Hanna Mazur
Zygmunt Mazur

METODYKA STRUKTURALNA PROJEKTOWANIA RELACYJNYCH BAZ DANYCH

Hanna Mazur
Zygmunt Mazur

METODYKA STRUKTURALNA PROJEKTOWANIA RELACYJNYCH BAZ DANYCH

Hanna Mazur
Zygmunt Mazur

METODYKA STRUKTURALNA PROJEKTOWANIA RELACYJNYCH BAZ DANYCH

Wydanie drugie, poprawione i rozszerzone, 2020.
Wydział Informatyki i Zarządzania Politechniki Wrocławskiej

Wydanie pierwsze: Hanna Mazur, Zygmunt Mazur, *Projektowanie relacyjnych baz danych.* Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2004.

Recenzent
Jacek Gruber

Korekta i DTP
Atelier 2

Fotografia na okładce
Hanna Mazur

ISBN 978-83-958341-0-3

SPIS TREŚCI

Projektowanie baz danych jest bardziej sztuką niż nauką ścisłą;
wymaga tyle samo intuicji, co wiedzy teoretycznej i praktycznej.

Michael J. Hernandez

PRZEDMOWA DO WYDANIA DRUGIEGO

Pół wieku temu, w roku 1970, za sprawą brytyjskiego informatyka Edgara Franka Codda, który opublikował pracę na temat relacyjnego modelu danych (ang. *relational data model*), zmieniło się postrzeganie danych i podejście do projektowania baz danych, zwanych wówczas bankami danych (ang. *data bank*). Od tego czasu nastąpił ogromny rozwój w obszarze nauk i technologii teleinformatycznych. Powszechność komputerów i urządzeń mobilnych, informatyzacja i cyfryzacja społeczeństwa mają duży wpływ na tworzenie nowych baz i systemów baz danych, aplikacji webowych i mobilnych, które przetwarzają dane przechowywane w różnego rodzaju bazach danych, w tym bardzo często w bazach relacyjnych. Systemy zarządzania relacyjnymi bazami danych są bardzo szybkie, stabilne i bezpieczne. Z tego powodu logiczne modele obiektowe są w wielu przypadkach transformowane do relacyjnych baz danych.

Niniejsza książka jest drugim wydaniem – poprawionym, uzupełnionym i rozszerzonym – książki z 2004 roku, zatytułowanej *Projektowanie relacyjnych baz danych*, tych samych autorów [MazMaz2004]. Od pierwszego wydania minęło zatem szesnaście lat, ale przedstawiona metodyka jest nadal aktualna i stosowana, szczególnie w małych projektach i w projektach studenckich. Według przedstawionej w książce strukturalnej metodyki projektowej wykonano, w ramach różnych kursów realizowanych na kierunku informatyka na Wydziale Informatyki i Zarządzania Politechniki Wrocławskiej, tysiące (co najmniej 3 000) studenckich projektów

baz danych dotyczących szerokiego spektrum zastosowań. W wielu przypadkach zaprojektowane bazy danych zostały z powodzeniem zaimplementowane. Zaimplementowano także i wdrożono wiele dobrze działających aplikacji i systemów do obsługi tych baz.

Jednym z takich systemów jest dostępny przez przeglądarkę internetową system do obsługi ogólnopolskiego konkursu na najlepsze prace magisterskie z informatyki, organizowanego od 1984 roku przez Polskie Towarzystwo Informatyczne [MaNoRaSz2020]. Pierwszą, jeszcze okrojoną wersję systemu wdrożono w 2017 roku. Do tego czasu dane ze wszystkich dotychczasowych trzydziestu czterech edycji konkursu były przechowywane w bazie zaprojektowanej według przedstawionej w tej książce metodyki i zaimplementowanej w 2002 roku przez ówczesnych studentów informatyki, a następnie utrzymywanej, rozwijanej i pielęgnowanej przez jednego z jej autorów aż do migracji danych w 2017 roku [Mazur2017]. Tak długi okres użytkowania bazy dobrze świadczy o poprawności i skuteczności proponowanej metodyki, wysokiej jakości wykonanego projektu, poprawnej implementacji bazy danych i systemu obsługi. Po migracji danych z poprzedniej bazy danych do nowej i po rozbudowie systemu, internetowy system *kpm.org.pl* w obecnej wersji realizuje wszystkie wymagania funkcjonalne dla pięciu grup użytkowników: organizatorów, komisji konkursowej, recenzentów, autorów prac i internautów.

Od 2004 roku wiele się zmieniło w zakresie metodyk projektowania baz danych i wytwarzania oprogramowania, narzędzi i języków programowania, technologii teleinformatycznych. Zmieniono lub wprowadzono nowe akty prawne, które w sposób mniej lub bardziej bezpośredni mają związek z istniejącymi lub tworzonymi bazami danych i systemami baz danych. Wiele zmian wymusiło m.in. unijne rozporządzenie o ochronie danych osobowych – RODO (ang. *General Data Protection Regulation* – GDPR), obowiązujące od 25 maja 2018 roku.

Wraz ze wzrostem powszechności systemów informatycznych i aplikacji mobilnych rosną także wymagania i oczekiwania użytkowników co do jakości, wydajności, zakresu możliwości i ogólnie funkcjonalności systemów i baz danych. Oczywiście każda baza danych musi spełniać podstawowe wymagania dotyczące prawidłowego i wydajnego przetwarzania danych oraz ich bezpieczeństwa (integralności, spójności, poufności i dostępności).

Przedstawiona w 2004 roku autorska strukturalna metodyka projektowania relacyjnych schematów baz danych, pomimo zmian zachodzących w otoczeniu, jest nadal z powodzeniem stosowana. Zasady przygotowywania opisów i dokumentowania wszystkich etapów projektowania bazy danych zostały w praktyce pozytywnie zweryfikowane.

W obecnym wydaniu dopracowano opis metodyki projektowania. Przy zachowaniu zasadniczych jej założeń, wprowadzono dodatkowe zalecenia dotyczące poszczególnych etapów projektowych (sposobu realizacji, dokumentowania i weryfikowania), wskazówki i rozszerzenia, które mają na celu szybsze wytwarzanie projektów lepszej jakości, np. zdefiniowano precyzyjniejsze definiowanie związków oraz wprowadzono weryfikowanie za pomocą macierzy powiązań poprawności i kompletności przypisania transakcji do perspektyw użytkowników. Zamieszczono także szablon wykorzystywany do realizacji projektu w proponowanej metodyce.

Zachowując zatem pierwotną istotę metodyki, przeprowadzono korektę całości materiału (teoretycznego i zamieszczonych przykładów) oraz uporządkowano go i rozbudowano, wzbogacając książkę nowymi treściami, opisami, przykładami, rysunkami i tabelami. Celem tych zmian było ułatwienie samodzielnego analizowania przedstawionych zagadnień i maksymalne podniesienie poziomu czytelności przedstawionego materiału.

Mamy nadzieję, że książka w obecnym kształcie będzie jeszcze bardziej zrozumiała, jej lektura sprawi Czytelnikowi przyjemność i satysfakcję, a ponadto umożliwi wykonywanie poprawnych projektów baz danych.

Umiejętność określa to, co jesteś w stanie zrobić.
Motywacja determinuje to, co robisz.
Podejście determinuje, jak dobrze to robisz.

Lou Holtz

WSTĘP

Prawie w każdej organizacji (przedsiębiorstwie, instytucji) niezależnie od jej wielkości konieczne jest odpowiednie przechowywanie przetwarzanych danych, umożliwiające ich szybkie wyszukiwanie, dopisywanie, aktualizację i usuwanie, generowanie raportów i archiwizowanie. Dotyczy to także osób prowadzących działalność gospodarczą oraz wszystkich zainteresowanych gromadzeniem i przetwarzaniem różnych danych firmowych (o klientach, fakturach, ofertach i cennikach) bądź danych tematycznych, na przykład o podróżach, zdrowiu czy treningach.

Ogólna dostępność i powszechność komputerów zachęca do przechowywania danych w postaci elektronicznej oraz do przechowywania danych w coraz szerszym zakresie, z przeróżnych obszarów. Dzięki nowoczesnym technologiom i różnorodnym systemom zarządzania bazami danych możliwe i opłacalne jest gromadzenie i przetwarzanie zarówno małych, jak i bardzo dużych zbiorów danych, związanych praktycznie z każdą dziedziną życia. Coraz więcej jest gromadzonych danych osobistych dotyczących zdrowia, diety, trybu życia. Firmy, szpitale, banki, biblioteki, urzędy, uczelnie, sklepy itd. odchodzą od przechowywania danych w wersji papierowej. Korzystanie z elektronicznych baz danych jest obecnie powszechne i konieczne.

Systemy zarządzania bazami danych (SZBD, ang. *Database Management System* – DBMS) umożliwiają szybkie zakładanie baz danych, proste formułowanie zapytań, szybkie wyszukiwanie danych i generowanie raportów.

Implementowana baza danych (ang. *database*) musi być jednak wcześniej poprawnie zaprojektowana. Celem niniejszej książki jest przedstawienie zagadnień związanych z relacyjnym modelem danych i relacyjnymi bazami danych oraz zwrócenie uwagi Czytelnika na podstawowe aspekty związane z wykonaniem poprawnego projektu bazy według zaproponowanej metodyki.

W książce przedstawiono autorską metodykę strukturalną projektowania relacyjnych baz danych i szczegółowo omówiono wszystkie niezbędne etapy umożliwiające opracowanie poprawnego logicznego modelu bazy danych.

Jakość wykonania poszczególnych etapów ma bardzo istotny wpływ na efekt końcowy, czyli na ostateczną postać bazy danych. Dbałość o wykonanie poszczególnych etapów zgodnie z przedstawionymi zasadami powinna doprowadzić do otrzymania poprawnego schematu bazy danych. Bywa jednak, że produkt końcowy nie spełnia oczekiwań użytkowników, jeśli np. na etapie analizy przyjęto początkowe błędne założenie lub w trakcie wykonywania projektu nastąpiły istotne zmiany w danym obszarze projektowym lub otoczeniu i nie zostały one uwzględnione.

Wiele faktów świata rzeczywistego pozornie nieistotnych, źle opisanych, nie dostrzeżonych lub pominiętych przez projektanta ma bardzo istotny wpływ na ostateczną postać bazy danych. Na przykład reguła, że student musi mieć zaliczenie z kursu *Bazy danych*, oznacza, iż z chwilą zapisania danych studenta do bazy jednocześnie trzeba wpisać ocenę z kursu, co nie jest zgodne z rzeczywistością, ponieważ ocenę wystawia się dopiero na koniec semestru czwartego. Często potoczne formułowanie wymagań jest niejednoznaczne bądź wręcz niepoprawne.

Prawidłowe wykonanie wszystkich etapów projektowania, opisanych w kolejnych rozdziałach niniejszej pracy, pozwoli na sporządzenie dobrego i w pełni udokumentowanego projektu bazy danych. Zaprojektowana baza danych jest tak dobra jak jej najsłabszy element.

Z relacyjnym modelem danych wiążą się zagadnienia logicznej definicji (struktury) danych, operowania danymi za pomocą podstawowych operatorów algebry relacji (selekcji, rzutu i złączenia) oraz zapewnienia integralności danych (poprzez określenie reguł zachowania poprawności danych).

Najczęściej do pracy z relacyjnymi bazami danych wykorzystuje się deklaratywny strukturalny język zapytań SQL (ang. *Structured Query Language*), w którym każde polecenie można przypisać do jednego z czterech podzbiorów:

- języka definiowania danych (ang. *Data Definition Language* – DDL),
- języka manipulowania danymi (ang. *Data Manipulation Language* – DML),
- języka definiowania zapytań (ang. *Data Query Language* – DQL),
- języka kontroli danych (ang. *Data Control Language* – DCL).

Instrukcje języka SQL (którego jest bardzo wiele dialektów) umożliwiają wykonywanie operacji zdefiniowanych w algebrze relacji.

Książka jest przeznaczona dla wszystkich osób zainteresowanych zagadnieniami projektowania baz danych, zarówno dla początkujących projektantów, jak również – ze względu na kompletność materiału – może być przydatna dla osób doświadczonych w projektowaniu baz. Może być także wykorzystywana jako podręcznik dydaktyczny w ramach zajęć z projektowania relacyjnych baz danych. Stanowi wprowadzenie do zaawansowanego projektowania strukturalnego relacyjnych baz danych narzędziami programowymi CASE służącymi do komputerowego wspomagania inżynierii oprogramowania (ang. *Computer Aided Software Engineering*) oraz do komputerowego wspomagania inżynierii systemów (ang. *Computer Aided Systems Engineering*), wspierającymi analizę i projektowanie systemów informatycznych wysokiej jakości.

Zaprezentowana w książce metodyka ma charakter ręcznego projektowania metodą tworzenia dokumentów projektowych.

Treść książki zawiera materiał realizowany w ramach kursów z baz danych prowadzonych dla studentów uczelni wyższych.

W rozdziale pierwszym podano krótką charakterystykę wybranych etapów projektowania systemów baz danych, w tym baz danych. W rozdziale drugim przedstawiono podstawowe pojęcia i zagadnienia dotyczące modelu relacyjnego takie jak relacja, schemat relacji, operatory algebry relacji, proces normalizacji schematów relacji. W rozdziale trzecim omówiono etapy projektowania konceptualnego od analizy wycinka rzeczywistości do konceptualnego modelu danych przedstawionego w graficznej postaci za pomocą diagramu związków encji. Rozdział czwarty dotyczy modelowania logicznego – zawiera opis podstawowych zasad transformacji modelu konceptualnego do relacyjnego modelu logicznego oraz omówienie zagadnień związanych z definiowaniem perspektyw. Po rozdziale piątym zawierającym podsumowanie książki, zamieszczono spis literatury, a następnie trzy dodatki. W „Dodatku A" podano wybrane definicje bazy danych dostępne w literaturze. W „Dodatku B" znajduje się wykaz wszystkich etapów projektowania bazy w zaprezentowanej metodyce, a w „Dodatku C" szablon wykorzystywany do realizacji projektu zgodnie z proponowanymi w metodyce etapami opisanymi w książce.

Autorzy dziękują recenzentowi za komentarze, sugestie i rady.

Jakość jest za darmo. To brak jakości kosztuje.

Philip Crosby

1. ETAPY PROJEKTOWANIA BAZ DANYCH

Materiał zawarty w książce dotyczy najważniejszych pojęć z teorii relacyjnych baz danych oraz etapów projektowania relacyjnych baz danych według autorskiej strukturalnej metodyki projektowej, w której zrealizowano kilka tysięcy projektów studenckich. Obecnie jest dostępnych wiele publikacji dotyczących tematyki baz danych, ale są to na ogół pozycje bardzo obszerne (kilkaset stron), poruszające wiele zagadnień z teorii baz danych, nie zawsze jednak bezpośrednio dotyczących projektowania baz danych, co utrudnia wydobycie informacji istotnych z punktu widzenia projektanta bazy danych.

Poprawnie zaprojektowana baza danych musi spełniać oczekiwania i wymagania potencjalnych użytkowników, a jednocześnie musi gwarantować bezbłędne, wydajne i wygodne w obsłudze funkcjonowanie. Takie własności można zapewnić, jeśli poprawnie przeprowadzi się wszystkie etapy projektowania bazy danych.

W metodykach strukturalnych stosuje się podejście zstępujące – od ogółu do szczegółu (ang. *top-down*). Definiuje się kolejne precyzyjnie określone etapy (czynności do wykonania), które są z sobą powiązane, a każdy z nich polega na wytworzeniu odpowiedniego produktu (dokumentu). Metodyki strukturalne są powszechnie stosowane, chociaż dużą popularnością cieszą się także inne podejścia, na przykład obiektowe.

Metodyka przedstawiona w tej książce spełnia wszystkie wymienione cechy. Dla każdego etapu jest jasno określony cel – czemu dany etap służy, jak go wykonać i udokumentować, jak przygotować przykładowe dane i sprawdzić poprawność etapu

w kontekście całości (wszystkich etapów). W każdym etapie, stosując odpowiednią notację i zdefiniowane pojęcia, wykonuje się ściśle określone działania, których wynikiem jest konkretny produkt zapisany w języku naturalnym, w formie tabel bądź w postaci diagramu. Weryfikację poprawności etapu można przeprowadzić empirycznie, za pomocą danych przykładowych. Proponowana metodyka składa się z trzynastu etapów. Etap 9 kończy modelowanie konceptualne danych, a etap 13 – modelowanie logiczne bazy danych.

Opisy kolejnych etapów metodyki projektowania relacyjnych baz danych zostały przedstawione w sposób systematyczny i uporządkowany, możliwie pełny, a zarazem prosty i zrozumiały. Intencją autorów było zwrócenie uwagi na konieczność znormalizowanego sposobu dokumentowania poszczególnych etapów („Dodatek C"), definiowania wszystkich używanych pojęć i stosowanej notacji, umiejętnego wyboru danych przykładowych oraz systematycznego weryfikowania zgodności różnych etapów i, w razie potrzeby, poprawiania projektu od początku. Takie wykonanie i udokumentowanie projektu wymaga rzetelnej i systematycznej pracy, ale ostatecznie daje duże korzyści – opracowana dokumentacja końcowa projektu bazy danych jest czytelna, zrozumiała i jednoznaczna dla wszystkich osób nią zainteresowanych i może być wykorzystana do implementacji bazy (a także aplikacji).

W prawidłowo zaprojektowanej bazie danych powinna być możliwość trwałego przechowywania wszystkich (ale tylko niezbędnych) danych oraz udostępniania ich uprawnionym użytkownikom. Struktura bazy powinna umożliwiać efektywną pracę z zaimplementowaną bazą danych oraz zapewniać możliwość jej modyfikacji i rozbudowy. Parafrazując radę Alberta Einsteina – zaprojektowana baza danych powinna być tak prosta jak to tylko możliwe, ale nie prostsza.

W projektowaniu relacyjnych baz danych wyróżnia się:

- **Projektowanie konceptualne** (ang. *conceptual design*), określane także jako koncepcyjne, pojęciowe; polega ono na zdefiniowaniu wymagań niezależnych od implementacji. Proces zbierania wymagań powinien doprowadzić do powstania konceptualnego modelu danych (ang. *Conceptual Data Model* – CDM), zwykle przedstawianego w postaci graficznej za pomocą **diagramu związku encji** (ang. *Entity Relationship Diagram* – ERD). Opracowany model jest niezależny od docelowej, implementacyjnej platformy sprzętowej.

- **Projektowanie logiczne** (ang. *logical design*), polegające na transformacji modelu konceptualnego do logicznego modelu danych (ang. *Logical Data Model* – LDM) przedstawiającego **relacyjny schemat bazy danych**, który

jest zbiorem schematów relacji definiującym logiczną strukturę danych (nie implementacyjną). Otrzymane schematy relacji powinny być co najmniej w trzeciej postaci normalnej i, przy prawidłowym wykonaniu projektu, warunek ten jest spełniony, a jeśli w wyniku popełnionych błędów tak nie jest, to przeprowadza się normalizację schematów relacji. Normalizacja jest sformalizowaną procedurą, w wyniku której eliminuje się redundancje (powtarzanie się) danych, ich niespójności i anomalie, czyli nieprawidłowości związane z aktualizacją, usuwaniem i dopisywaniem danych.

- **Projektowanie fizyczne** (ang. *physical design*), w wyniku którego powstaje fizyczny model danych (ang. *Physical Data Model* – PDM), zależy od planowanego środowiska implementacyjnego. Elementy logicznego projektu bazy danych, czyli schematy relacji, są przekształcane w obiekty bazy danych (strukturę implementacyjną bazy), np. w tabele w relacyjnej bazie danych. Dla zaimplementowanej bazy można zaprojektować i zaimplementować aplikacje realizujące transakcje zdefiniowane w projekcie bazy danych.

Główne etapy tworzenia systemu bazy danych – z punktu widzenia projektanta – obejmujące etapy projektowania i implementacji bazy danych oraz projektowania i implementacji aplikacji, przedstawiono graficznie na rysunku 1.1.

Zaprojektowana baza danych może być zaimplementowana w wybranym systemie zarządzania bazą danych. W różnych SZBD struktury baz są rozmaite, dane są przechowywane w odmienny sposób, dziedzinom atrybutów mogą odpowiadać specyficzne typy danych itd. Ten poziom szczegółowości na ogół jest nieistotny dla projektanta bazy danych, którego interesuje jedynie opracowanie logicznego modelu danych oraz określenie sposobu dostępu do danych dla wszystkich potencjalnych użytkowników.

Po prawidłowo przeprowadzonym procesie projektowania bazy danych można **zaprojektować aplikację** (lub kilka aplikacji) obsługującą bazę danych w sposób poprawny, efektywny i ergonomiczny. Wiedząc już (z projektu bazy danych), *co aplikacja ma robić*, należy podjąć decyzję, *jak to zrobić*. Po zaprojektowaniu aplikacji można ją **zaimplementować** w odpowiednio dobranym środowisku programistycznym na określoną platformę sprzętową. System bazy danych (baza danych i aplikacje) musi być **przetestowany** i **udokumentowany** według wymaganego standardu. **Dokumentacja** w zależności od przeznaczenia (rodzaju) może mieć różną strukturę, ale zazwyczaj zawiera istotne opisy etapów projektowania i implementacji bazy danych oraz aplikacji, jak również opisy konfiguracyjne, wymagania instalacyjne i instrukcję obsługi systemu.

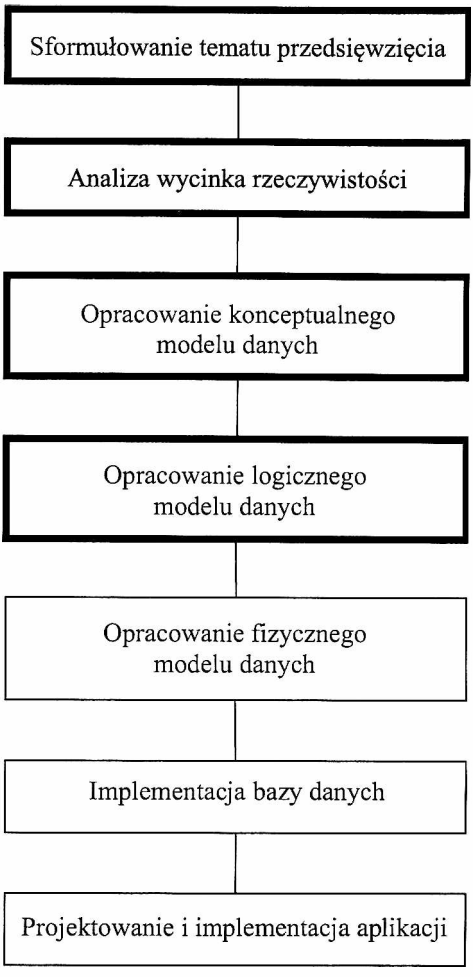
Elementy rysunku 1.1 narysowane linią pogrubioną odpowiadają etapom omówionym w tej książce, czyli etapom projektowania relacyjnej bazy danych, których efektem końcowym jest logiczny model bazy danych, będący abstrakcyjną reprezentacją struktury danych odizolowaną od szczegółów implementacyjnych, systemu zarządzania bazą danych, sposobem pamiętania danych itd.

Podstawą tworzenia każdej poprawnie zaprojektowanej bazy danych jest **analiza wymagań** (ang. *requirements analysis*), obejmująca proces identyfikacji i opisywania codziennych czynności wykonywanych w analizowanej rzeczywistości. Semantyka świata rzeczywistego musi wystąpić w bazie danych. Konieczne jest uwzględnienie wszystkich wymagań (funkcjonalnych i niefunkcjonalnych) i potrzeb przyszłego użytkownika [Jaszk1997, Sacha2010]. Zasadniczą częścią analizy wymagań jest przeprowadzanie wywiadów z odpowiednimi osobami, w tym z ekspertami dziedzinowymi i z przyszłymi potencjalnymi użytkownikami bazy danych. Projektując bazę, należy mieć na uwadze funkcje, jakie będzie się na tych danych wykonywać. Zły projekt bazy danych bardzo komplikuje zaprojektowanie poprawnie działającej aplikacji spełniającej wszystkie oczekiwania użytkownika.

Poprawnie wykonane etapy projektowania bazy danych odwzorowują pojęcia, fakty i procesy biznesowe ze świata rzeczywistego w dane w bazie danych. Identyfikacja zakresu odwzorowywanego wycinka rzeczywistości wymaga wiedzy na jego temat. Przekształcanie zdarzeń i faktów z wycinka świata rzeczywistego (wyrażanych najczęściej opisami w języku naturalnym) w dane nie odbywa się bezpośrednio – wymaga wykonania wielu udokumentowanych czynności, w tym opracowania modelu konceptualnego. Wykonywane etapy wymagają wielokrotnego analizowania, uzgadniania i weryfikowania ich poprawności, stąd powinny być przedstawiane w jak najprostszej, jednoznacznej i czytelnej postaci, najlepiej – jeśli to możliwe – za pomocą diagramów, rysunków i tabel.

W zaproponowanej metodyce najważniejszym diagramem jest diagram związków encji, który jest punktem odniesienia dla wielu innych etapów. Graficzny sposób jego przedstawienia ułatwia transformację do modelu logicznego – schematu bazy danych, czyli końcowego produktu udostępnianego do dalszych etapów: implementacji bazy i aplikacji. Z otrzymanego schematu logicznego możliwe jest przejście odwrotne – do modelu konceptualnego, a także, na podstawie semantycznych związków między danymi, możliwe jest odtworzenie faktów ze świata rzeczywistego.

W starszych systemach baz danych, oprogramowanie obsługujące bazę danych często było uzależnione od danych, czyli sposób przechowywania fizycznych danych i metody dostępu były określane (narzucane) przez wymagania danej aplikacji. Wiedza o organizacji danych i metodach dostępu była wbudowywana



Rys. 1.1. Wybrane etapy tworzenia systemu bazy danych

w logikę i kod aplikacji. Aplikacje były zależne od danych, w związku z czym nie można było zmieniać ani struktury przechowywanych danych, ani metod dostępu. Nowoczesne systemy zarządzania bazami danych pozwalają zapewnić niezależność aplikacji od danych, rozumianą jako odporność aplikacji na zmiany w strukturze przechowywanych danych i sposobie dostępu, co znaczy, że zmiany te nie wymagają już zmian w oprogramowaniu aplikacji. Postulat niezależności danych pozwala na oddzielenie wykonania projektu bazy danych od projektu aplikacji.

Jakość opracowanego systemu bazy danych zależy od poprawności i jakości projektu bazy danych oraz od wykonanego oprogramowania. Przedstawiony w książce materiał dotyczy projektowania baz danych, podkreślmy jednak, że

wytworzona w metodyce dokumentacja projektu bazy obejmuje wiele opisów i szczegółów istotnych podczas projektowania i implementacji aplikacji, m.in. zidentyfikowanych użytkowników i ich wymagania funkcjonalne, reguły funkcjonowania i ograniczenia dziedzinowe, schemat bazy danych, definicje transakcji i perspektyw.

Źle zaprojektowaną bazę danych można oczywiście zaimplementować, ale zazwyczaj pojawiają się potem duże trudności z utrzymaniem spójności danych, występują problemy podczas projektowania i implementacji aplikacji oraz podczas koniecznej rozbudowy czy modyfikacji systemu. Najczęściej trzeba wówczas wykonywać dodatkowe czynności związane z obsługą anomalii aktualizowania danych, dopisywania danych do bazy oraz ich usuwania. Aby baza danych była poprawnie zaprojektowana, należy bardzo dokładnie wyodrębnić i przeanalizować, a następnie uwzględnić w projekcie **reguły biznesowe** (ang. *business rules*). Z zapisanych w języku naturalnym reguł wynika sposób tworzenia, modyfikowania i usuwania danych oraz warunki, jakie te dane muszą spełniać. Zagadnienia te zostaną omówione w rozdziale trzecim.

Nawet w wypadku bardzo prostych baz danych, jeśli są źle zaprojektowane, mogą wystąpić problemy podczas wykonywania podstawowych operacji dodawania, modyfikowania i usuwania danych. Rozważmy prostą bazę danych **PANORAMA FIRM** składającą się tylko z jednego schematu relacji **Panorama**(*Firma*, *Adres*, *Towar*, *Cena*). W bazie mają być gromadzone następujące dane: unikalne nazwy firm, ich adresy, dostarczane przez nie towary i aktualne ceny tych towarów. W tak źle zaprojektowanej bazie danych będą występowały anomalie dopisywania danych, usuwania i aktualizacji oraz redundancja danych. Zauważmy, że adres firmy będzie się powtarzał tyle razy, ile różnych towarów dostarcza dana firma. Zmiana adresu firmy będzie wymagała aktualizowania wielu wpisów, w zależności od liczby oferowanych produktów. Kluczem głównym w schemacie relacji **Panorama** jest dwuelementowy zbiór atrybutów {*Firma*, *Towar*}, w związku z czym nie będzie można wprowadzić danych o firmie, dopóki firma nie będzie miała w swojej ofercie co najmniej jednego towaru. Ponadto, jeśli firma przestanie dostarczać towary, to jej dane trzeba usunąć z bazy (nie może być firmy bez towarów) i w ten sposób traci się dane, które być może należałoby nadal przechowywać. Wszystkie te anomalie mogą być wykluczone po wykonaniu poprawnego projektu bazy danych.

W kolejnych rozdziałach zostaną przedstawione wszystkie etapy konceptualnego i logicznego projektowania baz danych.

Najpierw jednak omówimy podstawowe pojęcia z teorii relacyjnych baz danych.

Świat wcale nie jest zwariowany,
choć nie jest dla ludzi normalnych.
Jest dla znormalizowanych.

Stanisław Jerzy Lec

2. PODSTAWY RELACYJNYCH BAZ DANYCH

Baza danych (ang. *database*) jest to trwały zbiór tematycznie z sobą powiązanych danych (zob. „Dodatek A"). Inaczej mówiąc, baza danych to dane i powiązania między nimi. Pojęcie bazy danych zostało sformułowane w latach 1962–1963.

Podczas projektowania bazy danych tworzone są kolejne modele danych (ang. *data model*): konceptualny (najbardziej abstrakcyjny), logiczny (w którym nie ma żadnych odniesień do fizycznej struktury bazy danych) i fizyczny (implementacyjny).

W modelu danych definiuje się:

- strukturę danych (część statyczna modelu),
- operacje na danych (część dynamiczna modelu).

Wyróżnia się następujące podstawowe modele danych:

- hierarchiczny,
- sieciowy,
- relacyjny,
- obiektowy.

Obszerne omówienie podstawowych modeli danych można znaleźć w literaturze [Allen2003, Date1981, Hernan1998, Pankow1992, UllWid2000].

W niniejszej książce skupimy się na **modelu relacyjnym** (ang. *relational model*). Twórcą relacyjnego modelu danych był E.F. Codd, który w 1970 roku opublikował fundamentalną pracę na ten temat [Codd1970].

2.1. RELACYJNY MODEL DANYCH

Relacyjny model danych (ang. *relational data model*) jest opisywany za pomocą pojęć takich jak: atrybut, dziedzina, schemat relacji, relacja, krotka, stopień i liczność relacji. Ich definicje i przykłady podamy w dalszej części pracy. Model ten jest wykorzystywany do projektowania i implementacji **relacyjnych baz danych** (ang. *relational database*), szczegółowo opisanych w dostępnej literaturze, np. [Date2000, DelAdi1989, GaUlWi2003, UllWid2011]).

W dalszej części rozdziału podamy definicje podstawowych pojęć dotyczących relacyjnego modelu danych.

Relacja *R* (ang. *relation*) w sensie matematycznym jest podzbiorem iloczynu kartezjańskiego n (n>0) dowolnych zbiorów wartości, co zapisujemy jako:

$$R \subseteq D_1 \times D_2 \times \ldots \times D_n$$

gdzie D_i ($i = 1, \ldots, n$) jest dowolnym zbiorem wartości – **dziedziną** (ang. *domain*).

Elementami iloczynu kartezjańskiego są n-krotki (zwane też po prostu **krotkami**) wartości $(d_1, d_2, \ldots, d_n)$, przy czym

$$d_i \in D_i, \text{ dla } i = 1, \ldots, n.$$

Aby uniezależnić relację od uporządkowania dziedzin, w relacyjnym modelu danych przyjęto inną (zmodyfikowaną) konwencję notacyjną.

Niech $U = \{A_1, \ldots, A_n\}$ będzie zbiorem elementów zwanych atrybutami i dla każdego $A \in U$ przyporządkowany jest pewien zbiór wartości dom(A) zwany **dziedziną** (domeną) atrybutu.

Dziedzina (ang. *domain*) – dom(A) – zbiór dopuszczalnych wartości atrybutu, w tym wartość pusta NULL. Wszystkie wartości w dziedzinie są proste (atomowe).

Atrybut (ang. *attribute*) – opisuje użycie pewnej dziedziny w danej relacji. Atrybuty mają nazwy i mogą przyjmować wartości z ustalonego zbioru – dziedziny atrybutu. Aby zaznaczyć różnicę pomiędzy pojęciem *atrybut* a pojęciem *dziedzina*, często nadaje się atrybutom inne nazwy niż dziedzinom, z których się wywodzą [Date1981].

Krotka (ang. *tuple*) – zbiór par postaci (nazwa atrybutu : wartość atrybutu), czyli:

$t = \{A_1 : d_1, \ldots, A_n : d_n\}$, gdzie $\text{dom}(A_i) = D_i$ oraz $d_i \in D_i$ dla $i = 1, \ldots, n$.

Kolejność krotek w relacji jest dowolna (nie ma znaczenia).

Składowa krotki (ang. *component of tuple*) – wartość atrybutu w krotce.

Schemat relacji (ang. *relation scheme*) – specyfikacja relacji, obejmuje nazwę schematu relacji oraz zbiór atrybutów relacji o określonych (niekoniecznie różnych) dziedzinach, co zapisujemy:

$$\mathbf{R}(\{A_1 : D_1, \ldots, A_n : D_n\}).$$

Z powyższej definicji wynika, że kolejność atrybutów w schemacie relacji jest dowolna (nie ma znaczenia). Nazwy atrybutów w schemacie relacji muszą być różne.

Relacja *R* (ang. *relation*) o schemacie **R** w relacyjnym modelu danych jest zbiorem dowolnej skończonej liczby krotek $\{t_1, \ldots, t_s\}$:

$$R = \{t_1, \ldots, t_s\}, \text{ gdzie } s \geq 0.$$

Schemat relacji **R** wyznacza relacje postaci

$$(A_{i_1} : D_{i_1}) \times \ldots \times (A_{i_n} : D_{i_n}),$$

gdzie $i_1, \ldots, i_n \in [1, n]$, $\text{dom}(A_i) = D_i$ (dla $i = 1, \ldots, n$) oraz dla każdego $k \neq j$ (gdzie k, $j \in [1, n]$) $i_k \neq i_j$, czyli ciąg $i_1, \ldots, i_n$ jest dowolną permutacją liczb $1, \ldots, n$.

Relacja może być nazwana i wówczas jest identyfikowana przez ***nazwę relacji***.

Dla graficznego rozróżnienia nazwy schematu relacji i nazwy relacji (bardzo często nazwa jest taka sama) nazwy schematów relacji będą zapisywane czcionką pogrubioną (np. **R**), a nazwy relacji czcionką pochyłą (np. *R*).

Instancja relacji (ang. *instance of relation*) – zbiór konkretnych krotek relacji *R* (aktualne dane) o schemacie relacji **R**.

Dla relacji *FakturyZakupu* i *FakturySprzedaży* o schemacie relacji **Faktury** zbiór krotek z danymi faktur zakupu (stan w danej chwili) jest instancją relacji *FakturyZakupu*, a z fakturami sprzedaży – instancją relacji *FakturySprzedaży*.

