat*.

Relacja

$$\text{IdF} \subseteq \text{TOb} \times \text{TRol} \quad \{*\} (2.132)$$

określa związek między identyfikowanym typem obiektowym i zbiorem identyfikującym τ .

Mamy więc

$$x \text{ IdF } \tau. \quad \{*\} (2.133)$$

Na podstawie punktu 5.4 można zdefiniować ścieżkę łączącą wszystkie role należące do typów faktów wyznaczonych przez zbiór identyfikujący τ . Ścieżka ta spełnia warunki wyznaczone przez zależność *Połączenie*(τ) (2.101) i dodatkowo (2.96, 2.101, 2.131)

$$\begin{aligned} \text{RoleIdF}(x, \tau) = \{p / p \in \text{Fakty}(\tau) \wedge \text{Połączenie}(\tau) \wedge \\ \forall p \notin \tau [p \in \text{KoRole}(x, \tau) \wedge \text{unikat}(p) \wedge \text{total}(p)]\}. \quad \{*\} (2.134) \end{aligned}$$

Jeżeli ścieżka powstaje w wyniku identyfikacji złożonej, wtedy ścieżka wynikowa jest sumą ścieżek cząstkowych (2.134, 2.131)

$$\text{SumaRoleIdF}((x_i)_{i \in n}, (\tau_i)_{i \in n}) = \{p \in \bigcup_{i=1}^n \text{RoleIdF}(x_i, \tau_i) / \\ \exists p \in \tau_i \exists q \in \text{KoRole}(x_{i+1}, \tau_{i+1}) [(p \sim q) \wedge \text{total}(q) \wedge \text{unikat}(p)]\}, \{*\} \quad (2.135)$$

gdzie τ_i jest zbiorem identyfikującym i -tego abstrakcyjnego typu obiektowego występującego w identyfikacji złożonej. Suma ścieżek również spełnia warunki wyznaczone przez zależność $\text{Połączenie}(\tau)$ (2.101). Oznacza to, że istnieje przynajmniej jedna rola p w ścieżce i oraz rola q w ścieżce $i+1$ taka, że spełniona jest między nimi zależność $p \sim q$ (A16-22 w rozdz. 2.1.2).

Role opisane wyrażeniami RoleIdF (2.134) i SumaRoleIdF (2.135) mogą być wyznaczone za pomocą wyrażenia relacyjnego ξ (2.95), przy czym role wyznaczone operatorem selekcji C (2.97) są objęte ograniczeniami total (2.80) i unikat (2.86).

Identyfikacja populacji struktury informacyjnej jest opisana następującymi twierdzeniami (2.122, 2.1, 2.41, 2.42, 2.46, A23-35).

Twierdzenie 2.1. $\forall x \in TA [\text{JestIdF}(x)] \Rightarrow \forall x \in TOB [\text{JestIdF}(x)].$

Niech struktura informacyjna zawiera identyfikowalne wszystkie typy abstrakcyjne i jest acykliczna (2.143). Z definicji identyfikowalności pozostałych typów obiektowych wszystkie pozostałe typy obiektowe są identyfikowalne

Twierdzenie 2.2.

$SI \text{ jest identyfikowalna} \Rightarrow \text{cała populacja jest niestukturalnie identyfikowalna.}$

Jeżeli SI jest identyfikowalna strukturalnie, wtedy istnieje unikatowy klucz τ identyfikujący dany typ obiektowy x , stąd wystąpienia np. o_1 i o_2 są powiązane przez relacje zawierające role klucza z tym samym typem obiektowym. Stąd można wnioskować, że oba wystąpienia mają te same właściwości, czyli są niestukturalnie identyfikowalne.

Twierdzenie 2.3.

$\text{JestIdF}(SI, \text{Ograniczenia}) \Rightarrow \forall_{Pop} [\text{JestPop}(SI, \text{Ograniczenia}, POP) \Rightarrow \text{JestPołączona}(POP)].$

Oznacza to, że istnieje zbiór ról τ (należący do ograniczeń) taki, że dla każdej roli p należącej do τ mamy $\text{Baza}(p) = a$, gdzie a należy do identyfikowalnych typów obiektowych SI . Stąd populacja jest połączona ze strukturą informacyjną.

Twierdzenie 2.4. $\forall x \in TE [\text{JestIdF}(x)] \Rightarrow \forall x \in TOB \exists y \in TWart [\text{JestIdF}(y) \Rightarrow \text{JestIdF}(x)].$
 $\{*\}$

Niech struktura informacyjna zawiera identyfikowalne wszystkie typy elementarne i jest acykliczna (2.143). Z definicji identyfikowalności typów obiektowych wszystkie pozostałe typy obiektowe (zbudowane na bazie identyfikowalnych typów elementarnych) są identyfikowalne. Jednocześnie z założenia wynika, że istnieją takie typy obiektowe, które identyfikują same siebie, tak aby proces identyfikacji złożonej typów elementarnych był skończony. Skończoność procesu identyfikacji jest warunkiem identyfikowalności każdego typu obiektowego. Zapewniają to typy atrybutowe $TWart$ (2.1).

Funkcja

$$AtrybutyIdF: \varphi(TOb, TRole) \rightarrow \varphi(TWart) \quad \{*\}(2.136)$$

wyznacza zbiór typów atrybutowych dla danego zbioru ról identyfikujących τ , pod warunkiem, że nie jest to proces zagnieżdżony (złożony) (2.1, 2.134, 2.3)

$$AtrybutyIdF(x, \tau) = \{y/y \in TWart \wedge \exists p \in RoleIdF(x, \tau) [Baza(p) = y]\}. \quad \{*\}(2.137)$$

Jeżeli identyfikacja danego typu obiektowego ma charakter złożony, wtedy zakłada się, że każdy typ obiektowy x_i jest identyfikowalny, czyli istnieje dla dowolnego typu x_i zbiór identyfikujący τ_i . Stąd można wyznaczyć zbiór atrybutów identyfikujących na podstawie ścieżki identyfikującej określonej przez $SumaRoleIdF$ (2.1, 2.135, 2.3)

$$AtrybutyIdF'((x_i)_{i \in n}, (\tau_i)_{i \in n}) = \{y/y \in TWart \wedge \exists p \in SumaRoleIdF((x_i)_{i \in n}, (\tau_i)_{i \in n}) [Baza(p) = y]\}. \quad \{*\}(2.138)$$

W przypadku identyfikacji złożonej danego typu abstrakcyjnego x_l można wyznaczyć wypadkowy zbiór identyfikujący, który jest sumą zbiorów identyfikujących τ_i wszystkich typów obiektowych abstrakcyjnych x_i biorących udział w identyfikacji złożonej. Zbiory identyfikujące τ_i spełniają następującą zależność: istnieje przynajmniej jedna rola p w τ_i oraz rola q w zbiorze $RoleIdF(\tau_{i+1})$ (2.134) taka, że spełniona jest między nimi zależność $p \sim q$ oraz $Baza(q) \sim Baza(p)$ (A16-22 w rozdz. 2.1.2, 2.3). Suma zbiorów identyfikujących również spełnia warunki wyznaczone przez zależność $Połączenie(\tau)$ (2.101). Oznacza to, że (2.134, 2.131, 2.3)

$$SumaIdF(x_l, (\tau_i)_{i \in n}) = \{p \in \cup_{i=1..n} \tau_i / \exists p \in RoleIdF(x_l, \tau_i) \exists q \in KoERole(x_{l+1}, \tau_{i+1}) [Baza(p) \sim Baza(q)]\}. \quad \{*\}(2.139)$$

Dla $i = 1$ zbiór identyfikujący jest określony jako

$$SumaIdF(x, \tau) = \tau. \quad \{*\}(2.140)$$

Stąd można wyznaczyć ścieżkę $RoleIdF'$ (2.96, 2.139, 2.101, 2.131)

$$\begin{aligned} RoleIdF'((x_l), (\tau_i)_{i \in n}) = \\ \{p/p \in Fakty(SumaIdF(x_l, (\tau_i)_{i \in n})) \\ \wedge Połączenie(SumaIdF(x_l, (\tau_i)_{i \in n})) \\ \wedge \forall p \notin SumaIdF(x_l, (\tau_i)_{i \in n}) [p \in KoRole(x_l, SumaIdF(x_l, (\tau_i)_{i \in n}))]\} \}. \end{aligned} \quad \{*\}(2.141)$$

Zbiór atrybutów identyfikujących jest teraz zdefiniowany jako (2.1., 2.141, 2.3)

$$AtrybutyIdF''((x_l), (\tau_i)_{i \in n}) = \{y/y \in TWart \wedge \exists p \in RoleIdF'((x_l), (\tau_i)_{i \in n}) [Baza(p) = y]\}. \quad \{*\}(2.142)$$

Proces identyfikacji strukturalnej polega więc na wyznaczeniu za pomocą ścieżki identyfikacyjnej zbioru jedno- lub wieloelementowego wystąpień typów atrybutowych związanych za pomocą tej ścieżki. Jeżeli jest to jeden atrybut, zwykle nie jest ta ścieżka zaznaczana na schemacie, lecz nazwa typu atrybutowego jest umieszczana przy oznaczeniu typu obiektowego. W przeciwnym przypadku na schemacie jest zaznaczony wypad-

kowy zbiór ról identyfikujących (połączenia linią przerywaną, oznaczone przez U).

Na rys. 2.11 przedstawiono przykład prostej identyfikacji typu *Zakup* oraz *Produkt_1*, natomiast na rys. 2.12 pokazano przykład identyfikacji złożonej typu *Numer*. W przykładzie dotyczącym rys. 2.12 przedstawiono dwa sposoby identyfikowania. Pierwszy sposób stanowi złożenie identyfikacji trzech typów: *Numer ulicy*, *Ulica*, *Miasto*, stanowiące schemat identyfikacji złożonej typu *Numer ulicy*. Drugi sposób polega na wyznaczeniu wypadkowego zbioru identyfikującego *Suma* i wykonaniu identyfikacji prostej (bez zagnieźdzeń).

Na rys. 2.11 ścieżka identyfikacji dla typu obiektowego *Produkt_1* jest wyznaczona następująco:

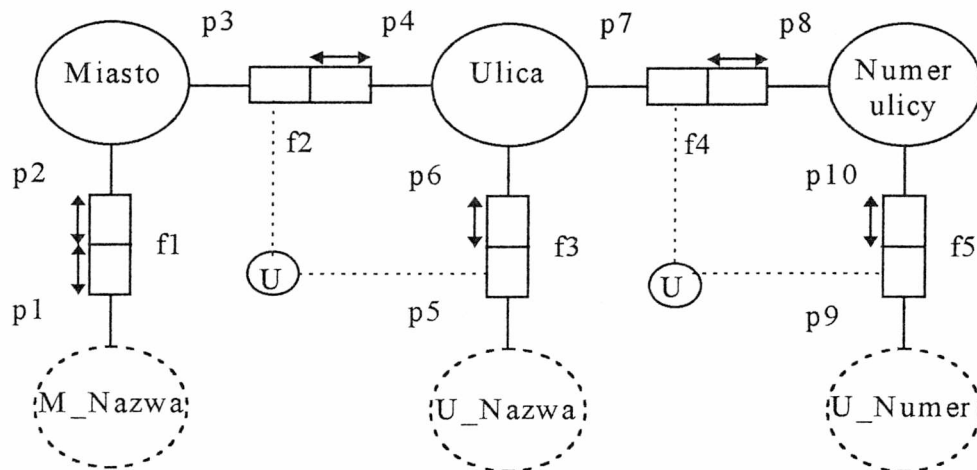
- $\tau = \{\pi \text{ należy do } m_3, \pi \text{ należy do } m_4\}$ - zbiór typu *unikat*(τ);
- zbiór $\sigma = \{\pi \text{ należy do } m_3, \pi \text{ należy do } m_4\}$ jest identyfikowalny, gdyż *Baza*($\pi \text{ należy do } m_3$) oraz *Baza*($\pi \text{ należy do } m_4$) są typami atrybutowymi, czyli zawsze identyfikowalnymi (2.131);
- $KoRole(\tau, Produkt_1) = \{\pi_{ma} m_3, \pi_{ma} m_4\}$;
- zbiór ról łączących spełnia (2.86, 2.80)

$$unikat(\pi_{ma} m_3), total(\pi_{ma} m_3), unikat(\pi_{ma} m_4), total(\pi_{ma} m_4).$$

Stąd ścieżka identyfikująca strukturalnie typ obiektowy *Produkt_1* jest równa (2.134, 2.136)

$$RoleIdF(Produkt_1, \tau) = \{\pi \text{ należy do } m_3, \pi_{ma} m_3, \pi_{ma} m_4, \pi \text{ należy do } m_4\},$$

$$AtrybutyIdF((Produkt_1, \tau) = \{Nazwa, Cena\}.$$



Rys. 2.12. Przykład złożonej identyfikacji

Na rys. 2.12 wg [15] przedstawiono przykład złożonej identyfikacji.

Sposób 1

- identyfikacja typu obiektowego *Numer ulicy* jest następująca (2.131, 2.80, 2.86, 2.123, 2.3, 2.129):

$$\tau_1 = \{p9, p7\},$$

$KoRole(\tau_1, Numer\ ulicy) = \{p10, p8\}$ oraz $unikat(p8), unikat(p10), total(p8), total(p10),$

$JestIdF(p9) = True$, gdyż $Baza(p9) = Numer$ jest typem atrybutowym,

$JestIdF(p7)$ wymaga spełnienia $JestIdF(Baza(p7))$, czyli identyfikacji typu obiektowego $Ulica$,

- identyfikacja typu obiektowego $Ulica$ jest następująca (2.131, 2.80, 2.86, 2.123, 2.3, 2.129):

$$\tau_2 = \{p5, p3\},$$

$KoRole(\tau, Ulica) = \{p6, p4\}$ oraz $unikat(p4), unikat(p6), total(p4), total(p6),$

$JestIdF(p5) = True$, gdyż $Baza(p5) = U_Nazwa$ jest typem atrybutowym,

$JestIdF(p3)$ wymaga spełnienia $JestIdF(Baza(p3))$, czyli identyfikacji typu obiektowego $Miasto$,

- identyfikacja typu obiektowego $Miasto$ jest następująca (2.131, 2.80, 2.86, 2.123, 2.3, 2.135, 2.138):

$$\tau_3 = \{p1\},$$

$KoRole(\tau, Miasto) = \{p2\}$ oraz $unikat(p2), unikat(p2),$

$JestIdF(p1) = True$, gdyż $Baza(p1) = M_Nazwa$ jest typem atrybutowym

$$SumaRoleIdF(\{Numer\ ulicy\}, \{Ulica\}, \{Miasto\}, \{p9, p7\}, \{p5, p3\}, \{p1\}) = \{p9, p10, p8, p7\} \cup \{p5, p6, p4, p3\} \cup \{p2, p1\},$$

$$AtrybutyIdF'(\{Numer_ulicy\}, \{Ulica\}, \{Miasto\}, \{p9, p7\}, \{p5, p3\}, \{p1\}) = \{U_Numer, U_Nazwa, M_Nazwa\}.$$

Sposób 2 (2.139, 2.3, 2.11, 2.141, 2.142)

$$\tau_1 = \{p9, p7\},$$

$$\tau_2 = \{p5, p3\},$$

$$\tau_3 = \{p1\},$$

$$SumaIdF(Numer_ulicy, \{p9, p7\}, \{p5, p3\}, \{p1\}) = \{p9, p7, p5, p3, p1\},$$

gdź

$$Baza(p7) \sim Baza(p6) \text{ oraz}$$

$$Inne_Role(p5) = \{p6\},$$

$$RoleIdF'(Numer_ulicy, \{p9, p7\}, \{p5, p3\}, \{p1\}) =$$

$$\{p9, p10, p8, p7, p6, p4, p5, p3, p2, p1\},$$

gdź

$$Baza(p3) \sim Baza(p2) \text{ oraz}$$

$$Inne_Role(p1) = \{p2\},$$

$$AtrybutyIdF''(Numer_ulicy, \{p9, p7\}, \{p5, p3\}, \{p1\}) = \{U_Numer, U_Nazwa, M_Nazwa\}.$$

c) Cykliczna struktura typów obiektowych

Obiekty o strukturze cyklicznej nie są identyfikowalne strukturalnie, natomiast są nie-strukturalnie identyfikowalne. Następujące twierdzenie określa warunki istnienia cykli

w strukturze informacyjnej (2.1, 2.3, 2.5, 2.8 i rozdz. 2.1.2):

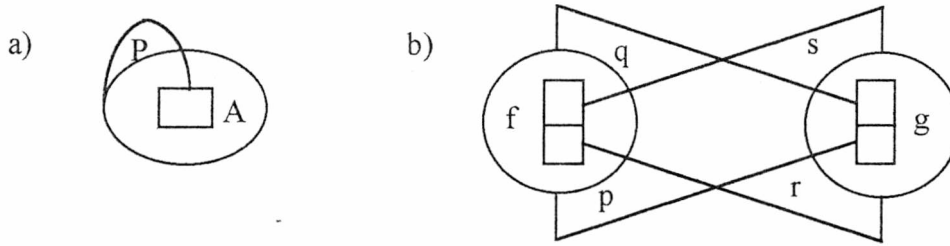
Twierdzenie 2.5 Struktura informacyjna nie zawiera cykli, gdy istnieje funkcja

$$h : TOB \rightarrow N, \quad (2.143)$$

oraz

- $f \in TFakt \wedge p \in f \Rightarrow h(f) > h(Baza(p))$,
- $g \in TZb \Rightarrow h(g) > h(Elt(g))$,
- $x Pol y \Rightarrow \exists y \in TOB [x Pol y \wedge h(x) > h(y)]$.

Funkcja h numeruje wszystkie typy obiektowe i następnie w czasie identyfikacji muszą być spełnione warunki uporządkowania malejącego.



Rys. 2.13. Typy obiektowe cykliczne

Przykład

Typ obiektowy z rys. 2.13 a jest cykliczny, gdyż prawdziwe są zależności (2.3, 2.2)

$$Baza(p) = Fakt(p) = A.$$

Stąd (2.42, 2.46, A26, 2.3)

$$Pop(Baza(p)) = \{x\} \text{ i } Pop(Fakt(p)) = \{x\}.$$

Na rys. 2.13 b mamy (2.3)

$$Baza(p) = Baza(q) = f,$$

$$Baza(r) = Baza(s) = g.$$

Stąd (2.42, 2.46, A26, 2.3)

$$Pop(f) = \{x\} = Pop(Baza(p)) = Pop(Baza(q)),$$

i podobnie (2.42, 2.46, A26, 2.3)

$$Pop(g) = \{y\} = Pop(Baza(r)) = Pop(Baza(s)).$$

Jednak strukturalnie nie są identyfikowalne, gdyż zawierają cykle, co jest niezgodne z definicją identyfikacji strukturalnej.

Na rys. 2.13 a) mamy $h(p) = h(Fakt(p))$, czyli nie są spełnione warunki monotonicznej numeracji elementów ścieżki identyfikującej.

Na rys. 2.13 b) mamy $Baza(p) = f$, stąd $h(f) < h(g)$, gdyż $(p \in g)$, lecz również mamy $Baza(r) = g$ stąd $h(g) < h(f)$, gdyż $(r \in f)$. Również w tym przypadku nie ma monotonicznej numeracji w ścieżce identyfikacji.

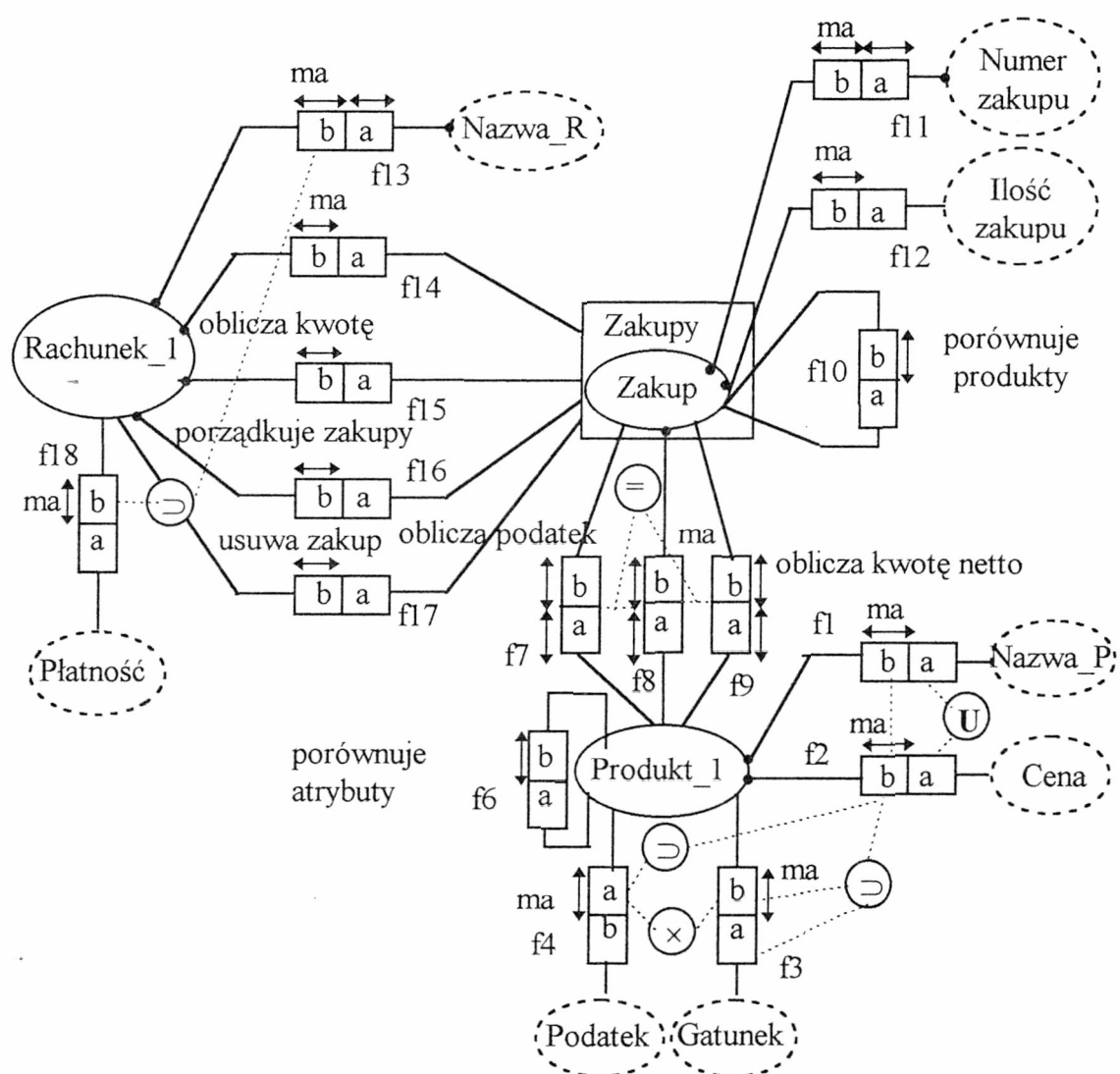
2.3. Przykład modeli ORM

Wstępny opis wybranego zagadnienia dotyczącego sporządzania rachunku, selekcjonuje i nazywa pewne byty "świata rzeczywistego" [81]:

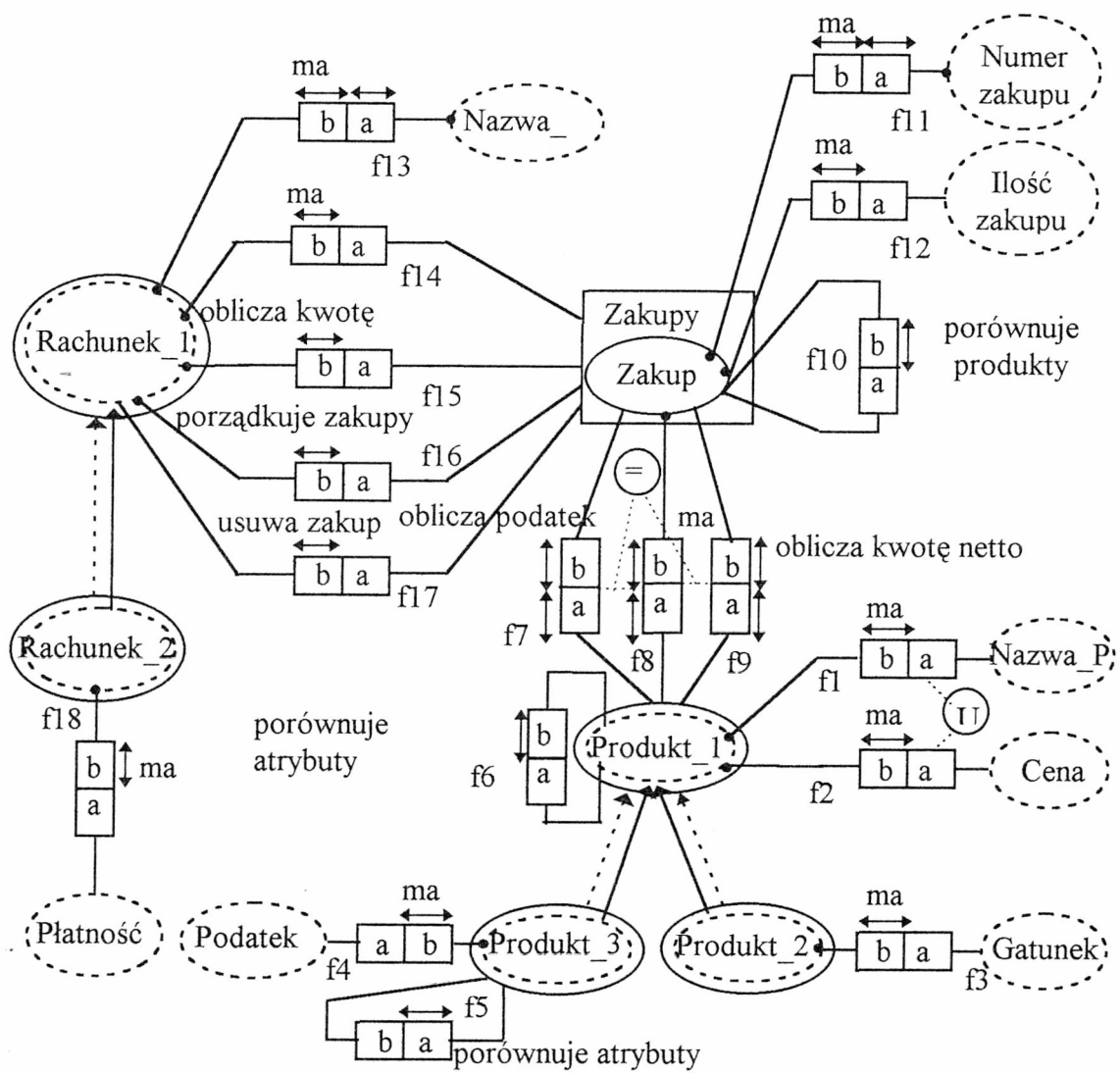
- a) *Rachunek* zawiera pewną liczbę elementów określanych jako *Zakup* towaru. Część *Rachunków* ma dodatkową informację o wyborze płatności i potrafi wstawiać nowy *Zakup* jedynie wtedy, kiedy zawiera on zupełnie nowy *Towar*. Jeżeli któryś z istniejących *Zakupów* ma ten sam *Towar*, należy jedynie powiększyć w nim liczbę zakupionego już *Towaru*. Potrafi usuwać wybrany *Zakup* oraz wyliczyć swoją wartość brutto.
- b) *Zakup* składa się z informacji o *Towarze*, jego ilości oraz numeru zakupu. Potrafi wstawiać zakupiony *Towar*, jego ilość oraz numer dokonanego zakupu; oblicza swoją wartość netto (liczba towaru * cena towaru) oraz podatek (wartość netto * podatek); potrafi sprawdzić inny *Zakup* czy nie zawiera tego samego *Towaru*.
- c) *Towar* ma podstawowe dane zawierające informację o nazwie, cenie oraz część *Towarów* ma dodatkowe informacje o gatunku, część *Towarów* ma dodatkowe informacje o podatku *Towar* potrafi porównać swoje dane z danymi innego *Towaru* oraz wyliczyć swoją cenę w oparciu o gatunek (cena * 0.8, jeśli gatunek=2 itp).

Na rys. 2.14, 2.15 i 2.16 przedstawiono trzy rozwiązania tego samego problemu sporządzania rachunku. W dalszej części pracy podana zostanie metoda wyboru najlepszego rozwiązania.

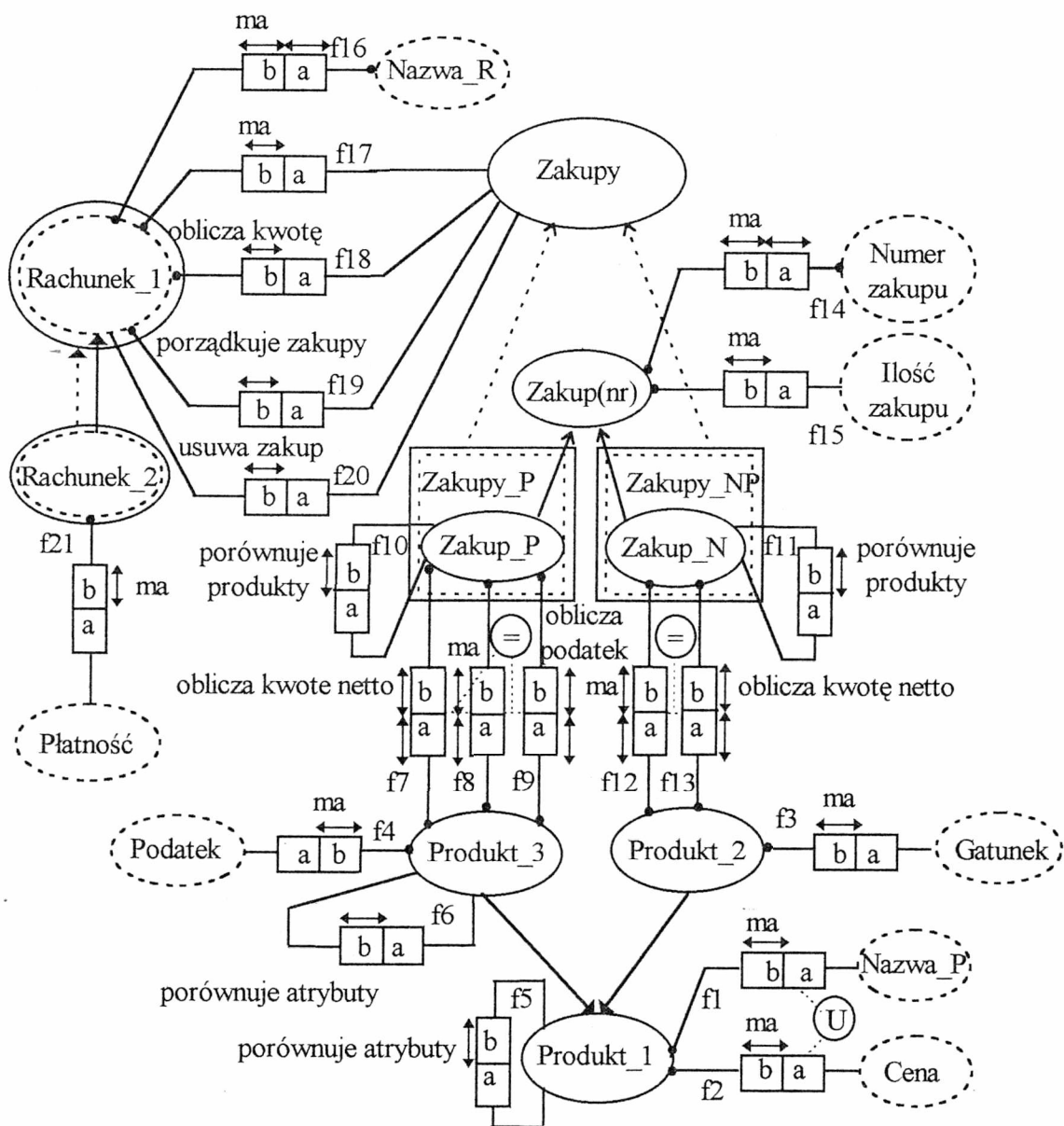
Na rys. 2.14, 2.15, 2.16 przyjęto nazwy ról w każdym typie faktu jako *a* i *b*, oprócz nazw mnemotechnicznych oraz zastosowano skróty nazw typów obiektowych. Ułatwia to oznaczanie wierzchołków i krawędzi w odpowiadających grafach funkcjonalnych (rozdziały 3, 5).



Rys. 2.14. Model ORM pierwszego rozwiązania - M_1



Rys. 2.15. Model ORM drugiego rozwiązania - M_2



Rys. 2.16. Model trzeciego rozwiązania - M_3

3. Równoważność schematów konceptualnych

Problem równoważności schematów konceptualnych związany jest z ich normalizacją. Normalizacja schematu jest stosowana w celu wyeliminowania redundancji i anomalii, jakie mogą wystąpić w aplikacji podczas modyfikacji danych np. w relacyjnych i obiektowych bazach danych. Normalizacja schematu odbywa się za pomocą jego transformacji, zachowującej równoważność schematu wejściowego i wyjściowego. Taka transformacja jest bijekcją.

Problemy transformacji na poziomie schematu logicznego relacyjnych baz danych przedstawiono w [91, 103], gdzie uwzględnia się zależności funkcyjne oraz dane bez zachowania semantyki innych ograniczeń między danymi. Problemy transformacji na poziomie schematu konceptualnego ER przedstawiono w [53, 76], gdzie rozpatruje się w sposób algebraiczny i funkcyjny równoważność semantyczną schematów tzn. zachowującą dane i wszystkie właściwości. Jakość schematu, uzyskanego w wyniku transformacji, bada się za pomocą pewnych miar statystycznych, określających efektywność zapytań w bazach danych, koszt i właściwości implementacji. W pracy [105] transformacji schematów ER dokonuje się za pomocą operacji na hypergrafach wiążących wystąpienia danych zgodnie z ich ograniczeniami. W pracy [99] bada się sensowność transformacji za pomocą grafu SIG (Schema Intension Graph), pomijając badanie jakości struktury uzyskanego schematu. W pracy [129] normalizacji schematu obiektowego dokonuje się za pomocą tzw. *ścieżek zależności* (*path dependency*) wykorzystując algebrę relacji. Oprócz zależności funkcyjnych bada się w [129] semantykę różnych powiązań między obiektami. W pracy [92] transformacja schematu ma charakter syntaktyczny i jest realizowana za pomocą rachunku relacji oraz rachunku zdań. Równoważność schematów jest oceniana na podstawie zapytań nad nowymi relacjami.

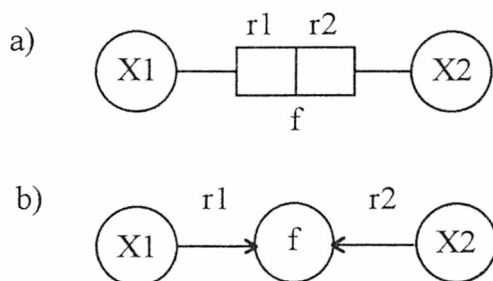
W niniejszej pracy równoważność schematów bada się za pomocą badania grafu funkcjonalnego diagramu ORM. W rozdziale 3.1. jest podana definicja grafu funkcjonalnego, natomiast w rozdziale 3.2. zdefiniowano pojęcie równoważności schematów ORM (rozdz. 3.2.1), podano podstawy formalne algorytmu do badania równoważności (rozdz. 3.2.2) oraz opracowano algorytm badania równoważności (rozdz. 3.2.3). W rozdziale 3.2 podano przykład zastosowania metody badania równoważności schematów ORM.

3.1. Własności grafu funkcjonalnego schematu konceptualnego

Graf funkcjonalny jest odwzorowaniem izomorficznym struktury informacyjnej. Każdy typ faktu jest przedstawiany jako graf skierowany. Krawędziami tego grafu są role które łączą wierzchołki, reprezentujące typy bazowe ról, z wierzchołkiem, który reprezentuje typ faktu. Na rys. 3.1 a) przedstawiono diagram ORM binarnego typu faktu i na rys. 3.1 b) przedstawiono graf funkcjonalny tego diagramu.

Graf funkcjonalny jest digrafem dwudzielnym. Oznacza to, że zawiera dwa podzbiory wierzchołków: podzbiór typów faktów i podzbiór typów obiektowych, natomiast krawędzie reprezentują role i są skierowane od typów obiektowych do typów faktów. Graf funkcjonalny wyraża wszystkie zależności funkcyjne i niefunkcyjne między typami obiektowymi. Każda z tych zależności funkcyjnych i niefunkcyjnych jest przedstawiona w postaci elementarnych części, jakimi są role. Jeżeli dwa typy bazowe są połączone

krawędzią z tym samym typem faktu rolą oznaczoną identycznie, oznacza to, że są w relacji specjalizacji lub polimorfizmu. Graf funkcjonalny uwidacznia skutek działania reguł specjalizacji i polimorfizmu, stąd nie zaznacza się tych relacji w grafie.



Rys. 3.1. Diagram ORM struktury informacyjnej typu faktu f (a), odpowiadający graf funkcjonalny (b)

Taki sposób reprezentowania typów faktów umożliwia przedstawienie skomplikowanych schematów wyrażeń relacyjnych realizowanych za pomocą elementów struktury informacyjnej ORM. Wyrażanie struktury informacyjnej za pomocą grafu funkcjonalnego jest podstawą do badania semantycznych własności różnych modeli konceptualnych. Zakłada się bowiem, że porównanie grafów sprowadzone do badania ich morfizmu może być wykorzystane do określenia równoważności schematów konceptualnych czyli porównania wyrażeń relacyjnych i odpowiadającej im populacji.

Grafy funkcjonalne zastosowano do przedstawienia własności rodzin typów obiektowych związanych relacjami *Spec* (rozdz. 2.1.2) lub *Pol* (rozdz. 2.1.2). Za pomocą grafów funkcjonalnych uwidoczniono wykonywane role, wynikające z reguły specjalizacji i zastosowania typów polimorficznych.

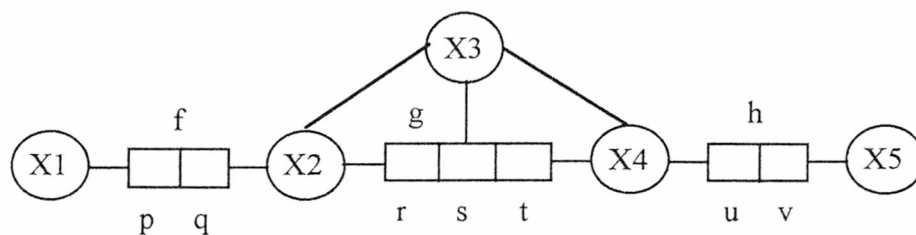
W pracy tej oparto się na idei grafów funkcjonalnych przedstawionych w [15]. Jednak wprowadzono pewne zmiany, niezbędne do badania równoważności diagramów zawierających rodziny typów obiektowych powiązanych relacjami specjalizacji *Spec* (rozdz. 2.1.2) i polimorfizmu *Pol* (rozdz. 2.1.2). Zmiany te wynikają z odmiennego wyrażania relacji specjalizacji za pomocą grafów.

Na rys. 3.3 a) przedstawiono graf funkcjonalny diagramu ORM z rys. 3.2, zaprezentowany w [15], natomiast na rys. 3.3 b) pokazano graf funkcjonalny tego samego diagramu ORM, odzwierciedlający zmiany dokonane w pracy. W grafie funkcjonalnym na rys. 3.3 b uwidoczniono wszystkie dostępne role, jakie wynikają z reguły specjalizacji przyjętej na diagramie z rys. 3.2. Z przyjętej reguły wynika, że role spełniane przez $X2$, $X3$, $X4$ są następujące (2.27):

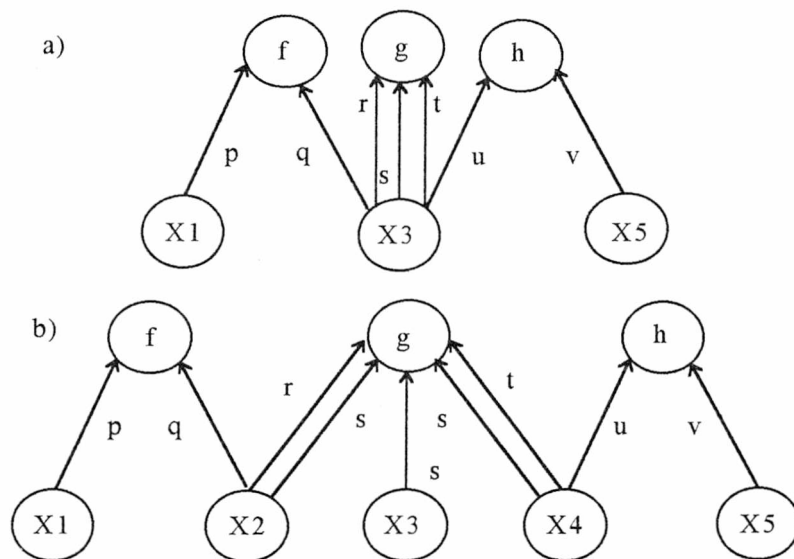
$$\begin{aligned} WRole(X2) &= \{q, r, s\}, \\ WRole(X3) &= \{s\}, \\ WRole(X4) &= \{s, t, u\}. \end{aligned}$$

Graf funkcjonalny z rys. 3.3 a) nie uwidacznia całej rodziny typów obiektowych wchodzących w relację specjalizacji sprowadzając elementy drzewa dziedziczenia do korzenia $X3$, traktując go jako reprezentanta wszystkich podzbiorów rodziny. To uproszczenie zostało w pracy pominięte i w grafie uwidacznia się regułę specjalizacji. Umożliwia to badanie wpływu struktury drzewa dziedziczenia na spójność całego sche-

matu konceptualnego.

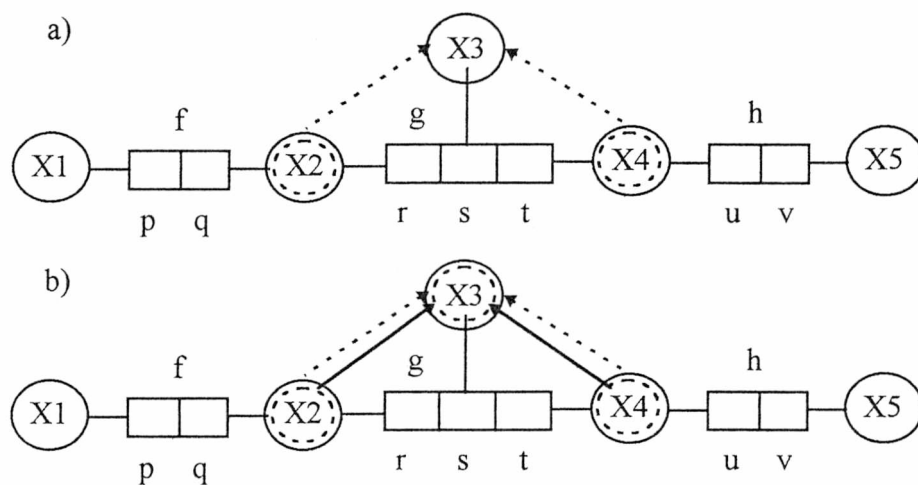


Rys. 3.2. Diagram ORM zawierający relację *Spec*

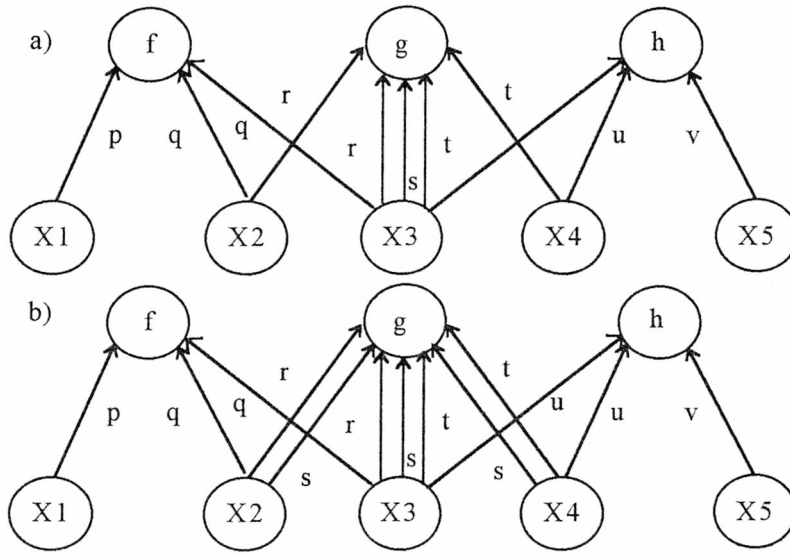


Rys. 3.3. Klasyczny graf funkcjonalny diagramu ORM z rys. 3.1 (a); graf funkcjonalny tego samego diagramu przyjęty w pracy (b)

Na rys. 3.4 a) przedstawiono diagram ORM zawierający relację *Pol* bez specjalizacji, natomiast na rys. 3.4. b) ze specjalizacją. Wykorzystano diagram z rys. 3.1, zastępując relację *Spec* relacją *Pol* oraz zamieniając typy *X2*, *X4* na typy polimorficzne na rys. 3.4 a) oraz *X2*, *X3*, *X4* na rys. 3.4 b).



Rys. 3.4. Diagramy ORM zawierający relację *Pol* (a), *Pol* ze specjalizacją (b)



Rys. 3.5. Grafy funkcjonalne diagramów z rys. 3.4 a (a), z rys. 3.4 b (b)

Graf funkcjonalny z rys. 3.5 a) prezentuje wszystkie role osiągane przez każdego z typów obiektowych wynikające z relacji *Pol*. Z reguł polimorfizmu bez specjalizacji wynika (2.32):

$$\begin{aligned} PolRole(X2) &= \{q, r\}, \\ PolRole(X3) &= \{q, r, s, t, u\}, \\ PolRole(X4) &= \{t, u\}. \end{aligned}$$

Graf funkcjonalny z rys. 3.5 b) prezentuje wszystkie role osiągane przez każdego z typów obiektowych wynikające z relacji *Pol* połączonej z relacją *Spec*. Z reguł specjalizacji i relacji polimorfizmu wynika (2.37):

$$\begin{aligned} PolRoleS(X2) &= \{q, r, s\}, \\ PolRoleS(X3) &= \{q, r, s, t, u\}, \\ PolRoleS(X4) &= \{s, t, u\}. \end{aligned}$$

Uwidocznianie różnic między relacjami *Pol* i *Spec* za pomocą grafów funkcjonalnych pomaga dokonać wyboru właściwych konstrukcji do realizacji schematu dziedziczenia przy wykonywaniu modelu koncepcyjnego danej aplikacji. Niezbędne jednak są pewne reguły dokonywania wyboru. Opracowanie takich reguł jest jednym z celów tej pracy.

Podstawy formalne analizy grafów funkcjonalnych

Podstawy formalne dotyczące analizy grafów funkcjonalnych oparto na pracach [15, 21, 45, 138]. Opis formalny jest podstawą do analizy równoważności grafów funkcjonalnych różnych struktur informacyjnych. Badanie równoważności oparto na badaniu morfizmu grafów funkcjonalnych.

Definicja 3.1

Grafem dwudzielnym $G(V1, V2)$ nazywamy taki graf, w którym wierzchołki można podzielić na dwa podzbiory $V1$ i $V2$ w taki sposób, że każda krawędź grafu G łączy dowolny wierzchołek zbioru $V1$ z dowolnym wierzchołkiem zbioru $V2$.

Definicja 3.2

Pełnym grafem dwudzielnym nazywamy taki graf dwudzielny, w którym wszystkie wierzchołki z $V1$ są połączone z wszystkimi wierzchołkami $V2$.

Definicja 3.3

Digrafem dwudzielnym $G(V1, V2, K)$ nazywamy taki graf, w którym wierzchołki można podzielić na dwa podzbiory $V1$ i $V2$ w taki sposób, że każda krawędź ze zbioru krawędzi K grafu G jest skończoną rodziną uporządkowanych par elementów po jednym ze zbioru $V1$ i $V2$ i zwana jest krawędzią skierowaną.

Definicja 3.4

Digrafem funkcjonalnym nazywamy taki digraf dwudzielny, w którym krawędzie są skierowane zawsze od wierzchołka ze zbioru $V1$ do wierzchołka ze zbioru $V2$.

Definicja 3.5

Graf funkcjonalny jest digrafem funkcjonalnym nad alfabetami WO_F , WF_F i K_F (alfabety wierzchołków i krawędzi) i jest reprezentowany przez krotkę $(WO, WF, K, wof, wff, kf)$, gdzie WO oznacza zbiór wierzchołków odpowiadających typom obiektowym, WF oznacza zbiór wierzchołków odpowiadających typom faktów i $K \subseteq WO \times WF$ oznacza zbiór krawędzi grafu odpowiadających rolom wiążącym typy obiektowe z typami faktów. Funkcje wof , wff i kf są funkcjami oznaczającymi odpowiedni wierzchołki i krawędzie grafu, czyli: $wof: WO \rightarrow WO_F$, $wff: WF \rightarrow WF_F$ oraz $kf: K \rightarrow K_F$.

Niech $\mathcal{G}(WO_F, WF_F, K_F)$ oznacza zbiór wszystkich grafów funkcjonalnych nad alfabetami WO_F , WF_F i K_F . Dla każdego $a \in K_F$, K_a jest zdefiniowana jako zbiór $\{x \mid x \in K \wedge kf(x) = a\}$, czyli zbiór wszystkich krawędzi oznaczonych przez symbole a z alfabetu krawędzi K_F .

Wszystkie krawędzie i wierzchołki grafu funkcjonalnego są oznaczone. Oprócz tych oznaczeń można wprowadzić etykiety krawędzi i wierzchołków pozwalające interpretować poszczególne elementy grafu. Jest to przydatne przy analizie porównawczej, gdyż algorytm przeszukiwania grafu może odwoływać się do pewnej klasyfikacji, co prowadzi do ograniczenia jego złożoności.

Definicja 3.6

Dane są grafy $G1$ i $G2$, należące do zbioru $\mathcal{G}(WO_F, WF_F, K_F)$. $G1$ i $G2$ są izomorficzne, jeżeli istnieje bijekcja $f: W1 \rightarrow W2$ taka, że dla dowolnego $a \in K_a$, prawdziwe są następujące warunki: $(w1, w2) \in K1_a \Leftrightarrow (f(w1), f(w2)) \in K2_a$.

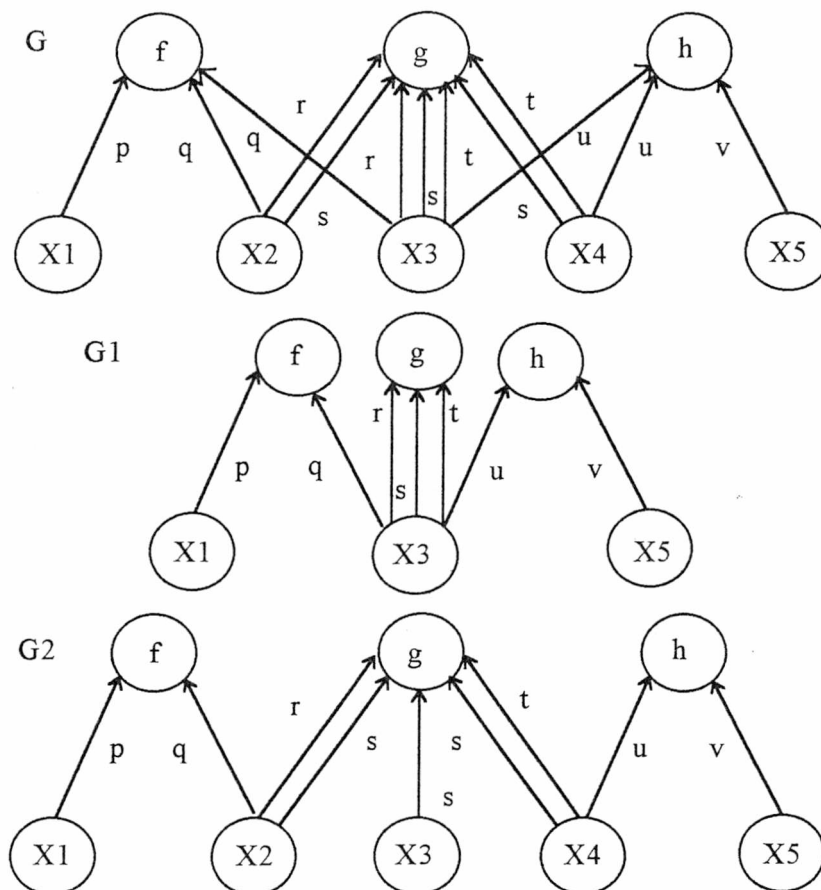
Oznacza to, że dwa grafy $G1$ i $G2$ są izomorficzne, jeżeli różnią się jedynie oznaczeniami i etykietami wierzchołków. Dwa grafy $G1 \in \mathcal{G}(WO_F, WF_F, K_F)$ i $G2 \in \mathcal{G}(WO_F, WF_F, K_F)$ są identyczne ($G1 \equiv G2$), jeżeli są izomorficzne i spełniona jest zależność: $wof1 = wof2 \bullet f$ oraz $wff1 = wff2 \bullet f$ (sekwencyjne prawostronne złożenie funkcji). Identyczne grafy różnią się jedynie etykietami wierzchołków.

W celu porównania grafów funkcjonalnych, wprowadza się pojęcie podgrafu i grafu częściowego.

Definicja 3.7

Niech grafy $G1$ i $G2$ należą do zbioru $\mathcal{G}(WO_F, WF_F, K_F)$:

- Graf $G2$ nazywamy *grafem częściowym* grafu $G1$ ($G2 \subseteq_p G1$), jeżeli dla każdego $a \in K_F$ są spełnione następujące warunki: $WO2 \subseteq WO1$, $WF2 \subseteq WF1$, $K2_a \subseteq K1_a$, $wof2 = wof1 / WO2$, $wff2 = wff1 / WF2$. Oznacza to, że w grafie częściowym $G2$ grafu $G1$ wszystkie oznaczenia wierzchołków w $G2$ znajdują również się w grafie $G1$, jednak w grafie $G1$ mogą pojawić się dodatkowe krawędzie łączące odpowiadające sobie wierzchołki,
- Graf $G2$ nazywamy *podgrafem* grafu $G1$, jeżeli $G2 \subseteq_p G1$ i $K2_a = K1_a \cap (W2 \times W2)$. Oznacza to, że w podgrafie $G2$ grafu $G1$ wszystkie krawędzie i wierzchołki $G2$ powinny pojawić się w $G1$ oraz wszystkie oznaczenia odpowiadających sobie wierzchołków są takie same.



Rys. 3.6. Graf G , podgraf $G1$ i graf częściowy $G2$

Na rys. 3.6 przedstawiono przykłady grafu G , grafu częściowego $G2$ i podgrafu $G1$. Wszystkie wierzchołki i krawędzie podgrafu $G1$ zawiera graf G i nie istnieją w grafie G dodatkowe krawędzie łączące wierzchołki należące do grafu $G1$. Graf częściowy $G2$ zawiera wierzchołki, które wszystkie należą do grafu G , lecz w grafie G istnieją dodatkowe gałęzie łączące te wierzchołki (wierzchołki f i $X3$ połączone krawędzią q , wierzchołki h i $X3$ krawędzią u , wierzchołki g i $X3$ połączone krawędziami r, t).

3.2. Metody oceny równoważności schematu funkcjonalnego

Strukturę informacyjną badamy za pośrednictwem grafu funkcjonalnego, który zawiera wszystkie związki funkcyjne i niefunkcyjne między typami obiektowymi wyrażanych za pośrednictwem krawędzi oznaczających role. Przy badaniu zakłada się, że nazwy typów atrybutowych powinny być jednoznaczne i identyczne w każdym z badanych grafów. Dzięki temu można wyznaczać podgrafy i grafy częściowe wyrażające odpowiednio powiązania między odpowiadającymi typami atrybutowymi, którymi zastępuje się typy abstrakcyjne.

Powyższy sposób wynika z tego, że typy abstrakcyjne są połączone zawsze typem faktu *Most* z typem atrybutowym, który go identyfikuje. W związku z tym przy rozpatrywaniu równoważności schematu konceptualnego typy abstrakcyjne zastępuje się identyfikującymi je typami atrybutowymi. Stąd populację typów abstrakcyjnych zastępuje się populacją identyfikujących je atrybutów. Założenie to wynika z twierdzenia 2.4.(rozdział 2.2.6).