Przykład 2.1

Rozważmy trzy zbiory krotek (instancje relacji) o schemacie

Osoby({*Id* : *Integer*, *Nazwisko* : *String*, *Imię* : *String*})

Z1:

{{*Id* : 1, *Nazwisko* : "Kowalski", *Imię* : "Jan"},
{*Id* : 2, *Nazwisko* : "Nowak", *Imię* : "Adam"}},

Z2:

{{*Id* : 2, *Imię* : "Adam", *Nazwisko* : "Nowak"},
{*Id* : 1, *Imię* : "Jan", *Nazwisko* : "Kowalski"}},

Z3:

{{*Id* : 2, *Nazwisko* : "Nowak", *Imię* : "Adam"},
{*Id* : 1, *Nazwisko* : "Kowalski", *Imię* : "Jan"}}.

Z definicji relacji i schematu relacji wynika, że te trzy instancje relacji są takie same – mają ten sam schemat relacji i takie same składowe krotek.

Jeśli z kontekstu i z nazw atrybutów wynika dziedzina atrybutów, to zamiast zapisu $\mathbf{R}(\{A_1: D_1, \ldots, A_n: D_n\})$ będziemy stosowali uproszczony zapis $\mathbf{R}(\{A_1, \ldots, A_n\})$.

Pamiętając, że kolejność atrybutów w schemacie relacji $\mathbf{R}(\{A_1, \ldots, A_n\})$ nie ma znaczenia, można wprowadzić kolejne uproszczenie i stosować zapis $\mathbf{R}(A_1, \ldots, A_n)$, przyjmując wskazane uporządkowanie atrybutów w schemacie relacji i takie samo uporządkowanie wartości w krotkach relacji R o schemacie $\mathbf{R}(A_1, \ldots, A_n)$.

Zapis $\text{sch}(R) = \mathbf{R}(A_1, \ldots, A_n)$ oznacza, że relacja R ma schemat $\mathbf{R}(A_1, \ldots, A_n)$.

Nazwa kwalifikowana atrybutu – zapis postaci $R.A$, czyli atrybut A z relacji R.

Uwzględniając powyższe uwagi, zamiast zapisu schematu relacji

Osoby({*Id* : *Integer*, *Nazwisko* : *String*, *Imię* : *String*})

można napisać

Osoby(*Id*, *Nazwisko*, *Imię*)

a przykładową krotkę zamiast {*Id* : 1, *Nazwisko* : "Kowalski", *Imię* : "Jan"} zapiszemy jako (1, "Kowalski", "Jan").

Stopień relacji (ang. *degree*) – liczba atrybutów relacji. Relację stopnia pierwszego (o jednym atrybucie) nazywamy relacją unarną, drugiego – binarną, trzeciego – ternarną, relację stopnia n-tego nazywamy relacją n-arną.

Liczność relacji (ang. *cardinality*) – liczba krotek w relacji (wartość całkowita nieujemna). Liczność relacji R zapisuje się jako $\text{card}(R)$. Relacja pusta ma liczność 0.

Dane w relacji mogą zmieniać swoją wartość (bez zmiany schematu relacji). Naturalnym sposobem reprezentowania relacji jest tabela dwuwymiarowa, w której kolumny odpowiadają atrybutom relacji, a wiersze – krotkom. Kolejność wierszy w tabeli jest dowolna. Reprezentując relację w formie tabeli, przyjmujemy, że nazwy i kolejność kolumn odpowiadają nazwom i kolejności atrybutów w relacji.

Przykład 2.2

Załóżmy, że interesują nas cechy samochodów takie jak:

marka, kolor, rok produkcji, spalanie, numer rejestracyjny, data rejestracji,

gdzie:

marka – przyjmuje wartości ze zbioru nazw marek, np. Opel, BMW, Honda;

kolor – przyjmuje wartości ze zbioru kolorów, np. czerwony, biało-zielony;

rok produkcji – liczba całkowita większa od 1900, np. 2019;

spalanie – liczba rzeczywista dodatnia z jednym miejscem dziesiętnym, oznacza zużycie paliwa w litrach na 100 km, np. 8.5;

numer rejestracyjny – ciąg 5 liter i 3 cyfr, np. NUMER123;

data rejestracji – data postaci dd-mm-rrrr (dd – dzień, mm – miesiąc, rrrr – rok), np. 02-05-2020.

Dane o samochodach można zapisać w postaci krotek relacji *Rejestr* o schemacie

Rejestr(*Marka, Kolor, RokProd, Spalanie, NrRej, DataRej*),

np.

{("Opel", "Biały", 1997, 6.0, "ABCDE123", 10-02-1998),
("Honda", "Czerwony", 2005, NULL, "DWWRO543", 31-07-2005),
("Opel", "Czerwony", 2019, 5.6, "ZDREW876", 02-06-2019)}

lub przedstawić w czytelniejszej postaci – w tabeli dwuwymiarowej (tabela 2.1).

Tabela 2.1

Atrybuty i krotki relacji *Rejestr*

Marka	*Kolor*	*RokProd*	*Spalanie*	*NrRej*	*DataRej*
Opel	Biały	1997	6.0	ABCDE123	10-02-1998
Honda	Czerwony	2005	NULL	DWWRO543	31-07-2005
Opel	Czerwony	2019	5.6	ZDREW876	02-06-2019

■

Podamy teraz definicje pojęć związanych ze schematami relacji.

Zbiór identyfikujący (ang. *identifying set*) – taki zbiór atrybutów schematu relacji, których kombinacje wartości jednoznacznie identyfikują każdą krotkę relacji o tym schemacie.

Klucz (ang. *key*) – minimalny zbiór identyfikujący, tzn. taki zbiór identyfikujący, którego żaden podzbiór nie jest zbiorem identyfikującym, czyli jest to jeden atrybut (lub minimalny zbiór atrybutów), którego wartość jednoznacznie identyfikuje każdą krotkę w relacji o danym schemacie.

Klucz kandydujący (ang. *candidate key*) – w schemacie relacji może istnieć kilka kluczy, które jednoznacznie identyfikują krotki relacji o tym schemacie. Klucze te nazywa się kluczami kandydującymi. Spośród kluczy kandydujących wybiera się jeden, tzw. klucz główny. W kluczu kandydującym żaden atrybut nie może przyjmować wartości pustej (NULL).

Klucz główny (ang. *primary key*) – w jednym schemacie relacji może istnieć kilka kluczy kandydujących. Jeden klucz, wybrany spośród kluczy kandydujących, nazywa się kluczem głównym.

Sztuczny klucz główny (ang. *artificial primary key*) – sztucznie utworzony klucz główny w przypadku, gdy klucze kandydujące są złożone są danymi chronionymi, gdy brak jest klucza głównego. Słowo „sztuczny" oznacza, że klucz ten nie pojawił się w sposób naturalny i trzeba go dodatkowo dodać do zbioru atrybutów schematu relacji, np. *IdWpisu*.

Klucz obcy (ang. *foreign key*) – atrybut lub kilka atrybutów w schemacie relacji, które wskazują na klucz główny (kandydujący) w innym (lub w tym samym) schemacie relacji. Odzwierciedla powiązanie między danymi w różnych relacjach (lub między krotkami tej samej relacji) i służy do zapewnienia zgodności (spójność) danych.

Klucz prosty (ang. *simple key*) – klucz jest prosty, jeśli minimalny zbiór identyfikujący jest jednoelementowy, czyli jeśli kluczem jest jeden atrybut.

Klucz złożony (ang. *complex key*) – klucz jest złożony, jeśli minimalny zbiór identyfikujący nie jest zbiorem jednoelementowym, czyli jeśli klucz składa się z kilku atrybutów.

Nadklucz (ang. *super key*) – zbiór atrybutów, który zawiera klucz (czyli zbiór atrybutów, spośród których można utworzyć klucz danego schematu relacji). W szczególności, zbiór wszystkich atrybutów schematu relacji jest nadkluczem.

Powiązanie (ang. *relationship*) – związek zachodzący pomiędzy określonymi atrybutami krotek różnych (lub tej samej) relacji.

2.2. PODSTAWOWE OPERACJE ALGEBRY RELACJI

W teoretycznym opisie modelu relacyjnego operacje na danych definiuje się w terminach algebry relacji [Codd1972, Beynon2000, Cell1988, Date1981, Date2000, Pankow1992].

Operatory algebry relacji mogą być jedno lub dwuargumentowe, przy czym argumentami są relacje (zbiory krotek) nazywane relacjami wejściowymi, a wynikiem ich działania jest nienazwana relacja zwana relacją wynikową.

Szczegółowe omówienie operatorów stosowanych w modelu relacyjnym do zapisu operacji algebry relacji wykonywanych na danych (relacjach) znajduje się w dalszej części rozdziału.

Podstawowe operacje algebry relacji to:

- ***Selekcja*** (ang. *select*)
  wybór krotek z relacji wejściowej spełniających zadany warunek dotyczący wartości atrybutów w danej relacji.
- ***Projekcja*** (ang. *project*) – rzutowanie
  wybór z relacji wejściowej określonego zbioru atrybutów.
- ***Iloczyn kartezjański*** (ang. *product*)
  operacja określona dla dwóch relacji wejściowych o różnych nazwach atrybutów. Relacja wynikowa zawiera atrybuty z jednej i z drugiej relacji. Zbiór krotek relacji wynikowej powstaje przez utworzenie iloczynu kartezjańskiego krotek obu relacji wejściowych.

- ***Złączenie wewnętrzne*** (ang. *inner join*)
  operacja polegająca na połączeniu krotek z dwóch relacji wejściowych o takich samych wartościach atrybutów łączących obie relacje (atrybutów o takich samych nazwach występujących w obu relacjach).
- ***Złączenie naturalne*** (ang. *join*)
  złączenie wewnętrzne, z tym, że atrybuty złączenia występują w relacji wynikowej tylko raz.
- ***Teta-złączenie*** (*Θ-złączenie*)
  złączenie wewnętrzne krotek spełniających zadany warunek Θ.
- ***Złączenie zewnętrzne*** (ang. *outer join*)
  złączenie dwóch relacji, w którym są zachowywane wszystkie krotki z obu relacji.
- ***Złączenie zewnętrzne lewostronne*** (ang. *left outer join*)
  złączenie, w którym są zachowane wszystkie krotki relacji z lewej strony złączenia.
- ***Złączenie zewnętrzne prawostronne*** (ang. *right outer join*)
  złączenie, w którym są zachowane wszystkie krotki relacji z prawej strony złączenia.
- ***Suma*** (ang. *union*)
  operacja określona dla dwóch relacji wejściowych o takich samych atrybutach. Wynikiem jest relacja, której zbiór krotek jest sumą teoriomnogościową zbioru krotek obu relacji wejściowych.
- ***Różnica*** (ang. *difference*)
  operacja określona dla dwóch relacji wejściowych o takich samych atrybutach. Wynikiem jest relacja, której zbiór krotek jest różnicą teoriomnogościową zbiorów krotek relacji wejściowych.
- ***Przecięcie*** (ang. *intersect*) – część wspólna
  operacja określona dla dwóch relacji wejściowych o takich samych atrybutach. Wynikiem jest relacja zawierająca krotki należące do obu relacji wejściowych.
- ***Iloraz*** (ang. *divide*)
  dwuargumentowa operacja dzielenia relacji *R1* przez relację *R2*. Relacja wynikowa zawiera krotki, których iloczyn kartezjański z dzielnikiem (*R2*) należy do dzielnej (*R1*).

Ponadto w rozdziale opisano następujące operacje wykonywane na relacjach:

- ***Przypisanie***
  operacja nadania nazwy relacji wynikowej.

- ***Nadanie aliasu***
  operacja polega na nadaniu nazwy alternatywnej relacji wejściowej.
- ***Zmiana nazwy***
  operacja umożliwia zmianę nazw atrybutów relacji i / lub nazwy relacji.

Ze względu na brak standardowej składni operatorów relacyjnych w literaturze do opisu operacji relacyjnych wykorzystuje się wysokiego poziomu subjęzyki danych, które są zbiorem operatorów wykonywanych na danych zapisanych w postaci relacji. Ponadto można definiować nowe (własne) operacje i operatory realizujące określne działania na relacjach, których wynikami są także relacje. Przykładem jest wprowadzona i stosowana w książce operacja nadania aliasu relacji. Można także zdefiniować operacje np. na atrybutach relacji.

Operacje sumy, przecięcia, iloczynu kartezjańskiego i złączenia naturalnego w algebrze relacji są łączne i przemienne.

Omówimy teraz podstawowe operatory algebry relacji, ilustrując ich użycie przykładami, w których będą wykorzystywane relacje *Pracownicy* i *Oddziały*.

Dana jest relacja *Pracownicy* (tabela 2.2) o schemacie:
Pracownicy(*IdP*, *Nazwisko*, *Imię*, *Pensja*, *IdOddz*, *MiastoZ*)

oraz relacja *Oddziały* (tabela 2.3) o schemacie:
Oddziały(*IdOddz*, *NazwaO*, *Adres*),

przy czym atrybut podkreślony oznacza klucz główny.

Tabela 2.2

Relacja *Pracownicy*

IdP	*Nazwisko*	*Imię*	*Pensja*	*IdOddz*	*MiastoZ*
0001	Wabacki	Jan	2300	2	Opole
0002	Jarecki	Jan	1500	2	Opole
0004	Kotecki	Adam	1500	4	Nysa
0005	Abacka	Ewa	1100	4	Warszawa
0006	Wabacki	Marek	2300	2	Opole

Tabela 2.3

Relacja *Oddziały*

IdOddz	*NazwaO*	*Adres*
1	Wrocław	50-370 Wrocław, Polna 1
2	Opole	48-100 Opole, Lipowa 2
3	Poznań	60-965 Poznań, Wiosenna 3
4	Warszawa	01-003 Warszawa, 3 Maja 4

Niech F oznacza formułę zawierającą jako argumenty nazwy atrybutów lub stałe, operatory porównania ($=, >, \geq, <, \leq, \neq$) oraz operatory logiczne ($\neg, \wedge, \vee$).

Operacja **selekcji** (wyboru) zapisywana przy użyciu operatora σ (SELECT) jako

$$\sigma_F(R)$$

oznacza wybór tych krotek z relacji R, które spełniają formułę F.

Jeśli relacja R ma schemat $\mathbf{R}(A_1, \ldots, A_n)$, to relacja $\sigma_F(R)$ ma również schemat $\mathbf{R}(A_1, \ldots, A_n)$. Stopień relacji wynikowej jest więc taki sam jak relacji R, natomiast $\text{card}(\sigma_F(R)) \leq \text{card}(R)$.

Za pomocą operatora selekcji można, np. zapisać operację wyszukania danych pracowników o nazwisku "Nowak" z relacji *Pracownicy*. Wynikiem będzie relacja zawierającą te krotki z relacji *Pracownicy*, których odpowiednie atrybuty spełniają zadany warunek, czyli *Nazwisko* = "Nowak". Schemat relacji wynikowej zawiera takie same atrybuty jak schemat relacji wejściowej – w tym przypadku takie same jak schemat relacji *Pracownicy*.

C.J. Date w pracy [Date2000] operację wyboru nazywa operacją restrykcji (ang. *restrict*), dla odróżnienia jej od polecenia SELECT występującego w języku SQL.

Przykład 2.3

Zdefiniujmy dla relacji *Pracownicy* (tabela 2.2) operację selekcji

$$\sigma_{MiastoZ = \text{"Opole"}}(Pracownicy),$$

którą czytamy jako

„Wybierz z relacji *Pracownicy* krotki, w których atrybut *MiastoZ* ma wartość Opole"

lub w sposób bardziej naturalny

„Podaj dane pracowników zamieszkałych w Opolu".

Wynik operacji selekcji przedstawiono w tabeli 2.4.

Zauważmy, że $\text{sch}(\sigma_{MiastoZ = \text{"Opole"}}(Pracownicy)) = \text{sch}(Pracownicy)$.

Tabela 2.4

Relacja $\sigma_{MiastoZ = \text{"Opole"}}(Pracownicy)$

IdP	*Nazwisko*	*Imię*	*Pensja*	*IdOddz*	*MiastoZ*
0001	Wabacki	Jan	2300	2	Opole
0002	Jarecki	Jan	1500	2	Opole
0006	Wabacki	Marek	2300	2	Opole

Z tabeli 2.4 wynika, że $card(\sigma_{MiastoZ = \text{"Opole"}}(Pracownicy)) = 3$.

Operacja **projekcji** (rzutowania) zapisywana jest za pomocą operatora π (PROJECT) jako $\pi_Z(R)$ i powoduje wybór tych atrybutów z relacji R, które należą do zbioru Z.

Jeśli relacja R ma schemat relacji $\mathbf{R}(A_1, \ldots, A_n)$ i zbiór atrybutów Z jest dowolnym podzbiorem zbioru atrybutów schematu **R**, czyli $Z = \{ A_{i_1}, \ldots, A_{i_j} \}$, gdzie $j \in [1, n]$, to wynikiem operacji rzutowania relacji R na zbiór atrybutów Z, co zapisujemy

$$\pi_Z(R)$$

jest relacja o atrybutach $A_{i_1}, \ldots, A_{i_j}$.

Stopień relacji wynikowej równa się liczbie atrybutów w zbiorze Z, liczność relacji wynikowej jest mniejsza lub równa liczności relacji wejściowej R.

Ponieważ relacja jest zbiorem krotek, a w zbiorach nie ma powtarzających się elementów, więc np. $\text{card}(\pi_{\{Nazwisko\}}(Pracownicy)) = 4 \leq \text{card}(Pracownicy)$.

W szczególnym przypadku zbiór Z może zawierać wszystkie atrybuty relacji R, wówczas schemat relacji wynikowej jest taki sam jak schemat relacji wejściowej R. Takie rzutowanie jest stosowane np. w celu zmiany kolejności atrybutów.

Przykład 2.4

a) Dla relacji *Pracownicy* (tabela 2.2) zdefiniujmy operację projekcji

$$\pi_{\{IdP,\ Nazwisko,\ Imię\}}(Pracownicy),$$

którą czytamy jako

„Wybierz z relacji *Pracownicy* atrybuty *IdP*, *Nazwisko*, *Imię*"

lub w sposób bardziej naturalny:

„Podaj identyfikator, nazwisko i imię wszystkich pracowników firmy".

W tabeli 2.5 przedstawiono relację wynikową otrzymaną po wykonaniu projekcji

$$\pi_{\{IdP,\ Nazwisko,\ Imię\}}(Pracownicy).$$

Tabela 2.5

Relacja wynikowa

IdP	*Nazwisko*	*Imię*
0001	Wabacki	Jan
0002	Jarecki	Jan
0004	Kotecki	Adam
0005	Abacka	Ewa
0006	Wabacki	Marek

Relacja *Pracownicy* jest stopnia 6 a relacja wynikowa jest stopnia 3. Liczność relacji wynikowej jest taka sama jak relacji *Pracownicy* i wynosi 5.

b) Z relacji *Pracownicy* (tabela 2.2) należy podać identyfikatory, nazwiska i imiona osób zamieszkałych w Opolu. W tym celu zapisujemy:

$$\pi_{\{IdP,\ Nazwisko,\ Imię\}}(\sigma_{MiastoZ = \text{"Opole"}}(Pracownicy))$$

Otrzymaną relację wynikową przedstawiono w tabeli 2.6.

Tabela 2.6

Relacja wynikowa

IdP	*Nazwisko*	*Imię*
0001	Wabacki	Jan
0002	Jarecki	Jan
0006	Wabacki	Marek

Operacja **iloczynu kartezjańskiego** relacji *R1* i *R2* zapisywana jest za pomocą operatora dwuargumentowego oznaczanego symbolem × następująco:

$$R1 \times R2$$

Nazwy relacji, które są argumentami iloczynu kartezjańskiego, muszą być różne i nazwy ich atrybutów też muszą być różne.

Wynikiem operacji iloczynu kartezjańskiego relacji *R1* i *R2* jest relacja, która zawiera wszystkie atrybuty jednej i drugiej relacji. Zbiór krotek relacji wynikowej powstaje przez utworzenie kombinacji (iloczynu kartezjańskiego) krotek relacji wejściowych *R1* i *R2*, a następnie każda para krotek z relacji wejściowych zostaje zastąpiona konkatenacją tych krotek. Jeśli krotka $k1 \in R1$, a krotka $k2 \in R2$, to do iloczynu kartezjańskiego $R1 \times R2$ należą krotki utworzone jako konkatenacja ze składowych krotek *k1* i *k2* dla wszystkich możliwych par krotek *k1* z krotkami *k2*.

Stopień relacji wynikowej jest równy sumie stopni relacji wejściowych.

Liczność relacji wynikowej jest równa iloczynowi liczności relacji wejściowych, czyli $\text{card}(R1 \times R2) = \text{card}(R1) * \text{card}(R2)$.

Iloczyn kartezjański w algebrze relacji jest operacją przemienną.

Jeśli relacje będące argumentami operatora (np. iloczynu kartezjańskiego) muszą mieć różne nazwy atrybutów, a tymczasem mają atrybuty o takich samych nazwach, to stosuje się wówczas **nazwy kwalifikowane** postaci

nazwa_relacji.nazwa_atrybutu

co pozwala na jednoznaczne identyfikowanie nazw atrybutów, np.

Pracownicy.IdOddz, *Oddziały.IdOddz*

Sposób ten nie zawsze jest skuteczny, np. jeśli trzeba wykonać operację $R \times R$. Należy wówczas relacji *R* nadać dodatkową nową nazwę, tzw. alias (pseudonim).

Zdefiniujmy operator tworzenia **aliasu** (nazwy alternatywnej) dla relacji *R* jako

R **ALIAS** *R_1*

Po wykonaniu operacji ALIAS do relacji R można się odwoływać, używając nazwy R lub alternatywnej nazwy R_1.

Zakładamy, że tam, gdzie jest wymagana unikalność nazw atrybutów oraz nazwy relacji są różne, stosowane są nazwy kwalifikowane, natomiast jeśli nazwy relacji są takie same, to do jednej z nich odwołujemy się poprzez alias.

Operator ALIAS można wykorzystać do utworzenia iloczynu kartezjańskiego $R \times R$. W tym celu dla relacji R należy utworzyć alias R_1, zapisując:

ALIAS(R) AS R_1

a następnie obliczyć iloczyn kartezjański jako

$R \times R\_1$

zakładając, że będą używane nazwy kwalifikowane atrybutów obu relacji. Te dwie operacje można zapisać w jednym wyrażeniu jako złożenie dwóch operacji:

$R \times (R$ ALIAS $R\_1)$.

Sytuacja, w której należy wykonać wiele operacji, zdarza się dosyć często i wówczas można je zapisać w jednym wyrażeniu złożonym z kilku operacji zagnieżdżonych. Taki zapis jest jednak na ogół skomplikowany i nieczytelny. Zaleca się wówczas relacjom wynikowym, otrzymywanym z poszczególnych operacji, nadawać nazwy i rozwiązanie przedstawić jako sekwencję operacji zapisanych w postaci prostych wyrażeń.

Operacja **nadania nazwy** (przypisania) dla nienazwanej relacji wynikowej jest zapisywana za pomocą operatora **przypisania** oznaczanego symbolem := następująco:

S := *wyrażenie algebry relacji*

Wynikiem operacji jest wykonanie operacji zapisanej w postaci wyrażenia po prawej stronie symbolu przypisania := i nadanie relacji wynikowej nazwy S.

Zapis $S := R1 \times R2$ oznacza zatem wykonanie operacji iloczynu kartezjańskiego dla relacji $R1$ i $R2$, a następnie nadanie nazwy S otrzymanej relacji wynikowej.

Nazwę relacji i nazwy atrybutów relacji można zmienić (przemianować) za pomocą operacji zmiany nazwy.

Operacja **zmiany nazwy** (**przemianowania**) relacji R o schemacie $\mathbf{R}(A_1, \ldots, A_n)$ i/lub zmiany nazw atrybutów relacji jest zapisywana za pomocą jednoargumentowego operatora RENAME w postaci

RENAME(R[: A_1 **AS** $NewA_1$, …, A_j **AS** $NewA_j$]) [**AS** $NewR$],

gdzie $1 \leq j \leq n$.

Nawiasy kwadratowe w definicji operatora RENAME oznaczają, że wyrażenie wewnątrz nawiasów jest opcjonalne.

Operator RENAME, w zależności od zapisu, umożliwia wykonanie operacji:

1) zmianę nazw atrybutów relacji, np. RENAME(*R*: *Symbol* AS *KlasaU*),
2) zapisanie relacji pod nową nazwą, np. RENAME(*R*) AS *R1*,
3) zmianę nazw atrybutów relacji i zapisanie relacji pod nową nazwą, np. RENAME(*R*: *Symbol* AS *KlasaU*) AS *R1*.

Operacja zmiany nazwy jest realizowana w ten sposób, że dla relacji *R* są dokonywane ewentualne zmiany nazwy wskazanych atrybutów i jeśli występuje parametr **AS** *NewR* – określający zmianę nazwy relacji – to relacji *R* jest nadawana nowa nazwa *NewR*.

Przykład 2.5

Dla działu kadr na podstawie relacji *Pracownicy* i *Oddziały* (tabele 2.2 i 2.3) należy utworzyć nową relację o nazwie *PracOddz* zawierającą dane pracowników pracujących w oddziałach (czyli z przypisanymi numerami oddziałów) wraz z nazwą oddziału zamiast jego numerem.

W tym celu wykonujemy następujące operacje:

Pracownicy ALIAS *P*

$$PracOddz := \pi_{\{IdP,\ Nazwisko,\ Imię,\ Pensja,NazwaO,MiastoZ\}}(\sigma_{P.IdOddz\ =\ Oddziały.IdOddz}(P \times Oddziały))$$

Dane relacji wynikowej *PracOddz* przedstawiono w tabeli 2.7.

Tabela 2.7

Relacja *PracOddz*

IdP	*Nazwisko*	*Imię*	*Pensja*	*NazwaO*	*MiastoZ*
0001	Wabacki	Jan	2300	Opole	Opole
0002	Jarecki	Jan	1500	Opole	Opole
0004	Kotecki	Adam	1500	Warszawa	Nysa
0005	Abacka	Ewa	1100	Warszawa	Warszawa
0006	Wabacki	Marek	2300	Opole	Opole

■

Przykład 2.6

Należy utworzyć iloczyn kartezjański relacji *Pracownicy* (tabela 2.2) i relacji *Oddziały* (tabela 2.3). W obu schematach relacji występuje atrybut o takiej samej nazwie *IdOddz*, dlatego będziemy się do niego odwoływać, stosując nazwy kwalifikowane.

Wykorzystując operator RENAME, zmieniamy w relacji *Pracownicy* nazwę atrybutu *IdOddz* na *IdO*

RENAME(*Pracownicy*: *IdOddz* AS *IdO*)

a następnie tworzymy iloczyn kartezjański *Pracownicy* × *Oddziały*, którego wynik przedstawiono w tabeli 2.8.

Stopień relacji wynikowej wynosi 9.

Relacja wynikowa zawiera 20 krotek (liczba krotek relacji *Pracownicy* pomnożona przez liczbę krotek relacji *Oddziały*). Iloczyn kartezjański *Pracownicy* × *Oddziały* zawiera wszystkie możliwe połączenia każdego pracownika z każdym oddziałem.

Relacja *Pracownicy* × *Oddziały* zawiera na przykład krotkę

(0001,"Wabacki","Jan",2300,2,"Opole",2,"Opole","48-100Opole,Lipowa2").

Tabela 2.8

Iloczyn kartezjański *Pracownicy* × *Oddziały*

IdP	*Nazwisko*	*Imię*	*Pensja*	*IdO*	*MiastoZ*	*IdOddz*	*NazwaO*	*Adres*
0001	Wabacki	Jan	2300	2	Opole	1	Wrocław	50-370 Wrocław, Polna 1
0001	Wabacki	Jan	2300	2	Opole	2	Opole	48-100 Opole, Lipowa 2
0001	Wabacki	Jan	2300	2	Opole	3	Poznań	60-965 Poznań, Wiosenna 3
0001	Wabacki	Jan	2300	2	Opole	4	Warszawa	01-003 Warszawa, 3 Maja 4
0002	Jarecki	Jan	1500	2	Opole	1	Wrocław	50-370 Wrocław, Polna 1
0002	Jarecki	Jan	1500	2	Opole	2	Opole	48-100 Opole, Lipowa 2
0002	Jarecki	Jan	1500	2	Opole	3	Poznań	60-965 Poznań, Wiosenna 3
0002	Jarecki	Jan	1500	2	Opole	4	Warszawa	01-003 Warszawa, 3 Maja 4
0004	Kotecki	Adam	1500	4	Nysa	1	Wrocław	50-370 Wrocław, Polna 1

0004	Kotecki	Adam	1500	4	Nysa	2	Opole	48-100 Opole, Lipowa 2
0004	Kotecki	Adam	1500	4	Nysa	3	Poznań	60-965 Poznań, Wiosenna 3
0004	Kotecki	Adam	1500	4	Nysa	4	Warszawa	01-003 Warszawa, 3 Maja 4
0005	Abacka	Ewa	1100	4	Warszawa	1	Wrocław	50-370 Wrocław, Polna 1
0005	Abacka	Ewa	1100	4	Warszawa	2	Opole	48-100 Opole, Lipowa 2
0005	Abacka	Ewa	1100	4	Warszawa	3	Poznań	60-965 Poznań, Wiosenna 3
0005	Abacka	Ewa	1100	4	Warszawa	4	Warszawa	01-003 Warszawa, 3 Maja 4
0006	Wabacki	Marek	2300	2	Opole	1	Wrocław	50-370 Wrocław, Polna 1
0006	Wabacki	Marek	2300	2	Opole	2	Opole	48-100 Opole, Lipowa 2
0006	Wabacki	Marek	2300	2	Opole	3	Poznań	60-965 Poznań, Wiosenna 3
0006	Wabacki	Marek	2300	2	Opole	4	Warszawa	01-003 Warszawa, 3 Maja 4

Operacja **złączenia wewnętrznego** relacji *R1* i *R2* zapisywana za pomocą operatora złączenia oznaczanego symbolem $\triangleright\triangleleft$ jako

$$R1 \triangleright\triangleleft R2$$

i polega na utworzeniu relacji wynikowej, w której krotki z relacji *R1* będą połączone (na zasadzie konkatenacji krotek) z odpowiednimi krotkami relacji *R2*, na podstawie tych samych wartości atrybutów o tych samych nazwach występujących w *R1* i w *R2*. Stopień relacji wynikowej jest równy sumie stopni relacji wejściowych *R1* i *R2*.

Złączenie relacji *R1* i *R2* polega na utworzeniu iloczynu kartezjańskiego relacji *R1* i *R2*, a następnie wybraniu tych krotek, dla których wartości atrybutów łączących występujących w relacji *R1* i w relacji *R2* są takie same.

Najczęściej zbiór łączący obie relacje jest jednoelementowy i jest to klucz główny z jednej relacji i odpowiadający mu klucz obcy z drugiej relacji.

Przykład 2.7

Należy utworzyć relację zawierającą dane o pracownikach i oddziałach, w których pracują.

W tym celu wykonujemy złączenie relacji *Pracownicy* (tabela 2.2) i *Oddziały* (tabela 2.3):

$$Pracownicy \triangleright\triangleleft Oddziały,$$

którego wynikiem jest relacja o atrybutach

{*IdP*, *Nazwisko*, *Imię*, *Pensja*, *Pracownicy.IdOddz*, *MiastoZ*, *Oddziały.IdOddz*, *NazwaO*, *Adres*}

reprezentowana jako tabela 2.9.

Tabela 2.9

Relacja *Pracownicy* ▷◁ *Oddziały*

IdP	*Nazwisko*	*Imię*	*Pensja*	*Pracownicy. IdOddz*	*MiastoZ*	*Oddziały. IdOddz*	*NazwaO*	*Adres*
0001	Wabacki	Jan	2300	2	Opole	2	Opole	48-100 Opole, Lipowa 2
0002	Jarecki	Jan	1500	2	Opole	2	Opole	48-100 Opole, Lipowa 2
0004	Kotecki	Adam	1500	4	Nysa	4	Warszawa	01-003 Warszawa, 3 Maja 4
0005	Abacka	Ewa	1100	4	Warszawa	4	Warszawa	01-003 Warszawa, 3 Maja 4
0006	Wabacki	Marek	2300	2	Opole	2	Opole	48-100 Opole, Lipowa 2

Dla atrybutu złączenia *IdOddz* występującego w obu relacjach stosowane są nazwy kwalifikowane *Pracownicy.IdOddz* i *Oddziały.IdOddz*.

W relacji *Pracownicy* ▷◁ *Oddziały* wartości atrybutu łączącego *IdOddz* występują dwukrotnie.

Stopień relacji wynikowej jest równy sumie stopni relacji *Pracownicy* i *Oddziały*, czyli 9.

■

W celu zdefiniowania operacji złączenia naturalnego relacji *R1* i *R2* wprowadzimy kilka oznaczeń. Niech

sch(*R1*) = **R1**(A_1, …, A_n, Z_1, …, Z_k),

sch(*R2*) = **R2**(Z_1, …, Z_k, B_1, …, B_m),

U1 = {A_1, …, A_n},

U2 = {B_1, …, B_m},

Z = {Z_1, …, Z_k} – zbiór atrybutów złączenia, występujących w **R1** i w **R2**,

$W = U1 \cup Z \cup U2 = \{A_1, \ldots, A_n, Z_1, \ldots, Z_k, B_1, \ldots, B_m\}$ – zbiór atrybutów występujących w **R1** i w **R2**, przy czym atrybuty powtarzające się są tylko raz.

Zakładamy, że atrybuty $Z_1, \ldots, Z_k$ (występujące w obu schematach relacji) o takich samych nazwach mają takie same dziedziny.

Złączenie naturalne relacji *R1* i *R2*, które zapisujemy jako

R1 **JOIN** *R2*

polega na wykonaniu operacji złączenia relacji *R1* i *R2* względem zbioru atrybutów *Z*, a następnie na rzutowaniu na zbiór atrybutów *W*, czyli na wykonaniu operacji rzutowania usuwającej powtarzające się atrybuty złączenia

$$\pi_W(R1 \triangleright\triangleleft R2).$$

Schemat relacji

R1 **JOIN** *R2*

zawiera atrybuty

$$A_1, \ldots, A_n, Z_1, \ldots, Z_k, B_1, \ldots, B_m.$$

Złączenie naturalne to złączenie wewnętrzne, ale bez powtarzających się atrybutów złączenia (występują tylko raz).

Tak zdefiniowana operacja złączenia naturalnego jest łączna i przemienna.

Jeśli $Z = \emptyset$, to *R1* **JOIN** *R2* = *R1* × *R2*.

Jeśli sch(*R1*) = sch(*R2*), to *R1* **JOIN** *R2* = *R1* ∩ *R2*.

Jeśli $F = (R1.Z_1 = R2.Z_1 \wedge \ldots \wedge R1.Z_k = R2.Z_k)$, to operację *R1* **JOIN** *R2* można zapisać jako

$$\pi_W(\sigma_F(R1 \times R2)).$$

W przykładzie 2.8 zilustrowano operację złączenia naturalnego.

Przykład 2.8

Wynikiem operacji złączenia naturalnego relacji *Pracownicy* (tabela 2.2) i *Oddziały* (tabela 2.3)

Pracownicy **JOIN** *Oddziały*

względem atrybutu *Pracownicy.IdOddz* i odpowiadającego mu atrybutu (o tej samej dziedzinie) *Oddziały.IdOddz* jest relacja

$$\pi_{U\text{-}Z}(Pracownicy \triangleright\triangleleft Oddziały)$$

o danych zawartych w tabeli 2.10 (zbiór *U* jest zbiorem wszystkich atrybutów relacji *Pracownicy* i *Oddziały*, zbiór $Z = \{Oddziały.IdOddz\}$).

Tabela 2.10

Relacja *Pracownicy* **JOIN** *Oddziały*

IdP	*Nazwisko*	*Imię*	*Pensja*	*IdOddz*	*MiastoZ*	*NazwaO*	*Adres*
0001	Wabacki	Jan	2300	2	Opole	Opole	48-100 Opole, Lipowa 2
0002	Jarecki	Jan	1500	2	Opole	Opole	48-100 Opole, Lipowa 2
0004	Kotecki	Adam	1500	4	Nysa	Warszawa	01-003 Warszawa, 3 Maja 4
0005	Abacka	Ewa	1100	4	Warszawa	Warszawa	01-003 Warszawa, 3 Maja 4
0006	Wabacki	Marek	2300	2	Opole	Opole	48-100 Opole, Lipowa 2

Teta-złączenie (Θ-złączenie) relacji *R1* i *R2*, które zapisujemy jako

$$R1 \triangleright\triangleleft_{\Theta} R2,$$

polega na utworzeniu relacji, w której będą krotki z relacji *R1* połączone z krotkami relacji *R2* spełniające warunek logiczny Θ dotyczący atrybutów z relacji *R1* i *R2* o tej samej dziedzinie w obu relacjach (warunek Θ musi mieć sens).

Jeśli warunek Θ jest równością, czyli ma postać *wyrażenie1* = *wyrażenie2*, to operację nazywa się równozłączeniem (ang. *equijoin*).

Operację teta-złączenia wykonuje się, tworząc iloczyn kartezjański relacji *R1* i *R2*, a następnie wybierając te krotki, dla których jest spełniony warunek Θ dotyczący atrybutów o zgodnych dziedzinach, występujących w relacji *R1* i w relacji *R2*, np.

$$R1.A1 < R2.A2, \text{ gdzie } \mathrm{dom}(A1) = \mathrm{dom}(A2).$$

Podobnie jak w przypadku iloczynu kartezjańskiego tak i w przypadku teta-złączenia obie relacje wyjściowe *R1* i *R2* muszą mieć atrybuty o różnych nazwach.

Przykład 2.9

Podać dane tylko tych pracowników, dla których istnieje oddział o numerze większym od numeru oddziału, w którym aktualnie pracuje dany pracownik.

RENAME(*Pracownicy*: *IdOddz* AS *IdO*)

Wynikiem teta-złączenia relacji *Pracownicy* (tabela 2.2) i *Oddziały* (tabela 2.3):

$$DoAwansu := Pracownicy \triangleright\triangleleft_{IdO < IdOddz} Oddziały$$

jest relacja przedstawiona w tabeli 2.11.

Tabela 2.11

Relacja *DoAwansu*

IdP	*Nazwisko*	*Imię*	*Pensja*	*IdO*	*MiastoZ*	*IdOddz*	*NazwaO*	*Adres*
0001	Wabacki	Jan	2300	2	Opole	3	Poznań	60-965 Poznań, Wiosenna 3
0001	Wabacki	Jan	2300	2	Opole	4	Warszawa	01-003 Warszawa, 3 Maja 4
0002	Jarecki	Jan	1500	2	Opole	3	Poznań	60-965 Poznań, Wiosenna 3
0002	Jarecki	Jan	1500	2	Opole	4	Warszawa	01-003 Warszawa, 3 Maja 4
0006	Wabacki	Marek	2300	2	Opole	3	Poznań	60-965 Poznań, Wiosenna 3
0006	Wabacki	Marek	2300	2	Opole	4	Warszawa	01-003 Warszawa, 3 Maja 4

Relacja wynikowa *DoAwansu* zawiera dane pracowników, dla których istnieje oddział o numerze większym od numeru oddziału, w którym aktualnie pracują. Jeśli przejście do oddziału o numerze większym oznacza awans dla pracownika, to relacja zawiera dane pracowników, którzy mogą być awansowani, oraz dane oddziałów, do których mogą być awansowani.

Aby uzyskać dane o pracownikach i oddziałach, w których pracują, należy wykonać równozłączenie relacji *Pracownicy* i *Oddziały*

$$Pracownicy \triangleright\triangleleft_{IdO = IdOddz} Oddziały$$

Dla oznaczenia dwuargumentowego **złączenia zewnętrznego** przyjmiemy operator $\lhd\rhd$, dla złączenia **lewostronnego** przyjmiemy operator $\lhd$, a dla złączenia **prawostronnego** operator $\rhd$. Złączenie zewnętrzne lewostronne jest to złączenie, które zachowuje wszystkie krotki z relacji po lewej stronie operatora $\lhd$, złączenie zewnętrzne prawostronne zachowuje wszystkie krotki z relacji po prawej stronie operatora $\rhd$. Złączenie zewnętrzne jest to złączenie zachowujące wszystkie krotki z obu relacji biorących udział w złączeniu.

Brakujące wartości składowych krotek w relacjach wynikowych w złączeniu zewnętrznym, lewostronnym i prawostronnym są uzupełniane wartością pustą NULL (więcej informacji na temat wartości NULL zamieszczono w rozdz. 3.7.1).

Przykład 2.10

Dane są dwie relacje: *Kadry* (tabela 2.12) i *Oddziały* (tabela 2.13).

Tabela 2.12

Relacja *Kadry*

IdP	*Nazwisko*	*Imię*	*Pensja*	*IdOddz*	*MiastoZ*
0001	Wabacki	Jan	2300	2	Opole
0002	Jarecki	Jan	1500	NULL	Opole
0004	Kotecki	Adam	1500	NULL	Nysa
0005	Abacka	Ewa	1100	4	Warszawa
0006	Wabacki	Marek	2300	2	Opole

Tabela 2.13

Relacja *Oddziały*

IdOddz	*NazwaO*	*Adres*
1	Wrocław	50-370 Wrocław, Polna 1
2	Opole	48-100 Opole, Lipowa 2
3	Poznań	60-965 Poznań, Wiosenna 3
4	Warszawa	01-003 Warszawa, 3 Maja 4

Wynikiem złączenia lewostronnego

Kadry ◁ *Oddziały*

jest relacja reprezentowana przez tabelę 2.14.

Tabela 2.14

Relacja *Kadry* ◁ *Oddziały*

IdP	*Nazwisko*	*Imię*	*Pensja*	*Kadry. IdOddz*	*MiastoZ*	*Oddziały. IdOddz*	*NazwaO*	*Adres*
0001	Wabacki	Jan	2300	2	Opole	2	Opole	48-100 Opole, Lipowa 2
0002	Jarecki	Jan	1500	NULL	Opole	NULL	NULL	NULL
0004	Kotecki	Adam	1500	NULL	Nysa	NULL	NULL	NULL
0005	Abacka	Ewa	1100	4	Warszawa	4	Warszawa	01-003 Warszawa, 3 Maja 4
0006	Wabacki	Marek	2300	2	Opole	2	Opole	48-100 Opole, Lipowa 2

W złączeniu lewostronnym *Kadry* ◁ *Oddziały* występują dane wszystkich pracowników i oddziałów, w których oni pracują. Jeśli pracownik nie jest przypisany do żadnego oddziału, to dane o oddziale mają wartości NULL.

Aby uzyskać dane o wszystkich oddziałach i zatrudnionych w nich pracownikach, należy wykonać złączenie prawostronne

Kadry ▷ *Oddziały*,

którego wynikiem jest relacja reprezentowana przez tabelę 2.15. Relacja *Kadry* ▷ *Oddziały* zawiera dane o wszystkich oddziałach, nawet jeśli do oddziału nie jest przypisany żaden pracownik.

Wynikiem złączenia zewnętrznego (obustronnego)

Kadry ◁ ▷ *Oddziały*

jest relacja reprezentowana przez tabelę 2.16, w której są dane o wszystkich pracownikach i wszystkich oddziałach.

Stopień relacji *Kadry* ◁ ▷ *Oddziały*, będącej wynikiem złączenia zewnętrznego jest równy sumie stopni relacji wejściowych *Kadry* i *Oddziały*, czyli wynosi 9.

Liczność relacji *Kadry* ◁ ▷ *Oddziały* jest równa sumie liczności relacji wejściowych *Kadry* i *Oddziały* minus liczność relacji *Kadry* **JOIN** *Oddziały*.

Tabela 2.15

Relacja *Kadry* ▷ *Oddziały*

IdP	*Nazwisko*	*Imię*	*Pensja*	*Kadry. IdOddz*	*MiastoZ*	*Oddziały. IdOddz*	*NazwaO*	*Adres*
NULL	NULL	NULL	NULL	NULL	NULL	1	Wrocław	50-370 Wrocław, Polna 1
0001	Wabacki	Jan	2300	2	Opole	2	Opole	48-100 Opole, Lipowa 2
0006	Wabacki	Marek	2300	2	Opole	2	Opole	48-100 Opole, Lipowa 2
NULL	NULL	NULL	NULL	NULL	NULL	3	Poznań	60-965 Poznań, Wiosenna 3
0005	Abacka	Ewa	1100	4	Warszawa	4	Warszawa	01-003 Warszawa, 3 Maja 4

Tabela 2.16

Relacja *Kadry* ◁ ▷ *Oddziały*

IdP	*Nazwisko*	*Imię*	*Pensja*	*Kadry. IdOddz*	*MiastoZ*	*Oddziały. IdOddz*	*NazwaO*	*Adres*
0001	Wabacki	Jan	2300	2	Opole	2	Opole	48-100 Opole, Lipowa 2
0002	Jarecki	Jan	1500	NULL	Opole	NULL	NULL	NULL
0004	Kotecki	Adam	1500	NULL	Nysa	NULL	NULL	NULL
0005	Abacka	Ewa	1100	4	Warszawa	4	Warszawa	01-003 Warszawa, 3 Maja 4
0006	Wabacki	Marek	2300	2	Opole	2	Opole	48-100 Opole, Lipowa 2
NULL	NULL	NULL	NULL	NULL	NULL	1	Wrocław	50-370 Wrocław, Polna 1
NULL	NULL	NULL	NULL	NULL	NULL	3	Poznań	60-965 Poznań, Wiosenna 3

Kolejne operatory algebry relacji – suma, różnica i przecięcie dwóch relacji – muszą mieć ten sam stopień i atrybuty o tych samych nazwach i dziedzinach.

Suma relacji *R1* i *R2*, o takich samych atrybutach, zawiera krotki należące do relacji *R1* lub do relacji *R2*. Operator sumy oznaczany jest symbolem $\cup$. Relacja wynikowa ma atrybuty takie same jak relacje wejściowe, liczba krotek jest mniejsza od sumy liczby krotek w relacjach wejściowych (w wyniku nie występują powtarzające się krotki) lub jej równa, czyli

$$\text{card}(R1 \cup R2) = \text{card}(R1) + \text{card}(R2) - \text{card}(R1 \cap R2),$$

gdzie $R1 \cap R2$ jest przecięciem relacji *R1* i *R2*.

Przykład 2.11

Rozważmy relacje: *Nauczyciele* (tabela 2.17) i *Studenci* (tabela 2.18) o schemacie relacji **Osoby**(*IdO*, *Nazwisko*, *Imię*, *MiastoZ*).

Relacja *Nauczyciele* $\cup$ *Studenci* zawiera dane nauczycieli i studentów, przedstawione w tabeli 2.19.

W relacjach wejściowych *Nauczyciele* i *Studenci* występuje identyczna krotka z danymi Jana Wabackiego, ale w relacji wynikowej *Nauczyciele* $\cup$ *Studenci* krotka ta występuje tylko jeden raz. Liczność relacji wynikowej jest więc mniejsza od sumy liczności relacji wejściowych. Schemat relacji wynikowej jest taki sam jak relacji wejściowych, czyli relacja *Nauczyciele* $\cup$ *Studenci* ma schemat **OSOBY**.

Tabela 2.17

Relacja *Nauczyciele*

IdO	*Nazwisko*	*Imię*	*MiastoZ*
0001	Wabacki	Jan	Opole
0002	Jarecki	Jan	Opole
0004	Kotecki	Adam	Nysa
0005	Abacka	Ewa	Warszawa
0006	Wabacki	Marek	Opole

Tabela 2.18

Relacja *Studenci*

IdO	*Nazwisko*	*Imię*	*MiastoZ*
0001	Wabacki	Jan	Opole
0002	Abacki	Jan	Warszawa
0003	Nowicki	Adam	Nysa
0004	Abacka	Ewa	Warszawa

Tabela 2.19

Relacja *Nauczyciele* ∪ *Studenci*

IdO	*Nazwisko*	*Imię*	*MiastoZ*
0001	Wabacki	Jan	Opole
0002	Jarecki	Jan	Opole
0004	Kotecki	Adam	Nysa
0005	Abacka	Ewa	Warszawa
0006	Wabacki	Marek	Opole
0002	Abacki	Jan	Warszawa
0003	Nowicki	Adam	Nysa
0004	Abacka	Ewa	Warszawa

Różnicę relacji (podobnie jak sumę) tworzy się dla dwóch relacji o takich samych atrybutach. Operatorem różnicy jest symbol –. Do różnicy dwóch relacji *R1* – *R2* należą krotki, które występują w relacji *R1*, ale nie występują w relacji *R2*. Schemat relacji wynikowej jest taki sam jak relacji wejściowych.

Dla różnicy relacji zachodzi zależność card(*R1* – *R2*) = card(*R1*) – card(*R1* ∩ *R2*).

Przykład 2.12

Podać dane nauczycieli, którzy nie są studentami.

W tym celu należy utworzyć różnicę relacji *Nauczyciele* (tabela 2.17) i *Studenci* (tabela 2.18). Relacja wynikowa

$$NieStudenci := Nauczyciele - Studenci$$

zawiera krotki przedstawione w tabeli 2.20.

Tabela 2.20

Relacja *NieStudenci*

IdO	*Nazwisko*	*Imię*	*MiastoZ*
0002	Jarecki	Jan	Opole
0004	Kotecki	Adam	Nysa
0005	Abacka	Ewa	Warszawa
0006	Wabacki	Marek	Opole

Przecięcie (część wspólną) dwóch relacji *R1* i *R2* (podobnie jak sumę i różnicę) tworzy się dla relacji o takich samych atrybutach. Dwuargumentowy operator przecięcia oznaczany jest symbolem $\cap$. Do przecięcia relacji należą krotki, które występują w obu relacjach wejściowych *R1* i *R2*.

Schemat relacji wynikowej jest taki sam jak relacji wejściowych.

Dla przecięcia relacji *R1* i *R2* zachodzi zależność

$$\text{card}(R1 \cap R2) \leq \min\{\text{card}(R1), \text{card}(R2)\}.$$

Przykład 2.13

Dla relacji *Nauczyciele* (tabela 2.17) i *Studenci* (tabela 2.18), relacja będąca przecięciem *Nauczyciele* $\cap$ *Studenci* zawiera krotki, które należą do obydwu relacji, czyli tylko dane Jana Wabackiego (tabela 2.21).

Tabela 2.21

Relacja *Nauczyciele* ∩ *Studenci*

IdO	*Nazwisko*	*Imię*	*MiastoZ*
0001	Wabacki	Jan	Opole

W języku naturalnym można powyższą operację sformułować następująco: „Podać dane nauczycieli, którzy są studentami".

Iloraz relacji jest operatorem dwuargumentowym i oznaczamy go symbolem ÷, a operację ilorazu zapisujemy jako

$$R1 \div R2.$$

Relacja wynikowa otrzymana po wykonaniu **operacji ilorazu** $R1 \div R2$ zawiera krotki, których iloczyn kartezjański z relacją $R2$ (dzielnikiem) zawarty jest w relacji $R1$ (dzielnej), czyli jeśli $W := R1 \div R2$, to $W \times R2 \subseteq R1$.

Operację ilorazu można także zdefiniować następująco:

Niech Z1 będzie zbiorem atrybutów relacji $R1$, Z2 zbiorem atrybutów relacji $R2$ oraz Z2 ⊆ Z1. Relacja wynikowa ilorazu $R1 \div R2$ jest zbiorem krotek

$$k1 \in \pi_{Z1-Z2}(R1)$$

takich, że dla każdej krotki $k2 \in R2$, krotka $(k1, k2) \in R1$.

Z definicji ilorazu wynika, że relacja wynikowa jest zbiorem takich krotek $k1$, które po połączeniu z każdą krotką $k2$ z relacji $R2$ należą do $R1$.

W szczególnym przypadku, jeżeli

$$\text{sch}(R1) = \mathbf{R1}(A, B)$$
$$\text{sch}(R2) = \mathbf{R2}(B)$$

to iloraz $R1 \div R2$ jest relacją unarną (o jednym atrybucie A) zawierającą takie wartości $R1.A$, że dla każdej wartości $R2.B$ krotka $(R1.A, R2.B) \in R1$.

Operację dzielenia wykorzystuje się do wykonania poleceń typu:

Podaj numery studentów, którzy zaliczyli **wszystkie** przedmioty.

Podaj nazwy kursów zaliczonych przez **wszystkich** studentów.

Podaj producentów, którzy produkują **wszystkie** towary.

Przykład 2.14

a) Dane są relacje *R1*(*Nauczyciel*, *Auto*) i *R2*(*Auto*). Obliczyć *R1* ÷ *R2*.

R1

Nauczyciel	*Auto*
Abacki	BMW
Abacki	Audi
Nowak	Ford
Lis	Audi
Nowak	Audi
Lis	BMW
Abacka	BMW

R2

Auto
BMW
Audi

W:= *R1* ÷ *R2*

Nauczyciel
Abacki
Lis

Zauważmy, że $W \times R2 \subseteq R1$.

W relacji wynikowej są nazwiska nauczycieli z relacji *R1*, którzy posiadają wszystkie auta z relacji *R2*. Jest dwóch takich nauczycieli: Abacki i Lis.

b) Dane są relacje *Nauczyciele* (tabela 2.22) i *Miasta* (tabela 2.23).
Należy wyszukać nazwiska nauczycieli, występujące w każdym mieście.

Tabela 2.22

Relacja *Nauczyciele*

IdN	*Nazwisko*	*Imię*	*IdMiasta*
0001	Wabacki	Jan	1
0002	Jarecki	Jan	1
0004	Kotecki	Adam	2
0005	Abacka	Ewa	3
0006	Wabacki	Marek	1
0007	Wabacki	Kornel	2
0008	Wabacki	Adam	3

Tabela 2.23

Relacja *Miasta*

IdMiasta	*Nazwa*
1	Opole
2	Nysa
3	Warszawa

W tym celu wykonajmy operację rzutowania relacji *Nauczyciele* na atrybuty {*Nazwisko*, *IdMiasta*} (tabela 2.24)

$$Nauczyciele\_rzut := \pi_{\{Nazwisko,\ IdMiasta\}}(Nauczyciele)$$

Tabela 2.24

Relacja *Nauczyciele_rzut*

Nazwisko	*IdMiasta*
Wabacki	1
Jarecki	1
Kotecki	2
Abacka	3
Wabacki	2
Wabacki	3

i tabeli *Miasta* na atrybut {*IdMiasta*}(tabela 2.25)

$$Miasta\_rzut := \pi_{\{IdMiasta\}}(Miasta)$$

Tabela 2.25

Relacja *Miasta_rzut*

IdMiasta
1
2
3

Relację

$$Wynik\_ilorazu := Nauczyciele\_rzut \div Miasta\_rzut$$

przedstawiono w tabeli 2.26 i wynika z niej, że w każdym mieście z relacji *Miasta* jest nauczyciel o nazwisku "Wabacki".

Tabela 2.26

Relacja *Wynik_ilorazu*

Nazwisko
Wabacki

Na zakończenie tego podrozdziału podkreślmy ważną własność operatorów algebry relacji: działają one na relacjach – zbiorach krotek, i wynikiem są relacje, czyli zbiory krotek.

Czytelnika zachęcamy do zdefiniowania własnych operatorów dla operacji wykonywanych na relacjach (np. usuwania danych, modyfikacji danych, dopisania nowego atrybutu, wyliczenia sumy dla atrybutu liczbowego), a także do zdefiniowania ich argumentów i wyników, określenia schematów relacji wejściowych i wynikowych, sposobu działania oraz do podania przykładów ich użycia.

2.3. NORMALIZACJA SCHEMATÓW RELACJI

Normalizacja schematu relacji jest sformalizowaną procedurą, w wyniku której eliminuje się redundancję (powtarzanie się) danych, ich niespójności i anomalie.

Normalizacja opiera się na idei funkcyjnej zależności atrybutów i polega na dekompozycji początkowego schematu relacji na kilka innych schematów relacji. Postacie normalne są kolejno numerowane. Im wyższy numer, tym projekt bazy danych powinien być coraz lepszy w tym sensie, że mniej podatny na anomalie, czyli nieprawidłowości związane z aktualizowaniem, usuwaniem i dopisywaniem danych.

Głównym celem projektowania relacyjnej bazy danych jest przedstawienie jej w postaci odpowiedniego zbioru schematów relacji.

Podkreślmy, że **redundancja** danych jest wówczas, gdy ta sama dana jest przechowywana w bazie więcej niż raz, np. te same dane nauczyciela są w dwóch relacjach *Nauczyciele* i *Kierownicy*, lub do schematu relacji **Uczniowie** w bazie dla liceum trzeba wpisywać dane szkoły podstawowej dla każdego ucznia, a powinien być osobny schemat relacji do przechowywania danych szkół. Inaczej mówiąc, aby nie było redundancji, to dla każdej danej (np. nazwisko danego nauczyciela) w całej bazie powinien być tylko jeden atrybut do jej zapisywania i ta dana jest zapisywana (przechowywana) tylko w jednej krotce. Do aktualizacji każdej danej wystarcza aktualizacja jednego zapisu w bazie. Jeżeli nazwisko "Kowalski" danego nauczyciela zapamiętane jest w bazie dwa razy, to jest to redundancja, ale jeżeli nazwisko "Kowalski" zapisane w bazie dwa razy odnosi się do dwóch różnych osób, to nie ma redundancji. Redundancją nie jest także nadawanie tych samych nazw atrybutom w różnych schematach relacji, np. atrybut *Nazwisko* może być w schemacie relacji **Nauczyciele** i **Uczniowie**.

Redundancja jest jednym z przypadków nadmiarowości danych. Nadmiarowość danych może wynikać także ze złego projektu schematu relacji zawierającego nadmiarowe (zbędne) atrybuty, często umieszczane bez uzasadnienia, na wszelki wypadek. W procesie normalizacji tego typu nadmiarowość nie jest usuwana.

W wyniku przeprowadzenia procesu **normalizacji** z początkowych schematów relacji tworzone są nowe schematy, wcześniej niedostrzegane lub pominięte. Na ogół normalizacja powoduje powstanie większej liczby schematów relacji w logicznym modelu bazy danych (rozdz. 4). Dobrze zaprojektowana baza danych powinna zawierać schematy relacji przynajmniej w trzeciej postaci normalnej, a ich liczba powinna być jak najmniejsza. Opis zagadnień związanych z normalizacją można znaleźć w wielu pracach, np. [Beynon1999, Beynon2000, Date2000, GaUlWi2006, Harris1994, Pankow1992, Riordan2000, Roszko1998].

Omówimy teraz pojęcia związane z normalizacją schematów relacji. Podstawowym pojęciem jest zależność funkcyjna atrybutów.

Niech A, B będą atrybutami w schemacie relacji **R**.

Atrybut B jest ***funkcyjnie zależny*** od atrybutu A, co zapisujemy

$$A \rightarrow B$$

jeśli każdej wartości a atrybutu A odpowiada co najwyżej jedna wartość b atrybutu B.

Stwierdzenie, że atrybut *B* jest funkcyjnie zależny od atrybutu *A*, jest równoważne stwierdzeniu, że atrybut *A* jednoznacznie wyznacza (identyfikuje) atrybut *B*.

Przykład 2.15

W schemacie relacji

Osoby(*PESEL*, *Nazwisko*)

atrybut *PESEL* jednoznacznie wyznacza nazwisko osoby (atrybut *Nazwisko* jest funkcyjnie zależny od atrybutu *PESEL*), czyli

$$PESEL \rightarrow Nazwisko$$

przy czym dane nazwisko może posiadać więcej niż jedna osoba (a więc *PESEL* nie jest funkcyjnie zależny od nazwiska).

■

Pojęcie funkcyjnej zależności stosuje się odpowiednio do zbiorów atrybutów.

Niech Z1 i Z2 będą dowolnymi podzbiorami zbioru atrybutów w schemacie relacji R.

Zbiór atrybutów Z2 jest funkcyjnie zależny od zbioru atrybutów Z1, co zapisujemy jako

$$Z1 \rightarrow Z2$$

jeśli każdej krotce wartości atrybutów zbioru Z1 odpowiada nie więcej niż jedna krotka wartości atrybutów zbioru Z2.

Zbiór Z1 nazywa się wyznacznikiem zależności funkcyjnej.

Czasami zamiast określenia *zależności funkcyjne* będziemy pisać po prostu *zależności*.

Zależność funkcyjna $\{A_1, A_2, \ldots, A_n\} \rightarrow \{B_1, B_2, \ldots, B_m\}$ jest **trywialna** wtedy i tylko wtedy, gdy prawa strona zależności jest podzbiorem lewej strony, czyli

$$\{B_1, B_2, \ldots, B_m\} \subseteq \{A_1, A_2, \ldots, A_n\},$$

np. $\{\underline{Student}, \underline{Kurs}\} \rightarrow Student$.

Jeśli zbiór atrybutów w zależności funkcyjnej jest jednoelementowy, to w zapisie zależności będziemy pomijali nawiasy klamrowe oznaczające zbiór.

Przykład 2.16

Dany jest schemat relacji

Egzaminy(*NumerIndeksu, Kurs, Ocena, Data*)

Zakładamy, że każdy student może mieć tylko jedną ocenę z danego kursu z daną datą, ale z danego kursu może mieć wiele ocen z różnymi datami, to znaczy, że zbiór {*NumerIndeksu, Kurs, Data*} jednoznacznie wyznacza atrybut *Ocena*, co zapisujemy

$$\{NumerIndeksu, Kurs, Data\} \rightarrow Ocena$$

czyli atrybut *Ocena* jest funkcyjnie zależny od zbioru atrybutów {*NumerIndeksu, Kurs, Data*}.

Ponadto zbiór {*NumerIndeksu, Kurs*} nie wyznacza jednoznacznie atrybutu *Ocena* (student mógł najpierw otrzymać ocenę niedostateczną), czyli

$$\{NumerIndeksu, Kurs\} \not\rightarrow Ocena$$

Również zbiór {*Ocena, Data*} nie wyznacza jednoznacznie zbioru {*NumerIndeksu, Kurs*}, ponieważ w danym dniu wielu studentów mogło otrzymać tę samą ocenę, czyli

$$\{Ocena, Data\} \not\rightarrow NumerIndeksu$$
$$\{Ocena, Data\} \not\rightarrow Kurs$$

co można zapisać

$$\{Ocena, Data\} \not\rightarrow \{NumerIndeksu, Kurs\}$$

Warto podkreślić różnicę pomiędzy pojęciami funkcja i zależność funkcyjna. Funkcja przyporządkowuje danemu elementowi zawszę tę samą wartość, np. f(1) = 7 dla f(x) = 2 * x + 5, natomiast z zależności funkcyjnej

$$PESEL \rightarrow Nazwisko$$

wynika, że PESEL wyznacza co najwyżej jedną wartość atrybutu *Nazwisko*, ale wartość ta może się zmieniać w czasie, np. po wykonaniu edycji danych.

Zdefiniujmy teraz kolejne ważne pojęcie, jakim jest pełna funkcyjna zależność zbiorów atrybutów, oznaczana symbolem $\Rightarrow$.

Jeżeli X, Y są zbiorami atrybutów w schemacie relacji **R**, to zależność

$$X \Rightarrow Y$$

jest **pełną zależnością funkcyjną**, jeżeli

$$X \rightarrow Y \text{ i nie istnieje } Z \subset X \text{ taki, że } Z \rightarrow Y$$

Zbiór X nazywa się wyznacznikiem pełnej zależności funkcyjnej.

Podaną definicję można wyrazić następująco:

Zbiór atrybutów Y w schemacie relacji **R** jest w ***pełni funkcyjnie zależny*** od zbioru atrybutów X tego schematu relacji, co zapisujemy

$$X \Rightarrow Y,$$

jeżeli zbiór Y jest funkcyjnie zależny od X, ale nie jest funkcyjnie zależny od żadnego właściwego podzbioru zbioru X.

Przykład 2.17

a) Zależność

$$\{NazwaFirmy, NazwaTowaru\} \rightarrow Cena$$

jest pełną zależnością funkcyjną, jeżeli *Cena* nie zależy od żadnego podzbioru atrybutów zbioru {*NazwaFirmy*, *NazwaTtowaru*}, czyli

$$NazwaFirmy \not\rightarrow Cena$$
$$NazwaTowaru \not\rightarrow Cena$$

b) Dany jest schemat relacji

Egzaminy(*NumerIndeksu*, *Kurs*, *Ocena*, *Data*)

Zakładamy, że student może zdawać kilka razy egzamin z danego kursu, ale w danym dniu tylko jeden raz, czyli zachodzi zależność

$$\{NumerIndeksu, Kurs, Data\} \Rightarrow Ocena$$

Zatem atrybut *Ocena* jest w pełni funkcyjnie zależny od zbioru atrybutów {*NumerIndeksu*, *Kurs*, *Data*}.

c) Dany jest schemat relacji

Oceny(*Student*, *Przedmiot*, *DataO*, *Ocena*),

do opisu danych o ocenach końcowych studentów z różnych przedmiotów, gdzie *DataO* jest datą uzyskania oceny końcowej z danego przedmiotu przez danego studenta.

W schemacie relacji **Oceny** zachodzą zależności:

$$\{Student, Przedmiot\} \Rightarrow DataO$$
$$\{Student, Przedmiot\} \Rightarrow Ocena$$

czyli atrybut *DataO*, podobnie jak atrybut *Ocena*, jest w pełni funkcyjnie zależny od zbioru atrybutów (klucza złożonego) {*Student*, *Przedmiot*}, ponieważ jest funkcyjnie zależny od zbioru atrybutów {*Student*, *Przedmiot*}, a nie jest funkcyjnie zależny ani od atrybutu *Student* ani od atrybutu *Przedmiot*.

■

W rozdziale 2.1 zdefiniowaliśmy klucz w schemacie relacji jako minimalny zbiór identyfikujący (którego żaden podzbiór nie jest zbiorem identyfikującym), czyli jest to atrybut (lub zbiór atrybutów), którego wartość jednoznacznie identyfikuje każdą krotkę w relacji o danym schemacie.

Jeśli klucz główny w schemacie relacji jest kluczem prostym, to wszystkie zależności funkcyjne względem klucza są zależnościami pełnymi.

Zanim omówimy postaci normalne schematów relacji, zauważmy, że dane mogą być przedstawione w postaci tabeli nieznormalizowanej, czyli dane w polach tabeli mogą być zbiorami wartości, a nie wartościami elementarnymi (atomowymi). Tabela, w której wszystkie wartości są elementarne (atomowe) i nie ma powtarzających się wierszy, jest reprezentacją relacji o schemacie będącym w pierwszej postaci normalnej.

Podamy teraz definicje postaci normalnych schematów relacji.

Pierwsza postać normalna **(1PN)** (ang. *first normal form*) – schemat relacji jest w pierwszej postaci normalnej wtedy i tylko wtedy, gdy każda wartość z dziedzin atrybutów jest atomowa (elementarna).

W każdej relacji R o schemacie **R** będącym w 1PN każda składowa każdej krotki jest wartością atomową, czyli jest jedną wartością z dziedziny atrybutu, a nie zbiorem wartości.

Nazwy atrybutów w schemacie relacji powinny być w liczbie pojedynczej – nazwa w liczbie mnogiej (np. *Lokatorzy*) sugeruje, że dopuszcza się wiele wartości dla danego atrybutu i schemat nie jest w 1PN.

Przykład 2.18

a) Schemat relacji **Prac1**(*NrPrac*, *ImionaDzieci*) nie jest w 1PN (tabela 2.27).

Tabela 2.27

Relacja *Prac1*

NrPrac	*ImionaDzieci*
1	Jan, Anna, Ewa
2	Marek, Adam
3	Julia

Schemat relacji **Prac2**(*NrPrac*, *ImięDziecka*) jest w 1PN (tabela 2.28)

Tabela 2.28

Relacja *Prac2*

NrPrac	*ImięDziecka*
1	Jan
1	Anna
1	Ewa
2	Marek
2	Adam
3	Julia

b) Dane są schematy relacji w pierwszej postaci normalnej:
Klienci(*NrKlienta*, *Nazwisko*, *Imię*, *Miasto*, *UlicaNr*)
Kasety(*NrKasety*, *Opis*)
Filmy(*KodFilmu*, *Tytuł*, *CzasTrwania*)
Gatunki(*KodGatunku*, *Rodzaj*)

Dowolnie wybrana składowa krotki z relacji o powyższych schematach ma zawsze dokładnie jedną wartość atomową z określonej dziedziny atrybutu.

Relację *Klienci* o schemacie **Klienci** przedstawiono w tabeli 2.29.

Tabela 2.29

Relacja *Klienci* w pierwszej postaci normalnej

NrKlienta	*Nazwisko*	*Imię*	*Miasto*	*UlicaNr*
0001	Batycki	Jerzy	Opole	Polna 34
0002	Batycki	Marek	Opole	Polna 34
0003	Widecka	Barbara	Sopot	Rynek 1/1
0004	Artecka	Barbara	Opole	Rynek 2

c) Rozważmy tabelę *Osoby* (tabela 2.30) zawierającą kolumny:
NrOsoby, Nazwisko, Imię, ImionaDzieci, DatyUrDzieci

Tabela 2.30 nie jest reprezentacją żadnej relacji, jest nieznormalizowana, czyli nie jest w pierwszej postaci normalnej, ponieważ np. w kolumnie *ImionaDzieci* dla Padurskiej Marii jest zbiór trzech wartości {Adam, Ewa, Hanna}, a nie pojedyncza wartość.

Tabela 2.30

Nieznormalizowana tabela *Osoby*

NrOsoby	*Nazwisko*	*Imię*	*ImionaDzieci*	*DatyUrDzieci*
1	Adurska	Janina		
2	Padurska	Genowefa	Adam	10-03-1960
3	Padurska	Maria	Adam	11-05-1990
			Ewa	01-01-1994
			Hanna	04-02-1998
4	Wadurska	Teresa	Monika	03-08-1962
			Roland	03-08-1962

d) Schemat relacji **Osoby**

Osoby(*NrOsoby, Nazwisko, Imię, ImięDziecka, DataUrDziecka*)

jest w pierwszej postaci normalnej.

Relacja *Osoby* o schemacie **Osoby**, reprezentowana przez tabelę 2.31, posiada wiele wad, np. dane osób, które mają więcej niż jedno dziecko, występują kilka razy oraz nie występują dane osób, które nie mają dzieci (np. Adurska Janina), ponieważ kluczem jest zbiór atrybutów {*NrOsoby, ImięDziecka*}, a atrybuty wchodzące w skład klucza są obligatoryjne.