3.2.1. Warunki równoważności schematów konceptualnych

Warunki równoważności schematów konceptualnych są następujące:

E1.Schematy konceptualne (2.1, 2.46, 2.43, rozdz. 2.2.5)

$$SK_1 = (SI_1, Ograniczenia_1, POP_1), \quad (3.1)$$

$$SK_2 = (SI_2, Ograniczenia_2, POP_2) \quad (3.2)$$

są równoważne w sensie zachowania zależności funkcyjnych, jeżeli mają identyczne zbiory typów atrybutowych

$$TWart_1 \equiv TWart_2, \quad (3.3)$$

gdzie

$$TWart_1 \subset SI_1 \text{ i } TWart_2 \subset SI_2 \quad (3.4)$$

oraz można zbudować wyrażenia relacyjne zawierające zbiory ról wyznaczające wszystkie zależności funkcyjne między typami atrybutowymi

$$\exists i=1..n, \sigma_i \in Ograniczenia_1, \tau_i \in TRol_1 \quad \exists j=1..m \sigma_j \in Ograniczenia_2, \tau_j \in TRol_2$$

$$[\{BazyAtr(\sigma_i) \rightarrow BazyPdAtr(\tau_i)\} \equiv \{BazyAtr(\sigma_j) \rightarrow BazyPdAtr(\tau_j)\}] \quad \{*\} \quad (3.5)$$

Zbiory ról σ_i, τ_i oraz σ_j, τ_j są powiązane funkcyjnie nad strukturą informacyjną SI_1 oraz SI_2 (2.89), czyli

$$\sigma_i \xrightarrow{SI_1} \tau_i, \quad (3.7)$$

$$\sigma_j \xrightarrow{SI_2} \tau_j, \quad (3.8)$$

natomiast zbiory typów atrybutowych wyrażających dziedzinę każdej zależności funkcyjnej *BazyAtr* oraz przeciwdziedzinę *BazyPdAtr* są zdefiniowane jako (2.3, 2.1, 2.134, 2.136, 2.139, 2.142)

$$BazyAtr(r) = \{y \mid y \in TWart \wedge \exists q \in r [Baza(q) = y]\}$$

$$\vee (\exists x \in TA, RoleIdF(x, r) [y \in AtrybutyIdF(x, r)])$$

$$\vee (\exists x \in TA, r = SumaIdF(x, (\tau_i)_{i \in n}) [y \in AtrybutyIdF(x, (\tau_i)_{i \in n})]) \quad \{*\} \quad (3.9)$$

oraz (2.3, 2.1, 2.80, 2.86, 2.4, 2.11, 2.132, 2.133, 2.137, 2.139, 2.142)

$$BazyPdAtr(r) = \{y \mid (y \in TWart \wedge \exists q \in r, unikat(q) \wedge total(q) [y \in Bazy(Inne_Role(q))]) \}$$

$$\vee (\exists q \in r, unikat(q) \wedge total(q), x \in Bazy(Inne_Role(q)), x IdF \tau [y \in AtrybutyIdF(x, \tau)])$$

$$\vee (\exists q \in r, unikat(q) \wedge total(q), x \in Bazy(Inne_Role(q)), SumaIdF(x, (\tau_i)_{i \in n})$$

$$[y \in AtrybutyIdF''(x, (\tau_i)_{i \in n})]) \}. \quad \{*\} (3.10)$$

Przy definicji dziedziny i przeciwdziedziny zależności funkcyjnych typy abstrakcyjne są zastępowane typami atrybutowymi należącymi do zbiorów identyfikujących *AtrybutyIdF* (przy identyfikacji prostej) i *AtrybutyIdF''* (przy identyfikacji złożonej). Dotyczy to drugiego i trzeciego zdania w definicjach *BazyAtr* i *BazyPdAtr*.

E2.Schematy konceptualne (2.1, 2.46, 2.43, rozdz. 2.2.5)

$$SK_1 = (SI_1, Ograniczenia_1, POP_1), \quad (3.11)$$

$$SK_2 = (SI_2, Ograniczenia_2, POP_2) \quad (3.12)$$

są równoważne w sensie zachowania danych, jeżeli mają identyczne zbiory typów atrybutowych

$$TWart_1 \equiv TWart_2, \quad (3.13)$$

gdzie

$$TWart_1 \subset SI_1 \text{ i } TWart_2 \subset SI_2 \quad (3.14)$$

oraz identyczne populacje określone zależnościami funkcyjnymi *Pop_W_1* i *Pop_W_2*, czyli

$$Pop_W_1 = Pop_W_2, \quad (3.15)$$

gdzie

$$Pop_W_1 \subseteq POP_1 \text{ i } Pop_W_2 \subseteq POP_2. \quad (3.16)$$

E3.Schematy konceptualne (2.1, 2.46, 2.43, rozdz. 2.2.5)

$$SK_1 = (SI_1, Ograniczenia_1, POP_1), \quad (3.17)$$

$$SK_2 = (SI_2, Ograniczenia_2, POP_2) \quad (3.18)$$

są równoważne w sensie zależności нефunkcyjnych, jeżeli mają identyczne zbiory typów atrybutowych

$$TWart_1 \equiv TWart_2, \quad (3.19)$$

gdzie

$$TWart_1 \subset SI_1 \text{ i } TWart_2 \subset SI_2 \quad (3.20)$$

oraz można zbudować wyrażenia relacyjne wynikające z ograniczeń wyznaczających identyczne zależności нефunkcyjne między typami atrybutowymi

$$\exists_{i=1..n} \sigma_i, \tau_i \in Ograniczenia_1 \exists_{i=1..n} \sigma_j, \tau_j \in Ograniczenia_2 [\{BazyAtr(\sigma_i), BazyAtr(\tau_i)\} \equiv \{BazyAtr(\sigma_j), BazyAtr(\tau_j)\}] \quad \{*\} (3.21)$$

oraz populacja tych zbiorów typów atrybutowych jest identyczna

$$Pop_O_1 = Pop_O_2, \quad (3.22)$$

gdzie

$$Pop_O_1 \subseteq POP_1 \text{ i } Pop_O_2 \subseteq POP_2. \quad (3.23)$$

E4. Muszą być spełnione warunki 1, 2, 3, aby schematy konceptualne były semantycznie równoważne.

Przy badaniu równoważności typów abstrakcyjnych pomija się różne liczby typów atrybutowych należących do standardowych struktur danych np. typy indeksowe w typach sekwencyjnych.

3.2.2. Proces badania równoważności schematów konceptualnych

a) Podstawy formalne badania równoważności modeli

Proces badania równoważności składa się z czterech etapów, w których sprawdza się identyczność:

- 1) typów atrybutowych,
- 2) zależności funkcyjnych,
- 3) ograniczeń ilościowych i zbiorowych,
- 4) populacji wynikającej z zależności funkcyjnych i niefunkcyjnych.

Badanie zależności funkcyjnych opiera się na twierdzeniu 3.1.

Twierdzenie 3.1 (2.89, 2.60, 2.61, 2.3, A16-A22, A26, 2.40, 2.102). {*}

Jeżeli dwa typy faktów f, t są relacyjnie powiązane i prawdziwe są wyrażenia

$$\pi_a f \rightarrow^f \pi_b f \text{ i } \pi_a t \rightarrow^t \pi_b t,$$

to wynika z tego, że prawdziwe są również wyrażenia

$$\pi_a f \rightarrow^{f|></t} \pi_b t \text{ i } \text{Baza}(\pi_a f) \rightarrow^{f|></t} \text{Baza}(\pi_b t).$$

Dowód

Dowód tego twierdzenia jest przeprowadzony na zasadzie nie wprost. Na podstawie definicji zależności funkcyjnych, jeżeli nieprawdziwą byłaby zależność

$$\neg \pi_a f \rightarrow^{f|></t} \pi_b t,$$

to wtedy rola $\pi_a f \notin \text{RoleUnikat}$, co przeczy założeniu, gdyż wtedy nieprawdziwa byłaby zależność

$$\neg \pi_a f \rightarrow^f \pi_b f.$$

Dla ról relacyjnie powiązanych $\pi_b f \sim \pi_a t$ prawdziwa jest zależność

$$\pi_a f \rightarrow^f \pi_a t.$$

Z założenia mamy $\pi_a t \rightarrow^t \pi_b t$, stąd dla wyrażenia $f|></t$ mamy $\pi_a f \rightarrow^{f|></t} \pi_b t$.

b) Podstawy formalne algorytmu sprawdzania równoważności modeli

Krok 1. Sprawdzenie typów atrybutowych polega na sprawdzeniu w zbiorze wierzchołków *WO* (def. 3.3-3.4 w rozdz. 3.1) grafu funkcjonalnego równości podzbiorów typów atrybutowych, zakładając identyczność ich nazw.

Krok 2. Sprawdzenie zależności funkcyjnych jest podobne do procesu wyznaczania połączenia unikatowego wymaganego do identyfikacji. Jednak warunki były tam ostrzejsze - wszystkie role należące do ścieżki identyfikującej musiały należeć do zbiorów ograniczeń *unikat*(τ) (2.86) i *total*(τ) (2.80), co wynikało z wymagania, aby zależności między wystąpieniami atrybutów klucza a każdym wystąpieniem identyfikowanego typu były różnowartościowe.

Wyrażenie *Droga_f* definiuje własności połączenia między typami obiektowymi wynikające ze zbioru σ . Zbiór σ jest zestawem ról objętych ograniczeniem *total* i *unikat*, wyznaczających połączenie z typami abstrakcyjnymi lub typami atrybutowymi i muszą do niego należeć wszystkie role wychodzące z kolejnych, połączonych wierzchołków grafu funkcjonalnego, którymi są typy obiektowe (2.96, 2.2, 2.80, 2.86, 2.59)

$$Droga_f(\sigma) \equiv |Fakty(\sigma)| > 1 \Rightarrow \exists p \in \sigma [\exists q \in \sigma \setminus Fakt(p) [(Fakt(p) \setminus \sigma) \sim q] \wedge total(q) \wedge unikat(q)] \wedge Droga(\sigma \setminus Fakt(p))]. \quad \{*\}(3.24)$$

Wyrażenia *ścieżka_f* łącząca atrybuty powiązane funkcyjnie nad strukturą informacyjną jest równa (2.64, 2.65, 2.60, 2.61, 2.96)

$$ścieżka_f(\sigma) = \delta_{S(\sigma)}(|><| f \in Fakty(\sigma) f). \quad \{*\}(3.25)$$

Funkcja *Fakty* wyznacza zbiór typów faktów związanych ze zbiorem ról σ , spełniających ograniczenia *total* (2.80) i *unikat* (2.86).

Formuła selekcji $S(\sigma)$ określa sposób wyboru ról łączących role ze zbioru unikatowego, jeżeli nie należą one do jednego typu faktu. Jest ona zdefiniowana następująco (2.11, A16-A22 w rozdz. 2.1.2)

$$S(\sigma) = \forall p \in \sigma, q \in Ime_Role(p) \exists r \in \sigma \setminus p [p \sim q]. \quad \{*\}(3.26)$$

Jeżeli wszystkie role wyznaczone przez wyrażenie *ścieżka_f* (3.25) są objęte ograniczeniami *unikat* (2.86) oraz *total* (2.80), można wyznaczyć identyfikator (klucz) dla zbioru tych ról.

Stąd *identyfikator_f* wyrażony przez funkcją *ścieżka_f* (3.25) i zbiór σ jest równy

$$identyfikator_f(ścieżka_f(\sigma), \sigma). \quad \{*\}(3.27).$$

Jeżeli połączenie składa się tylko z jednego typu faktu, *ścieżka_f* (3.25) jest równa

$$ścieżka_f(\sigma) = f. \quad \{*\}(3.28)$$

Algorytm badania równoważności schematów konceptualnych jest oparty na wyszukiwaniu podgrafów częściowych zawierających ścieżkę obejmującą zależności funkcyjne między typami obiektowymi. Ścieżka ta łączy typy atrybutowe stanowiące dziedzinę i przeciwdziedzinę zależności funkcyjnych, ponieważ każdy typ abstrakcyjny jest zastępowany odpowiednio przez zbiory *AtrybutyIdF* (2.136) lub *AtrybutyIdF''* (2.142), które są dołączane do dziedziny *BazyAtr* (3.9) lub przeciwdziedziny *BazyPdAtr* (3.10) odpowiadających im zależności funkcyjnych.

Aby określić wszystkie związki funkcyjne, należy wyznaczyć połączenie

typu *ścieżka_f* (3.25) od każdego typu abstrakcyjnego do innych typów obiektowych. Połączenie może przebiegać przez różne typy abstrakcyjne. Aby istniało połączenie, *rola wyjściowa* każdego z tych typów obiektowych musi być objęta ograniczeniami *total* (2.80) i *unikat* (2.86). Te role tworzą zbiór σ . Oznacza to funkcjonalny związek między typem bazowym tej roli a pozostałymi typami bazowymi ról spełniających funkcję *Inne_Role* (2.11). Jeżeli role należące do zbioru *Inne_Role* (2.11) są również objęte ograniczeniami *total* (2.80) i *unikat* (2.86), są również dołączane do dziedziny kolejnej zależności funkcyjnej, a ich przeciwdziedziną są typy atrybutowe należące do dziedziny poprzedniej zależności. Wynika to z tego, że wystąpiła zależność funkcyjna typu bijekcja.

Na rys. 3.7 b) i rys. 3.8 b) role wyznaczające zależności kluczowe i funkcyjne zaznaczono pogrubionymi strzałkami.

- Krok 3.** Sprawdzenie zależności wynikających z ograniczeń (rozdz. 2.2.5) ilościowych (*częstotliwość*) lub zbiorowych polega na wyznaczeniu typów obiektowych, które są typami bazowymi ról należących do badanego ograniczenia. Następnie wyznacza się zbiór typów atrybutowych identyfikujących te typy abstrakcyjne. W badanych schematach konceptualnych wymaga się, aby każde z ograniczeń wyznaczało identyczną populację zbiorów typów atrybutowych.

Zbiór typów atrybutowych wynikających z tego zbioru typów bazowych jest następujący (3.9):

$$Zbiór_A(\sigma_i, \tau_i) = BazyAtr(\sigma_i) \cup BazyAtr(\tau_i) \quad \{*\} (3.29)$$

dla (2.114, 2.115, 2.116)

$$\begin{aligned} & (\sigma_i, \tau_i \in RolePodzbiór) \vee (\sigma_i, \tau_i \in RoleRozłączność) \vee (\sigma_i, \tau_i \in RoleRówność) \\ & \vee (\sigma_i, \tau_i \in RoleCzęstotliwość) \}. \quad \{*\} (3.30) \end{aligned}$$

Sprawdzenie ograniczeń zbiorowych polega na wyznaczeniu rodziny indeksowanej zbiorów $Zbiór_A(\sigma_i, \tau_i)$ typów atrybutowych należących do każdej z dwóch sekwencji τ_i, σ_i ról danego ograniczenia. Mamy więc

$$Zbiór_B = (Zbiór_A(\sigma_i, \tau_i)_{i \in n}). \quad \{*\} (3.31)$$

Zawartość rodziny zbiorów $Zbiór_B$ w porównywanych schematach konceptualnych jest identyczna, jeżeli oba schematy są równoważne.

- Krok 4.** Sprawdzenie identyczności populacji w obu schematach dokonuje się przez porównanie populacji typów atrybutowych powiązanych funkcyjnie i niefunkcyjnie. Zakłada się, że populacja danego typu abstrakcyjnego x może być zastąpiona populacją typów atrybutowych identyfikujących *AtrybutyIdF* (2.136) lub *AtrybutyIdF''* (2.142) oraz (A24)

$$\begin{aligned} Pop_a(x) &= \{y \mid \exists p \in AtrybutyIdF(x, \tau) [Pop(p) = y] \\ &\vee (\forall q \in AtrybutyIdF''(x, (\tau_i)_{i \in n}) [Pop(q) = y])\}. \quad \{*\} (3.32) \end{aligned}$$

Populacja każdego typu faktu f łączącego typy abstrakcyjne jest równa (2.14, 2.15, 2.3, A26, A24, 2.42, 3.32)

$$Pop_f(f) = \{x \mid (\forall p \in \text{abstr}(f) [x[p] \in Pop_a(\text{Baza}(p))]) \wedge (\forall q \in \text{konkr}(f) [x[q] \in Pop(\text{Baza}(q))])\}. \quad \{*\} (3.34)$$

Populacja wyrażenia relacyjnego, którym jest *ścieżka_f* jest więc równa (3.25, 2.60, 2.61, 2.96, 3.34)

$$Pop(\text{ścieżka_f}(\sigma)) = \{></f \in \text{Fakty}(\sigma) Pop_f(f)\}. \quad \{*\} (3.35)$$

Populacja wypadkowa jest złączeniem tych populacji (2.60, 2.61, 3.25, 2.102, 2.85, 3.35)

$$Pop_W((\sigma_i)_{i \in n}, \{></i=1..n \text{ścieżka_f}(\sigma_i)) = \\ = \{t \mid \forall \sigma_i \subseteq \text{RoleUnikat} \cap \text{RoleTotal} [t[\sigma_i] \in Pop(\text{ścieżka_f}(\sigma_i))]\}. \quad \{*\} (3.36)$$

Populacja każdego zbioru typów obiektowych objętych zależnościami nie-funkcyjnymi (ilościowymi i zbiorowymi) jest wyznaczona jako (3.29, 3.31, 3.38)

$$Pop(\text{Zbiór_B}) = (Pop(\text{Zbiór_A}(\sigma_i, \tau_i)))_{i \in n}, \quad \{*\} (3.37)$$

gdzie (3.29, 3.34)

$$Pop(\text{Zbiór_A}(\sigma_i, \tau_i)) = \{y \mid \forall f \in \sigma_i, p \in \tau_i [y \in Pop_f(f) \vee y \in Pop_f(p)]\}. \quad \{*\} (3.38)$$

3.2.3. Projekt algorytmu sprawdzania równoważności schematów konceptualnych

Na podstawie zależności podanych w rozdz. 3.2.1 i 3.2.2 zaprojektowano algorytmy testowania równoważności schematów konceptualnych. Algorytmy sprawdzania równoważności schematu konceptualnego oparte są na algorytmie przeszukiwania jego grafu funkcjonalnego podobnym do algorytmu Dijkstry [42, 45, 128, 138].

Graf funkcjonalny jest reprezentowany przez macierz schematu konceptualnego M o wymiarach $n \times m$. Kolumny macierzy odpowiadają typom faktów, natomiast wiersze typom obiektowym. Elementy macierzy są krotkami zdefiniowanymi jako funkcja f_{ij}

$$f_{ij}: U \rightarrow \cup \{Dom(A) : A \in U\}. \quad \{*\} (3.39)$$

Elementy macierzy są krotkami, których atrybutami są

$$UI = (\text{przyległość}, o_unikat, o_total, odwiedzony). \quad \{*\} (3.40)$$

Funkcja generująca wartości atrybutów (*przyległość*, *o_unikat*, *o_total*, *odwiedzony*) jest zdefiniowana następująco:

- $f_{ij}(\text{przyległość}) = 1$, gdy i -ty typ obiektowy odgrywa rolę w j -tym typie faktu,
- $f_{ij}(\text{przyległość}) = 0$, gdy i -ty typ obiektowy nie odgrywa roli w j -tym typie faktu,
- $f_{ij}(o_unikat) = 1$, gdy i -ty typ obiektowy odgrywa rolę należącą do zbioru *RoleUnikat* w j -tym typie faktu (2.102),
- $f_{ij}(o_unikat) = 0$, gdy i -ty typ obiektowy nie odgrywa roli należącej do zbioru *RoleUnikat* (2.102) w j -tym typie faktu,
- $f_{ij}(o_total) = 1$, gdy i -ty typ obiektowy odgrywa rolę należącą do zbioru *RoleTotal* w j -tym typie faktu,

$f_{ij}(o_total) = 0$, gdy i -ty typ obiektowy nie odgrywa roli należącej do zbioru $RoleTotal$ (2.85) w j -tym typie faktu,

$f_{ij}(odwiedzony) = 1$, oznacza że rola i -tego typu obiektowego w j -tym typie faktu była już rozpatrywana w czasie realizacji algorytmu,

$f_{ij}(odwiedzony) = 0$, oznacza że rola i -tego typu obiektowego w j -tym typie faktu nie była jeszcze rozpatrywana w czasie realizacji algorytmu.

Wiersze macierzy odpowiadają wierzchołkom należącym do zbioru typów obiektowych, natomiast kolumny odpowiadają wierzchołkom należącym do zbioru typów faktów grafu funkcjonalnego. Wartość 1 w polu *przyległość* oznacza połączenie między i -tym wierzchołkiem ze zbioru typów obiektowych a j -tym wierzchołkiem ze zbioru typów faktów, reprezentujące rolę. Wartości 1 w polach o_unikat oraz o_total oznaczają odpowiednio ograniczenia *unikat* (2.80) i *total* (2.86) związane z daną rolą. Wartość 1 w polu *odwiedzony* oznacza uwzględnienie (lub inaczej odwiedzenie) wierzchołka w czasie realizacji algorytmu.

Zbiory typów atrybutowych i związane z nimi zbiory ról identyfikujących są podane w postaci jednowymiarowej tablicy N o elementach, z których każdy element jest krotką o atrybutach

$$U2 = (Nr_TA, A_IDF, R_IDF). \quad \{*\} (3.41)$$

Funkcja generująca wartości atrybutów (Nr_TA, A_IDF, R_IDF) jest następująca:

$f_i(Nr_TA) = i$, czyli numer i -tego abstrakcyjnego typu obiektowego,

$f_i(A_IDF) =$ zbiór numerów atrybutów wygenerowanych przez funkcję *AtrybutyIdF* (2.136) lub *AtrybutyIdF''* (2.142) dla i -tego typu abstrakcyjnego,

$f_i(R_IDF) =$ zbiór ról τ_i identyfikujących i -ty typ abstrakcyjny lub wyznaczonych przez funkcję *SumaIdF* (2.139) w przypadku identyfikacji złożonej.

Pole Nr_TA zawiera i -ty numer typu abstrakcyjnego. Pole A_IDF zawiera zbiór numerów typów atrybutowych identyfikujących typ abstrakcyjny. Pole R_IDF zawiera połączenia między typami faktów i typami obiektowymi (numer typu obiektowego) czyli role należące do i -tego procesu identyfikacji. Rozmiar macierzy N jest równy t . Numery typów atrybutowych oraz typów abstrakcyjnych podane w tablicy N odpowiadają numerom odpowiednich wierszy w tabeli M .

Tabela jednowymiarowa P zawiera elementy, które są krotkami o atrybutach

$$U3 = (dziedzina, przeciwdziedzina). \quad \{*\} (3.42)$$

Funkcja generująca wartości atrybutów $(dziedzina, przeciwdziedzina)$ jest następująca:

$f_i(dziedzina) =$ zbiór numerów atrybutów wygenerowanych przez funkcję *AtrybutyIdF* (2.136) lub *AtrybutyIdF''* (2.142) dla i -tego typu abstrakcyjnego - w postaci listy uporządkowanej jednokierunkowej bez powtórzeń,

$f_i(przeciwdziedzina) =$ zbiór numerów typów atrybutowych odgrywających role wyznaczone przez wyrażenie *ścieżka_f* (3.25) oraz zbiór numerów typów atrybutowych identyfikujących typy abstrakcyjne odgrywające role należące do zbioru wyznaczonego przez wyrażenie *ścieżka_f* - w postaci listy uporządkowanej jednokierunkowej bez powtórzeń.

Pole *dziedzina* zawiera listę numerów atrybutów stanowiących dziedzinę zależności funkcyjnych. Pole *przeciwdziedzina* zawiera listy numerów typów atrybutowych, które stanowią przeciwdziedzinę zależności funkcyjnych.

Danymi wejściowymi algorytmu wyszukiwania zależności funkcyjnych są: macierz M , macierz N oraz zbiór numerów typów atrybutowych Zb_Atr . Dane wyjściowe umieszczone są w tabeli P .

a) Algorytm wyznaczenia zależności funkcyjnych

1. Inicjalizacja.

- 1.1. Wczytać: macierz M grafu funkcjonalnego, macierz N zawierającą zbiory identyfikujące oraz zbiór Zb_Atr numerów typów atrybutowych.
- 1.2. Na podstawie list R_IDF umieszczonych w tablicy N należy ustawić wartość „1” pola *odwiedzony* w elementach macierzy M .
- 1.3. W tablicy P ustawić puste listy *przeciwdziedzina*, natomiast listy *dziedzina* kolejnych elementów wypełnić listami pobranymi z pól A_IDF odpowiadających elementów tablicy N .

2. Za pomocą procedury *Zależności_funkcyjne* wyznaczyć wszystkie zależności funkcyjne.

- 2.1. Dopóki w tablicy N istnieje i -ty element pobierz numer typu abstrakcyjnego oraz numery identyfikujących typów atrybutowych.

- 2.2. Powtarzać dla każdego j -tego elementu i -tego wiersza macierzy M , w którym pola: *przyległość* jest równe „1”, *o_unikat* oraz *o_total* równe „1” i *odwiedzony* jest równe „0”:

- 2.2.1 ustawić pole *odwiedzony* na „1”,

- 2.2.2 powtarzać dla wszystkich s -tych elementów j -tej kolumny, w których pole *przyległość* jest równe „1”; jeżeli w s -tym wierszu pole *odwiedzony* jest równe „0”, wtedy należy ustawić je na „1”:

- 2.2.2.1 dla typów atrybutowych :

- dołączyć numer typu atrybutowego do pola listy *przeciwdziedziny* bieżącego i -tego elementu tablicy P ,
- jeżeli pola *o_unikat* i *o_total* są równe „1” (oznacza to zależność funkcyjną różnowartościową) należy w tablicy P ustawić nowy element do pola *dziedzina* wstawić numer s , natomiast do pola listy *przeciwdziedziny* należy dołączyć typy atrybutowe identyfikujące i -ty typ abstrakcyjny oraz ich przeciwdziedzinę,
- jeżeli koniec kolumny, należy kontynuować od p.2.2,

- 2.2.2.2 dla typów abstrakcyjnych, jeżeli ich pola *o_unikat* i *o_total* są równe „0”, należy dołączyć do pola listy *przeciwdziedziny* i -tego typu abstrakcyjnego typy atrybutowe identyfikujące ten s -ty typ abstrakcyjny; jeżeli koniec kolumny, należy kontynuować p.2.2,

- 2.2.2.3 jeżeli pola *o_unikat* i *o_total* są równe „1” (oznacza to zależność

funkcyjną różnowartościową) należy:

- dołączyć do pola listy *przeciwdziedziny* s-tego typu abstrakcyjnego typu atrybutowe identyfikujące *i*-ty typ abstrakcyjny; jeżeli koniec kolumny, należy kontynuować p.2.2,
- dołączyć do pola listy *przeciwdziedziny* i-tego typu abstrakcyjnego typu atrybutowe identyfikujące *s*-ty typ abstrakcyjny; jeżeli koniec kolumny, należy kontynuować p.2.2,
- jeśli koniec kolumny, powtórzyć od p.2.2.

2.3. Po wyczerpaniu elementów i-tego wiersza kontynuować algorytm od p.2.1.

Zapis algorytmu w języku **Pascal** jest następujący:

procedure Zależności_funkcyjne (**var** P : PTab; **var** M : MTab; **var** ile : word);

var i, j, k, s, r : integer; pom : pa;

begin

 r := 0; k:=1

{dla każdego typu abstrakcyjnego pobranego z tablicy N powtarzaj}

while k <= ile **do**

begin

 i := N [k].Nr_TA;

for j := 1 **to** m **do**

{dla wszystkich ról i-tego typu abstrakcyjnego należących do zbioru ograniczeń RoleUnikat oraz RoleTotal należy znaleźć typy obiektowe odgrywające role w tym samym j-tym typie faktu}

if (M[i, j].przyległość = 1) **and** (M[i, j].o_unikat = 1) **and**
 (M[i, j].o_total = 1) **and** (M[i, j].odwiedzony = 0) **then**

begin

 M[i, j].odwiedzony := 1;

{dla j-tego typu faktu wyznacz zbiór typów obiektowych}

for s := 1 **to** n **do**

begin

if (M[s, j].przyległość = 1) **and** (s in Zb_Atr) **then**

begin

 M[s, j].odwiedzony:=1;

{dodaj do listy jednokierunkowej uporządkowanej przeciwdziedziny i-tego typu obiektowego typy atrybutowe}

 Dodaj_do_listy (s, P[k].przeciwdziedzina_w);

{jeżeli rola typu atrybutowego należy do zbioru RoleUnikat (2.102) i RoleTotal (2.85), należy założyć nowy element w tablicy P, którego pole dziedzina zawiera typ s, natomiast przeciwdziedzina zawiera typy atry-

```

butowe identyfikujące i-ty typ abstrakcyjny oraz przeciwdziedzinę i-tego
typu}
if (M[s, j].o_unikat = 1) and (M[s, j].o_total = 1) then
begin
r := r+1;
Dodaj_do_listy(s, P[ile + r].dziedzina);
Dodaj_do_listy_a(k, s, true, N, P, ile, r)
end;
end
else
if (M[s, j].przyległość = 1) and (M[s, j].odwiedzony = 0) then
begin
M[s, j].odwiedzony:=1;
{dodaj do tablicy P typy atrybutowe identyfikujące s-ty typ obiektowy
pobrane z tablicy N do listy jednokierunkowej uporządkowanej przeciw-
dziedziny i-tego typu obiektowego - przypadek zależności funkcyjnej nie-
różnowartościowej}
Dodaj_do_listy_a(k, s, false, N, P, ile, 0);
if (M[s, j].o_unikat = 1) and (M[s, j].o_total = 1) then
{dodaj do tablicy P typy atrybutowe identyfikujące i-ty typ obiektowy po-
brane z tablicy N do listy jednokierunkowej uporządkowanej przeciwdzie-
dziny s-tego abstrakcyjnego typu obiektowego - przypadek zależności
funkcyjnej typu bijekcja}
Dodaj_do_listy_a(k, s, true, N, P, ile, 0);
end;
end;
end;
k := k + 1;
end;
ile := ile + r;
end;

```

Zapis całego programu w postaci źródłowej jest zamieszczony w załączniku do pracy.

3. Tablica P zawiera wszystkie zależności funkcyjne między typami atrybutowymi.

Równoważność schematów konceptualnych w sensie zachowania zależności funkcyjnych E2 odpowiada identycznej zawartości tabeli P_i każdego z nich. Identyczność stwierdza się za pomocą aksjomatów Armstronga [103], czyli reguł zwrotności, poszerzalności oraz przechodniości lub reguł wyprowadzonych z aksjomatów Armstronga czyli: pseudopzechodniości, addytywności oraz dekompozycyjności.

b) Algorytm sprawdzania równoważności w sensie zachowania danych

Dla każdej tabeli P_i na podstawie jej zawartości należy wyszczególnić wszystkie zbiory wartości należące do dziedziny i przeciwdziedziny każdej zależności funkcyjnej. Następnie porównać zawartość zbiorów wynikających z równoważnych w sensie zachowania zależności funkcyjnych E2 tabel P_i .

c) Algorytm sprawdzania zależności w sensie zachowania zależności niefunkcyjnych.

Danymi wejściowymi są tablice M oraz N oraz tablica Q . Tablica Q jest jednowymiarowa. Jej elementy zawierają typ ograniczenia, listę łuków odpowiadających zbiorowi ról należących do danego ograniczenia oraz listę typów atrybutowych (pustą na początku). Lista ta w wyniku wykonania algorytmu zawiera typy atrybutowe związane bezpośrednio z rolami należącymi do danego ograniczenia lub typy atrybutowe identyfikujące typy abstrakcyjne, również związane z rolami należącymi do danego ograniczenia. Są to dane wyjściowe.

Elementy tablicy Q są krotkami o atrybutach

$$U4 = (Nr_OGR, R_OGR, A_IDF). \quad \{*\}(3.43)$$

Funkcja generująca wartości atrybutów (Nr_OGR, R_OGR, A_IDF) jest następująca:

$f_i(Nr_OGR) = i$, czyli numer i -tego typu ograniczeń,

$f_i(R_OGR) =$ zbiór ról należących do i -tego typu ograniczenia,

$f_i(A_IDF) =$ zbiór numerów typów atrybutowych związanych z rolami należącymi do danego ograniczenia lub zbiór atrybutów identyfikujących typ abstrakcyjny związany rolami należącymi do danego ograniczenia wyznaczonych przez funkcję $AtrybutyIdF$ (2.136) lub $AtrybutyIdF''$ (2.142).

Algorytm generujący zbiory typów atrybutowych należących do danego ograniczenia:

1. Wczytać macierz N , Q oraz M grafu funkcjonalnego i wyzerować wszystkie pola *odwiedzony* każdego elementu zawierającego wartość „1” w polu *przyległość*.
2. Dla każdego elementu należącego do tablicy Q należy przeszukać tablicę M .
 - 2.1. Jeżeli rola łączy typ atrybutowy, należy dołączyć jej numer do pola A_IdF listy atrybutów w bieżącym elemencie tablicy Q .
 - 2.2. Jeżeli rola łączy typ abstrakcyjny, należy dołączyć jej zbiór identyfikujący z tablicy N do pola A_IdF listy atrybutów w bieżącym elemencie tablicy Q .
3. Powtórzyć algorytm dla każdego schematu konceptualnego i porównać zbiory typów atrybutowych związanych z odpowiadającymi ograniczeniami.
4. Dla każdej tabeli Q_i na podstawie jej zawartości należy wyszczególnić wszystkie zbiory wartości należące do zbiorów typów atrybutowych odpowiadających ograniczeń w porównywanych schematach. Następnie porównać zawartość tych zbiorów dla obu schematów.

Algorytmy testowania równoważności schematów konceptualnych posiada złożoność proporcjonalną do iloczynu $m * n$, gdzie m jest liczbą typów obiektowych, natomiast n jest liczbą typów faktów.

3.3. Przykłady zastosowania metody oceny równoważności schematów konceptualnych

Przedstawione w rozdz. 3.2.2 zasady badania równoważności schematów konceptualnych zilustrowano za pomocą dwóch przykładów. Każdy z przykładów zawiera dwa rozwiązania tego samego problemu. Aby udowodnić równoważność obu schematów, bada się ich grafy funkcjonalne za pomocą algorytmu stwierdzającego równoważność. Algorytm oparty jest na czwartej zasadzie równoważności schematów konceptualnych. W przykładzie pierwszym bada się schematy konceptualne odwzorowujące równoważne schematy relacyjnej bazy danych. Celem takiego postępowania jest wykazanie poprawności metody badania równoważności schematów konceptualnych ORM.

Przykład 1.

1. Schemat relacyjnej bazy danych jest następujący $S = \{R = (U, F)\}$, [91, 103] czyli jest zbiorem relacji R zawierającej atrybuty należące do zbioru U i zależności funkcyjne. Dwa badane schematy wyprowadzono z następującego schematu wyjściowego:

$$R = (\{Hasło, Konto, Numer, Nazwa\}, \{Numer \rightarrow Nazwa, Hasło \rightarrow Konto, Konto \rightarrow Hasło, Hasło \rightarrow Numer, Numer \rightarrow Hasło\}).$$

Po rozłożeniu na równoważne schematy $SR1$ i $SR2$

$$SR1 = \{R1, R2\}, \text{ gdzie}$$

$$R1 = (\{Hasło, Konto\}, \{Hasło \rightarrow Konto, Konto \rightarrow Hasło\}), \text{ klucze: Hasło, Konto,}$$

$$R2 = (\{Numer, Hasło, Nazwa\}, \{Numer \rightarrow Nazwa, Numer \rightarrow Hasło, Hasło \rightarrow Numer\}), \text{ klucze: Numer, Hasło,}$$

$$SR2 = \{R1, R2, R3\}, \text{ gdzie}$$

$$R1 = (\{Konto, Nazwa\}, \{Konto \rightarrow Nazwa\}), \text{ klucz: Konto,}$$

$$R2 = (\{Numer, Hasło\}, \{Numer \rightarrow Hasło, Hasło \rightarrow Numer\}), \text{ klucze: Numer, Hasło,}$$

$$R3 = (\{Konto, Numer\}, \{Konto \rightarrow Numer, Numer \rightarrow Konto\}), \text{ klucze: Konto, Numer.}$$

Na rys. 3.7 i rys. 3.8 przedstawiono schematy konceptualne odpowiadające schematom $SR1$ i $SR2$ oraz ich grafy funkcjonalne. Model konceptualny odpowiadający schematowi $SR1$ zawiera dwa typy abstrakcyjne połączone typem faktu $f5$, którego role zawierają ograniczenia funkcyjne. Oznacza to, że za pomocą tego faktu można składać funkcje lewo- i prawostronnie, występujące w obu typach: *Klient* i *Poczta*, czyli składać zależności funkcyjne między atrybutami tych typów wzdłuż ścieżki połączeniowej. Wynik składania zależności funkcyjnych jest przedstawiony poniżej.

- 1.1 Zależności funkcyjne (2.136, 3.25, 2.60, 2.61).

Graf $SR1$ zawiera wszystkie zależności funkcyjne:

$$\text{AtrybutyIdF}(\text{Poczta}, \{a\}) = \{Hasło\},$$

$$\text{AtrybutyIdF}(\text{Klient}, \{f\}) = \{Numer\},$$

$$\text{ścieżka}_f(\{d\}) = f2 \Rightarrow Hasło \rightarrow Konto,$$

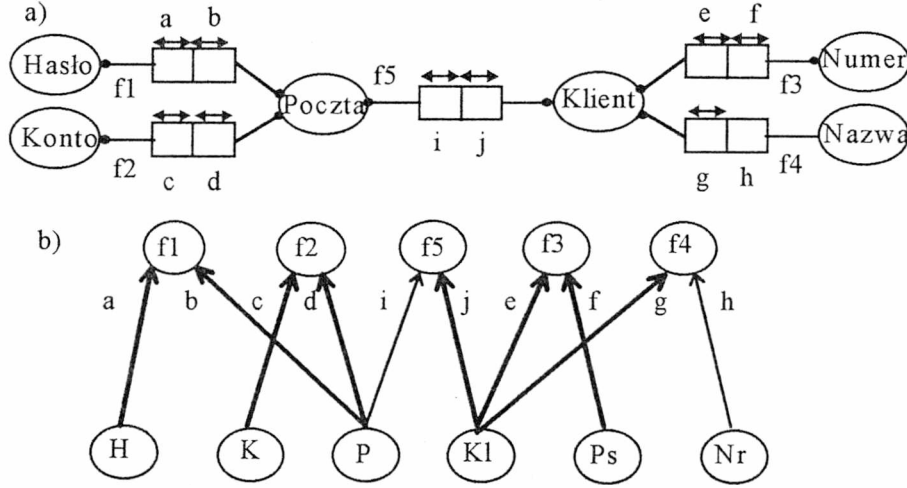
$$\text{ścieżka}_f(\{c\}) = f2 \Rightarrow Konto \rightarrow Hasło,$$

$$\text{ścieżka}_f(\{j\}) = f5 \Rightarrow Numer \rightarrow Hasło,$$

$$\text{ścieżka}_f(\{i\}) = f5 \Rightarrow Hasło \rightarrow Numer,$$

$$\text{ścieżka}_f(\{g\}) = f4 \Rightarrow Numer \rightarrow Nazwa,$$

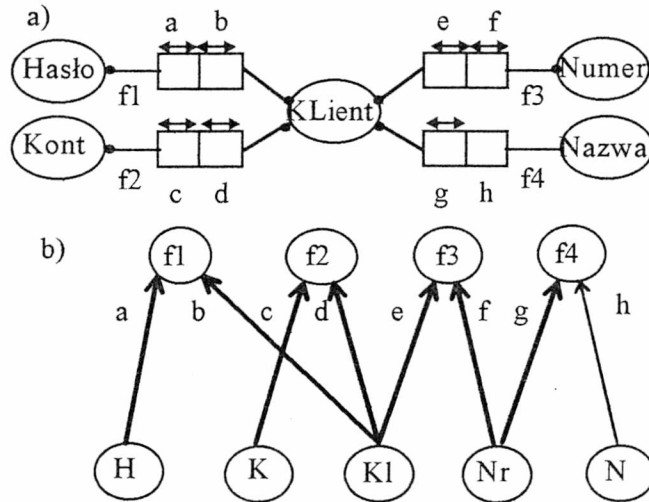
$\text{ścieżka}_f(\{j,d\}) = \sigma_{i=d} (f5 \mid \times \mid f2) \Rightarrow \text{Numer} \rightarrow \text{Konto},$
 $\text{ścieżka}_f(\{c,i\}) = \sigma_{d=i} (f2 \mid \times \mid f5) \Rightarrow \text{Konto} \rightarrow \text{Numer},$
 $\text{ścieżka}_f(\{c,i,g\}) = \sigma_{d=i \wedge j=g} (f2 \mid \times \mid f5 \mid \times \mid f4) \Rightarrow \text{Konto} \rightarrow \text{Nazwa},$
 $\text{ścieżka}_f(\{i,g\}) = \sigma_{j=g} (f5 \mid \times \mid f4) \Rightarrow \text{Hasło} \rightarrow \text{Nazwa}.$



Rys. 3.7. Schemat konceptualny SR1 (a), graf funkcjonalny (b)

Graf SR2 zawiera wszystkie zależności funkcyjne:

$\text{AtrybutyIdF}(\text{Klient}, \{a\}) = \{\text{Hasło}\},$
 $\text{ścieżka}_f(\{d\}) = f2 \Rightarrow \text{Hasło} \rightarrow \text{Konto},$
 $\text{ścieżka}_f(\{c\}) = f2 \Rightarrow \text{Konto} \rightarrow \text{Hasło},$
 $\text{ścieżka}_f(\{f\}) = f3 \Rightarrow \text{Numer} \rightarrow \text{Hasło},$
 $\text{ścieżka}_f(\{e\}) = f3 \Rightarrow \text{Hasło} \rightarrow \text{Numer},$
 $\text{ścieżka}_f(\{f,g\}) = \sigma_{e=g} (f3 \mid \times \mid f4), \Rightarrow \text{Numer} \rightarrow \text{Nazwa},$
 $\text{ścieżka}_f(\{f,d\}) = \sigma_{e=d} (f3 \mid \times \mid f2), \Rightarrow \text{Numer} \rightarrow \text{Konto},$
 $\text{ścieżka}_f(\{c,e\}) = \sigma_{d=e} (f2 \mid \times \mid f3), \Rightarrow \text{Konto} \rightarrow \text{Numer},$
 $\text{ścieżka}_f(\{c,g\}) = \sigma_{d=g} (f2 \mid \times \mid f4), \Rightarrow \text{Konto} \rightarrow \text{Nazwa},$
 $\text{ścieżka}_f(\{g,h\}) = f4 \Rightarrow \text{Hasło} \rightarrow \text{Nazwa}.$



Rys. 3.8. Schemat konceptualny SR2 (a), graf funkcjonalny (b)

1.2. Populacja (2.46, A24, 3.35, 3.36).

$Pop(Nazwa) = \{A, B\}, \quad Pop(Numer) = \{1, 2, 3\},$
 $Pop(Hasło) = \{X, Y, Z\}, \quad Pop(Konto) = \{101, 102, 103\}.$

Z grafu SR1 mamy:

$Pop(ścieżka_f1(\{d\})) = \{\{X, 101\}, \{Y, 102\}, \{Z, 103\}\},$
 $Pop(ścieżka_f2(\{c\})) = \{\{101, X\}, \{102, Y\}, \{103, Z\}\},$
 $Pop(ścieżka_f3(\{j\})) = \{\{1, X\}, \{2, Y\}, \{3, Z\}\},$
 $Pop(ścieżka_f4(\{i\})) = \{\{X, 1\}, \{Y, 2\}, \{Z, 3\}\},$
 $Pop(ścieżka_f5(\{g\})) = \{\{1, A\}, \{2, A\}, \{3, B\}\},$
 $Pop(ścieżka_f6(\{j, d\})) = \{\{1, 101\}, \{2, 102\}, \{3, 103\}\},$
 $Pop(ścieżka_f7(\{c, i\})) = \{\{101, 1\}, \{102, 2\}, \{103, 3\}\},$
 $Pop(ścieżka_f8(\{c, i, g\})) = \{\{101, A\}, \{102, A\}, \{103, B\}\},$
 $Pop(ścieżka_f9(\{i, g\})) = \{\{X, 1\}, \{Y, 2\}, \{Z, 3\}\},$
 $Pop_W = \{\{X, 101, 1, A\}, \{Y, 102, 2, A\}, \{Z, 103, 3, B\}\}.$

Z grafu SR2 mamy:

$Pop(ścieżka_f1'(\{a, d\})) = \{\{X, 101\}, \{Y, 102\}, \{Z, 103\}\},$
 $Pop(ścieżka_f2'(\{c, b\})) = \{\{101, X\}, \{102, Y\}, \{103, Z\}\},$
 $Pop(ścieżka_f3'(\{f, b\})) = \{\{1, X\}, \{2, Y\}, \{3, Z\}\},$
 $Pop(ścieżka_f4'(\{a, e\})) = \{\{X, 1\}, \{Y, 2\}, \{Z, 3\}\},$
 $Pop(ścieżka_f5'(\{f, g\})) = \{\{1, A\}, \{2, A\}, \{3, B\}\},$
 $Pop(ścieżka_f6'(\{f, d\})) = \{\{1, 101\}, \{2, 102\}, \{3, 103\}\},$
 $Pop(ścieżka_f7'(\{c, e\})) = \{\{101, 1\}, \{102, 2\}, \{103, 3\}\},$
 $Pop(ścieżka_f8'(\{c, g\})) = \{\{101, A\}, \{102, A\}, \{103, B\}\},$
 $Pop(ścieżka_f9'(\{g\})) = \{\{1, A\}, \{2, A\}, \{3, B\}\},$
 $Pop_W = \{\{X, 101, 1, A\}, \{Y, 102, 2, A\}, \{Z, 103, 3, B\}\}.$

Populacje wynikające z atrybutów uzależnionych funkcyjnie są identyczne. W przykładzie dla grafu SR1 na podstawie (3.25, 2.102, 2.85, 3.27, 2.3, 2.137, 2.11) przeanalizowano kilka przykładów.

Wyrażenie $ścieżka_f(\{j, d\})$ wyznacza zbiór ról należących do zbiorów *RoleUnikat* i *RoleTotal*. Stąd mamy identyfikator $f((\sigma_{i=d} (f5 \mid >< / f2), \{c, i\}))$ i mamy zależność funkcyjną $Numer \rightarrow Konto$, gdyż $Baza(j) = Klient$ oraz $AtrybutyIDF(Klient, \{f\}) = \{Numer\}$, $Baza(Inne_Role(j)) = Poczta$ i $Baza(Inne_Role(d)) = Konto$.

$Ścieżka_f(\{j\})$ wyznacza zbiór ról $\{j, i\} = f5$ należący do zbiorów *RoleUnikat* i *RoleTotal*. Stąd identyfikator $f(f5, \{j\})$ wyznacza zależność funkcyjną $Numer \rightarrow Hasło$, gdyż $Baza(Inne_Role(j)) = Poczta$, $AtrybutyIDF(Poczta, \{a\}) = \{Hasło\}$, $Baza(j) = Klient$ i $AtrybutyIDF(Klient, \{f\}) = \{Numer\}$.

W wyrażeniu $ścieżka_f(\{g\}) = f4 = \{g, h\}$ jedynie rola g należy do zbiorów *RoleUnikat* i *RoleTotal*. Wyrażenia $Baza(g) = Klient$ i $AtrybutyIDF(Klient, \{f\}) = \{Numer\}$, $Baza(Inne_Role(g)) = Nazwa$. Stąd prawdziwa jest zależność funkcyjna $Numer \rightarrow Nazwa$.

Wyrażenie $ścieżka_f(\{c, i\})$ generuje wszystkie role należące do zbiorów *RoleUnikat* i *RoleTotal*. Stąd istnieją identyfikator $f(\delta_{a=i} (f2 \mid >< / f5, \{c, i\}))$, $Baza(c) = Konto$, $Baza(Inne_Role(j)) = Klient$, $AtrybutyIDF(Klient, \{f\}) = \{Numer\}$. Stąd istnieje zależność funkcyjna $Konto \rightarrow Numer$.

Wyniki uzyskane z analizy grafu funkcjonalnego na podstawie opisu procesu badania równoważności schematów konceptualnych w rozdz. 3.2.2 są potwierdzone wynikami zastosowanego algorytmu z rozdz. 3.2.3.

Na rys. 3.9 - 3.14 przedstawiono „symbolicznie” zawartość macierzy *N* i *M* oraz dane wyjściowe z macierzy *P* dla grafów *SR1* i *SR2*.

Dane grafu *SR1*.

Elementy macierzy *N* zamieszczono w tabeli na rys. 3.9.

Pola elementu macierzy <i>N</i>			
Numer elementu	<i>Nr_TA</i>	<i>A_IDF</i>	<i>R_IDF</i>
1	<i>Poczta</i>	<i>Hasło</i>	<i>f1</i>
2	<i>Klient</i>	<i>Numer</i>	<i>f3</i>

Rys. 3.9. Tabela z danymi macierzy *N*

Elementy macierzy *M* zamieszczono w tabeli na rys. 3.10.

	Pola elementu macierzy <i>M</i>																			
Typy	<i>przyległość</i>					<i>o_unikat</i>					<i>o_total</i>					<i>odwiedzony</i>				
obiektowe	<i>f1</i>	<i>f2</i>	<i>f3</i>	<i>f4</i>	<i>f5</i>	<i>f1</i>	<i>f2</i>	<i>f3</i>	<i>f4</i>	<i>f5</i>	<i>f1</i>	<i>f2</i>	<i>f3</i>	<i>f4</i>	<i>f5</i>	<i>f1</i>	<i>f2</i>	<i>f3</i>	<i>f4</i>	<i>f5</i>
<i>Hasło</i>	1					1					1					(1)				
<i>Konto</i>		1					1					1					1*			
<i>Poczta</i>	1	1			1	1	1			1	1	1			1	(1)	1			1
<i>Klient</i>			1	1	1			1	1	1			1	1	1			(1)	1	1+
<i>Numer</i>			1					1					1					(1)		
<i>Nazwa</i>				1					0					0					1	

Rys. 3.10. Tabela z danymi macierzy *M*

Elementy macierzy *P* zamieszczono w tabeli na rys. 3.11.

Nazwa pola elementu		
Numer elementu	<i>dziedzina</i>	<i>przeciwdziedzina</i>
1	<i>Hasło</i>	<i>Konto, Numer, Nazwa</i>
2	<i>Numer</i>	<i>Konto, Hasło, Nazwa</i>
3	<i>Konto</i>	<i>Numer, Hasło, Nazwa</i>

Rys. 3.11. Tabela z danymi macierzy *P*

Dane grafu *SR2*.

Elementy macierzy *N* zamieszczono w tabeli na rys. 3.12.

Nazwa pola elementu			
Numer elementu	<i>Nr_TA</i>	<i>A_IDF</i>	<i>R_IDF</i>
1	<i>Klient</i>	<i>Hasło</i>	<i>f1</i>

Rys. 3.12. Tabela z danymi macierzy *N*

Elementy macierzy *M* zamieszczono w tabeli na rys. 3.13.

		Pola elementu macierzy M															
Typy		<i>przyległość</i>				<i>o_unikat</i>				<i>o_total</i>				<i>odwiedzony</i>			
		<i>f1</i>	<i>f2</i>	<i>f3</i>	<i>f4</i>	<i>f1</i>	<i>f2</i>	<i>f3</i>	<i>f4</i>	<i>f1</i>	<i>f2</i>	<i>f3</i>	<i>f4</i>	<i>f1</i>	<i>f2</i>	<i>f3</i>	<i>f4</i>
<i>Hasło</i>	obiektywne	1				1				1				(1)			
<i>Konto</i>			1				1				1				1*		
<i>Klient</i>		1	1	1	1	1	1	1	1	1	1	1	1	(1)	1	1	
<i>Numer</i>				1				1				1				1*	
<i>Nazwa</i>					1				0				0				1

Rys. 3.13. Tabela z danymi macierzy M

Uwaga:

- (1) - oznacza ustawienie pól elementów na podstawie pola A_IdF elementów tabeli N
- 1* - oznacza utworzenie nowego elementu w tabeli P zgodnie p. 2.2.2.1 algorytmu wyznaczenia zależności funkcyjnych
- 1+ - oznacza czynność opisaną w p. 2.2.2.3 algorytmu wyznaczenia zależności funkcyjnych

Elementy macierzy P zamieszczono w tabeli na rys. 3.14.

Numer elementu	Nazwa pola elementu	
	<i>diedzina</i>	<i>przeciwdiedzina</i>
1	<i>Hasło</i>	<i>Konto, Numer, Nazwa</i>
2	<i>Konto</i>	<i>Hasło, Numer, Nazwa</i>
3	<i>Numer</i>	<i>Hasło, Konto, Nazwa</i>

Rys. 3.14. Tabela z danymi macierzy P

Zależności funkcyjne w tabelach P obu grafów $SR1$ i $SR2$ są równoważne na podstawie praw Armstronga [103].

Przykład 2.

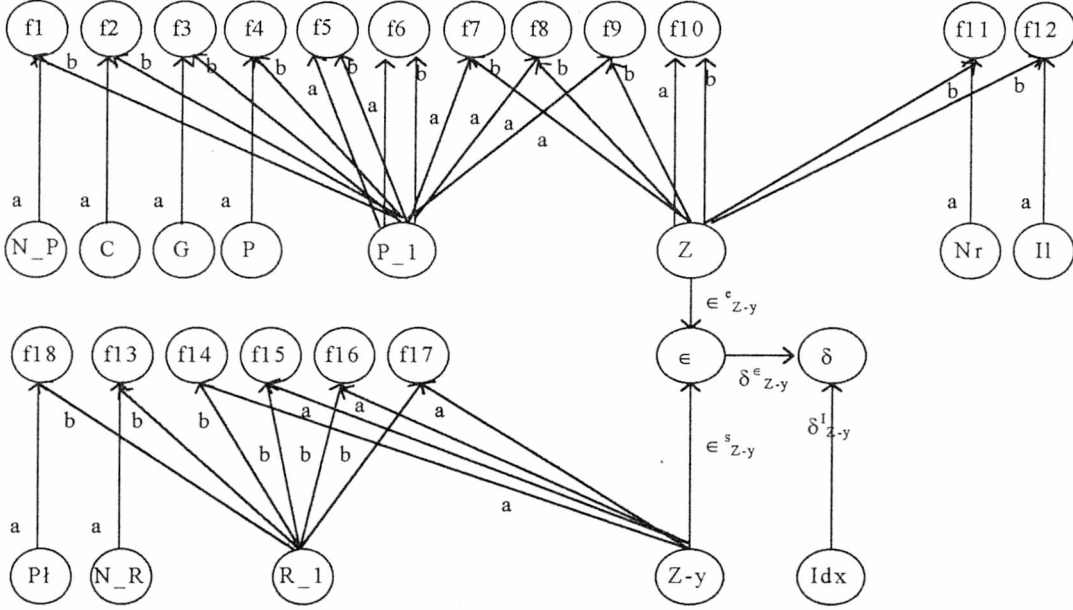
Na podstawie schematów konceptualnych z rozdziału 2.3 (rys. 2.14, 2.15 i 2.16) dokonano w nich analizy zależności funkcyjnych. Na rys. 2.14, 2.15 i 2.16 przyjęto nazwy ról w każdym typie faktu jako a i b (oprócz nazw mnemotechnicznych) oraz zastosowano skróty nazw typów obiektowych. Ułatwia to oznaczanie wierzchołków i krawędzi w odpowiadających grafach funkcjonalnym na rys. 3.15, 3.22, 3.29. W grafach funkcjonalnych pominięto wierzchołki, które nie muszą być uwzględnione przy badaniu związków funkcjonalnych. Wynika to z dziedziczenia i polimorfizmu. Zostały one zaznaczone linią przerywaną.

1. Rozwiązanie 1 - model M_I (rys. 2.14 i 3.15):

1.1. Zależności funkcyjne (2.136, 3.25).

- $$\begin{aligned} \tau 1 &= \{\pi_a f1, \pi_a f2\} \text{ i } AtrybutyIdF(Produkt_1) = \{Nazwa_P, Cena\}, \\ \tau 2 &= \{\pi_a f11\} \text{ i } AtrybutyIdF(Zakup) = \{Numer \text{ zakupu}\}, \\ \tau 3 &= \{\pi_a f13\} \text{ i } AtrybutyIdF(Rachunek_1) = \{Nazwa_R\}, \\ &\bullet \text{ ścieżka } f(\pi_b f3) = f3 \Rightarrow Nazwa_P \text{ Cena} \rightarrow Gatunek, \\ &\bullet \text{ ścieżka } f(\pi_b f4) = f4 \Rightarrow Nazwa_P \text{ Cena} \rightarrow Podatek, \end{aligned}$$

- $\text{ścieżka}_f(\pi_a f8) = f8 \Rightarrow \text{Nazwa_P} \text{ Cena} \rightarrow \text{Numer zakupu}$,
- $\text{ścieżka}_f(\pi_b f8) = f8 \Rightarrow \text{Numer zakupu} \rightarrow \text{Nazwa_P, Cena}$,
- $\text{ścieżka}_f(\pi_b f12) = f12 \Rightarrow \text{Numer zakupu} \rightarrow \text{Ilość}$,
- $\text{ścieżka}_f(\pi_b f14, \in^s \text{Zakupy}) = \delta \pi_a f14 = \in^s \text{Zakupy} \cdot f14 / > < / \in \text{Zakupy}$,
 $\Rightarrow \text{Nazwa_R} \rightarrow \text{Numer zakupu}$,
- $\text{ścieżka}_f(\pi_b f18) = f18 \Rightarrow \text{Nazwa_R} \rightarrow \text{Płatność}$.



Rys. 3.15. Graf funkcjonalny modelu M_1

1.2. Populacja (2.46, A24, 3.35, 3.36).

$\text{Nazwa_P} = \{A, B\}$, $\text{Cena} = \{10, 20\}$,
 $\text{Gatunek} = \{1, 2\}$, $\text{Podatek} = \{7\%, 22\%\}$,
 $\text{Numer zakupu} = \{1, 2\}$, $\text{Nazwa_R} = \{101, 102\}$,
 $\text{Płatność} = \{\text{Przekaz}, \text{Gotówka}\}$,

- $\text{Pop}(\text{ścieżka}_f(\pi_b f3)) = \{\{B, 10, 2\}\}$,
 $\text{Pop}(\text{ścieżka}_f(\pi_b f4)) = \{\{A, 20, 7\%\}\}$,
 $\text{Pop}(\text{ścieżka}_f(\pi_a f8)) = \{\{A, 20, 1\}, \{B, 10, 2\}\}$,
- $\text{Pop}(\text{ścieżka}_f(\pi_b f8)) = \{\{1, A, 20\}, \{2, B, 10\}\}$,
 $\text{Pop}(\text{ścieżka}_f(\pi_b f12)) = \{\{1, 15\}, \{2, 5\}\}$,
- $\text{Pop}(\text{ścieżka}_f(\pi_b f14, \in^s \text{Zakupy})) = \{\{101, 1\}, \{101, 2\}\}$,
 $\text{Pop}(\text{ścieżka}_f(\pi_b f18)) = \{\{101, \text{Przekaz}\}\}$,
- $\text{Pop}_W = \{\{101, \text{Przekaz}, 1, 15, A, 20, 7\%\}, \{101, \text{Przekaz}, 2, 5, B, 10, 2\}\}$.

1.3. Ograniczenia zbiorowe (3.29, 3.31, 3.37).

$\sigma_1 = \{\pi_b f9, \pi_a f9\}$, $\tau_1 = \{\pi_b f8, \pi_a f8\}$, $\delta_1 = \{\pi_b f7, \pi_a f7\}$,
 $\sigma_2 = \{\pi_b f4, \pi_a f4\}$, $\tau_2 = \{\pi_b f1, \pi_a f1\}$, $\delta_2 = \{\pi_b f2, \pi_a f2\}$,
 $\sigma_3 = \{\pi_b f3, \pi_a f3\}$, $\tau_3 = \{\pi_b f1, \pi_a f1\}$, $\delta_3 = \{\pi_b f2, \pi_a f2\}$,
 $\text{Zbiór}_A(\sigma_1 \tau_1) = \{\{\text{Numer zakupu}, \text{Nazwa_P}, \text{Cena}\}_1, \{\text{Numer zakupu}, \text{Nazwa_P}, \text{Cena}\}_1\} =$

$\{\text{Numer zakupu}, \text{Nazwa_P}, \text{Cena}\}_1$
 $\text{Zbiór_A}(\tau_1 \delta_1) = \{\{\text{Podatek}, \text{Nazwa_P}, \text{Cena}\}_1, \{\text{Numer zakupu}, \text{Nazwa_P}, \text{Cena}\}_1\} =$
 $\{\text{Numer zakupu}, \text{Nazwa_P}, \text{Cena}\}_1$
 $\text{Zbiór_A}(\sigma_2 \tau_2) = \{\{\text{Nazwa_P}, \text{Cena}\}_2, \{\text{Nazwa_P}, \text{Cena}\}_2\} = \{\text{Nazwa_P}, \text{Cena}\}_2$
 $\text{Zbiór_A}(\sigma_2 \delta_2) = \{\{\text{Nazwa_P}, \text{Cena}\}_2, \{\text{Nazwa_P}, \text{Cena}\}_2\} = \{\text{Nazwa_P}, \text{Cena}\}_2$
 $\text{Zbiór_A}(\sigma_3 \tau_3) = \{\{\text{Nazwa_P}, \text{Cena}\}_3, \{\text{Nazwa_P}, \text{Cena}\}_3\} = \{\text{Nazwa_P}, \text{Cena}\}_3$
 $\text{Zbiór_A}(\sigma_3 \delta_3) = \{\{\text{Nazwa_P}, \text{Cena}\}_3, \{\text{Nazwa_P}, \text{Cena}\}_3\} = \{\text{Nazwa_P}, \text{Cena}\}_3$
 $\text{Zbiór_B} = \{\{\text{Numer zakupu}, \text{Nazwa_P}, \text{Cena}\}_1, \{\text{Nazwa_P}, \text{Cena}\}_2, \{\text{Nazwa_P}, \text{Cena}\}_3\}$
 $\text{Pop}(\text{Zbiór_B}) = \{\{\{1, A, 20\}, \{2, B, 10\}\}\}$

Wyniki uzyskane za pomocą algorytmu z rozdz. 3.2.2 są podane w tabelach N, M, P na rys. 3.16 - 3.21.

Numer elementu	Nazwa pola elementu macierzy N		
	Nr_TA	A_IDF	R_IDF
1	P_I	N_P, C	$f1, f2$
2	Z	Nr	$f11$
3	Z_y	Nr	$f11$
4	R_I	N_R	$f13$

Rys. 3.16. Tabela z danymi macierzy N

	Pole przyległość elementu macierzy M												
	N_P	C	G	P	P_I	Z	Nr	IL	Z_y	Idx	R_I	N_R	$Pł$
Nr	1	2	3	4	10	11	5	6	12	9	13	7	8
$f1$	1				1								
$f2$		1			1								
$f3$			1		1								
$f4$				1	1								
$f5$					2								
$f6$					2								
$f7$					1	1							
$f8$					1	1							
$f9$					1	1							
$f10$						2							
$f11$						1	1						
$f12$						1		1					
$f13$											1	1	
$f14$									1		1		
$f15$									1		1		
$f16$									1		1		
$f17$									1		1		
$f18$											1		1
ϵ						1			1				
δ									1	1			

Rys. 3.17. Tabela z danymi macierzy M - pole przyległość

Pole o_unikat elementu macierzy M													
	N_P	C	G	P	P_I	Z	Nr	IL	$Z-y$	Idx	R_I	N_R	Pl
Nr	1	2	3	4	10	11	5	6	12	9	13	7	8
f1	1				1								
f2		1			1								
f3			0		1								
f4				0	1								
f5					0								
f6					0								
f7					1	1							
f8					1	1							
f9					1	1							
f10						0							
f11						1	1						
f12						1		0					
f13											1	1	
f14									0		1		
f15									0		1		
f16									0		1		
f17									0		1		
f18											1		0
$\in$						1			1				
δ									1	1			

Rys. 3.18. Tabela z danymi macierzy M - pole o_unikat .

Pole o_total elementu macierzy M													
	N_P	C	G	P	P_I	Z	Nr	IL	$Z-y$	Idx	R_I	N_R	Pl
Nr	1	2	3	4	10	11	5	6	12	9	13	7	8
f1	1				1								
f2		1			1								
f3			0		1								
f4				0	1								
f5					0								
f6					0								
f7					1	1							
f8					1	1							
f9					1	1							
f10						0							
f11						1	1						
f12						1		0					
f13											1	1	
f14									0		1		
f15									0		1		
f16									0		1		
f17									0		1		
f18											1		0

$\in$						1			1				
δ									1	1			

Rys. 3.19. Tabela z danymi macierzy M - pole o_total

Pole odwiedzenie elementu macierzy M													
	N_P	C	G	P	P_I	Z	Nr	IL	$Z-y$	Idx	R_I	N_R	Pt
Nr	1	2	3	4	10	11	5	6	12	9	13	7	8
$f1$	(1)				(1)								
$f2$		(1)			(1)								
$f3$			1		1								
$f4$				1	1								
$f5$					0								
$f6$					0								
$f7$					1	1+							
$f8$					1	1+							
$f9$					1	1+							
$f10$						0							
$f11$						(1)	(1)						
$f12$						1		1					
$f13$											(1)	(1)	
$f14$									1		1		
$f15$									1		1		
$f16$									1		1		
$f17$									1		1		
$f18$											1		1
$\in$						1			1				
δ									1	1			

Rys. 3.20. Tabela z danymi macierzy M - pole odwiedzenie

Numer elementu	Nazwa pola elementu macierzy P	
	<i>dziedzina</i>	<i>przeciwdziedzina</i>
1	N_P, C	G, P, Nr
2	Nr	N_P, C, Nr, IL
3	Nr	Nr, Idx
4	N_R	Nr, Pt

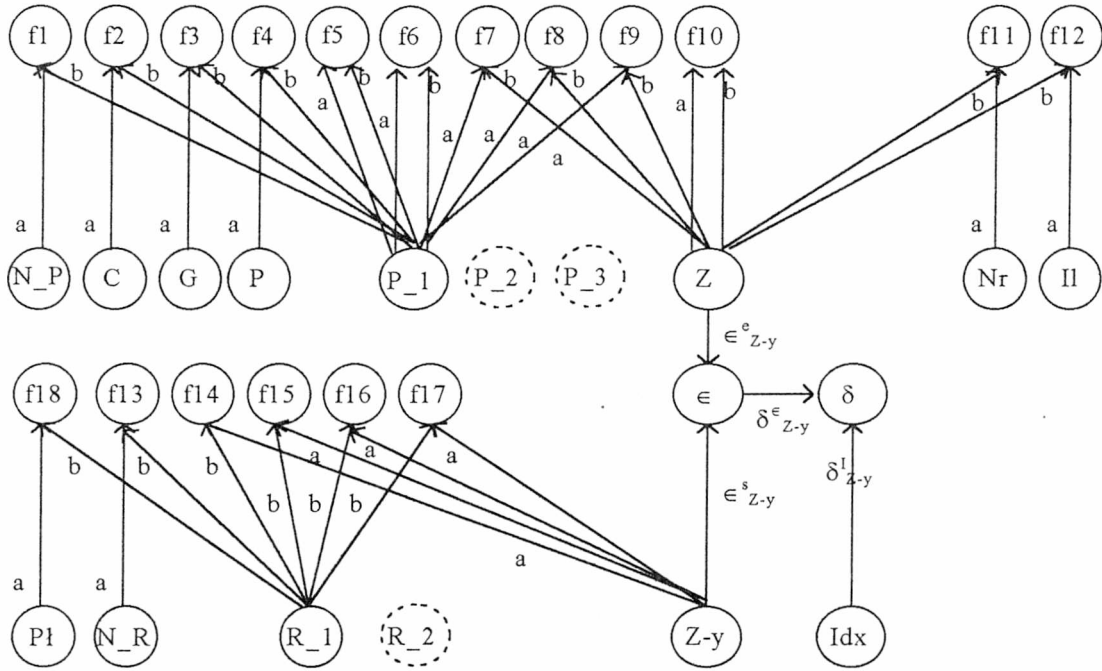
Rys. 3.21. Tabela z danymi macierzy P

2. Rozwiązanie 2 - model M_2 (rys. 2.15 i 3.22):

2.1. Zależności funkcyjne (2.136, 3.25).

- $\tau1 = \{\pi_{af1}, \pi_{af2}\}$ i $AtrybutyIdF(Produkt_1) = \{Nazwa_P, Cena\}$,
- $\tau1 = \{\pi_{af1}, \pi_{af2}\}$ i $AtrybutyIdF(Produkt_2) = \{Nazwa_P, Cena\}$,
- $\tau1 = \{\pi_{af1}, \pi_{af2}\}$ i $AtrybutyIdF(Produkt_3) = \{Nazwa_P, Cena\}$,
- $\tau2 = \{\pi_{af11}\}$ i $AtrybutyIdF(Zakup) = \{Numer\ zakupu\}$,
- $\tau3 = \{\pi_{af13}\}$ i $AtrybutyIdF(Rachunek_1) = \{Nazwa_R\}$,
- $\tau3 = \{\pi_{af13}\}$ i $AtrybutyIdF(Rachunek_2) = \{Nazwa_R\}$,
- ścieżka $f(\pi_b f3) = f3 \Rightarrow Nazwa_P, Cena \rightarrow Gatunek$,

- $\text{ścieżka}_f(\pi_b f4) = f4 \Rightarrow \text{Nazwa_P}, \text{Cena} \rightarrow \text{Podatek},$
- $\text{ścieżka}_f(\pi_a f8) = f8 \Rightarrow \text{Nazwa_P}, \text{Cena} \rightarrow \text{Numer zakupu},$
- $\text{ścieżka}_f(\pi_b f8) = f8 \Rightarrow \text{Numer zakupu} \rightarrow \text{Nazwa_P}, \text{Cena},$
- $\text{ścieżka}_f(\pi_b f12) = f12 \Rightarrow \text{Numer zakupu} \rightarrow \text{Ilość},$
- $\text{ścieżka}_f(\pi_b f14, \in^s \text{Zakupy}_i) = \delta \pi_a f14 = \in^s \text{Zakupy}_i f14 \mid > < \mid \in \text{Zakupy}_i \Rightarrow \text{Nazwa_R} \rightarrow \text{Numer zakupu},$
- $\text{ścieżka}_f(\pi_b f18) = f18 \Rightarrow \text{Nazwa_R} \rightarrow \text{Płatność}.$



Rys. 3.22. Graf funkcjonalny modelu M_2

2.2. Populacja (2.46, A24, 3.35, 3.36).

$\text{Nazwa_P} = \{A, B\},$

$\text{Cena} = \{10, 20\},$

$\text{Gatunek} = \{1, 2\},$

$\text{Podatek} = \{7\%, 22\%\},$

$\text{Numer zakupu} = \{1, 2\},$

$\text{Nazwa_R} = \{101, 102\},$

$\text{Płatność} = \{\text{Przekaz}, \text{Gotówka}\},$

- $\text{Pop}(\text{ścieżka}_f(\pi_b f3)) = \{\{B, 10, 2\}\},$
 $\text{Pop}(\text{ścieżka}_f(\pi_b f4)) = \{\{A, 20, 7\%\}\},$
 $\text{Pop}(\text{ścieżka}_f(\pi_a f8)) = \{\{A, 20, 1\}, \{B, 10, 2\}\},$
- $\text{Pop}(\text{ścieżka}_f(\pi_b f8)) = \{\{1, A, 20\}, \{2, B, 10\}\},$
 $\text{Pop}(\text{ścieżka}_f(\pi_b f12)) = \{\{1, 15\}, \{2, 5\}\},$
- $\text{Pop}(\text{ścieżka}_f(\pi_b f14, \in^s \text{Zakupy}_i)) = \{\{101, 1\}, \{101, 2\}\},$
 $\text{Pop}(\text{ścieżka}_f(\pi_b f18)) = \{\{101, \text{Przekaz}\}\},$
- $\text{Pop}_W = \{\{101, \text{Przekaz}, 1, 15, A, 20, 7\%\}, \{101, \text{Przekaz}, 2, 5, B, 10, 2\}\},$

2.3. Ograniczenia zbiorowe (3.29, 3.31, 3.37).

$$\begin{aligned}
 \sigma_1 &= \{\pi_b f9, \pi_a f9\}, \tau_1 = \{\pi_b f8, \pi_a f8\}, \delta_1 = \{\pi_b f7, \pi_a f7\}, \\
 \sigma_2 &= \{\pi_b f4, \pi_a f4\}, \tau_2 = \{\pi_b f1, \pi_a f1\}, \delta_2 = \{\pi_b f2, \pi_a f2\}, \\
 \sigma_3 &= \{\pi_b f3, \pi_a f3\}, \tau_3 = \{\pi_b f1, \pi_a f1\}, \delta_3 = \{\pi_b f2, \pi_a f2\}, \\
 \text{Zbiór}_A(\sigma_1 \tau_1) &= \{\{\text{Numer zakupu}, \text{Nazwa_P}, \text{Cena}\}_1, \{\text{Numer zakupu}, \text{Nazwa_P}, \text{Cena}\}_1\} \\
 &= \{\text{Numer zakupu}, \text{Nazwa_P}, \text{Cena}\}_1, \\
 \text{Zbiór}_A(\sigma_1 \delta_1) &= \{\{\text{Numer zakupu}, \text{Nazwa_P}, \text{Cena}\}_1, \{\text{Numer zakupu}, \text{Nazwa_P}, \text{Cena}\}_1\} \\
 &= \{\text{Numer zakupu}, \text{Nazwa_P}, \text{Cena}\}_1, \\
 \text{Zbiór}_A(\sigma_2 \tau_2) &= \{\{\text{Nazwa_P}, \text{Cena}\}_2, \{\text{Nazwa_P}, \text{Cena}\}_2\} = \{\text{Nazwa_P}, \text{Cena}\}_2, \\
 \text{Zbiór}_A(\sigma_2 \delta_2) &= \{\{\text{Nazwa_P}, \text{Cena}\}_2, \{\text{Nazwa_P}, \text{Cena}\}_2\} = \{\text{Nazwa_P}, \text{Cena}\}_2, \\
 \text{Zbiór}_A(\sigma_3 \tau_3) &= \{\{\text{Nazwa_P}, \text{Cena}\}_3, \{\text{Nazwa_P}, \text{Cena}\}_3\} = \{\text{Nazwa_P}, \text{Cena}\}_3, \\
 \text{Zbiór}_A(\sigma_3 \delta_3) &= \{\{\text{Nazwa_P}, \text{Cena}\}_3, \{\text{Nazwa_P}, \text{Cena}\}_3\} = \{\text{Nazwa_P}, \text{Cena}\}_3, \\
 \text{Zbiór}_B &= \{\{\text{Numer zakupu}, \text{Nazwa_P}, \text{Cena}\}_1, \{\text{Nazwa_P}, \text{Cena}\}_2, \{\text{Nazwa_P}, \text{Cena}\}_3\}, \\
 \text{Pop}(\text{Zbiór}_B) &= \{\{1, A, 20\}, \{2, B, 10\}\}.
 \end{aligned}$$

Wyniki uzyskane za pomocą algorytmu z rozdz. 3.2.2 są podane w tabelach N , M , P na rys. 3.23 - 3.28.

Numer elementu	Nazwa pola elementu macierzy N		
	Nr_TA	A_IDF	R_IDF
1	P_I	N_P, C	$f1, f2$
2	Z	Nr	$f11$
3	$Z-y$	Nr	$f11$
4	R_I	N_R	$f13$

Rys. 3.23. Tabela z danymi macierzy N

	Pole przyległość elementu macierzy M												
	N_P	C	G	P	P_I	Z	Nr	IL	$Z-y$	Idx	R_I	N_R	$Pł$
Nr	1	2	3	4	10	11	5	6	12	9	13	7	8
$f1$	1				1								
$f2$		1			1								
$f3$			1		1								
$f4$				1	1								
$f5$					2								
$f6$					2								
$f7$					1	1							
$f8$					1	1							
$f9$					1	1							
$f10$						2							
$f11$						1	1						
$f12$						1		1					
$f13$											1	1	
$f14$									1		1		
$f15$									1		1		
$f16$									1		1		

f17									1		1		
f18											1		1
ε						1			1				
δ									1	1			

Rys. 3.24. Tabela z danymi macierzy M - pole przyległość

Pole o_unikat elementu macierzy M													
	N_P	C	G	P	P_I	Z	Nr	IL	$Z-y$	Idx	R_I	N_R	$Pł$
Nr	1	2	3	4	10	11	5	6	12	9	13	7	8
f1	1				1								
f2		1			1								
f3			0		1								
f4				0	1								
f5					0								
f6					0								
f7					1	1							
f8					1	1							
f9					1	1							
f10						0							
f11						1	1						
f12						1		0					
f13											1	1	
f14									0		1		
f15									0		1		
f16									0		1		
f17									0		1		
f18											1		0
ε						1			1				
δ									1	1			

Rys. 3.25. Tabela z danymi macierzy M - pole o_unikat

Pole o_total elementu macierzy M													
	N_P	C	G	P	P_I	Z	Nr	IL	$Z-y$	Idx	R_I	N_R	$Pł$
Nr	1	2	3	4	10	11	5	6	12	9	13	7	8
f1	1				1								
f2		1			1								
f3			0		1								
f4				0	1								
f5					0								
f6					0								
f7					1	1							
f8					1	1							
f9					1	1							
f10						0							
f11						1	1						
f12						1		0					
f13											1	1	

f14									0		1		
f15									0		1		
f16									0		1		
f17									0		1		
f18											1		0
ε						1			1				
δ									1	1			

Rys. 3.26. Tabela z danymi macierzy M - pole o_total

Pole odwiedzenie elementu macierzy M													
	N_P	C	G	P	P_I	Z	Nr	IL	$Z-y$	Idx	R_I	N_R	$Pł$
Nr	1	2	3	4	10	11	5	6	12	9	13	7	8
f1	(1)				(1)								
f2		(1)			(1)								
f3			1		1								
f4				1	1								
f5					0								
f6					0								
f7					1	1+							
f8					1	1+							
f9					1	1+							
f10						0							
f11						(1)	(1)						
f12						1		1					
f13											(1)	(1)	
f14									1		1		
f15									1		1		
f16									1		1		
f17									1		1		
f18											1		1
ε						1			1				
δ									1	1			

Rys. 3.27. Tabela z danymi macierzy M - pole odwiedzenie

Numer elementu	Nazwa pola elementu macierzy P	
	<i>dziedzina</i>	<i>przeciwdziedzina</i>
1	$N P, C$	G, P, Nr
2	Nr	$N P, C, Nr, IL$
3	Nr	Nr, Idx
4	$N R$	$Nr, Pł$

Rys. 3.28. Tabela z danymi macierzy P

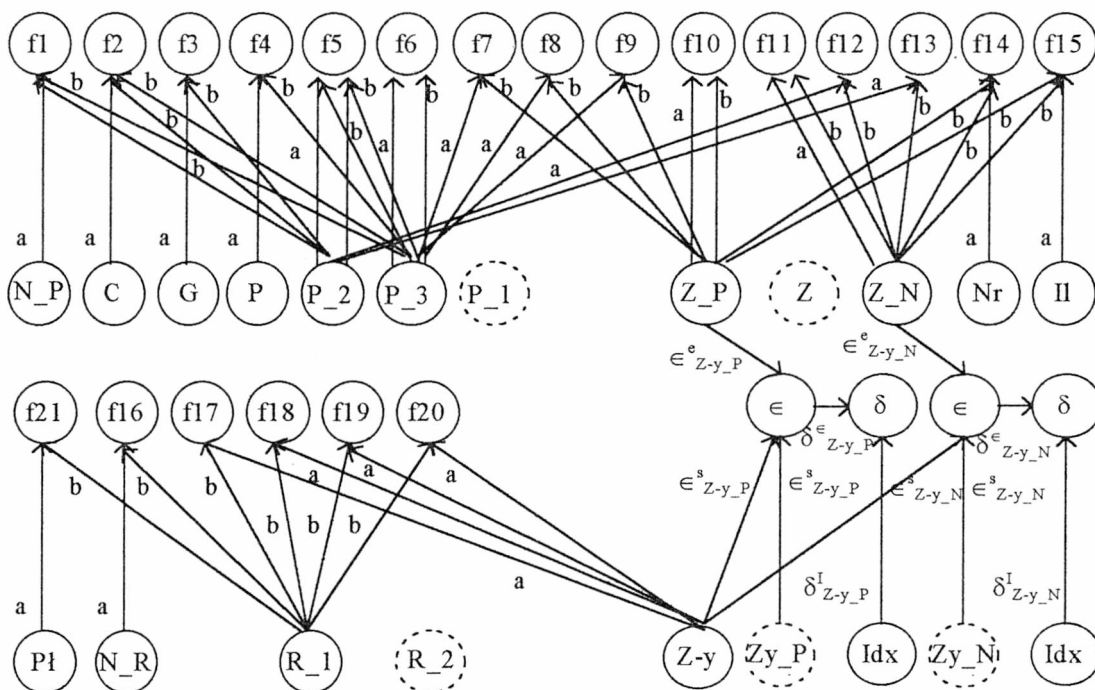
3. Rozwiązanie 3 - model M_3 (rys. 2.16 i 3.29):

3.1. Zależności funkcyjne (2.136, 3.25).

$\tau 1 = \{\pi_q f1, \pi_q f2\}$ i $AtrybutyIdF (Produkt_1) = \{Nazwa_P, Cena\}$,

$\tau 1 = \{\pi_q f1, \pi_q f2\}$ i $AtrybutyIdF (Produkt_2) = \{Nazwa_P, Cena\}$,

- $\tau_1 = \{\pi_{af1}, \pi_{af2}\}$ i $AtrybutyIdF(Produkt_3) = \{Nazwa_P, Cena\}$,
 $\tau_2 = \{\pi_{af14}\}$ i $AtrybutyIdF(Zakup) = \{Numer\ zakupu\}$,
 $\tau_2 = \{\pi_{af14}\}$ i $AtrybutyIdF(Zakup_P) = \{Numer\ zakupu\}$,
 $\tau_2 = \{\pi_{af14}\}$ i $AtrybutyIdF(Zakup_N) = \{Numer\ zakupu\}$,
 $\tau_3 = \{\pi_{af16}\}$ i $AtrybutyIdF(Rachunek_1) = \{Nazwa_R\}$,
 $\tau_3 = \{\pi_{af16}\}$ i $AtrybutyIdF(Rachunek_2) = \{Nazwa_R\}$,
 • $\text{ścieżka}_f(\pi_{bf3}) = f3 \Rightarrow Nazwa_P, Cena \rightarrow Gatunek$,
 • $\text{ścieżka}_f(\pi_{bf4}) = f4 \Rightarrow Nazwa_P, Cena \rightarrow Podatek$,
 • $\text{ścieżka}_f(\pi_{af8}) = f8 \Rightarrow Nazwa_P, Cena \rightarrow Numer\ zakupu$,
 • $\text{ścieżka}_f(\pi_{af12}) = f12 \Rightarrow Nazwa_P, Cena \rightarrow Numer\ zakupu$,
 • $\text{ścieżka}_f(\pi_{bf8}, \pi_{af8}) = f8 \Rightarrow Numer\ zakupu \rightarrow Nazwa_P, Cena$,
 • $\text{ścieżka}_f(\pi_{bf12}) = f12 \Rightarrow Numer\ zakupu \rightarrow Nazwa_P, Cena$,
 • $\text{ścieżka}_f(\pi_{bf15}) = f15 \Rightarrow Numer\ zakupu \rightarrow Ilość$,
 • $\text{ścieżka}_f(\pi_{bf17}, \in^S Zakupy) = \delta \pi_{af17} = \in^S Zakupy f17 / > < / \in Zakupy$
 $\Rightarrow Nazwa_R \rightarrow Numer\ zakupu$,
 • $\text{ścieżka}_f(\pi_{bf21}) = f21 \Rightarrow Nazwa_R \rightarrow Płatność$.



Rys. 3.29. Graf funkcjonalny modelu M_3

3.2. Populacja (2.46, A24, 3.35, 3.36).

- | | |
|---|-----------------------------|
| $Nazwa_P = \{A, B\}$, | $Cena = \{10, 20\}$, |
| $Gatunek = \{1, 2\}$, | $Podatek = \{7\%, 22\%\}$, |
| $Numer\ zakupu = \{1, 2\}$, | $Nazwa_R = \{101, 102\}$, |
| $Płatność = \{Przekaz, Gotówka\}$, | |
| • $Pop(\text{ścieżka}_f(\pi_{bf3})) = \{\{B, 10, 2\}\}$, | |

- $Pop(\text{ścieżka}_f(\pi_b f4) = \{\{A, 20, 7\%\}\},$
 $Pop(\text{ścieżka}_f(\pi_a f8) = \{\{A, 20, 1\}\},$
 $Pop(\text{ścieżka}_f(\pi_a f12) = \{\{B, 10, 2\}\},$
 • $Pop(\text{ścieżka}_f(\pi_b f8) = \{\{1, A, 20\}\},$
 $Pop(\text{ścieżka}_f(\pi_b f12) = \{\{2, B, 10\}\},$
 $Pop(\text{ścieżka}_f(\pi_b f15) = \{\{1, 15\}, \{2, 5\}\},$
 • $Pop(\text{ścieżka}_f(\pi_b f17, \in^S \text{Zakupy}) = \{\{101, 1\}, \{101, 2\}\},$
 $Pop(\text{ścieżka}_f(\pi_b f21) = \{\{101, Przekaz\}\},$
 • $Pop_W = \{\{101, Przekaz, 1, 15, A, 20, 7\% \}, \{101, Przekaz, 2, 5, B, 10, 2\}\},$

3.3. Ograniczenia zbiorowe (3.29, 3.31, 3.37, 3.38).

$\sigma_1 = \{\pi_b f7, \pi_a f7\}, \tau_1 = \{\pi_b f8, \pi_a f8\}, \delta_1 = \{\pi_b f9, \pi_a f9\},$
 $\sigma_2 = \{\pi_b f12, \pi_a f12\}, \tau_2 = \{\pi_b f13, \pi_a f13\},$
 $Zbiór_A(\sigma_1 \tau_1) = \{\{Numer\ zakupu, Nazwa_P, Cena\}_1, \{Numer\ zakupu, Nazwa_P, Cena\}_1\}$
 $= \{Numer\ zakupu, Nazwa_P, Cena\}_1,$
 $Zbiór_A(\tau_1 \delta_1) = \{\{Numer\ zakupu, Nazwa_P, Cena\}_1, \{Numer\ zakupu, Nazwa_P, Cena\}_1\}$
 $= \{Numer\ zakupu, Nazwa_P, Cena\}_1,$
 $Zbiór_A(\sigma_2 \tau_2) = \{\{Numer\ Zakupu, Nazwa_P, Cena\}_2, \{Numer\ zakupu, Nazwa_P, Cena\}_2\}$
 $= \{Numer\ zakupu, Nazwa_P, Cena\}_2,$
 $Zbiór_B = \{\{Numer\ zakupu, Nazwa_P, Cena\}_1, \{Numer\ zakupu, Nazwa_P, Cena\}_2\},$
 $Pop(Zbiór_B) = \{\{1, A, 20\}, \{2, B, 10\}\}.$

Wyniki uzyskane za pomocą algorytmu z rozdz. 3.2.3 są podane w tabelach N, M, P na rys. 3.30 - 3.35.

Numer elementu	Nazwa pola elementu macierzy N		
	Nr_TA	A_IDF	R_IDF
1	P_2	N_P, C	$N_P, f1; C, f2$
2	P_3	N_P, C	$N_P, f1; C, f2$
3	Z_P	Nr	$Nr, f14$
4	Z_N	Nr	$Nr, f14$
5	Z_y	Nr	$Nr, 14$
6	R_1	N_R	$N_R, f16$

Rys. 3.30. Tabela z danymi macierzy N

	Pole przyległość elementu macierzy M															
	N_P	C	G	P	P_2	P_3	Z_P	Z_N	Nr	IL	Z_y	Idx_P	Idx_N	R_1	N_R	PI
Nr	1	2	3	4	11	12	13	14	5	6	15	9	10	16	7	8
$f1$	1				1	1										
$f2$		1			1	1										
$f3$			1		1											
$f4$				1		1										
$f5$					2	2										

f6						2										
f7						1	1									
f8						1	1									
f9						1	1									
f10							2									
f11								2								
f12					1			1								
f13					1			1								
f14							1	1	1							
f15							1	1		1						
f16														1	1	
f17										1				1		
f18										1				1		
f19										1				1		
f20										1				1		
f21														1		1
ϵ_{zP}							1				1					
δ_{zP}											1	1				
ϵ_{zN}								1			1					
δ_{zN}											1		1			

Rys. 3.31. Tabela z danymi macierzy M - pole przyległość

Pole o_unikat elementu macierzy M																
	N_P	C	G	P	P_2	P_3	Z_P	Z_N	Nr	IL	$Z-y$	Idx_P	Idx_N	R_I	N_R	Pl
Nr	1	2	3	4	11	12	13	14	5	6	15	9	10	16	7	8
f1	1				1	1										
f2		1			1	1										
f3			0		1											
f4				0		1										
f5					0	0										
f6						0										
f7						1	1									
f8						1	1									
f9						1	1									
f10							0									
f11								0								
f12					1			1								
f13					1			1								
f14							1	1	1							
f15							1	1		0						
f16														1	1	
f17											0			1		
f18											0			1		
f19											0			1		
f20											0			1		
f21														1		0
ϵ_{zP}							1				1					

δ_{zP}											1	1				
ϵ_{zN}								1			1					
δ_{zN}											1		1			

Rys. 3.32. Tabela z danymi macierzy M - pole o_unikat

Pole o_total elementu macierzy M																
	N_P	C	G	P	P_2	P_3	Z_P	Z_N	Nr	IL	$Z-y$	Idx_P	Idx_N	R_I	N_R	Pt
Nr	1	2	3	4	11	12	13	14	5	6	15	9	10	16	7	8
$f1$	1				1	1										
$f2$		1			1	1										
$f3$			0		1											
$f4$				0		1										
$f5$					0	0										
$f6$						0										
$f7$						1	1									
$f8$						1	1									
$f9$						1	1									
$f10$							0									
$f11$								0								
$f12$					1			1								
$f13$					1			1								
$f14$							1	1	1							
$f15$							1	1		0						
$f16$														1	1	
$f17$											0			1		
$f18$											0			1		
$f19$											0			1		
$f20$											0			1		
$f21$														1		0
ϵ_{zP}							1				1					
δ_{zP}											1	1				
ϵ_{zN}								1			1					
δ_{zN}											1		1			

Rys. 3.33. Tabela z danymi macierzy M - pole o_total

Pole odwiedzenie elementu macierzy M																
	N_P	C	G	P	P_2	P_3	Z_P	Z_N	Nr	IL	$Z-y$	Idx_P	Idx_N	R_I	N_R	Pt
Nr	1	2	3	4	11	12	13	14	5	6	15	9	10	16	7	8
$f1$	(1)				(1)	(1)										
$f2$		(1)			(1)	(1)										
$f3$			1		1											
$f4$				1		1										
$f5$					0	0										
$f6$						0										
$f7$						1	1+									

f_8						1	1+									
f_9						1	1+									
f_{10}							0									
f_{11}								0								
f_{12}					1			1+								
f_{13}					1			1+								
f_{14}							(1)	(1)	(1)							
f_{15}							1	1		1						
f_{16}														(1)	(1)	
f_{17}										1				1		
f_{18}										1				1		
f_{19}										1				1		
f_{20}										1				1		
f_{21}														1		1
ϵ_{zP}							1				1					
δ_{zP}											1	1				
ϵ_{zN}								1			1					
δ_{zN}											1		1			

Rys. 3.34. Tabela z danymi macierzy M - pole odwiedzenie

Elementy macierzy P zamieszczono w tabeli na rys. 3.28.

Numer elementu	Nazwa pola elementu macierzy P	
	<i>dziedzina</i>	<i>przeciwdziedzina</i>
1	$N\ P,\ C$	$G,\ Nr$
2	$N\ P,\ C$	$P,\ Nr$
3	Nr	$N\ P,\ C,\ Nr,\ IL$
4	Nr	$N\ P,\ C,\ Nr$
5	Nr	$Nr,\ Idx\ N,\ Idx\ P$
6	$N\ R$	$Nr,\ Pl$

Rys. 3.35. Tabela z danymi macierzy P

Wniosek

Analiza zależności funkcjonalnych, populacji wynikającej z zależności funkcjonalnych oraz ograniczeń zbiorowych wykazała równoważność w sensie E4 modeli: M_1 , M_2 i M_3 .

4. Algorytm podziału schematu konceptualnego na klastry jako podstawa specyfikacji klas

Podział schematu konceptualnego na klastry stanowi podstawę do wykonania specyfikacji modelu zorientowanej obiektowo. Podział ten polega na zgromadzeniu typów obiektowych według znaczenia ich powiązań strukturalnych i funkcjonalnych. Łączenie typów ORM w podzbiory określane jako klastry przebiega zgodnie z pewnymi zasadami, które zostaną przedstawione w dalszej części pracy. Klaster wyznacza klasy podane w specyfikacji modelu zorientowanej obiektowo. Powiązania między klastrami odpowiadają powiązaniom strukturalnym i funkcjonalnym między klasami w tej specyfikacji.

Model klastrowy pozwala również badać efektywność oprogramowania zorientowanego obiektowo na poziomie modelu, a w szczególności spójność oraz ilość połączeń.

Podział na klastry przedstawiono w pracach [2, 34, 43]. W pracy [100] podano algorytm do wyodrębniania klas w schemacie konceptualnym typu ER na podstawie analizy ograniczeń. Proces ten polega na przekształceniu schematu ER w hypergraf i następnie stopniowej redukcji elementów hypergrafu przy jednoczesnej budowie drzewa klas.

W pracy [2] jest podany algorytm podziału na klastry schematu ER oraz obiektowego schematu OOA w celu ograniczenia ich złożoności w przypadku dużych aplikacji. Podejście to odpowiada podejściu zaprezentowanemu w pracy [34].

W pracy [34] autorzy opracowali sposób przedstawiania schematu konceptualnego ORM na różnych poziomach abstrakcji w celu stopniowego uwidaczniania szczegółów modelu. Podział na klastry nie odpowiadał jednak łączeniu elementów modelu w części, które można byłoby potraktować jako modele przyszłych obiektów zgodnie z wymaganiami techniki obiektowej. Nie opracowano zasad zagnieżdżania klastrow wynikających ze związku typu agregacja lub związku funkcjonalnego nie powiązanych strukturalnie klas.

W pracy [43] autorzy również zajęli się ujmowaniem w klastry elementów modelu ORM. Jednak to podejście polegało na stopniowej budowie modelu za pomocą klastrow, które zagnieżdżano w istniejących. Odpowiadało to rozbudowie modelu na kolejnym poziomie abstrakcji ukazującym strukturę zagnieżdżonych klastrow. Po zakończeniu rozbudowy klastrow powstawał model końcowy, który mógł być potraktowany jako zbiór klas modelu zorientowanego obiektowo. Jednocześnie likwidując podział na klastry można było uzyskać poprawny diagram ORM. W tym podejściu nie przedstawiono metody tworzenia klastrow, lecz podano konkretny przykład. Na tym przykładzie przedstawiono związek między elementami klastrow a elementami struktury informacyjnej, natomiast nie podano wszystkich podstaw formalnych decydujących o tym, które elementy ORM należy połączyć w klastry.

W rozdziale 4.1 niniejszej pracy podano podstawy formalne budowy klastrow, w rozdziale 4.2 sformułowano algorytm do budowy klastrow, oparty na analizie ograniczeń między danymi. W rozdziale 4.3 podano przykład zastosowania metody do budowy modelu klastrowego ORM.

4.1. Podstawy formalne budowy klastrowej schematu konceptualnego

4.1.1. Syntaktyka klastrów

Podział struktury informacyjnej na klastry jest określony za pomocą funkcji (2.1)

$$Klaster: N \times M \times TOB \rightarrow \varphi(TOB). \quad \{*\}(4.1)$$

Oznacza to, że funkcja

$$Klaster(i, j, x) = y \quad \{*\}(4.2)$$

zawiera zbiór y typów obiektowych reprezentujących klasę x na i -tym poziomie syntezy, zwanym również poziomem abstrakcji oraz j -tym poziomem ukrywania zagnieżdżenia swojej struktury. Klasa x odpowiada jednemu z typów abstrakcyjnych przydzielonych do klastra, pełniących nadrzędną rolę w stosunku do pozostałych typów. Klasa pełniąca nadrzędną rolę jest określana jako klasa wiodąca klastra.

Funkcja wyznaczająca elementy składowe klastra jest zdefiniowana jako

$$ZbKLaster(i, j, x) = \{y \mid (y \in TOB) \wedge (y \in Klaster(i, j, x) \wedge \exists y [y \in Klaster(i+1, t, y)])\}. \quad \{*\}(4.3)$$

Elementy składowe klastra wyznaczonego na i -tym poziomie syntezy, zgodnie z powyższą funkcją, są następujące:

- elementy o ujawnionej strukturze należące do zbioru typów obiektowych TOB ,
- elementy przydzielone do klastra $i+1$ -ego zagnieżdżonego w i -tym klastrze.

Funkcja określająca zbiór klastrów na i -tym poziomie syntezy jest zdefiniowana jako

$$Klaster_i: N \rightarrow \varphi(TOB)$$

czyli mamy

$$Klaster_i(i) = \bigcup_{x \in TOB} Klaster(i, j, x). \quad \{*\}(4.4)$$

Funkcja określająca strukturę zagnieżdżoną klastra na i -tym poziomie syntezy jest zdefiniowana jako

$$Klaster_x: N \times TOB \rightarrow \varphi(TOB) \quad \{*\}$$

czyli mamy

$$Klaster_x(i, x) = \bigcup_{j \in N} Klaster(i, j, x). \quad \{*\}(4.5)$$

Własności syntaktyki klastra są następujące:

1. Typ obiektowy wiodący klastra spełnia zależność

$$x \in Klaster(i, j, x). \quad \{*\}(4.6)$$

2. Klaster na $i+1$ -tym poziomie syntezy jest podzbiorem klastra na i -tym poziomie syntezy, czyli

$$i < j \Rightarrow Klaster(j, t, x) \subseteq Klaster(i, r, x). \quad \{*\}(4.7)$$

3. Klaster na i -tym poziomie ukrywania zagnieżdżenia jest podzbiorem klastra na $i+1$ -tym poziomie ukrywania zagnieżdżenia, czyli

$$i < j \Rightarrow \text{Klaster}(t, i, x) \subseteq \text{Klaster}(r, j, x). \quad \{*\}(4.8)$$

4. Klastry są dziedziczone między typami wiodącymi, jeżeli spełniają relację specjalizacji *Spec* (2.7, rozdz. 2.1.2), czyli

$$x \text{ Spec } y \Rightarrow \text{Klaster}(i, t, y) \subseteq \text{Klaster}(i, r, x). \quad \{*\}(4.9)$$

5. Typy obiektowe będące w relacji polimorfizmu *Pol* wyznaczają typ wiodący klastra jako sumę tych typów (2.8, rozdz. 2.1.2), czyli

$$x \text{ Pol } y \Rightarrow \text{Klaster}(i, j, x) \equiv \text{Klaster}(i, j, y \cup x). \quad \{*\}(4.10)$$

6. Wszystkie typy bazowe należące do typów faktów przydzielonych do danego klastra muszą spełniać zależność (2.4)

$$\text{Bazy}(\text{Klaster}(i, j, x)) \subseteq \text{Klaster}(i, j, x). \quad \{*\}(4.11)$$

7. Zasada przechodniości dla typów będących w relacji *Spec* jest następująca (A8 w rozdz. 2.1.2):

$$a \text{ Spec } b \text{ Spec } c \wedge a, c \in \text{Klaster}(i, j, x) \Rightarrow b \in \text{Klaster}(i, j, x). \quad \{*\}(4.12)$$

8. Zasada przechodniości dla typów będących w relacji *Pol* jest następująca (A14 w rozdz. 2.1.2):

$$a \text{ Pol } b \text{ Pol } c \wedge a, c \in \text{Klaster}(i, j, x) \Rightarrow b \in \text{Klaster}(i, j, x). \quad \{*\}(4.13)$$

9. Drzewo dziedziczenia musi należeć w całości do klastra (2.9), czyli

$$y \in \text{Klaster}(i, j, x) \Rightarrow \text{Korzeń}(y) \in \text{Klaster}(i, j, x). \quad \{*\}(4.14)$$

10. Klaster jest podgrafem struktury informacyjnej *SI* (2.1), czyli

$$\text{Klaster}(i, j, x) \subseteq \text{SI}. \quad \{*\}(4.15)$$

11. Klaster jest podgrafem połączonym z oryginalnym schematem konceptualnym ORM (2.1), czyli

$$\text{SI} = \{\text{TOB} - \text{Klaster}(i, j, x)\} \cup \{y\} \quad \{*\}(4.16)$$

oraz

$$\text{SI} = \{\text{TOB} - \text{Klaster}(i, x)\} \cup \{y\}. \quad \{*\}(4.17)$$

12. Typem wiodącym klastra nie może być typ atrybutowy, czyli

$$x \in \text{TWart} \Rightarrow \neg \text{Klaster}(i, x) = x. \quad \{*\}(4.18)$$

4.1.2. Semantyka klastrów

W niniejszej pracy podano, podobnie jak przy opisie syntaktyki, tylko pewne charakterystyczne cechy semantyki klastrów. Ponieważ *Klaster* jest podgrafem połączonym ze strukturą informacyjną schematu konceptualnego, cechy semantyczne elementów klastra są identyczne z cechami schematu konceptualnego.

a) Ograniczenia

Przy podziale schematu konceptualnego na klastry ograniczenia łączące role odgrywane przez różne typy obiektowe są przydzielane do tego klastra, w którym są

zawarte te typy obiektowe. Role objęte ograniczeniami *unikat* i *total* oraz ograniczenia zbiorowe (*Równość*, *Podzbiór* i *Rozłączność*) służą do wyznaczania wiodących typów obiektowych określających klastry oraz związków strukturalnych i funkcjonalnych pomiędzy klastrami.

Jest to ustalane w następujący sposób:

- a) w sekwencji ról objętych ograniczeniami są sprawdzane wszystkie typy bazowe ról należących do sekwencji,
- b) typy obiektowe z punktu a, wyznaczone z ograniczeń *total* i *unikat*, związki *n:m*, *1:n* oraz ścieżki identyfikacji *RoleIdF* (2.134) lub *RoleIdF'* (2.141) określają typy wiodące klastrow lub zagnieżdżenia klastrow i typów obiektowych. Rolę wiodącą można również wyznaczyć projektant modelu,
- c) jeżeli wśród typów obiektowych, określonych w punkcie a na podstawie ograniczeń zbiorowych, znajduje się typ wiodący wyznaczający klastrow główny (np. na podstawie punktu b), natomiast pozostałe typy bazowe są typami wiodącymi innych klastrow lub są typami atrybutowymi, należy te klastry zagnieżdżyć w klastrze głównym. Wtedy sekwencja ograniczeń jest zawarta w klastrze głównym.

W rozdziale 4.2. opracowano algorytm podziału schematu ORM na klastry na podstawie klasyfikacji przypadków wynikających z ograniczeń.

b) Hermetyzacja

Przy podziale schematu konceptualnego na klastry zastosowano mechanizm hermetyzacji. Polega on na tym, że klastry ukrywają swoją strukturę. Mechanizm ten opisany jest przez funkcję

$$Hmt: TOB \rightarrow \varphi(TOB). \quad (4.19)$$

Cechy te opisane są następująco:

1. Typy obiektowe objęte hermetyzacją zawsze należą do klastra

$$y \in Hmt(x) \cap TOB \Rightarrow \exists_i \exists_j [y \in Klaster(i, j, x)]. \quad (4.20)$$

2. Jeżeli dwa typy występują w relacji specjalizacji *Spec* i są objęte hermetyzacją, należą do tego samego zbioru wyznaczonego funkcją *Hmt* (2.7, rozdz. 2.1.2), czyli

$$x Spec y \Rightarrow Hmt(y) \subseteq Hmt(x). \quad (4.21)$$

3. Jeżeli dwa typy występują w relacji specjalizacji *Pol* i są objęte hermetyzacją, należą do tego samego zbioru wyznaczonego funkcją *Hmt* (2.8, rozdz. 2.1.2), czyli

$$x Pol y \Rightarrow Hmt(y) \subseteq Hmt(x). \quad \{*\}(4.22)$$

4. Na *i*-tym poziomie syntezy uwidocznione są typy obiektowe nie należące do klastrow na wyższych poziomach (4.4), czyli

$$Widz(i) = Klaster_i(i) - \bigcup_y Hmt(y). \quad \{*\}(4.23)$$

Funkcja wyznaczająca typy obiektowe widziane z *i*-tego klastra jest następująca:

$$Widz(i, x) = Klaster(i, j, x) - \bigcup_y Hmt(y) \cup Hmt(x). \quad \{*\}(4.24)$$

c) Identyfikacja

Proces identyfikacji typów obiektowych klastra $i+I$ -ego powinien być zawarty na poziomie klastra i - tego. Jest to określone następująco (2.122, 2.132, 2.139):

$$y \in \text{Klaster}(i, j, x) \wedge \text{JestIdF}(y) \wedge y \text{ IdF } \tau \Rightarrow \text{SumaIdF}(x, \tau) \subseteq \text{Klaster}(i, j, x). \quad \{*\}(4.25)$$

d) Populacja

Populacja klastrów jest równoważna populacji struktury informacyjnej (2.52, rozdz. 2.2.1), do której należą klastry. Funkcja opisująca populację spełniającą dany klaster jest następująca:

$$\text{PopKlaster}: TA \rightarrow \wp(\Omega). \quad (4.26)$$

Jest to populacja typu obiektowego y należącego do zbioru typów obiektowych opisanych funkcją $\text{Klaster_x}(i, x)$ (4.5).

Mamy więc

$$\text{PopKlaster}(x, y), \text{ gdzie } y \in \text{Klaster_x}(i, x) \cap TOb. \quad (4.27)$$

Reguły opisujące związek populacji z elementami klastra są następujące:

1. Wystąpienie p typu x jest również wystąpieniem klastra typu x , czyli

$$p \in \text{Pop}(x) \vdash p \in \text{PopKlaster}(x, x). \quad (4.28)$$

2. Związek populacji typów obiektowych będących w relacji specjalizacji Spec i należące do tego samego klastra jest zdefiniowany jako (rozdz. 2.1.2, 4.5)

$$\begin{aligned} \text{jeśli } a \text{ Spec } b \wedge a, b \in \text{Klaster_x}(i, x) \text{ to wtedy } p \in \text{PopKlaster}(x, a) \\ \vdash p \in \text{PopKlaster}(x, b). \end{aligned} \quad (4.29)$$

3. Związek populacji typów obiektowych będących w relacji specjalizacji Spec , dotyczący populacji podtypu w odniesieniu do ich populacji klastrowej jest zdefiniowany jako (rozdz. 2.1.2, 4.5)

$$\begin{aligned} \text{jeśli } a \text{ Spec } b \wedge a, b \in \text{Klaster_x}(i, x), \text{ to } p \in \text{PopKlaster}(x, b) \cap \text{Pop}(a) \\ \vdash p \in \text{PopKlaster}(x, a). \end{aligned} \quad (4.30)$$

4. Związek populacji typów obiektowych będących w relacji polimorfizmu Pol jest zdefiniowany jako (rozdz. 2.1.2, 4.5)

$$\begin{aligned} \text{jeśli } a \text{ Pol } b \wedge a, b \in \text{Klaster_x}(i, x) \text{ to } p \in \text{PopKlaster}(x, b) \\ \vdash p \in \text{PopKlaster}(x, a). \end{aligned} \quad \{*\}(4.31)$$

5. Związek populacji typów obiektowych będących w relacji specjalizacji Pol , dotyczący populacji podtypu w odniesieniu do ich populacji klastrowej jest zdefiniowany jako (rozdz. 2.1.2, 4.5)

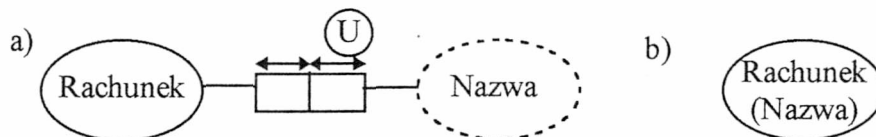
$$\begin{aligned} \text{jeśli } a \text{ Pol } b \wedge a, b \in \text{Klaster_x}(i, x), \text{ to } p \in \text{PopKlaster}(x, b) \cup \text{Pop}(a) \\ \vdash p \in \text{PopKlaster}(x, a). \end{aligned} \quad \{*\}(4.32)$$

4.2. Algorytm podziału na klastry

Metoda podziału na klastry umożliwia konstruowanie ich na dowolnym etapie tworzenia schematu konceptualnego.

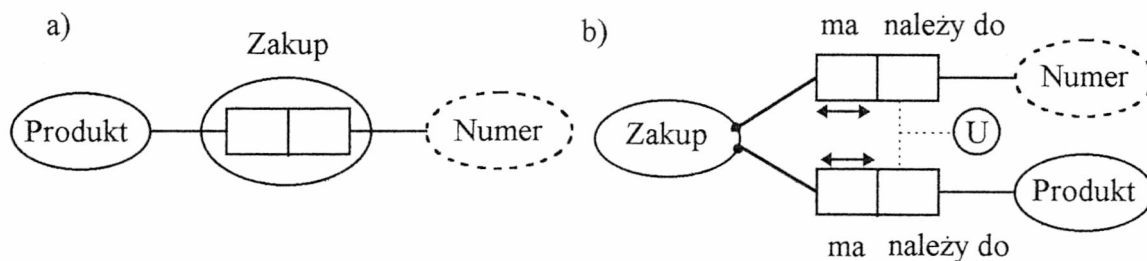
Zakłada się, że:

- dane schematu są sklasyfikowane na różne typy i ograniczenia.
- nazwy ról oznaczone kropką na diagramie, są rolami o większej wadze konceptualnej w danym typie faktu,
- schemat konceptualny jest identyfikowalny strukturalnie,
- typy atrybutowe są przydzielone do typów abstrakcyjnych wg trzeciej zasady normalizacji relacji (teoria relacyjnych baz danych),
- wszystkie typy abstrakcyjne, które posiadają zbiór identyfikacyjny składający się tylko z jednej roli należącej do relacji typu *Most* (2.13) (rys. 4.1 a), relacja ta staje się relacją domyślną, a typ atrybutowy jest na diagramie ORM dołączony do oznaczenia odpowiadającego typu abstrakcyjnego (rys. 4.1 b),



Rys. 4.1. Uproszczenia diagramu ORM ze względu na prostą ścieżkę identyfikacji typu abstrakcyjnego *Zakup*

- wszystkie typy strukturalnie zagnieżdżone (rys. 4.2 a) są przedstawione w postaci uproszczonej, podobnie jak w pracy [43]. Takie odwzorowanie umożliwia zaznaczenie zbioru identyfikującego, który wynika automatycznie z postaci zagnieżdżenia (rys. 4.2 b),

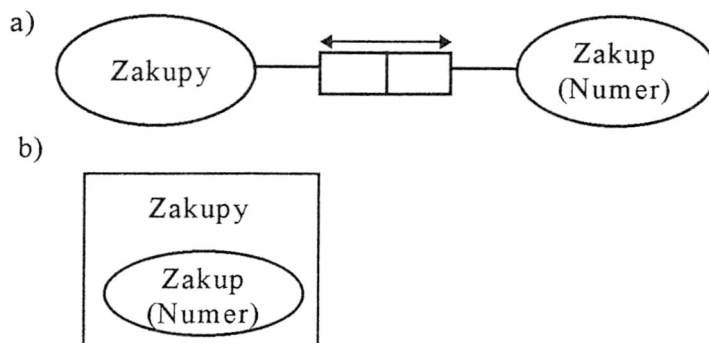


Rys. 4.2. Uproszczenia diagramu ORM typów zagnieżdżonych: postać zagnieżdżona (a) oraz postać uproszczona (b)

- wszystkie zależności między typami obiektowymi oznaczające związki $m:n$ lub $1:n$ mogą być zastąpione odpowiednimi typami złożonymi (typami sekwencyjnymi lub zbiorowymi) i ograniczeniami częstotliwości.