Analizując relację *Osoby* można zauważyć, że dane osoby powtarzają się dla każdego jej dziecka, co znacznie wydłuża i utrudnia wykonywanie operacji na danych, np. aktualizowanie (anomalie przy aktualizacji), oraz bardzo łatwo można utracić spójność danych np. przez zmodyfikowanie nazwiska "Padurska" na "Paderska" tylko dla dziecka o imieniu Ewa lub zmodyfikować wszystkie nazwiska "Padurska" na "Paderska" zamiast tylko Padurskiej Marii. Ponadto występują anomalie przy

Tabela 2.31

Znormalizowana relacja *Osoby* (w pierwszej postaci normalnej)

NrOsoby	*Nazwisko*	*Imię*	*ImięDziecka*	*DataUrDziecka*
2	Padurska	Genowefa	Adam	10-03-1960
3	Padurska	Maria	Adam	11-05-1950
3	Padurska	Maria	Ewa	01-01-1954
3	Padurska	Maria	Hanna	04-02-1960
4	Wadurska	Teresa	Monika	03-08-1962
4	Wadurska	Teresa	Roland	03-08-1962

dopisywaniu danych, np. nie można dopisać danych dziecka bez dopisania danych rodzica i nie można dopisać osoby bez dziecka (atrybut *ImięDziecka* wchodzi w skład klucza głównego, więc nie może mieć wartości NULL). Problemem jest także usuwanie danych – usuwając dane o dzieciach, trzeba sprawdzać, czy jest to ostatnie dziecko tej osoby, i jeśli tak, to trzeba także usunąć dane osoby.

■

***Druga postać normalna* (2PN)** (ang. *second normal form*) – schemat relacji **R**(U) jest w drugiej postaci normalnej, jeśli jest w pierwszej postaci normalnej i każdy atrybut $A \in U$, nie wchodzący w skład żadnego klucza kandydującego, jest w pełni funkcyjnie zależny od każdego klucza kandydującego tego schematu.

Istotą sprowadzania schematu relacji do drugiej postaci normalnej jest zapewnienie, że wszystkie atrybuty niekluczowe zależą od całego klucza głównego, tzn. od wszystkich atrybutów wchodzących w skład klucza (tym samym od każdego całego klucza kandydującego), a nie od jego podzbioru.

Zauważmy, że schematy relacji, które są w pierwszej postaci normalnej i które mają klucz główny prosty (czyli wszystkie klucze kandydujące są kluczami prostymi), są również w drugiej postaci normalnej. Zauważmy także, że nie jest możliwe, aby klucze kandydujące miały różną liczbę atrybutów, ponieważ z definicji klucz kandydujący jest to minimalny zbiór identyfikujący.

W celu uzyskania drugiej postaci normalnej należy z danego schematu relacji utworzyć kilka schematów relacji (na ogół o mniejszej liczbie atrybutów), tak aby

wszystkie atrybuty niekluczowe w poszczególnych schematach relacji były w pełni funkcyjnie zależne od ich kluczy kandydujących.

Sprowadzanie do drugiej postaci normalnej schematu relacji

R(*A*, *B*, *C*, *D*),

w którym zachodzą zależności:

$\{A, B\} \rightarrow C$

$A \rightarrow D$

możemy zapisać następująco:

1) dla danego schematu relacji należy wyznaczyć klucze kandydujące i spośród nich wybrać klucz główny, czyli:
   KK: $\{A, B\}$

i w tym przypadku jest to także klucz główny schematu relacji **R**($\underline{A}$, $\underline{B}$, *C*, *D*).

2) Schemat relacji **R** należy zdekomponować na dwa schematy:
   R1(#$\underline{A}$, $\underline{B}$, *C*), w którym $\{\underline{A}, \underline{B}\} \rightarrow C$

oraz

R2($\underline{A}$, *D*), w którym $\underline{A} \rightarrow D$

W utworzonych schematach **R1** i **R2** będą zachowane zależności funkcyjne i klucze ze schematu **R**. Otrzymane schematy są w 2PN.

Przykład 2.19

a) Dany jest schemat relacji

Osoby(*NrOsoby*, *Nazwisko*, *Imię*, *ImięDziecka*, *DataUrDziecka*)

do opisu danych o pracownikach i ich dzieciach (np. w celu przyznawania zasiłków). Każda osoba, której dane przechowujemy, ma co najmniej jedno dziecko.

W schemacie relacji **Osoby** istnieją następujące zależności funkcyjne:

NrOsoby $\rightarrow$ *Nazwisko*

NrOsoby $\rightarrow$ *Imię*,

{*NrOsoby*, *ImięDziecka*} $\rightarrow$ *DataUrDziecka*

Jedynym kluczem kandydującym w schemacie relacji **Osoby** jest zbiór dwuelementowy {*NrOsoby*, *ImięDziecka*}. Atrybuty niekluczowe *Nazwisko*, *Imię* nie są w pełni funkcyjnie zależne od klucza (bo zależą funkcyjnie od atrybutu *NrOsoby*), czyli schemat relacji **Osoby** nie jest w drugiej postaci normalnej. Dlatego należy zdekomponować go na dwa schematy relacji:

Pracownicy(*NrOsoby*, *Nazwisko*, *Imię*)
Dzieci(*#NrOsoby*, *ImięDziecka*, *DataUrDziecka*)
które są w drugiej postaci. Zapis *#NrOsoby* w schemacie **Dzieci** oznacza klucz obcy (czyli atrybut, który jest kluczem głównym w schemacie **Pracownicy**).

b) Na podstawie poniższych reguł:
   1. Każda firma ma dokładnie jeden adres.
   2. Towar z danej firmy ma ustaloną jedną cenę.
   3. Firma może sprzedawać wiele towarów.
   4. Towar może być sprzedawany przez różne firmy.
   5. Wszystkie atrybuty są obligatoryjne.

zdefiniowano schemat relacji
OfertyFirm(*NazwaFirmy*, *AdresF*, *NazwaTowaru*, *Cena*)
i zależności funkcyjne
$$NazwaFirmy \rightarrow AdresF$$
$$\{NazwaFirmy, NazwaTowaru\} \rightarrow \text{Cena}$$
Kluczem kandydującym i kluczem głównym jest {*NazwaFirmy*, *NazwaTowaru*}.
Schemat
OfertyFirm(*NazwaFirmy*, *AdresF*, *NazwaTowaru*, *Cena*)
nie jest w drugiej postaci normalnej, bo np. *AdresF* zależy funkcyjnie od podzbioru klucza, a nie od całego klucza głównego.

Przykładową relację *OfertyFirm* przedstawiono w tabeli 2.32.

Tabela 2.32

Relacja *OfertyFirm*

NazwaFirmy	*AdresF*	*NazwaTowaru*	*Cena*
Mexa	Główna 1, Opole	pralka	1000
Wawel	Zielna 5, Kraków	pralka	900
Wawel	Zielna 5, Kraków	lodówka	800
Vobo	Główna 1, Opole	lodówka	700
Wawel	Zielna 5, Kraków	telewizor	1500
Mexa	Główna 1, Opole	telewizor	1200

W relacjach o schemacie **OfertyFirm** występuje redundancja danych oraz anomalie dopisywania, modyfikowania i usuwania danych, np.:

- **Redundancja danych**: *AdresF* (adres firmy) pamiętany jest tyle razy, ile towarów dostarcza dana firma.
- **Aktualizacja adresu**: zmiana adresu danej firmy wymaga wielokrotnej aktualizacji adresu, zależnej od liczby dostarczanych towarów, pomimo założenia, że każda firma ma jeden adres.
- **Wprowadzanie danych:** nie można wprowadzić danych adresowych nowo zarejestrowanej firmy, dopóki nie rozpocznie sprzedaży.
- **Usuwanie danych:** po usunięciu danych ostatniego sprzedawanego towaru przez firmę jej dane są automatycznie usuwane z bazy danych.

W celu wyeliminowania anomalii dekomponujemy schemat relacji **OfertyFirm** na dwa schematy, które są w 2PN:

Firmy(*NazwaFirmy*, *AdresF*)

Oferty(#*NazwaFirmy*, *NazwaTowaru*, *Cena*)

W tak zaprojektowanej bazie nie ma już wcześniej występujących anomalii.

Przykładowe relacje o schematach **Firmy** i **Oferty** przedstawiono w tabelach 2.33 i 2.34.

Tabela 2.33

Relacja *Firmy*

NazwaFirmy	*AdresF*
Mexa	Główna 1, Opole
Wawel	Zielna 5, Kraków
Vobo	Główna 1, Opole

Tabela 2.34

Relacja *Oferty*

NazwaFirmy	*NazwaTowaru*	*Cena*
Mexa	pralka	1000
Wawel	pralka	900
Wawel	lodówka	800
Vobo	lodówka	700
Wawel	telewizor	1500
Mexa	telewizor	1200

W schematach, które są w drugiej postaci normalnej, mogą nadal występować anomalie. Na przykład, jeśli pracownik przypisany jest do jednego wieloosobowego pokoju, a w pokoju jest tylko jeden telefon, to w relacjach w schemacie

Kontakty(*Pracownik, Pokój, Telefon*)

występuje redundancja danych i są anomalie, np. przy usuwaniu danych pracownika.

Przed omówieniem trzeciej postaci normalnej wprowadzimy pojęcie **przechodniej nieprzemiennej zależności funkcyjnej**.

Niech A, B, C będą trzema rozłącznymi podzbiorami atrybutów danego schematu relacji.

Podzbiór atrybutów C jest **przechodnio (tranzytywnie) nieprzemiennie funkcyjnie zależny** od podzbioru atrybutów A (ang. *transitively non-commutative dependent*), co zapisujemy

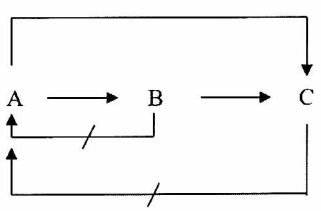
$$A \twoheadrightarrow C$$

jeśli są spełnione cztery warunki:

$$A \rightarrow B,\ B \rightarrow C,$$
$$B \not\rightarrow A,\ C \not\rightarrow A$$

czyli B zależy funkcyjnie od A i C zależy funkcyjnie od B,
ale A nie zależy funkcyjnie od B i A nie zależy funkcyjnie od C.

Przechodnią nieprzemienną zależność funkcyjną przedstawiono na rysunku 2.1.



Rys. 2.1. Przechodnia nieprzemienna zależność funkcyjna podzbioru atrybutów C od A

Trzecia postać normalna **(3PN)** (ang. *third normal form*) – schemat relacji jest w trzeciej postaci normalnej, jeśli jest w drugiej postaci normalnej i każdy atrybut niekluczowy (nie wchodzący w skład żadnego klucza kandydującego) nie jest przechodnio nieprzemiennie funkcyjnie zależny od żadnego klucza kandydującego (brak zależności funkcyjnych między atrybutami niekluczowymi).

Jeśli dla danego schematu relacji zachodzi przechodnia nieprzemienna zależność funkcyjna taka jak na rysunku 2.1, to schemat nie jest w trzeciej postaci normalnej.

Jeśli schemat relacji jest w 2PN i nie ma w nim atrybutów niekluczowych lub jest tylko jeden, to schemat jest także w 3PN.

Przykład 2.20

W schemacie relacji

Faktury(*NrFaktury*, *NazwaKlienta*, *AdresKlienta*)

istnieją następujące zależności funkcyjne:

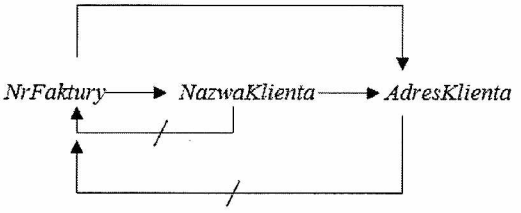
$NrFaktury \rightarrow NazwaKlienta$

$NazwaKlienta \rightarrow AdresKlienta$

Atrybut *AdresKlienta* zależy od atrybutu *NazwaKlienta*, który jest funkcyjnie zależny od *NrFaktury*, natomiast *NrFaktury* nie jest funkcyjnie zależny od atrybutu *NazwaKlienta* (jeden klient może mieć wiele faktur).

Również *NrFaktury* nie zależy funkcyjnie od atrybutu *AdresKlienta*, ponieważ dany adres może być na wielu fakturach. Zatem *AdresKlienta* jest przechodnio nieprzemiennie funkcyjnie zależny od *NrFaktury*, co zilustrowano na rysunku 2.2.

Schemat relacji **Faktury** jest w 2PN, ale nie jest w trzeciej postaci normalnej.



Rys. 2.2. Przechodnia nieprzemienna zależność funkcyjna atrybutów, schemat nie jest w 3PN

Aby schemat relacji doprowadzić do trzeciej postaci normalnej, należy zdekomponować go na nowe schematy relacji w następujący sposób:
dla schematu relacji

$$\mathbf{R}(\underline{A}, B, C)$$

z zależnościami funkcyjnymi

$$A \rightarrow B, B \rightarrow C$$
$$B \not\rightarrow A, C \not\rightarrow A$$

czyli

$$A \rightarrow C$$

należy utworzyć (poprzez odpowiednie operacje rzutowania) dwa schematy relacji

R1($\underline{A}$, $\#B$)

R2($\underline{B}$, C)

w których będą zachowane zależności funkcyjne.

Otrzymane schematy są w 2PN i w 3PN.

Przykład 2.21

Schemat relacji

Faktury(*NrFaktury*, *NazwaKlienta*, *AdresKlienta*),

w którym są zależności funkcyjne:

NrFaktury → *NazwaKlienta*

NazwaKlienta → *AdresKlienta*

jest w 2PN, ale ponieważ jest w nim przechodnia nieprzemienna zależność funkcyjna, więc nie jest w 3PN (przykład 2.20) i należy zastąpić go dwoma schematami relacji:

Wykazy(*NrFaktury*, *#NazwaKlienta*)

Klienci(*NazwaKlienta*, *AdresKlienta*),

które są w 2PN i w 3PN.

■

Przykład 2.22

Dany jest schemat relacji

Zamówienia(*NrZ*, *IdD*, *ND*, *AD*, *NrC*, *NC*, *Ile*, *NrM*, *AdrM*)

oraz zależności funkcyjne

1. *NrZ* → *IdD*
2. *IdD* → *ND*
3. *IdD* → *AD*
4. *NrC* → *NC*
5. *NrC* → *NrM*
6. *NrM* → *AdrM*
7. {*NrZ*, *NrC*} → *Ile*

gdzie

NrZ – numer zamówienia
IdD – identyfikator dostawcy
ND – nazwa dostawcy
AD – adres dostawcy
NrC – numer części
NC – nazwa części
Ile – liczba zamówionych części
NrM – numer magazynu
AdrM – adres magazynu

Klucz kandydujący: {*NrZ*, *NrC*}

Klucz główny: {*NrZ*, *NrC*}

W relacjach o schemacie **Zamówienia** są możliwe redundancje danych oraz mogą występować anomalie podczas operacji usuwania, modyfikacji i dopisywania danych.

Schemat relacji **Zamówienia**(*NrZ*, *IdD*, *ND*, *AD*, *NrC*, *NC*, *Ile*, *NrM*, *AdrM*) jest w 1PN, ale nie jest w 2PN (gdyż np. *IdD* zależy od podzbioru klucza głównego).

Etap 1. Przeprowadzamy dekompozycję schematu **Zamówienia** do schematów będących w 2PN:

Dostawcy_na_zamówieniach(*NrZ*, *IdD*, *ND*, *AD*) (2PN, nie 3PN)
Części_w_magazynie(*NrC*, *NC*, *NrM*, *AdrM*) (2PN, nie 3PN)
Zamawiane_części(#*NrZ*, #*NrC*, *Ile*) (3PN)

Nadal występują anomalie podczas wykonywania operacji CRUD w schematach:

Dostawcy_na_zamówieniach
Części_w_magazynie

Etap 2. Dekomponujemy schematy, które nie są w 3PN do schematów w 3PN. Ze schematu **Dostawcy_na_zamówieniach** otrzymujemy dwa schematy w 3PN:

Zamówienia_do_dostawców(*NrZ*, #*IdD*)
Dostawcy(*IdD*, *ND*, *AD*)

Ze schematu **Części_w_magazynie** otrzymujemy dwa schematy w 3PN:

Lista_części(*NrC*, *NC*, #*NrM*)
Magazyny(*NrM*, *AdrM*)

Schemat relacji **Zamawiane_części**(#*NrZ*, #*NrC*, *Ile*) jest w 3PN i nie wymaga dekompozycji, czyli wszystkie schematy są w 3PN, co kończy proces normalizacji do 3PN. Schemat bazy danych zawiera pięć schematów relacji (każdy jest w 3PN):

Zamówienia_do_dostawców(*NrZ*, #*IdD*)
Dostawcy(*IdD*, *ND*, *AD*)
Lista_części(*NrC*, *NC*, #*NrM*)
Magazyny(*NrM*, *AdrM*)
Zamawiane_części(#*NrZ*, #*NrC*, *Ile*)

Podczas dekompozycji zamiast wprowadzania określonych nazw dla nowych schematów relacji można zastosować następującą konwencję tworzenia nazw schematów: ze schematu Z po dekompozycji do 2PN powstają schematy o nazwach Z1, Z2, następnie ze schematu Z1 po dekompozycji do schematów w 3PN powstają schematy Z11, Z12, a ze schematu Z2 powstają schematy Z21, Z22.

Proces dekompozycji w przyjętej notacji dla schematu relacji

Z(*NrZ, IdD, ND, AD, NrC, NC, Ile, NrM, AdrM*)

który nie jest w 2PN, ma postać:

Etap 1. Dekompozycja do schematów relacji w 2PN

Z1(*NrZ, IdD, ND, AD*)	nie 3PN	(Dostawcy na zamówieniach)
Z2(*NrC, NC, NrM, AdrM*)	nie 3PN	(Części w magazynie)
Z3(*#NrZ, #NrC, Ile*)	3PN	(Zamawiane części)

W schematach **Z1** i w **Z2** występują nadal anomalie.

Etap 2. Dekompozycja do schematów relacji w 3PN (schemat bazy danych)

Z11(*NrZ, #IdD*)	z dekompozycji Z1 (Zamówienia do dostawców)
Z12(*IdD, ND, AD*)	z dekompozycji Z1 (Dostawcy)
Z21(*NrC, NC, #NrM*)	z dekompozycji Z2 (Lista części)
Z22(*NrM, AdrM*)	z dekompozycji Z2 (Magazyny)
Z3(*#NrZ, #NrC, Ile*)	(Zamawiane części)

Poszczególne etapy normalizacji schematu relacji przedstawiono w tabeli 2.35.

Tabela 2.35

Etapy normalizacji schematu relacji

Tabela nieznormalizowana	Wartości w tabeli **nie są atomowe**, tabela nie reprezentuje relacji.
Pierwsza postać normalna	Wszystkie wartości atrybutów relacji o danym schemacie są **atomowe**.
Druga postać normalna	Schemat jest w pierwszej postaci normalnej i atrybuty niekluczowe są **w pełni funkcyjnie zależne** od każdego klucza kandydującego.
Trzecia postać normalna	Schemat relacji jest w drugiej postaci normalnej i każdy atrybut niekluczowy **nie jest przechodnio nieprzemiennie funkcyjnie** zależny od żadnego klucza kandydującego.

W praktyce normalizację schematu relacji kończy się najczęściej na trzeciej postaci normalnej. Jednak w schemacie będącym w trzeciej postaci normalnej mogą występować anomalie związane z operacjami CRUD.

Przykład 2.23

Załóżmy, że w schemacie relacji

Zajęcia(*Student*, *Kurs*, *Wykładowca*)

zachodzą zależności funkcyjne:

{*Student*, *Kurs*} → *Wykładowca*	(student może uczęszczać na dany kurs tylko do jednego wykładowcy)
Wykładowca → *Kurs*	(każdy wykładowca prowadzi dokładnie jeden kurs)

W schemacie relacji **Zajęcia** są dwa klucze kandydujące:

{*Student*, *Kurs*} oraz {*Student*, *Wykładowca*}

W schemacie relacji **Zajęcia**, który jest w drugiej i w trzeciej postaci normalnej (wszystkie atrybuty są kluczowe), nadal występują anomalie związane z usuwaniem danych (po usunięciu danych o ostatnim studencie uczęszczającym na dany kurs tracimy dane o tym kursie i o wykładowcy) oraz anomalie dopisywania danych (nie można dopisać kursu i wykładowcy, zanim na kurs nie zapisze się co najmniej jeden student).

W teorii relacyjnych baz danych definiowane są kolejne postaci normalne:

- Postać normalna Boyce'a-Codda (PNBC)
- Czwarta postać normalna (4PN)
- Piąta postać normalna (5PN).

Postać normalna Boyce'a-Codda **(PNBC)** (ang. *Boyce-Codd normal form*) jest to poprawiona wersja trzeciej postaci normalnej.

Schemat relacji **R** jest w postaci normalnej Boyce'a-Codda, jeśli wszystkie zależności funkcyjne w schemacie relacji determinowane są przez klucze (**wszystkie atrybuty zależą tylko i wyłącznie od kluczy lub od nadkluczy**).

Z definicji PNBC wynika, że w schemacie relacji **R** będącym w PNBC, jeśli zbiór atrybutów Z jest wyznacznikiem jakiegokolwiek atrybutu spoza Z, to zbiór Z musi być wyznacznikiem każdego atrybutu w tym schemacie (Z jest kluczem lub zawiera klucz).

Schemat relacji, który jest w postaci normalnej Boyce'a-Codda, nie zawiera redundancji, niespójności i anomalii.

W schemacie relacji, który jest w trzeciej postać normalnej, nie ma zależności funkcyjnych między atrybutami niekluczowymi, ale mogą być:

- zależności między atrybutami kluczowymi (atrybutami wchodzącymi w skład klucza kandydującego),
- zależności atrybutów kluczowych od atrybutów niekluczowych,
- zależności atrybutów kluczowych od części klucza, do którego nie należą.

Trzecia postać normalna dopuszcza więc istnienie w schemacie relacji zależność postaci $A \rightarrow B$, gdzie A wchodzi w skład klucza kandydującego, a B jest atrybutem klucza głównego (zob. przykł. 2.23). Zależność ta jest przyczyną redundancji, którą może wyeliminować silniejsza postać normalna, czyli postać normalna Boyce'a-Codda.

W schemacie relacji w PNBC każdy wyznacznik zależności funkcyjnej jest kluczem kandydującym.

Postać normalna Boyce'a-Codda określa, że nie mogą istnieć zależności funkcyjne między kluczami kandydującymi.

Przykład 2.24

W schemacie relacji

Zajęcia(*Student*, *Kurs*, *Wykładowca*),

w którym zachodzą zależności funkcyjne:

{*Student*, *Kurs*} → *Wykładowca*	(student może uczęszczać na dany kurs tylko do jednego wykładowcy)
Wykładowca → *Kurs*	(każdy wykładowca prowadzi dokładnie jeden kurs),

są dwa klucze kandydujące: {*Student*, *Kurs*} i {*Student*, *Wykładowca*}.

Przypadek I

Wybierzmy jako klucz główny {*Student*, *Kurs*}.

W schemacie relacji **Zajęcia** nie ma atrybutów niekluczowych, więc jest w 2PN i w 3PN, ale nie jest w postaci normalnej Boyce'a-Codda, dlatego występują anomalie usuwania i dopisywania danych.

Schemat relacji **Zajęcia** można zastąpić dwoma schematami, które są w PNBC:

Przydział(*Wykładowca*, *Kurs*)

Zapisy(*Student*, #*Kurs*)

ale to powoduje utratę zależności

{*Student*, *Kurs*} → *Wykładowca*

Przypadek II

Wybierzmy jako klucz główny {*Student*, *Wykładowca*}.

Schemat relacji **Zajęcia** można zastąpić dwoma schematami, które są w PNBC:

Przydział(*Wykładowca*, *Kurs*)

Zapisy(*Student*, #*Wykładowca*)

ale to także powoduje utratę zależności {*Student*, *Kurs*} → *Wykładowca*, przez co student może zapisać się kilka razy na ten sam kurs prowadzony przez różnych wykładowców.

Nie każdy schemat relacji można sprowadzić do zbioru schematów relacji w postaci normalnej Boyce'a-Codda bez utraty zawartych w relacjach danych i z zachowaniem zależności funkcyjnych, np. schematu nie da się zdekomponować na kilka schematów z zachowaniem zależności funkcyjnych, jeśli jedna z zależności funkcyjnych obejmuje wszystkie atrybuty.

W praktyce silniejsza postać normalna Boyce'a-Codda nie zawsze jest pożądana, gdyż może doprowadzić do utraty pewnych zależności funkcyjnych, dlatego często jest lepiej poprzestać na trzeciej postaci normalnej.

W schematach relacji oprócz zależności funkcyjnych często występują zależności wielowartościowe.

Oprócz zależności funkcyjnych rozważa się także zależności wielowartościowe oznaczane jako

$$X \rightarrow\!\!\!\rightarrow Y$$ (co czytamy: X wyznacza wielowartościowo Y),

które służą do zdefiniowania czwartej postaci normalnej.

Załóżmy, że X, Y, Z są rozłącznymi zbiorami atrybutów schematu relacji **R**, *R* jest relacją o schemacie **R**, $x \in X$, $y, y' \in Y$, $z, z' \in Z$.

Zależność funkcyjną $X \rightarrow Y$ można interpretować jako regułę:

jeżeli krotki (x, y, z) i (x, y', z') $\in R$, to $y = y'$,

natomiast zależność wielowartościową $X \rightarrow\!\!\!\rightarrow Y$ można interpretować jako regułę:

jeżeli krotki (x, y, z) i (x, y', z') $\in R$, to także krotki (x, y, z'), (x, y', z) $\in R$.

Zależność wielowartościowa (ang. *multivalued dependency*) zbioru atrybutów Y od zbioru atrybutów X, co zapisujemy jako

$$X \rightarrow\!\!\!\rightarrow Y$$

oznacza, że w schemacie relacji, który zawiera dowolne podzbiory atrybutów X, Y, Z, zbiór wartości Y odpowiadający danej parze (wartość X, wartość Z) zależy jedynie od wartości X i jest niezależny od Z.

Zależność funkcyjna X → Y oznacza, że krotki wartości atrybutów zbioru X wyznaczają jednoznacznie krotkę wartości atrybutów zbioru Y, natomiast zależność wielowartościowa X →→ Y oznacza, że krotki wartości atrybutów zbioru X wyznaczają **zbiór** krotek wartości atrybutów zbioru Y.

Każda zależność funkcyjna jest trywialną zależnością wielowartościową.

Przykład 2.25

Schemat relacji

Osoby(*NrPrac*, *Stanowisko*, *ImięDziecka*)

jest w pierwszej postaci normalnej i zawiera dwie wielowartościowe zależności:

(1) *NrPrac* →→ *Stanowisko*

czyli dla dowolnej pary wartości atrybutów (*NrPrac*, *ImięDziecka*) atrybut *Stanowisko* zależy tylko od atrybutu *NrPrac*, a jest niezależny od atrybutu *ImięDziecka*,

(2) *NrPrac* →→ *ImięDziecka*

czyli dla dowolnej pary wartości atrybutów (*NrPrac*, *Stanowisko*) atrybut *ImięDziecka* zależy tylko od atrybutu *NrPrac* i jest niezależny od atrybutu *Stanowisko*.

Atrybutowi *NrPrac* w relacji *Osoby* o schemacie **Osoby** może odpowiadać wiele wartości atrybutu *Stanowisko* (dotychczas zajmowane stanowiska w firmie) i wiele wartości atrybutu *ImięDziecka* (tabela 2.36).

Tabela 2.36

Relacja *Osoby*

NrPrac	*Stanowisko*	*ImięDziecka*
1	dyrektor	Adam
2	programista	Adam
2	projektant	Adam
2	programista	Ewa
2	projektant	Ewa
2	programista	Anna
2	projektant	Anna
3	programista	Monika
3	analityk	Monika
3	programista	Ewa
3	analityk	Ewa

Relacja *Osoby* zawiera dużo powtarzających się danych. W relacjach o schemacie **Osoby,** który jest w postaci normalnej Boyce'a-Codda (ponieważ zawiera same atrybuty kluczowe), mogą występować anomalie dopisywania, edycji i usuwania danych.

***Czwarta postać normalna* (4PN)** – schemat relacji jest w czwartej postaci normalnej wtedy i tylko wtedy, gdy jest w PNBC i nie zawiera wielowartościowej nietrywialnej zależności atrybutów.

Zależność jest trywialna (oczywista), jeżeli zawsze jest spełniona (nie może być nie spełniona).

Sprowadzenie schematu relacji do czwartej postaci normalnej polega na usunięciu wielowartościowych zależności funkcyjnych.

Sprowadzenie dowolnego schematu relacji do PNBC jest zawsze możliwe.

Schemat relacji **Osoby** (przykład 2.24) nie jest w 4PN ponieważ zawiera nietrywialne wielowartościowe zależności funkcyjnych.

Przykład 2.26

Schemat relacji (z przykładu 2.25)

Osoby(*NrPrac*, *Stanowisko*, *ImięDziecka*)

jest w PNBC, ale nie jest w 4PN, ponieważ zawiera dwie nietrywialne wielowartościowe zależności funkcyjne:

$NrPrac \rightarrow\!\!\!\rightarrow Stanowisko$

$NrPrac \rightarrow\!\!\!\rightarrow ImięDziecka$

Schemat należy zdekomponować na dwa schematy, które są również w PNBC:

Pracownicy(*NrPrac*, *Stanowisko*)

Dzieci(*NrPrac*, *ImięDziecka*)

Schematy relacji **Pracownicy** i **Dzieci** są w 4PN.

Relację *Osoby* przedstawioną w tabeli 2.36 o schemacie **Osoby** zastępujemy relacjami:

Pracownicy (tabela 2.37) o schemacie **Pracownicy**,

Dzieci (tabela 2.38) o schemacie **Dzieci**.

Relację *Osoby* można otrzymać ponownie przez złączenie naturalne relacji *Pracownicy* i *Dzieci*.

Tabela 2.37

Relacja *Pracownicy*

NrPrac	*Stanowisko*
1	dyrektor
2	programista
2	projektant
3	programista
3	analityk

Tabela 2.38

Relacja *Dzieci*

NrPrac	*ImięDziecka*
1	Adam
2	Adam
2	Ewa
2	Anna
3	Monika
3	Ewa

Aby zdefiniować piątą postać normalną, wprowadzimy pojęcie zależności złączenia.

Niech A, B, ..., Z będą dowolnymi podzbiorami zbioru atrybutów w schemacie relacji **R**.

Mówimy, że relacja *R* o schemacie **R** spełnia **zależność złączenia** (A, B, ..., Z) wtedy i tylko wtedy, gdy relacja *R* jest równa złączeniu swoich rzutów na A, B, ..., Z.

Piąta postać normalna **(5PN) – postać normalna rzutu-złączenia** – schemat relacji **R** jest w piątej postaci normalnej wtedy i tylko wtedy, gdy każda zależność złączenia w **R** jest implikowana kluczami kandydującymi schematu relacji.

Schemat relacji jest w piątej postaci normalnej, jeżeli jest w czwartej postaci normalnej i nie istnieje odwracalny rozkład na schematy relacji.

Schemat relacji, który jest w 5PN, nie zawiera anomalii, które można usunąć za pomocą operacji projekcji.

Piąta postać normalna w praktyce ma zastosowanie w wyjątkowo rzadkich przypadkach.

Przykład 2.27

Schemat relacji

Dostawy(*Dostawca*, *Towar*, *Odbiorca*),

który jest w 4PN, ale nie jest w 5PN, zastępujemy trzema schematami relacji, które są w 5PN:

Kto_Co(*Dostawca*, *Towar*)
Kto_Komu(*Dostawca*, *Odbiorca*)
Co_Komu(*Towar*, *Odbiorca*)

W niniejszym rozdziale przedstawiono sposób tworzenia poprawnego schematu bazy danych jako zbioru schematów relacji (rozdz. 4.3) poprzez wyjście od jednego uniwersalnego schematu relacji i określenie zależności funkcyjnych, a następnie przeprowadzenie procesu normalizacji i sprowadzenie schematów relacji co najmniej do trzeciej postaci normalnej.

Na zakończenie tego rozdziału należy podkreślić, że normalizacja jest procesem tworzenia schematów relacji, które nie podlegają anomaliom. Dane przechowywane w relacjach znormalizowanych zajmują mniej miejsca, nie powtarzają się (brak redundancji danych), łatwiejsza jest ich modyfikacja, dopisywanie oraz ich usuwanie (nie ma anomalii przy dodawaniu, usuwaniu, aktualizowaniu danych).

Głównym celem normalizacji jest usunięcie redundancji z bazy danych. C.J. Date definiuje redundancję następująco: jeżeli w relacji R występuje zależność funkcyjna $A \rightarrow B$ oraz A nie jest kluczem kandydującym i relacja ma co najmniej dwie krotki, to w relacji R występuje **redundancja** [Date2000].

Proces normalizacji ma na celu sprowadzenie schematów relacji do takich postaci, w których:

- wartości atrybutów są atomowe (jedna wartość z dziedziny) – 1PN,
- atrybuty niekluczowe zależą od całego klucza (a nie od jego podzbioru) – 2PN,
- atrybuty niekluczowe zależą tylko od klucza (a nie od innych atrybutów niekluczowych) – 3PN.

Zbyt duża liczba schematów relacji wynikająca z normalizacji wymaga częstego wykonywania operacji złączenia, co jest czasochłonne. Jeśli projektujemy bazę dla szkoły gminnej, a w gminie jest dużo rodzin wielodzietnych, to pożądane jest, aby zamiast do schematu relacji **Uczniowie** wpisywać dane rodzica tyle razy, ile ma dzieci w szkole, był osobny schemat relacji dla danych rodziców (opiekunów). Takie podejście natomiast może nie być sensowne przy projekcie bazy dla uczelni wyższej, gdzie procentowy udział rodzeństw w bazie jest bardzo mały.

W praktyce często po przeprowadzonej normalizacji schematów przeprowadza się denormalizację [Barker1996, Harris1994] polegającą na łączeniu niektórych znormalizowanych schematów relacji (np. gdy redundancje danych nie są zbyt duże) w celu przyspieszenia operacji wykonywanych na danych. Jeśli projektant przewiduje, że często będą wykonywane operacje wymagające łączenia relacji, które są bardzo czasochłonne, to musi dokonać wyboru, czy lepiej wykonać denormalizację (pojawią się znowu anomalie, które eliminowała normalizacja), czy pogodzić się z bardziej złożonym i wolniejszym wykonywaniem operacji na danych.

2.4. PODSTAWOWE POJĘCIA BAZ DANYCH

Na podstawie poprawnie zdefiniowanego schematu bazy danych będącego zbiorem znormalizowanych schematów relacji można zaimplementować fizyczną bazę danych. W przypadku relacyjnej bazy danych reprezentacją relacji o danym schemacie relacji jest tabela o strukturze odpowiadającej schematowi relacji.

Podstawowe własności relacji wynikające z definicji to:

1. Relacja jest zbiorem krotek, więc nie zawiera dwóch identycznych krotek.
2. Kolejność krotek w relacji jest dowolna (elementy zbioru bez określonego porządku).
3. Kolejność atrybutów w relacji jest dowolna, ale ustalona.
4. Składowe krotek są wartościami elementarnymi (atomowymi) należącymi do dziedziny atrybutu.

Reprezentowanie relacji w postaci tabel sprawia, że relacyjny model danych jest czytelny i zrozumiały.

Tabela (ang. *table*) jest jedną z najczęstszych reprezentacji relacji. W tabeli wyróżnia się kolumny i wiersze.

Kolumna (ang. *column)* zawiera wartości z danej dziedziny, a nazwa kolumny jest odpowiednikiem atrybutu schematu relacji.

Wiersz (ang. *row*) odpowiada krotce relacji.

Pole (ang. *field*) jest to przecięcie wiersza i kolumny, jest odpowiednikiem składowej krotki. Wartość każdego pola powinna być wartością atomową (elementarną). Odwołanie do pola następuje poprzez nazwę kolumny (nazwę pola).

Należy pokreślić, że określenie *tabela* i *relacja* to nie jest to samo. Tabela jest jedynie wizualnym sposobem przedstawienia relacji. W relacji nie ma dwóch takich samych krotek, a w tabeli mogą być dwa takie same wiersze. Reprezentacja relacji w postaci tabeli sugeruje wiele nieprawdziwych skojarzeń, np. że wiersze tabeli (czyli krotki relacji) są uporządkowane w przedstawiony sposób, że jest wiersz pierwszy, następny, poprzedni (czyli że w relacji jest krotka pierwsza, następna, poprzednia), że kolumny tabeli (czyli atrybuty schematu relacji) są uporządkowane od lewej do prawej – wszystkie te stwierdzenia są oczywiście nieprawdziwe.

Tabelę można traktować jako obraz relacji, jeśli ustalimy odpowiednie reguły interpretacji takiej reprezentacji, które podajemy poniżej.

Każda tabela ma swoją unikalną nazwę odpowiadającą nazwie schematu relacji, którego relację reprezentuje. Struktura tabeli odpowiada schematowi relacji. Tabela składa się z **nagłówka**, zawierającego nazwy kolumn (odpowiadające nazwom atrybutów), oraz z wierszy (odpowiedników krotek), stanowiących **treść** tabeli. Nazwa kolumny musi być jednoznaczna w obrębie danej tabeli. Kolejność kolumn jest dowolna, ale ustalona. Klucze główne schematów relacji są kluczami głównymi w tabelach. Klucze obce ze schematów relacji są kluczami obcymi w tabelach.

Tabela odpowiadająca schematowi relacji powinna mieć utworzony klucz główny. Mogą jednak istnieć tabele bez kluczy głównych.

Każdy wiersz tabeli odpowiada jednej krotce relacji i wówczas musi się różnić od pozostałych wierszy w tabeli chociaż jedną wartością. W tabeli bez ustalonego klucza mogą wystąpić powtarzające się wiersze.

Wynikiem operacji na relacjach są nienazwane relacje (bez powtarzających się krotek), a wynikiem wykonania zapytań na tabelach są nienazwane tabele, w których wiersze mogą się powtarzać.

Kolejność wierszy w tabelach jest dowolna.

Tabela może być wypełniana przez dopisywanie wierszy. Dane w tabeli można zmieniać. Można usuwać wiersze z tabeli. Można wyszukiwać i sortować dane.

Czynności takie jak dopisywanie kolumn do tabeli, modyfikowanie nazw kolumn tabeli, usuwanie kolumn z tabeli (czyli zmiany struktury tabeli) również można wykonywać, jednak w praktyce struktura bazy danych (w tym tabel) powinna być stała i w zaimplementowanej bazie danych takie operacje wykonuje się bardzo rzadko i powinno się ich unikać.

Zwróćmy uwagę, że w tabeli może być kilka kolumn o takiej samej dziedzinie, przy czym nazwy kolumn muszą być różne, np. w bazie poborowych dla dwóch różnych kolumn *Imię* i *Imię Ojca*, może być ta sama dziedzina o nazwie *Imię* będąca zbiorem imion jako ciągów znaków maksymalnie 15-literowych. Kolejność dziedzin w relacji matematycznej jest ściśle ustalona, ale w relacjach w modelu relacyjnym nie ma ona znaczenia, z kolei ma znaczenie np. w strukturalnym języku zapytań do zarządzania bazami danych **SQL** ([DateDar2000], s. 102).

Podkreślmy, że w bazie danych przechowywane są dane (a nie informacje).

Omówimy teraz pojęcia: wartość, dana, informacja, wiedza i mądrość.

Wartość to konkretna liczba, znak, data, tekst itd., np. 5, "Agata", 07-05-2020. Wartości trudno nadać znaczenie.

Dana (ang. *data, datum*) to para **(nazwa, wartość)** – podstawowa jednostka w bazach danych, bez kontekstu nie ma większego znaczenia. Dane mogą być przetwarzane i prezentowane. Angielskie słowo *datum,* wywodzące się z języka łacińskiego, oznacza dosłownie „fakt, przesłankę".

Dane przechowywane w bazie danych to na przykład: (*Peron*, 5), (*Nazwisko*, "Agata"), (*Imię*: "Karol"), (*Kod*, 07-05-2020). Wartości danych są trwałe, czyli nie ulegają samoistnym zmianom aż do ich zmodyfikowania bądź usunięcia (ręcznego lub przez jakiś zautomatyzowany proces). Same w sobie są niezrozumiałe, można im nadać różne interpretacje. Nawet gdybyśmy wiedzieli, że wartość 12/02/1970 oznacza datę, to i tak dalej nie wiemy, co ta data oznacza, jak ją interpretować.

Informacje to dane zinterpretowane, przetworzone w sposób, który uwidacznia ich znaczenie, są dynamiczne w tym sensie, że podlegają ciągłym zmianom w stosunku do danych przechowywanych w bazie danych i można je przetwarzać na nieograniczoną liczbę sposobów. Informacją jest wyświetlenie na ekranie komputera komunikatu „Pracownik: Karol Agata, data urodzenia: 02 grudnia 1970". Informacje można przedstawiać na różne sposoby (jako tekst komunikatu, arkusz kalkulacyjny, formularz, raport, wyświetlić na ekranie komputera lub wydrukować na papierze, w formie tabeli lub na wykresie itp.).

Dane muszą zostać poddane przetworzeniu, aby stały się informacjami.

Baza danych dzięki poprawnemu skonstruowaniu powinna umożliwiać przetwarzanie danych i dostarczanie użytkownikom odpowiednich informacji.

Wiedza jest to zbiór reguł i warunków, które pozwalają podjąć decyzję na podstawie dostępnych informacji. Przechowywana w bazie danych wartość 12 nic nie oznacza, można ją różnie zinterpretować (godzina, dzień, miesiąc, cena towaru, liczba studentów w laboratorium, numer domu itd.). Dana (*Staż*, 12) stanie się informacją, jeśli będziemy wiedzieli, że oznacza ona staż pracy pracownika Jana Kowalskiego. Kierownik zakładu, korzystając z informacji dotyczących stażu pracy pracowników i przepisów firmy (np. że pracownik po przepracowaniu 12 lat otrzymuje nagrodę), może przyznać nagrodę pracownikowi. Znając więc przepisy firmy i staż pracy pracownika, wiemy, że dostał nagrodę. Wiedza w węższym znaczeniu to ogół wiarygodnych informacji wraz z umiejętnością ich wykorzystania. Z wyników badań pacjenta lekarz otrzymuje informacje dotyczące jego zdrowia. Aby postawić diagnozę, musi dysponować jednak wiedzą medyczną.

Mądrość to umiejętność korzystania z wiedzy, doświadczenia i intuicji, np. podczas projektowania baz danych.

Dane przechowywane w bazie danych powinny być:

- pełne, np. numer telefonu musi zawierać także numer kierunkowy,
- aktualne, np. musi być przechowywana data zatrudnienia, a nie staż pracy,
- trwałe, np. dane z rozliczeń finansowych są przechowywane przez 5 lat,
- odpowiednie, np. kurs waluty ma odpowiednią liczbą miejsc po przecinku, w bazie są wszystkie nazwy miejscowości udostępnianych użytkownikowi do wyboru miejsca zamieszkania.

Dane w systemie bazy danych obejmują:

- dane **pamiętane w bazie danych**, np.
  - dane pracownika: Nazwisko, Imię, Pensja, DataUr, Płeć, PESEL;
- dane **wejściowe:**
  - wprowadzane przez **użytkownika**, np.
    - procent nagrody jubileuszowej zależnej od pensji,
    - parametry wyszukiwania: nazwisko, zakres płac, zakres dat;
  - pobierane z **bazy danych** przez użytkownika, np.
    - pensja pracownika,
    - adres zamieszkania klienta;
  - pobierane z **systemu** (SZBD, operacyjnego), np. aktualna data, czas;
- dane **wyjściowe**, np.
  - wyliczona premia za dany okres,

– wyszukane dane pracownika,
– staż pracy każdego pracownika obliczony na podstawie danych z bazy (nie wymaga danych wejściowych od użytkownika).

Relacyjna baza danych (ang. *relational database*) – baza danych oparta na relacyjnym modelu danych. W praktycznych implementacjach założenia modelu relacyjnego sformułowane przez E.F. Codda są modyfikowane lub nie są przestrzegane. Z tego względu wiele z baz danych określanych jako relacyjne tak naprawdę relacyjna nie jest [Codd1985a]. Relacyjna baza danych jest postrzegana jako zbiór nazwanych dwuwymiarowych tabel o określonej liczbie kolumn i nieograniczonej (ale skończonej) liczbie wierszy, co nie oznacza, że dane są pamiętane w postaci fizycznych tablic. Wartości przechowywane w relacyjnej bazie danych są atomowe i jest do nich zapewniony dostęp poprzez języki zapytań wysokiego poziomu (np. QBE, SQL). Na danych można wykonywać operacje zdefiniowane w algebrze relacji.

System zarządzania bazą danych **(SZBD)** (ang. *Database Management System* – DBMS) – oprogramowanie narzędziowe umożliwiające zakładanie, przechowywanie i wydajne przetwarzanie zbiorów danych, np. Oracle, Ingres, Progress, Access, dBase, ObjectStore, Jasmine, GemStone, Versant, Firebird, PostgreSQL (Postgres), MySQL, SQLite, MariaDB, SQL Server, DB2. Podstawowe funkcje SZBD to: zapewnienie bezpieczeństwa i integralności danych; umożliwianie autoryzacji użytkowników, kontroli dostępu do danych i przetwarzania transakcji; obsługa wielodostępu; utrzymywanie dzienników systemowych; tworzenie i uaktualnianie słownika danych; tworzenie kopii zapasowych i odtwarzanie bazy po awariach.

System zarządzania relacyjną bazą danych **(SZRBD)** (ang. *Relational Database Management System* – RDBMS) – oprogramowanie narzędziowe wykorzystywane do zakładania, przechowywania i przetwarzania relacyjnych baz danych, np. Oracle, Sybase, Informix, dBase, Ingres, Progress, Access.

System bazy danych (ang. *database system*) jest to baza danych wraz z jej systemem obsługi wykorzystującym określony SZBD i inne programy narzędziowe. Według C.J. Date [Date2000] system bazy danych to: **użytkownicy, dane, sprzęt** (np. komputer, kamera, drukarka, skaner) i **oprogramowanie** (m.in. SZBD, programy narzędziowe, narzędzia do budowania aplikacji, pomoce projektowe, edytory, arkusze kalkulacyjne, przeglądarki internetowe, generatory raportów itd.).

Trwałość (ang. *durability*) bazy danych oznacza, że dane są trwale przechowywane w bazie danych i dostępne dla uprawnionych użytkowników przez określony czas. Na przykład, jeśli faktury mają być przechowywane przez 5 lat, to muszą odnosić się do danych klienta, nazw i cen towarów z okresu wystawienia faktury, a to wymaga odpowiedniego zaprojektowania struktury danych.

Transakcja (ang. *transaction*) jest to ciąg poleceń (operacji) przeznaczonych do wykonywania w całości albo wcale (musi być wówczas przywrócony stan sprzed rozpoczęcia transakcji). Na przykład przelanie pewnej kwoty pieniędzy z jednego konta na drugie wiąże się z wykonaniem transakcji składającej się z kilku czynności takich jak sprawdzenie, czy na pierwszym koncie jest żądana kwota pieniędzy, i jeśli tak, to odjęcie danej kwoty z pierwszego konta oraz dodanie kwoty do stanu drugiego konta.

Perspektywa (ang. *view*) – inaczej widok, to wirtualna tabela, zdefiniowana za pomocą innych wejściowych tabel, zawiera dane z wybranych kolumn i wierszy z jednej lub więcej tabel, nie może istnieć samodzielnie [Date2000].

Bezpieczeństwo danych (ang. *data security*) polega na ich ochronie, czyli na zabezpieczeniu przed nieuprawnionym lub nieprawidłowym, przypadkowym lub umyślnym ujawnieniem, modyfikacją lub zniszczeniem danych. Wyróżnia się cztery podstawowe aspekty bezpieczeństwa danych:

- poufność,
- integralność,
- dostępność,
- spójność.

Poufność (ang. *confidentiality*) oznacza niedostępność danych dla podmiotów nieuprawnionych. Bezpośrednim sposobem zapewnienia poufności jest szyfrowanie danych. Procedury uwierzytelniania, ograniczania uprawnień dostępu czy ograniczanie fizycznego dostępu do danych są środkami pośrednimi, prowadzącymi do osiągnięcia tego celu. Pomimo stosowania różnego rodzaju środków zapewniających poufność istnieje niebezpieczeństwo przypadkowego lub celowego jej naruszenia. Dlatego system ochrony powinien nie tylko zapewniać poufność, lecz także gwarantować możliwość wykrycia każdej próby i przypadku jej naruszenia.

Integralność danych (ang. *data integrity*) oznacza, że dane nie zostaną zmienione w żaden nieuprawniony sposób, są stale zgodne z rzeczywistością, są

z sobą niesprzeczne, odzwierciedlają aktualny stan danych w rzeczywistości oraz ich zmiany. Integralność danych wiąże się z zapewnieniem danym:

- **dokładności** (ang. *accuracy*), m.in. z poprawnym zdefiniowaniem typów danych, np. dla dat, czasu, ocen, pensji,
- **prawdziwości** (ang. *correctness*), np. w bazie musi być uwzględniona możliwość zapisywania wielu adresów dla danej osoby (zameldowania, zamieszkania, do korespondencji),
- **aktualności, ważności** (ang. *validity*), np. w bazie zamiast wieku osoby należy zapisywać jej datę urodzenia.

Integralność referencyjną w bazach danych zapewnia się poprzez definiowanie kluczy głównych i więzów integralności z kluczami obcymi. W bazie danych pomiędzy dwiema tabelami można wymusić tylko jedno powiązanie z wymuszonymi więzami integralności.

Dostępność (ang. *availability*) danych oznacza możliwość korzystania z określonych danych w określonym czasie, na żądanie uprawnionych użytkowników.

Spójność (ang. *consistency*) bazy danych oznacza poprawność poszczególnych danych oraz poprawność bazy danych jako całości. Zapewnienie spójności bazy danych jest podstawowym problemem związanym z projektowaniem bazy i definiowaniem transakcji. Gdyby w trakcie wykonywania transakcji przepisywania ucznia z dziennika klasy Ia do dziennika klasy Ic po wypisaniu ucznia z Ia nastąpiła awaria systemu, to w celu zapewnienia spójności bazy danych należy przywrócić stan bazy do stanu początkowego (zanim rozpoczęła się transakcja). Transakcja powinna więc być w całości zrealizowana poprawnie albo wcale. Źle zaprojektowana baza danych może prowadzić do niespójności danych, np. schematy relacji

Klasy(*Symbol*, *LiczbaUcz*),
Uczniowie(*IdU*, *Nazwisko*, *Imię*, *#Symbol*)

naruszają spójność bazy, ponieważ liczba uczniów w danej klasie wynika z dwóch niezależnych od siebie atrybutów: *LiczbaUcz* oraz klucza obcego *Symbol* w schemacie relacji **Uczniowie** i otrzymane wartości mogą być różne.

W każdej chwili baza danych znajduje się w pewnym stanie, czyli jest konkretnym zbiorem danych. Należy zadbać o to, by baza danych znajdowała się w stanie spójnym i integralnym. Stan bazy nazwiemy integralnym, jeśli wartości zgromadzonych danych są zgodne z wartościami w świecie rzeczywistym. Stan ten

może się zmienić wskutek wykonania pewnych transakcji, ale nadal powinny być poprawne. Wszelkie zmiany w bazie zachodzą w sposób dyskretny.

Baza danych może być wykorzystywana w tym samym czasie nie tylko przez różnych użytkowników tej samej aplikacji, ale również może być wykorzystywana przez różne systemy. Istnieje wówczas potrzeba współdzielenia danych przez kilka aplikacji (procesów). Mówimy wówczas o współdzielonych bazach danych. Współbieżny dostęp do danych współdzielonych może powodować ich niespójność. Zagadnienia te wykraczają jednak poza ramy niniejszego opracowania.

W literaturze można znaleźć również informacje o bazach danych innych typów niż relacyjne, np. obiektowych, sieciowych, hierarchicznych, nierelacyjnych (NoSQL), dedukcyjnych, temporalnych, dokumentów XML, strumieniowych, wielowersyjnych, rozproszonych. Zainteresowanych Czytelników zachęcamy do zapoznania się z tymi zagadnieniami we własnym zakresie.

Bazy danych możemy podzielić na bazy analityczne, operacyjne oraz na hurtownie danych.

Operacyjna baza danych to baza, w której podstawowymi operacjami na danych są operacje dopisywania, usuwania, modyfikowania i wyszukiwania danych.

Analityczna baza danych to baza, której dane często pochodzą z bazy analitycznej, ale po wprowadzeniu do tej bazy są stałe, nie podlegają modyfikacjom, tylko analizie. Główne operacje wykonywane na tych danych to wyszukiwanie danych, sporządzanie zestawień statystycznych, przeprowadzanie analiz i prognoz. Baza analityczna (archiwalna) na ogół jest zupełnie inaczej zaprojektowana niż baza operacyjna.

Hurtownia danych (ang. *data warehouse*) to zintegrowana, zorientowana tematycznie zmienna w czasie baza danych (analityczna, wykorzystująca dane z wielu operacyjnych baz danych) służąca analitykom do wspomagania podejmowania decyzji, np. w zakresie planowania sprzedaży ratalnej, udzielania kredytów. Jeśli istnieje potrzeba szybkiego analizowania, raportowania i podejmowania decyzji na podstawie danych pochodzących z różnych baz operacyjnych, np. ze wszystkich uczelni w kraju, gdzie każda uczelnia ma inną bazę danych, to można utworzyć hurtownię danych, co jest jednak zadaniem trudnym, czasochłonnym i kosztownym. Głównym problemem jest uzgodnienie schematów baz danych, sposób transformacji danych do jednego formatu (przez wykorzystanie metabazy, czyli słownika danych) oraz bardzo duża ilość przechowywanych i wielowymiarowo analizowanych danych. Przetwarzanie danych w hurtowniach danych wymaga ogromnych zasobów obliczeniowych.

Problem dobrze ujęty, to problem w połowie rozwiązany.
Charles Franklin Kettering

3. MODEL KONCEPTUALNY

Projektowanie relacyjnej bazy danych ma na celu opracowanie poprawnego i spełniającego wymagania użytkowników logicznego schematu bazy danych, czyli przedstawienie bazy w postaci odpowiedniego zbioru schematów relacji będących co najmniej w trzeciej postaci normalnej.

Projektowanie bazy danych można podzielić na projektowanie konceptualne, logiczne i fizyczne.

W procesie projektowania relacyjnej bazy danych wyróżnia się opracowanie:

- Modelu konceptualnego (ang. *Conceptual Data Model* – CDM),
- Modelu logicznego (ang. *Logical Data Model* – LDM).

Implementacja bazy danych związana jest z opracowaniem

- Modelu fizycznego (ang. *Physical Data Model* – PDM).

Model danych (ang. *database model*) pozwala za pomocą ustalonego zestawu pojęć (metajęzyka) opisać określony wycinek świata rzeczywistego.

W modelu danych wyróżnia się opis:

- struktury danych, w tym więzów integralności danych,
- operacji wykonywanych na danych.

Celem projektowania konceptualnego (pojęciowego) jest opracowanie modelu konceptualnego niezależnego od szczegółów implementacyjnych, przedstawionego w ustalonej notacji graficznej zapewniającej jednoznaczność i czytelność.

W niniejszym opracowaniu skupimy się na projektowaniu **operacyjnych relacyjnych baz danych.**

Projekt bazy danych można wykonać:

- przeprowadzając normalizację uniwersalnego schematu relacyjnego bazy danych w postaci **R**(U, F), gdzie U – zbiór wszystkich atrybutów, F – zbiór wszystkich zależności funkcyjnych,
- poprzez zastosowanie zaproponowanej w tej książce metodyki strukturalnej obejmującej opracowanie modelu konceptualnego i relacyjnego modelu logicznego, co umożliwi wykonanie modelu fizycznego, a następnie implementację bazy danych w wybranym narzędziu.

Projektowanie bazy danych w metodyce strukturalnej obejmuje wykonanie 13 określonych etapów i udokumentowanie ich w wybranym edytorze tekstowym.

Etapy projektowania modelu konceptualnego:

Etap 1. Zdefiniowanie tematu, celu, zakresu i użytkowników bazy danych
Etap 2. Analiza rzeczywistości
Etap 3. Zdefiniowanie kategorii
Etap 4. Wyodrębnienie reguł funkcjonowania
Etap 5. Wyodrębnienie ograniczeń dziedzinowych
Etap 6. Zdefiniowanie transakcji
Etap 7. Identyfikacja typów encji i związków
Etap 8. Definicje predykatowe typów encji i związków
Etap 9. Diagram związków encji

Etapy projektowania modelu logicznego:

Etap 10. Transformacja modelu konceptualnego do modelu logicznego
Etap 11. Zdefiniowanie schematów relacji i podanie danych przykładowych
Etap 12. Opracowanie schematu bazy danych i słownika atrybutów
Etap 13. Zdefiniowanie perspektyw dla wszystkich użytkowników bazy

Aby projekt bazy danych był poprawny i spełniał oczekiwania przyszłych użytkowników, wszystkie etapy muszą być wykonane poprawnie i muszą być zgodne z sobą (spójne jako całość). Wymaga to weryfikowania poszczególnych etapów z innymi i nierzadko powrotu do samego początku projektu (Etap 1), w celu wyeliminowania zidentyfikowanych błędów.

Poszczególne etapy projektu dokumentowane są w ściśle określony sposób („Dodatek C"). Niektóre z nich są opracowywane w języku naturalnym, np. opis wycinka rzeczywistości (rozdz. 3.2.1). W innych etapach (np. rozdz. 3.3–3.6) wypowiedzi o faktach w świecie rzeczywistym formułowane są w sformalizowanym

podzbiorze języka naturalnego – umożliwia to wyrażanie wiedzy w sposób jednolity, zwięzły i uporządkowany. Z kolei model konceptualny (rozdz. 3.9) jest przedstawiany w formie diagramu. W wielu etapach wykorzystywane są tabele jako forma prezentacji opisów i danych przykładowych.

3.1. TEMAT, CEL, ZAKRES I UŻYTKOWNICY BAZY

Przystępując do projektowania bazy danych, należy precyzyjnie sformułować temat, cel i zakres projektowanej bazy, określić jej właściciela i użytkowników, ich wymagania, potrzeby i ograniczenia. Zbyt ogólne i pobieżne zdefiniowanie tych elementów może doprowadzić do wykonania projektu bazy nie spełniającej oczekiwania użytkowników. Sformułowanie „projektujemy bazę dla szkoły" nie odzwierciedla ani celu, ani zakresu. W projektowanej bazie danych w zależności od jej potencjalnych użytkowników, wymagań i oczekiwań może wystąpić potrzeba gromadzenia różnych danych i wykonywania różnych operacji. Na przykład w bazie dotyczącej szkoły mogą być przetwarzane tylko dane pracowników („dział kadr pracowników szkoły") bądź tylko dane uczniów („księga uczniów"), bądź też dane zarówno pracowników, jak i uczniów („kompleksowa obsługa szkoły"), a także np. dane dotyczące remontów szkoły.

Pierwszym i zasadniczym etapem rozpoczynającym opracowanie projektu bazy danych jest więc zwięzłe i czytelne:

- sformułowanie tematu (tytułu) przedsięwzięcia,
- określenie celu, czyli ustalenie, po co projektujemy bazę, w jakim celu, jakie korzyści dzięki zaprojektowanej bazie danych chcemy osiągnąć,
- zdefiniowanie zakresu, czyli określenie granic projektu (danych i operacji na danych),
- zidentyfikowanie wszystkich potencjalnych użytkowników bazy i ich uprawnień, a w szczególności, czy korzystanie z bazy będzie wymagało ich identyfikacji (zalogowania).

Cel powinien określać korzyści możliwe do osiągnięcia dzięki zaprojektowanej bazie, np. szybsze wprowadzanie i wyszukiwanie danych, wygodne sporządzanie statystyk i generowanie raportów parametrycznych, przyspieszenie obsługi klienta.

Cel powinien być mierzalny i weryfikowalny. Często określany jest za pomocą sformułowań: baza powinna *umożliwiać...*, *ułatwiać...*, *przyspieszać...* itp.

Przykładem błędnie sformułowanego celu dla bazy do oceny jakości usług sklepu internetowego jest: *celem jest poprawa obsługi klientów.* Tak sformułowany cel jest nieweryfikowalny i jego osiągnięcie (nieosiągnięcie) nie wynika z projektu bazy. Poprawnie sformułowanym celem jest np. *umożliwienie klientom komentowania i oceny usług sklepu przez przeglądarkę internetową.*

W etapie 1 projektu definiujemy:

- temat,
- cel,
- zakres,
- użytkowników.

Przykładowy opis etapu 1 może mieć następującą postać:

Etap 1. Temat, cel, zakres i użytkownicy bazy danych

1.1. Temat

Katastrofy budowlane

1.2. Cel

Celem przedsięwzięcia jest umożliwienie pracownikom Ministerstwa Infrastruktury gromadzenia i przetwarzania danych (m.in. szybkie wielokryterialne wyszukiwanie danych oraz sporządzanie raportów i statystyk) o katastrofach budowlanych.

1.3. Zakres

Baza danych powinna umożliwiać gromadzenie, wyszukiwanie, sortowanie i czytelne prezentowanie danych o katastrofach budowlanych w Polsce oraz danych pracowników upoważnionych do obsługi bazy.

1.4. Użytkownicy

- **Pracownik** – po zalogowaniu ma możliwość wprowadzania, edycji i usuwania danych,
- **Internauta** – przeglądanie danych katastrof (nie wymaga logowania).

Dobrze wykonany etap 1 umożliwia przystąpienie do etapu 2 – analizy rzeczywistości.

3.2. ANALIZA RZECZYWISTOŚCI

Dla danej rzeczywistości (organizacji, firmy, społeczności, przedsiębiorstwa) chcemy zaprojektować bazę danych, czyli opracować model pewnych aspektów rozważanej rzeczywistości. Ten wyodrębniony wycinek rzeczywistości nazywamy **obszarem analizy** – OA (ang. *universe of discourse*) lub dziedziną analizy (miniświatem bazy danych). Proces analizowania potrzeb przedsiębiorstwa i zbierania od użytkowników wymagań wobec projektowanej bazy danych nazywamy **analizą wymagań** (ang. *requirements analysis*).

Prawidłowo zaprojektowana baza danych nie powinna zawierać żadnych zbędnych danych, powinna natomiast umożliwiać efektywną pracę zaimplementowanego systemu oraz umożliwiać jego rozbudowę.

Celem etapu analizy wycinka rzeczywistości jest:

a) zweryfikowanie celów przedsięwzięcia i ich związków z projektowaną bazą danych,
b) przeprowadzenie wnikliwej analizy wycinka rzeczywistości w odniesieniu do sformułowanego tematu, na przykład poprzez przeprowadzenie z ekspertem dziedzinowym szczegółowego wywiadu, na podstawie którego będzie można precyzyjnie opisać określony wycinek rzeczywistości oraz zaplanować przyszłe funkcje systemu dla zidentyfikowanych użytkowników systemu,
c) ocena stanu ewentualnie istniejącej bazy danych,
d) oszacowanie kosztów związanych z budową nowej bazy danych i ewentualnie nowego systemu zarządzania tą bazą.

Aby prawidłowo sformułować cel przedsięwzięcia, należy określić przyszłych użytkowników projektowanej bazy danych. **Użytkownikami** bazy danych mogą być zarówno ludzie (projektanci, administratorzy, operatorzy), jak i programy (systemy komputerowe, aplikacje) lub urządzenia peryferyjne. Użytkowników bazy danych można pogrupować według przypisanych im ról, przy czym jedna osoba może mieć przypisanych wiele ról (należeć do wielu grup użytkowników), a jedną rolę można przypisać jednej lub wielu osobom. Poszczególne role zwykle różnią się funkcjami wykonywanymi na bazie danych oraz przydzielonymi uprawnieniami. Szczególną rolę pełni zazwyczaj administrator bazy danych. Dla bazy danych **WYDZIAŁ** można określić kilka grup użytkowników o różnych uprawnieniach, np.: dziekan, nauczyciel, pracownik dziekanatu, administrator, student, kandydat. Korzystanie z bazy danych wiąże się wówczas z identyfikacją użytkownika, np.

podczas logowania, ale mogą też być grupy użytkowników, dla których nie jest to konieczne, np. kandydat nie musi się logować i ma uprawnienia tylko do przeglądania określonych danych. Nazwy grup użytkowników zazwyczaj są rzeczownikami w liczbie pojedynczej.

W modelowanym OA należy wyodrębnić dwa aspekty:

- statyczny, czyli dane,
- dynamiczny, czyli operacje na danych,

oraz reguły funkcjonowania (reguły zachowania spójności, tzw. więzy integralności).

W fazie analizy wyodrębniamy więc:

- obiekty (kategorie, encje), czyli rzeczy istotne (i nadajemy im nazwy, które są rzeczownikami),
- związki zachodzące pomiędzy obiektami (i nadajemy im nazwy, które są czasownikami).
- operacje (transakcje) wykonywane na danych,
- reguły funkcjonowania i ograniczenia dziedzinowe.

W obiektach wyróżniamy cechy dla nich charakterystyczne, czyli atrybuty.

W etapie 2 – **Analiza rzeczywistości**, należy uwzględnić następujące podpunkty:

2.1. Szczegółowy opis wycinka rzeczywistości
2.2. Słownik pojęć
2.3. Użytkownicy i zakres uprawnień
2.4. Wymagania funkcjonalne
2.5. Wymagania niefunkcjonalne
2.6. Analiza istniejących baz danych
2.7. Analiza kosztów

W podrozdziałach 3.2.1–3.2.7 opisano istotne zagadnienia etapu 2.

3.2.1. SZCZEGÓŁOWY OPIS WYCINKA RZECZYWISTOŚCI

Wykonanie projektu bazy danych wymaga przeprowadzenia szczegółowej **analizy wycinka rzeczywistości**, który chcemy modelować w bazie danych [Górski2000, Hernan1998]. W analizowanym wycinku rzeczywistości możemy wyróżnić rzeczywistość fizyczną (rzeczywiście istniejącą, np. książka, bilet, faktura, towar, arkusz ocen) i rzeczywistość abstrakcyjną, czyli konceptualną, pojęciową, istniejącą jedynie jako pojęcia w wyobraźni ludzkiej (np. wypożyczenie, lot, kupno).

W celu rzetelnego przeprowadzenia analizy wycinka rzeczywistości należy przeprowadzić – jeżeli to możliwe – wnikliwy wywiad z ekspertem dziedzinowym i określić założenia wstępne dotyczące przechowywanych danych.

Dobrze przeprowadzony wywiad pozwoli wyodrębnić:

a) grupy użytkowników bazy danych (systemu bazy danych),
b) oczekiwania przyszłych użytkowników bazy,
c) reguły funkcjonowania i ograniczenia dziedzinowe,
d) wymagania funkcyjne – należy uwzględnić operacje, jakie na danych użytkownicy będą chcieli wykonywać oraz w jaki sposób będą chcieli prezentować dane i uzyskane informacje,
e) rzeczy istotne, czyli obiekty, które trzeba uwzględnić w bazie danych,
f) atrybuty obiektów i ich dziedziny, czyli zbiory wartości,
g) powiązania między obiektami.

Pomyślność przeprowadzenia poprawnej analizy zależy zarówno od eksperta, który udziela odpowiedzi na zadawane pytania, od jego wiedzy, jak również od osoby przeprowadzającej wywiad. Często tworzy się zespoły składające się ze specjalistów z danej dziedziny, kierownika działu, informatyków, przedstawiciela inwestora (sponsora przedsięwzięcia) itd., aby wszechstronnie przeanalizować zagadnienie i wyznaczyć jego zakres.

Zaleca się zapoznanie ze wszystkimi dostępnymi materiałami i dokumentami związanymi z analizowaną rzeczywistością, np.:

- przepisami określającymi funkcjonowanie danego przedsiębiorstwa,
- istotnymi ustawami i rozporządzeniami,
- obowiązującą polityką bezpieczeństwa,
- drukami i formularzami papierowymi używanymi dotychczas (np. fakturami, zamówieniami),
- postaciami raportów i zestawień itp.

Należy określić również, czy postać dotychczasowych druków jest odpowiednia i ma być zachowana, czy raczej jest przestarzała i należy ją zmodyfikować.

W trakcie wywiadu należy przeanalizować zarówno sytuacje typowe (czynności zwykle wykonywane, np. przy rejestrowaniu pacjenta czy wystawianiu faktury VAT), jak również sytuacje wyjątkowe (np. co dzieje się z kartą pacjenta po jego śmierci). Bardzo istotne są informacje na temat postępowania w przypadkach, gdy dane są niekompletne (czy dane niekompletne się przechowuje, odrzuca, uzupełnia itp.).

Projektant bazy danych powinien wiedzieć wszystko na temat pozyskiwania danych – skąd się je pozyskuje, w jaki sposób, jaką mają postać, jak się postępuje,

gdy są w innej postaci niż wymagana lub są niekompletne albo nieczytelne, czy dopuszcza się brak danych itp.

Ze względu na złożoność świata rzeczywistego w wielu projektach ogranicza się modelowaną rzeczywistość do pewnego wycinka, wprowadza się niejednokrotnie pewne uproszczenia i ograniczenia, lecz istotna wiedza dotycząca faktów i zdarzeń musi być zachowana i opisana.

Opracowując model świata rzeczywistego, czyli projektując bazę danych, musimy jednak uważać, by przyjęte uproszczenia nie zmieniły zasad funkcjonowania rzeczywistości lub nie ograniczyły jej w sposób niedozwolony.

Pomimo ograniczenia analizy do pewnego wycinka pełny jego opis może być bardzo obszerny (nawet kilkaset stron), a przygotowanie go bardzo czasochłonne.

Wyczerpująco i poprawnie przeprowadzona analiza wycinka rzeczywistości, przedstawiona w postaci opisu w języku naturalnym, jest punktem wyjścia do prawidłowego zaprojektowania bazy danych. Ze względu na brak umiejętności w przeprowadzaniu wywiadu bądź brak kompetencji osoby dostarczającej wiedzy o analizowanej rzeczywistości może powstać zły projekt, którego wady mogą zostać zauważone dopiero w trakcie eksploatacji bazy. Usunięcie wad systemu informatycznego wykorzystującego źle zaprojektowaną bazę danych w trakcie jego eksploatacji jest bardzo kosztowne, a często (z wielu względów) wręcz niemożliwe.

Zrozumienie wszystkich faktów i zapoznanie się z wiedzą dziedzinową zawartą w opisie napotyka często problemy związane z terminologią specjalistyczną pojęć dziedzinowych, np. projektanci mogą nie znać terminologii z księgowości, a zlecono im zaprojektowanie bazy danych dla systemu finansowo-księgowego.

W ramach prac projektowych należy zatem przygotować słownik podstawowych pojęć używanych do przedstawienia wiedzy przez specjalistów dziedzinowych.

3.2.2. SŁOWNIK POJĘĆ

Podczas wywiadów i analizy wycinka rzeczywistości używane są pojęcia związane z danym obszarem analizy, których znaczenie nie zawsze jest oczywiste i prawidłowo interpretowane. Pojęcia istotne najczęściej występujące w sporządzanych opisach faktów należy bardzo precyzyjnie zdefiniować, aby ich znaczenie było ściśle ustalone i jednoznaczne. Bardzo często terminy, nawet powszechnie używane, mogą budzić rozmaite skojarzenia i osoby posługujące się nimi mogą mieć na myśli zupełnie inne desygnaty.