Algorytm przedstawiony w tej pracy jest oparty na:

- rozbudowie modelu metodą zagnieżdżania nowych klastrów w istniejących klastrach,
- metodzie łączenia elementów istniejącego modelu w klastry.



Rys. 4.3. Przekształcanie związków $n : m$ z rys. 4.3 (a) do obiektowego złożonego typu sekwencyjnego na rys. 4.3 (b)

Metoda ta przedstawiona za pomocą opisu algorytmu, składającego się z 4 części. Każda część składa się z sekwencji kroków.

I. Część I. Wstępna normalizacja schematu.

1. Na początku ustala się $i = 0$ jako numer początkowy wyodrębnianych klastrów oraz $j = 0$ jako poziom ukrywania typów obiektowych w klastrze.
2. Każdy z typów abstrakcyjnych (z domyślnym kluczowym atrybutem), który jest typem bazowym kilku ról i żadna z tych ról nie jest objęta ograniczeniami ani nie jest rolą wiodącą, przydziela się niezależnie do każdej z tych ról. Oznacza to, że te typy stają się *Liśćmi*. Nie oznacza to powielania ich na schemacie, gdyż zakłada się, że są niezależnie powiązane w poszczególnych typach faktów z innymi typami obiektowymi.
3. Typy obiektowe strukturalnie zagnieżdżone doprowadza się do prostszej postaci nie zagnieżdżonej (rys.4.2). Typy obiektowe zagnieżdżone są powielane na schemacie. Oznacza to powielenie typu bez jego pozostałych ról. Na diagramie jest on oznaczony za pomocą znaku '*'. Typ agregujący te typy zagnieżdżone staje się klastrem o numerze i , a zagnieżdżone typy są jego elementami. Jeżeli są klastrami, wraz z zagnieżdżonymi klastrami zwiększają swoje numery i o „1”.
4. Każdy typ abstrakcyjny staje się typem wiodącym klastra o numerze i , a jego typy atrybutowe stają się elementami tego klastra.
5. Wszystkie typy, które należą do rodziny *Spec* lub *Pol* typów obiektowych stają się typami wiodącymi klastrów o takiej samej wartości numeru i .
6. Na koniec wstępnych przygotowań ukrywa się wszystkie typy obiektowe atrybutowe w wyodrębnionych klastrach nadając im wartość $j = 1$.
7. Zaznacza się na role wiodące zgodnie z punktem b) z rozdziału 4.1.2.- na diagramie ORM podkreśla się ich nazwę.

II. Druga część algorytmu. Rozwiązuje się problem klastrów typu *Liść*, dołączonych do

jednego typu faktu, rozważając następujące przypadki (punkty 1 i 2):

1. Jeżeli rola tego klastra nie jest objęta żadnym ograniczeniem, poszukuje się bazowego klastra *KLA* tego typu faktu, którego zbiór ról zawiera rolę wiodącą oraz rolę *total* lub/i *unikat*. Dalej należy wykonać krok 3.
2. Jeżeli role w typie faktu *Liścia* są objęte ograniczeniami, poszukuje się bazowego klastra *KLA* tego typu faktu - jeżeli ograniczeniami są: związek $n:\dots:m$ (jedna strzałka nad wszystkimi rolami tego typu faktu, zaznaczona na diagramie), związek $1:\dots:n$ (strzałka nad jedną z ról) lub związek $1:\dots:1$ (niezależne strzałki nad każdą z ról na diagramie) lub *total*, typem wiodącym *i-tego* klastra staje się ten klaster bazowy, który posiada rolę wiodącą. Mamy wtedy kolejne przypadki.
 - 2.1. Brak nadrzędnej roli *Liścia* i klastra *KLA* - jeżeli klaster *KLA* jest też *Liściem*, klastry należące do *Liścia* oraz klaster *KLA* są klastrami o takim samym numerze *i* są wyłączone z drugiej części algorytmu. Jeżeli klaster *KLA* nie ma innych liści lub są już zagnieżdżone w wyniku kroku 3 i jest on połączony tylko jednym typem faktu z pozostałą częścią schematu ORM, wtedy *Liściem* staje dotychczasowy *Liść* oraz klaster *KLA* - przejście do kroku 4.
 - 2.2. Nadrzędnym klastrem zostaje *KLA* - dalej należy wykonać krok 3.
 - 2.3. Nadrzędna rola *Liścia* - wtedy wszystkie klastry bazowe należące do typu faktu związane z rolą *Liścia* stają się kandydatami (na diagramie oznacza się je za pomocą znaku 'k'). Jeżeli *kandydaci* są *Liśćmi*, stają się klastrami zagnieżdżonymi *Liścia* wejściowego (klastra będącego typem bazowym roli należącej do rozpatrywanego typu faktu) i zwiększają swoje numery *i* o 1 na wszystkich poziomach zagnieżdżenia - dalej należy wykonać krok 4.
3. Dany *Liść* staje się elementem tego klastra *KLA*, jeżeli pozostałe klastry tego faktu są połączone tylko z tym *KLA* innym typem faktu lub też są *Liśćmi*. Jeżeli jednak te pozostałe klastry są połączone z innymi klastrami lub mają role wiodące w zbiorze swoich ról, są powielane w tym typie faktu. Wszystkie te klastry stają się również elementami *klastra*. *KLA* i wraz z klastrami zagnieżdżonymi zwiększają swoje numery *i* o 1. Jeżeli klaster *KLA* jest *kandydatem*, jest zagnieżdżony w swoim nadrzędnym klastrze wyznaczonym w kroku 2.3 - wszystkie jego klastry automatycznie zwiększają swoje numery *i* o „1” na wszystkich poziomach zagnieżdżeń. Jeżeli w wyniku tego kroku powstał nowy klaster typu *Liść*, zostaje dołączony do zbioru *Liści* rozpatrywanego w drugiej części algorytmu.
4. Jeżeli punkty od 1 do 3 dotyczą klastra typu *Liść*, który jest elementem drzewa *Spec* lub *Pol* (połączony tylko jedną z tych relacji), we wszystkich tych przypadkach dołącza się całe drzewo dziedziczenia. Wszystkie elementy drzewa mają ten sam numer *i* klastra. Jeżeli inne elementy tego drzewa są połączone z innymi klastrami, należy powielić całe drzewo.
5. Powtarzać kroki od 1- 4, aż wyczerpany zostanie zbiór klastrów typu *Liść*.

III. Trzecia część algorytmu. Zakłada się, że po drugiej części klastry zostały połączone z innym klastrami wieloma rolami. Wtedy rozpatruje się kolejne przypadki.

1. Jeżeli klaster nie zawiera ról objętych ograniczeniami i żadna z pozostałych ról należących do typów faktów, z którymi dany klaster jest powiązany, wtedy

wyłącza się ten klaster z III części algorytmu i przechodzi się do kroku 4, w przeciwnym wypadku należy przejść do kroku 2.

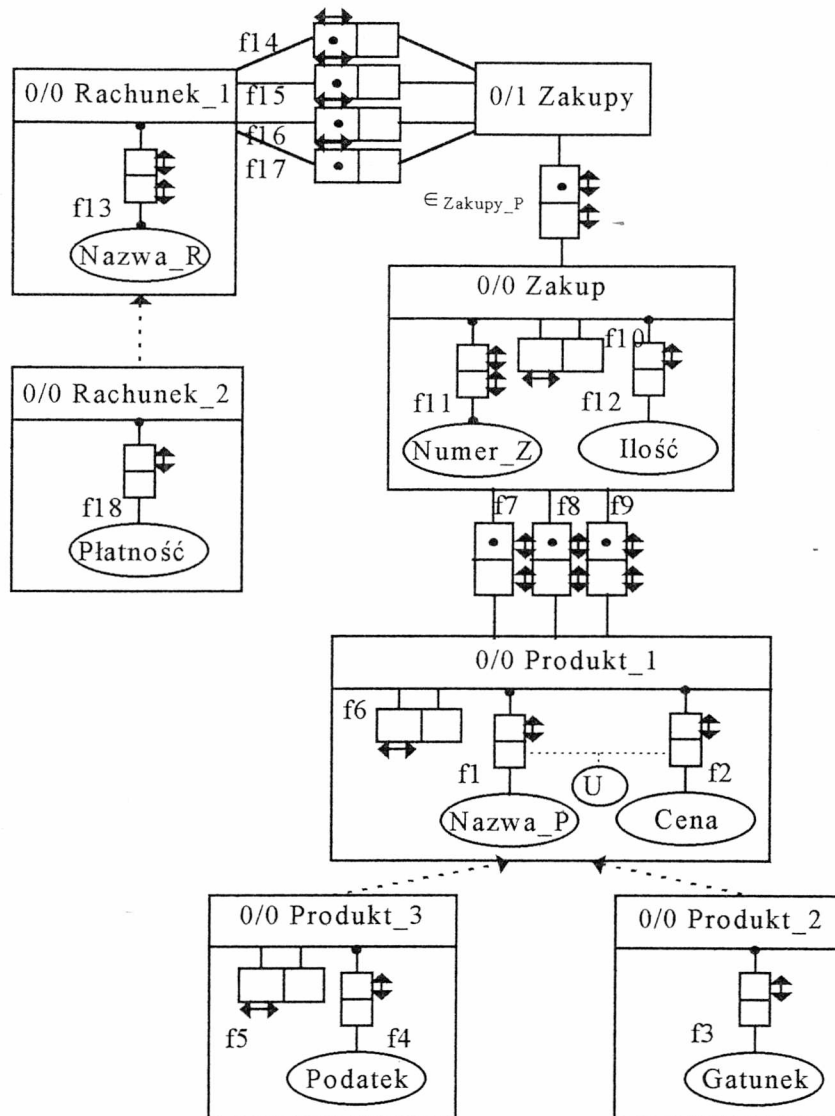
2. Jeżeli klaster jest połączony w związku $n:\dots:m$, $l:\dots:n$ lub $l:\dots:l$, wtedy rozpatruje się kolejne przypadki.
 - 2.1. Jeśli nie istnieje rola wiodąca, klastry posiadają ten sam numer i , - dalej należy przejść do kroku 3.
 - 2.2. Jeśli istnieje rola wiodąca, klaster nie zawierający roli wiodącej jest zagnieżdżany w klastrze bazowym roli należącej do rozpatrywanego typu faktu - dalej należy przejść do kroku 3.
3. W przypadku, gdy istnieją *Liście*, należy wykonać kroki drugiej części algorytmu.
4. Powtarzać kroki 1-3 aż do wyczerpania zbioru typów obiektowych.

IV. Czwarta część algorytmu. Po trzeciej części algorytmu zakłada się, że schemat konceptualny został sprowadzony do postaci szeregu zagnieżdżonych klastrów połączonych relacjami $l:\dots:l$, $n:\dots:m$ lub niezależnych. Taka postać schematu odpowiada głównemu poziomowi abstrakcji modelu.

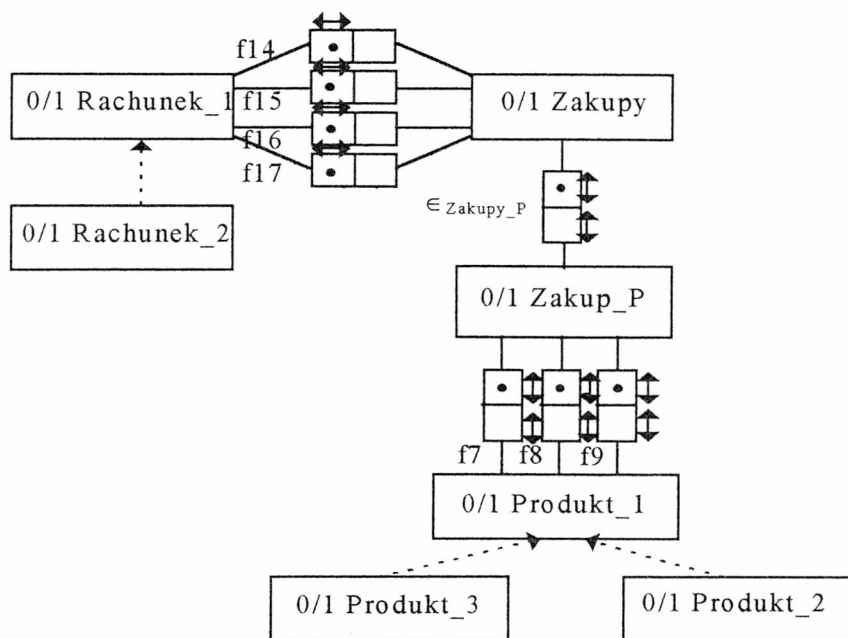
1. Ukrywa strukturę klastrów, odpowiednio zaczynając od klastrów z największym numerem i . Proces ukrywania jest następujący: wyszukuje się klastry zagnieżdżone najgłębiej i zwiększa ich numer j o „1”. Należy to wykonać na każdym z klastrów o takim samym numerze i , zawartych w tym samym klastrze o numerze $i-1$. Wówczas wybiera się klaster o największym numerze ukrycia j i dodaje do numeru ukrycia j klastra nadrzędnego o numerze $i-1$. Czynność tę należy powtarzać, aż ukryte zostaną wszystkie klastry o numerze $i=1$.
2. Wszystkie typy bazowe typów faktów oznaczających związek $m:n$ oraz $l:n$ powinny zostać przekształcone do typów sekwencyjnych lub zbiorowych (rys. 4.3).

4.3. Przykłady podziału na klastry modeli ORM

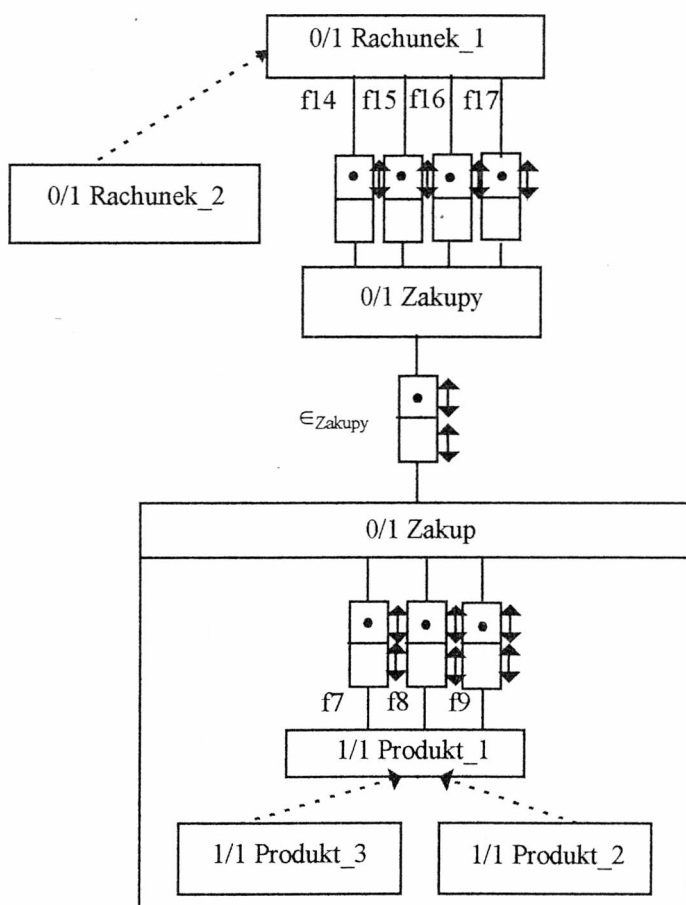
a) model M_I (rozdział 2.3, rys. 2.15):



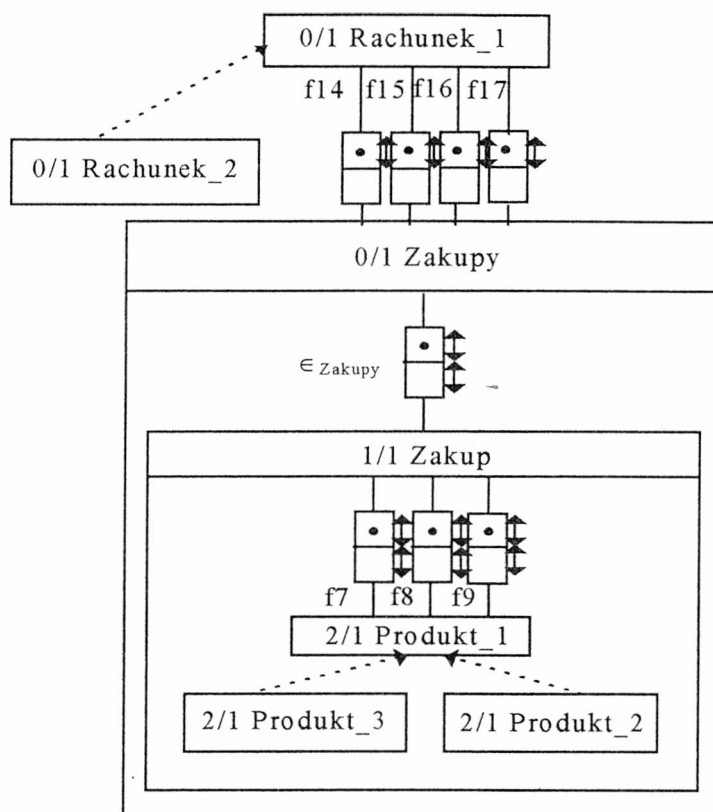
Rys. 4.4. Część I, krok 4



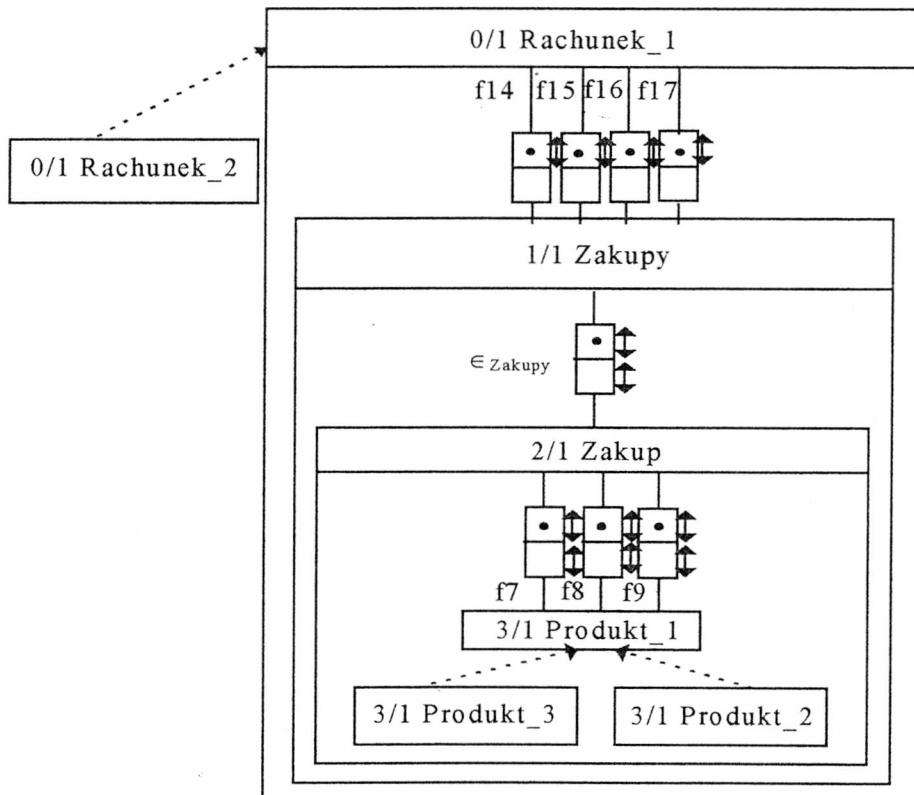
Rys. 4.5. Część 1, krok 6



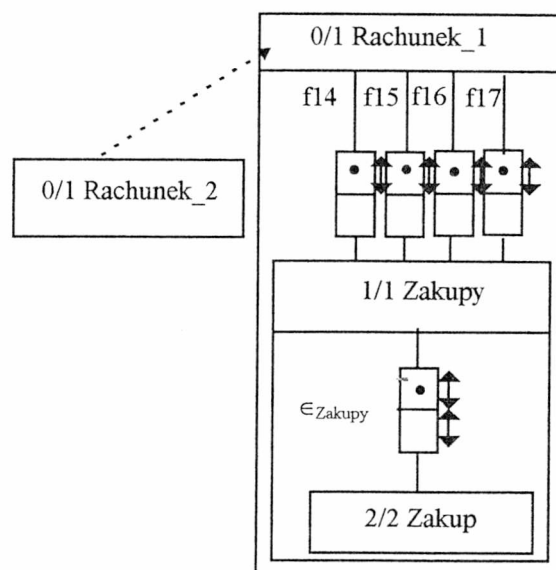
Rys. 4.6. Część 3 - zagnieżdżanie klastrów (1-y poziom syntezy klastrów)



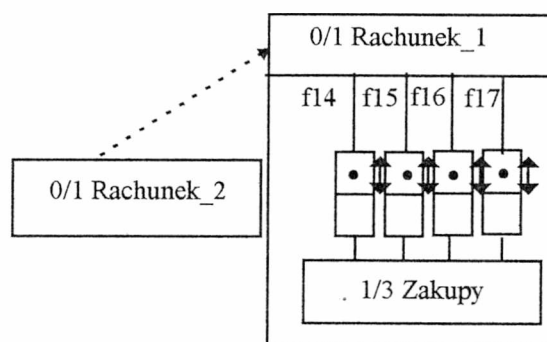
Rys. 4.7. Część 2 - 2-i poziom syntezy klastrow



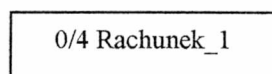
Rys. 4.8. Część 3- 3-i poziom syntezy klastrow - koniec syntezy



Rys. 4.9. Część 3 - ukrywanie struktury na 2-im poziomie syntezy klastrow

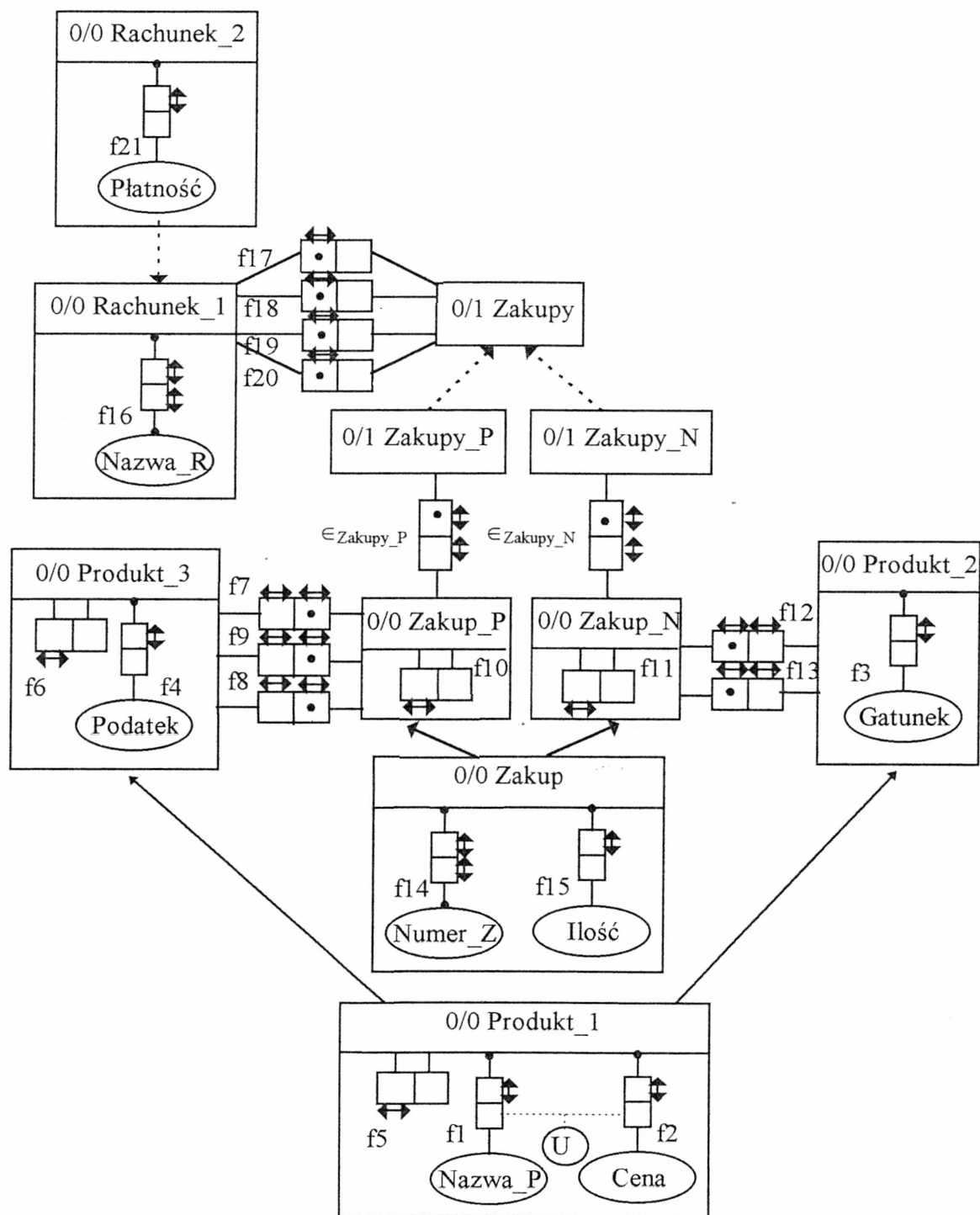


Rys. 4.10. Część 4 - ukrywanie struktury klastrow na 1-ym poziomie syntezy klastrow

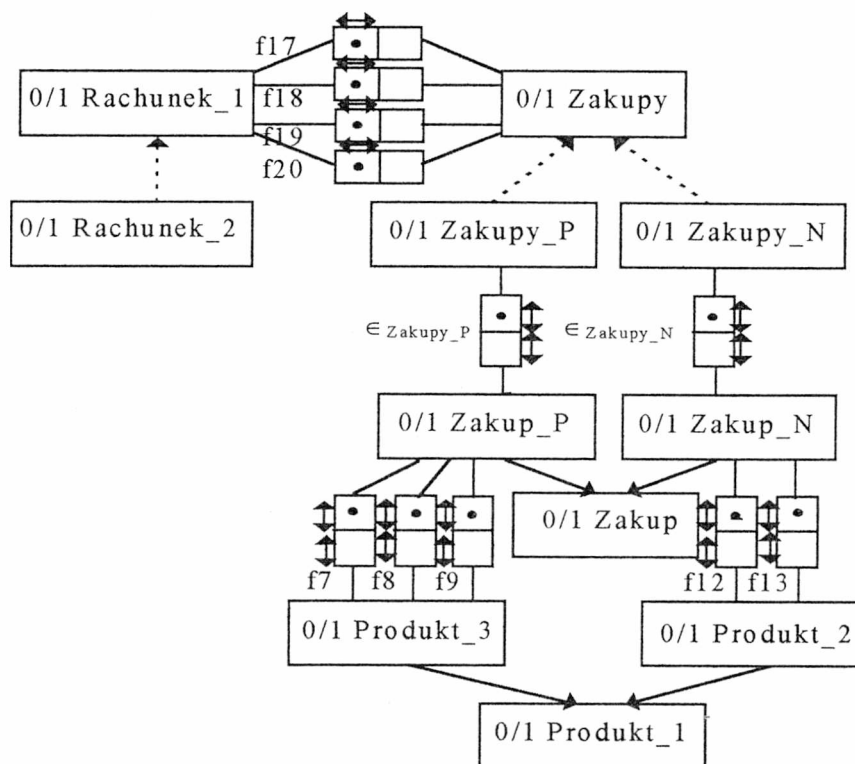


Rys. 4.11. Część 4 - ukrywanie struktury klastrow na 0 - ym poziomie syntezy klastrow

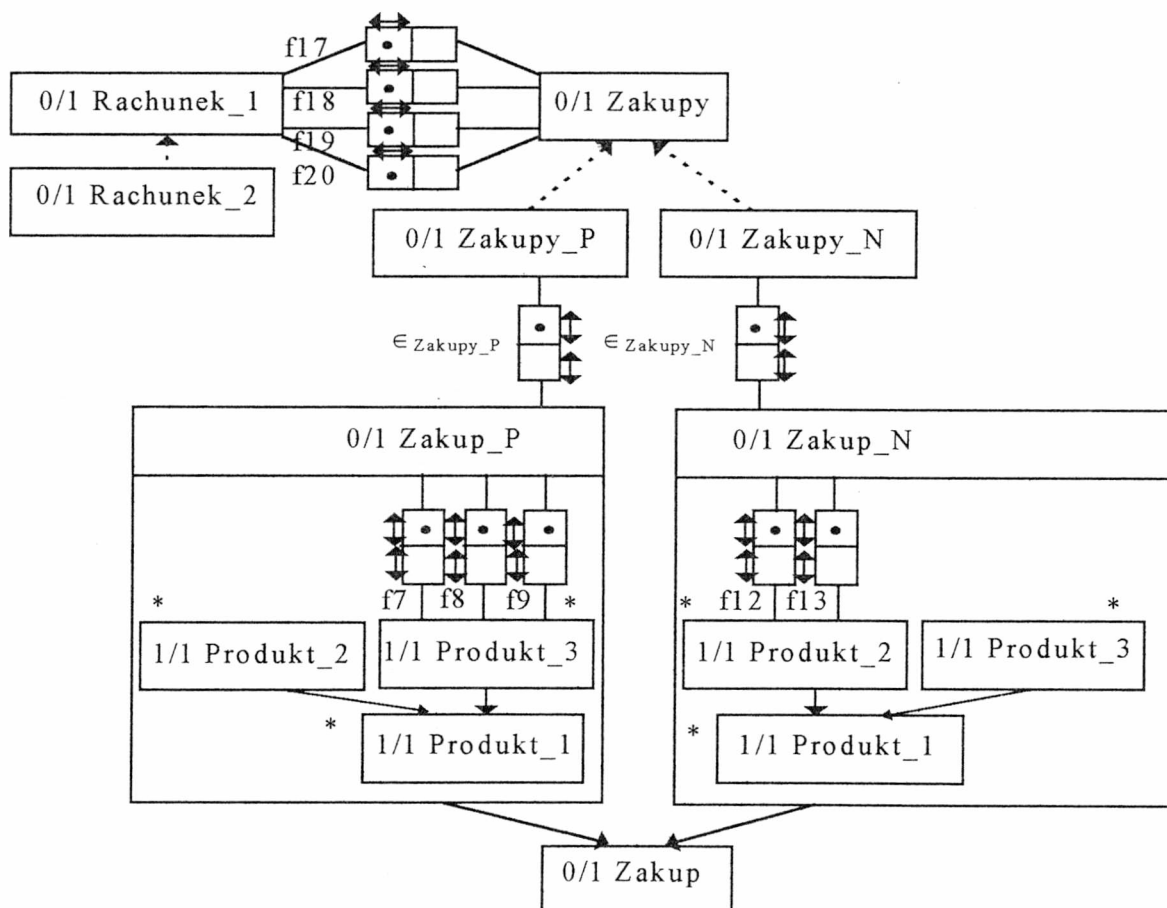
b) Model M_2 (rozdział 2.3, rys. 2.16):



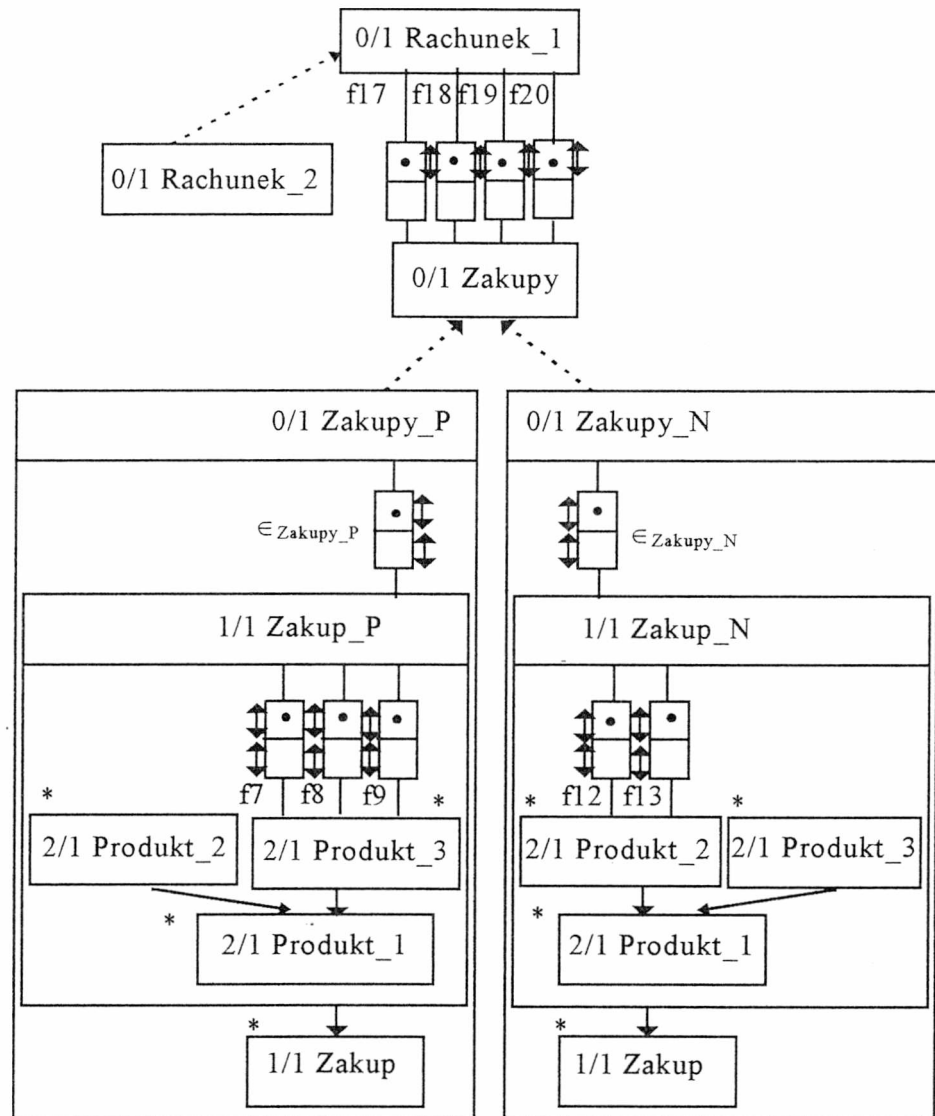
Rys. 4.12. Część I - krok 4



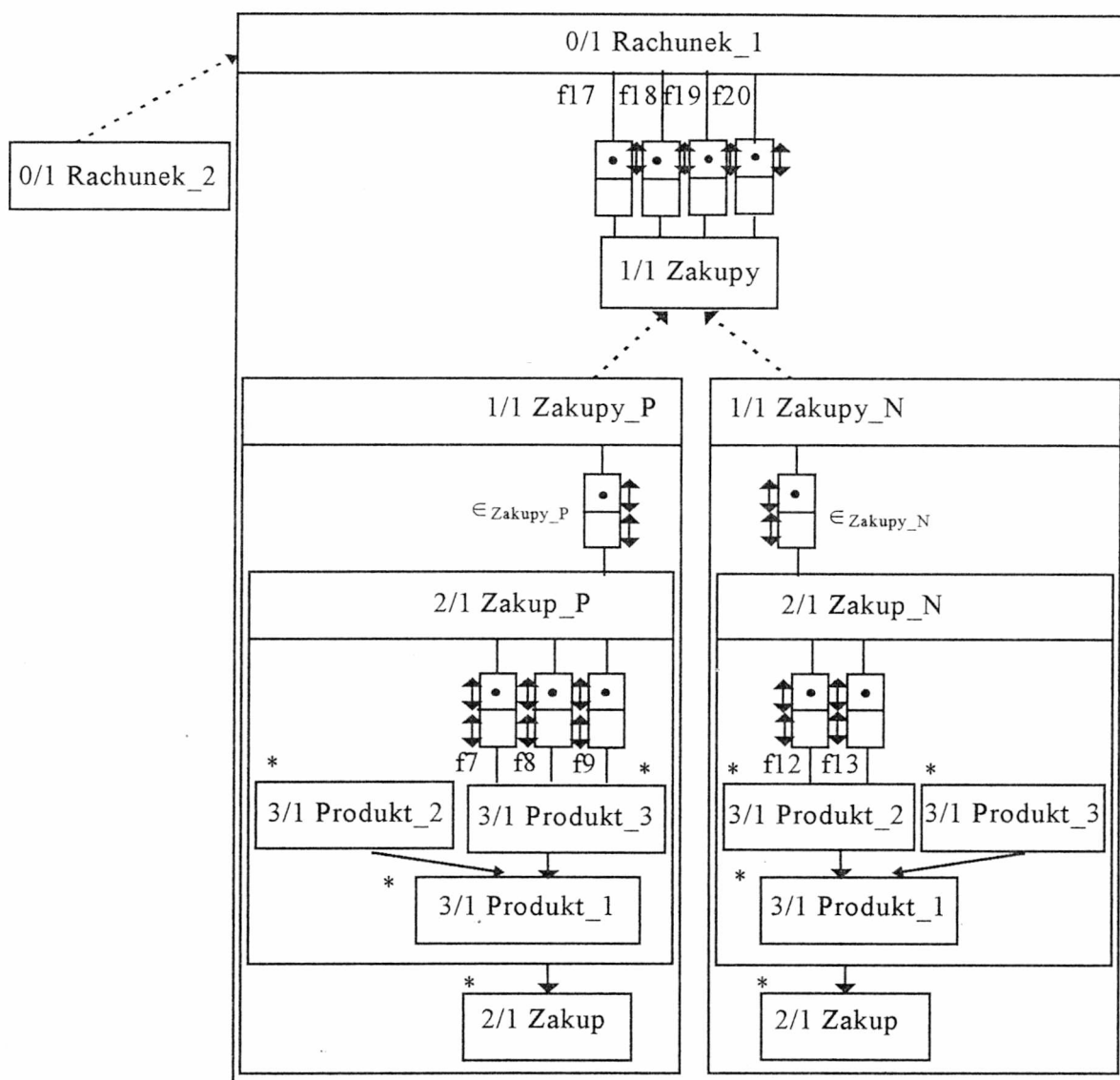
Rys. 4.13. Część 1 - krok 6



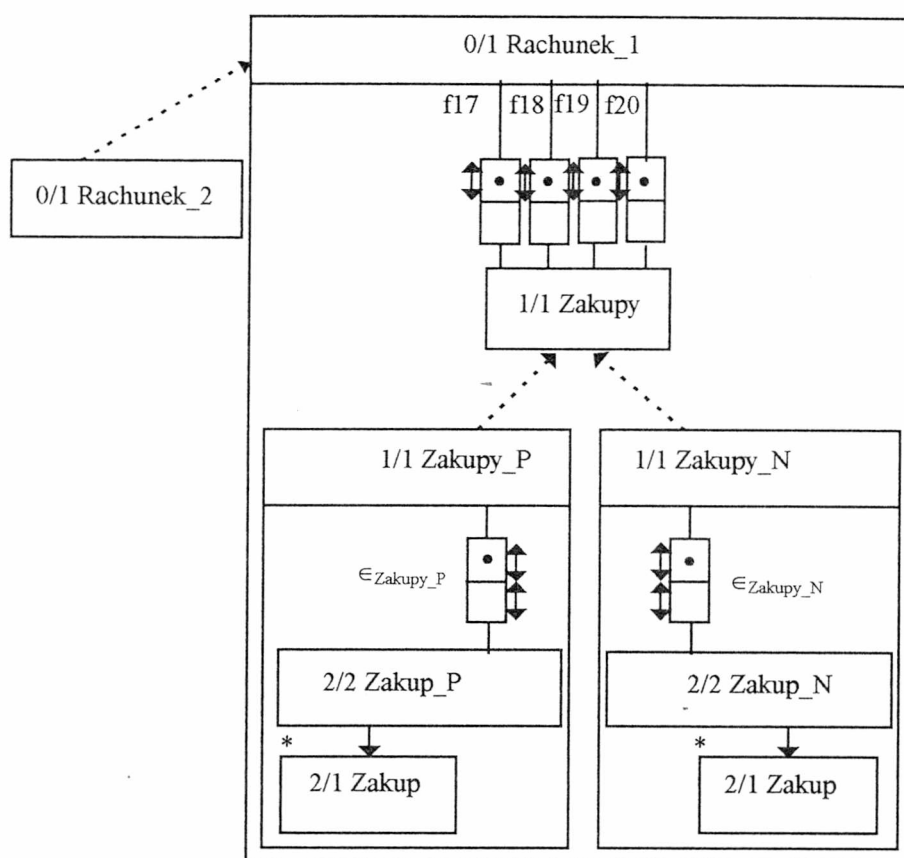
Rys. 4.14. Część 3 - 1-y poziom syntezy klastrów



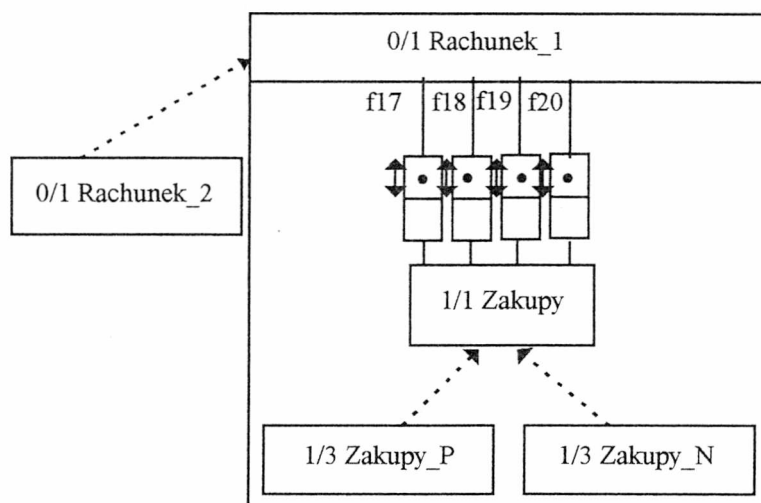
Rys. 4.15. Część 2 - 2-i poziom syntezy klastrów



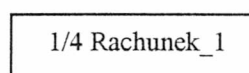
Rys. 4.16. Część 3 - 3-i poziom syntezy klastrow- koniec syntezy



Rys. 4.17. Część 4 - ukrywanie struktury na 2-ym poziomie syntezy klastrow



Rys. 4.18. Część 4 - ukrywanie struktury na 1-m poziomie syntezy klastrow



Rys. 4.19. Część 4 - ukrywanie struktury na 0-ym poziomie syntezy klastrow

5. Metody oceny równoważnych schematów konceptualnych

Celem badań jest opracowanie metody stosowania efektywnych konstrukcji do budowy modeli ORM oprogramowania zorientowanego obiektowo. W pracach [1, 6, 7, 11, 14, 22, 23, 24, 25, 26, 27, 29, 30, 31, 37, 46, 49, 55, 57, 68, 69, 77, 82, 84, 93, 112, 119, 132, 133] przedstawiono szereg problemów związanych z pomiarami efektywności oprogramowania. Metody określania jakości oprogramowania są ściśle związane z paradygmatami jego powstawania. Są one następujące:

- abstrakcja modelowania,
- iteracyjny charakter powstawania oprogramowania na drodze modelowania, projektowania i programowania,
- wieloużywalność elementów stosowanych we wszystkich fazach powstawania oprogramowania,
- integracja pojęciowa elementów stosowanych do budowy oprogramowania we wszystkich fazach jego powstawania:
 - * klasy stanowiące połączenie danych z operacjami / metodami,
 - * abstrakcja danych i operacji / metod,
 - * dziedziczenie i polimorfizm,
 - * hermetyzacja,
 - * przesyłanie komunikatów między wystąpieniami klas czyli obiektami.

Dzięki iteracyjnemu charakterowi tworzenia oprogramowania powstają jego części w różnych fazach powstawania. Te elementy można optymalizować zgodnie z ich właściwościami charakterystycznymi w danej fazie życia. Istnieją opracowane metody pomiaru efektywności w każdej z tych faz, szczególnie dotyczące fazy programowania. Stan prac w tej dziedzinie został scharakteryzowany w dalszej części rozdziału.

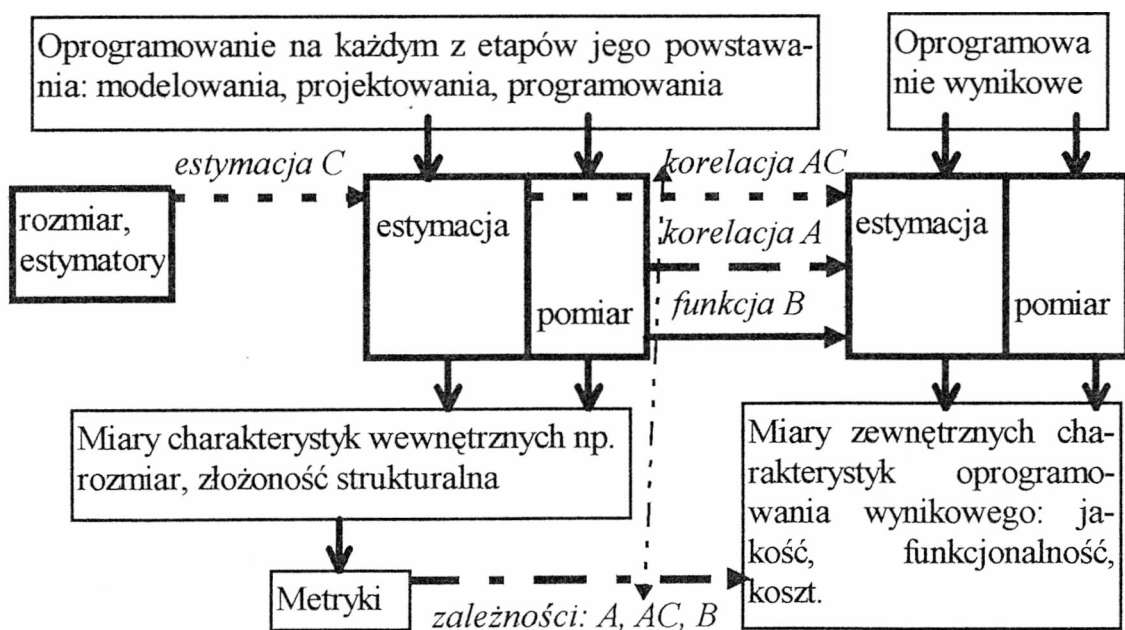
Efektywność oprogramowania wynika z właściwości strukturalnych i procesowych. Efektywność wynikająca ze struktury oprogramowania jest wyznaczana na podstawie charakterystyk statycznych. Właściwości strukturalne można oceniać zgodnie z paradygmatem iteracyjnego tworzenia oprogramowania, czyli w każdej fazie jego powstawania. Właściwości procesowe oprogramowania można oceniać na każdym z etapów powstawania wtedy, gdy produkty (model, projekt lub program) są kompletne. Gotowy produkt każdego z tych etapów można dopiero poddać pomiarom czasowym i optymalizacji. Badanie wydajności procesu realizowanego przez oprogramowanie nie jest rozważane w tej pracy.

W niniejszej pracy zakłada się, że w pierwszej kolejności należy dążyć do uzyskania efektywnych modeli struktur obiektów, a następnie należy badać wydajność modelu własności dynamicznych obiektów. Stąd w rozprawie skupiono się nad problematyką uzyskania efektywnych modeli struktur danych oprogramowania obiektowego. W rozdziałach 5.2 i 5.3 opracowano metodę badania charakterystyk wewnętrznych oprogramowania. Dzięki temu można zbadać wpływ polimorficznych struktur danych na uzyskanie efektywnych modeli obiektowych.

5.1. Podstawy oceny oprogramowania we wszystkich fazach jego powstawania

W niniejszej pracy badano się wpływ konstrukcji modelu na jego charakterystyki wewnętrzne. Modelowanie jest etapem nadrzędnym w tworzeniu oprogramowania za pomocą techniki obiektowej. Pierwszym powodem tej cechy jest spełnienie postulatu abstrakcji, co oznacza że modelowanie konceptualne ORM jest niezależne od implementacji realizowanej podczas projektowania i programowania. Druga przyczyna, określająca nadrzędną rolę modelowania, wynika z tego, że model struktury klas stanowi szkielet klas uzupełniany pewnymi szczegółami na poziomie implementacji. Stąd złożoność tego modelu wpływa na złożoność przyszłego programu. *Można stwierdzić na tej podstawie, że warunkiem koniecznym do uzyskania efektywnego programu wykonywalnego jest zastosowanie efektywnego modelu tego oprogramowania.* Stąd wynika celowość prowadzonych badań nad syntezą efektywnych modeli oprogramowania obiektowego.

Ocena efektywności oprogramowania jest wyznaczana na podstawie zewnętrznych charakterystyk, których miarą mogą być: jakość, funkcjonalność oraz koszt (rys. 5.1 na podstawie [57]). Powyższe pojęcia zdefiniowano w dalszej części pracy. Charakterystyki te wyznacza się za pomocą metod statystycznych, gdyż często niemożliwe jest wykonanie obiektywnych, bezpośrednich pomiarów [6, 11, 25, 27, 29, 57, 77]. Innym sposobem, stosowanym w praktyce jest wyznaczenie charakterystyk zewnętrznych za pomocą charakterystyk wewnętrznych.



Rys. 5.1. Model metryk obiektowych

Mogą one być powiązane [6, 7, 14, 22, 26, 28, 30, 37, 57, 77, 84] poprzez zależności funkcyjne B, korelację A lub AC z charakterystykami wewnętrznymi oprogramowania. Charakterystyki wewnętrzne znacznie łatwiej wyznaczyć poprzez pomiar konkretnych parametrów lub przez estymację C. W przypadku wyznaczenia charakterystyk wewnętrznych za pośrednictwem obiektywnych pomiarów, korzysta się z powiązań funk-

cyjnych B lub korelacji A z charakterystykami zewnętrznymi. Natomiast przy stosowaniu estymacji C przy wyznaczaniu charakterystyk wewnętrznych, charakterystyki zewnętrzne określa się za pomocą techniki korelacji AC . Zależności te mają charakter złożony tzn. dana charakterystyka wewnętrzna może określać kilka charakterystyk zewnętrznych za pomocą różnych zależności z A , B lub AC . Charakterystyki wewnętrzne wyrażają złożoność strukturalną oprogramowania (rys. 5.2 na podstawie [57]).

Charakterystyki wewnętrzne oprogramowania określają złożoność wewnętrzną oprogramowania. Obiektywne wyznaczanie miar tej złożoności wewnętrznej odbywa się za pośrednictwem *metryk* lub *miar* powiązanych funkcyjnie z kilkoma metrykami.

Metryka zdefiniowana jest jako zależność wiążącą pewne *wewnętrzne* parametry oprogramowania (liczba połączeń między klasami, liczba metod...), wyznaczone na podstawie obiektywnych pomiarów i umożliwia określanie *zewnętrznych* charakterystyk. Metryki są programowym odpowiednikiem np. długości, ciężaru w innych systemach.

Rodzaj metryk jest uzależniony od etapu powstawania oprogramowania, od rodzaju elementów oprogramowania oraz od wybranej charakterystyki zewnętrznej. W dalszej części zostanie przedstawiona na podstawie [57] idea wielowymiarowej przestrzeni pomiarowej badania efektywności oprogramowania.

Złożoność zewnętrzna				
obliczeniowa		psychologiczna		reprezentacji
złożoność wewnętrzna				
nieobiektywna		obiektywna		
problemu	programisty	strukturalna		
wpływ pośrednio na złożoność strukturalną, lecz brak obiektywnych metryk	metryki modułowe		metryki międzymodułowe	
	metryki złożoności proceduralnej		metryki złożoności semantycznej	metryki połączeń
	rozmiar			
	struktur danych			
	przepływ sterowania			
	spójność klas			

Rys 5.2. Klasyfikacja złożoności oprogramowania i odpowiadających jej metryk w przypadku istnienia obiektywnych możliwości ich wyznaczenia

- Należy formalnie zdefiniować elementy środowiska pomiarowego metryk, m. in.:
- system relacji:
 $A = (A, R_j, o_j)$, gdzie A oznacza niepusty zbiór obiektów rzeczywistych z dziedziny oprogramowania, R_j są empirycznymi relacjami (np. równy, większy), o_j oznacza binarne operacje w zbiorze A ;

$B = (B, S_j, o_j)$, gdzie B jest niepustym zbiorem formalnych obiektów (np. liczby, wektory) w dziedzinie pomiarów, S_j są relacjami (np. równy, większy) w zbiorze B , o_j oznacza binarne operacje w zbiorze B ;

- odwzorowanie homomorficzne: $\mu: A \rightarrow B$, stąd mamy $\forall a \in A \exists b \in B \mid \mu(a) = b$;
- typy skal jako (A, B, μ) : nominalna, porządkowa, przedziałowa, absolutna, względna.

Innym fundamentem oceny efektywności oprogramowania jest zbiór następujących aksjomatów, dzięki którym wyznacza się metryki złożoności oprogramowania (rozdziały 5.2 i 5.3):

1. Przy porównywaniu dwóch modeli pomija się fragmenty o tej samej złożoności,
2. Istnieje skończona liczba modeli o tej samej złożoności.
3. Istnieją różne modele P i Q , takie że mają identyczną złożoność
4. Istnieją modele $P \equiv Q$ spełniające te same funkcje, lecz o różnej złożoności.
5. Złożoność modeli połączonych jest większa od złożoności każdego z osobna:
6. Dodanie do pewnych modeli o identycznej złożoności pewnego modelu R może spowodować różnice w złożoności uzyskanych modeli.
7. Jeżeli model Q powstał przez permutację porządku elementów modelu P , to może być, że mają różną złożoność.
8. Złożoność modelu w wyniku połączenia dowolnych modeli P i Q jest większa niż suma ich złożoności.

a) Ocena oprogramowania na etapie modelu

Ocena efektywności oprogramowania i dobór metryk są ściśle związane z jego budową wewnętrzną zorientowaną obiektowo oraz sposobem tworzenia:

Ze względu na nadrzędną rolę modelowania w powstawaniu oprogramowania, w pracy badane jest oprogramowanie na poziomie struktury informacyjnej modelu statycznego, zakładając że efektywna struktura informacyjna może być podstawą do wyznaczania modelu dynamicznego (np. diagramy stanów). Opracowano szereg zasad (wg [57]), umożliwiających uzyskanie efektywnej struktury informacyjnej modelu, który reprezentuje rozwiązanie problemu przy minimalnej liczbie połączeń między klasami i maksymalnej spójności klas (połączenia atrybutów i metod w klasie):

- klasa może być uzależniona od kilku innych klas,
- połączenie między klasami powinno służyć jedynie do przesłania informacji,
- każda podklasa (następca) posiada podzbiór pewnych cech nadklasy (poprzednika) uzyskanych na drodze dziedziczenia,
- *Korzeń* klas może być abstrakcyjnym modelem następców dzięki polimorfizmowi,
- zbiór klas wielokrotnego użytku powinien maksymalnie wykorzystywać dziedziczenie do modelowania zależności w dziedzinie problemu,
- liczba powiązań, które reprezentują zadanie klasy, powinna być ograniczona.

Cechy oprogramowania obiektowego tworzą warunki do wyodrębnienia wielowymiarowych metryk do oceny efektywności oprogramowania obiektowego [46,

57, 84, 133].

1. Charakterystyki zewnętrzne oprogramowania wynikowego:
 - 1.1. Jakość:
 - a) stopień osiągniętej abstrakcji,
 - b) wieloużywalność,
 - c) zrozumiałość,
 - d) konserwacja,
 - e) testowalność.
 - 1.2. Koszt produkcji.
 - 1.3. Funkcjonalność.
2. Etap tworzenia oprogramowania: rekursyjno-iteracyjny charakter powstawania oprogramowania:
 - 2.1. Modelowanie (M).
 - 2.2. Projektowanie (D).
 - 2.3. Programowanie (P).
3. Elementy oprogramowania:
 - 3.1. Metody.
 - 3.2. Usługi.
 - 3.3. Klasy.
 - 3.4. Podsystemy.
 - 3.5. Systemy.
4. Metryki (przykład kilku metryk używanych na etapie modelowania) :
 - 4.1. Połączenia między klasami.
 - 4.2. Głębokość drzewa dziedziczenia.
 - 4.3. Miara spójności.
 - 4.4. Wskaźnik wieloużywalności.