W trakcie wywiadu na temat funkcjonowania hotelu jeden z pracowników bardzo często używał słowa „gość". Po dłuższej rozmowie okazało się, że dla niego „gość" nie oznacza „gościa hotelowego", tylko jest to każda (dowolna) osoba, o której w danej chwili mówi (kierownik hotelu, recepcjonistka, klient hotelu itd.). Jeżeli przyjmiemy, że *gość* to osoba przebywająca (zameldowana) w hotelu, to sformułowanie, że w bazie mają być przechowywane dane o gościach hotelu, oznaczałoby, że nie przechowuje się danych osób rezerwujących pobyt ani osób już wymeldowanych. Zatem w bazie będą dane tylko osób aktualnie zameldowanych w hotelu, a wraz z wymeldowaniem te dane będą usuwane z bazy. Jeśli jednak należy przechowywać historię zameldowań, to pojęcie *gość* musi oznaczać osobę, która jest lub była klientem hotelu. Czy to znaczy, że dane klientów hotelu nigdy nie będą usuwane z bazy, czy też będą przechowywane tylko przez określony czas, np. przez 5 lat, a potem będzie można je usunąć? Tak więc znowu należy doprecyzować, że *gość* to osoba, która korzysta lub korzystała z usług hotelu co najmniej raz w przeciągu ostatnich 5 lat. Jeśli jednak dopuszczamy rezerwację pokoju i możliwość rezygnacji z rezerwacji, to czy dane osób rezerwujących też będą przechowywane? Jak długo?

Z powyższych rozważań wynika, że nawet tak mały, prosty wycinek rzeczywistości wymaga dokładnego przeanalizowania, precyzyjnego zdefiniowania pojęć i podjęcia decyzji odnośnie odwzorowania tego wycinka w projekcie.

Istotne pojęcia należy zdefiniować, porządkując je alfabetycznie, np.:

Klient – osoba korzystająca z usług biura podróży – składająca zapytanie ofertowe lub zawierająca umowę z biurem (reprezentowanym przez pracownika) na zakup wybranej usługi z aktualnej oferty.

Oferta – pełny zestaw usług oferowanych przez biuro podróży, takich jak: usługi noclegowe, transportowe, żywieniowe, ubezpieczeniowe, rozrywkowe.

Umowa – dokument potwierdzający sprzedaż usługi klientowi przez pracownika biura podróży, zawierający dane osób objętych umową.

3.2.3. UŻYTKOWNICY I ZAKRES UPRAWNIEŃ

Z projektowanej bazy danych być może będą korzystały różne grupy użytkowników. Wstępnie zostali oni wyszczególnieni w etapie 1. Podczas analizy rzeczywistości należy zweryfikować, czy grupy użytkowników, które zostały wymienione w etapie 1, wyczerpują zbiór wszystkich potencjalnych użytkowników projektowanej bazy,

oraz jakie są ich oczekiwania i uprawnienia. Należy zadecydować, czy ich dane mają być przechowywane w bazie, czy korzystanie z bazy powinno być poprzedzone logowaniem poprzez podanie loginu i hasła dostępu do bazy.

Do danej grupy użytkowników może należeć wiele osób. Wszystkie osoby należące do danej grupy użytkowników mają te same uprawnienia. Dana osoba może należeć do wielu grup użytkowników. Nazwa grupy użytkowników jest rzeczownikiem w liczbie pojedynczej, np. Pracownik, Klient, Internauta, Obserwator.

Identyfikacja użytkowników bazy danych jest podstawą uwierzytelnienia (ang. *authentication*), autoryzacji (ang. *authorization*) i rozliczalności (ang. *accounting*), w skrócie nazywanych procesami AAA.

Uwierzytelnianie to jednoznaczna weryfikacja tożsamości danego użytkownika. Z konieczności uwierzytelniania użytkownika wynika konieczność przechowywania w bazie danych odpowiednich danych do uwierzytelniania. Sposoby uwierzytelniania mogą być różne. Najczęściej jest to login i hasło ustalone przez użytkownika lub system, przy czym jako login podawany jest np. adres e-mail użytkownika lub dowolna unikalna nazwa. W internetowych systemach bankowych do uwierzytelniania wykorzystuje się login (jako ciąg cyfr oznaczających numer klienta) i jednorazowe hasło SMS-owe. Korzystanie ze smartfona może wymagać podania PIN-u lub odczytania danych biometrycznych (linii papilarnych lub zdjęcia twarzy). Z kolei w internetowym systemie e-pit, aby uzyskać dostęp do rozliczenia podatkowego za rok 2019, należy podać dane logowania:

- PESEL (albo NIP i datę urodzenia),
- kwotę przychodu z rozliczenia za poprzedni rok (2018),
- kwotę przychodu z jednej informacji od płatników za 2019 rok (np. z PIT-11 od pracodawcy).

Z przyjętego sposobu uwierzytelniania wynika konieczność odpowiedniego zdefiniowania atrybutów i ich dziedzin oraz uwzględnienia ich w definiowanych kategoriach (rozdz. 3.3), a także zdefiniowania odpowiednich wymagań funkcjonalnych (rozdz. 3.2.4) i transakcji, np. logowania, wylogowania, przypomnienia hasła (rozdz. 3.6).

Autoryzacja wiąże się z nadawaniem uprawnień użytkownikom do dostępu do określonych danych i wykonywania na nich określonych operacji oraz z weryfikowaniem tych uprawnień. Autoryzacja wymaga wcześniejszego uwierzytelnienia. Uprawnienia nadane użytkownikom są odzwierciedlane m.in. w wymaganiach funkcjonalnych, w regułach funkcjonowania, w transakcjach i w perspektywach.

Rozliczalność wiąże się z kontrolą wykonywanych operacji na danych. W tym celu w bazie danych muszą być gromadzone odpowiednie dane umożliwiające weryfikowanie wykonanych działań. Rozliczalność użytkowników wymaga trwałego przechowywania odpowiednich danych przez określony czas.

Grupy użytkowników określonych w etapie 1 i w etapie 2.3 projektu muszą być takie same, np.

- **Pracownik** – po zalogowaniu ma możliwość wprowadzania, edycji i usuwania danych katastrof, generowania raportów.
- **Internauta** – może przeglądać dane katastrof (nie wymaga logowania).

Zauważmy, że jeśli są co najmniej dwie grupy użytkowników, to żadna z nich nie powinna mieć nazwy „Użytkownik", gdyż prowadziłoby to do niejednoznaczności.

3.2.4. ANALIZA WYMAGAŃ FUNKCJONALNYCH

Baza danych musi być zaprojektowana w taki sposób, by możliwe było wykonywanie określonych funkcji (operacji realizujących procesy biznesowe) przez jej użytkowników. Operacje te należy zidentyfikować, wyszczególnić i precyzyjnie opisać (rozdz. 3.6). W tym punkcie projektu trzeba wypisać listę funkcji, jakie dla zgromadzonych danych powinny być realizowane, np.

- wprowadzanie, modyfikowanie, usuwanie i sortowanie danych,
- wyszukiwanie danych,
- sporządzanie statystyk,
- generowanie raportów,
- wysyłanie e-maili.

Dla różnych grup użytkowników wymagania funkcjonalne (ang. *functional requirements*) mogą być różne, dlatego ich wykaz należy sporządzić z podziałem na wyodrębnione w etapie 3.2.3 grupy użytkowników, np.:

Kierownik

- logowanie,
- wylogowanie,
- dodawanie pozycji w menu,
- edytowanie pozycji w menu,

- usuwanie pozycji z menu,
- dodawanie danych pracowników,
- edytowanie danych pracowników,
- zmiana swojego hasła.

Kelner

- logowanie,
- wylogowanie,
- tworzenie zamówienia,
- dodawanie pozycji do zamówienia,
- edycja zamówienia,
- rozliczanie zamówienia,
- anulowanie zamówienia,
- zmiana swojego hasła.

Oprócz wymagań funkcjonalnych w projekcie należy także wyszczególnić wymagania niefunkcjonalne.

3.2.5. ANALIZA WYMAGAŃ NIEFUNKCJONALNYCH

Wymagania niefunkcjonalne (ang. *non-functional requirements*) opisują ograniczenia (parametry), przy zachowaniu których możliwa jest realizacja wymagań funkcjonalnych. Wymagania niefunkcjonalne dotyczą często aspektu jakościowego, np. odnośnie do liczby obsługiwanych klientów, czasu działania, niezawodności, bezpieczeństwa, skalowalności, użyteczności (ergonomii systemu).

Jeśli chodzi o wymagania niefunkcjonalne, często trudno jest je sprecyzować i zweryfikować, czy zostały spełnione, jeśli nie są one mierzalne, np. baza powinna działać wydajnie, obsługa klienta powinna być wygodna i szybka, dokumentacja projektu nie powinna być obszerna.

Na etapie analizy wymagań niefunkcjonalnych opisywane są wszystkie istotne aspekty związane z opracowywaną bazą, które nie zostały zdefiniowane w wymaganiach funkcjonalnych. Na ogół dotyczą one bazy jako całości, a nie poszczególnych funkcji czy użytkowników.

Często, chociaż nie zawsze, wymagania niefunkcjonalne formułuje się nie tylko w odniesieniu do projektowanej bazy danych, ale także do ewentualnych aplikacji korzystających z tej bazy, czyli do całego systemu bazy danych obejmującego

zaprojektowaną bazę danych (rysunek 1.1). Określa się na przykład możliwości i oczekiwania użytkownika przyszłego systemu dotyczące platformy sprzętowej, sposobu pracy z systemem (klawiatura, mysz), trybu pracy (tryb tekstowy, graficzny), użycia dźwięków, sposobu dostępu (wersja sieciowa lub jednostanowiskowa). Wśród tych wymagań określa się:

- platformę sprzętową (np. komputer stacjonarny, urządzenie mobilne, drukarka kolorowa),
- platformę systemową (np. Windows),
- środowisko implementacyjne bazy danych (np. Oracle),
- środowisko programistyczne,
- rodzaj bazy danych (np. relacyjna),
- oszacowanie liczby danych wejściowych, tempo przyrostu danych,
- sposób archiwizowania danych.

Mimo że opracowywany system bazy danych może być przeznaczony tylko do jednostanowiskowej pracy (np. na komputerze właściciela małej firmy będzie zainstalowany system finansowo-księgowy do pełnej obsługi jego firmy i nie będzie wymagane logowanie do systemu), to często wiele aplikacji musi współpracować z istniejącym już systemem i z innymi programami. Dokładna analiza środowiska jest wówczas niezbędną czynnością, aby nowy produkt mógł dobrze działać i współpracować z pozostałym oprogramowaniem.

Wymagania niefunkcjonalne mogą dotyczyć:

- bezpieczeństwa danych:
  - sposobu uwierzytelniania,
  - szyfrowania danych (w tym haseł),
  - kont użytkowników wymagających uwierzytelniania,
  - historii logowania,
- ciągłości działania bazy danych i czasu odtwarzania po awarii,
- usuwania danych (według określonych kryteriów lub na żądanie),
- sposobu usuwania danych (ograniczone / kaskadowe),
- sposobu edycji danych (ograniczona / kaskadowa),
- oprogramowania, np. baza danych w MS Access (firma ma odpowiednią licencję i inne bazy w MS Access),
- generowania raportów, np. parametryzowanych,
- sprzętu, np. drukarka monochromatyczna,
- sposobu reprezentacji danych, np. dat, pensji,
- wykorzystywanych jednostek monetarnych,

- wersji językowych, np. do prezentacji danych i komunikatów dla użytkowników posługujących się różnymi językami,
- wymaganego języka zapisu danych w bazie, np. w języku polskim,
- dostępu do danych przez dowolną przeglądarkę internetową,
- wymagań wynikających, np. z rozporządzeń wewnętrznych lub branżowych.

3.2.6. ANALIZA ISTNIEJĄCYCH BAZ DANYCH

W tym etapie należy przeprowadzić analizę być może istniejących już baz danych związanych z rozważanym obszarem rzeczywistości. Należy wówczas zapoznać się ze sposobem gromadzenia danych, ich zakresem i wykorzystywaniem oraz z ewentualnym funkcjonującym systemem informatycznym obsługującym dotychczasową bazę danych (jeśli taki istnieje). Należy przeanalizować zalety i wady istniejącej bazy i systemu informatycznego, a także wszelkie uwagi pozytywne i krytyczne użytkowników. Rzetelna ocena dotychczasowej bazy i systemu komputerowego (jakości pracy, funkcjonalności, szybkości itp.) powinna pozwolić na właściwą ocenę, czy rzeczywiście istnieje potrzeba opracowania nowego projektu bazy danych, czy też niezadowolenie użytkownika wynika na przykład z nieznajomości zasad obsługi posiadanego oprogramowania lub braku dostępu do pewnych niezbędnych danych czy raportów.

Bezkrytyczne wzorowanie się na istniejącej analizie rzeczywistości, bazie danych, formularzach i raportach, może doprowadzić do niewłaściwego zaprojektowania bazy danych i powielenia poprzednich błędów.

Istniejące bazy danych powinny raczej zostać wykorzystane do weryfikacji, czy zawierają elementy, które nie zostały uwzględnione w nowej bazie danych, a są istotne.

3.2.7. ANALIZA KOSZTÓW

Zaprojektowanie i implementacja bazy danych oraz systemu obsługującego tę bazę wymaga oczywiście czasu i nakładów finansowych. Jeśli analiza kosztów ma obejmować tylko koszty wykonania projektu bazy danych i jego udokumentowania, to należy wyszczególnić liczbę osób zespołu projektowego, stawki godzinowe, liczbę godzin pracy i podać ostateczny koszt zaprojektowania bazy danych.

W pełnej analizie kosztów wykonania projektu systemu bazy danych oprócz kosztów zaprojektowania bazy danych trzeba jeszcze uwzględnić koszty związane z wytworzeniem oprogramowania, zakupem licencji, wdrożeniem i eksploatacją systemu, szkoleniem pracowników, zakupem lub modernizacją sprzętu komputerowego, ze zorganizowaniem miejsc pracy użytkownikom systemu.

Po oszacowaniu kosztów realizacji nowej bazy (systemu) należy przeanalizować dotychczasowe koszty związane z wykonywaniem prac (np. liczba pracowników, ich wynagrodzenie itp.) oraz uwzględnić skuteczność, szybkość i wydajność pracy, poprawność osiąganych wyników itd. i porównać je ze wszystkimi kosztami, jakie będą związane z wykonaniem i eksploatacją nowego systemu bazodanowego, oraz z korzyściami, jakie przyniesie wykorzystywanie nowego systemu, takimi jak np. zwiększenie konkurencyjności i rozpoznawalności firmy, mniejsza liczba błędnych danych.

Po precyzyjnej i wyczerpującej analizie wycinka rzeczywistości, która powinna być spisana w języku naturalnym z uwzględnieniem wszelkich szczegółów rozpatrywanego wycinka i przedstawiona przyszłym użytkownikom do zatwierdzenia, mając określone założenia i wymagania użytkowników, można przejść do etapu trzeciego, czyli do określenia kategorii.

3.3. DEFINICJE KATEGORII

Ze szczegółowej analizy rzeczywistości oraz na podstawie słownika pojęć należy wyodrębnić rzeczy istotne, których dane chcemy przechowywać, np. towar, uczeń, kaseta, lek, faktura, wypożyczenie itp., zwane **kategoriami** (obiektami, pakietami danych). Kategoria jest reprezentacją istotnego (z punktu widzenia projektu) faktu lub zdarzenia.

Dla każdej kategorii wyodrębnia się atrybuty, czyli ich cechy charakterystyczne.

Definicja kategorii:
KAT/*xxx* Nazwa kategorii
Opis: Semantyka kategorii i związek z innymi kategoriami
Atrybuty: Wykaz atrybutów, ich opisy i dane przykładowe
gdzie *xxx* jest kolejnym trzycyfrowym numerem kategorii.

Każdej kategorii nadaje się jednoznaczny identyfikator postaci KAT/*xxx* i nazwę.

Nazwy kategorii i atrybutów powinny być krótkie, najlepiej jednowyrazowe (jednoczłonowe), semantycznie związane z danymi, których dotyczą. Nazwy kategorii powinny być rzeczownikami w liczbie pojedynczej i muszą być unikalne (nie mogą się powtarzać). Nazwy atrybutów w jednej kategorii muszą także być unikalne, ale takie same nazwy mogą być nadane atrybutom w różnych kategoriach. Zaleca się nadawanie nazw bez użycia liter charakterystycznych dla języka polskiego (ze znakami diakrytycznymi), oczywiście nie dotyczy to danych przechowywanych w bazie czy komunikatów – te aspekty muszą wynikać z analizy rzeczywistości. W celu podniesienia poziomu czytelności omawianego materiału w zamieszczanych przykładach używane są jednak nazwy zawierające polskie litery.

Nie należy nadawać nazw takich jak słowa kluczowe lub nazwy systemowe używane w bazach danych, np. zamiast nazw: *Nazwa*, *Klasa*, *Wiersz*, *Kolumna*, *Data*, lepiej nadać nazwę: *NazwaP*, *KlasaU*, *WierszX*, *KolumnaK*, *DataUr* stosownie do okoliczności użycia.

W każdej kategorii powinien być atrybut jednoznacznie identyfikujący każdą daną tej kategorii i zaleca się, by nazwy atrybutów identyfikujących były jednoznaczne w całej bazie danych.

Jeśli wśród wyodrębnionych atrybutów nie ma atrybutu jednoznacznie identyfikującego dane, to należy sztucznie wprowadzić taki atrybut, np. *Symbol*, *Nr*, *Id*, *IdP*.

Atrybuty chronione i takie, co do których nie ma pewności, że są unikalne, nie powinny być używane do identyfikowania danych, np. PESEL czy login.

Jeśli atrybut e-mail ma także pełnić rolę loginu użytkownika, to taka informacja musi być zawarta w opisie atrybutu.

Dla każdego atrybutu należy podać co najmniej jedną wartość przykładową. Trzeba dobrze przeanalizować możliwe wartości dla danego atrybutu w celu poprawnego określenia typów i rozmiarów pól (w następnych etapach projektu), np. nazwisko nie jest tylko ciągiem liter, ale oprócz liter może zawierać myślnik, apostrof, spację, może mieć różną długość, więc jako wartości przykładowe można podać: Abadżakanowska-Śliwakowska, Konstantynopolitańczykiewiczówna, Ak, Żyżyński, O'Neil, de Jong.

Ze zwięzłego opisu kategorii powinna wynikać jej semantyka (znaczenie) oraz powiązania z innymi kategoriami.

Analizując fakty wynikające z opisu rzeczywistości, należy rozważyć, czy powinny one być zdefiniowane jako atrybut kategorii, czy jako odrębna kategoria, np. imię i data urodzenia dziecka pracownika. Jeśli mogą być pracownicy bez dzieci,

ale inni mogą mieć wiele dzieci, to należy utworzyć dwie kategorie: **Pracownik** oraz **Dziecko**, a związek między nimi wyrazić w opisie kategorii **Dziecko**, np. kategoria zawiera dane dzieci pracowników.

Jeśli w bazie mają być przechowywane dane o dostawcach i dostarczanych przez nich towarach i z analizy rzeczywistości wynika, że dostawca może dostarczać wiele towarów, a towar może być dostarczany przez wielu dostawców, to kategoria **Towar** nie powinna mieć atrybutu *Dostawca*, tylko należy zdefiniować trzy odrębne kategorie: **Towar**, **Dostawca** i **Dostawa**, a w opisie powinny znaleźć się informacje o odpowiednich powiązaniach.

Bardzo często dane słownikowe, takie jak kraj czy miasto, definiuje się jako kategorię. Należy jednak dobrze przeanalizować opis rzeczywistości, sprawdzając, czy dane w bazie będą stałe i odpowiednie, a ponadto który użytkownik ma uprawnienia do wpisywania i modyfikowania danych. Jeśli dane klienta (w tym miasto zamieszkania) wprowadza pracownik firmy, to aby zapewnić poprawność danych i wyeliminować redundancje, należy utworzyć kategorię **Miasto**. Rozważmy jednak inny przypadek – jeśli projektowana jest baza w ramach internetowego systemu wypełnianiu wniosków o wizę, to miejsce urodzenia powinno być raczej atrybutem *Miejsce*, tak by każdy starający się o wizę mógł niezależnie wpisać odpowiednią nazwę.

Przykład 3.1 (opisy kategorii)

KAT/001 Towar
Opis: Kategoria **Towar** służy do opisu towaru.
Atrybuty:
IdT – identyfikator towaru, np. 1, 2, 3
NazwaT – nazwa towaru, np. cukier biały
SymbolT – unikalny symbol towaru, np. cuk001/99
Jednostka – jednostka miary towaru, np. kg, szt., km
Uwaga – dowolna uwaga, np. towar niepełnowartościowy

KAT/002 Dostawca
Opis: Kategoria **Dostawca** opisuje dane firm, które mogą dostarczać towary.
Atrybuty:
IdD – identyfikator dostawcy, np. 1
NazwaSkD – skrócona nazwa dostawcy, np. MEGA-HIT

Każdej kategorii nadaje się jednoznaczny identyfikator postaci KAT/*xxx* i nazwę.

Nazwy kategorii i atrybutów powinny być krótkie, najlepiej jednowyrazowe (jednoczłonowe), semantycznie związane z danymi, których dotyczą. Nazwy kategorii powinny być rzeczownikami w liczbie pojedynczej i muszą być unikalne (nie mogą się powtarzać). Nazwy atrybutów w jednej kategorii muszą także być unikalne, ale takie same nazwy mogą być nadane atrybutom w różnych kategoriach. Zaleca się nadawanie nazw bez użycia liter charakterystycznych dla języka polskiego (ze znakami diakrytycznymi), oczywiście nie dotyczy to danych przechowywanych w bazie czy komunikatów – te aspekty muszą wynikać z analizy rzeczywistości. W celu podniesienia poziomu czytelności omawianego materiału w zamieszczanych przykładach używane są jednak nazwy zawierające polskie litery.

Nie należy nadawać nazw takich jak słowa kluczowe lub nazwy systemowe używane w bazach danych, np. zamiast nazw: *Nazwa*, *Klasa*, *Wiersz*, *Kolumna*, *Data*, lepiej nadać nazwę: *NazwaP*, *KlasaU*, *WierszX*, *KolumnaK*, *DataUr* stosownie do okoliczności użycia.

W każdej kategorii powinien być atrybut jednoznacznie identyfikujący każdą daną tej kategorii i zaleca się, by nazwy atrybutów identyfikujących były jednoznaczne w całej bazie danych.

Jeśli wśród wyodrębnionych atrybutów nie ma atrybutu jednoznacznie identyfikującego dane, to należy sztucznie wprowadzić taki atrybut, np. *Symbol*, *Nr*, *Id*, *IdP*.

Atrybuty chronione i takie, co do których nie ma pewności, że są unikalne, nie powinny być używane do identyfikowania danych, np. PESEL czy login.

Jeśli atrybut e-mail ma także pełnić rolę loginu użytkownika, to taka informacja musi być zawarta w opisie atrybutu.

Dla każdego atrybutu należy podać co najmniej jedną wartość przykładową. Trzeba dobrze przeanalizować możliwe wartości dla danego atrybutu w celu poprawnego określenia typów i rozmiarów pól (w następnych etapach projektu), np. nazwisko nie jest tylko ciągiem liter, ale oprócz liter może zawierać myślnik, apostrof, spację, może mieć różną długość, więc jako wartości przykładowe można podać: Abadżakanowska-Śliwakowska, Konstantynopolitańczykiewiczówna, Ak, Żyżyński, O'Neil, de Jong.

Ze zwięzłego opisu kategorii powinna wynikać jej semantyka (znaczenie) oraz powiązania z innymi kategoriami.

Analizując fakty wynikające z opisu rzeczywistości, należy rozważyć, czy powinny one być zdefiniowane jako atrybut kategorii, czy jako odrębna kategoria, np. imię i data urodzenia dziecka pracownika. Jeśli mogą być pracownicy bez dzieci,

ale inni mogą mieć wiele dzieci, to należy utworzyć dwie kategorie: **Pracownik** oraz **Dziecko**, a związek między nimi wyrazić w opisie kategorii **Dziecko**, np. kategoria zawiera dane dzieci pracowników.

Jeśli w bazie mają być przechowywane dane o dostawcach i dostarczanych przez nich towarach i z analizy rzeczywistości wynika, że dostawca może dostarczać wiele towarów, a towar może być dostarczany przez wielu dostawców, to kategoria **Towar** nie powinna mieć atrybutu *Dostawca*, tylko należy zdefiniować trzy odrębne kategorie: **Towar**, **Dostawca** i **Dostawa**, a w opisie powinny znaleźć się informacje o odpowiednich powiązaniach.

Bardzo często dane słownikowe, takie jak kraj czy miasto, definiuje się jako kategorię. Należy jednak dobrze przeanalizować opis rzeczywistości, sprawdzając, czy dane w bazie będą stałe i odpowiednie, a ponadto który użytkownik ma uprawnienia do wpisywania i modyfikowania danych. Jeśli dane klienta (w tym miasto zamieszkania) wprowadza pracownik firmy, to aby zapewnić poprawność danych i wyeliminować redundancje, należy utworzyć kategorię **Miasto**. Rozważmy jednak inny przypadek – jeśli projektowana jest baza w ramach internetowego systemu wypełnianiu wniosków o wizę, to miejsce urodzenia powinno być raczej atrybutem *Miejsce*, tak by każdy starający się o wizę mógł niezależnie wpisać odpowiednią nazwę.

Przykład 3.1 (opisy kategorii)

KAT/001 Towar
Opis: Kategoria **Towar** służy do opisu towaru.
Atrybuty:

IdT	– identyfikator towaru, np. 1, 2, 3
NazwaT	– nazwa towaru, np. cukier biały
SymbolT	– unikalny symbol towaru, np. cuk001/99
Jednostka	– jednostka miary towaru, np. kg, szt., km
Uwaga	– dowolna uwaga, np. towar niepełnowartościowy

KAT/002 Dostawca
Opis: Kategoria **Dostawca** opisuje dane firm, które mogą dostarczać towary.
Atrybuty:

IdD	– identyfikator dostawcy, np. 1
NazwaSkD	– skrócona nazwa dostawcy, np. MEGA-HIT

NazwaD – pełna nazwa dostawcy, np.
Hurtownia artykułów spożywczych MEGA-HIT
AdresD – aktualny adres dostawcy (zarządu firmy), np.
11-123 Miasteczko, ul. Nowa 2
TelD – numer telefonu kontaktowego do dostawcy, np.
123 123 123 lub 1234 w. 23
NIP – dziesięciocyfrowy Numer Identyfikacji Podatkowej (NIP) firmy, np. 896-000-58-51
REGON – numer REGON firmy, 000001614

Należy zwrócić uwagę, by wśród kategorii i ich atrybutów były uwzględnione wszystkie dane potrzebne do realizowania funkcji wymaganych przez użytkowników, a w szczególności np. dane użytkowników, loginy, hasła, role itp.

Nie należy wprowadzać kategorii i atrybutów nieistotnych dla użytkowników i dla realizowanych funkcji zbędnych („na wszelki wypadek").

Zbiory atrybutów poszczególnych kategorii muszą być rozłączne, czyli jeśli *IdK* jest numerem klienta w kategorii **Klient**, to nie może w tej roli (jako numer klienta) wystąpić w żadnej innej kategorii (np. w danych pracownika), gdyż cecha ta opisuje klienta. Formalnie dozwolone jest użycie nazwy *IdK* jako numeru kraju w kategorii **Kraj**, ale nie jest to zalecane, gdyż nazwa ta przestaje być jednoznaczna w projekcie, co zmniejsza jego czytelność, utrudnia poprawne posługiwanie się nazwami atrybutów i formułowanie zapytań.

Definicje kategorii należy wypisać, porządkując je logicznie i rozpoczynając od najważniejszych.

Po zdefiniowaniu kategorii można sformułować dla nich reguły funkcjonowania (rozdz. 3.4) i ograniczenia dziedzinowe (rozdz. 3.5).

3.4. REGUŁY FUNKCJONOWANIA

Z przeprowadzonej analizy rzeczywistości przedstawionej w postaci opisu w języku naturalnym (rozdz. 3.2) należy sporządzić listę istotnych reguł funkcjonowania zapisanych w określony sposób.

Reguła funkcjonowania jest sformalizowanym zapisem faktu z rzeczywistości dotyczącym albo powiązań pomiędzy kategoriami zdefiniowanymi w rozdziale 3.3., albo uprawnień użytkowników. Każda reguła ma swój unikalny identyfikator postaci REG/*xxx*.

Definicja reguły funkcjonowania:

REG/*xxx* sformułowanie reguły funkcjonowania w języku naturalnym

gdzie *xxx* jest kolejnym trzycyfrowym numerem reguły.

Reguły należy wypisywać w sposób systematyczny, nadając im unikalne identyfikatory. Każda reguła powinna być zapisana w sposób możliwie krótki, a zarazem jednoznaczny i precyzyjny. Powinna opisywać jeden wybrany fakt.

Wśród reguł powinny być zarówno reguły dotyczące powiązań między kategoriami, jak i dotyczące uprawnień użytkowników do operacji wykonywanych na danych kategoriach. W regułach dotyczących powiązań należy uwzględnić to, czy dana kategoria **musi** czy **może** brać udział w danym zdarzeniu (powiązaniu) oraz czy jej udział może być **jednokrotny** czy **wielokrotny**.

Prawidłowe i wyczerpujące określenie reguł funkcjonowania dotyczących realizowanych procesów biznesowych, istotnych z punktu projektu bazy danych, oraz ich przestrzeganie we wszystkich kolejnych etapach to warunki konieczne do poprawnego zaprojektowania bazy, a w przyszłości – systemu bazy danych.

Przykład 3.2 (reguły funkcjonowania)

REG/001 Dane towarów wpisuje kierownik magazynu

REG/002 Faktura musi być przypisana do klienta

REG/003 Fakturę można usunąć po upływie 5 lat od wystawienia

REG/004 Przy sprzedaży towaru sprzedawca może przyznać rabat w określonej wysokości 3%, 5% lub 8% ceny netto towaru

REG/005 Towar nie musi być powiązany z żadną dostawą

REG/006 Towar może być powiązany z wieloma dostawami
REG/007 Dostawa musi być powiązana co najmniej z jednym towarem
REG/008 Dostawa może być powiązana z wieloma towarami

Reguły można pogrupować ze względu na zdefiniowane kategorie, których dotyczą, oraz zdefiniować reguły ogólne, np.

Reguły ogólne
REG/001 Tekst
REG/002 Tekst

Reguły dotyczące KAT/001 Nazwa kategorii
REG/003 Tekst
REG/004 Tekst

Reguły dotyczące KAT/002 Nazwa kategorii
REG/005 Tekst
REG/006 Tekst

Reguły funkcjonowania muszą być formułowane jednoznacznie i precyzyjnie, ponieważ służą do definiowania transakcji (rozdz. 3.6) i do definiowania związków (rozdz. 3.7).

3.5. OGRANICZENIA DZIEDZINOWE

Ograniczenia dziedzinowe powinny wynikać z analizy wycinka rzeczywistości (rozdz. 3.2.1) i należy je uwzględnić w dalszych etapach projektowania bazy danych.

Ograniczenia dziedzinowe dotyczą atrybutów kategorii: ich typów, dopuszczalnych wartości, obligatoryjności / opcjonalności, unikalności, wartości domyślnych itp. Każde ograniczenie ma swój unikalny identyfikator w postaci OGR/xxx.

Definicja ograniczenia dziedzinowego

OGR/*xxx* sformułowanie ograniczenia w języku naturalnym

gdzie *xxx* jest kolejnym trzycyfrowym numerem ograniczenia.

Ograniczenia dziedzinowe należy wypisywać w sposób systematyczny – można je pogrupować z podziałem na kategorie, których dotyczą, oraz zdefiniować ograniczenia ogólne.

Jeśli w opisie ograniczeń są używane specyficzne oznaczenia (symbole), to należy je zdefiniować przed listą ograniczeń wcześniej lub w danym ograniczeniu.

Przykład 3.3 (ograniczenia dziedzinowe)

OGR/001	Data urodzenia jest wcześniejsza niż data zatrudnienia
OGR/002	Data zatrudnienia jest wcześniejsza niż data zwolnienia
OGR/003	Kod pocztowy ma postać 99-999, gdzie 9 oznacza dowolną cyfrę
OGR/004	Numer legitymacji jest ciągiem 10 znaków w postaci: Numer ucznia/Rok wystawienia
OGR/005	Liczba sztuk towaru na fakturze jest liczbą całkowitą dodatnią
OGR/006	Każdy towar ma unikalny symbol nadany przez kierownika
OGR/007	Nazwisko jest ciągiem maksymalnie 30 znaków: liter, myślnika, apostrofu i spacji
OGR/008	NIP klienta jest atrybutem obowiązkowym
OGR/009	Wszystkie atrybuty dostawcy są obligatoryjne
OGR/010	Liczba uczniów w klasie jest liczbą całkowitą z przedziału [0, 35]

■

Ograniczenia można pogrupować ze względu na zdefiniowane kategorie, których dotyczą, oraz zdefiniować ograniczenia ogólne, np.

Ograniczenia ogólne

OGR/001 Format daty: dd-mm-rrrr (dd – dzień, mm – miesiąc, rrrr – rok)

Ograniczenia dla KAT/001 Nazwa kategorii

OGR/002 Tekst

OGR/003 Tekst

Ograniczenia dla KAT/002 Nazwa kategorii
OGR/004 Tekst
OGR/005 Tekst

Należy zwrócić uwagę, by dla wszystkich atrybutów sprecyzować wymagane ograniczenia. Są one istotne przy definiowaniu typów encji oraz schematów relacji podczas implementacji bazy. Sformułowane ograniczenia nie mogą być jednak nadmierne, zbyt restrykcyjne, ograniczające w przyszłości wprowadzanie danych do bazy.

Zdefiniowane ograniczenia są podstawą do zdefiniowania dziedzin atrybutów encji w rozdz. 3.7.

3.6. TRANSAKCJE

Transakcje są to operacje wykonywane na danych. Są wygodnym mechanizmem pozwalającym na grupowanie wielu operacji wykonywanych na danych i traktowanie ich jako jednej niepodzielnej funkcjonalnie całości. Wynikają z przeprowadzonej analizy i są opisywane za pomocą pojęć używanych w opisie rzeczywistości formułowanych w języku naturalnym.

Transakcje powinny mieć cztery podstawowe własności, zwane w skrócie **własnościami ACID** (*Atomicity, Consistency, Isolation, Durability*):

- **niepodzielność, atomowość** (ang. *atomicity*) – transakcja jest wykonywana w całości albo wcale. Po przerwaniu transakcji musi być odtworzony stan bazy danych sprzed rozpoczęcia transakcji,
- **spójność** (ang. *consistency*) – transakcje zachowują spójność bazy danych, chociaż w trakcie działania transakcji stan bazy może być chwilowo niespójny,
- **izolacja** (ang. *isolation*) – transakcje są całkowicie od siebie niezależne, jedna transakcja od drugiej jest odizolowana,
- **trwałość** (ang. *durability*) – zmiany dokonane przez pomyślnie zakończoną transakcję są zachowywane na trwałe.

3.6.1. IMPLEMENTACJA TRANSAKCJI

Implementacja transakcji zawiera trzy podstawowe części:

- rozpoczęcie transakcji, np. BeginTrans, BeginTransaction,
- treść transakcji, czyli ciąg instrukcji wykonywanych na danych,
- zakończenie transakcji wskutek napotkania jednej z instrukcji: Commit (pomyślne zakończenie transakcji), Abort (przerwanie transakcji), Rollback – przywrócenie stanu bazy sprzed rozpoczęcia transakcji.