Charakterystyki zewnętrzne oprogramowania obiektowego są określone przez jakość oprogramowania, funkcjonalność i koszt - te charakterystyki nie są powiązane funkcyjnie. *Jakość* oprogramowania jest określana przez: stopień abstrakcji, wieloużywalność, zrozumiałość, łatwość konserwacji oraz testowalności. Stopień abstrakcji, wyrażający możliwości łatwego rozwoju oprogramowania, określa się przez liczbę klas powiązanych relacją generalizacji z innymi klasami. Wieloużywalność jest cechą oznaczającą uniwersalność wyodrębnionych klas, podsystemów i systemów. Zrozumiałość, łatwość konserwacji oraz testowalności są wynikiem *wewnętrznej złożoności oprogramowania*, rozumianej jako: rozmiar (np. liczba klas), spójność klas, struktura danych, struktura logiczna, spójność semantyczna oraz połączenia między klasami.

Te powiązania między charakterystykami wewnętrznymi związanymi z etapem tworzenia oprogramowania i obszarem oprogramowania (klasa, podsystem, moduł...) a charakterystykami zewnętrznymi wyznaczyły przestrzeń badań nad uzyskaniem efektywnych programów - *wybrano etap modelowania, poziom systemu (cały model), poziom klas (model klastrowy) oraz metryki spójności modelu i połączeń*. W pracach [23, 24, 57] określono związek między wartością tych metryk a miarami charakterystyk zewnętrznych.

Niżej podano przykład metryki określającą wieloużywalność U mierzoną na dowolnym etapie tworzenia oprogramowania (M, D, P) i jej wpływ na ch-ki zewnętrzne.

W nawiasie podano numer wymiaru obszaru badań oprogramowania.

$U(4) \rightarrow \text{systemy}(3) \rightarrow M, D, P(2) \rightarrow \text{wieloużywalność}(1),$

$U(4) \rightarrow \text{systemy}(3), \text{podsystemy}(3) \rightarrow M, D, P(2) \rightarrow \text{zrozumiałość}(1),$

$U(4) \rightarrow \text{systemy}(3), \text{podsystemy}(3) \rightarrow M, D, P(2) \rightarrow \text{testowalność}(1).$

Przy doborze właściwych metryk, które umożliwiają ocenę charakterystyk zewnętrznych oprogramowania na wybranym etapie jego powstawania, należy:

- wybrać charakterystykę zewnętrzną,
- dobrać odpowiadającą jej charakterystykę wewnętrzną reprezentowaną przez taką metrykę, która najlepiej określa wybraną charakterystykę zewnętrzną (rys. 5.1).

b) Przykłady metryk

Poniżej podano pewien zestaw metryk, które umożliwiają właściwą ocenę jakości i funkcjonalności oprogramowania [wg 57] na etapie budowy statycznego modelu obiektowego:

1. Złożoność połączeniowa: liczba połączeń między obiektami

$$RFC = |\dot{M}_i \cup \text{wszystkie } j \{R_{ij}\}|,$$

gdzie M_i jest zbiorem wszystkich operacji klasy, natomiast $\{R_{ij}\}$ jest zbiorem operacji innych klas wywoływanych przez operację M_i .

2. Spójność semantyczna: Klasy powinny znajdować swoje odwzorowanie w *UoD*, stąd sztucznie połączone wprawdzie ograniczają liczbę połączeń między klasami, lecz naruszają zasady modelowania dotyczące wielokrotnego wykorzystania klasy.
3. Metryki określające złożoność proceduralną,

- 3.1. Liczba atrybutów (rozmiar)

$$NOA = NIV + DAC, \quad (5.1)$$

gdzie NIV - atrybuty proste,

DAC - atrybuty typu klasy.

- 3.2. Suma złożoności metod w klasie (struktura logiczna i rozmiar)

$$WMC = \sum_{i=1}^n c_i, \quad (5.2)$$

gdzie c_i jest statyczną złożonością każdej z i - metod (złożoność cyklomatyczna). Jeżeli c_i jest równe 1, wtedy WMC jest równe liczbie metod n . WMC maleje przy wykorzystaniu polimorfizmu i dziedziczenia.

- 3.3 Spójność klasy (połączenia atrybutów i metod)

$$LCOM = \frac{(\frac{1}{a} \sum_{j=1}^a \mu(A_j)) - m}{1 - m}, \quad (5.3)$$

gdzie

$\{M_i\}$ ($i = 1, \dots, m$) - zbiór metod w klasie oraz $\{A_j\}$ - zbiór atrybutów ($j = 1, \dots, a$),

$\mu(A_j)$ - liczba metod związanych z danym atrybutem.

3.4. Stopień specjalizacji (rozmiar)- S oraz wielokrotne użycie klas (rozmiar)- U

$$U = \frac{\text{liczba NadTypów}}{\text{całkowita liczba klas}}, \quad (5.4)$$

$$S = \frac{\text{liczba PodTypów}}{\text{liczba NadTypów}}, \quad (5.5)$$

Wartość S i U bliska 1 określa liniowy model dziedziczenia.

3.5. Liczba *PodTypów* (rozmiar) w drzewie dziedziczenia, czyli *NOC*.

3.6. Głębokość dziedziczenia, czyli liczba poziomów w drzewie dziedziczenia odniesiona do liczby klas, określająca zakres dziedziczenia (rozmiar)

$$DIT = \frac{\sum \text{głębokość dziedziczenia}}{\text{liczba klas}}. \quad (5.6)$$

Powyższe metryki nadają się do wykorzystania na poziomie modelu po wykonaniu specyfikacji obiektowej modelu, która wymaga wykonania:

- podziału modelu ORM na klastry, czyli wzorce przyszłych klas,
- odwzorowanie elementów klastra na atrybuty i metody.

W niniejszej pracy opracowano metryki pozwalające oceniać efektywność modelu przed wykonaniem specyfikacji obiektowej. W niniejszej pracy założono, że metody wyznaczone z modelu statycznego uzupełnia się w trakcie modelowania zachowania obiektów, czyli tworzenia modelu dynamicznego aplikacji. W niniejszej pracy założono również, że typy faktów określają nawigację modelu ORM, czyli pozwalają na oszacowanie liczby metod obsługujących powiązania typów obiektowych. Dotyczy to dwóch metryk:

- metryki strukturalnej złożoności połączeniowej, wyznaczającej liczbę połączeń między typami obiektowymi,
- metryki strukturalnej złożoności proceduralnej, określającej spójność klas.

5.2. Podstawy formalne oceny spójności grafu funkcjonalnego schematu konceptualnego

Spójność grafu dwudzielnego (def. 3.3 - 3.5 w rozdz. 3.1) wyznaczana na podstawie metryki *LCOM* jest oparta na założeniu, że wartość 0 odpowiada przypadkowi maksymalnej spójności tego grafu, natomiast wartość 1 odpowiada równej liczbie wierzchołków w zbiorze $V1$ i $V2$ i takiej samej liczbie krawędzi łączących wierzchołki należące do tych zbiorów.

W pracy przyjęto inną metodę oceny spójności grafu, rozróżniającą przypadek jednej składowej spójności grafu i przypadek kilku składowych. Warunki graniczne zostały takie same, z tym że uwzględniono przypadek minimalnej spójności grafu czyli istnienie jednej składowej grafu przy minimalnej liczbie krawędzi. Metoda ta umożliwia porównanie spójności różnych grafów funkcjonalnych wykluczając przypadki, gdy wartość miary spójności grafu niespójnego jest równa wartości miary spójności grafu spójnego.

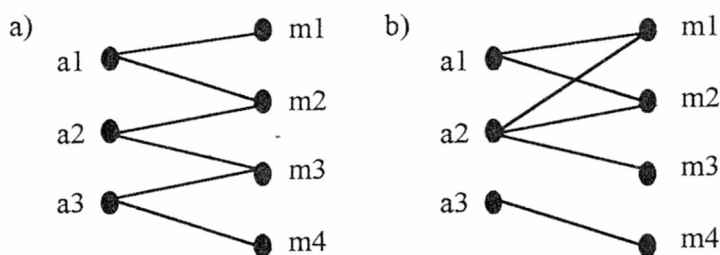
Sytuację taką przedstawiono na rys. 5.3. W przypadku zastosowania miary spójności $LCOM$ (5.3)

$$LCOM = \frac{(\frac{1}{a} \sum_{j=1}^a \mu(A_j)) - m}{1 - m}, \quad (5.7)$$

gdzie m jest liczbą wierzchołków $V1$, a jest liczbą wierzchołków $V2$, natomiast wyrażenie $\mu(A)$ liczbą krawędzi grafu, mamy identyczne wartości tej miary dla grafów z rys. 5.3 a) i b):

$$a) \quad LCOM = \frac{\frac{1}{3} * 6 - 4}{1 - 4} = 0.667,$$

$$b) \quad LCOM = \frac{\frac{1}{3} * 6 - 4}{1 - 4} = 0.667.$$



Rys. 5.3. Grafy dwudzielne o różnej spójności: graf spójny (a), graf niespójny (b)

W pracy przyjęto inne podejście, zakładające dwa przypadki:

- 1) istnienie jednej składowej,
- 2) istnienie wielu składowych spójności.

Do wyznaczenia liczby składowych grafu zastosowano algorytm odwiedzania jego wierzchołków. Każdy wierzchołek ze zbiorów $V1$ i $V2$ jest odwiedzany tylko raz. Liczba składowych spójności jest równa liczbie wierzchołków ze zbioru $V1$, które nie mają odwiedzonych w poprzednim kroku przyległych wierzchołków ze zbioru $V2$.

a) Algorytm wyznaczania składowych spójnych grafu dwudzielnego.

1. Odwiedzaj kolejno i -ty wierzchołek zbioru $V1$ grafu i zaznacz odwiedzenie wszystkich j -tych skojarzonych z nim wierzchołków ze zbioru $V2$ oraz odwiedzenie wszystkich p -tych wierzchołków ze zbioru $V1$ skojarzonych z każdym j -tym wierzchołkiem.
2. Jeżeli i -ty wierzchołek ze zbioru $V1$ jest skojarzony z przynajmniej jednym odwiedzionym wierzchołkiem ze zbioru $V2$ w poprzednim kroku algorytmu, oznacza to przynależność do tej samej składowej spójnej. Wówczas nie zwiększaj liczby składowych spójności.
3. W przeciwnym przypadku, jeżeli liczba odwiedzonych wierzchołków ze zbioru $V1$ jest większa od jeden i nie stwierdzono skojarzenia i -tego wierzchołka ze zbioru $V1$ odwiedzionym w poprzednim kroku z wierzchołkiem ze zbioru $V2$, wtedy należy zwiększyć liczbę składowych spójnych o jeden.

Projekt algorytmu wykorzystuje macierz T . Elementami macierzy T są krotki U

$$U = (\text{przyległość}, \text{odwiedzony}). \quad \{*\} (5.8)$$

Funkcja generująca wartości atrybutów (*przyległość*, *odwiedzony*) jest zdefiniowana następująco:

$f_{ij}(\text{przyległość}) = 1$, gdy i -ty typ obiektowy odgrywa rolę w j -tym typie faktu,
 $f_{ij}(\text{przyległość}) = 0$, gdy i -ty typ obiektowy nie odgrywa roli w j -tym typie faktu,
 $f_{ij}(\text{odwiedzony}) = 1$, oznacza że rola i -tego typu obiektowego w j -tym typie faktu była już rozpatrywana w czasie realizacji algorytmu,
 $f_{ij}(\text{odwiedzony}) = 0$, oznacza że rola i -tego typu obiektowego w j -tym typie faktu nie była jeszcze rozpatrywana w czasie realizacji algorytmu.

Daną wejściową algorytmu jest macierz T , natomiast daną wyjściową jest liczba składowych spójnych grafu.

Projekt programu realizującego algorytm wyznaczania liczby spójnych składowych grafu:

1. Inicjalizacja danych algorytmu:
 - 1.1. Wczytaj macierz T .
 - 1.2. Ustaw w każdym elemencie macierzy T pola *odwiedzony* na false.
 - 1.3. Ustaw liczbę składowych na 1, czyli $\text{Składowe} := 1$.
2. Wyznacz liczbę składowych spójnych grafu dwudzielnego za pomocą algorytmu przedstawionego w postaci procedury napisanej w języku podobnym do języka **Pascal**.

procedure Liczba_Składowych(**var** T ; **var** Składowe);

var Zwiększ : Boolean;

i, j, p : integer;

begin

Zwiększ:=false;

$i:=1$;

Składowe:=1;

while $i \leq n$ **do**

{przełączaj wszystkie wiersze macierzy T odpowiadające i -tym wierzchołkom ze zbioru $V1$ }

begin

$j:=1$;

while $j \leq m$ **do**

{przełączaj wszystkie wierzchołki ze zbioru $V2$ z j -tych kolumn skojarzone z i -tym wierzchołkiem ze zbioru $V1$ }

begin

if $T[i,j].\text{odwiedzony}$ **then** Zwiększ := true

else

```

if (T[i,j].przyległość=1) and (not T[i,j].odwiedzony) then
    {jeśli j-ty wierzchołek jest skojarzony i nie został odwiedzony w i-1
     kroku, zaznacz jego odwiedzenie czyli skojarzenie z wierzchołkiem
     i-tym }
    begin
        T[i,j].odwiedzony := true;
        p := i+1;
        while p <= n do
            {przełóż teraz wszystkie p-te wierzchołki ze zbioru V1, za-
             znacząc odwiedzenie wszystkich wierzchołków skojarzonych
             z wierzchołkiem j-tym ze zbioru V2}
            begin
                if T[p,j].przyległość = 1 then
                    T[p,j].odwiedzony := true;
                    p := p+1;
                end;
            end;
            j:=j+1;
        end;
        if (not Zwiększ) and (i > 1) then Składowe := Składowe+1;
        Zwiększ := false;
        i := i+1;
    end;
end.

```

Uwagi:

- 2.1. Złożoność algorytmu wyznaczania spójnych składowych jest proporcjonalna do iloczynu $n * m$. Osiągnięto taką złożoność dzięki odwiedzaniu każdego wierzchołka ze zbiorów V1 i V2 tylko raz.
- 2.2. Treść całego programu źródłowego wyznaczającego liczbę spójnych składowych zamieszczono w załączniku do pracy.

b) Metryki spójności grafu dwudzielnego

Na podstawie liczby składowych spójności wyznaczonych za pomocą algorytmu przedstawionego w kroku 2 należy wyznaczyć wartość jednej z metryk określającej spójność grafu w każdym z tych przypadków. W tym celu należy wybrać metrykę *LW1* dla grafu spójnego lub *LW2* dla grafu niespójnego z wyznaczoną liczbą składowych spójnych.

Obie metryki wyprowadzono w dalszej części tego rozdziału. Wyznaczenie tych metryk określających spójność grafu uzależnione jest od liczby spójnych składowych grafu dwudzielnego.

Przyjęto skalę względną, w której wartość „0” odpowiada wartości metryki dla grafów minimalnie spójnych, wartości dodatnie dla grafów spójnych oraz wartości

ujemne dla grafów niespójnych. Przyjęta dziedzina wartości pozwoli wyrazić znaczenie wartości metryki (przykład na rys. 5.3) dla przypadku grafu spójnego i niespójnego.

Pierwsza z metryk dotycząca grafów spójnych jest wyprowadzona z przeznaczeniem do wyznaczania spójności grafów funkcjonalnych modeli ORM. Jest to graf dwudzielny, gdzie wierzchołki $V1$ są typami faktów, wierzchołki $V2$ typami bazowymi tych faktów, natomiast krawędzie są rolami należącymi do tych faktów, dziedziczonych przez relację specjalizacji i polimorfizmu. Dodatkowo uwzględnia się połączenia wynikające z ograniczeń zbiorowych. Sposób pomiaru spójności za pomocą wyznaczonych metryk opisano w rozdziale 5.3.2.

W przypadku jednej składowej spójnej wyprowadzono wyrażenie LW stanowiące podstawę do wyznaczenia metryki LWI spójności grafu spójnego. Wyrażenie LW jest równe

$$LW = \frac{a - c}{b},$$

gdzie

- a - liczba krawędzi grafu dwudzielnego,
- c, b - współczynniki wyznaczane z warunków brzegowych narzuconych na krawędzie grafu maksymalnie spójnego i minimalnie spójnego,
- x - liczba wierzchołków ze zbioru $V1$,
- y - liczba wierzchołków ze zbioru $V2$,
- z - liczba krawędzi podwójnych.

Stąd mamy graniczne wartości metryki LW

$$\begin{cases} 0 = \frac{x + y + z - 1 - c}{b}, & \text{gdy } a = x + y + z - 1 - \text{dla grafu minimalnie spójnego,} \\ 1 = \frac{x * y + z - c}{b}, & \text{gdy } a = x * y + z - \text{dla grafu maksymalnie spójnego.} \end{cases}$$

Ponieważ $b \neq 0$, stąd wyznaczono wartość metryki LWI równą „0” w przypadku, gdy $x = 1$ przy wartości $y > 1$ lub $y = 1$ przy wartości $x > 1$ lub $x = 1$ i $y = 1$. Pierwszy z przypadków odpowiada wystąpieniu jednego n -krotnego typu faktu, natomiast drugi odpowiada przypadkowi wystąpienia jednego typu obiektowego i wielu typów faktów zawierających jedną rolę lub wystąpieniu rekurencyjnych typów faktów.

Stąd po przekształceniach mamy następującą metrykę LWI dla grafów spójnych

$$LWI = LW * \frac{x}{y} = \frac{x}{y} * \frac{a - x - y - z + 1}{x * y - x - y + 1} \quad \text{dla } x \neq 1 \text{ lub } y \neq 1$$

oraz

$$LWI = 0 \text{ dla } x = 1, y > 1 \text{ lub } y = 1, x > 1 \text{ lub } x = 1 \text{ i } y = 1. \quad \{*\}(5.9)$$

W przypadku liczby składowych spójnych większej od jeden, wyprowadzono wyrażenie stanowiące metrykę spójności grafu niespójnego. Metryka ta szczególną rolę odgrywa w wyznaczaniu wartości metryki spójności klastrowego modelu ORM. W modelu klastrowym spójność każdego klastra jest obliczana oddzielnie. Każda z ról jest prototypem metody realizującej nawigację między atrybutami wewnątrz klastra i stanowi wierzchołki ze zbioru $V1$. Wszystkie typy obiektowe przydzielone do klastra są wierzchołkami

należącymi do zbioru $V2$. Krawędzie grafu są połączeniami między typami obiektowymi ze zbioru $V2$ a rolami stanowiącymi wierzchołki ze zbioru $V1$. Połączenia te wynikają z liczby ról łączących każdy z wierzchołków $V2$ z typem wiodącym klastra oraz ograniczeń zbiorowych (rozdz. 5.3.2). W wyniku podziału na klastry można uzyskać graf funkcjonalny występujący najczęściej jako dwudzielny graf niespójny.

Metryka $LW2$ jest równa

$$LW2 = \frac{a - c - k}{b},$$

gdzie

k - liczba składowych spójnych.

Parametry: a, x, y, c, b, z mają takie samo znaczenie, jak przy wyprowadzaniu metryki $LW1$.

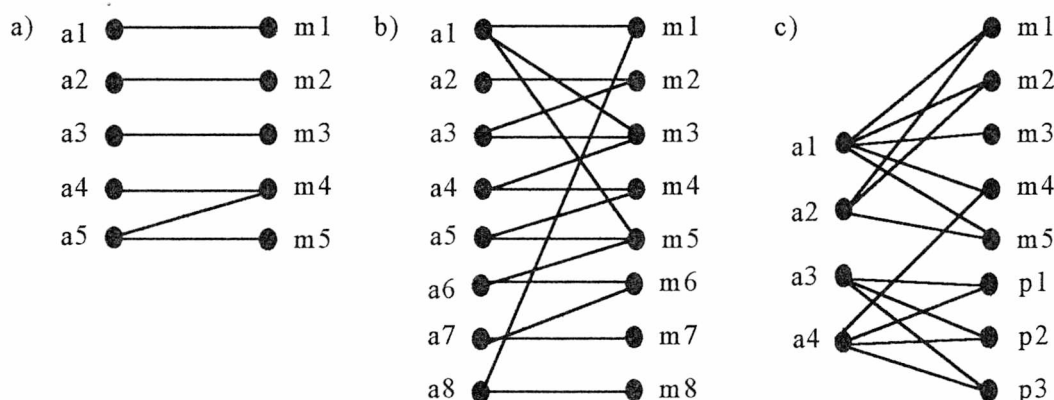
Stąd mamy graniczne wartości metryk grafu niespójnego

$$\begin{cases} 0 = \frac{x + y + z - 1 - c - 1}{b}, & \text{gdy } a = x + y + z - 1, k = 1 - \text{dla grafu min. Spójnego,} \\ -1 = \frac{y + z - c - k}{b}, & \text{gdy } a = y + z, k \neq 1 - \text{dla grafu niespójnego.} \end{cases}$$

Po przekształceniach otrzymujemy metrykę $LW2$ dla grafów niespójnych zawierających k składowych spójnych

$$LW2 = \frac{a - x - y - z + 2 - k}{x - 2 + k}. \quad \{*\} (5.10)$$

Na rys. 5.4 przedstawiono trzy przykłady grafów dwudzielnych. Następnie wyznaczono wartości metryk $LCOM$ (5.7), $LW1$ i $LW2$.



Rys. 5.4. Przykłady grafów dwudzielnych

Na tym przykładzie pokazano różnice tych metryk w rozróżnianiu liczby składowych spójności grafów dwudzielnych:

a) $k = 4, x = 5, y = 5, a = 6,$
 $LCOM = 0.95,$
 $LW2 = -0.857,$

b) $k = 1, x = 8, y = 8, a = 16,$
 $LCOM = 0.857,$
 $LW2 = 0.02,$

c) $k = 1, x = 3, y = 8, a = 15,$
 $LCOM = 0.292,$
 $LW1 = 0.095.$

W tabeli na rys. 5.5 przedstawiono miary $LCOM$ (5.7) oraz $LW1$ (5.9) i $LW2$ (5.10) dla różnych grafów dwudzielnych.

Rodzaj grafu	Rodzaj metryki	Liczba krawędzi										
		12	11	10	9	8	7	6	5	4	3	2
2/3	LW							0.67	0.33	0	-1	
	$LCOM$							0	0.33	0.67	1	
3/2	LW							1.5	0.75	0	-0.5	-1
	$LCOM$							0	0.25	0.5	0.75	1
3/3	LW				1	0.75	0.5	0.2	0	-0.5	-1	
	$LCOM$				0	0.17	0.33	0.5	0.67	0.83	1	
2/4	LW					0.5	0.33	0.17	0	-1		
	$LCOM$					0	0.25	0.5	0.75	1		
4/2	LW					2	1.33	0.67	0	-0.3	-0.7	-1
	$LCOM$					0	0.17	0.33	0.5	0.67	0.83	1
3/4	LW	0.75	0.62	0.5	0.38	0.25	0.13	0	-0.7	-1		
	$LCOM$	0	0.08	0.17	0.25	0.33	0.42	0.5	0.58	1		
4/3	LW	1.33	1.11	0.89	0.67	0.44	0.22	0	-0.3	-0.7	-1	
	$LCOM$	0	0.11	0.22	0.33	0.44	0.55	0.66	0.77	0.88	1	
2/5	LW			0.4	0.3	0.2	0.1	0	-1			
	$LCOM$			0	0.2	0.4	0.6	0.8	1			
5/2	LW			2.5	1.88	1.25	0.63	0	-0.3	-0.5	-0.8	-1
	$LCOM$			0	0.13	0.25	0.36	0.5	0.63	0.75	0.88	1

Rys. 5.5 Wartości metryk $LCOM$, $LW1$ i $LW2$ dla różnych grafów dwudzielnych

Oznaczenie grafu dwudzielnego x/y oznacza, że x jest liczbą wierzchołków ze zbioru $V1$, natomiast y jest liczbą wierzchołków ze zbioru $V2$.

c) Ocena spójności dwudzielnych grafów funkcjonalnych na podstawie metryk LW1 i LW2

Porównanie spójności grafów dwudzielnych odgrywa zasadniczą rolę w wyborze efektywnych grafów funkcjonalnych modeli ORM. Rozważono kilka przypadków związanych z metryką LW1(5.9) dla grafów spójnych oraz LW2 (5.10) dla grafów niespójnych. Ustalanie danych wejściowych wynikało z założenia, że elementy grafu dwudzielnego są odwzorowaniem grafu funkcjonalnego modelu ORM.

1. Metryka LW1:

- 1.1. Wpływ różnicy między liczbą x wierzchołków ze zbioru $V1$ a liczbą y wierzchołków ze zbioru $V2$.

Dane wejściowe:

$x = y + p$ - gdzie p jest różnicą liczby typów faktów i liczby typów bazowych ról należących do tych typów faktów,

$a_{min} = 2 * (y + p)$ - przy założeniu, że typy faktów zawierają jedynie dwie role,

$z = 0$ - brak rekurencyjnych typów faktów.

Obliczenia

$$LW1 = \frac{x}{y} * \frac{a - x - y + 1}{x * y - x - y + 1} = \frac{y + p}{y} * \frac{2 * y + 2 * p - y - p - y + 1}{(y + p) * y - y - p - y + 1} = \frac{p^2 + p * (y + 1) + y}{y * (y - 1) * (p + y - 1)}$$

Wartość wyrażenia rośnie przy wzroście wartości p czyli z szybszym wzrostem parametru x niż y , przy spełnieniu następujących warunków: $y \neq 0$, $y > 1$, $p \neq 1 - y$ czyli $x > 1$.

- 1.2. Wpływ różnicy między liczbą x wierzchołków ze zbioru $V1$ a liczbą y wierzchołków ze zbioru $V2$.

Dane wejściowe:

$x = y - p$ - gdzie p jest różnicą liczby typów faktów i liczby typów bazowych ról należących do tych typów faktów,

$a_{min} = y - p + y - 1 + m$ - przy założeniu, że graf jest spójny i istnieją n -krotne typy faktów (typów faktów składających się z wielu ról) lub istnieją role dziedziczone w wyniku specjalizacji i polimorfizmu (gdyż $y - p + y - 1 \geq 2 * (y - p)$ dla $p > 0$),

$z = 0$ - brak rekurencyjnych typów faktów.

Obliczenia

$$LW1 = \frac{x}{y} * \frac{a - x - y + 1}{x * y - x - y + 1} = \frac{y - p}{y} * \frac{y - p + y - 1 + m - y + p - y + 1}{(y - p) * y - y + p - y + 1} = \frac{(y - p) * m}{y * (y - 1) * (-p + y - 1)}$$

W przypadku braku n -krotnych typów faktów lub braku specjalizacji lub braku polimorfizmu wartość metryki byłaby równa „0” lub graf stałby się niespójny. Wartość wyrażenia powinna być dodatnia dla grafów spójnych, stąd przy wzroście wartości p rośnie wartość metryki aż do spełnienia warunku $p < y$ przy speł-

nieniu następujących warunków: $y \neq 0$, $y > 1$, $p \neq y-1$ czyli $p < y-1$ i $x > 1$.

1.3. Wpływ wartości parametru a na wartość wyrażenia $LW1$.

Wartość wyrażenia zawsze rośnie wraz ze wzrostem wartości parametru a , czyli liczbą krawędzi w grafie. Przy stałej liczbie typów faktów i typów bazowych wzrost wartości parametru a oznacza wpływ relacji *Spec* i *Pol* (rozdz. 2.1.2), ograniczeń zbiorowych lub n -krotnych typów faktów.

2. Metryka $LW2$:

2.1. Wpływ różnicy między liczbą x wierzchołków ze zbioru $V1$ a liczbą y wierzchołków ze zbioru $V2$, przy zachowaniu tej samej liczby składowych niespójności k .

Dane wejściowe:

$x = y + p$ - różnica między liczbą wierzchołków w grafie dwudzielnym,

$a_{min} = y + p$ - przy założeniu, że graf reprezentuje klaster modelu ORM z rozdziału 4 lub istnieją typy faktu zawierające jedną rolę,

$z = 0$ - brak rekurencyjnych typów faktów.

Obliczenia

$$LW2 = \frac{a - x - y + 2 - k}{x - 2 + k} = \frac{y + p - y - p - y + 2 - k}{y + p - 2 + k} = \frac{2 - y - k}{p + y - 2 + k}.$$

Wartość wyrażenia rośnie ($LW2 < 0$) przy wzroście wartości p czyli z szybszym wzrostem parametru x niż y przy spełnieniu następujących warunków: $y \geq 2$ i $k \neq p$ czyli $p \neq 2 - y - k$.

2.2. Wpływ wartości parametru a na wartość wyrażenia $LW2$.

Wartość wyrażenia zawsze rośnie wraz ze wzrostem wartości parametru a , czyli liczbą krawędzi w grafie aż do spełnienia warunku, że liczba składowych spójnych będzie większa od 1 ($k \geq 2$). Przy stałej liczbie typów faktów i typów bazowych wzrost wartości parametru a oznacza wpływ relacji *Spec* i *Pol*, ograniczeń zbiorowych lub n -krotnych typów faktów.

2.3. Wpływ liczby składowych spójnych - wzrost liczby składowych spójnych k zmniejsza spójność przy zachowaniu bez zmian pozostałych parametrów metryki $LW2$.

5.3. Metody pomiaru charakterystyk wewnętrznych modelu ORM uwzględniające wpływ polimorfizmu ze specjalizacją

5.3.1. Metryka połączeń między typami obiektowymi

Liczba połączeń między typami obiektowymi w modelu ORM wynika z liczby typów faktów, jakimi są one powiązane. Połączenia te można podzielić na dwie grupy:

- połączenia wyznaczone przez typy faktów, które są *Mostami*,
- połączenia między typami abstrakcyjnymi, w tym połączenia wynikające ze specjalizacji i polimorfizmu.

Podstawą do wyznaczenia tych połączeń jest graf funkcjonalny. W grafie funkcjonalnym zaznaczone są wszystkie połączenia z obu grup. Jednocześnie jest to graf całego modelu, stąd można oszacować całkowitą złożoność połączeniową.

Przyjęto, że liczba połączeń jest wyznaczana dla wszystkich typów abstrakcyjnych jako zbiór wszystkich ról wynikających z typów faktów, w których uczestniczy dany typ obiektowy. Zbiór ról *Role* każdego typu obiektowego x można wyznaczyć z zależności (2.27, 2.32, 2.37, 2.131)

$$Role(x) = WRole(x) \cup PolRole(x) \cup PolRoleS(x). \quad \{*\}(5.11)$$

Wzór ten jest prawdziwy dla każdego przypadku, gdyż poszczególne zbiory ról $PolRole(x)$ (2.32), $PolRoleS(x)$ (2.37) są zbiorami pustymi w przypadku braku polimorfizmu, natomiast zbiór $WRole$ (2.27) w przypadku braku specjalizacji traktuje typ obiektowy jako *Korzeń* (2.9) drzewa dziedziczenia i zbiór $WRole$ (2.27) zawiera role tego typu.

Funkcja *Połączenia*_{OB}, wyznaczająca liczbę połączeń dla każdego typu obiektowego jest zdefiniowana jako

$$Połączenia_OB: \varphi(TOB) \rightarrow \varphi(TRole),$$

czyli (5.11, 2.2)

$$Połączenia_OB(x) = \{y \mid \forall z \in Role(x) [y \in Fakt(z)]\}. \quad \{*\}(5.12)$$

Wypadkowy zbiór połączeń jest sumą indeksowanej rodziny zbiorów połączeń poszczególnych typów obiektowych, czyli

$$Połączenia_W(Połączenia_OB(x_i)_{i \in TA}) = \cup_{i \in TA} Połączenia_OB(x_i). \quad \{*\}(5.13)$$

Metryka wyznaczająca liczbę połączeń jest mocą zbioru $Połączenia_W$, czyli

$$Połączenia = |Połączenia_W|. \quad \{*\}(5.14)$$

Oznacza to, że uzyskamy liczbę wszystkich ról występujących w modelu. Metryka ta oznacza, że:

- obsługa każdego typu faktu wymaga przynajmniej takiej liczby wywołań, jaka wynika z ról łączących dany typ faktu z typami bazowymi,
- podana jest liczba połączeń całego modelu.

Zbiór połączeń całej aplikacji nie zmieni się po podziale na klastry (rozdz. 4). Wynika to z tego, że jedynie poszczególne fragmenty struktury informacyjnej zostaną przydzielone do klastrów bez dodawania nowych typów faktów. Można wtedy podać zbiór połączeń określonych dla każdego z klastrów i wyznaczyć sumę zbiorów tych połączeń.

5.3.2. Metryki spójności modelu

Spójność modelu ORM jest wyznaczona na podstawie jego grafu funkcjonalnego. Opracowano dwie metryki określające spójność modelu. Są one następujące:

- *Spójność*_M - metryka mierząca wartość spójności grafu funkcjonalnego modelu ORM na podstawie metryk *LW1* (5.9) i *LW2* (5.10),
- *Spójność*_K - metryka mierząca średnią wartość spójności *LW1* (5.9) i *LW2* (5.10)

wszystkich klastrow, wyznaczonych za pomocą algorytmu klastrowego z rozdz. 4.2.

Metryka $Spójność_M$ wyznacza wartość zależności wiążącej liczbę ról całego modelu z liczbą typów faktów i pozostałych typów obiektowych występujących w tym modelu. Jest to zależność

$$Spójność_M = \varphi(TFakt, TA \cup TWart, Role_M),$$

gdzie zbiór $Role_M$ jest wyznaczony jako suma zbiorów, czyli (2.32, 2.37, 2.27, 2.114, 2.115)

$$Role_M(z) = \bigcup_z PolRole(z) \cup \bigcup_z PolRoleS(z) \cup \bigcup_z WRole(z) \cup \bigcup_z RolePodzbiór \cup \bigcup_z RoleRówność. \quad \{*\}(5.15)$$

W zależności $LW1$ i $LW2$ parametr a jest więc równy mocy zbioru $Role_M(x)$

$$a = |Role_M(z)|. \quad \{*\}(5.16)$$

Parametr x jest równy liczbie typów faktów

$$x = |TFakt|. \quad \{*\}(5.17)$$

natomiast parametr y jest równy liczbie pozostałych typów obiektowych, czyli

$$y = |TA \cup TWart|. \quad \{*\}(5.18)$$

Metryka wyznaczająca spójność modelu ORM jest zdefiniowana jako

$$Spójność_M = \begin{cases} LW1, & \text{gdy graf funkcjonalny modelu jest spójny (5.9),} \\ LW2, & \text{gdy graf funkcjonalny modelu jest niespójny (5.10).} \end{cases} \quad \{*\}(5.19)$$

Metryka uwzględnia więc różne struktury modelu zmieniając dziedzinę wartości w zależności od liczby składowych spójności grafu funkcjonalnego.

Model klastrowy wymaga wyznaczenia średniej wartości spójności wszystkich klastrow. Stąd wyrażenie $Spójność_K$ stanowi średnią wartość sumy spójności poszczególnych klastrow, czyli

$$Spójność_K = \frac{\sum Spójność_kl}{n}. \quad \{*\}(5.20)$$

Spójność klastra $Spójność_kl$ wyznaczana jest przy założeniu, że grafem zastępczym jest graf dwudzielny, którego zbiorem $V1$ jest zbiór wszystkich ról, zbiorem $V2$ jest zbiór wszystkich typów obiektowych oprócz typów faktów, natomiast krawędziami są wszystkie połączenia, wynikające z definicji typów bazowych, relacji specjalizacji i polimorfizmu oraz ograniczeń zbiorowych, określających dodatkowe połączenia między rolami za pośrednictwem typów ze zbioru $V2$. Wyrażenie $Spójność_kl$ jest wyznaczana na podstawie metryk $LW1$ (5.9) i $LW2$ (5.10). Metryka $Spójność_kl$ dla elementów drzew specjalizacji i polimorfizmu jest wyznaczana tylko raz, niezależnie od liczby wystąpień poszczególnych typów w schemacie konceptualnym.

Krawędzie grafu funkcjonalnego klastra zdefiniowanego jako $Klaster(i, j, x)$ bez hermetryzacji (4.2) są wyznaczone jako zbiór $Role_kl$ (2.1, 2.3, 4.2, 2.114, 2.115)

$$Role_kl(z) = \{y / y \in TRol \wedge Baza(y) \in Klaster(i, j, z) \vee y \in RolePodzbiór \vee y \in RoleRówność\}. \quad \{*\} (5.21)$$

W zależności *LW1* i *LW2* parametr *a* jest więc równy mocy zbioru *Role_kl(x)*

$$a = |Role_kl(z)|. \quad \{*\} (5.22)$$

Parametr *x* jest równy liczbie ról należących do zbioru typów faktu wiążących typy obiektowe zawarte w klastrze z typem wiodącym klastra (2.1, 2.3, 4.2)

$$x = |\{y / y \in TRol \wedge Fakt(y) \in Klaster(i, j, z)\}|, \quad \{*\} (5.23)$$

natomiast parametr *y* jest równy liczbie pozostałych typów obiektowych, zawartych w klastrze, czyli

$$y = |\{t / t \in TOB \wedge t \in Klaster(i, j, z) \wedge t \neq z\}|. \quad \{*\} (5.24)$$

Spójność klastra jest więc równa (5.9, 5.10)

$$Spójność_kl = \begin{cases} LW1, & \text{gdy graf funkcjonalny klastra jest spójny (5.9),} \\ LW2, & \text{gdy graf funkcjonalny klastra jest niespójny (5.10).} \end{cases} \quad \{*\} (5.25)$$

Wyrażenia *LW1* i *LW2* stanowiące podstawę do określania spójności modelu ORM mogą być wykorzystane do badania jego efektywności.

Na podstawie oceny spójności grafów dwudzielnych podanej w rozdziale 5.2 można dokonać porównania różnych modeli reprezentujących tę samą aplikację. W tych różnych modelach mogą wystąpić następujące przypadki:

- modele zawsze posiadają tę same typy atrybutowe,
- w modelach może ulec zmianie liczba typów abstrakcyjnych i pomocniczych typów atrybutowych np. związanych z typami sekwencyjnymi,
- modele mogą posiadać różną liczbę typów faktów,
- w modelach może ulec zmianie liczba ról tworzących odpowiednio zbiory *Role_M* (5.15) lub *Role_kl* (5.21).

W celu porównania wartości spójności modelu klastrowego za pomocą metryki *LCOM* (5.7) wprowadzono metrykę *LCOM_W*, zdefiniowaną następująco:

$$LCOM_W = \frac{\sum_{n \in klaster} LCOM_n}{n}. \quad \{*\} (5.26)$$

W rozdziale 5.2 opisano wpływ poszczególnych parametrów na zmiany wartości metryk używanych do określenia spójności grafów funkcjonalnych.

Wartości metryk *Spójność_M* (5.19) oraz *Spójność_K* (5.20) powinny być możliwie duże przy zachowaniu możliwie małych wartości metryki *Połączenia*. (5.14).

5.4. Przykład zastosowania metryk do oceny równoważnych schematów konceptualnych ORM

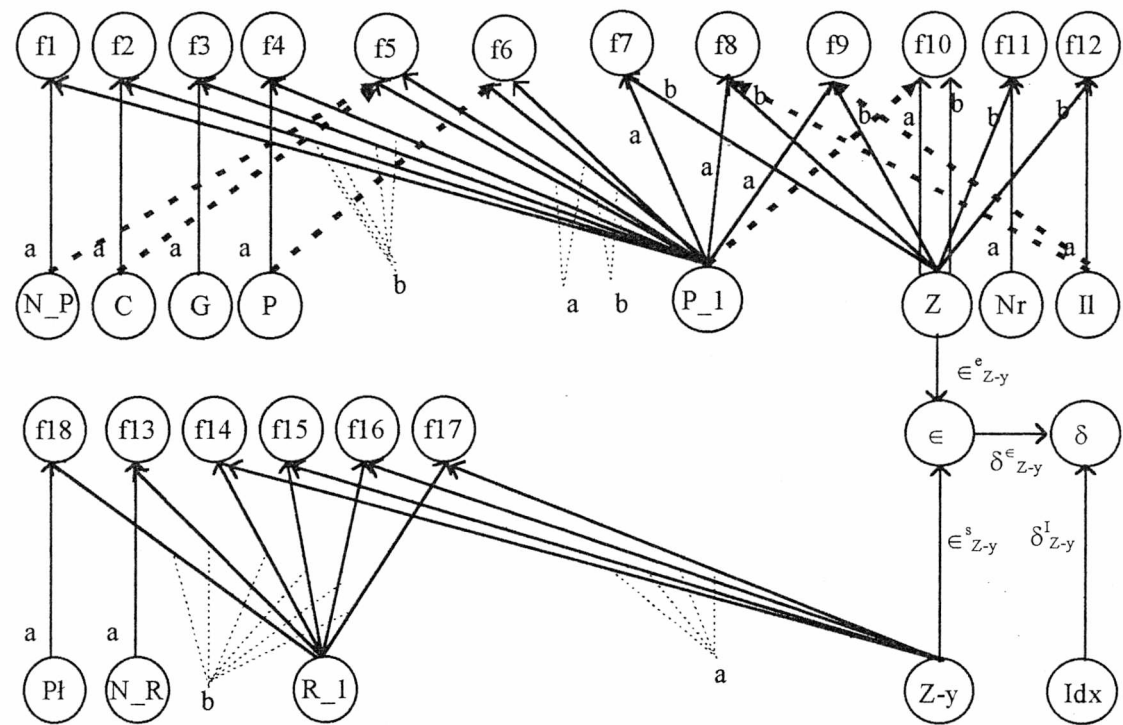
Mimo pewnych zasad określających tworzenie modelu konceptualnego można uzyskać kilka rozwiązań tego samego problemu (aksjomat d) w postaci kilku modeli. Ocena jakości ich zewnętrznych charakterystyk jest wyznaczana poprzez pewne metryki określające spójność i liczbę połączeń między klasami. Wybór modelu będzie określony najlepszymi wartościami tych metryk.

Metryki z rozdziału 5.3 zastosowano do oceny efektywności dwóch rozwiązań problemu sporządzania prostych rachunków, w postaci modeli zaprezentowanych w rozdziale 2.3. Dokonano tam analizy obu schematów konceptualnych, która wykazała ich równoważność. Wybór rozwiązania powinien wynikać z większej efektywności danego modelu. Za kryterium efektywności przyjęto liczbę połączeń między typami obiektowymi oraz spójność modelu.

Przykład:

Uwagi: Graf funkcjonalny zawiera krawędzie narysowane linią przerywaną, które reprezentują ograniczenia zbiorowe. W tabeli macierzy incydencji zaznaczono taką krawędź za pomocą 1*, natomiast wartości 2 oznaczają rekurencyjne typy faktów i są podane w parametrze *z* w metrykach spójności *LW1* (5.9) i *LW2* (5.10).

1. Model *M_1* (rozdział 2.3, rys. 2.14)

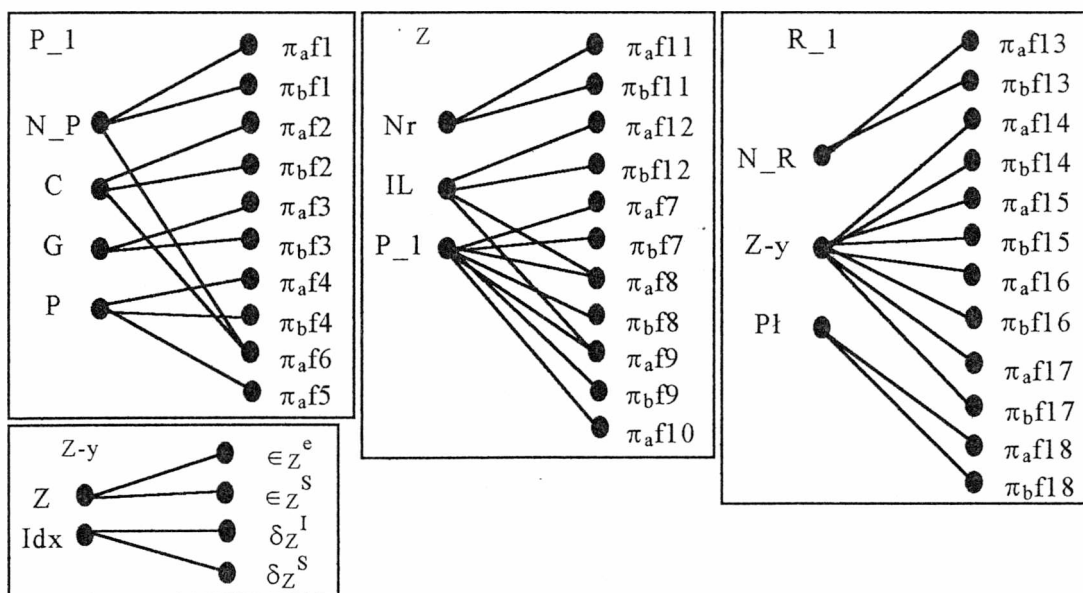


Rys. 5.6. Graf funkcjonalny modelu *M_1*

	<i>N_P</i>	<i>C</i>	<i>G</i>	<i>P</i>	<i>P_1</i>	<i>Z</i>	<i>Nr</i>	<i>IL</i>	<i>Z-y</i>	<i>Idx</i>	<i>R_1</i>	<i>N_R</i>	<i>Pł</i>
<i>f1</i>	1				1								
<i>f2</i>		1			1								

f3			1		1									
f4				1	1									
f5	1*	1*			2									
f6				1*	2									
f7					1	1								
f8					1	1		1*						
f9					1	1		1*						
f10					1*	2								
f11						1	1							
f12						1		1						
f13									-	1	1			
f14								1		1				
f15								1		1				
f16								1		1				
f17								1		1				
f18										1			1	
∈						1			1					
δ									1	1				

Rys. 5.7. Tabela macierzy incydencji grafu funkcjonalnego modelu M_1



Rys. 5.8. Podział modelu M_1 na klastry- grafy funkcjonalne

1.1. Połączenia między typami obiektowymi na podstawie wyrażeń: $Połączenia_{OB}$ (5.12), $Połączenia_W$ (5.13), $Połączenia$ (5.14).

$$Połączenia_{OB}(Produkt_1) = \{\pi_{af1}, \pi_{bf1}, \pi_{af2}, \pi_{bf2}, \pi_{af3}, \pi_{bf3}, \pi_{af4}, \pi_{bf4}, \pi_{af5}, \pi_{bf5}, \pi_{af6}, \pi_{bf6}, \pi_{af7}, \pi_{af8}, \pi_{af9}\},$$

$$Połączenia_{OB}(Zakup) = \{\pi_{af11}, \pi_{bf11}, \pi_{af12}, \pi_{bf12}, \pi_{af10}, \pi_{bf10}, \pi_{bf9}, \pi_{bf8}, \pi_{bf7}, \in^e_{z-y}\},$$

$$Połączenia_{OB}(Zakupy) = \{\in^s_{z-y}, \delta^s_{z-y}, \delta^I_{z-y}, \pi_{af17}, \pi_{af16}, \pi_{af15}, \pi_{af14}\},$$

$$Połączenia_{OB}(Rachunek_1) = \{\pi_{af18}, \pi_{bf18}, \pi_{af13}, \pi_{bf13}, \pi_{bf14}, \pi_{bf15}, \pi_{bf16}, \pi_{bf17}\},$$

$Połączenia_W = \{\pi_{af1}, \pi_{bf1}, \pi_{af2}, \pi_{bf2}, \pi_{af3}, \pi_{bf3}, \pi_{af4}, \pi_{bf4}, \pi_{af5}, \pi_{bf5}, \pi_{af6}, \pi_{bf6}, \pi_{af7}, \pi_{bf7}, \pi_{af8}, \pi_{bf8}, \pi_{af9}, \pi_{bf9}\} \cup \{\pi_{af11}, \pi_{bf11}, \pi_{af12}, \pi_{bf12}, \pi_{af10}, \pi_{bf10}, \pi_{bf9}, \pi_{bf8}, \pi_{bf7}, \in^e_{z-y}\} \cup \{\in^e_{z-y}, \delta^s_{z-y}, \delta^l_{z-y}, \pi_{af17}, \pi_{af16}, \pi_{af15}, \pi_{af14}\} \cup \{\pi_{af18}, \pi_{bf18}, \pi_{af13}, \pi_{bf13}, \pi_{bf14}, \pi_{bf15}, \pi_{bf16}, \pi_{bf17}\} = \{\pi_{af1}, \pi_{bf1}, \pi_{af2}, \pi_{bf2}, \pi_{af3}, \pi_{bf3}, \pi_{af4}, \pi_{bf4}, \pi_{af5}, \pi_{bf5}, \pi_{af6}, \pi_{bf6}, \pi_{af7}, \pi_{bf7}, \pi_{af8}, \pi_{bf8}, \pi_{af9}, \pi_{bf9}, \pi_{af11}, \pi_{bf11}, \pi_{af12}, \pi_{bf12}, \pi_{af10}, \pi_{bf10}, \pi_{bf9}, \pi_{bf8}, \pi_{bf7}, \in^e_{z-y}, \in^s_{z-y}, \delta^s_{z-y}, \delta^l_{z-y}, \pi_{af17}, \pi_{af16}, \pi_{af15}, \pi_{af14}, \pi_{af18}, \pi_{bf18}, \pi_{af13}, \pi_{bf13}, \pi_{bf14}, \pi_{bf15}, \pi_{bf16}, \pi_{bf17}\},$

$Połączenia = |Połączenia_W| = 40.$

1.2. Metryki spójności: $LCOM$ (5.7), $LCOM_W$ (5.26), $Spójność_kl$ (5.25), $Spójność_K$ (5.20), $Spójność_M$ (5.19).

Sekwencje ról:

$S1 = \{\pi_{bf15}, \pi_{af15}, \in^e_{z-y}, \pi_{bf9}, \pi_{af9}, \pi_{bf11}, \pi_{af11}, \pi_{bf12}, \pi_{af12}, \pi_{bf8}, \pi_{af8}, \pi_{bf2}, \pi_{af2}, \pi_{bf7}, \pi_{af7}, \pi_{bf4}, \pi_{af4}\},$

$S2 = \{\pi_{bf16}, \pi_{af16}, \in^e_{z-y}, \pi_{bf11}, \pi_{af11}, \pi_{bf10}, \pi_{af10}, \pi_{bf8}, \pi_{af8}, \pi_{bf6}, \pi_{af6}, \pi_{bf1}, \pi_{af1}, \pi_{bf2}, \pi_{af2}, \pi_{bf5}, \pi_{af5}, \pi_{bf4}, \pi_{af4}\},$

$\mu(Nazwa_P) = |\{\pi_{af1}, \pi_{bf1}, \pi_{bf6}\}| = 3,$

$\mu(Cena) = |\{\pi_{af2}, \pi_{bf2}, \pi_{bf6}\}| = 3,$

$\mu(Gatunek) = |\{\pi_{af3}, \pi_{bf3}\}| = 2,$

$\mu(Podatek) = |\{\pi_{af4}, \pi_{bf4}, \pi_{af5}\}| = 3,$

$\mu(Numer\ zakupu) = |\{\pi_{af11}, \pi_{af11}\}| = 2,$

$\mu(Ilość) = |\{\pi_{af12}, \pi_{af12}, \pi_{bf7}, \pi_{bf9}\}| = 4,$

$\mu(Indeks) = |\{\delta^l_{z-y}, \delta^s_{z-y}\}| = 2,$

$\mu(Numer\ rachunku) = |\{\pi_{af13}, \pi_{bf13}\}| = 2,$

$\mu(Płatność) = |\{\pi_{af18}, \pi_{bf18}\}| = 2,$

$\mu(Produkt) = |\{\pi_{af8}, \pi_{bf8}, \pi_{af7}, \pi_{bf7}, \pi_{bf9}, \pi_{af9}, \pi_{af10}\}| = 7,$

$\mu(Zakup) = |\{\in^e_{z-y}, \in^s_{z-y}\}| = 2,$

$\mu(Zakupy) = |\{\pi_{af14}, \pi_{bf14}, \pi_{af15}, \pi_{bf15}, \pi_{af16}, \pi_{bf16}, \pi_{af17}, \pi_{bf17}\}| = 8,$

$LCOM(Produkt) = ((3+3+2+3)/4 - 10)/(-9) = 29/36 = 0.806,$

$LCOM(Zakup) = ((2+4+7)/3 - 11)/(-10) = 20/30 = 0.667,$

$LCOM(Zakupy) = ((2+2)/2 - 4)/(-3) = 2/3 = 0.667,$

$LCOM(Rachunek) = ((8+2+2)/3 - 12)/(-11) = 8/11 = 0.727,$

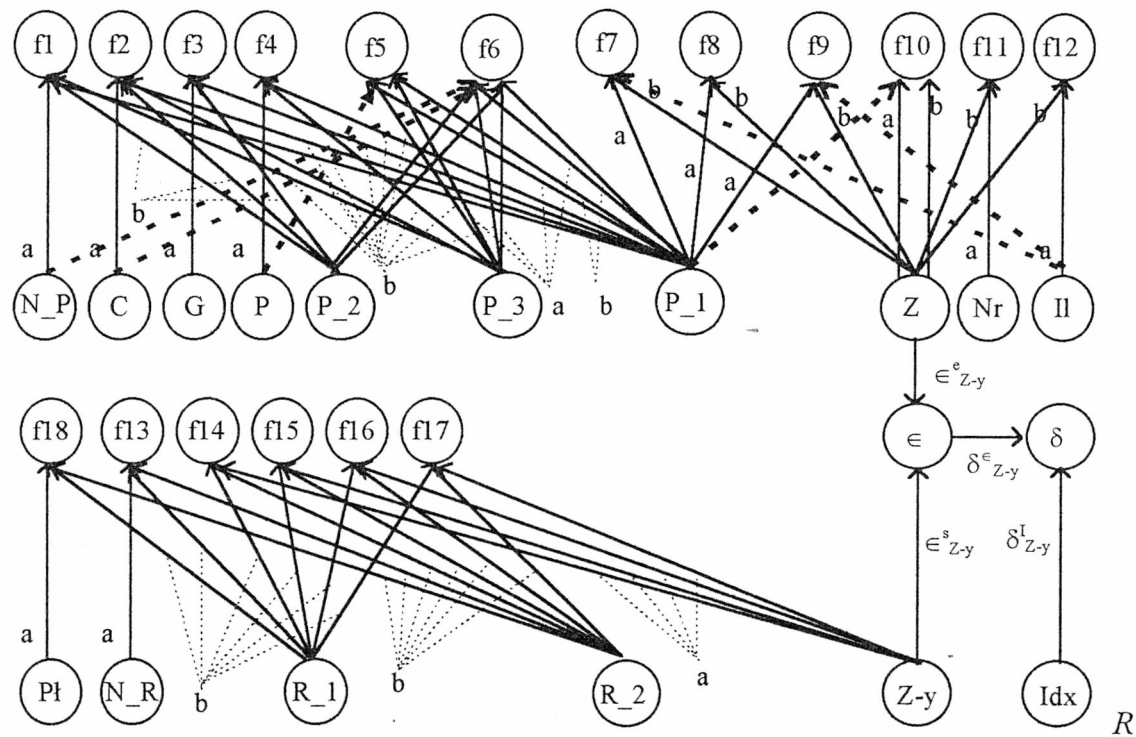
$LCOM_W = (0.806 + 0.667 + 0.667 + 0.727)/4 = 0.717,$

	P_1	Z	$Z-y$	R_1
$LCOM$	0.806	0.667	0.667	0.727
$LCOM_W$	0.717			
$Spójność_kl$	-0.3	-0.091	-0.5	-0.31
$Spójność_K$	-0.3			

Rys. 5.9. Tabela klastrowych metryk spójności modelu M_1

$$Spójność_M = \frac{20(46 - 35)}{13(20 * 13 - 32)} = 0,074, \quad LCOM = \frac{\frac{46}{13} - 20}{1 - 20} = 0,866.$$

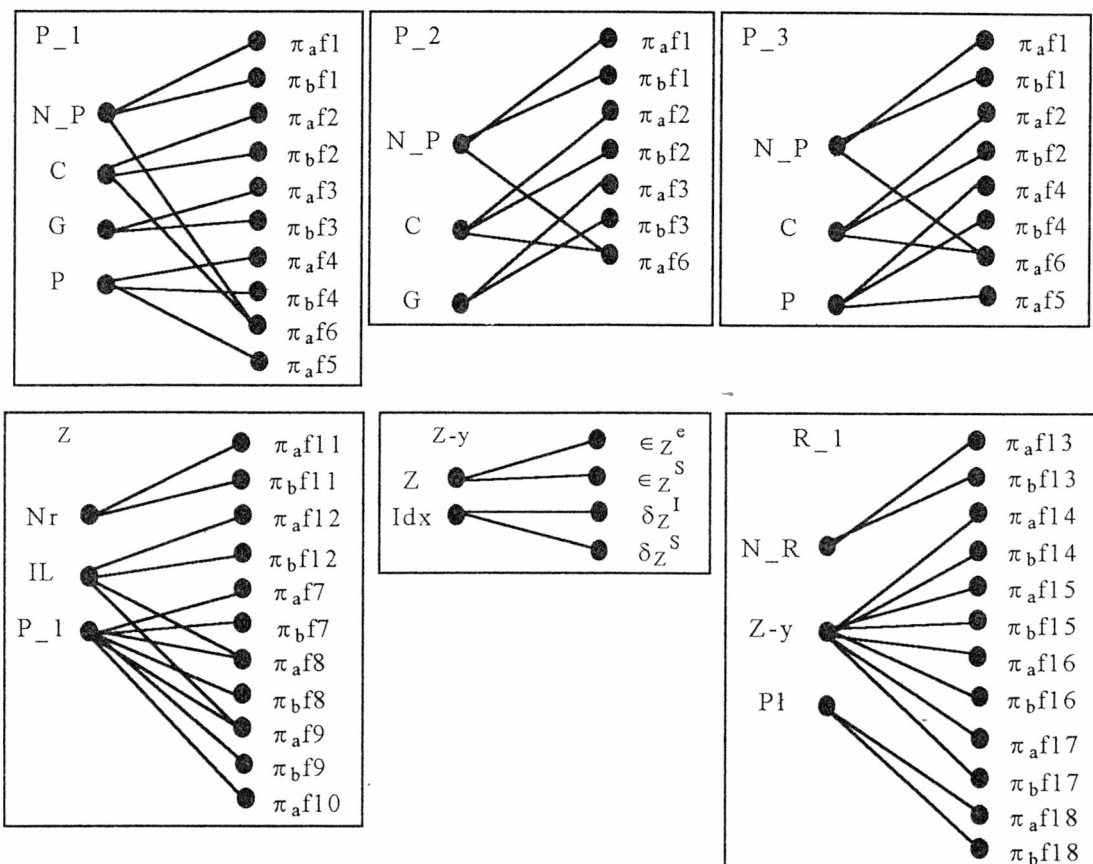
2. Model M_2 (rozdział 2.3, rys. 2.15):



Rys. 5.10. Graf funkcjonalny modelu M_2

	N_P	C	G	P	P_1	P_2	P_3	Z	Nr	IL	$Z-y$	Idx	R_1	R_2	N_R	Pt
$f1$	1				1	1	1									
$f2$		1			1	1	1									
$f3$			1		1	1										
$f4$				1	1		1									
$f5$				1*	2		2									
$f6$	1*	1*			2	2	2									
$f7$					1			1		1*						
$f8$					1			1								
$f9$					1			1		1*						
$f10$					1*			2								
$f11$								1	1							
$f12$								1		1						
$f13$											1		1	1	1	
$f14$											1		1	1		
$f15$											1		1	1		
$f16$											1		1	1		
$f17$											1		1	1		
$f18$													1	1		1
ϵ								1			1					
δ											1	1				

Rys. 5.11. Tabela macierzy incydencji grafu funkcjonalnego modelu M_2



Rys. 5.12. Podział modelu M_2 na klastry

2.1. Połączenia między typami obiektowymi na podstawie wyrażeń: $Połączenia_OB$ (5.12), $Połączenia_W$ (5.13), $Połączenia$ (5.14).

$$Połączenia_OB(Produkt_1) = \{\pi_{af1}, \pi_{bf1}, \pi_{af2}, \pi_{bf2}, \pi_{af3}, \pi_{bf3}, \pi_{af4}, \pi_{bf4}, \pi_{af5}, \pi_{bf5}, \pi_{af6}, \pi_{bf6}, \pi_{af7}, \pi_{af8}, \pi_{af9}\},$$

$$Połączenia_OB(Produkt_2) = \{\pi_{af1}, \pi_{bf1}, \pi_{af2}, \pi_{bf2}, \pi_{af3}, \pi_{bf3}, \pi_{af5}, \pi_{bf5}\},$$

$$Połączenia_OB(Produkt_3) = \{\pi_{af1}, \pi_{bf1}, \pi_{af2}, \pi_{bf2}, \pi_{af4}, \pi_{bf4}, \pi_{af5}, \pi_{bf5}, \pi_{af6}, \pi_{bf6}\},$$

$$Połączenia_OB(Zakup) = \{\pi_{af11}, \pi_{bf11}, \pi_{af12}, \pi_{bf12}, \pi_{af10}, \pi_{bf10}, \pi_{bf9}, \pi_{bf8}, \pi_{bf7}, \in^e_{z-y}\},$$

$$Połączenia_OB(Zakupy) = \{\in^s_{z-y}, \delta^s_{z-y}, \delta^I_{z-y}, \pi_{af17}, \pi_{af16}, \pi_{af15}, \pi_{af14}\},$$

$$Połączenia_OB(Rachunek_1) = \{\pi_{af18}, \pi_{bf18}, \pi_{af13}, \pi_{bf13}, \pi_{bf14}, \pi_{bf15}, \pi_{bf16}, \pi_{bf17}\},$$

$$Połączenia_OB(Rachunek_2) = \{\pi_{af18}, \pi_{bf18}, \pi_{af13}, \pi_{bf13}, \pi_{bf14}, \pi_{bf15}, \pi_{bf16}, \pi_{bf17}\},$$

$$Połączenia_W = \{\pi_{af1}, \pi_{bf1}, \pi_{af2}, \pi_{bf2}, \pi_{af3}, \pi_{bf3}, \pi_{af4}, \pi_{bf4}, \pi_{af5}, \pi_{bf5}, \pi_{af6}, \pi_{bf6}, \pi_{af7}, \pi_{af8}, \pi_{af9}\} \cup \{\pi_{af1}, \pi_{bf1}, \pi_{af2}, \pi_{bf2}, \pi_{af3}, \pi_{bf3}, \pi_{af5}, \pi_{bf5}\} \cup \{\pi_{af1}, \pi_{bf1}, \pi_{af2}, \pi_{bf2}, \pi_{af4}, \pi_{bf4}, \pi_{af5}, \pi_{bf5}, \pi_{af6}, \pi_{bf6}\} \cup \{\pi_{af11}, \pi_{bf11}, \pi_{af12}, \pi_{bf12}, \pi_{af10}, \pi_{bf10}, \pi_{bf9}, \pi_{bf8}, \pi_{bf7}, \in^e_{z-y}\} \cup \{\in^e_{z-y}, \delta^s_{z-y}, \delta^I_{z-y}, \pi_{af17}, \pi_{af16}, \pi_{af15}, \pi_{af14}\} \cup \{\pi_{af18}, \pi_{bf18}, \pi_{af13}, \pi_{bf13}, \pi_{bf14}, \pi_{bf15}, \pi_{bf16}, \pi_{bf17}\} = \{\pi_{af1}, \pi_{bf1}, \pi_{af2}, \pi_{bf2}, \pi_{af3}, \pi_{bf3}, \pi_{af4}, \pi_{bf4}, \pi_{af5}, \pi_{bf5}, \pi_{af6}, \pi_{bf6}, \pi_{af7}, \pi_{af8}, \pi_{af9}, \pi_{af11}, \pi_{bf11}, \pi_{af12}, \pi_{bf12}, \pi_{af10}, \pi_{bf10}, \pi_{bf9}, \pi_{bf8}, \pi_{bf7}, \in^e_{z-y}, \in^s_{z-y}, \delta^s_{z-y}, \delta^I_{z-y}, \pi_{af17}, \pi_{af16}, \pi_{af15}, \pi_{af14}, \pi_{af18}, \pi_{bf18}, \pi_{af13}, \pi_{bf13}, \pi_{bf14}, \pi_{bf15}, \pi_{bf16}, \pi_{bf17}\},$$

$$Połączenia = |Połączenia_W| = 40.$$

2.2. Metryki spójności: $LCOM$ (5.7), $LCOM_W$ (5.26), $Spójność_kl$ (5.25), $Spójność_K$ (5.20), $Spójność_M$ (5.19):

Sekwencje ról:

$$S1 = \{\pi_{bf15}, \pi_{af15}, \in^e_{z-y}, \pi_{bf9}, \pi_{af9}, \pi_{bf11}, \pi_{af11}, \pi_{bf12}, \pi_{af12}, \pi_{bf8}, \pi_{af8}, \pi_{bf2}, \pi_{af2}, \pi_{bf7}, \pi_{af7}, \pi_{bf4}, \pi_{af4}\},$$

$$S2 = \{\pi_{bf16}, \pi_{af16}, \in^e_{z-y}, \pi_{bf11}, \pi_{af11}, \pi_{bf10}, \pi_{af10}, \pi_{bf8}, \pi_{af8}, \pi_{bf6}, \pi_{af6}, \pi_{bf1}, \pi_{af1}, \pi_{bf2}, \pi_{af2}, \pi_{bf5}, \pi_{af5}, \pi_{bf4}, \pi_{af4}\},$$

$$\mu(Nazwa_P) = |\{\pi_{af1}, \pi_{bf1}, \pi_{bf6}\}| = 3,$$

$$\mu(Cena) = |\{\pi_{af2}, \pi_{bf2}, \pi_{bf6}\}| = 3,$$

$$\mu(Gatunek) = |\{\pi_{af3}, \pi_{bf3}\}| = 2,$$

$$\mu(Podatek) = |\{\pi_{af4}, \pi_{bf4}, \pi_{af5}\}| = 3,$$

$$\mu(Numer\ zakupu) = |\{\pi_{af11}, \pi_{af11}\}| = 2,$$

$$\mu(Ilość) = |\{\pi_{af12}, \pi_{af12}, \pi_{bf7}, \pi_{bf9}\}| = 4,$$

$$\mu(Indeks) = |\{\delta^I_{z-y}, \delta^S_{z-y}\}| = 2,$$

$$\mu(Numer\ rachunku) = |\{\pi_{af13}, \pi_{bf13}\}| = 2,$$

$$\mu(Płatność) = |\{\pi_{af18}, \pi_{bf18}\}| = 2,$$

$$\mu(Produkt_1) = |\{\pi_{af8}, \pi_{bf8}, \pi_{af7}, \pi_{bf7}, \pi_{bf9}, \pi_{af9}, \pi_{af10}\}| = 7,$$

$$\mu(Zakup) = |\{\in^e_{z-y}, \in^S_{z-y}\}| = 2,$$

$$\mu(Zakupy) = |\{\pi_{af14}, \pi_{bf14}, \pi_{af15}, \pi_{bf15}, \pi_{af16}, \pi_{bf16}, \pi_{af17}, \pi_{bf17}\}| = 8,$$

$$LCOM(Produkt_1) = ((3+3+2+3)/4 - 10)/(-9) = 29/36 = 0.806,$$

$$LCOM(Produkt_2) = ((3+3+2)/3 - 7)/(-6) = 13/18 = 0.722,$$

$$LCOM(Produkt_3) = ((3+3+3)/3 - 8)/(-7) = 5/7 = 0.714,$$

$$LCOM(Zakup) = ((2+4+7)/3 - 11)/(-10) = 20/30 = 0.667,$$

$$LCOM(Zakupy) = ((2+2)/2 - 4)/(-3) = 2/3 = 0.667,$$

$$LCOM(Rachunek_1) = (8+2+2)/3 - 12)/(-11) = 8/11 = 0.727,$$

$$LCOM(Rachunek_2) = (8+2+2)/3 - 12)/(-11) = 8/11 = 0.727,$$

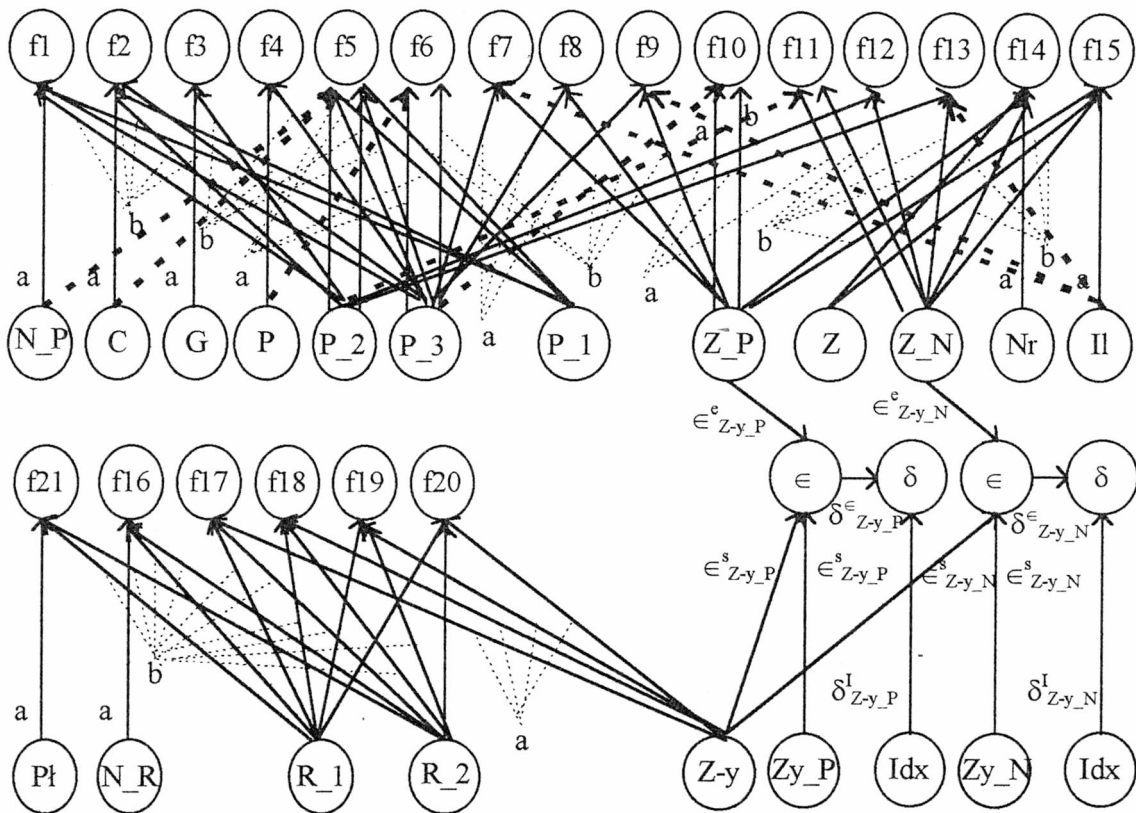
$$LCOM_W = (0.806 + 0.722 + 0.714 + 0.667 + 0.667 + 0.727 + 0.727)/7 = 5.03/7 = 0.719,$$

	P_1	P_2	P_3	Z	$Z-y$	R_1	R_2
$LCOM$	0.806	0.722	0.714	0.667	0.667	0.727	0.727
$LCOM_W$	0.719						
$Spójność_kl$	-0.3	-0.29	-0.25	-0.09	-0.5	-0.31	-0.31
$Spójność_K$	-0.293						

Rys. 5.13. Tabela wartości klastrowych metryk spójności modelu M_2

$$Spójność_M = \frac{20(64 - 41)}{16(20 * 16 - 35)} = 0,101, \quad LCOM = \frac{\frac{64}{16} - 20}{1 - 20} = 0,842.$$

3. Model M_3 (rozdział 2.3, rys. 2.16)

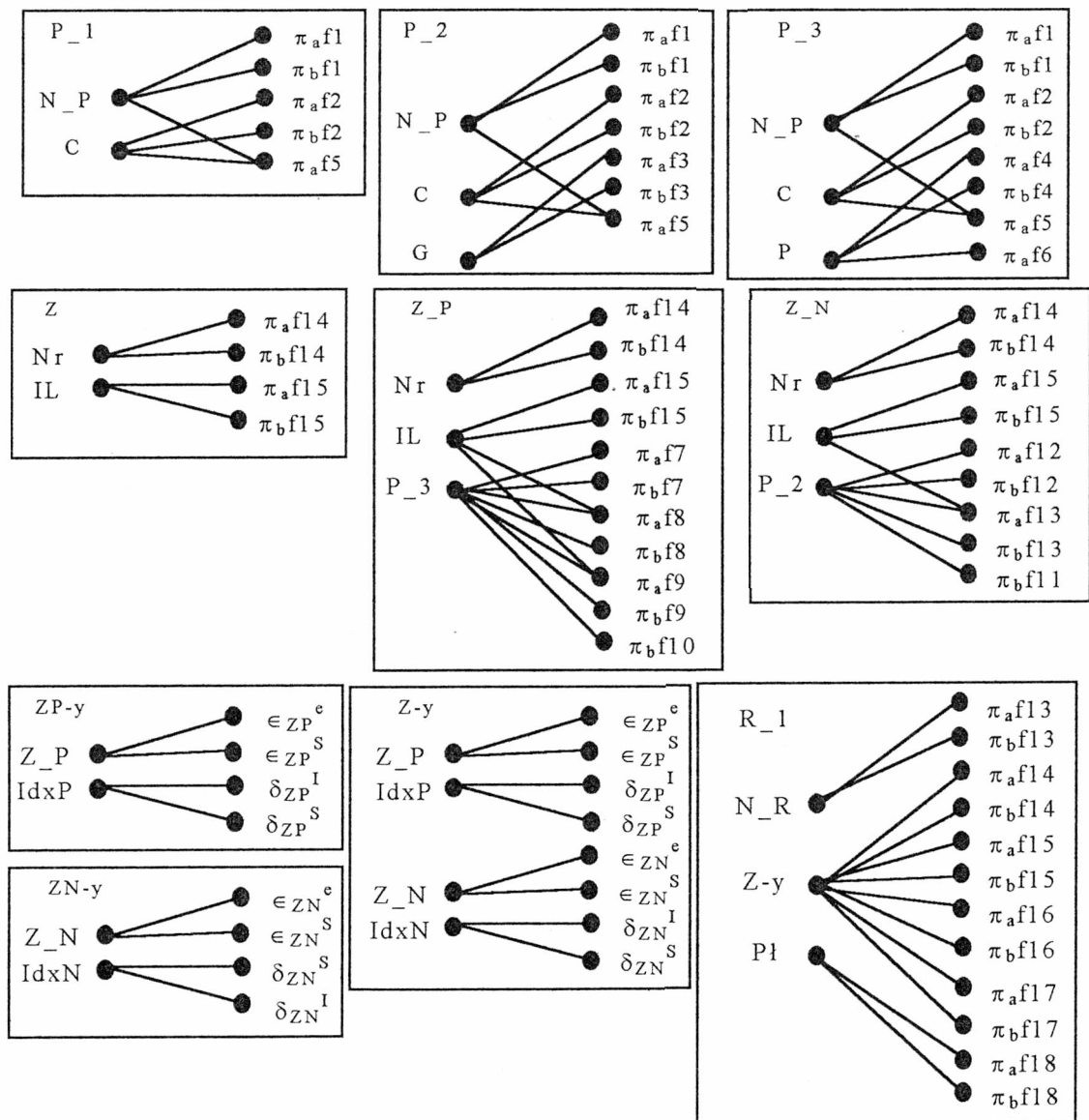


Rys. 5.14. Graf funkcjonalny modelu M_3

	N_P	C	G	P	P_1	P_2	P_3	Z	Z_P	Z_N	Nr	Il
$f1$	1				1	1	1					
$f2$		1			1	1	1					
$f3$			1			1						
$f4$				1			1					
$f5$	1*	1*			2	2	2					
$f6$				1*			2					
$f7$							1		1			1*
$f8$							1		1			
$f9$							1		1			1*
$f10$							1*		2			
$f11$						1*				2		
$f12$						1				1		
$f13$						1				1		1*
$f14$								1	1	1	1	
$f15$								1	1	1		1
$f21$												
$\in_{z_P}$									1			
δ_{z_P}												
$\in_{z_N}$										1		
δ_{z_N}												

		ZP_y	ZN_y	Idx_P	Idx_N	Z-y	R_1	R_2	N_R	Pł
f1										
f16							1	1	1	
f17						1	1	1		
f18						1	1	1		
f19						1	1	1		
f20						1	1	1		
f21							1	1		1
ϵ_{zP}		1				1				
δ_{zP}		1		1		1				
ϵ_{zN}			1			1				
δ_{zN}			1		1	1				

Rys. 5.15. Tabela macierzy incydencji grafu funkcjonalnego modelu M_3



Rys. 5.16. Podział modelu M_3 na klastry

3.1. Połączenia między typami obiektowymi na podstawie wyrażeń: *Połączenia_OB* (5.12), *Połączenia_W* (5.13), *Połączenia* (5.14).

$$Połączenia_OB(Produkt_1) = \{\pi_{af1}, \pi_{bf1}, \pi_{af2}, \pi_{bf2}, \pi_{af5}, \pi_{bf5}\},$$

$$Połączenia_OB(Produkt_2) = \{\pi_{af1}, \pi_{bf1}, \pi_{af2}, \pi_{bf2}, \pi_{af3}, \pi_{bf3}, \pi_{af5}, \pi_{bf5}, \pi_{af12}, \pi_{af13}\},$$

$$Połączenia_OB(Produkt_3) = \{\pi_{af1}, \pi_{bf1}, \pi_{af2}, \pi_{bf2}, \pi_{af4}, \pi_{bf4}, \pi_{af5}, \pi_{bf5}, \pi_{af6}, \pi_{bf6}, \pi_{af7}, \pi_{af8}, \pi_{af9}\},$$

$$Połączenia_OB(Zakup_P) = \{\pi_{af14}, \pi_{bf14}, \pi_{af15}, \pi_{bf15}, \pi_{bf7}, \pi_{bf8}, \pi_{bf9}, \pi_{af10}, \pi_{bf10}, \in^e_{z-y_P}\},$$

$$Połączenia_OB(Zakup_N) = \{\pi_{af14}, \pi_{bf14}, \pi_{af15}, \pi_{bf15}, \pi_{bf12}, \pi_{bf13}, \pi_{af11}, \pi_{bf11}, \in^e_{z-y_N}\},$$

$$Połączenia_OB(Zakup) = \{\pi_{af14}, \pi_{bf14}, \pi_{af15}, \pi_{bf15}\},$$

$$Połączenia_OB(Zakupy_P) = \{\in^s_{z-y_P}, \delta^s_{z-y_P}, \delta^I_{z-y_P}\},$$

$$Połączenia_OB(Zakupy_N) = \{\in^s_{z-y_N}, \delta^s_{z-y_N}, \delta^I_{z-y_N}\},$$

$$Połączenia_OB(Zakupy) = \{\in^s_{z-y_P}, \delta^s_{z-y_P}, \delta^I_{z-y_P}, \in^s_{z-y_N}, \delta^s_{z-y_N}, \delta^I_{z-y_N}, \pi_{af17}, \pi_{af18}, \pi_{af19}, \pi_{af20}\},$$

$$Połączenia_OB(Rachunek_1) = \{\pi_{af21}, \pi_{bf21}, \pi_{af16}, \pi_{bf16}, \pi_{bf17}, \pi_{bf18}, \pi_{bf19}, \pi_{bf20}\},$$

$$Połączenia_OB(Rachunek_2) = \{\pi_{af21}, \pi_{bf21}, \pi_{af16}, \pi_{bf16}, \pi_{bf17}, \pi_{bf18}, \pi_{bf19}, \pi_{bf20}\},$$

$$\begin{aligned} Połączenia_W = & \{\pi_{af1}, \pi_{bf1}, \pi_{af2}, \pi_{bf2}, \pi_{af5}, \pi_{bf5}\} \cup \{\pi_{af1}, \pi_{bf1}, \pi_{af2}, \pi_{bf2}, \pi_{af3}, \pi_{bf3}, \pi_{af5}, \pi_{bf5}, \pi_{af12}, \pi_{af13}\} \cup \{\pi_{af1}, \pi_{bf1}, \pi_{af2}, \pi_{bf2}, \pi_{af4}, \pi_{bf4}, \pi_{af5}, \pi_{bf5}, \pi_{af6}, \pi_{bf6}, \pi_{af7}, \pi_{af8}, \pi_{af9}\} \cup \{\pi_{af14}, \pi_{bf14}, \pi_{af15}, \pi_{bf15}, \pi_{bf7}, \pi_{bf8}, \pi_{bf9}, \pi_{af10}, \pi_{bf10}, \in^e_{z-y_P}\} \cup \{\pi_{af14}, \pi_{bf14}, \pi_{af15}, \pi_{bf15}, \pi_{bf12}, \pi_{bf13}, \pi_{af11}, \pi_{bf11}, \in^e_{z-y_N}\} \cup \{\pi_{af14}, \pi_{bf14}, \pi_{af15}, \pi_{bf15}\} \cup \{\in^s_{z-y}, \delta^s_{z-y}, \delta^I_{z-y}\} \cup \{\in^s_{z-y}, \delta^s_{z-y}, \delta^I_{z-y}\} \cup \{\in^s_{z-y_P}, \delta^s_{z-y_P}, \delta^I_{z-y_P}, \in^s_{z-y_N}, \delta^s_{z-y_N}, \delta^I_{z-y_N}, \pi_{af17}, \pi_{af18}, \pi_{af19}, \pi_{af20}\} \cup \{\pi_{af21}, \pi_{bf21}, \pi_{af16}, \pi_{bf16}, \pi_{bf17}, \pi_{bf18}, \pi_{bf19}, \pi_{bf20}\} \cup \{\pi_{af21}, \pi_{bf21}, \pi_{af16}, \pi_{bf16}, \pi_{bf17}, \pi_{bf18}, \pi_{bf19}, \pi_{bf20}\} = \\ & \{\pi_{af1}, \pi_{bf1}, \pi_{af2}, \pi_{bf2}, \pi_{af3}, \pi_{bf3}, \pi_{af4}, \pi_{bf4}, \pi_{af5}, \pi_{bf5}, \pi_{af6}, \pi_{bf6}, \pi_{af12}, \pi_{af13}, \pi_{af7}, \pi_{af8}, \pi_{af9}, \pi_{af14}, \pi_{bf14}, \pi_{af15}, \pi_{bf15}, \pi_{bf7}, \pi_{bf8}, \pi_{bf9}, \pi_{af10}, \pi_{bf10}, \pi_{bf12}, \pi_{bf13}, \pi_{af11}, \pi_{bf11}, \in^e_{z-y_N}, \in^s_{z-y_P}, \in^e_{z-y_P}, \delta^s_{z-y_P}, \delta^I_{z-y_P}, \in^s_{z-y_N}, \in^e_{z-y_N}, \delta^s_{z-y_N}, \delta^I_{z-y_N}, \pi_{af17}, \pi_{af18}, \pi_{af19}, \pi_{af20}, \pi_{af21}, \pi_{bf21}, \pi_{af16}, \pi_{bf16}, \pi_{bf17}, \pi_{bf18}, \pi_{bf19}, \pi_{bf20}\}, \end{aligned}$$

$$Połączenia = |Połączenia_W| = 50.$$

3.2. Metryki spójności: *LCOM* (5.7), *LCOM_W* (5.26), *Spójność_kl* (5.25), *Spójność_K* (5.20), *Spójność_M* (5.21).

Sekwencje ról:

$$S1 = \{\pi_{bf18}, \pi_{af18}, \in^e_{z-y_P}, \pi_{bf14}, \pi_{af14}, \pi_{bf15}, \pi_{af15}, \pi_{bf7}, \pi_{af7}, \pi_{bf2}, \pi_{af2}, \pi_{bf9}, \pi_{af9}, \pi_{bf4}, \pi_{af4}\},$$

$$S2 = \{\pi_{bf18}, \pi_{af18}, \in^e_{z-y_N}, \pi_{bf14}, \pi_{af14}, \pi_{bf15}, \pi_{af15}, \pi_{bf13}, \pi_{af13}, \pi_{bf2}, \pi_{af2}\},$$

$$S3 = \{\pi_{bf19}, \pi_{af19}, \in^e_{z-y_P}, \pi_{bf14}, \pi_{af14}, \pi_{bf10}, \pi_{af10}, \pi_{bf8}, \pi_{af8}, \pi_{bf5}, \pi_{af5}, \pi_{bf1}, \pi_{af1}, \pi_{bf2}, \pi_{af2}, \pi_{bf6}, \pi_{af6}, \pi_{bf4}, \pi_{af4}\},$$

$$S4 = \{\pi_{bf19}, \pi_{af19}, \in^e_{z-y_N}, \pi_{bf14}, \pi_{af14}, \pi_{bf11}, \pi_{af11}, \pi_{bf12}, \pi_{af12}, \pi_{bf5}, \pi_{af5}, \pi_{bf1}, \pi_{af1}, \pi_{bf2}\},$$

$$\begin{aligned}
\mu(\text{Nazwa_P}) &= |\{\pi_{af1}, \pi_{bf1}, \pi_{bf5}\}| = 3, \\
\mu(\text{Cena}) &= |\{\pi_{af2}, \pi_{bf2}, \pi_{bf5}\}| = 3, \\
\mu(\text{Gatunek}) &= |\{\pi_{af3}, \pi_{bf3}\}| = 2, \\
\mu(\text{Podatek}) &= |\{\pi_{af4}, \pi_{bf4}, \pi_{af6}\}| = 3, \\
\mu(\text{Numer zakupu}) &= |\{\pi_{af14}, \pi_{af4}\}| = 2, \\
\mu_1(\text{Ilość}) &= |\{\pi_{af15}, \pi_{af15}, \pi_{bf13}\}| = 3, \\
\mu_1(\text{Ilość}) &= |\{\pi_{af15}, \pi_{af15}, \pi_{bf8}, \pi_{bf9}\}| = 4, \\
\mu(\text{Indeks_P}) &= |\{\delta^I_{z-y_P}, \delta^S_{z-y_P}\}| = 2, \\
\mu(\text{Indeks_N}) &= |\{\delta^I_{z-y_N}, \delta^S_{z-y_P}\}| = 2, \\
\mu(\text{Numer rachunku}) &= |\{\pi_{af16}, \pi_{bf16}\}| = 2, \\
\mu(\text{Płatność}) &= |\{\pi_{af21}, \pi_{bf21}\}| = 2, \\
\mu(\text{Produkt_2}) &= |\{\pi_{af12}, \pi_{bf12}, \pi_{af11}, \pi_{bf13}, \pi_{af13}\}| = 5, \\
\mu(\text{Produkt_3}) &= |\{\pi_{af8}, \pi_{bf8}, \pi_{af7}, \pi_{bf7}, \pi_{af9}, \pi_{bf9}, \pi_{af10}\}| = 7, \\
\mu(\text{Zakup_P}) &= |\{\epsilon^e_{z-y_P}, \epsilon^S_{z-y_P}\}| = 2, \\
\mu(\text{Zakup_N}) &= |\{\epsilon^e_{z-y_N}, \epsilon^S_{z-y_N}\}| = 2, \\
\mu(\text{Zakupy}) &= |\{\pi_{af17}, \pi_{bf17}, \pi_{af18}, \pi_{bf18}, \pi_{af19}, \pi_{bf19}, \pi_{af20}, \pi_{bf20}\}| = 8,
\end{aligned}$$

$$\begin{aligned}
LCOM(\text{Produkt_1}) &= ((3+3)/2 - 5)/(-4) = 3/5 = 0.6, \\
LCOM(\text{Produkt_2}) &= ((3+3+2)/3 - 7)/(-6) = 16/21 = 0.762, \\
LCOM(\text{Produkt_3}) &= ((3+3+3)/3 - 8)/(-7) = 5/7 = 0.714, \\
LCOM(\text{Zakup}) &= ((2+2)/2 - 4)/(-3) = 2/3 = 0.667, \\
LCOM(\text{Zakup_P}) &= ((2+4+7)/3 - 11)/(-10) = 20/30 = 0.667, \\
LCOM(\text{Zakup_N}) &= ((2+3+5)/3 - 9)/(-8) = 17/24 = 0.708, \\
LCOM(\text{Zakupy_P}) &= ((2+2)/2 - 4)/(-3) = 2/3 = 0.667, \\
LCOM(\text{Zakupy_N}) &= ((2+2)/2 - 4)/(-3) = 2/3 = 0.667, \\
LCOM(\text{Zakupy}) &= ((2+2+2+2)/4 - 8)/(-7) = 6/7 = 0.857, \\
LCOM(\text{Rachunek_1}) &= (8+2+2)/3 - 12)/(-11) = 0.727, \\
LCOM(\text{Rachunek_2}) &= (8+2+2)/3 - 12)/(-11) = 0.727, \\
LCOM_W &= (0.600+0.762+0.714+0.667+0.667+0.708+0.667+0.667+0.857+0.727; \\
&\quad +0.727)/11 = 7.763/11 = 0.706.
\end{aligned}$$

	P_1	P_2	P_3	Z	Z_P	Z_N	ZP-y	ZN-y	Z-y	R_1	R_2
LCOM	0.600	0.762	0.714	0.667	0.667	0.708	0.667	0.667	0.857	0.727	0.727
LCOM_W	0.706										
Spójność_kl	0	-0.29	-0.25	-0.5	-0.09	-0.22	-0.5	-0.5	-0.6	-0.31	-0.31
Spójność_K	-0.324										

Rys. 5.17. Tabela wartości klastrowych metryk spójności modelu M_3

$$\text{Spójność_M} = \frac{25(80 - 51)}{21(25 * 21 - 45)} = 0,072, \quad LCOM = \frac{\frac{80}{21} - 25}{1 - 25} = 0,883.$$

Wniosek:

Model M_2 (rys 5.10) jest efektywniejszy, gdyż cechują go: mniejsza liczba połączeń (5.14) oraz wyższa spójność czyli większa wartość $Spójność_M$ (5.19), $Spójność_K$ (5.20) oraz mniejsza wartość $LCOM$ (5.7). Na drugim miejscu jest model bez dziedziczenia M_1 (rys. 5.6), natomiast najgorsze wyniki uzyskano z modelu M_3 , gdzie zastosowano rozbudowany model dziedziczenia.

Jedynie wyniki związane z wartością metryki klastrowej spójności $LCOM_W$ (5.26) odbiegają od pozostałych, co wynika z braku wpływu na tę metrykę liczby składowych niespójności grafu dwudzielnego, jakimi są właśnie grafy funkcjonalne klastrow.

Metodę wyznaczania zbioru rozwiązań efektywnych i wybór rozwiązania dominującego z tego zbioru przedstawiono w rozdziale 6. 2.2.

Model	Połączenia	$Spójność_K$	$LCOM_W$	$Spójność_M$	$LCOM$
M_1	40	-0.3	0.717	0.074	0.866
M_2	40	-0.293	0.719	0.101	0.842
M_3	50	-0.324	0.706	-0.072	0.883

Rys. 5.18. Wartości metryk połączeń i spójności w modelach M_1 , M_2 , M_3

6. Metoda tworzenia i wyboru efektywnego modelu opartego na polimorficznej strukturze danych

6.1. Podstawy formalne metody syntezy modeli opartych na polimorficznych strukturach danych

*Metoda tworzenia efektywnego modelu opracowana w rozdziale podaje sposób stosowania relacji *Spec* lub *Pol*, poprawia charakterystyki wewnętrzne rozwiązania (spójność i połączenia) za pomocą polimorficznych struktur danych i umożliwia wybór najlepszego z uzyskanych poprawnych rozwiązań.*

Metoda tworzenia efektywnego modelu oparta jest na stosowaniu polimorficznych struktur danych, które ograniczają liczbę połączeń i zwiększają spójność modelu: globalną i klastrową. Podstawą metody są twierdzenia 6.1, 6.2 i 6.3. Twierdzenie 6.1 podaje warunki, w których można zastąpić relację specjalizacji relacją polimorfizmu, natomiast twierdzenia 6.2 i 6.3 określają poprawę charakterystyk wewnętrznych oprogramowania. Przedstawione algorytmy w rozdziale 6.2.1 opisują sposób tworzenia modeli ORM, umożliwiając odpowiednio zastosowanie polimorfizmu w uzasadnionych przypadkach. Ponieważ elementy stosowane w modelu wyrażają semantykę problemu, daje to możliwość uzyskania różnych poprawnych semantycznie rozwiązań.

W celu udowodnienia tych twierdzeń wprowadza się funkcję *Atrybuty_Spec* generującą wszystkie typy atrybutowe związane ze zbiorem ról *WRole* (2.27) dowolnego typu x należącego do zbioru typów obiektowych połączonych relacją *Spec*, zwanego drzewem specjalizacji (2.25, 2.22, 2.2, 2.14)

$$\begin{aligned} Atrybuty_Spec(x) = \{y \mid y \in SpecAtrybuty(x) \vee y \in NadAtrybuty(x) \\ \vee (\exists q \in \delta \wedge p \in Fakt(q) [konkr(p)])\}. \end{aligned} \quad \{*\} (6.1)$$

Zbiór δ jest opisany za pomocą wyrażenia *Połączenie_SI*(σ , δ). Początkowa zawartość zbioru jest równa $\sigma = WRole(x)$ (2.27) oraz $\delta = \sigma$. Stąd mamy (2.96, 2.11, 2.14, 2.15, 2.13, 2.3)

$$\begin{aligned} Połączenie_SI(\sigma, \delta) \equiv Fakty(\sigma) \neq \emptyset \Rightarrow \exists p \in \sigma \exists q \in Inne_Role(p) [konkr(q)] \vee \\ (\exists p \in \sigma \exists q \in Inne_Role(p) [abstr(q)] \wedge \\ Połączenie_SI(Role_Bazy(Baza(q)), Role_Bazy(Baza(q)) \cup \sigma)), \end{aligned} \quad \{*\} (6.2)$$

które wyznacza zbiór wszystkich typów atrybutowych połączonych z typem x należącym do rodziny specjalizacji.

Podobnie dla wyznaczenia zbioru typów atrybutowych dla dowolnego typu x należącego do zbioru typów obiektowych związanych relacją *Pol*, zwanego drzewem polimorficznym bez specjalizacji, zdefiniowano funkcję *Atrybuty_Pol* (2.33, 2.2, 2.14)

$$\begin{aligned} Atrybuty_Pol(x) = \{y \mid y \in PolAtrybuty(x) \\ \vee (\exists q \in \gamma \wedge p \in Fakt(q) [konkr(p)])\}. \end{aligned} \quad \{*\} (6.3)$$

Zbiór γ jest opisany za pomocą wyrażenia *Połączenie_SI*(τ , γ) (6.2). Początkowa zawartość zbioru jest równa $\tau = PolRole(x)$ (2.32) oraz $\gamma = \tau$.

W zbiorze typów obiektowych związanych relacją *Pol*, w którym występuje specjali-

zacja, zwanym drzewem specjalizacji z dziedziczeniem, zbiory typów atrybutowych są wyznaczone za pomocą funkcji $Atrybuty_PolS(x)$ (2.38, 2.2, 2.14)

$$Atrybuty_PolS(x) = \{y / y \in PolAtrybutyS(x) \vee (\exists q \in \varphi \wedge p \in Fakt(q) [konkr(p)])\} \quad \{*\} (6.4)$$

Zbiór φ jest opisany za pomocą wyrażenia $Połączenie_SI(\xi, \varphi)$ (6.2). Początkowa zawartość zbioru jest równa $\tau = PolRoleS(x)$ (2.37) oraz $\gamma = \tau$.

Warunki zastąpienia drzewa specjalizacji drzewem polimorficznym podaje twierdzenie 6.1.

Twierdzenie 6.1: Fragment struktury informacyjnej będący drzewem specjalizacji może być zastąpiony fragmentem struktury będącym drzewem polimorficznym wtedy i tylko wtedy, jeżeli oba fragmenty są równoważne w sensie E4. {*}

Dowód

Wystarczalność:

W celu przeprowadzenia dowodu badamy typ obiektowy typu *Korzeń* (2.9) drzewa polimorficznego oraz *Liście* drzewa specjalizacji (2.21), zgodnie z własnościami obu relacji: *Pol* i *Spec* (2.7, 2.8). Przy danym założeniu zaprzeczamy tezie. Stwierdzamy, że istnieje dodatkowa liczba typów atrybutowych w zbiorze wyznaczonym za pomocą funkcji $Atrybuty_Pol$ (6.3) lub $Atrybuty_PolS$ (6.4) w *Korzeniu* drzewa polimorficznego. Są to takie typy atrybutowe, które nie są połączone za pomocą zbioru ról $\bigcup_{x \in Liście_Spec(Korzeń(x))} Atrybuty_Spec(x)$ na poziome *Liści* w drzewie specjalizacji (2.1, 2.9, 6.1). Jeśli tak, to drzewo specjalizacji nie zawiera wszystkich zależności funkcyjnych lub zależności niefunkcyjnych, które zawiera drzewo polimorficzne. Oznacza to brak równoważności w sensie E4. Oznacza to zaprzeczenie założenia.

Konieczność:

Założmy, że dwa schematy drzew nie są równoważne w sensie E4. Zakłada się, że typy atrybutowe w zbiorze $\bigcup_{x \in Liście_Spec(Korzeń(x))} Atrybuty_Spec(x)$ na poziome *Liści* w drzewie specjalizacji (2.1, 2.9, 6.1) są identyczne z typami atrybutowymi w zbiorze wyznaczonym za pomocą funkcji $Atrybuty_Pol$ (6.3) lub $Atrybuty_PolS$ (6.4) w *Korzeniu* drzewa polimorficznego. W obu schematach istnieją różne zależności funkcyjne, gdyż nie są równoważne w sensie E4. W takiej sytuacji nie można zastąpić drzewa specjalizacji drzewem polimorficznym. Doszliśmy więc do sprzeczności. Potwierdza to możliwość zastąpienia w schemacie konceptualnym fragmentów modelu np. drzewo specjalizacji drzewem polimorficznym, jeżeli są równoważne w sensie E4.

Wniosek 6.1: Można dokonać konwersji wybranego fragmentu schematu ORM na inny fragment, jeżeli oba fragmenty są równoważne w sensie E4.

Twierdzenie 6.2 określa warunek wystarczalności uzyskania lepszych charakterystyk wewnętrznych tych fragmentów modeli ORM, które zawierają drzewo specjalizacji przez zastosowanie konwersji tego drzewa na drzewo polimorficzne równoważne w sensie E4.

Twierdzenie 6.2: Zastąpienie drzewa specjalizacji drzewem polimorficznym równoważnym w sensie E4 zwiększa wartość metryki *Spójność_M* tego fragmentu modelu, natomiast wartość metryki *Połączenia_W* tego fragmentu modelu jest równa lub mniejsza po konwersji. {*}

Dowód

Dowód przeprowadzono dla: $a1 \text{ Spec } c1, \dots, an \text{ Spec } c1$ oraz równoważnego schematu $c2 \text{ Pol } b1, \dots, c2 \text{ Pol } bn$ bez specjalizacji i ze specjalizacją $c3 \text{ Pol } d1, \dots, c3 \text{ Pol } dn$ (rys. 6.1, 6.2, 6.3). Po zastosowaniu algorytmu z rozdziału 3.2. można udowodnić równoważność obu schematów w sensie E4. Stąd możliwe jest zastąpienie w dowolnym schemacie ORM drzewa specjalizacji drzewem polimorficznym. Dodatkowo założono taką samą liczbę typów atrybutowych i typów faktów we wszystkich schematach oraz (2.12)

$$\text{Role_Bazy}(a1) = \text{Role_Bazy}(b1) = \text{Role_Bazy}(d1), \quad \{*\} (6.5)$$

.....

$$\text{Role_Bazy}(an) = \text{Role_Bazy}(bn) = \text{Role_Bazy}(dn), \quad \{*\} (6.6)$$

$$\text{Role_Bazy}(c1) = \text{Role_Bazy}(c2) = \text{Role_Bazy}(c3). \quad \{*\} (6.7)$$

Każdy typ faktu jest typem binarnym. Wymienione zbiory ról są wzajemnie rozłączne. Pozostałe zbiory ról wyznacza się za pomocą funkcji (2.12, 2.2)

$$\text{Role_Bazy}'(x) = \{y \mid \exists p \in \text{Role_Bazy}(x) [y \in \text{Fakt}(p) \setminus p]\}. \quad \{*\} (6.8)$$

Na podstawie tej funkcji oraz dwóch wymienionych własności moce zbiorów $\text{Role_Bazy}'(x)$ odpowiadające zbiorom $\text{Role_Bazy}(x)$ są równe (2.12)

$$|\text{Role_Bazy}(x)| = |\text{Role_Bazy}'(x)|. \quad \{*\} (6.9)$$

Zgodnie z definicjami (2.27, 5.16) można wyznaczyć wartości metryki *Spójność_M* (5.19). Oba drzewa zawierają taką samą liczbę typów atrybutowych, natomiast różnią się liczbą ról.

1. Stąd dla drzewa specjalizacji (rys. 6.1) mamy na podstawie (2.27, 2.12)

$$|\text{WRole}(c1)| = |\text{Role_Bazy}(c1)|, \quad \{*\} (6.10)$$

$$|\text{WRole}(a1)| = |\text{Role_Bazy}(c1) \cup \text{Role_Bazy}(a1)| = |\text{Role_Bazy}(c1)| + |\text{Role_Bazy}(a1)|, \quad \{*\} (6.11)$$

.....

$$|\text{WRole}(an)| = |\text{Role_Bazy}(c1) \cup \text{Role_Bazy}(an)| = |\text{Role_Bazy}(c1)| + |\text{Role_Bazy}(an)|. \quad \{*\} (6.12)$$

Całkowita liczba ról w drzewie dziedziczenia jest równa (2.9, 2.27, 6.10 - 6.12))

$$\begin{aligned} \sum_{x \text{ Spec Korzen}(x)} |\text{WRole}(x)| = \\ |\text{WRole}(c1)| + |\text{WRole}(a1)| + \dots + |\text{WRole}(c1)| + |\text{WRole}(an)| = \\ (n+1) * |\text{Role_Bazy}(c1)| + |\text{Role_Bazy}(a1)| + \dots + |\text{Role_Bazy}(an)|. \end{aligned} \quad \{*\} (6.13)$$

Stąd liczba wszystkich ról w drzewie specjalizacji jest równa (6.13, 6.8., 6.9)

$$\begin{aligned} a1 = \sum_{x \text{ Spec Korzen}(x)} |\text{WRole}(x)| + \\ |\text{Role_Bazy}'(c1)| + |\text{Role_Bazy}'(a1)| + \dots + |\text{Role_Bazy}'(an)| = \\ (n+2) * |\text{Role_Bazy}(c1)| + 2 * |\text{Role_Bazy}(a1)| + \dots + 2 * |\text{Role_Bazy}(an)|. \end{aligned} \quad \{*\} (6.14)$$

Na podstawie (5.14, 2.12, 6.8, 6.9) i przy założeniu, że każdy typ faktu jest binarny, metryka połączeń jest równa

$$\begin{aligned} \text{Połączenia} = |\text{Połączenia}_W| &= |\{ \text{Role_Bazy}(c1) \cup \text{Role_Bazy}'(c1) \\ &\cup \text{Role_Bazy}(a1) \cup \text{Role_Bazy}'(a1) \cup \dots \cup \text{Role_Bazy}(an) \cup \text{Role_Bazy}'(an) \}| = \\ &2 * |\text{Role_Bazy}(a1)| + \dots + 2 * |\text{Role_Bazy}(an)| + 2 * |\text{Role_Bazy}(c1)|. \end{aligned} \quad \{*\} (6.15)$$

2. Dla drzewa polimorfizmu (rys. 6.2) mamy na podstawie (2.32, 2.12)

$$\begin{aligned} |\text{PolRole}(c2)| &= |\text{Role_Bazy}(c2) \cup \text{Role_Bazy}(b1) \cup \dots \cup \text{Role_Bazy}(bn)| = \\ &|\text{Role_Bazy}(c2)| + |\text{Role_Bazy}(b1)| + \dots + |\text{Role_Bazy}(bn)|, \end{aligned} \quad \{*\} (6.16)$$

$$|\text{PolRole}(b1)| = |\text{Role_Bazy}(b1)|, \quad \{*\} (6.17)$$

.....

$$|\text{PolRole}(bn)| = |\text{Role_Bazy}(bn)|. \quad \{*\} (6.18)$$

Całkowita liczba ról w drzewie dziedziczenia jest równa (2.9, 2.32, 6.16 - 6.18)

$$\begin{aligned} \sum_{\text{Korzen}(x) \text{ Pol } x} |\text{PolRole}(x)| &= |\text{PolRole}(c2)| + |\text{PolRole}(b1)| + \dots + |\text{PolRole}(bn)| = \\ &|\text{Role_Bazy}(c2)| + |\text{Role_Bazy}(b1)| + \dots + |\text{Role_Bazy}(bn)| + |\text{Role_Bazy}(b1)| \\ &+ \dots + |\text{Role_Bazy}(bn)| = 2 * |\text{Role_Bazy}(b1)| + \dots + 2 * |\text{Role_Bazy}(bn)| + |\text{Role_Bazy}(c2)|. \end{aligned} \quad \{*\} (6.19)$$

Stąd liczba wszystkich ról w drzewie polimorfizmu bez specjalizacji jest równa (6.19, 6.8, 6.9)

$$\begin{aligned} a2 = \sum_{\text{Korzen}(x) \text{ Pol } x} |\text{PolRole}(x)| &+ |\text{Role_Bazy}'(c2)| + |\text{Role_Bazy}'(b1)| + \dots + \\ &|\text{Role_Bazy}'(bn)| = 3 * |\text{Role_Bazy}(b1)| + \dots + 3 * |\text{Role_Bazy}(bn)| + 2 * |\text{Role_Bazy}(c2)|. \end{aligned} \quad \{*\} (6.20)$$

Na podstawie (5.14, 2.12, 6.8, 6.9) i przy założeniu, że każdy typ faktu jest binarny, metryka połączeń jest równa

$$\begin{aligned} \text{Połączenia} = |\text{Połączenia}_W| &= |\{ \text{Role_Bazy}(b1) \cup \text{Role_Bazy}'(b1) \cup \dots \cup \text{Role_Bazy}(bn) \\ &\cup \text{Role_Bazy}'(bn) \cup \text{Role_Bazy}(c2) \cup \text{Role_Bazy}'(c2) \}| = \\ &2 * |\text{Role_Bazy}(b1)| + \dots + 2 * |\text{Role_Bazy}(bn)| + 2 * |\text{Role_Bazy}(c2)|. \end{aligned} \quad \{*\} (6.21)$$

3. W przypadku zastosowania polimorfizmu (rys. 6.3) połączonego z dziedziczeniem mamy (2.37, 2.12)

$$\begin{aligned} |\text{PolRoleS}(c2)| &= |\text{Role_Bazy}(c3) \cup \text{Role_Bazy}(d1) \cup \dots \cup \text{Role_Bazy}(dn)| = \\ &|\text{Role_Bazy}(c3)| + |\text{Role_Bazy}(d1)| + \dots + |\text{Role_Bazy}(dn)|, \end{aligned} \quad \{*\} (6.22)$$

$$|\text{PolRoleS}(d1)| = |\text{Role_Bazy}(d1) \cup \text{Role_Bazy}(c3)| = |\text{Role_Bazy}(d1)| + |\text{Role_Bazy}(c3)|, \quad \{*\} (6.23)$$

.....

$$|\text{PolRoleS}(dn)| = |\text{Role_Bazy}(dn) \cup \text{Role_Bazy}(c3)| = |\text{Role_Bazy}(dn)| + |\text{Role_Bazy}(c3)|. \quad \{*\} (6.24)$$

Całkowita liczba ról w drzewie dziedziczenia jest równa (2.21, 2.37, 6.22 - 6.24)

$$\begin{aligned} \sum_{Korzen(x) Pol x} |PolRoleS(x)| &= |Role_Bazy(c3)| + |Role_Bazy(d1)| + \dots + |Role_Bazy(dn)| \\ &+ |Role_Bazy(d1)| + |Role_Bazy(c3)| + \dots + |Role_Bazy(dn)| + |Role_Bazy(c3)| \\ &= (n+1) * |Role_Bazy(c3)| + 2 * |Role_Bazy(d1)| + \dots + 2 * |Role_Bazy(dn)|. \{*\} (6.25) \end{aligned}$$

Stąd liczba wszystkich ról w drzewie polimorfizmu ze specjalizacją jest równa (6.25, 6.8, 6.9)

$$\begin{aligned} a3 &= \sum_{Korzen(x) Pol x} |PolRole(x)| + |Role_Bazy'(c3)| + |Role_Bazy'(d1)| + \dots + \\ &|Role_Bazy'(dn)| = \\ &(n+2) * |Role_Bazy(c3)| + 3 * |Role_Bazy(d1)| + \dots + 3 * |Role_Bazy(dn)|. \{*\} (6.26) \end{aligned}$$

Na podstawie (5.14, 2.12, 6.8, 6.9) i przy założeniu, że każdy typ faktu jest binarny, metryka połączeń jest równa

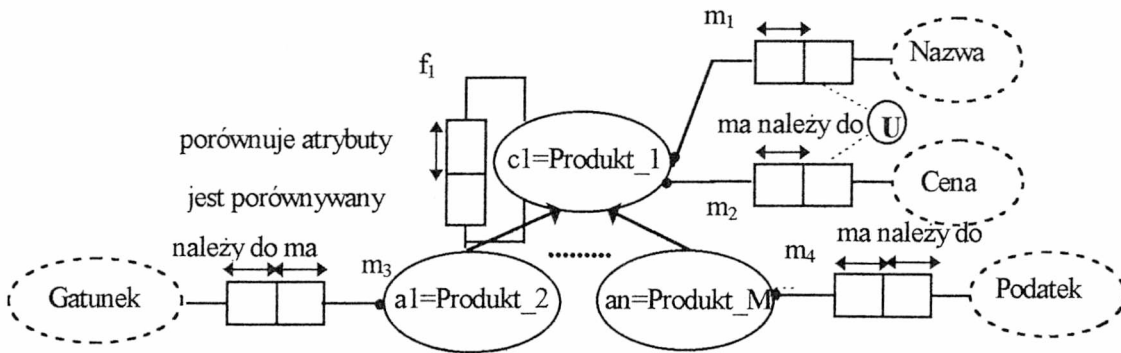
$$\begin{aligned} Połączenia &= |Połączenia W| = |\{Role_Bazy(d1) \cup Role_Bazy'(d1) \cup \dots \cup Role_Bazy(bn) \\ &\cup Role_Bazy'(bn) \cup Role_Bazy(c3) \cup Role_Bazy'(c3)\}| = \\ &2 * |Role_Bazy(d1)| + \dots + 2 * |Role_Bazy(dn)| + 2 * |Role_Bazy(c2)|. \{*\} (6.27) \end{aligned}$$

Na podstawie rozważań z rozdziału 5.2 wartość metryki *Spójność_M* zwiększa się, jeżeli rośnie liczba ról, natomiast liczba typów obiektowych i typów faktów nie ulega zmianie. Na podstawie różnicy liczby ról drzewa specjalizacji i pozostałych schematów dziedziczenia (6.14, 6.20, 6.26):

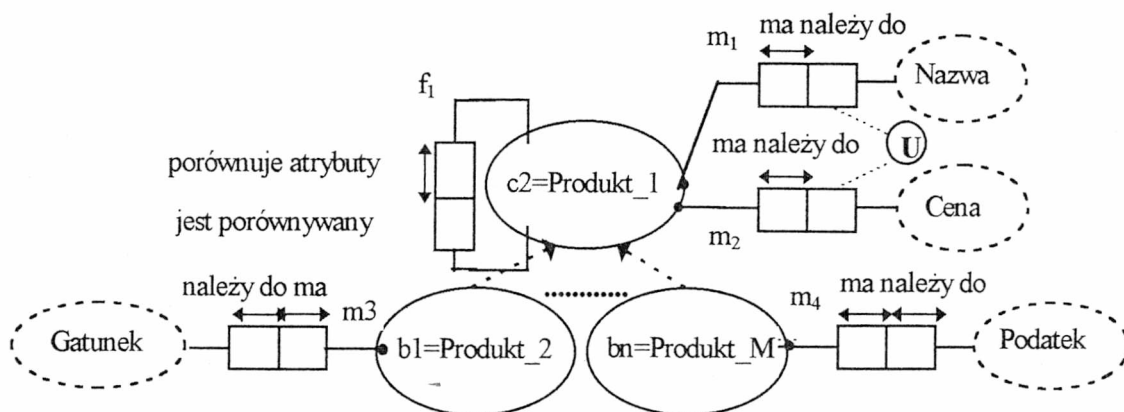
$$a) \quad a2 - a1 = |Role_Bazy(b1)| + \dots + |Role_Bazy(bn)| - n * |Role_Bazy(c1)|, \quad \{*\} (6.28)$$

$$b) \quad a3 - a1 = |Role_Bazy(d1)| + \dots + |Role_Bazy(dn)|, \quad \{*\} (6.29)$$

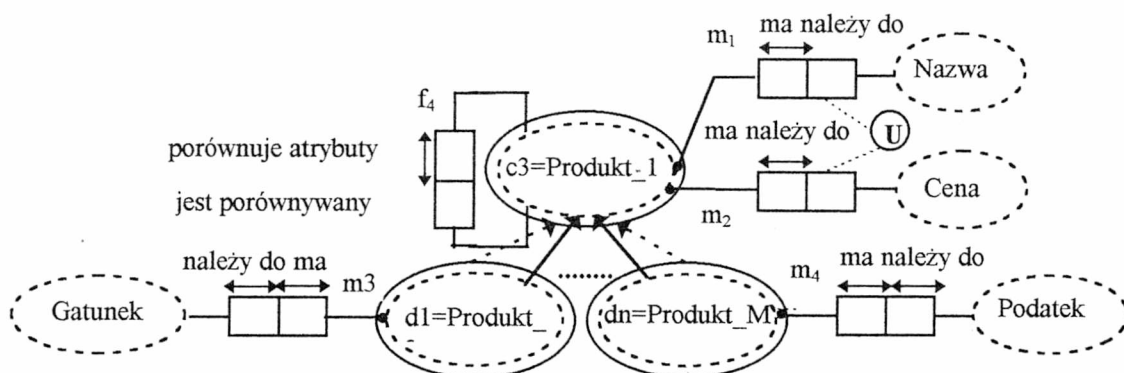
uzyskuje się w przypadku b) zawsze zwiększenie wartości metryki spójności *Spójność_M* modelu, natomiast w przypadku a) wartość tej metryki zależy od tego, czy liczba ról *Korzenia* drzewa specjalizacji nie jest większa od średniej wartości ról jego następców. We wszystkich przypadkach wartość metryki *Połączenia* jest taka sama. Stąd przypadki 1 i 3 dowodu są udowodnieniem tezy twierdzenia 6.2.