W przypadku zerwania transakcji (ang. *abort*) przywracany jest stan bazy sprzed rozpoczęcia transakcji na podstawie dziennika transakcji (plik o nazwie *log*), w którym są zapisywane poszczególne stany bazy danych przed i po wykonaniu każdej transakcji. Dziennik taki jest czytany od ostatniego zapisu do początku i przywracany jest stan bazy (ang. *rollback*).

Na listingu 3.1 przedstawiono napisany w języku Visaul Basic for Application (VBA) kod procedury zawierającej transakcję, której zadaniem jest wstawienie do pola *Pensja* najpierw wartości 100, a potem wartości 200 (dla wszystkich nauczycieli).

Po napotkaniu instrukcji rs.Open – ze względu na brak instrukcji zamknięcia rs.Close (zaznaczonej jako komentarz: , rs.Close) – system sygnalizuje błąd wykonania procedury, cofa wprowadzone zmiany pensji (RollbackTrans) i pensje w bazie danych mają nadal dotychczasowe wartości.

Jeśli w kodzie procedury symbol komentarza występujący przy instrukcji rs.Close zostanie usunięty, to transakcja wykona się poprawnie i wszystkie pensje będą miały wartość 200.

Jeżeli ciąg instrukcji nie byłby zapisany jako transakcja, to wszystkie pensje otrzymałby wartość 100 i system zasygnalizowałby błąd wykonania podczas próby wykonania instrukcji rs.Open.

Przyczyną przerwania transakcji może być awaria sprzętu, brak prądu, błędna instrukcja, zły rozmiar pola itp. Jeśli pole *Nazwisko* ma dopuszczalny rozmiar 10 znaków i należy wykonać instrukcję zastąpienia nazwiska łańcuchem: „P. „ & *Nazwisko*, to dla nazwiska Kotarski otrzymany łańcuch ma 11 znaków, co spowoduje przerwanie transakcji ze względu na przekroczony rozmiar pola.

```
Private Sub Polecenie0_Click()
Dim conn As ADODB.Connection
Dim rs As ADODB.Recordset
On Error GoTo err_Trans
Set conn = CurrentProject.Connection

conn.BeginTrans
 Set rs = conn.Execute("Update Nauczyciele Set Pensja = 100;")
 Set rs = conn.Execute("SELECT COUNT(*) FROM Nauczyciele")
 MsgBox "100 - Liczba nauczycieli: " & rs(0)    'Wynik: Liczba nauczycieli: 54

 ' rs.Close                                      'komentarz zamiast instrukcji
 rs.Open "Select * From Nauczyciele", conn, adOpenStatic
 Do While Not rs.EOF
  Debug.Print rs.Fields("Pensja").Value
  rs.MoveNext
 Loop
 rs.Close
   Set rs = conn.Execute("Update Nauczyciele Set Pensja = 200;")
 Set rs = conn.Execute("SELECT COUNT(*) FROM Nauczyciele")
 MsgBox "200 - Liczba nauczycieli: " & rs(0)    'Wynik: Liczba nauczycieli: 54

conn.CommitTrans
conn.Close
Exit Sub

err_Trans:
 MsgBox Err.Description
 conn.RollbackTrans
 conn.Close

End Sub
```

Listing. 3.1. Transakcja w VBA

3.6.2. DEFINIOWANIE TRANSAKCJI

Etap 6 projektu związany z definiowaniem transakcji jest w zasadzie uszczegółowieniem etapu analizy wymagań funkcjonalnych (rozdz. 3.2.4). Opisując poszczególne transakcje, należy określić, kto jest uprawniony do wykonania transakcji, które kategorie biorą udział w transakcji, co jest źródłem danych wejściowych oraz co jest wynikiem transakcji (jakie dane są na wyjściu).

W określeniu, skąd pochodzą dane wejściowe i gdzie są kierowane wyniki transakcji, można użyć oznaczeń:

U – użytkownik,
BD – baza danych,
S – system.

Przez użytkownika rozumiemy osobę lub aplikację korzystającą z bazy danych.

Przez system rozumiemy system operacyjny, z którego mogą być pobrane dane systemowe, np. aktualna data, czas, nazwa katalogu. W wielu transakcjach nie pobiera się żadnych danych z systemu, wówczas nie trzeba umieszczać go na wejściu transakcji.

W etapie 6 należy zdefiniować wszystkie przewidywane transakcje dla wszystkich grup użytkowników, w tym także transakcje logowania / wylogowania użytkownika, generowania statystyk i raportów.

Definicja transakcji

TRA/*xxx* Nazwa transakcji

Opis: Słowny opis transakcji. Użytkownicy uprawnieni do jej wykonywania.
Uwarunkowania: Specyfikacja warunków dla prawidłowego wykonania transakcji oraz opis ścieżki alternatywnej, do której może dojść, jeśli warunki te nie będą spełnione.
Wejście/Wyjście: Tabela transakcji

Wejście/Wyjście TRA/*xxx*

	Wejście	**Wyjście**
Użytkownik		
Baza danych		
System		

gdzie *xxx* jest kolejnym numerem transakcji.

Przykład 3.4 (opisy transakcji)

TRA/001 Edycja danych towaru

Opis: Zadaniem transakcji jest wyszukanie danych o wybranym towarze i edycja tych danych, np. ceny sprzedaży. Wyszukanie danych towaru może być wykonane po jego nazwie lub symbolu. Edycję może wykonać tylko kierownik magazynu.

Uwarunkowania: Dane towaru muszą istnieć w bazie. Jeśli dane towaru nie zostaną znalezione, zostanie wyświetlony komunikat *Brak danych*. Po wykonaniu edycji danych towaru, użytkownik otrzymuje komunikat *Edycja zakończona.*

Tabela 3.1

Wejście/Wyjście TRA/001

	Wejście	**Wyjście**
Użytkownik	Dane identyfikujące towar Nowe dane towaru	Komunikat
Baza danych	Dane towarów	Dane towarów
System	Data aktualizacji	

TRA/002 Zapisanie klienta na trening

Opis: Zadaniem transakcji jest zapisanie klienta na trening. Transakcję może wykonać recepcjonista.

Uwarunkowania: Dane klienta i treningu muszą musi istnieć w bazie, w przeciwnym razie transakcja nie zostanie wykonana i zostanie wyświetlony komunikat *Brak danych.* Po poprawnym wykonaniu transakcji użytkownik otrzymuje komunikat potwierdzający zapisanie danego klienta na wybrany trening przez danego pracownika recepcji.

Tabela 3.2

Wejście/Wyjście TRA/002

	WEJŚCIE	**WYJŚCIE**
Użytkownik	Dane identyfikujące klienta Dane identyfikujące trening	Komunikat
Baza danych	Dane klientów Dane treningów Dane zapisów Dane pracowników	Dane zapisów

TRA/003 Usunięcie danych miasta

Opis: Zadaniem transakcji jest usunięcie danych miasta o podanej nazwie. Usunięcie miasta może wykonać tylko administrator.

Uwarunkowania: Dane miasto musi istnieć w bazie. Jeśli nie ma takiego miasta, to transakcja nie zostanie wykonana i zostanie wyświetlony komunikat *Brak danych*. Jeśli dane miasto jest związane z jakimś klientem, to miasta nie można usunąć z bazy (usuwanie ograniczone) i jest wyświetlany komunikat *Danych miasta nie można usunąć*. Po usunięciu miasta użytkownik otrzymuje komunikat *Miasto zostało usunięte*.

Tabela 3.3

Wejście/Wyjście TRA/003

	WEJŚCIE	WYJŚCIE
Użytkownik	Nazwa miasta	Komunikat
Baza danych	Dane miast Dane klientów	Dane miast

Definicje transakcji wynikają z wcześniejszych etapów, głównie z analizy wymagań funkcjonalnych (rozdz. 3.2.4), z definicji kategorii (rozdz. 3.3) i z reguł funkcjonowania (rozdz. 3.4). Z tego względu nie jest możliwa ocena poprawności transakcji tylko na podstawie jej definicji. Poprawność transakcji musi być zweryfikowana poprzez zestawienie z regułami funkcjonowania.

3.7. IDENTYFIKACJA TYPÓW ENCJI I ZWIĄZKÓW

Opracowanie modelu konceptualnego wymaga wyodrębnienia i zdefiniowania typów encji oraz związków między nimi. Model danych jest pewnym uproszczonym, lecz zgodnym z modelowanym wycinkiem rzeczywistości opisem. Celem modelowania konceptualnego jest zapewnienia wyższego poziomu abstrakcji widzenia danych z perspektywy użytkownika oraz niezależności danych od implementacji (aplikacji).

Aby zapewnić poprawność modelu danych w sensie zgodności, często podaje się *metamodel danych*, czyli model modelu. Metamodel obejmuje definicje pojęć (np. encji, atrybutu) i reguły dotyczące pojęć wykorzystywanych w modelu, np. jeśli model zawiera co najmniej dwie encje, to każda encja musi być co najmniej w jednym związku z dowolną inną encją.

Etap 7 obejmuje dwa podetapy:

Etap 7.1 – Definicje typów encji

Etap 7.2 – Definicje typów związków

Po wyodrębnieniu typów encji i związków [ElNav2005], przyjmując ustaloną notację (np. Chena [Chen1976, Barker1996, MurRyb1993], Martina [Beynon2000] lub Yourdona [Yourdon1996]), można przedstawić graficznie opracowany model, czyli zbudować diagram związków encji (rozdz. 3.9).

Na podstawie prawidłowo przeprowadzonej analizy rzeczywistości zdefiniowano kategorie (etap 3, rozdz. 3.3) i ograniczenia dziedzinowe (etap 5, rozdz. 3.5), które są podstawą do zdefiniowania typów encji. Natomiast na podstawie reguł funkcjonowania (etap 4, rozdz. 3.4) można zdefiniować związki. Opracowany model konceptualny będzie podstawą do zdefiniowania znormalizowanych schematów relacji bazy danych (rozdz. 4.2). Znormalizowana baza danych zajmuje mniej miejsca, dane się nie powtarzają, łatwiejsze jest dopisywanie, modyfikowanie i usuwanie danych. Dzięki normalizacji (rozdz. 2.3) baza danych jest lepsza w tym sensie, że nie występują w niej anomalie, czyli nieprawidłowości związane z uaktualnianiem (ang. *update anomalies*), usuwaniem (ang. *deletion anomalies*) i dopisywaniem danych (ang. *insertion anomalies*).

W rozdziałach 3.7.1 i 3.7.2 podamy definicje podstawowych pojęć wykorzystywanych w tym etapie projektowania bazy danych oraz sposoby opisu typów encji i związków.

3.7.1. DEFINICJE TYPÓW ENCJI

Etap definiowania typów encji, które wynikają z wyodrębnionych w rozdziale 3.3 kategorii, obejmuje ich prawidłowe wyodrębnienie i opisanie oraz zdefiniowanie ich atrybutów, dziedzin, kluczy kandydujących i głównych.

Encja (ang. *entity*) – w języku łacińskim *esse* znaczy „być" [Boratyn1995]. Encja jest to pewien obiekt, rzecz, zjawisko, zdarzenie, byt. Encje posiadają atrybuty,

czyli cechy je opisujące (charakteryzujące), istotne z punktu widzenia eksperta dziedzinowego, przyjmujące wartości z ustalonego zbioru – **dziedziny**.

Encje o takich samych cechach charakterystycznych możemy pogrupować w typy encji. Zwróćmy uwagę, że encja to nie jest to samo co typ encji.

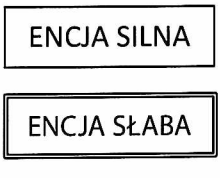
Typ encji jest definicją bytu (odpowiada kategorii), a encja jest instancją (wystąpieniem) danego typu encji [Beynon2000].

Typem encji jest np. OSOBA o atrybutach (*IdO*, *Nazwisko*, *Imię*), a encją jest jeden obiekt danego typu OSOBA, np. (5, "Kowalski", "Jan"). Powszechnie używa się terminu encja zarówno na określenie typu encji, jak i samej encji, czyli określenie encja OSOBA należy rozumieć jako encja typu OSOBA. Tam, gdzie nie prowadzi to do nieporozumień, można używać pojęcia encja zamiast typ encji.

Typom encji nadajemy nazwy, które powinny być rzeczownikami w mianowniku liczby pojedynczej. Nazwy typów encji piszemy wielkimi literami, np. PRACOWNIK. Nazwa typu encji jest jednocześnie nazwą encji. Graficznym przedstawieniem encji danego typu jest prostokąt narysowany linią pojedynczą (encja silna) lub podwójną (encja słaba) z umieszczoną wewnątrz nazwą typu encji, co zilustrowano na rysunku 3.1. W niektórych notacjach wyszczególniane są również atrybuty typu encji.

Encja silna (ang. *strong entity*) – zwana też encją regularną lub właściwą – oznacza obiekt, który może istnieć niezależnie od innych obiektów (encji). Graficznie encja silna jest reprezentowana przez prostokąt z nazwą umieszczoną wewnątrz narysowany linią **pojedynczą**.

Encja słaba (ang. *weak entity*) oznacza obiekt, który nie może istnieć samodzielnie, czyli nie może istnieć bez istnienia innych obiektów (encji). Graficznie encja silna jest reprezentowana przez prostokąt z nazwą umieszczoną wewnątrz narysowany linią **podwójną**.



Rys. 3.1. Graficzne przedstawienie encji silnej i encji słabej

Encja słaba jest reprezentowana przez prostokąt narysowany linią podwójną, gdyż na ogół na ten typ encji trzeba zwrócić szczególną uwagę.

Jeżeli w projekcie zdefiniowano dwie kategorie **Uczelnia** i **Wydział**, a z reguł funkcjonowania wynika, że każdy wydział **musi** być przypisany do dokładnie jednej uczelni, to encja WYDZIAŁ jest encją słabą. Jeśli nie ma żadnej reguły funkcjonowania, z której wynika, że kategoria **Uczelnia** musi być powiązana z inną kategorią, to encja UCZELNIA jest encją silną. Jeśli natomiast jest chociaż jedna reguła, że uczelnia musi być powiązana z inną kategorią, np. z kategorią **Kraj**, to encja UCZELNIA jest wówczas też encją słabą (nie może istnieć bez encji KRAJ).

Atrybut (ang. *attribute*) – cecha charakterystyczna encji lub związku, element służący do identyfikowania, opisu, określania ilości bądź stanu encji lub związku; dowolny opis mający znaczenie dla encji lub związku. Atrybut powinien przyjmować dokładnie jedną wartość w danej chwili (w danym stanie bazy), np. liczbę, datę, tekst, która z punktu widzenia projektanta jest atomowa (niepodzielna). Nazwa atrybutu powinna być rzeczownikiem w liczbie pojedynczej, np. *Nazwisko*) – jeśli występuje w liczbie mnogiej to jest to sygnał, że należy utworzyć nową encję, np. *Imiona dzieci*, *Języki obce* – lub może być nazwą jednoczłonową, np. *IdU*, *DataUr*.

Atrybut opcjonalny (ang. *optional attribute*) – atrybut nieobowiązkowy, jego wartość może być nieokreślona, pusta, nieznana (oznaczana jako NULL).

Atrybut obligatoryjny *(ang. mandatory attribute)* – atrybut wymagany, jego wartość musi być zawsze określona. Nie może przyjmować wartości NULL.

NULL – symbol (oznaczenie) wartości pustej, różnie interpretowanej, np. jako wartość nieznana, nieistniejąca, nieokreślona, niepewna, zastrzeżona. Dwa atrybuty o wartości NULL nie są traktowane jako sobie równe w tym sensie, że taka wartość może być różnie interpretowana, np. wartość NULL jako wartość atrybutu *Telefon* może oznaczać jego brak, nieznany numer telefonu lub daną chronioną.

Christopher Date w rozdziale piątym pracy [Date2000] omówił zagadnienia związane z wprowadzeniem pojęcia NULL do baz danych. Jego zdaniem problem nie jest w pełni zrozumiany i nie jest rozwiązany, a pojęcie NULL nie powinno być wprowadzone do modelu relacyjnego ([Date2000], s. 152), ale E.F. Codd miał na ten temat zupełnie przeciwne zdanie [Codd1985b].

Definiując atrybuty typu encji, powinno się:

- określić nazwy dla atrybutów,

- podać opis (znaczenie) każdego atrybutu, jego typ (format, dziedzinę), dopuszczalne wartości,
- zidentyfikować atrybuty obligatoryjne (wymagane) i opcjonalne (dopuszczające wartości NULL).

Dla każdego typu encji należy określić klucze kandydujące i klucz główny oraz charakter encji (silna czy słaba). W przypadku, gdy w danym typie encji:

- brak jest naturalnego klucza kandydującego,
- klucze kandydujące są złożone z wielu atrybutów,
- klucze kandydujące są proste (jeden atrybut), ale są to dane chronione, atrybuty tekstowe o dużej liczbie znaków lub liczby o wielu cyfrach znaczących,
- wówczas należy zdefiniować sztuczny klucz encji.

Dane osobowe i dane chronione nie powinny być kluczami głównymi np. PESEL, login, numer dowodu osobistego lub paszportu, numer konta w banku (szczególnie, gdy nie ma całkowitej pewności co do unikalności ich wartości wynikających z błędnego tworzenia numerów lub z błędnego ręcznego wprowadzenia danych, np. numeru PESEL).

Definicja typu encji:

ENC/*xxx* NAZWA TYPU ENCJI

Semantyka encji: opis typu encji

Wykaz atrybutów: lista atrybutów, ich opis i dziedzina (typ), przedstawiona w formie tabeli atrybutów

Nazwa atrybutu	**Opis atrybutu**	**Typ**	**OBL (+)** **OPC (–)**

Klucze kandydujące: minimalne zbiory atrybutów jednoznacznie identyfikujących encje

Klucz główny: jeden z kluczy kandydujących wybrany jako główny

Charakter encji: encja silna lub słaba

gdzie *xxx* jest kolejnym trzycyfrowym numerem typu encji.

Definicje typów encji należy wypisać w kolejności zgodnej z definicjami kategorii w rozdziale 3.3.

Na bazie tego samego typu można zdefiniować wiele dziedzin poprzez określenie ich cech indywidualnych, takich jak np. maksymalna liczba znaków, liczba miejsc po przecinku, wzorzec (maska) wprowadzania danych, wartości domyślne, ograniczenie zakresu dozwolonych wartości, dopuszczalność wartości NULL itp. Można określić np. dziedzinę *Nazwisko*, która będzie oznaczała ciągi co najwyżej 25-znakowe, które mogą zawierać litery, myślnik, apostrof, spację. Tak zdefiniowana dziedzina może być przypisana do atrybutów *NazwiskoKlienta, NazwiskoP, NazwaFirmy.*

Przykład 3.5 (opisy typów encji)

ENC/001 OSOBA

Semantyka encji: encja zawiera dane pracownika szkoły. Pracownikami są nauczyciele, pracownicy administracyjni, techniczni i obsługi szkoły. Szkoła może zatrudniać obcokrajowców pochodzących z różnych krajów.

Wykaz atrybutów:

Tabela 3.4

Wykaz atrybutów encji typu OSOBA

Nazwa atrybutu	Opis atrybutu	Typ	OBL (+) OPC (–)
NrO	Identyfikator osoby	Liczba naturalna	+
Nazwisko	Nazwisko osoby	Ciąg max. 25 znaków	+
Imie	Imię osoby	Ciąg max. 15 znaków	+
DataUr	Data urodzenia osoby	Data	+
NIP	Numer Identyfikacji Podatkowej	Znakowy postaci {999-999-99-99}	–
PESEL	Numer PESEL	Ciąg 11 cyfr	–
NrDok	Numer dokumentu tożsamości	Ciąg max. 15 znaków	+
TypD	Typ dokumentu tożsamości	D – dowód osobisty, P – paszport	+
WykO	Wykształcenie osoby	{podstawowe, średnie, wyższe}	+
Login	Nazwa użytkownika	Ciąg liter i cyfr, max. 15 znaków	+
Haslo	Hasło logowania na konto	Ciąg max. 15 znaków	+

Klucze kandydujące: *NrO*, *Login*, *NrDok*
Klucz główny: *NrO*
Charakter encji: encja słaba

ENC/002 PRZEDMIOT

Semantyka encji: encja zawiera dane o przedmiocie nauczanym w szkole

Wykaz atrybutów:

Tabela 3.5

Wykaz atrybutów encji typu PRZEDMIOT

Nazwa atrybutu	Opis atrybutu	Typ	OBL (+) OPC (–)
NrP	Numer porządkowy	Liczba całkowita dodatnia (< 100)	+
NazwaP	Nazwa przedmiotu nauczanego w szkole	Ciąg max. 30 znaków	+

Klucze kandydujące: *NrP*, *NazwaP*
Klucz główny: *NrP*
Charakter encji: encja silna

Aby ocenić, czy zdefiniowane typy encji są poprawne, trzeba odnieść się do analizy rzeczywistości. Załóżmy, że zdefiniowano typ encji KSIĄŻKA. Z analizy rzeczywistości powinno wynikać, czy *wydawnictwo* powinno być postrzegane jako typ encji WYDAWNICTWO, czy jako atrybut *Wydawnictwo* w typie encji KSIĄŻKA. Definiując *Wydawnictwo* jako atrybut, należy podać dziedzinę, czyli zbiór wartości, jakie ten atrybut może przyjmować. W tym wypadku możemy zdefiniować dziedzinę np. jako zbiór łańcuchów co najwyżej 50-znakowych (nazwy wydawnictw) lub jako zbiór symboli (np. WNT, PWN) przypisanych poszczególnym wydawnictwom. Definiując WYDAWNICTWO jako typ encji, można podać wiele atrybutów opisujących wydawnictwo, np. nazwę pełną wydawnictwa, nazwę skróconą, adres, telefon, e-mail, stronę WWW, rok założenia, nazwisko i imię dyrektora, itp. Jeśli definicje encji nie odpowiadają wymaganiom, należy poprawić projekt od początku.

Podobnie jak w kategoriach, atrybuty encji powinny opisywać cechy charakterystyczne danej encji, a nie jej związek z innymi encjami. Na przykład, jeśli

została wyodrębniona encja DOSTAWCA i jej kluczem głównym jest *IdDostawcy*, to encja TOWAR nie może mieć atrybutu *IdDostawcy* wskazującego na związek z encją DOSTAWCA, gdyż jest to cecha encji DOSTAWCA. Zatem encje nie powinny mieć żadnych atrybutów wspólnych (łączących). W encjach (podobnie jak w kategoriach) nie ma żadnych kluczy obcych.

Po zdefiniowaniu encji należy sprawdzić, czy każdy atrybut ma tylko jedną wartość w danej chwili oraz czy nie istnieje potrzeba przechowywania kolejnych zmieniających się wartości atrybutu, np. adresu zameldowania. Jeśli w celu zapewnienia trwałości bazy danych konieczne jest przechowywanie historii zmian danych, czyli wszystkich zmieniających się wartości atrybutu (np. stanowisko pracownika, cena towaru, adres zamieszkania, wypożyczenia książki, pobyt w hotelu), to należy uwzględnić takie wymagania w projekcie, definiując odpowiednie kategorie, reguły funkcjonowania, ograniczenia dziedzinowe, transakcje, typy encji i związków.

W celu zweryfikowania, czy w projektowanej bazie danych nie będzie zbędnych (nigdy niewykorzystywanych) danych, należy sprawdzić, czy każda encja i każdy atrybut uczestniczy w co najmniej jednej transakcji.

Wyodrębnienie typów encji i ich precyzyjny opis (atrybutów i dziedzin) ułatwi zdefiniowanie modelu logicznego (schematów relacji tworzących schemat bazy danych), a następnie modelu fizycznego relacyjnej bazy danych (tabel).

3.7.2. DEFINICJE TYPÓW ZWIĄZKÓW

Na podstawie reguł funkcjonowania (rozdz. 3.4) i wyodrębnionych typów encji (rozdz. 3.7.1) należy zidentyfikować i zdefiniować wszystkie typy związków.

Typ związku (ang. *relationship type*) jest to powiązanie (ang. *association*) pomiędzy typami encji (najczęściej dwoma).

Formalnie typ związku **Z** można zapisać jako relację o *n* argumentach (typach encji E_1, ..., E_n):

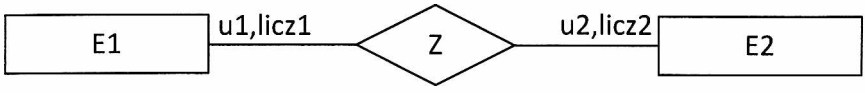
$$\mathbf{Z}(E_1,\ldots, E_n)$$

co oznacza, że encje typu $E_1,\ldots, E_n$ uczestniczą w związku **Z**.

Podobnie jak w przypadku encji należy rozróżniać typ związku od instancji związku. Na przykład instancja związku binarnego $\mathbf{Z}(E_1, E_2)$ jest dwuargumentową relacją na iloczynie kartezjańskim zbiorów instancji encji E_1 i E_2. Powszechnie

rozróżnienie między typem związku a związkiem jest ignorowane i nazwa **Z** jest oznacza zarówno nazwę typu związku, jak i nazwę związku.

Graficznie związek reprezentujemy jako romb zawierający nazwę związku (rysunek 3.2). Jest on połączony krawędziami z encjami uczestniczącymi w związku.



Rys. 3.2. Graficzny zapis związku

Związek binarny (między dwiema encjami) ma dwie krawędzie, związek ternarny zachodzi pomiędzy trzema encjami. Związek może zachodzić również między encjami tego samego typu.

Nazwami związków powinny być czasowniki w 3. osobie liczby pojedynczej, np. Mieszka, Jest, Ma, Dotyczy, Ocenia, Produkuje, Dostarcza, Zatwierdza, Wynajmuje. Nazwy związków muszą być unikalne i dobrze dobrane semantycznie.

Typy związków opisuje się, używając pojęć: **stopień, typ uczestnictwa** i **liczność.**

Stopień związku (ang. *degree*) – określa liczbę encji uczestniczących w związku, np. stopień związku binarnego wynosi 2.

Uczestnictwo encji w związku może być:

- **opcjonalne** (ang. *optional*) – oznaczane jako 0, jeśli nie każda encja typu E1 musi uczestniczyć w związku **Z**, czyli mogą istnieć encje typu E1 nie uczestniczące w związku **Z**,
- **obligatoryjne** (ang. *obligatory*) – oznaczane jako 1, jeśli wszystkie encje typu E1 biorą udział w związku **Z**.

W omawianej metodyce w graficznym przedstawieniu związku **Z**(E1, E2) (rys. 3.2) uczestnictwo w związku **Z** encji E1 zapisywane jest przy encji E2 jako *u2*.

W innych notacjach symbole te mogą mieć inną postać, np. w notacji Chena, wprowadzonej w 1976 roku, uczestnictwo obligatoryjne oznacza symbol OBL obok specyfikowanego typu encji, a opcjonalne – symbol OPC obok typu encji.

Liczność (ang. *cardinality*) określana jest dla każdego końca związku jako:

- **jeden** (zapisujemy jako **1**), tzn. że w związku może uczestniczyć co najwyżej jedna encja danego typu,

- **wiele** (zapisujemy jako **N**), tzn. że w związku może uczestniczyć zero lub więcej encji.

Ze względu na liczność wyróżniamy następujące typy związków binarnych (stopnia drugiego) $\mathbf{Z}(E_1, E_2)$:

1:1 – **jeden do jeden**

jedna encja typu E_1 może być powiązana co najwyżej z jedną encją typu E_2 i jedna encja typu E_2 może być powiązana co najwyżej z jedną encją typu E_1, np.

– osoba ma co najwyżej jeden dowód osobisty
– dowód osobisty należy co najwyżej do jednej osoby

1:N – **jeden do wielu**

jedna encja typu E_1 może być powiązana z wieloma encjami typu E_2 i jedna encja typu E_2 może być powiązana co najwyżej z jedną encją typu E_1, np.

– w mieście może mieszkać wiele osób
– osoba ma tylko jedno miasto stałego zameldowania

N:N – **wiele do wielu**

jedna encja typu E_1 może być powiązana z wieloma encjami typu E_2 i jedna encji typu E_2 może być powiązana z wieloma encjami typu E_1, np.

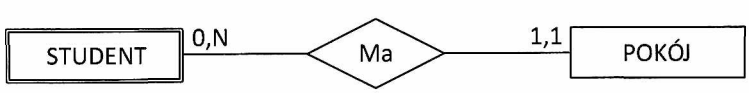
– student może się zapisać na wiele kursów
– na kurs może się zapisać wielu studentów

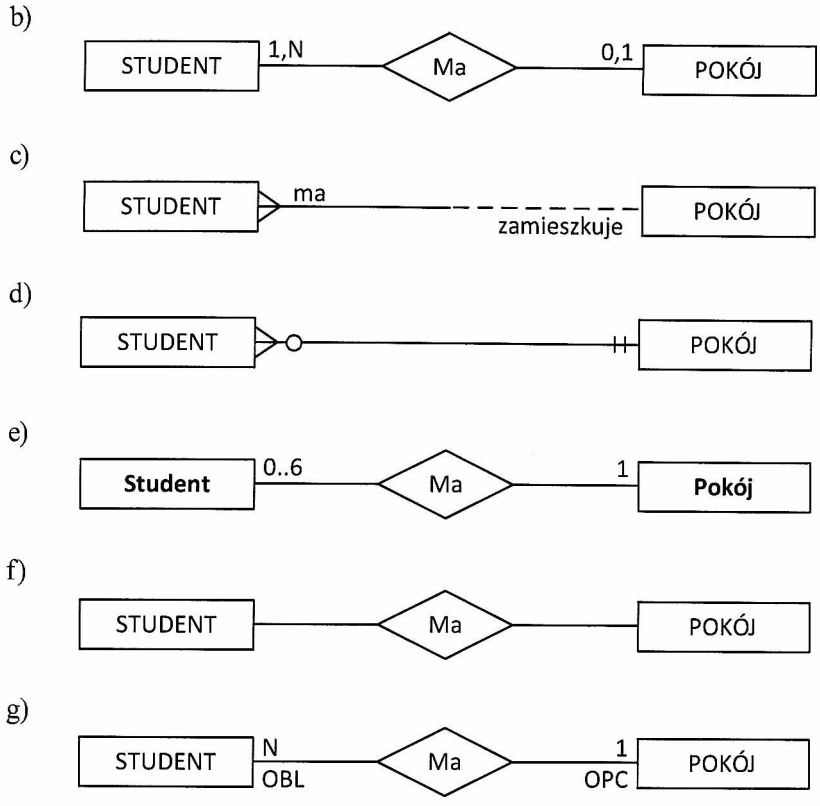
Przykład 3.6

Załóżmy, że w bazie danych **AKADEMIK** obowiązują następujące reguły:
REG/001 Student musi mieć przypisany pokój
REG/002 Student może mieć przypisany co najwyżej jeden pokój
REG/003 Pokój nie musi mieć przypisanego studenta
REG/004 Pokój może mieć przypisanych wielu studentów (max. 6)

Graficzny zapis tych reguł w różnych notacjach przedstawiono na rysunku 3.3.

a)





Rys. 3.3. Różne sposoby graficznego przedstawienia typu związku

a) notacja stosowana w niniejszej książce
b) notacja zmodyfikowana Chena
c) notacja wg Barkera
d) notacja Martina
e) notacja UML
f) notacja Yourdona (bez liczności i typu uczestnictwa)
g) notacja Chena

Przeanalizujmy podane reguły i ich zapis graficzny w notacji przedstawionej na rysunku 3.3a:

REG/001 Student musi mieć przypisany pokój
tzn. każda encja STUDENT **musi** być powiązana z encją POKÓJ (uczestnictwo obligatoryjne w związku **Ma**) i dlatego przy encji POKÓJ na pierwszym miejscu jest symbol 1. Nie może być studenta bez przypisanego pokoju.

REG/002 Student może mieć przypisany co najwyżej jeden pokój
tzn. jedna encja STUDENT może być powiązana co najwyżej z jedną encją POKÓJ, a więc na rysunku przy encji POKÓJ na drugim miejscu (jako liczność) jest 1.

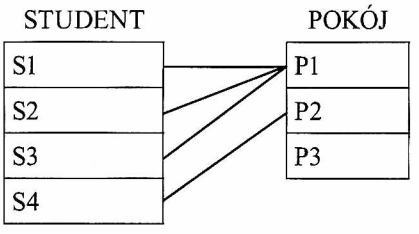
REG/003 Pokój nie musi mieć przypisanego studenta
tzn. może być encja POKÓJ, która nie jest powiązana z żadną encją STUDENT, czyli nie uczestniczy w związku (uczestnictwo opcjonalne w związku **Ma**) i dlatego przy encji STUDENT na pierwszym miejscu jest 0.

REG/004 Pokój może mieć przypisanych wielu (max. 6) studentów
tzn. jedna encja POKÓJ może być powiązana z wieloma encjami STUDENT (każda liczba większa od jeden oznacza wiele), a więc przy encji STUDENT na drugim miejscu (jako liczność) jest symbol N. Na rysunku 3.3 nie ma odzwierciedlonego ograniczenia, że w pokoju może być maksymalnie 6 osób.

Interpretacja diagramów przedstawionych na rysunku 3.3 jest następująca:
każdy student **musi** mieć przydzielony **dokładnie jeden** pokój, natomiast w pokoju **może** mieszkać **od 0 do N** studentów, czyli pokój może być zamieszkany przez wielu studentów lub nie być zamieszkany.
Jeśli jest czterech studentów i trzy pokoje, to każdy student musi uczestniczyć w związku dokładnie raz, a pokój może (nie musi) uczestniczyć wiele razy. Na rysunku 3.4a przedstawiono przykładową instancję relacji dla związku **Ma** (każdy związek jest relacją). Na rysunku 3.4b przedstawiono w sposób graficzny powiązania pomiędzy encjami STUDENT i POKÓJ odpowiadające tej instancji relacji.

S1	P1
S2	P1
S3	P1
S4	P2

a)



b)

Rys. 3.4. Graficzne przedstawienie
a) instancji relacji dla związku **Ma** b) powiązań między encjami STUDENT i POKÓJ

Sposób zaznaczania oraz interpretowania liczności i uczestnictwa encji w związkach (a także samych związków) na diagramach nie jest jednoznaczny. Różni autorzy w sposób odmienny definiują te pojęcia. Na przykład:

a) W notacji stosowanej w niniejszej książce (rysunek 3.3a) zaznaczana jest opcjonalność jako 0 i obligatoryjność jako 1, natomiast liczności 1 (jeden) lub N (wiele). Związek **Ma** z rysunku 3.3a odczytujemy następująco: student **musi** mieć przypisany pokój (obligatoryjność występowania encji STUDENT w związku **Ma**) i może mieć przypisany co najwyżej jeden pokój (liczność 1), co oznacza, że student **musi** mieć przypisany dokładnie **jeden** pokój. Natomiast pokój **może** nie mieć przypisanych studentów (opcjonalność występowania encji POKÓJ w związku **Ma**) i może mieć przypisanych wielu studentów (liczność N). Stosowane oznaczenia mogą być interpretowane jako podanie minimalnej i maksymalnej liczności, ale przy definiowaniu związków nie powinno się używać innych oznaczeń na pierwszym miejscu niż 0 lub 1 oraz 1 lub N na miejscu drugim, zatem nie jest poprawny zapis np. 3,7.
b) Notacja wyprowadzona od Chena przez zastąpienie symbolu OPC oznaczającego opcjonalność przez symbol 0 oraz symbolu OBL (obligatoryjność) przez symbol 1 (rysunek 3.3b).
c) Według Barkera [Barker 1996] związek jest obligatoryjny po stronie encji A, jeśli każda instancja encji A może istnieć tylko w ramach powiązania z encją B (z drugiej strony związku). Jeśli encje B mogą istnieć niezależnie od A, to uczestnictwo encji B w związku jest opcjonalne (rysunek 3.3c).
d) Hernandez ([Hernan1998], s. 46, 62), który wykorzystuje metodykę Martina, definiuje uczestnictwo encji w związku jako obowiązkowe i oznacza symbolem „|" przy encji), jeśli encja musi wystąpić, zanim określi się dane encji po drugiej stronie związku. Uczestnictwo opcjonalne w tej notacji zaznacza się symbolem „O" (rysunek 3.3d).
e) W **UML** (*Unified Modelling Language*) liczność nazywa się krotnością ([Muller2000], s. 184, [Subieta1998], s. 231) i na diagramach opisuje się ją przez podanie zbiorów wartości, np.