Rys. 6.1. Schemat klas oparty na relacji specjalizacji



Rys. 6.2. Schemat klas oparty na polimorfizmie bez specjalizacji



Rys. 6.3. Schemat klas oparty na polimorfizmie ze specjalizacją

Twierdzenie 6.3: Konwersja drzewa specjalizacji na drzewo polimorficzne w modelu ORM prowadzi do zmniejszenia wartości jego metryki *Połączenia* oraz zwiększenia wartości jego metryki *Spójność_M*. { * }

Dowód

Dowód przeprowadzono dla: $a1 \text{ Spec } c1, \dots, an \text{ Spec } c1$ oraz równoważnego schematu $c2 \text{ Pol } b1, \dots, c2 \text{ Pol } bn$ bez specjalizacji i ze specjalizacją $c3 \text{ Pol } d1, \dots, c3 \text{ Pol } dn$ (rys. 6.4, 6.5, 6.6).

Po zastosowaniu algorytmu z rozdziału 3.2 można udowodnić równoważność schematów z rys. 6.5-6.7 w sensie E4. Stąd możliwe jest zastąpienie w dowolnym schemacie ORM drzewa specjalizacji drzewem polimorficznym.

Dodatkowo założono taką samą liczbę typów atrybutowych i typów faktów we wszystkich schematach (6.5, 6.6, 6.7, 6.8, 6.9) oraz

$$\text{Role_Bazy}(a1) = \text{Role_Bazy}(a2) = \dots = \text{Role_Bazy}(an),$$

$$\text{Role_Bazy}(b1) = \text{Role_Bazy}(b2) = \dots = \text{Role_Bazy}(bn),$$

$$\text{Role_Bazy}(d1) = \text{Role_Bazy}(d2) = \dots = \text{Role_Bazy}(dn),$$

$$\text{Role_Bazy}(c1) = \text{Role_Bazy}(c2) = \dots = \text{Role_Bazy}(c3).$$

Każdy typ faktu jest typem binarnym.

Połączenia z pozostałą częścią schematu koncepcyjnego wynikają z reguły specjalizacji w przypadku drzewa specjalizacji oraz reguły polimorfizmu w przypadku drzewa polimorficznego ze specjalizacją i bez specjalizacji.

1. W przypadku drzewa specjalizacji (rys. 6.5) niezbędny jest minimalny zbiór następujących binarnych typów faktów (2.24, 2.12, 2.108):

$f1$ spełniający ograniczenie $Podzbiór(f1, Role_Bazy(c1))$,

$f21$ spełniający ograniczenie $Podzbiór(f21, SpecRole(a1)) \equiv Podzbiór(f21, Role_Bazy(a1))$,

.....

$f2n$ spełniający ograniczenie $Podzbiór(f2n, SpecRole(an)) \equiv Podzbiór(f2n, Role_Bazy(an))$.

Każdy z typów faktów $f1, f21, \dots, f2n$ posiada role spełniające funkcję $Role_Bazy(e)$, gdyż populacja jednego typu e spełnia wyspecyfikowane ograniczenia zbiorowe. Ponadto mamy

$$f1 \cap WRole(c1) \neq \emptyset,$$

$$f1 \cap f21 \cap WRole(a1) \neq \emptyset, \dots,$$

$$f1 \cap f2n \cap WRole(an) \neq \emptyset.$$

Liczba typów faktów wynika z liczby abstrakcyjnych typów obiektowych tworzących drzewo specjalizacji i zasady specjalizacji, czyli liczby podzbiorów ról tworzących zbiory $SpecRole$ (2.24) określających właściwości populacji tego drzewa.

Na podstawie (5.14, 5.19, 6.14, 6.15) mamy

$$a1' = a1 + \frac{n}{2} * |f1| + |f1| + |f21| + \dots + |f2n| = a1 + n + 2 + 2 * n = a1 + 3 * n + 2,$$

$$a1 = (n + 2) * 4 + 2 * n = 6 * n + 8,$$

$$x = 2 * n + 4,$$

$$y = 2 * n + 4,$$

$$\begin{aligned} Spójność_M &= \frac{2 * n + 4}{2 * n + 4} * \frac{a1 + 3 * n - (2 * n + 4) - (2 * n + 4) + 3}{(2 * n + 4)(2 * n + 4) - (2 * n + 4) - (2 * n + 4) + 1} = \\ &= \frac{5 * n + 3}{4 * n^2 + 12 * n + 9}, \end{aligned} \quad \{*\} (6.30)$$

$$Połączenia = 2 * n + 8,$$

$$\begin{aligned} Połączenia' &= Połączenia + |Role_Bazy(e)| + |Role_Bazy'(e)| = Połączenia + 2 * (n + 1) = \\ &= 2 * n + 8 + 2 * (n + 1) = 4 * n + 10. \end{aligned} \quad \{*\} (6.31)$$

2. W przypadku drzewa polimorfizmu bez specjalizacji (rys. 6.6) niezbędny jest minimalny zbiór następujących binarnych typów faktów (2.12, 2.108) $f1, f2$ spełniające ograniczenia

$Podzbiór(f1, Role_Bazy(c2))$,

$Podzbiór(f2, Role_Bazy(b1))$,

.....

$Podzbiór(f2, Role_Bazy(bn))$.

Każdy z typów faktów $f1, f2$ posiada role spełniające funkcję $Role_Bazy(e)$, gdyż populacja jednego typu obiektowego e spełnia wyspecyfikowane ograniczenia zbiorowe. Ponadto mamy $f1 \cap f2 \cap PolRole(c2) \cap PolRole(b1) \cap \dots \cap PolRole(bn) \neq \emptyset$.

Liczba typów faktów wynika z liczby abstrakcyjnych typów obiektowych tworzących drzewo polimorficzne i zasady polimorfizmu, czyli liczby podzbiorów ról tworzących zbiory $PolRole$ (2.32) określających właściwości populacji tego drzewa.

Na podstawie (5.14, 5.19, 6.20, 6.21) mamy

$$a2' = a2 + |f1| + |f2| = a2 + 4,$$

$$a2 = 3 * n + 8,$$

$$x = n + 5,$$

$$y = 2 * n + 4,$$

$$\begin{aligned} \text{Spójność}_M &= \frac{n+5}{2*n+4} * \frac{a2 - (n+5) - (2*n+4) + 5}{(n+5)(2*n+4) - (n+5) - (2*n+4) + 1} = \\ &= \frac{n+5}{n+2} * \frac{2}{2*n^2 + 11*n + 12}, \end{aligned} \quad \{*\}(6.32)$$

$$\text{Połączenia} = 2 * n + 8,$$

$$\begin{aligned} \text{Połączenia}' &= \text{Połączenia} + |Role_Bazy(e)| + |Role_Bazy'(e)| = \text{Połączenia} + 4 = \\ &= 2 * n + 12. \end{aligned} \quad \{*\}(6.33)$$

3. W przypadku drzewa polimorfizmu ze specjalizacją (rys. 6.7) niezbędny jest minimalny zbiór następujących binarnych typów faktów (2.12, 2.108): typy faktów $f1$, $f2$ spełniające ograniczenia

$$\text{Podzbiór}(f1, Role_Bazy(c3)),$$

$$\text{Podzbiór}(f2, Role_Bazy(d1)),$$

$$\dots\dots\dots,$$

$$\text{Podzbiór}(f2, Role_Bazy(dn)).$$

Każdy z typów faktów $f1$, $f2$ posiada role spełniające funkcję $Role_Bazy(e)$, gdyż populacja jednego typu e spełnia wyspecyfikowane ograniczenia zbiorowe.

Ponadto mamy $f1 \cap f2 \cap PolRoleS(c3) \cap PolRoleS(d1) \cap \dots \cap PolRoleS(dn) \neq \emptyset$.

Liczba typów faktów wynika z liczby abstrakcyjnych typów obiektowych tworzących drzewo polimorficzne i zasady polimorfizmu ze specjalizacją, czyli liczby podzbiorów ról tworzących zbiory $PolRoleS$ (2.37) określających właściwości populacji tego drzewa.

Na podstawie (5.14, 5.19, 6.26, 6.27) mamy

$$a3' = a3 + \frac{n}{2} * (|f1| + |f2|) + (|f1| + |f2|) = a3 + 2 * n + 4,$$

$$a3 = (n + 2) * 4 + 3 * n = 7 * n + 8,$$

$$x = n + 5,$$

$$y = 2 * n + 4,$$

$$\begin{aligned} \text{Spójność}_M &= \frac{n+5}{2*n+4} * \frac{a3 + 2*n - (n+5) - (2*n+4) + 5}{(n+5)(2*n+4) - (n+5) - (2*n+4) + 1} = \\ &= \frac{n+5}{n+2} * \frac{3*n+2}{2*n^2 + 11*n + 12}, \end{aligned} \quad \{*\}(6.34)$$

$$\text{Połączenia} = 2 * n + 8,$$

$$\begin{aligned} \text{Połączenia}' &= \text{Połączenia} + |Role_Bazy(e)| + |Role_Bazy'(e)| = \text{Połączenia} + 4 = \\ &= 2 * n + 12. \end{aligned} \quad \{*\}(6.35)$$

W celu sprawdzenia wartości metryk obliczenia przeprowadzono dla $n = 2$. Wyniki przedstawiono w tabeli na rys. 6.4 na podstawie (6.30, 6.31, 6.32, 6.33, 6.34, 6.35).

Wartość metryki spójności jest największa dla drzewa polimorfizmu ze specjalizacją (przypadek 3), natomiast najmniejsza dla modelu zawierającego drzewo polimorficzne bez specjalizacji. Liczby połączeń są identyczne dla modeli 2 i 3 i lepsze od modelu 1.

Dla każdego $n > 2$ również są prawdziwe te zależności:

a) różnica wartości metryki spójności i połączeń dla przypadków 3 i 1 twierdzenia 6.3

$$\frac{n+5}{n+2} * \frac{3*n+2}{2*n^2+11*n+12} - \frac{5*n+3}{4*n^2+12*n+9} = \frac{2*n^4+23*n^3+56*n^2+51*n+18}{(n+2)(2*n^2+11*n+12)(2*n+3)^2} > 0,$$

$$2 * n + 12 - 4 * n - 10 = -2 * n + 2 < 0 - \text{zawsze więcej połączeń w modelu 1,}$$

b) różnica wartości dla przypadków 2 i 1 twierdzenia 6.3:

$$\frac{n+5}{n+2} * \frac{2}{2*n^2+11*n+12} - \frac{5*n+3}{4*n^2+12*n+9} = -\frac{10*n^4+73*n^3+151*n^2+84*n-18}{(n+2)(2*n^2+11*n+12)(2*n+3)^2} < 0,$$

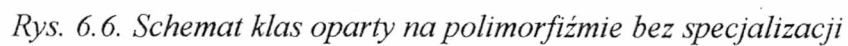
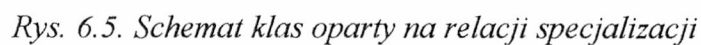
$$2 * n + 12 - 4 * n - 10 = -2 * n + 2 < 0 - \text{zawsze więcej połączeń w modelu 1.}$$

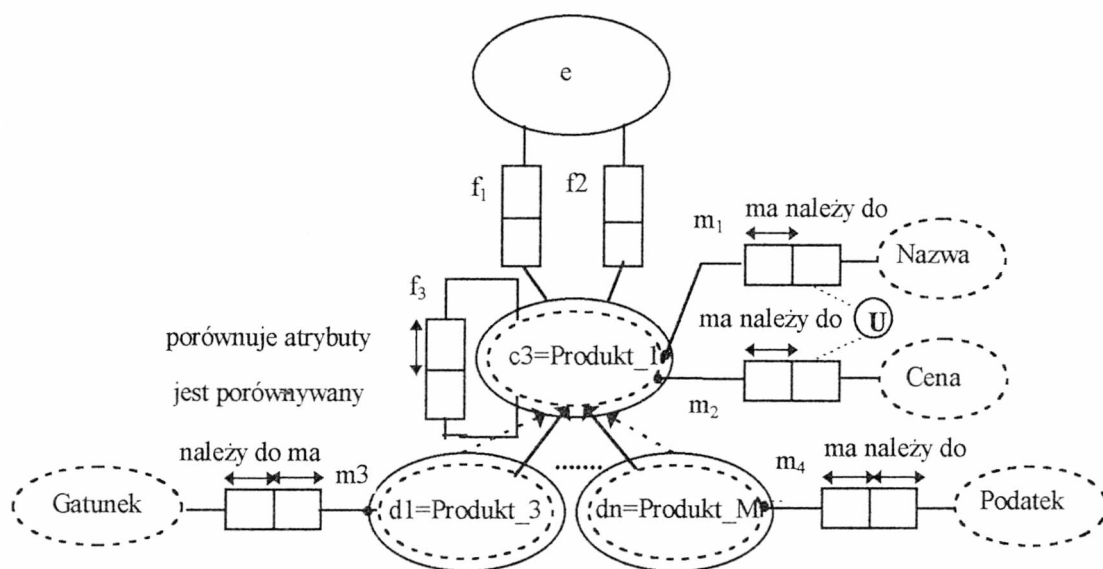
Na podstawie punktów a) i b) stwierdzono, że dla dowolnego n jedynie model zawierający drzewo polimorficzne ze specjalizacją ma lepszą wartość metryki spójności *Spójność_M* od modelu zawierającego równoważne drzewo ze specjalizacją. Model z przypadku 1) ma gorszą metrykę liczby połączeń w stosunku do równych wartości metryk połączeń z przypadku 2) i 3). Wynika z tego, że zawsze otrzymamy poprawę charakterystyk po zastosowaniu drzewa polimorficznego ze specjalizacją. Oznacza to, że przypadki 1) i 3) stanowią udowodnienie tezy twierdzenia 6.3.

Przypadek	<i>Spójność_M</i>	<i>Połączenia'</i>
1	0.265	18
2	0.083	16
3	0.333	16

Rys. 6.4. Wyniki dowodu twierdzenia 6.3 dla $n = 2$

Nie podano twierdzenia związanego ze spójnością klastrową. Z przykładów modeli badanych w rozdziale 5.4 model *M_2*, oparty na strukturach polimorficznych, miał najlepszą wartość obu metryk spójności. Opracowana metoda syntezy pomaga podjąć decyzję o wyborze rodzaju dziedziczenia analizując sposób wykorzystania rodzaju dziedziczenia tak, aby np. uniknąć rozwiązania pokazanego w modelu *M_3* badanego w rozdziale 5.4.





Rys. 6.7. Schemat klas oparty na polimorfizmie ze specjalizacją

6.2. Metody syntezy modeli ORM

Metody syntezy modeli ORM tworzą zbiór sześciu metod. Sposób ich wykonywania nawiązuje do stylu *Pinball Model*. Są one wymienione w punktach 1- 6 rozdziału 6.2 w opisie algorytmu stosowania metod syntezy modeli ORM. Zbiór wymienionych metod stanowi jeden z wyników osiągniętych w niniejszej pracy.

Przedstawiono wszystkie możliwe scenariusze użycia tych sześciu metod opracowanych w niniejszej rozprawie. Dzięki temu można w sposób automatyczny zrealizować różne postulaty z dziedziny modelowania obiektowego - konstruowania efektywnych modeli konceptualnych operujących na polimorficznych strukturach danych, transformacji schematów, oceny równoważności schematów konceptualnych, tworzenia obiektowych modeli klas, oceny złożoności strukturalnej modeli, eliminacji rozwiązań nieefektywnych wg zasady Pareta, wyboru rozwiązania dominującego ze zbioru rozwiązań efektywnych.

Użycie *metody 1* prowadzi do budowy modelu ORM w oparciu o prawidłowe konstrukcje drzew specjalizacji i polimorfizmu. Opracowaną *metodę 1* opisano w rozdziale 6.2.1. W *metodzie 1* wykorzystano własności konwersji struktur specjalizacji i polimorfizmu (twierdzenia 6.1-6.3). Po zastosowaniu tej metody można zakończyć syntezę modelu, jeżeli nie zostały postawione inne cele. W przypadku otrzymania zbioru rozwiązań w postaci modeli można kontynuować syntezę w celu sprawdzenia równoważności schematów tych modeli za pomocą *metody 2*. Umożliwi to wyeliminowanie niewłaściwych rozwiązań. Użycie *metody 2* sprawdzania równoważności w sensie E4 może nastąpić po wykonaniu modelu, czyli zastosowaniu *metody 1*.

Użycie *metody 3*, służącej do budowy klastrowej modelu ORM musi nastąpić po zastosowaniu *metody 1* i może być zrealizowane niezależnie od użycia pozostałych metod. *Metodę 3* stosuje się w dwóch głównych przypadkach. W pierwszym przypadku *metoda 3* może być wykorzystana do wykonania obiektowej specyfikacji modelu statycznego. Wtedy zastosowanie *metody 3* może zakończyć syntezę modelu ORM. W drugim przypadku, oprócz obiektowej specyfikacji modelu, stosowanie modelu klastrowego daje

możliwość oceny poszczególnych „cegiełek” modelu jako modeli klas. W modelu klastrowym można wyodrębnić mniej efektywne elementy i dokonać korekty. Stąd należy użyć *metody 3* w drugim przypadku, gdy zachodzi potrzeba wyznaczenia kryteriów oceny charakterystyk wewnętrznych rozwiązań za pomocą *metody 4*, ocenianych również z poziomu klas.

W przypadku, gdy należy wyznaczyć miary charakterystyk wewnętrznych rozwiązań równoważnych w sensie E4, należy zastosować *metodę 4* określającą charakterystyki wewnętrzne modelu za pomocą metryk spójności i połączeniowej. Użycie *metody 4* nie musi następować jedynie po zastosowaniu *metody 2*, lecz również bezpośrednio po zastosowaniu *metody 1* lub/i *metody 3*.

W przypadku, gdy należy wyeliminować ze zbioru rozwiązań wszystkie nieefektywne rozwiązania, należy użyć *metody 5*. Zastosowanie jej musi być poprzedzone wyznaczeniem wartości funkcji kryterialnych za pomocą *metody 4*. Model bez podziału na klastry daje możliwość oceny grupy klas tworzących model modułu lub systemu, natomiast model klastrowy daje możliwość oceny modelu z poziomu klas. W czasie syntezy należy badać obie formy modelu programu dążąc do wyboru jak najefektywniejszego w każdej postaci. Stąd *metoda 5* musi być zawsze stosowana po wykorzystaniu *metody 4*, która z kolei powinna wtedy być poprzedzona zastosowaniem *metody 2* i *metody 3*.

W przypadku, gdy należy wybrać jedno rozwiązanie ze zbioru rozwiązań efektywnych, należy użyć *metody 6*, która powinna być zastosowana po wykorzystaniu *metody 5*.

Metody syntezy modeli ORM są wykonywane wg różnych scenariuszy, które powinny być realizowane za pośrednictwem metod wymienionych w punktach 1 - 6. Scenariusze przedstawiono w opisie algorytmu wykonywania metod syntezy modeli ORM.

Algorytm wykonywania metod syntezy modeli ORM

1. Zastosowanie *metody 1* skupiającej dwie metody iteracyjnej budowy modelu zgodnie z semantyką aplikacji, którą opracowano w niniejszej pracy (rozdział 6.2.1).
 - 1.1. Użycie metody opartej na *algorytmie wyznaczania poszczególnych rodzin typu specjalizacja*. Prowadzi ona do tworzenia grupy obiektów opartych na specjalizacji na podstawie powiązań między członkami tych grup. Każdy pojedynczy abstrakcyjny typ obiektowy jest traktowany jako drzewo specjalizacji z jednym elementem, który jest *Korzeniem* drzewa dziedziczenia. W zależności od struktury powiązań należy dokonać wyboru właściwej struktury specjalizacji prowadzącej model do większej spójności i mniejszej liczby połączeń.
 - 1.2. Użycie metody opartej na *algorytmie wyznaczania poszczególnych rodzin typu polimorfizm*. W metodzie tej na podstawie powiązań między drzewami utworzonymi w poprzednim kroku poszukuje się przypadków, w których celowe jest tworzenie grupy obiektów opartych na polimorficznych strukturach danych lub przekształcanie drzew specjalizacji na drzewa polimorficzne ze specjalizacją na podstawie powiązań między tymi drzewami. W zależności od struktury powiązań należy dokonać wyboru właściwej struktury polimorficznej prowadzącej model do większej spójności i mniejszej liczby połączeń.
 - 1.3. Przejście do zastosowania *metody 2* lub *metody 3* lub *metody 4* lub zakończenie syntezy modelu.

2. W *metodzie 2* w przypadku uzyskania wielu rozwiązań w postaci modeli ORM należy zbadać ich równoważność w sensie E4. Czynności te należy realizować za pomocą metody opartej algorytmie badania równoważności opracowanego w rozdziale 3.2. Pozwala to określić zbiór rozwiązań problemu z dziedziny aplikacji. Dalsze działania mogą kontynuowane od zastosowania *metody 3* lub *metody 4* lub można zakończyć syntezę modelu ORM.
3. W *metodzie 3* opartej na algorytmie podziału na klastry modelu ORM wyznacza się klasy metodą opisaną w rozdziale 4.2. Po zakończeniu metody można zakończyć proces syntezy modelu ORM lub kontynuować syntezę od użycia *metody 2* lub *metody 4*.
4. W *metodzie 4* dla zbioru rozwiązań w postaci modeli ORM (po zastosowaniu jedynie *metody 1*), a szczególnie zbioru rozwiązań równoważnych w sensie E4 (po wykonaniu *metody 2*) należy wyznaczyć charakterystyki wewnętrzne. Czynności te należy realizować za pomocą *metody 4* opartej na algorytmie określania charakterystyk wewnętrznych przedstawionym w rozdziale 5.3. Pozwala ona wyznaczyć wartości funkcji kryterialnych całego modelu i oraz jego poszczególnych klas z modelu klastrowego: *Spójność_M*, *Spójność_kl*, *Połączenia*. Wyznaczenie funkcji kryterialnej *Spójność_kl* jest możliwe po użyciu *metody 3* przed zastosowaniem *metody 4*. Dalszą syntezę można kontynuować od użycia *metody 5* lub można zakończyć działania.
5. W *metodzie 5* dokonuje się wyboru zbioru rozwiązań efektywnych za pomocą metody Pareta. Metoda ta została opisana w rozdziale 6.2.2. Wykonanie tej metody, opartej na wyborze trójkryterialnym, powinno być poprzedzone wykorzystaniem *metody 1*, *metody 2*, *metody 3* i *metody 4*. Dalszą syntezę można kontynuować od zastosowania *metody 6* lub można zakończyć działania.
6. W *metodzie 6* dokonuje się wyboru jednego rozwiązania po wykonaniu *metody 5*. Jeżeli w *metodzie 5* nie wyodrębniono rozwiązania dominującego, dokonuje się tego zadania za pomocą metody kompromisowej opisanej w rozdziale 6.2.2.

W algorytmie syntezy modeli ORM podstawową rolę ogrywa *metoda 1*. Algorytm tej metody oparto na twierdzeniach 6.1 - 6.3. *Metoda 1* została opracowana w rozdz. 6.2.1. W metodzie tej na podstawie semantyki aplikacji można zbudować modele ORM. Dzięki tej metodzie można ograniczyć zbiór rozwiązań nieefektywnych rozpatrywanych podczas realizacji algorytmu syntezy modeli ORM (*metody 1 - 6*). Opracowano taką metodę, która ogranicza liczbę abstrakcyjnych typów obiektowych i pozwala podjąć decyzję o właściwym wyborze modelu dziedziczenia i prowadzi do uzyskania efektywnego modelu.

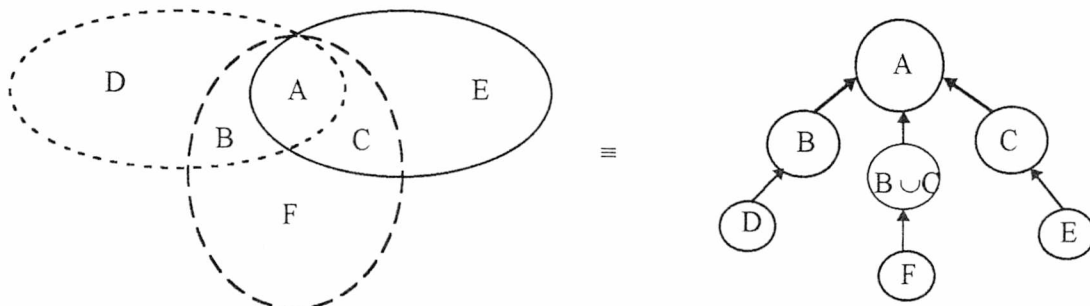
6.2.1. Iteracyjna budowa modelu opartego na polimorficznych strukturach danych

Tok postępowania

1. Wstępna budowa modelu.
Ustalenie struktury drzew dziedziczenia dokonuje się podczas realizacji algorytmu metody syntezy modelu, stąd wszystkie abstrakcyjne typy obiektowe są tworzone na początku bez wykorzystania specjalizacji i polimorfizmu.

2. W początkowej fazie tworzenia modelu zbiory ról i typów obiektowych związanych takimi samymi własnościami identyfikacji strukturalnej należy powiązać z jednym typem abstrakcyjnym pełniącym rolę typu (2.9) $Korzeń(x) = x$. Tak wydzielony zbiór typów atrybutowych i typu abstrakcyjnego będzie nazywać rodziną. Odpowiada to utworzeniu klastra opisanego zależnością $Klastra(0,0,x)$ (4.2).
3. W celu stworzenia struktur drzew specjalizacji należy wykonać następujące czynności:
 - 3.1. Wyróżnić rodzinę podzbiorów ról objętych ograniczeniami zbiorowymi typu *Równość* (2.110), *Podzbiór* (2.108), sekwencji ról *Zbior* (2.106) $\sigma_{F_1} = TRol[1], \dots, \sigma_{F_n} = TRol[n]$ i umieścić w tablicy *TRol*.
 - 3.2. Wyznaczyć w tablicy *Spec_T* zbiór ról należących do *Korzenia* następująco: $Spec_T[0,1] = TRol[1] \cap \dots \cap TRol[n] \neq \emptyset = \{r \in (\cap_{i \in Zb} TRol_i) \neq \emptyset\}$. Jeżeli istnieje rozłączny zbiór ról, oznacza to brak powiązań dziedziczenia w danym klastrze, czyli błędny podział na klastry. W tablicy *Zb* znajduje się zbiór numerów zbiorów ról, które są indeksami tablicy *TRol*. Dla elementu typu *Korzeń* mamy $Zb[0,1] = \{i\}$, gdzie $i = 1 \dots n$.
 - 3.3. Odjąć od każdego zbioru *TRol[i]* zbiór *Spec_T[0,1]*. Każdy pusty zbiór ról jest wykluczony z dalszej analizy, gdyż nie jest pamiętany w tablicy *Zb*, czyli $Zb[0,1] := Zb[0,1] - [i]$.
 - 3.4. Wyodrębnić podtypy na poziomie *k*-tym w postaci niepustych iloczynów $Spec_T[k,y]$, gdzie $y = 1 \dots Ilość[k]$ podzbiorów ról z tablicy *TRol*. Wyznaczają one *x1* następców *j*-tego nadtypu *k*-tego poziomu. Liczbę podtypów należy przechować w *k*-tym elemencie tablicy *Ilość*.
 - 3.5. Dla każdego nadtypu $Spec_T[k, i]$ sprawdzić, czy podzbiory $TRol[j]$ nie należą do kilku nadtypów. Jeżeli dany zbiór ról należy do kilku nadtypów, odpowiada to przypadkowi dziedziczenia wielobazowego, który nie jest realizowany w schematach ORM. Wtedy należy utworzyć nowy nadtyp $Spec_T[k, x1]$ jako sumę podzbiorów $Spec_T[k, i], \dots \cup Spec_T[k, i+m]$.
 - 3.6. Odjąć te iloczyny od wyznaczających je podzbiorów ról, czyli wyznaczyć podzbiory $TRol[k]$, gdzie $(i=1..n)$. Każdy pusty zbiór ról jest wykluczony z dalszej analizy, gdyż jest usuwany z tablicy *Zb*, czyli $Zb[k, j] := Zb[k, j] - [i]$.
 - 3.7. Proces ten powtarzać (kroki 3.4 - 3.6) dla *Ilość[k]* podzbiorów, które są niepustymi iloczynami z poprzedniego kroku (poziom *k-1*) aż do otrzymania zbiorów ról rozłącznych. Są to *Liście* drzewa dziedziczenia połączone z iloczynem niepustym z poprzedniego kroku.

Sposób wyodrębniania typów obiektowych na kolejnych poziomach dziedziczenia został przedstawiony na rys. 6.8.



Rys. 6.8. Budowa drzewa specjalizacji

Algorytm wyznaczania poszczególnych rodzin typu specjalizacja

- dane wejściowe

$\sigma_{F_1} = \text{TRol}[1] \text{ set of Rol}, \dots, \sigma_{F_n} = \text{TRol}[n] \text{ set of Rol}$, gdzie

Rol = **record**

r_Fakt : TFakt; {numer typ faktu, do którego należy rola}

r_Baza: TOb; {numer typu bazowego Korzenia rodziny}

r_Lista : TOb;; {numer Korzeni rodzin, z którym połączona jest rola}

end;

Spec = **array** [1..n.1..m] **of set of Rol**.

- inicjalizacja

{tworzenie zbioru ról Spec_T[0,1] dla typu Korzeń}

Zb[1] := []; Spec_T[0,1] := []; nr_drzewa := a;

for i:=1 **to** n **do** Spec_T[0,1] := Spec_T[0,1] $\cup$ TRol[i];

{jeśli Spec_T[0,1] jest zbiorem pustym, to wystąpił błąd w wyborze algorytmu}

if Spec_T[0,1] = $\emptyset$ **then** koniec

{tworzenie podzbiorów ról dla 1-poziomu dziedziczenia}

for i:=1 **to** n **do begin**

TRol[i] := TRol[i] - Spec_T[0,1];

if TRol[i] $\neq \emptyset$ **then** Zb[1, i] := Zb[1, i] + [i];

end;

- wyodrębnianie typów na poszczególnych poziomach dziedziczenia

k := 1; Ilość[k] := 1; y := 1;

repeat

while y $\leq$ Ilość[k-1] **do begin** {wyznacz dla każdego podtypu z pośród}

x1 := 0; {Ilość[k-1] na poziomie k-1 jego podtypy na poziomie k}

for i := 1 **to** n-1 **do** {nowy numer x1 podtypu na poziomie k dla zbioru Zb[k-1], y nadtypu Spec_T[k-1, y]}

if i in Zb[k-1, y] **then begin**

x1 := x1 + 1; Zb[k, x1] := [i]; Spec_T[k, x1] := TRol[i];

for j := i+1 **to** n **do**

if j in Zb[k-1, y] **then begin**

Spec_T[k, x1] = Spec_T[k, x1] $\cup$ TRol[j];

```

        if Spec_T[k, x1]  $\neq \emptyset$  then Zb[k, x1] := Zb[k, x1] + [j];
    end;
end;
{tworzenie nowych podtypów na danym poziomie dziedziczenia w celu zastąpienia
dziedziczenia wielobazowego - tworzenie nowego podtypu jako sumy tych podty-
pów Spec_T[k, i], do których należy rola TRol[j]}
for j := 1 to n do
    if j in Zb[k-1, y] then begin
        Pom1 := []; Pom2 := []; i := 1; x := 0;
        while i <= x1 do begin
            if TRol[j]  $\cap$  Spec_T[k, i]  $\neq \emptyset$  then begin
                x:=x+1; Pom1:= Pom1 $\cup$ Spec_T[k, i]; Pom2:=Pom2 + [j];
                if x = 1 then Pom3 := i;
                if x > 1 then begin
                    {wykluczenie numerów ról ze zbiorów numerów podtypów, do których
należał dotąd zbiór ról TRol[j]}
                    Zb[k, i] := Zb[k, i] - [j]; Zb[k, Pom3] := Zb[k, Pom3] - [j]
                end;
            end;
            i := i + 1;
        end;
        if x > 1 then begin
            {utworzenie podtypu jako sumy ról podtypów obliczonych w Pom1}
            x1 := x1 + 1; Spec_T[k, x1] := Pom1; Zb[k, x1] := Pom2;
            for r in Spec_T[k, x1] do {zapamiętanie numeru typu bazowego roli r}
                if r.r_Baza <> nr_drzewa then r.r_Baza:=x1;
            end;
            TRol[j]:=TRol[j]-Pom1; {wyznaczenie nowej zawartości zbiorów TRol[j]}
            if TRol[j] =  $\emptyset$  then Zb[k, x1] := Zb[k, x1] - [j];
        end;
        y := y + 1; Pom4 := Pom4 + x1; {nowa liczba podtypów na k-tym poziomie}
    end;
    Ilość[k] := Pom4; k := k + 1; y := y + 1;
until  $\cup_p^n \text{TRol}[p] = \emptyset$ ;

```

4. Związki $n:\dots:m$ lub $1:\dots:n$ należy realizować za pomocą typów sekwencyjnych i typów zbiorowych (rys. 4.3). Elementy sekwencji należące do drzew dziedziczenia powinny być w relacji polimorfizmu [59, 60]. W przeciwnym wypadku każdy element drzewa powinien należeć do różnych sekwencji lub typów zbiorowych. Wynika to z aksjomatów A27-A31. Relacja specjalizacji uniemożliwia spełnienie aksjomatu A29 w przypadku jednego typu sekwencyjnego dla całego drzewa specjalizacji, gdyż musiałaby być prawdziwy przypadek, że populacja typu sekwencyjnego jest podzbiorem populacji elementu sekwencji, który jest podtypem, co jest niemożliwe do spełnienia.
5. Po utworzeniu drzew specjalizacji wg p. 3 algorytmu należy zgodnie z punktem 1 scenariusza działań z rozdziału 6.2 sprawdzić, czy istnieją podstawy do utworzenia

polimorficznych struktur danych lub przekształcenia drzew specjalizacji na drzewa polimorficzne ze specjalizacją. W tym celu w każdym z drzew specjalizacji bada się zbiory ról objęte ograniczeniami zbiorowymi, które wiążą role z różnych drzew dziedzinienia.

5.1. Jeżeli w kilku takich zbiorach istnieje ten sam podzbiór ról z jednego drzewa A związany z różnymi podzbiorami ról należącymi do innego drzewa B, to należy to drzewo B przekształcić na drzewo polimorficzne. Wtedy wybiera się taki nadtyp w drzewie B, który będzie posiadał zbiór *PolRoleS* zawierający wszystkie podzbiory należące do tych ograniczeń. Dane potrzebne do podjęcia takiej decyzji przechowywane są w każdym drzewie specjalizacji utworzonym w kroku 3 metody syntezy modelu ORM. W każdej z tych n rodzin znajduje się $m_1, \dots, m_n$ podtypów opisanych zbiorami *Spec_T*. Każda rola w zbiorze *Spec_T* przechowuje informację o swoim typie bazowym (w polu *r_Baza*), który może należeć do innego drzewa specjalizacji. W celu wyodrębniania rodzin polimorficznych ze specjalizacją należy wykonywać następujące kroki algorytmu.

5.1.1. Dla każdej wieloelementowej rodziny i sprawdź jej zbiory *Spec_T*. Jeżeli istnieje rola r' należąca do innej i' rodziny, to usuń ją ze zbioru *Spec_T* i zapamiętaj ją na liście *Zb[i].Lista*. Wielokrotne wystąpienie tej samej roli w różnych zbiorach *Spec_T* w tym samym drzewie i określa warunki do przekształcenia go na drzewo polimorficzne. Typ bazowy roli należącej do tego samego typu faktu, do którego należy rola r' , jest określony jako *Korzeń* drzewa polimorficznego.

5.1.2. Pozostałe role, które należą do innych drzew, lecz wystąpiły tylko raz, mogą zmodyfikować typ bazowy roli, spełniając zależność *Inne_Role(r')* (2.11) do typu odpowiadającego aktualnemu zbiorowi *Spec_T*. Wszystkie role r' są w swoim drzewie zaznaczone jako *odwiedzone*, co jest wyrażone wartością „0” wpisaną w pole *R_Lista*. Jeżeli dana rola r' jest już zaznaczona, wystąpił błąd aplikacji.

5.2. W przypadku, gdy dany zbiór ról jednej rodziny A jest powiązany z różnymi rolami różnych rodzin jednoelementowych B, C... tworzy się rodzinę polimorficzną bez specjalizacji. Wtedy *Liściami* takiego drzewa są te typy B, C... natomiast ich *Korzeń* posiada zbiór ról *PolRole*. Populacja *Korzenia* jest sumą populacji poszczególnych liści B, C...

5.2.1. Dla każdej jednoelementowej rodziny i sprawdź jej zbiory *Spec_T*. Jeżeli istnieje rola r' należąca do innej i' rodziny, to usuń ją ze zbioru *Spec_T* i zapamiętaj na jej liście $r'.r_Lista$ numer drzewa i oraz numer drzewa i' w zbiorze *Zb_1*. Wielokrotne wystąpienie tej samej roli w różnych drzewach i (różnych zbiorach *Spec_T*) określa warunki do przekształcenia tych drzew na *Liście* drzewa polimorficznego. Typ bazowy roli należącej do tego samego typu faktu, do którego należy rola r' , jest określony jako *Korzeń* drzewa polimorficznego bez specjalizacji.

5.2.2. Pozostałe role, które należą do innych drzew, lecz wystąpiły tylko raz, mogą zmodyfikować typ bazowy roli, spełniając zależność *Inne_Role(r')* (2.11) do typu odpowiadającego aktualnemu zbiorowi

Spec_T. Wszystkie role r' są w swoim drzewie zaznaczone jako *odwiedzone*, co jest wyrażone wartością „0” wpisaną w pole *R_Lista*. Jeżeli dana rola r' jest już zaznaczona, wystąpił błąd aplikacji.

5.3. Powtarzać kroki 5.1 i 5.2 aż do wyczerpania zbiorów ról w poszczególnych rodzinach.

Algorytm wyznaczania poszczególnych rodzin typu polimorfizm

- dane wejściowe: rodziny zbiorów *Spec*, odpowiadające drzewom specjalizacji. Zbiory *Spec* są zbiorami ról reprezentujących typy obiektowe należące do drzew specjalizacji.

$\sigma_{F1} = \{\text{Spec}[1, a_1], \dots, \text{Spec}[k_1, m_1]\}$, $\text{Zb}[1] := \text{nil}$;

$\sigma_{F2} = \{\text{Spec}[2, a_2], \dots, \text{Spec}[k_2, m_2]\}$, $\text{Zb}[2] := \text{nil}$;

$\sigma_{Fn} = \{\text{Spec}[n, a_n], \dots, \text{Spec}[k_n, m_n]\}$, $\text{Zb}[n] := \text{nil}$;

$\text{Zb_1} := []$;

- wyznaczanie danych dla drzew polimorficznych ze specjalizacją i bez specjalizacji

```

for i := 1 to n do begin    {dla każdej rodziny i}
  for p := 1 to ki do begin {dla każdego poziomu ki rodziny i}
    for j := 1 to mi do {dla każdego typu mi obiektowego poziomu kj}
      for r ∈ Spec[p, j] do {dla każdej roli r ze zbioru ról węzła drzewa i}
        if r.r_Baza <> i then begin {jeśli rola r nie należy do drzewa i}
          i' = r.r_Baza;
          for p' := 1 to kp do
            for j' := 1 to mp do
              for r' ∈ Spec[p', j'] do
                if r.r_Fakt = r'.r_Fakt then {poszukiwanie roli r' w drzewie i'}
                  if r'.r_Lista = 0 then Błąd
                else
                  if ( p=1) and (p'=1) then begin {dane dla drzewa Pol}
                    Dodaj(r'.r_Lista, i); Zb_1 := Zb_1+[i'];
                  end
                  else r'.r_Lista := 0 {odwiedzenie roli r' w rodzinie i' - drzewo PolS}
                end;
              Spec[p, i] := Spec[p, i] - r';
            if (p>1) and (p'>1) then Dodaj(Zb[i], r');
          end;
        Spec[p, i] := Spec[p, i] - r;
      if (p>1) and (p'>1) then Dodaj(Zb[i], r');
    end;
  end;

```

- tworzenie drzew polimorficznych

```

for i:=1 to n do
  for r:= Następny_element_listy(Zb[i]) to nil do
    if Liczba_wystapien(k, Zb[i]) > 1 then begin {dla każdego wystąpienia roli k większego od 1 dla rodziny i}
      Przekształć_DrzewoSpec_na_PolSpec(i);
      Inne_Role(r).r_Baza:= Korzen(i); {nowy typ bazowy dla roli z tego samego typu
    end;                                     {faktu, do którego należy rola r' z części „b” algorytmu}

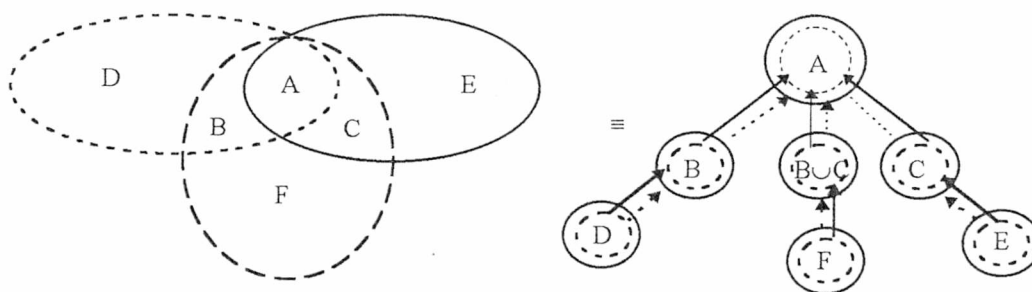
```

```

for i := 1 to n do
  if i in Zb_1 do {dla jednoelementowego drzewa i posiadającego role związane z in-
    nymi jednoelementowymi drzewami utwórz z listy drzew r.r_Lista
    Liście w drzewie Pol}
  for r ∈ Spec[1,1] do
    if r.r_Lista <> nil then begin
      Utwórz_DrzewoPol(r.r_Lista);
      Inne_Role(r).r_Baza := Korzen(i);
    end;
  end;
end;

```

Sposób wyodrębniania typów obiektowych na kolejnych poziomach dziedziczenia zostało przedstawione na rys. 6.9.



Rys. 6.9. Budowa drzewa polimorficznego ze specjalizacją

6.2.2. Wybór najlepszego rozwiązania

Problem wyboru najlepszego z rozwiązań w postaci modelu ORM, równoważnych w sensie E4, jest zaliczany do klasy problemów optymalizacji wielokryterialnej [9, 18, 104, 113] jako szczególny przypadek problematyki wyboru. Przy braku rozwiązania dominującego należy sprowadzić problem do klasy problemów optymalizacji jednokryterialnej. Problem wielokryterialnego wyboru efektywnego rozwiązania można sprowadzić do problemu wyboru najlepszego rozwiązania metodą Pareta. Wybór dominującego rozwiązania ze zbioru rozwiązań optymalnych wg Pareta można zrealizować z pomocą jednokryterialnej metody rozwiązań kompromisowych.

Formalne definicje istniejące w problemie wyboru podano na podstawie [113].

Dana jest przestrzeń n - wymiarowa rozwiązań dopuszczalnych $T \subset N^n$, gdzie N oznacza zbiór liczb naturalnych. Obrazem w przestrzeni kryteriów zbioru rozwiązań dopuszczalnych poprzez odwzorowanie $F(t)$ jest

$$Z_T = \{F(t): t \in T\} \subset R^s \quad (6.36)$$

gdzie R oznacza zbiór wszystkich liczb rzeczywistych.

Wówczas mamy

$$F_i(t') > F_i(t'') \Rightarrow t' \text{ przewyższa rozwiązanie } t''. \quad (6.37)$$

Rozwiązanie optymalne lub rozwiązanie absolutnie optymalne $\bar{t}_r$ względem r -tej funkcji kryterianej jest zdefiniowane jako

$$\max_{t \in T} F_r(t) = F_r(\bar{t}_r) = \bar{z}_r, r = 1, \dots, s. \quad (6.38)$$

Rozwiązanie idealne w przestrzeni kryteriów jest zdefiniowane jako

$$\bar{z} = [\bar{z}_1, \bar{z}_2, \dots, \bar{z}_s] \in R^s. \quad (6.39)$$

Wektor $\bar{t} \in R^n$ taki, że $F(\bar{t}) = \bar{z}$ jest rozwiązaniem idealnym w przestrzeni rozwiązań.

$$(6.40)$$

Wystąpienie rozwiązania idealnego jest szczególnym przypadkiem w problematyce wyboru.

Wg [113] problematyka wyboru w przypadku istnienia zbioru rozwiązań polega na postawieniu problemu w kategoriach wyboru jednego najlepszego rozwiązania (wariantu). Oznacza to skierowanie badań na poszukiwanie podzbioru rozwiązań (możliwie jak najmniej liczne) ze zbioru wszystkich rozwiązań w celu wyboru:

- zalecanego rozwiązania z maksymalną ścisłością i dokładnością,
- metodyki opartej na procedurze selekcji najlepszego rozwiązania nadającej się do wielokrotnego i automatycznego zastosowania.

Jako metodę selekcji podzbioru najlepszych rozwiązań ze zbioru wszystkich rozwiązań przyjęto metodę Pareta. Mówi ona, że dla danego zbioru funkcji kryterialnych rozwiązanie jest efektywne, jeżeli nie istnieje inne rozwiązanie, w którym poprawienie wartości jednej funkcji kryterialnej nie pogorszyłoby wartości innej funkcji kryterialnej.

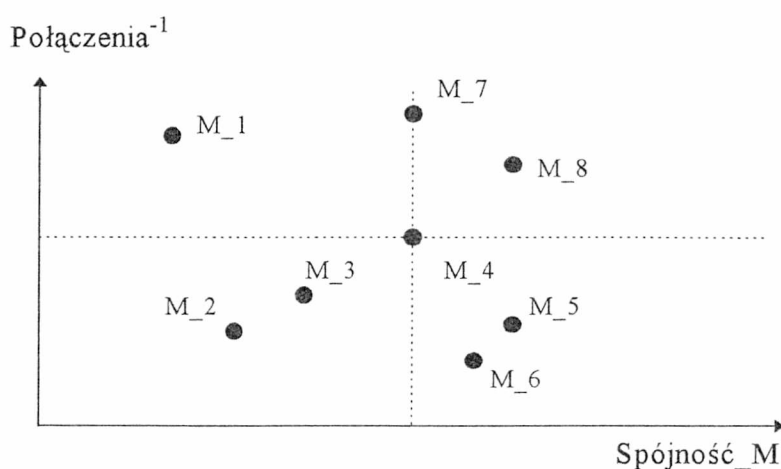
Formalna definicja zasady efektywności wg Pareta jest następująca:

Rozwiązanie $\hat{t} \in T$ nazywamy rozwiązaniem optymalnym według Pareta wtedy i tylko wtedy gdy nie istnieje $t \in T$ taki, że $F(t) \geq F(\hat{t})$ i $F(t) \neq F(\hat{t})$. (6.41)

Definicja efektywności w sensie Pareta w odniesieniu do oprogramowania jest następująca:

Dla danego zbioru funkcji kryterialnych, ograniczeń z dziedziny aplikacji oraz możliwości syntaktycznych i semantycznych modelowania ORM rozwiązanie jest efektywne w sensie Pareta, jeśli nie istnieje inne rozwiązanie, w którym polepszenie jednego kryterium nie pogorszyłoby innego.

Interpretację tej definicji przedstawia rysunek 6.10. Przyjęto metryki *Spójność_M* oraz *Połączenia⁻¹* jako funkcje kryterialne. Zgodnie z definicją rozwiązania M_7 i M_8, w postaci modeli, są optymalne wg Pareta, gdyż wybór rozwiązania M_8 zamiast M_7 prowadzi do pogorszenia wartości metryki *Połączenia* i daje poprawę metryki *Spójność_M*. Pozostałe rozwiązania nie spełniają kryterium Pareta w stosunku do punktów M_7 i M_8, gdyż można poprawić w dowolnym z nich wartości obu kryteriów. Podobnie odbywałby się wybór rozwiązań optymalnych wg Pareta dla przestrzeni trójkryterialnej: *Spójność_M*, *Spójność_kl*, *Połączenia*.



Rys. 6.10. Diagram rozwiązań M_x optymalizowany za pomocą metody Pareta

Zastosowanie zasady Pareta do wyboru najefektywniejszego modelu jest sformułowane jako zadanie wyboru:

- 1) takiego rozwiązania $t = (x, y, a, k, kl)$, w którym wartości spójności klastrowej $Spójność_{kl}$ oraz spójności całego modelu $Spójność_M$ są największe oraz liczba połączeń $Połączenia_W$ jest najmniejsza w porównaniu z odpowiadającymi wartościami pozostałych modeli,
- 2) lub zbioru rozwiązań, w których wartości tych metryk są równe,
- 3) lub zbioru rozwiązań, w których dowolne z tych metryk mają lepsze wartości od pozostałych i wybór któregośkolwiek z tych rozwiązań prowadzi do rozwiązania z gorszymi wartościami pozostałych metryk. W tym przypadku braku jednoznacznego rozwiązania wybiera się jedno z nich metodą kompromisową.

Dane metody wyboru wg Pareta są następujące:

1) Przestrzeń rozwiązań dopuszczalnych $t = (x, y, a, k, kl) \in T$ i $T \subset N^n$.

2) Zbiór kryteriów $\{F(t)\} \subset R^n$:

$$F_1(t) = Spójność_M(x, y, a, k) = \max_M,$$

$$F_2(t) = Spójność_{kl}(x, y, a, k, kl) = \max_{kl},$$

$$F_3(t) = Połączenia(x, y, p) = \min.$$

3) Zbiór ograniczeń:

$$k = 1: \max_M \leq \frac{x}{y}, \max_M \geq 0, \min \geq x + y - 1, \min \leq x * y,$$

$$k \geq 2: \max_M \leq 0, \max_M \geq -1, \min \geq x + y - k,$$

$$\max_{kl} \leq \max_M, \max_{kl} \geq \frac{-1}{kl}, kl < y.$$

Wartości poszczególnych zmiennych wynikają z doboru elementów i ograniczeń w schemacie koncepcyjnym ze względu na semantykę aplikacji.

W tabeli na rys. 6.11 podano trzy rozwiązania wraz z wartościami funkcji kryterialnych. Rozwiązaniem optymalnym jest model M_2 , gdyż spełnia warunek 1 metody Pareta zdefiniowanej dla modeli ORM.

Dane SI modelu										
	Rozwiązanie t							Wartości funkcji kryterialnych $F_i(t)$		
Numer modelu	x	y	z	a	k	kl	p	$Spójność_M$	$Spójność_kl$	$Połączenia$
M_1	20	13	6	46	1	4	40	0.074	- 0.3	40
M_2	20	16	6	64	1	7	40	0.101	- 0.293	40
M_3	25	21	6	80	1	11	50	0.072	- 0.324	50

Rys 6.11. Tabela danych modeli M_1, M_2, M_3 tej samej aplikacji

W przypadku gdy nie istnieje rozwiązanie optymalne, należy wybrać *rozwiązanie kompromisowe*, które jest rozwiązaniem sprawnym bliskim, według przyjętej miary odległości, rozwiązaniu idealnemu. Do często stosowanych miar odległości należą funkcje z rodziny tzw. L_p - norm, definiowanych następująco:

$$L_p = \left\{ \sum_{r=1}^s \left[\frac{z_r - F_r(t)}{h_r} \right]^p \right\}^{\frac{1}{p}} \tag{6.42}$$

przyjęto, że $h_r = \bar{z}_r$ lub $h_r = \bar{z}_r - m_r$, gdzie $m_r = \min_{t \in T} F_r(t)$. Stąd można zadanie wyboru kompromisowego rozwiązania sprowadzić do zadania

$$L_{\infty}(t) = \lim_{p \rightarrow \infty} L_p(t) = \max_{1 \leq r \leq s} \left\{ \frac{\bar{z}_r - F_r(t)}{h_r} \right\} \rightarrow \min_{p \rightarrow \infty} \tag{6.43}$$

W tabeli na rys. 6.12 przedstawiono wyniki obliczeń związanych z wyborem optymalnego modelu wśród modeli M_1, M_2, M_3 wg metody kompromisowej.

Dane SI modelu									
	Rozwiązanie t						Wartości miar odległość		
Nr	x	y	a	k	kl	p	$\bar{z}_1 - Spójność_M$	$\bar{z}_2 - Spójność_kl$	$\bar{z}_3 - Połączenia^{-1}$
							h_1	h_2	h_3
M_1	20	13	46	1	4	40	0.58	0.767	0.228
M_2	20	16	64	1	7	40	0.51	0.764	0.140
M_3	25	21	80	1	11	50	0.51	0.775	0.109

Rys. 6.12. Tabela rozwiązań kompromisowych

Dla metryki $Spójność_M$ największą wartością, jaką może osiągnąć, jest wartość $\frac{x}{y}$, natomiast minimalną jest wartość -1. Stąd wartości parametrów h_r i $\bar{z}_r$ potrzebnych do wyznaczenia „odległości” są równe

$$h_1 = \frac{x}{y} - (-1) = \frac{x}{y} + 1, \quad \bar{z}_1 = \frac{x}{y}.$$

Wartość maksymalną metryki $Spójność_K$ wyznacza się na podstawie założenia, że graf funkcjonalny każdego klastra jest grafem maksymalnie spójnym oraz że liczba ról jest przynajmniej dwa razy większa od liczby typów atrybutowych w klastrze. Oznacza

to, że wartość maksymalną metryki spójności $Spójność_{kl}$ każdego klastra można wyrazić jako wyrażenie $\frac{2 * x}{x} * 1 = 2$. Wartość wypadkowej metryki spójności całego modelu klastrowego jest równa $\frac{2 * kl}{kl} = 2$. Minimalna wartość tej metryki jest równa $\frac{(-1)}{kl} * kl = -1$. Stąd wartości parametrów h_r i $\overline{z_r}$ potrzebnych do wyznaczenia „odległości” są równe

$$h_2 = 2 - \left(-\frac{kl}{kl}\right) = 2 + 1 = 3, \quad \overline{z_2} = 2.$$

Wartość maksymalna metryki $Połączenia^{-1}$ jest równa $(x + y - 1)^{-1}$ i oznacza minimalną liczbę połączeń dla spójnego grafu funkcjonalnego. Wartość minimalną wyznacza się na podstawie maksymalnej wartości połączeń dla spójnego grafu funkcjonalnego. Stąd wartości parametrów h_r i $\overline{z_r}$ potrzebnych do wyznaczenia „odległości” są równe

$$h_3 = (x + y - 1)^{-1} - (x * y)^{-1}, \quad \overline{z_3} = (x + y + z - 1)^{-1}.$$

Rozwiązanie kompromisowe potwierdza wybór modelu M_2 , gdyż dla $r = 1$ mamy najmniejszą z największych odległości równą 0.870.

7. Podsumowanie i wnioski

W rozprawie doktorskiej opracowano metody syntezy modeli struktur obiektów, operujących na polimorficznych strukturach. Jest to model struktur danych występujących w programach obiektowych. Rozpatrywanie modeli struktur danych jest uzasadnione, gdyż stanowią one szkielet struktur danych występujących w obiektowych programach wykonywalnych. Oznacza to, że na etapie projektowania i programowania te modele struktur klas są uzupełniane szczegółami implementacyjnymi. Stąd wyjściową złożoność struktur danych realizowanych podczas implementacji określa model struktur obiektów. Fakt ten jest podstawą do stwierdzenia użytego w tytule pracy doktorskiej, że dokonując syntezy struktur klas dokonujemy w ten sposób syntezy struktury programów obiektowych.

Metody syntezy modeli obiektów zostały przedstawione w rozdziale 6, gdzie głównym zadaniem tych metod jest badanie ograniczeń występujących w danych wejściowych do modelowania pod względem możliwości zastosowania polimorficznych struktur danych. Udowodniono bowiem w tym rozdziale, że zastosowanie w pewnych warunkach polimorficznych struktur danych zamiast drzew specjalizacji poprawia wartość metryki spójności *Spójność* M modelu podsystemu lub systemu i wartość metryki *Połączenia*. Te warunki stwarzają podstawy do zastosowania konwersji drzew specjalizacji na drzewa polimorficzne ze specjalizacją. Możliwość konwersji drzew udowodniono w twierdzeniu 6.1. Warunki te zdefiniowano opisując algorytm syntezy modeli struktur obiektowych. Polegają one na istnieniu takich ograniczeń zbiorowych, które dają podstawy do utworzenia drzew specjalizacji i jednocześnie określają warunki dla populacji tego drzewa - tworząc z niej sumy zbiorów. Odpowiada to operacji łączenia populacji różnych elementów drzewa poprzez te same typy faktów z populacją typów bazowych pozostałych ról należących do tych typów faktów. Udowodniono, że można tę sytuację zamodelować za pomocą różnych konstrukcji modelu. Zaprezentowano taką możliwość w dowodzie twierdzenia 6.3 i pomocniczego twierdzenia 6.2. W twierdzeniu 6.3 udowodniono, że najefektywniejszą konstrukcją w takich przypadkach są polimorficzne struktury danych ze specjalizacją. Udowodniono w ten sposób tezę niniejszej rozprawy doktorskiej. Na podkreślenie zasługuje fakt, że właśnie model drzewa polimorficznego ze specjalizacją jest modelem polimorfizmu stosowanym w wielu językach programowania obiektowego np. C++, Pascal, Object Pascal.

Aby udowodnić tezę niniejszej pracy doktorskiej, należało opracować szereg elementów uzupełniających postawy formalne metody modelowania ORM oraz zdefiniować i opracować szereg narzędzi operacyjnych do badania modeli ORM. Są one następujące:

- 1) opracowanie dla modeli ORM podstaw formalnych syntaktyki (rozdziały 2.1.2) i semantyki polimorfizmu ze specjalizacją (2.2.2) oraz opracowanie zasad identyfikacji strukturalnej (punkt b w rozdziale 2.2.6) - te modyfikacje umożliwiają operowanie taką samą konstrukcją polimorfizmu na etapie modelowania jaka występuje w językach programowania obiektowego. Prowadzi to do operowania na etapie modelowania takimi strukturami danych, które charakteryzują się porównywalną złożonością ze złożonością struktur danych w programach wykonywalnych.

- 2) zdefiniowanie zasad równoważności modeli ORM (rozdział 3.2.1) oraz algorytmu badania równoważności tych modeli (rozdział 3.2.2),
- 3) zdefiniowanie podstaw formalnych metody podziału modelu ORM na klastry jako prototypów klas (rozdział 4.1) oraz opracowanie algorytmu metody podziału modelu na klastry (rozdział 4.2) - w celu umożliwienia badania charakterystyk wewnętrznych modelu z poziomu klas,
- 4) zdefiniowanie podstaw formalnych i opracowanie metody określania charakterystyk wewnętrznych modeli struktur obiektów (rozdział 5.2 i 5.3) - za pomocą metryk *Spójność_M* i *Spójność_K* (rozdziały 5.2 i rozdział 5.3.2) oraz *Połączenia* (rozdział 5.3.1). Metody wyznaczania spójności modeli uwzględniają liczbę składowych spójności, co pozwala na obiektywną ocenę modelu. W rozdziale 5.2 pokazano inne metryki spójności stosowane w praktyce [23, 57], które nie umożliwiają oceny spójności w sposób jednoznaczny.

Udowodniona teza stała się podstawą do opracowania metody syntezy modeli struktur obiektów operujących na polimorficznych strukturach danych, przedstawionej w rozdziale 6.2.1. Metoda ta pozwala ograniczyć zbiór nieefektywnych programów wg kryterium Pareta przez zastosowanie we właściwy sposób polimorficznych struktur danych, co udowodniono w twierdzeniu 6.3. Ocenę efektywności zbioru rozwiązań wg kryterium Pareta opisano w rozdziale 6.2.2. Podano również w tym rozdziale metodę wyboru jednego rozwiązania ze zbioru rozwiązań optymalnych (efektywnych) wg Pareta za pomocą metody kompromisowej. Metoda ta jest oparta na miarach odległości wartości poszczególnych kryteriów (czyli metryk) od rozwiązania maksymalnego i minimalnego. Na podstawie zasad tworzenia modeli ORM zdefiniowano rozwiązanie maksymalne i minimalne dla każdej z metryk.

W rozdziale 6.2 przedstawiono opracowany algorytm stosowania metod syntezy modeli ORM operujących na polimorficznych strukturach danych. Przedstawiono wszystkie możliwe scenariusze użycia sześciu metod opracowanych w niniejszej rozprawie. Dzięki temu można w sposób automatyczny zrealizować różne postulaty z dziedziny modelowania obiektowego - konstruowania efektywnych modeli konceptualnych operujących na polimorficznych strukturach danych, transformacji schematów, oceny równoważności schematów konceptualnych, tworzenia obiektowych modeli klas, oceny złożoności strukturalnej modeli, eliminacji rozwiązań nieefektywnych wg zasady Pareta, wyboru rozwiązania dominującego ze zbioru rozwiązań efektywnych. Odniesienia do istniejących rozwiązań, prezentowanych w literaturze technicznej zawarto w poszczególnych rozdziałach. Opracowane metody nadają się do wykonania systemów automatycznego wspomagania tworzenia programów obiektowych.

Przedstawione w punktach 1- 4 formalizmy i narzędzia zostały zilustrowane przykładami potwierdzającymi przydatność wprowadzonych zmian w modelowaniu ORM oraz użyteczność opracowanych algorytmów. Wykonano program do oceny spójności grafów dwudzielnych reprezentujących grafy funkcjonalne ORM, co pozwala wybrać wyrażenie *LW1* lub *LW2* do wyznaczenia wartości metryki *Spójność_M* lub *Spójność_kl*. Wykonano również program do badania równoważności w sensie zachowania zależności funkcyjnych. Rezultaty praktyczne uzyskane w niniejszej pracy potwierdzają przydatność wykonanych badań w dziedzinie zastosowań inżynierii oprogramowania.

Zdaniem autora, w niniejszej rozprawie osiągnięto podstawowe cele - udowodniono tezę i opracowano metodę syntezy programów obiektowych operujących na polimorficznych strukturach danych.

Dalsze prace będą dotyczyły realizacji środowiska wspomagającego tworzenie oprogramowania obiektowego wykorzystywanego między innymi do celów dydaktycznych.

Szczególnie pożądane są dalsze badania nad złożonością strukturalną modeli związaną ze specjalizacją i polimorfizmem oraz opracowanie odpowiednich metryk. W badaniach tych należy położyć szczególny nacisk na określenie powiązań między opracowanymi miarami wewnętrznej złożoności strukturalnej a charakterystykami zewnętrznymi oprogramowania obiektowego. Szczególnie ważne wydaje się ustalenie powiązań między testowalnością i „rozmiarem” polimorfizmu zastosowanego w modelach.

Spis literatury

1. F.B.e Abreu, *Design Quality Metrics for Object-Oriented Software Systems*, <http://www.ercim.irin/abreu.html>.
2. J.Akoka, I.Comyn-Wattiau, *Entity-Relationship and object-oriented model automatic clustering*, Data & Knowledge Engineering 20 (1996) 87-117.
3. A.Analyti, N.Spyratos, P.Constantopoulos, *Deriving semantic information through property covering and inheritance*, Data & Knowledge Engineering 28 (1998) 3-30.
4. V. De Antonellis, S.Castano, L.Vandoni, *Building reusable components through project evolution analysis*, Information Systems, 1994, vol. 19, no. 3, 259-274.
5. R.Barker, *Case Method, Modelowanie związków encji*, WNT, Warszawa 1996.
6. V.R.Basili, L.Briand, W.L.Melo, *A Validation of Object-Oriented Design Metrics as Quality Indicators*, IEEE Transactions On Software Engineering vol. 22, no. 10, October 1996, 751-761.
7. V.R.Basili, S.Condon, K.El Eman, B.Hendrick, W.Melo, *Characterizing and Modeling the Cost of Rework in Library of Reusable Software Components*, University of Maryland, 1997, ISERN-97-07, http://www.iese.fhg.de/ISERN/pub/isern_biblio_tech.html.
8. R.Barker, C.Longman, *Case Method, Modelowanie funkcji i procesów*, WNT, Warszawa 1996.
9. D.Begg, S.Fisher, R.Dornbusch, *Ekonomia, Mikroekonomia*, PWE Warszawa, 1997.
10. E.Bertino, H. Weigand, *An approach to authorization modeling in object-oriented database systems*, Data & Knowledge Engineering 12 (1994) 1-29.
11. A.Bertolino, L.Strigini, *On the Use of Testability Measures for Dependability Assessment*, IEEE Transactions On Software Engineering vol. 22, no. 2, February 1996, 97-108.
12. P.Biggs, *A Survey of Object-Oriented Methods*, University of Durham, <http://www.dur.ac.uk/-dcs3pjb/survey.html>
13. J.Bishop, *Data abstraction in programming languages*, Addison-Wesley Publishing Company, 1986.
14. I.Blumke, *Miary oprogramowania obiektowego*, Informatyka 2/99, 18-21.
15. P.van Bommel, A.H.M. ter Hofstede, Th.P.van der Weide, *Semantics and verification of object-role models*, Information Systems, 1991, vol. 16, no. 5, 471-495.
16. P. van Bommel, Th.P.van der Weide, *Reducing the search space for schema transformation*, Data & Knowledge Engineering 8 (1992) 269-292.
17. G.Booch, *Object Oriented design with application*, The Benjamin/Cummings Publishing Company, Inc, 1991.
18. D.Buedene, *Decision Analysis Software Survey*, <http://lionhrtpub.com/orms-8-96/buede.html>

19. D.Boulanger, G.Dubuis, *An object approach for information system cooperation*, Information Systems, 1998, vol. 23, no. 6.
20. J.M.Brady, *Informatyka teoretyczna w ujęciu programistycznym*, WNT Warszawa, 1983
21. C.J.Breiteneder, T.A.Muck, *Graph rewriting systems for the Entity-Relationship approach*, Data & Knowledge Engineering 17 (1995) 215-243.
22. L.C.Briand, J.W.Daly, V.Porter, J.Wust, *A Comprehensive Empirical Validation of Product Measures for Object-Oriented Systems*, Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern, Germany 1998, ISERN-98-07, http://www.iese.fhg.de/ISERN/pub/isern_biblio_tech.html.
23. C.Briand, J.W.Daly, J.Wust, *A Unified Framework for Cohesion Measurement in Object-Oriented Systems*, Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern, Germany 1997, ISERN-97-05, http://www.iese.fhg.de/ISERN/pub/isern_biblio_tech.html.
24. C.Briand, J.W.Daly, J.Wust, *A Unified Framework for Coupling Measurement in Object-Oriented Systems*, IEEE Transactions On Knowledge and Data Engineering, vol. 25, no. 1, January-February 1999.
25. L.C.Briand, K. El Eman, S.Morasca, *On the Application of Measurement Theory in Software Engineering*, ISERN-95-04, http://www.iese.fhg.de/ISERN/pub/isern_biblio_tech.html.
26. L.C.Briand, K. El Eman, S.Morasca, *Theoretical and Empirical Validation of Software Product Measures*, ISERN-97-03, http://www.iese.fhg.de/ISERN/pub/isern_biblio_tech.html.
27. L.C.Briand, K. El Eman, D.Surmann, I.Wieczorek, *An Assessment and Comparison of Common Software Cost Estimation Modelling Techniques*, ISERN-98-27, http://www.iese.fhg.de/ISERN/pub/isern_biblio_tech.html.
28. C.Briand, S.Morasca, V.R.Basili, *Defining and Validation Measures for Object-Based High-Level Design*, Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern, Germany 1998, ISERN-98-04, http://www.iese.fhg.de/ISERN/pub/isern_biblio_tech.html.
29. C.Briand, S.Morasca, V.R.Basili, *Property Based Software Engineering Measurement*, IEEE Transactions On Knowledge and Data Engineering, vol.22, no. 1, January 1996.
30. C.Briand, J.Wust, S.Ikonomowski, H. Lounis, *A Comprehensive Investigation of Quality Factors in Object-Oriented Designs: an Industrial Case Study*, ISERN-98-29, http://www.iese.fhg.de/ISERN/pub/isern_biblio_tech.html.
31. C.Briand, J.Wust, H.Lounis, *Using Coupling Measurement for Analysis in Object-Oriented Systems*, ISERN-99-03, http://www.iese.fhg.de/ISERN/pub/isern_biblio_tech.html.
32. G.H.W.M.Bronts, S.J.Brouwer, C.L.J.Martens, H.A.Proper, *Unifying object role modelling theory*, Information Systems, 1995, vol. 20, no. 3, 213-235.
33. J.Brunet, C.Cauvet, D.Meddahi, F.Semmak, *Applying object-oriented analysis on a*

- case study*, Information Systems, 1994, vol. 19, no. 3, 199-209.
34. L.I.Campbell, T.A.Halpin, H.A.Proper, *Conceptual schemas with abstractions making flat conceptual schemas more comprehensible*, Data & Knowledge Engineering 20 (1996) 39-85.
 35. T.Catarci, G.D'Angiolini, M.Lenzerini, *Conceptual language for statistical data modeling*, Data & Knowledge Engineering 17 (1995) 93-125.
 36. P.P.Chen, *Entity-Relation approach to systems analysis and design*, Proceedings of International Conference on Entity-Relation Approach to Systems Analysis and Design, Los Angeles, Decemeber 10-12 1979.
 37. S.R.Chidamber, D.P.Darcy, C.F.Kemer, *Managerial Use of Metrics for Object-Oriented Software: An Exploratory Analysis*, IEEE Transactions On Knowledge and Data Engineering, vol.24, no. 8, August 1998, 629-639.
 38. P.Coad, E.Yourdon, *Analiza obiektowa*, Oficyna Wydawnicza READ ME, Warszawa 1994.
 39. P.Coad, E.Yourdon, *Projektowanie obiektowe*, Oficyna Wydawnicza READ ME, Warszawa, 1994.
 40. P.Coad, J.Nicola, *Programowanie obiektowe*, Oficyna Wydawnicza READ ME, Warszawa 1994.
 41. S.Conrad, *A basic calculus for verifying properties of interacting objects*, Data & Knowledge Engineering 18 (1996) 119-145.
 42. T.H.Cormen, C.E.Leiserson, R.L.Rivest, *Wprowadzenie do algorytmów*, WNT Warszawa, 1998.
 43. P.N.Creasy, H.A.Proper, *A generic model for 3-dimensional conceptual modelling*, Data & Knowledge Engineering 20 (1996) 119-145.
 44. G.Dedene, M.Snoeck, *Formal deadlock elimination in an object oriented conceptual schema*, Data & Knowledge Engineering 15 (1995) 1-30.
 45. N.Deo, *Teoria grafów i jej zastosowania w technice i informatyce*, Biblioteka Naukowa Inżyniera, Warszawa 1980.
 46. R.G.Dromey, *A Model for Software Product Quality*, IEEE Transactions On Software Engineering vol. 21, no. 2, February 1995, 146-162.
 47. G.Engels, M.Gogolla, U.Hohenstein, K.Hulsmann, P.Lohr-Richter, G.Saake, H.D.Ehrich, *Conceptual modelling of database applications using an extended ER model*, Data & Knowledge Engineering 9 (1992/93) 157-204.
 48. H.E.Eriksson, M.Penker, *UML Toolkit*, Wiley Computer Publishing, John Wiley & Sons, Inc, 1998.
 49. T.Fetcke, A.Abran, T.-H. Nguyen, *Mapping the OO-Jacobson Approach to Function Point Analysis*,
<http://www.scit.wlv.ac.uk/university/scit/research/internal.papers.html>
 50. M.Flaśiński, *Wstęp do analitycznych metod projektowania systemów informatycznych*, WNT Warszawa ,1997.
 51. M.Gargano, E.Nardelli, M.Talamo, *Abstract data types for the logical modeling of*

- complex data*, Information Systems, 1991, vol.16, no. 6, 565-583.
52. J.Gutttag, *Abstract Data Types and the Development of Data Structures*, Communication of the ACM, June 1977, vol. 20, no 6, 396-404.
 53. J.-L.Hainaut, *Specification preservation in schema transformations-application to semantics and statistics*, Data & Knowledge Engineering 19 (1996) 99-134.
 54. T.A.Halpin, H.A.Proper, *Subtyping and polymorphism in object-role modelling*, Data & Knowledge Engineering 15 (1995) 251-281.
 55. R.Harrison, S.J.Counsell, R.V.Nithi, *An Evaluation of the MOOD Set of Object-Oriented Software Metrics*, IEEE Transactions On Knowledge and Data Engineering, vol. 24, no. 6, June 1998.
 56. I.Hatzilygeroudis, H.Reichgelt, *Handling inheritance in a system integrating in objects*, Data & Knowledge Engineering 21 (1997) 253-280.
 57. B.Henderson-Sellers, *Object-Oriented Metrics, Measures of Complexity*, Prentice Hall PTR 1996.
 58. A.H.M. ter Hofstede, H.A.Proper, Th.P.van der Weide, *Exploiting facts verbalisation in conceptual information modelling*, Information Systems, 1997, vol. 22, no. 6/7, 349-385.
 59. A.H.M. ter Hofstede, H.A.Proper, Th.P.van der Weide, *Formal definition of a conceptual language for the description and manipulation of information models*, Information Systems, 1993, vol. 18, no. 7, 489-523.
 60. A.H.M. ter Hofstede, Th.P.van der Weide, *Expressiveness in conceptual data modelling*, Data & Knowledge Engineering 10 (1993) 65-100.
 61. J.W.G.M. Hubers, A.H.M ter Hofstede, *Formalization of communication and behaviour in object-oriented analysis*, Data & Knowledge Engineering 23 (1997) 147-183.
 62. M.S.Jackson, *Database Management System are not Programming Languages*, <http://www.scit.wlv.ac.uk/university/scit/research/internal.papers.html>
 63. M.S.Jackson, *The Object-Oriented Database Management Interface*, <http://www.scit.wlv.ac.uk/university/scit/research/internal.papers.html>
 64. M.S.Jackson, *Using Object Role Modelling to Design Databases*, <http://www.scit.wlv.ac.uk/university/scit/research/internal.papers.html>
 65. R.B.Jackson, D.W.Embley, S.N.Woodfield, *Developing formal object-oriented requirements specifications: a model, tool and technique*, Information Systems, 1995, vol. 20, no. 4, 273-289.
 66. D.A.Jardine, *A logic-based information modelling language*, Data & Knowledge Engineering 2 (1987) 59-81.
 67. P.Johannesson, *Schema standarization as an aid in view integration*, Information Systems, 1994, vol. 19, no. 3.
 68. C.Jones, *The economics of object-oriented software*, April 14, 1997, <http://dec.bournemouth.ac.uk/ESRG/downloads/Capers-strength.rtf>
 69. C.Jones, *Strengths and weaknesses of software metrics*, July 29, 1997,

<http://dec.bournemouth.ac.uk/ESRG/downloads/Capers-strength.rtf>

70. T.H.Jones, II-Yeol Song, *Analysis of binary/tenary cardinality combinations in entity-relationship modeling*, Data & Knowledge Engineering 19 (1996) 39-64.
71. H.Kangassalo, *COMIC: A system and methodology for conceptual modelling and information construction*, Data & Knowledge Engineering 9 (1993) 287-319.
72. M.Ketabchi, K. Jambor-Sadeghi, *Modeling application domains*, Data & Knowledge Engineering 22 (1997) 261-281.
73. Z.Kierzkowski, *Komputerowa wymiana informacji*, SORUS Poznań, 1995
74. D.H.Kim, S.J.Park, *FORM: A flexible data model for integrated CASE environments*, Data & Knowledge Engineering 22 (1997) 133-158.
75. J.L.Knapp, *ER isomorphisms and uniqueness conditions*, Data & Knowledge Engineering 26 (1998) 271-290.
76. Y.Kobayashi, *Losslessness and semantic correctness of database schema transformation: another look of schema equivalence*, Information Systems, 1986, vol. 11, no. 1, 41-60.
77. A.Kobylnski, *Miary oprogramowania*, Informatyka 10/98, 26-29.
78. Y.Kornatzky, P.Shoval, *Conceptual design of object-oriented database schemas using the binary-relationship model*, Data & Knowledge Engineering 14 (1994) 265-288.
79. G.Kovacs, P.van Bommel, *Conceptual Modelling Based Design of Object-oriented Databases*, <http://www.cs.kun.nl/~pvb/reports.html>.
80. G.Kovacs, P.van Bommel, *From Conceptual Model to OO Databases via Intermediate Specification*, <http://www.cs.kun.nl/~pvb/reports.html>.
81. Z.Kruczkiewicz, *Metody programowania obiektowego*, Oficyna Wydawnicza Politechniki Wrocławskiej, 1996.
82. Z.Kruczkiewicz, *Efektywność modeli oprogramowania obiektowego*, Zeszyty Naukowe Wydziału Mechanicznego, Politechnika Koszalińska, Materiały XVI Ogólnopolskiej Konferencji: Polioptymalizacja i komputerowe wspomaganie projektowania Mielno '98.
83. Z.Kruczkiewicz, A.Sielicki, *Specyfikacja obiektowa zasobów i systemów informacyjnych*, Materiały IV Krajowego Forum Informacji Naukowej i Technicznej, Zakopane '97.
84. O.Laitenberger, J.-M. DeBaund, *An Encompassing Life-Cycle Centric Survey of Software Inspection*, ISERN-98-32, http://www.iese.fhg.de/ISERN/pub/isern_biblio_tech.html.
85. J.Y.Lee, R.Elmarasi, Jongho Won, *An integrated temporal data model incorporating time series concept*, Data & Knowledge Engineering 24 (1998) 257-276.
86. Levy A., *Basic Set Theory*, Springer-Verlag, Berlin, Heidelberg, New York 1979.
87. K.Lieberherr, *Formal Foundations for Object-Oriented Data Modeling*, IEEE

- Transactions on Knowledge and Data Engineering, vol. 5, no. 3, June 1993.
88. B.Liskov, J.Gutttag, *Abstraction and specification in program development*, wydanie rosyjskie „Mir” Moskwa, 1989 (tłumaczenie wydania The Massachusetts Institute of Technology, 1986).
 89. L.Liu, R.Zicari, W.Hursch, K.J.Lieberherr, *The Role of Polymorphic Reuse Mechanisms in Schema Evolution in an Object-Oriented Database*, IEEE Transactions On Knowledge and Data Engineering, vol. 9, no. 1, January-February 1997, 50-67.
 90. S.Maffeis, Olsen and Associates, D.C.Schmidt, Washington University, *Constructing Reliable Distributed Communication Systems with CORBA*, IEEE Communications Magazine, February 1997, 56-60
 91. D.Maier, *The Theory of Relational Databases*, wydanie rosyjskie „Mir” Moskwa, 1987 (tłumaczenie wydania Computer Science Press, Inc. 1983).
 92. J.A.Makovsky, E.V.Ravve, *Dependency preserving refinements and the fundamental problem of database design*, Data & Knowledge Engineering 24 (1998) 277-312.
 93. R.Martin, *OO Design Quality Metrics and Analysis of Dependencies*, October 28, 1994, <http://www.oma.com/offerings/catalog.html>.
 94. J.Martin, J.Odell, *Podstawy metod obiektowych*, WNT Warszawa, 1997.
 95. N.M.Mattos, K.Meyer-Wegener, B.Mitschang, *Grand tour of concepts for object-orientation from database point of view*, Data & Knowledge Engineering 9 (1992/93) 321-352.
 96. B.Meyer, G.D.Westerman, M.Gogolla, *Drafting ER and OO schemas in prototype environments*, Data & Knowledge Engineering 19 (1996) 201-240.
 97. A.McAllister, *Complete rules for n-ary relationship cardinality constraints*, Data & Knowledge Engineering 27 (1998) 255-288.
 98. Xue-Miao Lu, T.S.Dillon, *An Algebraic Theory of Object-Oriented Systems*. IEEE Transactions on Knowledge and Data Engineering, vol. 6, no. 2, June 1994.
 99. R.J.Miller, J.E.Ioamidis, R.Ramakrishnan, *Schemat equivalence in heterogeneous systems: bridging theory and practice*, Information Systems, 1994, vol. 19, no 1, 3-31.
 100. W.Y.Mok, D.W.Embley, *Using NNF to transform conceptual data models to object-oriented database designs*, Data & Knowledge Engineering 24 (1998) 313-336.
 101. B.Moulin, P.Creasy, *Extending the conceptual graph approach for data conceptual modelling*, Data & Knowledge Engineering 8 (1992) 223-248.
 102. H.J.Muller, *Towards agent systems engineering*, Data & Knowledge Engineering 23 (1997) 217-245.
 103. T.Pankowski, *Podstawy baz danych*, Wydawnictwo Naukowe PWN, Warszawa 1992.
 104. V.Podinowskij, V.D.Nogin, *Pareto-opimal'nyje resenija mnogokriterial'nych*

- zadac, Moskva Nauka, 1982.
105. A. Poulouvasilis, P.M. Brien, *A general formal framework for schema transformation*, Data & Knowledge Engineering 28 (1998) 47-71.
 106. H.A. Proper, *Data schema design as a schema evolution process*, Data & Knowledge Engineering 22 (1997) 159-189.
 107. H.A. Proper, Th.P. van der Weide, *EVORM: A conceptual modelling technique for evolving application domains*, Data & Knowledge Engineering 12 (1994) 313-359.
 108. H.A. Proper, Th.P. van der Weide, *Information disclosure in evolving information systems: taking a shot at a moving target*, Data & Knowledge Engineering 15 (1995) 135-168.
 109. C. Quer, A. Olive, *Determining object interaction on object-oriented deductive conceptual models*, Information Systems, 1994, vol. 19, no. 5, 211-227.
 110. M.A. Qutaishat, N.J. Fiddian, W.A. Gray, *Extending OMT to support bottom-up design modelling in a heterogeneous distributed database environment*, Data & Knowledge Engineering 22 (1997) 191-205.
 111. A. Rochfeld, P. Negros, *Relationship of relationships and other inter-relationship links in E-R model*, Data & Knowledge Engineering 9 (1992) 205-221.
 112. M. Rosi, S. Brinkkemper, *Complexity metrics for systems development methods and techniques*, Information Systems, 1996, vol. 21, no. 2, 209-227.
 113. B. Roy, *Wielokryterialne wspomaganie decyzji*, Bernard Roy, WNT Warszawa, 1990.
 114. J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen, *Object-oriented modeling and design*, Prentice-Hall International, Inc., 1991.
 115. K.D. Schewe, *Consistency enforcement in Entity-Relationship and object-oriented models*, Data & Knowledge Engineering 28 (1998) 121-140.
 116. D.C. Schmidt, A.S. Gokhale, T.H. Harrison, G. Parulkar, Washington University, *A high-performance end system architecture for real-time CORBA*, IEEE Communications Magazine, February 1997, 72-77.
 117. S. Schwiderski, T. Hartmann, G. Saake, *Monitoring temporal preconditions in a behaviour oriented object model*, Data & Knowledge Engineering 14 (1994) 143-186.
 118. C. Sernadas, J. Fiadeiro, *Towards object-oriented conceptual modeling*, Data & Knowledge Engineering 6 (1991) 479-508.
 119. M. Shepperd, M. Cartwright, *An Empirical Investigation of an Object-Oriented Software System*, October 1997,
<http://xanadu.bournmouth.ac.uk/staff/mshepperd/OOMetrics.html>.
 120. P. Shoval, S. Shiran, *Entity-relational and object-oriented data modeling-an experimental comparison of design quality*, Data & Knowledge Engineering 21 (1997) 297-315.
 121. P. Shoval, N. Shreiber, *Database reverse engineering: from the relational to the binary relationship model*, Data & Knowledge Engineering 10 (1993) 293-315.

122. P. Shoval, S. Zohn, *Binary-relationship integration methodology*, Data & Knowledge Engineering 6 (1991) 225-250.
123. M. Snoeck, G. Dedene, *Existence Dependency: The Key to Semantic Integrity Between Structural and Behavioral Aspects of Object Types*, IEEE Transactions On Software Engineering vol. 24, no. 4, April 1998, 233-251.
124. M. Snoeck, G. Dedene, *Generalization/specialization and role in object oriented conceptual modeling*, Data & Knowledge Engineering 19 (1996) 171-195.
125. A. Steiner, M. Norrie, *Temporal Object Role Modelling*, Conference paper: Advanced Information Systems Engineering 9th International Conference, CAiSE 97 Proceedings p. 245-258.
126. V.C. Storey, Heng_Li Yang, R.C. Goldstein, *Semantic integrity constraints in knowledge-based database design systems*, Data & Knowledge Engineering 20 (1996) 1-37.
127. R. Studer, V.R. Benjamins, D. Fensel, *Knowledge Engineering: Principles and methods*, Data & Knowledge Engineering 25 (1998) 161-197.
128. M. Sysło, N. Deo, J.S. Kowalik, *Algorytmy optymalizacji dyskretnej z programami w języku Pascal*, Wydawnictwo Naukowe PWN, Warszawa 1993.
129. Z. Tari, J. Stokes, S. Spaccapietra, *Object Normal Forms and Dependency Constraints for Object-Oriented Schemata*, ACM Transactions on Databases, December 1997 vol. 22, no. 4, 513-569.
130. A. Tarski, *Wprowadzenie do logiki*, Filia Uniwersytetu Warszawskiego PHILOMATH, 1994.
131. D.A. Taylor, *Technika obiektowa, Rzecz o zastosowaniach programowania obiektowego*, Helion, 1994.
132. R.G. Thompson, *Assesing the Quality of OpenVMS AXP: Software Measurement Using Subjective Data*,
http://ftp.digital.com/pub/DEC/DECinfo/DTJ/v5n4/Assessing_the_Quality_of_Open_V_01apr1994DTJC06SC.txt
133. J. Tian, M.V. Zelkovitz, *Complexity Measure Evaluation and Selection*, IEEE Transactions On Software Engineering vol. 21, no. 8, April 1995, 641-650.
134. P.C. Treleaven, *Parallel Computers, Object-Oriented, Functional, Logic*, John Wiley & Sons, 1990.
135. O. De Troyer, *A formalization of the Binary Object-Role Model based on logic*, Data & Knowledge Engineering 19 (1996) 1-37.
136. V. Turau, H. Duchene, *Equality testing for complex objects based on hashing*, Data & Knowledge Engineering 10 (1993) 101-111.
137. W. Turski, *Metodologia programowania*, WNT, Warszawa, 1978.
138. R.J. Wilson, *Wprowadzenie do teorii grafów*, PWN, Warszawa 1985.
139. S. Vinoski, *CORBA: Integrating Diverse Applications within Distributed Heterogeneous Environments*, IEEE Communications Magazine, February 1997, 46-55.

Załączniki

1. Pakiet <i>ModGraf</i> zawierający klasę do badania zależności funkcyjnych.....	173
2. Pakiet <i>Strukt</i> zawierający struktury danych typu <i>Kolekcja</i>	182
3. Program obiektowy <i>Grafy</i> zawierający obiekt do badania zależności funkcyjnych.....	186
4. Pakiet <i>Dodatki</i> zawierający pomocnicze podprogramy.....	187
5. Program <i>Spójne_Składowe</i> do wyznaczania liczby spójnych składowych w grafie dwudzielnym.....	188

```

{-----}
{               Pakiet ModGraf               }
{      Klasa sprawdzająca zależności funkcyjne w grafach      }
{      dwudzielnym funkcjonalnym                      }
{      (Rozdział 3.2.3)                                }
{-----}

```

```

unit ModGraf;
interface
  uses Dodatki, Strukt, Crt;
  const n=30;
        m=30;
type
  zb=set of byte;
const Zb_a : zb = {[1,2,3,4,5,6,7,8,9]; }
              [1,2,3,4,5,6,7,8,9,10];
type
  pa=^a;
  a=record
    atrybut: byte; nast_atorybut: pa;
  end;

  pb=^b;
  b=record
    b_1, b_2: byte; nast_rola: pb;
  end;

  pc=^c;
  c=record
    nr_ta: byte; a_idf: pa; r_idf: pb;
  end;

  pf=^f;
  f=record
    dziedzina, przeciwdziedzina_w: pa;
  end;

  d=record
    przyleglosc, o_total, o_unikat, odwiedzony : byte;
  end;
MTab=array[1..n, 1..m] of d;

```

```

    Zal_Funk = object
private
    ile_n, ile_p: word;
    NT, PT: PKolekcja;
    i_MT, j_MT: integer;
    MT: MTab;
    procedure Czytaj_MT;
    procedure Wyświetl_MT;
    procedure Czytaj_NT;
    procedure Wyświetl_NT;
    procedure Ustaw_MT;
    procedure Ustaw_PT;
    procedure Dodaj_do_listy(a:integer;var p:pa);
    procedure Dodaj_do_listy_a(Zrodlo,Cel,Jak:byte;Ile,Co:word);
public
    procedure Inicjuj;
    procedure Zaleznosci_funkcyjne;
    procedure Wyświetl_zaleznosci_funkcyjne;
    procedure Usun;
end;

implementation
    procedure Zal_Funk.Inicjuj;
    begin
        Czytaj_MT;
        Wyświetl_MT;
        Czytaj_NT;
        Wyświetl_NT;
        Ustaw_MT;
        Ustaw_PT;
    end;

    procedure Zal_Funk.Wyświetl_MT;
    {drukowanie na ekranie danych macierzy M}
    var z:char;
        i,j:integer;
    begin
        ClrScr;
        for i:=1 to i_MT do
            for j:=1 to j_MT do
                with MT[i,j] do
                    Write(przyleglosc:2,o_total:2,o_unikat:2,odwiedzony:2,' ');
                    Writeln;
                    Pauza;
                end;
            end;
        end;
    end;

```

```

procedure Zal_Funk.Czytaj_MT;
{Czytanie danych macierzy M}
var Plik:Text;
    i,j:integer;
begin
    Assign(Plik,'MT2.pas');
    {$I-}
    Reset(Plik);
    {$I+}
    if IOResult <>0 then Halt;
    Readln(Plik,i_MT,j_MT);
    if (i_MT>n) or (j_MT>m) then Halt;
    while not Seekeof(Plik) do
        while not Seekeoln(Plik) do
            for i:=1 to i_MT do
                for j:=1 to j_MT do
                    with MT[i,j] do Read(Plik,przyleglosc,o_total,o_unikat,odwiedzony);
    end;

```

```

procedure Zal_Funk.Wyswietl_NT;
{drukowanie na ekranie zawarości macierzy N}
var Pom1:pa; Pom2:pb;
    Pom3:pc;i:integer;
begin
    i:=1;
    Pom3:=NT^.Podaj_Element(i);
    while Pom3<>nil do
        begin
            Pom1:=Pom3^.a_idf; Pom2:=Pom3^.r_idf; Write(Pom3^.nr_ta:3);
            while Pom1<>nil do
                begin
                    Write(Pom1^.atrybut:4); Pom1:=Pom1^.nast_atrybut;
                end;
            while Pom2<>nil do
                begin
                    Write(Pom2^.b_1:3, Pom2^.b_2:3); Pom2:=Pom2^.nast_rola;
                end;
            Writeln;
            Inc(i);
            Pom3:=NT^.Podaj_Element(i);
        end;
    end;

```

```

procedure Zal_Funk.Czytaj_NT;
  {czytanie danych macierzy N}
var Pom1:pa;Pom2:pb; Pom3:pc;
      Plik:Text; Buf1,Buf2:byte;
begin
  Assign(Plik,'nt2.pas');
  {$I-}
  Reset(Plik);
  {$I+}
  if IOResult <> 0 then Halt;
  ReadLn(Plik,ile_n);
  Pamiec(ile_n*SizeOf(Pointer)); New(Nt,Inicjuj(ile_n));
  while not Seekeof(Plik) do
    begin
      New(Pom3);
      Read(Plik,Pom3^.nr_ta); Pom3^.a_idf:=nil; Pom3^.r_idf:=nil;
      Read(Plik,Buf1);
      while Buf1<>0 do
        begin
          if Pom3^.a_idf=nil then
            begin
              new(Pom3^.a_idf); Pom1:=Pom3^.a_idf;
            end
          else
            begin
              new(Pom1^.nast_atrybut); Pom1:=Pom1^.nast_atrybut;
            end;
            Pom1^.atrybut:=Buf1; Pom1^.nast_atrybut:=nil; Read(Plik,Buf1);
          end;
        while not Seekeofln(Plik) do
          begin
            Read(Plik,Buf1); Read(Plik,Buf2);
            if Pom3^.r_idf=nil then
              begin
                new(Pom3^.r_idf); Pom2:=Pom3^.r_idf;
              end
            else
              begin
                new(Pom2^.nast_rola); Pom2:=Pom2^.nast_rola;
              end;
            Pom2^.b_1:=Buf1; Pom2^.b_2:=Buf2; Pom2^.nast_rola:=nil;
          end;
          NT^.dodaj(Pom3);
        end;
      close(Plik);
    end;
  end;

```

```

procedure Zal_Funk.Ustaw_MT;
  var i,j,k,l:integer; Pom2:pb;
      Pom1:pc;
  begin
    k:=1;
    Pom1:=NT^.Podaj_Element(k);
    while Pom1<>nil do
      begin
        l:=Pom1^.nr_ta; Pom2:=Pom1^.r_idf;
        while Pom2 <> nil do
          begin
            i:=Pom2^.b_1; j:=Pom2^.b_2;
            {odwiedzony}
            MT[i,j].odwiedzony:=1; MT[l,j].odwiedzony:=1;
            Pom2:=Pom2^.nast_rola;
          end;
          Inc(k); Pom1:=NT^.Podaj_Element(k);
        end;
      end;
    end;

procedure Zal_Funk.Ustaw_PT;
  var Pom1:pf;Pom2:pc;i:integer;
  begin
    Pamiec(ile_n*SizeOf(Pointer));
    New(PT,Inicjuj(ile_n));
    ile_p:=ile_n;
    for i:=1 to ile_p do
      begin
        Pom2:=NT^.Podaj_element(i);
        New(Pom1);
        Pom1^.dziedzina:=Pom2^.a_idf; Pom1^.przeciwdziedzina_w:=nil;
        PT^.Dodaj(Pom1);
      end
    end;

procedure Zal_Funk.Dodaj_do_listy(a:integer;var p:pa);
  var Pom,Pop,Nast:pa;
  begin
    New(Pom); Pom^.atrybut:=a; Pom^.nast_atrybut:=nil;
    if p=nil then p:=Pom
    else
      if p^.atrybut = a then
        begin
          Dispose(Pom); Exit
        end
      else

```



```

if p^.atrybut>a then
begin
  Pom^.nast_atrybut:=p; p:=Pom;
end
else
begin
  Pop:=p; Nast:=p^.nast_atrybut;
  while (Nast<>nil) and (Nast^.atrybut<a) do
  begin
    Pop:=Nast; Nast:=Nast^.nast_atrybut;
  end;
  if (Nast<>nil) and (Nast^.atrybut=a) then
  begin
    Dispose(Pom); Exit;
  end;
  Pop^.nast_atrybut:=Pom;
  Pom^.nast_atrybut:=Nast;
end;
end;

procedure Zal_Funk.Dodaj_do_listy_a(Zrodlo,Cel,Jak:byte;Ile,Co:word);
var r,a,b:byte;Pom:pa;Pom1:pc;Pom2:pf;
begin
  r:=0;
  repeat
    Inc(r);
    Pom1:=NT^.Podaj_Element(r);
  until (r=Ile) or (Pom1^.nr_ta=Cel);
  r:=r+Co;
  if Jak=1 then
  begin
    a:=r; b:=Zrodlo;
  end
  else
  begin
    a:=Zrodlo; b:=r;
  end;
  Pom1:=NT^.Podaj_Element(b);
  Pom:=Pom1^.a_idf;
  Pom2:=PT^.Podaj_Element(a);
  while Pom<> nil do
  begin
    Dodaj_do_listy(Pom^.atrybut,Pom2^.przeciwdziedzina_w);
    Pom:=Pom^.nast_atrybut;
  end;
end;

```

```

procedure Zal_Funk.Zaleznosci_funkcyjne;
var i,j,k,l,r:integer; Pom:pa;Pom1:pc; Pom2:pf;
begin
  k:=1;r:=0;
  while k <= ile_n do
{1} begin
  Pom1:=NT^.Podaj_Element(k);
  i:=Pom1^.nr_ta;
  for j:=1 to j_MT do
    if (MT[i,j].przyleglosc=1) and (MT[i,j].o_total=1)
      and (MT[i,j].o_unikat=1)
      and (MT[i,j].odwiedzony=0) then
{2}   begin
      MT[i,j].odwiedzony:=1;
      for l:=1 to n do
{3}     begin
{4}       if (MT[l,j].przyleglosc =1) and (l in Zb_a) then
         begin
           MT[l,j].odwiedzony:=1;
           Pom2:=PT^.Podaj_element(k);
           Dodaj_do_listy(l,Pom2^.przeciwdziedzina_w);
           if (MT[l,j].o_total=1) and (MT[l,j].o_unikat=1) then
             begin
               Inc(r);
               Pom2:=PT^.Podaj_element(ile_n+r);
               Dodaj_do_listy(l,Pom2^.dziedzina);
               Dodaj_do_listy_a(k,l,1,ile_n,r)
             end;
           end
         end
       else
         if (MT[l,j].przyleglosc=1) and (MT[l,j].odwiedzony=0) then
{4}       begin
           MT[l,j].odwiedzony:=1;
           {l-ty obiekt abstrakcyjny w i-tym}
           Dodaj_do_listy_a(k,l,0,ile_n,0);
           if (MT[l,j].o_total=1) and (MT[l,j].o_unikat=1) then
             {teraz i-ty abstrakcyjny obiekt w l-tym}
             Dodaj_do_listy_a(k,l,1,ile_n,0);
           end;
         end;
       end;
     end;
{3}   end;
{2}   end;
      Inc(k);
{1} end;
      Inc(ile_p,r);
end;

```

```

procedure Zal_Funk.Wyswietl_zaleznosci_funkcyjne;
var i:integer; Pom:pf;Pom1:pa;
begin
  for i:=1 to ile_n do
    begin
      Pom:=PT^.Podaj_Element(i);
      Pom1:=Pom^.dziedzina;
      while Pom1<>nil do
        begin
          Write(Pom1^.atrybut:3);
          Pom1:=Pom1^.nast_atrybut;
        end;
        Write(' // ');
        Pom1:=Pom^.przeciwdziedzina_w;
        while Pom1<>nil do
          begin
            Write(Pom1^.atrybut:3);
            Pom1:=Pom1^.nast_atrybut;
          end;
          Writeln;
        end;
      end;
    end;

```

```

procedure Usun_pf(var P:Pointer;b:integer);far;
var Pom1,Pom2:pa;
begin
  Pom1:=pf(P)^.dziedzina;
  while Pom1<>nil do
    begin
      Pom2:=Pom1^.nast_atrybut;
      Dispose(Pom1);
      Pom1:=Pom2;
    end;
  Pom1:=pf(P)^.przeciwdziedzina_w;
  while Pom1<>nil do
    begin
      Pom2:=Pom1^.nast_atrybut;
      Dispose(Pom1);
      Pom1:=Pom2;
    end;
  Dispose(pf(P));
end;

```

```

procedure Usun_nc(var P:Pointer;b:integer);far;
var Pom1,Pom2:pa; Pom3,Pom4:pb;
begin
  Pom3:=pc(P)^.r_idf;
  while Pom3<>nil do
    begin
      Pom4:=Pom3^.nast_rola;
      Dispose(Pom3);
      Pom3:=Pom4;
    end;
  Dispose(pc(P));
end;

procedure Zal_Funk.Usun;
begin
  PT^.Usun_Tablice(Usun_pf);
  Dispose(Pt,Usun);
  NT^.Usun_Tablice(Usun_nc);
  Dispose(NT,Usun);
end;

end.

```

```

{-----}
{               Pakiet Strukt               }
{      Uniwersalne struktury danych - kolekcje      }
{-----}

```

```

unit Strukt;
interface
uses Crt, Dodatki;
const
  Max_Kolekcji = 65519 div SizeOf(Pointer);
type
  PTab_Kolekcji = ^Tab_Kolekcji;
  Tab_Kolekcji = array[1..Max_Kolekcji] of Pointer;
  typ_Proc=procedure(var A:Pointer;B:Integer);

  PObiekt1=^TObiekt1;
  TObiekt1=object
    constructor Inicjuj;
    destructor Usun;virtual;
  end;
  PKolekcja = ^TKolekcja;
  TKolekcja =
    object
      Tablica : PTab_Kolekcji;
      Ilosc : Integer;
      Ostatni : Integer;
      constructor Inicjuj(N_Ilosc : Integer);
      destructor Usun;virtual;
      function Podaj_Liczbe_Elementow : Integer;
      procedure Ustaw_Liczbe_Elementow (Nowa_Liczba : Integer);
      function Dodaj(Nastepny:Pointer) : Boolean;
      function Usun_Element(Numer:Integer; Operacja:typ_proc): Boolean;
      function Zakres (Wartosc : Integer) : Boolean;
      procedure Usun_Kolekcje;
      procedure Usun_Tablice(Operacja:typ_proc);
      function Podaj_Element(Numer_Elementu : Integer) : Pointer;
    end;

  PKolekcja1 = ^TKolekcja1;
  TKolekcja1 = object(TKolekcja)
    procedure Dla_kazdego(Operacja:Typ_Proc);
    procedure Usun_Jeden(I:Integer);
    function Usun_Element(Numer : Integer) : Boolean;
    destructor Usun;virtual;
  end;

```

implementation

```
{-----TObiekt1-----}
constructor TObiekt1.Inicjuj;
begin
end;

destructor TObiekt1.Usun;
begin
end;
{-----TKolekcja-----}
constructor TKolekcja.Inicjuj(N_Ilosc:Integer);
begin
  Pamiec( N_Ilosc * SizeOf(Pointer));
  Ilosc := N_Ilosc;
  if Zakres(N_Ilosc) then Tablica := nil
  else GetMem(Tablica,N_Ilosc*SizeOf(Pointer));
  Ostatni := 0;
end;

function TKolekcja.Zakres (Wartosc : Integer) : Boolean;
var Log : Boolean;
begin
  Log := (Wartosc > Max_Kolekcji) or (Wartosc > Ilosc) or (Wartosc <= 0) ;
  Zakres := Log;
end;

function TKolekcja.Podaj_Liczbe_Elementow : Integer;
begin
  Podaj_Liczbe_Elementow := Ostatni;
end;

function TKolekcja.Usun_Element(Numer:Integer;Operacja:typ_proc):Boolean;
begin
  if Zakres(Numer) or (Numer>Ostatni) then Usun_Element := False
  else
    begin
      Operacja(Tablica^[Numer],0); {usuwanie elementu kolekcji}
      Usun_Element :=True;
      if Numer <> Ostatni then
        Move(Tablica^[Numer+1],Tablica^[Numer],(Ostatni-Numer)*SizeOf(Pointer));
        Dec(Ostatni);
      end;
    end;
  end;
end;
```

```

procedure TKolekcja.Ustaw_Liczbe_Elementow (Nowa_Liczba : Integer);
begin
    Ostatni := Nowa_Liczba;
end;

```

```

function TKolekcja.Dodaj(Nastepny:Pointer) : Boolean;
var Pom:PTab_Kolekcji;
begin
    Dodaj:=True;
    if Ostatni< Ilosc then
        begin
            Inc(Ostatni);
            Tablica^[Ostatni] := Nastepny;
        end
    else
        begin
            Pamiec((Ilosc+1)*SizeOf(Pointer));
            if Zakres(Ilosc+1) then Dodaj:=False
            else
                begin
                    Inc(Ilosc);
                    GetMem(Pom,Ilosc*SizeOf(Pointer));
                    Move(Tablica^,Pom^,(Ilosc-1)*SizeOf(Pointer));
                    FreeMem(Tablica,(Ilosc-1)*SizeOf(Pointer));
                    Tablica:=Pom;
                    Ostatni:=Ilosc;
                    Tablica^[Ostatni] := Nastepny;
                end;
            end;
        end;
    end;

```

```

procedure TKolekcja.Usun_Kolekcje;
var i:integer;
begin
    FreeMem(Tablica,Ilosc * SizeOf(Pointer));
    Tablica := nil;
end;

```

```

procedure TKolekcja.Usun_Tablice(Operacja:typ_proc);
var i:integer;
begin
    for i:=1 to Ostatni do Operacja(Tablica^[i],0);
end;

```



```

function TKolekcja.Podaj_Element(Numer_Elementu : Integer) : Pointer;
var I : Integer;
begin
  if Zakres(Numer_Elementu) or (Numer_Elementu > Ostatni) then
    Podaj_Element := nil
  else
    Podaj_Element := Tablica^[Numer_Elementu];
  end;

destructor TKolekcja.Usun;
begin
  Usun_Kolekcje;
end;
{-----TKolekcja1-----}
procedure TKolekcja1.Usun_Jeden(I:Integer);
begin
  Dispose(PObiekt1(Tablica^[I]),Usun);
end;

function TKolekcja1.Usun_Element(Numer : Integer) : Boolean;
begin
  if Zakres(Numer) or (Numer>Ostatni) then
    Usun_Element := False
  else
    begin
      Usun_Element :=True;
      Usun_Jeden(Numer);
      if Numer <> Ostatni then
        Move(Tablica^[Numer+1],Tablica^[Numer],(Ostatni-Numer)*SizeOf(Pointer));
        Dec(Ostatni);
      end;
    end;

procedure TKolekcja1.Dla_Kazdego(Operacja : Typ_Proc);
var I:Integer;
begin
  for I:=1 to Ostatni do Operacja(Tablica^[I],I);
end;

destructor TKolekcja1.Usun;
var I:Integer;
begin
  for I:=1 to Ostatni do Usun_Jeden(I);
inherited Usun_Kolekcje;
end;
end.

```

```

{-----}
{          Program Zależności_Funkcyjne          }
{      Obiekt sprawdzający zależności funkcyjne w grafach      }
{          dwudzielnych funkcjonalnych          }
{          (Rozdział 3.2.3)          }
{-----}

```

```

program Zaleznosci_funkcyjne;
uses ModGraf;
var Zaleznosci:Zal_Funk;
begin
    Zaleznosci.Inicjuj;
    Zaleznosci.Zaleznosci_funkcyjne;
    Zaleznosci.Wyswietl_Zaleznosci_funkcyjne;
    Zaleznosci.usun;
end.

```

```

{-----}
{               Pakiet Dodatki               }
{               Podprogramy pomocnicze         }
{               (Rozdział 3.2.3)              }
{-----}

```

```
unit Dodatki;
```

```
interface
```

```
uses Crt;
```

```
function Pauza : Char;
```

```
procedure Pamiec(Rozmiar:Longint);
```

```
implementation
```

```
procedure Pamiec(Rozmiar:Longint);
```

```
begin
```

```
if MaxAvail<Rozmiar then
```

```
begin
```

```
ClrScr;
```

```
Writeln('Brak Pamieci');
```

```
Pauza;
```

```
Halt;
```

```
end;
```

```
end;
```

```
function Pauza : Char;
```

```
var Z:Char;
```

```
begin
```

```
while Keypressed do
```

```
Z:=Readkey;
```

```
Z:=Readkey;
```

```
Pauza := Z;
```

```
end;
```

```
end.
```

```

{-----}
{          Program Spójne_Składowe          }
{      Program wyznaczający liczbę składowych spójności      }
{      w dwudzielnym grafach funkcjonalnych                }
{          (Rozdział 5.2)                                    }
{-----}

```

```

program Spojne_skladowe;
uses Crt;
const n = 3;
        m = 11;
type a = set of byte;
        b = record
            d_1: byte;
            d_2: boolean;
        end;
tab = array[1..n, 1..m] of b;
{macierz przyległości dwudzielnego grafu funkcjonalnego}
const T: tab = (((d_1:1;d_2:false),(d_1:1;d_2:false),(d_1:0;d_2:false),
                    (d_1:0;d_2:false),(d_1:0;d_2:false),(d_1:0;d_2:false),
                    (d_1:0;d_2:false),(d_1:0;d_2:false),(d_1:0;d_2:false),
                    (d_1:0;d_2:false),(d_1:0;d_2:false)),
                ((d_1:1;d_2:false),(d_1:0;d_2:false),
                    (d_1:1;d_2:false),(d_1:1;d_2:false),(d_1:1;d_2:false),
                    (d_1:0;d_2:false),(d_1:0;d_2:false),(d_1:0;d_2:false),
                    (d_1:0;d_2:false),(d_1:0;d_2:false),(d_1:1;d_2:false)),
                ((d_1:0;d_2:false),(d_1:0;d_2:false),
                    (d_1:0;d_2:false),(d_1:0;d_2:false),(d_1:0;d_2:false),
                    (d_1:1;d_2:false),(d_1:1;d_2:false),(d_1:1;d_2:false),
                    (d_1:1;d_2:false),(d_1:1;d_2:false),(d_1:1;d_2:false))));

var Skladowe : integer;

procedure w_t(T: tab);
var i, j : integer;
begin
    for j:=1 to m do
        begin
            for i:=1 to n do
                Write(T[i,j].d_1:3, T[i,j].d_2:6);
                Writeln;
            end
        end;

```

```

procedure Liczba_Skladowych (var T:Tab;var Skladowe : Integer);
var i,p,j:integer;
    Tak:boolean;
begin
    Tak:=false;
    i:=1;
    P:=1;
    Skladowe:=1;
while i<= n do
    begin
        j:=1;
        while j<=m do
        begin
            if T[i,j].d_2 then Tak:=true
            else
                if (T[i,j].d_1=1) and (not T[i,j].d_2) then
                begin
                    T[i,j].d_2:=true;
                    p:=i+1;
                    while p <= n do
                    begin
                        if T[p,j].d_1=1 then T[p,j].d_2:=true;
                        p:=p+1;
                    end
                end;
                j:=j+1;
            end;
            if (not Tak) and (i>1) then Skladowe:=Skladowe+1;
            Tak:=false;
            i:=i+1;
        end;
    end;

begin
    Clrscr;
    W_t(t);
    Liczba_Skladowych(T, Skladowe);
    W_t(t);
    Writeln(Skladowe);
end.

```

mgr inż. Zofia Kruczkiewicz

Instytut Cybernetyki Technicznej
Politechniki Wrocławskiej
ul Janiszewskiego 11/17
50-372 Wrocław

Niniejszy raport otrzymują:

1. OINT	1 egz.
2. Biblioteka Główna PWr.	1 egz.
3. Z-ca Dyrektora Instytutu	1 egz.
4. Promotor	1 egz.
5. Recenzenci	2 egz.
6. Autor	3 egz.
Razem: 9 egz.	

Raport wpłynął do Redakcji ICT
w kwietniu 1999 r.