0..*	zero lub więcej (opcjonalność, wiele)
0..1	zero lub jeden (opcjonalność, jeden)
1..*	co najmniej jeden (obligatoryjność, wiele)
*	zero lub więcej (opcjonalność, wiele)
2..6	co najmniej 2, co najwyżej 6
1, 5-7	jeden, pięć, sześć lub siedem
brak wartości	1..1 (obligatoryjność, jeden)

(rysunek 3.3e).

f) Edward Yourdon ([Yourdon1996], rozdz. 12) uważa, że na diagramach związków-encji (E-R) nie powinno się umieszczać żadnych dodatkowych informacji, takich jak liczność, uczestnictwo czy kierunek związku (niektórzy zaznaczają na diagramach dodatkowo kierunek związku), gdyż takie informacje tylko „odwracają uwagę od podstawowego przeznaczenia diagramu E-R, jakim jest przegląd składników i interfejsów między elementami danych w systemie, a można je łatwo opisać w słowniku danych". Związek w notacji Yourdona przedstawiono na rysunku 3.3f.

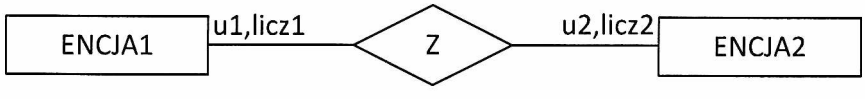
g) Date ([Date2000], s. 398–400) wzorując się na pracy Chena [Chen1976], definiuje uczestnictwo encji E jako obligatoryjne (OBL po stronie encji E, wymagane, ang. *total*) w związku Z, jeśli każda instancja encji E bierze udział w co najmniej jednej instancji związku Z; w przeciwnym razie jest ono opcjonalne (skrót OPC po stronie encji E, ang. *partial*). Notację Chena (rysunek 3.3g) stosuje również Pankowski w pracy [Pankow1992].

W pracy [Olle1982] podano 13 różnych notacji do graficznego przedstawiania encji i związków na diagramach ER – oprócz wyżej podanych są jeszcze np. notacje Bachmana, DeMarco, Jacksona.

■

Definicja typu związku:

ZWI/*xxx* Nazwa związku(ENCJA1(*u1*, *licz1*) : ENCJA2(*u2*, *licz2*); *lista*)



REG/*xxx Tekst reguły1*
REG/*xxx Tekst reguły2*
REG/*xxx Tekst reguły3*
REG/*xxx Tekst reguły4*

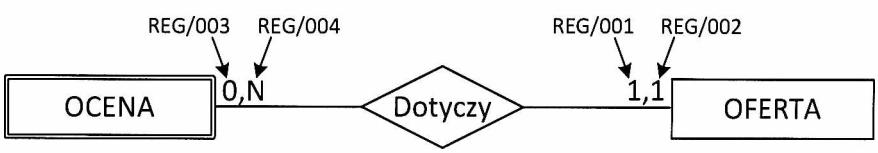
gdzie:
xxx – kolejny numer związku,
u1, *u2* – typ uczestnictwa encji w związku: 0 – opcjonalny, 1 – obligatoryjny,
licz1, *licz2* – liczność encji w związku (1 lub N),
lista – lista atrybutów związku.

Definiowane związki można porządkować alfabetycznie według nazw związków, co ułatwia odszukanie wybranego związku lub w innym wybranym porządku.

Zauważmy, że definicja typu związku obejmuje nazwę związku i uczestniczących typów encji, graficzne przedstawienie związku oraz **cztery** reguły funkcjonowania zdefiniowane w etapie 4 projektu (rozdz. 3.4), z których wynikają typy uczestnictwa i liczności encji w związku. Dla zapewnienia pełnej zgodności reguł w obu etapach zaleca się ich przekopiowywanie. Każda zmiana typu związku wymaga zmiany reguł w obu etapach (etapie 4 i etapie 7.2) i każda zmiana reguł wymaga sprawdzenia poprawności związków i ewentualnej zmiany.

Przykład 3.7 (definicja związku)

ZWI/001 Dotyczy(OCENA(0,N) : OFERTA(1,1))



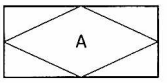
REG/001 Ocena musi dotyczyć oferty
REG/002 Ocena może dotyczyć maksymalnie jednej oferty
REG/003 Oferta nie musi posiadać żadnej oceny
REG/004 Oferta może posiadać wiele ocen

■

Zauważmy, że w przykładzie 3.7 każda encja typu OCENA musi uczestniczyć w związku, czyli musi być powiązana z jakąś encją typu OFERTA (nie ma ocen nie dotyczących ofert), natomiast mogą być encje typu OFERTA, które nie uczestniczą w związku (czyli mogą być oferty bez ocen). Dlatego OCENA jest encją słabą, a OFERTA encją silną. Jest to bardzo często spotykany typ związku **jeden do wielu**, obligatoryjny po stronie jeden – każda encja typu OCENA musi mieć przypisaną dokładnie jedną encję typu OFERTA, a z jedną encją typu OFERTA może (ale nie musi) być związanych wiele encji typu OCENA.

Uwaga. W celu prawidłowego wykonania projektu bazy danych i uproszczenia transformacji modelu konceptualnego do modelu logicznego (rozdz. 4.1) zaleca się **zastępowanie związków typu N:N (wiele do wielu) dwoma związkami typu 1:N**, w wyniku czego pojawia się encja powiązania (asocjacyjna), oznaczana czasami tak

jak na rysunku 3.5. Zaleca się wprowadzenie encji asocjacyjnej także w przypadkach, gdy podczas transformacji (rozdz. 4.1) pojawiają się nowe schematy relacji.

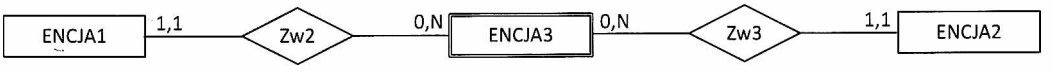


Rys. 3.5. Graficzne przedstawienie encji asocjacyjnej A

W niniejszej książce encję asocjacyjną będziemy oznaczać za pomocą symbolu prostokąta, tak jak inne encje.
Aby wyeliminować związek Zw1 typu N:N przedstawiony na rysunku 3.6a, należy zastąpić go encją asocjacyjną ENCJA3 i dwoma związkami: Zw2 i Zw3 (rysunek 3.6b), a reguły dla związku Zw1 zastąpić nowymi regułami dla związków Zw2 i Zw3.

a)

b)



Rys. 3.6. Graficzne przedstawienie związku wiele do wielu (N:N)
a) przed dekompozycją b) po dekompozycji do dwóch związków 1:N

W wyniku eliminacji związku N:N powstają dwa związki 1:N, z których każdy musi mieć cztery nowe reguły funkcjonowania.

Uwaga. Po eliminacji związku N:N należy przeanalizować i ewentualnie poprawić Etap 1 projektu – cel, zakres, użytkowników – oraz wszystkie kolejne etapy projektu: analizę wycinka rzeczywistości, kategorie, reguły funkcjonowania, ograniczenia dziedzinowe, transakcje, encje, związki itd. Ponadto po każdej transformacji (etap 10, rozdz. 4), w wyniku której powstają nowe schematy relacji, należy poprawić projekt od początku (od etapu 1).

W projektach złożonych, w celu uproszczenia zapisu graficznego, w przypadku gdy związek N:N nie ma dodatkowych atrybutów, można pozostawić go bez zmian i nie zastępować go encją asocjacyjną.

Teoretycznie można wyrysować na kartce wszystkie możliwe kombinacje typów związków, biorąc pod uwagę ich liczność i uczestnictwo. W praktyce niektórych

typów związków nie spotyka się wcale, niektóre rzadko, a niektóre bardzo często. Dokładniejsze omówienie różnych typów związków zamieszczono w rozdz. 4.1, natomiast w tabeli 4.1 w rozdz. 4.1.2 zestawiono wszystkie typy związków.

Zakwalifikowanie danej jako encję, atrybut czy związek nie zawsze jest jednoznaczne. Niektóre dane można uważać za encję, atrybut lub związek. Rozważmy na przykład dane związane z rozwodem:

- rozwód możemy zdefiniować jako encję opisaną przez datę rozwodu, miejsce rozwodu, nazwisko i imię żony, nazwisko i imię męża,
- rozwód może być atrybutem w encji OSOBA (np. tylko data ostatniego rozwodu),
- rozwód może być związkiem encji KOBIETA z encją MĘŻCZYZNA.

Wybór odpowiedniej decyzji, jak zaklasyfikować dane, należy do projektanta bazy danych i wynika nie tylko z jego subiektywnej oceny, intuicji, wiedzy oraz doświadczenia w zakresie projektowania baz danych, ale także z oczekiwań co do szczegółowości i trwałości danych oraz operacji wykonywanych na danych.

Zaleca się, aby wyodrębniane typy encji miały co najmniej dwa atrybuty. Często jednak definiuje się typy encji tylko z jednym atrybutem – kluczem sztucznym. Formalnie mogą być także typy encji bez żadnego atrybutu – w proponowanej metodyce należy wówczas zdefiniować klucz sztuczny.

Pomocą w znalezieniu właściwych atrybutów mogą być wykorzystywane dotychczas formularze papierowe lub inne dostępne druki. Należy je dokładnie przeanalizować, gdyż bardzo często oprócz danych wpisywanych w odpowiednie rubryki na formularzach znajduje się wiele danych dopisywanych dodatkowo poza rubrykami, co może świadczyć o potrzebie wprowadzenia dodatkowych atrybutów dotychczas nieuwzględnionych. Aby określić, do którego typu encji dany atrybut przypisać, należy zadać pytanie „co dany atrybut opisuje?". Często w odpowiedzi na to pytanie kryje się nazwa encji. Dobrym rozwiązaniem jest również uwzględnienie atrybutu *Uwagi*, do zapisywania możliwych dodatkowych danych czy komentarzy.

Ponadto w wielu przypadkach brakuje atrybutów umożliwiających trwałe przechowywanie danych (historii zmian), np. historię wypożyczeń, przebieg kariery zawodowej.

Przeprowadzając analizę rzeczywistości, napotykamy często na dane, które są wyliczane na podstawie zgromadzonych danych, np. średnia ocen studenta, dochód na osobę itp. Powstaje wówczas pytanie, czy daną wielkość lepiej przechowywać w bazie danych jako dodatkowy atrybut, czy wyliczać za każdym razem, kiedy jest potrzebna. Odpowiedź zależy od rodzaju danej wyliczanej, od częstości jej

wyliczania oraz możliwości i częstości zmian. Zasadniczo dane wyliczane nie powinny być przechowywane w bazie, gdyż może to być przyczyną utraty spójności danych, jeśli po zmianie danych cząstkowych (dopisaniu, edycji lub usunięciu) odpowiednie dane wyliczane nie zostaną zaktualizowane.

Po wyodrębnieniu encji i związków możemy przejść do kolejnego etapu – definicji predykatowych typów encji i typu związków.

3.8. DEFINICJE PREDYKATOWE TYPÓW ENCJI I ZWIĄZKÓW

W tym rozdziale omówimy etap 8 projektu, którego celem jest przedstawienie wyodrębnionych typów encji oraz typów związków w sposób zwięzły w ustalonej notacji.

3.8.1. DEFINICJE PREDYKATOWE TYPÓW ENCJI

Definicje predykatowe typów encji przedstawiają w zwarty i czytelny sposób typy encji zdefiniowane w etapie 7.1. Takie zestawienie definicji jest bardzo pomocne podczas opracowywania diagramu związków encji oraz transformacji modelu konceptualnego do modelu logicznego.

Definicje typów encji, szczególnie gdy jest ich dużo, stanowią obszerny materiał, w którym jest wiele ważnych szczegółowych opisów i informacji. Nie wszystkie szczegóły są tak samo istotne na każdym etapie, natomiast wykaz nazw typów encji i ich atrybutów, jest bardzo częstym punktem odniesienia podczas projektowania bazy i weryfikowania jego poprawności.

Na podstawie definicji typów encji z rozdz. 7.1 sporządzamy zestawienie wszystkich typów encji.

W definicji predykatowej typu encji należy podać identyfikator encji, nazwę typu i zapisane w nawiasach atrybuty w kolejności zgodnej z ich wcześniejszym

zdefiniowaniem. Kolejność definicji predykatowych powinna być zgodna z definiowaniem typów encji w rozdz. 7.1.

Definicja predykatowa typu encji:
ENC/*xxx* NAZWA TYPU ENCJI (lista atrybutów)
gdzie *xxx* jest kolejnym trzycyfrowym numerem typu encji.

Przykład 3.8 (definicje predykatowe typów encji)

ENC/001 OSOBA(*IdO, Nazwisko, Imię, DataUr*)
ENC/002 PRZEDMIOT(*NrP, NazwaP*)

■

Klucz główny w definicji predykatowej typu encji, zaznaczony przez podkreślenie odpowiedniego atrybutu, powinien być umieszczony jako pierwszy na liście atrybutów (tak samo jak w definicji kategorii i w typie encji).

3.8.2. DEFINICJE PREDYKATOWE TYPÓW ZWIĄZKÓW

Podobnie jak w przypadku definicji predykatowych typów encji, tak i w przypadku definicji predykatowych typów związków należy wypisać kolejne nazwy typów związków zachodzących między encjami (w takiej samej kolejności jak w etapie 7.2) oraz w nawiasach podać nazwy typów encji, między którymi związek zachodzi, uczestnictwo i liczność oraz (jeśli występują) nazwy atrybutów związku. Nazwy typów związków muszą być unikalne.

Definicja predykatowa typu związku binarnego:
ZWI/*xxx* Nazwa związku(ENCJA1(u1, licz1) : ENCJA2(u2, licz2); lista)
gdzie:

xxx	– kolejny numer związku,
ucz1, ucz2	– typ uczestnictwa encji w związku (0 lub 1),
licz1, licz2	– liczność encji w związku (1 lub N),
lista	– lista atrybutów związku.

Przykład 3.9 (definicje predykatowe typów związków)

ZWI/001 Zna(OSOBA(0,N) : JĘZYK(0,N); *Poziom*)
ZWI/002 Mieszka(OSOBA(0,N) : MIASTO(1,1))
ZWI/003 Pełni(OSOBA(1,1) : ROLA(0,N))

Pełne zestawienie definicji predykatowych typów encji i typów związków w zwarty i czytelny sposób przedstawia statyczną część konceptualnego modelu danych zapisanego w formie tekstowej. Część dynamiczną modelu stanowią definicje transakcji (rozdz. 3.6).

Przykład 3.10 (definicje predykatowe typów encji i związków)

a) **ENC/001 NAUCZYCIEL**(*IdN, Nazwisko, Imie, DataUr, PESEL, Login, Haslo*)
ENC/002 OBSADA(*IdO, LGodz*)
ENC/003 PRZEDMIOT(*IdP, NazwaP*)

ZWI/001 Ma(NAUCZYCIEL(1, 1) : OBSADA(0, N))
ZWI/002 Naucza(OBSADA(0, N) : PRZEDMIOT(1, 1))

b) **8.1. Definicje predykatowe typów encji**
ENC/001 PRODUKT(*IdP, NazwaP, DataDodania, Opis*)
ENC/002 SKLEP(*IdS, NazwaS, Adres, DataDodaniaS, OpisS*)
ENC/003 OFERTA (*IdO, Cena, Ilosc, Jednostka, DataP, DataK*)
ENC/004 KATEGORIA(*IdK, NazwaK*)
ENC/005 OCENA(*IdOc, Ocena, DataOc, Komentarz*)

8.2. Definicje predykatowe typów związków
ZWI/001 Ma(PRODUKT(0,N) : KATEGORIA(1,1))
ZWI/002 Dotyczy(OCENA(0,N) : OFERTA(1,1))
ZWI/003 Obejmuje(OFERTA(O,N) : PRODUKT(1,1))
ZWI/004 Oferuje(SKLEP(1,1) : OFERTA(0,N))

Po wykonaniu etapu 8 można przejść do etapu 9 i opracować **diagram związków encji**, czyli przedstawić model konceptualny w postaci graficznej.

3.9. DIAGRAM ZWIĄZKÓW ENCJI

Wszelkie modele zyskują przy ich graficznym przedstawieniu. W metodyce strukturalnej model konceptualny przedstawiany jest w postaci graficznej jako diagram związków encji.

Diagram związków encji (ang. *Entity Relationship Diagram* – ERD) jest graficznym przedstawieniem w ustalonej notacji modelu konceptualnego danych, czyli danych reprezentowanych przez typy encji oraz związków między nimi [Barker1996, Date2000, Rodgers1995].

W niektórych notacjach na diagramach ER (związków encji) są wyszczególniane dodatkowo atrybuty i klucze encji oraz atrybuty związków, ale takie podejście zmniejsza czytelność diagramów, szczególnie w przypadku dużych projektów. W omawianej metodyce uzupełnieniem diagramów są definicje predykatowe encji i związków, z których wynikają atrybuty encji i związków.

Każda encja na diagramie musi być powiązana z co najmniej jedną encją innego typu (z wyjątkiem diagramu zawierającego tylko jedną encję).

Na diagramie muszą być umieszczone wszystkie zdefiniowane typy encji i związków.

Analiza poprawności modelu danych za pomocą definicji predykatowych encji i związków (tekstowego zapisu) jest trudna. Dopiero graficzny zapis w postaci ERD pozwala zauważyć popełnione błędy projektowe i wynikające z tego pewne anomalie. Należy np. zwrócić uwagę, czy nie zdefiniowano zbyt dużo związków z obligatoryjnym uczestnictwem encji, czy nie ma niepoprawnych cykli, które mogą być przyczyną niespójności danych, źle zdefiniowanych związków referencyjnych (czyli między encjami tego samego typu). Sytuacja, gdy dwie encje połączone związkami w sposób pośredni są także połączone w sposób bezpośredni, może być poprawna lub wynikać ze źle zdefiniowanych związków. Aby projekt był poprawny, należy zwrócić uwagę na spójność danych i poprawność reguł funkcjonowania i definicje związków.

Opracowując graficznie diagram związków encji, należy kierować się zasadą, aby uzyskany diagram był poprawny, a jednocześnie czytelny. W tym celu należy (w miarę możliwości) wyrównywać w poziomie prostokąty symbolizujące encje, linie związków prowadzić poziomo lub pionowo, ewentualnie załamywać pod kątem prostym lub 45 stopni. Jeśli linie muszą się przecinać, to najlepiej pod kątem 30–60 stopni. Nie należy rysować zbyt wielu linii równoległych blisko siebie. Należy

zostawiać odpowiednio dużo wolnego miejsca w celu zwiększenia czytelności diagramu. Nazwy encji umieszczane w prostokątach powinny być w obrębie całego diagramu jednakowo sformatowane, najlepiej wyśrodkowane. Nie należy wprowadzać żadnych dodatkowych oznaczeń, które nie wynikają z przyjętej notacji. W szczególności nie należy wprowadzać kolorów, tła, nawiasów itp. Wszelkie odstępstwa od ustalonej notacji sprawiają, że diagram staje się niejednoznaczny i nieczytelny i trudno zweryfikować jego poprawność.

Ze względu na zmiany dokonywane w diagramach w trakcie projektowania bazy danych przez zespół projektowy dobrym zwyczajem jest opisywanie diagramu danymi: autor i tytuł diagramu, data opracowania diagramu, opis zmian itp.

Diagramy można odczytywać zgodnie z przyjętą kolejnością czytania tekstów, a więc od lewej do prawej, z góry na dół. Dlatego dobrze jest umieszczać encje o dużym znaczeniu w lewym górnym rogu, ale w zależności od projektu czytelniejsze może być inne podejście, na przykład umieszczenie ich w środku diagramu. Wielkość prostokątów dla oznaczania encji może być różna, ale zachowanie regularnego wyglądu poprawia czytelność.

Jeśli w projektowanej bazie dane użytkowników mają być w niej przechowywane, to należy zwrócić uwagę, czy na ERD one występują.

Poprawiając projekt, np. dodając nowy typ encji, należy zwrócić uwagę, aby miał co najmniej jeden atrybut identyfikujący encję (klucz główny), występował w co najmniej jednym związku z innymi encjami i była dla niego zdefiniowana co najmniej jedna transakcja.

Przykład 3.11

Załóżmy, że projektujemy bazę danych **WYPOŻYCZALNIA** dla wypożyczalni kaset wideo i w wyniku analizy rzeczywistości wyodrębniliśmy cztery typy encji: KLIENT, KASETA, FILM, GATUNEK oraz trzy związki: **Zawiera**, **Jest**, **Wypożycza**.

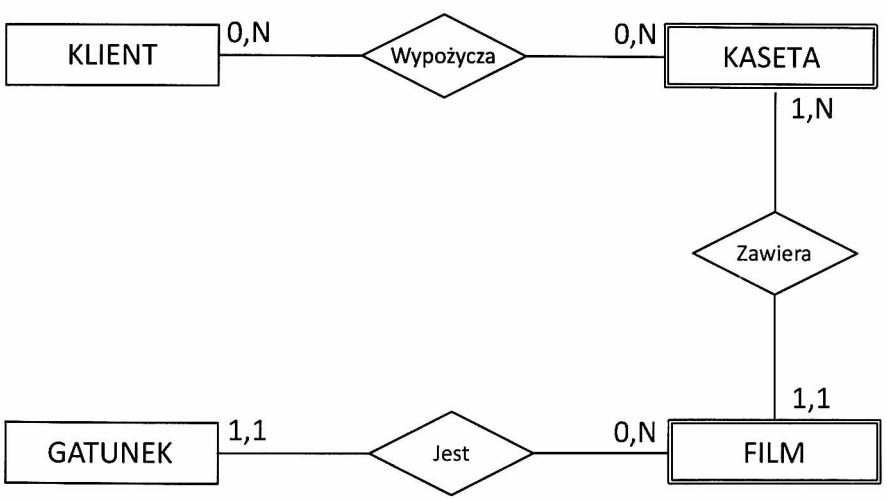
Definicje predykatowe wyodrębnionych typów encji i związków mają następującą postać:

ENC/001 KLIENT(*NrKlienta*, *Nazwisko*, *Imię*, *Miasto*, *UlicaNr*)
ENC/002 KASETA(*NrKasety*, *Opis*)
ENC/003 FILM(*KodF*, *Tytuł*, *CzasTrwania*)
ENC/004 GATUNEK (*KodG*, *NazwaG*)
ZWI/001 Zawiera(KASETA(1,N) : FILM(1,1))

ZWI/002 Jest(FILM(0,N) : GATUNEK(1,1))
ZWI/003 Wypożycza(KLIENT(0,N) : KASETA(0,N); *NrW*, *DataW*, *DataZw*, *Opłata*)

Zwróćmy uwagę, że związek **Wypożycza**, jest związkiem obustronnie opcjonalnym wiele do wielu (jeden klient może wypożyczyć wiele kaset lub żadną, kaseta z wypożyczalni może być niewypożyczona dotychczas ani razu lub mogła być wypożyczana przez wielu różnych klientów).

Na rysunku 3.7 przedstawiono diagram związków encji dla bazy danych **WYPOŻYCZALNIA**. Encja FILM jest zaznaczona jako słaba, ponieważ jest zależna od encji silnej GATUNEK (nie może istnieć bez niej), encja KASETA jest również zaznaczona jako słaba, ponieważ jest zależna od encji FILM.



Rys. 3.7. Niepoprawny diagram związków encji dla bazy WYPOŻYCZALNIA

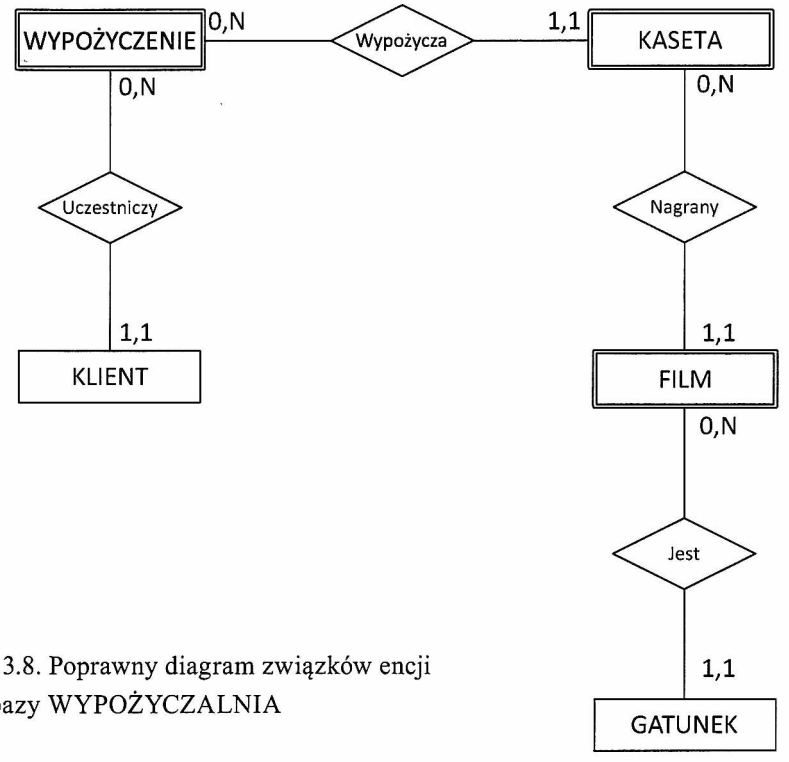
Zgodnie z uwagą zawartą w rozdz. 3.7.2, zalecającą zastępowanie związków N:N dwoma związkami 1:N, należy zastąpić związek **Wypożycza** z rysunku 3.7 encją WYPOŻYCZENIE (nazywaną *encją powiązania* lub *encją asocjacyjną*) i ponownie narysować diagram (i oczywiście poprawić wszystkie wcześniejsze etapy projektu). Ponadto na diagramie 3.7 występuje niemożliwy do realizacji związek **Zawiera**(KASETA(1,N) : FILM(1,1)), który powinien zostać zastąpiony związkiem **Nagrany**(KASETA(0,N) : FILM(1,1)).

Czytelników zaintrygowanych związkiem **Zawiera**, który jest obligatoryjny zarówno po stronie encji FILM, jak i encji KASETA, odsyłamy do rozdz. 4.1.3, gdzie jest przeprowadzona dokładna analiza tego typu związków.

Po zmianach dla projektowanej bazy danych **WYPOŻYCZALNIA** otrzymamy następujące definicje predykatowe typów encji i związków:

ENC/001 KLIENT(*NrKlienta*, *Nazwisko*, *Imię*, *Miasto*, *UlicaNr*)
ENC/002 KASETA(*NrKasety*, *Opis*)
ENC/003 FILM(*KodF*, *Tytuł*, *CzasTrwania*)
ENC/004 GATUNEK(*KodG*, *NazwaG*)
ENC/005 WYPOŻYCZENIE(*NrW*, *DataW*, *DataZw*, *Opłata*)
ZWI/001 Nagrany(KASETA(0,N) : FILM(1,1))
ZWI/002 Jest(FILM(0,N) : GATUNEK(1,1))
ZWI/003 Uczestniczy(KLIENT(1,1) : WYPOŻYCZENIE(0,N))
ZWI/004 Dotyczy(WYPOŻYCZENIE (0,N) : KASETA(1,1))

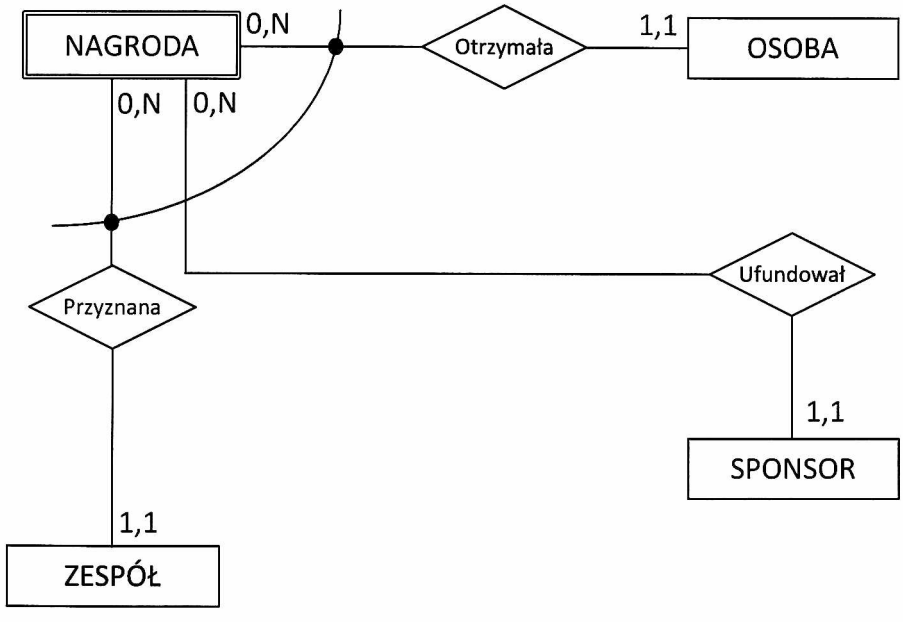
Na rysunku 3.8 przedstawiono poprawny diagram związków encji dla bazy **WYPOŻYCZALNIA**.



Rys. 3.8. Poprawny diagram związków encji dla bazy WYPOŻYCZALNIA

Związki typu *wiele do wielu* powinny być eliminowane z diagramu przez zastąpienie ich encją asocjacyjną o nazwie wyprowadzonej z nazwy związku, np. ze związku **Wypożyczenie** powstała encja WYPOŻYCZENIE. Taka zmiana wymaga naniesienia zmian w wielu etapach projektu, dlatego zaleca się przeanalizowanie całego projektu, począwszy od etapu 1.

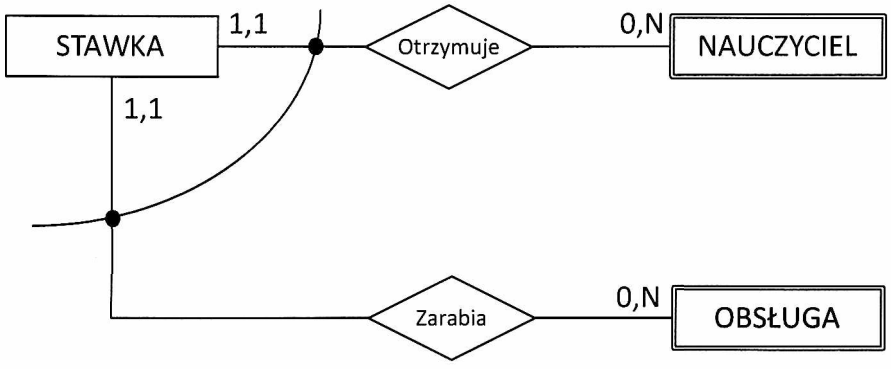
Często zachodzi potrzeba przedstawienia na diagramach sytuacji powiązania encji typu A z encją typu B1 **albo** z encją typu B2 (ale tylko z jedną z nich). Można to wyrazić na diagramie, prowadząc łuk wykluczający przez linie związków i zaznaczając punkty przecięcia z właściwymi związkami czarnymi kropkami, tak jak przedstawiono to na rysunku 3.9.



Rys. 3.9. Diagram związków encji z łukiem wykluczającym po stronie *wiele* związku

Nagroda (opisywana przez atrybuty *TytułNagrody*, *RokPrzyznania*) jest przyznana **albo** jako indywidualna dla jednej osoby (encja OSOBA o atrybutach *Nazwisko*, *Imię*, *DaneAdresowe*), **albo** dla zespołu (encja ZESPÓŁ o atrybutach *NazwaZespołu*, *LiczbaOsóbWZespole*, *DataPowstania*). Związek **Ufundował** nie jest częścią łuku wykluczającego. Obowiązuje zasada, że dany związek może należeć tylko do jednego łuku wykluczającego. Zwykle łuki rysuje się przy końcach związków po stronie *wiele*. Końce związków należące do łuku wykluczającego muszą być wszystkie opcjonalne albo wszystkie obligatoryjne.

Na rysunku 3.10 przedstawiono diagram, w którym łuk wykluczający jest poprowadzony po stronie *jeden* związku. Jest to przypadek poprawny, który umożliwia zmiany w stawkach zarobkowych dla pracowników będących nauczycielami i pozostałego personelu. Dana kategoria zarobkowa jest przypisana albo do nauczycieli, albo do obsługi (pozostałych pracowników).



Rys. 3.10. Diagram związków encji z łukiem wykluczającym po stronie jeden związku

Czasami w projekcie istnieje potrzeba precyzyjnego odzwierciedlenia ograniczenia dotyczącego maksymalnej liczności danego końca związku. Ze związku przedstawionego na rysunku 3.11 wynika, że klient może nie mieć konta w banku albo może mieć wiele kont, ale konto musi należeć co najmniej do jednej osoby, a może należeć maksymalnie do dwóch osób.

Rys. 3.11. Związek **Ma** z ograniczoną do dwóch licznością występowania encji typu KLIENT

Zgodnie z notacją przyjętą w tej książce dla oznaczania uczestnictwa i liczności encji w związku (które można interpretować jako podanie minimalnej i maksymalnej liczności dla każdego końca związku), związek **Ma** z rysunku 3.11 odczytujemy następująco: klient **może** założyć w banku od **0** (opcjonalność występowania encji **KLIENT**) do N kont, konto **musi** należeć do **1** (obligatoryjność występowania encji **KONTO** w związku) lub do 2 osób. Pamiętajmy jednak, że liczność 2 (i każda

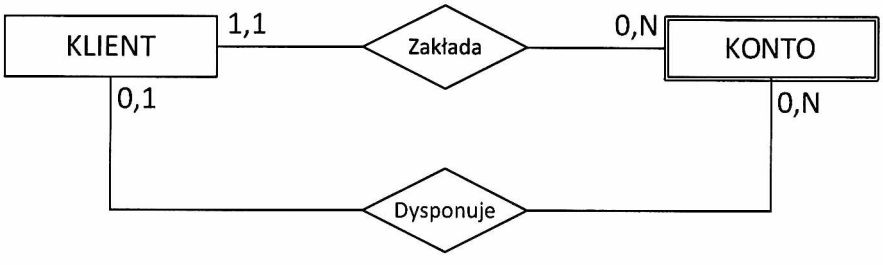
większa od 1) jest traktowana jak *wiele* (N), czyli na rysunku 3.11 przedstawiony jest związek **Ma**(KLIENT(1,N) : KONTO(0:N).

Zatem w poprawnym projekcie uwzględniającym ograniczenie, że konto musi należeć co najmniej do jednej osoby, a maksymalnie do dwóch, związek **Ma**(KLIENT(1,2) : KONTO(0:N) przedstawiony na rysunku 3.11 należy zastąpić dwoma związkami:

Zakłada(KLIENT(1,1) : KONTO(0:N))

Dysponuje(KLIENT(0,1) : KONTO(0:N)),

z których jeden związek musi być obligatoryjny, a drugi opcjonalny po stronie 1, co graficznie przedstawiono na rysunku 3.12.



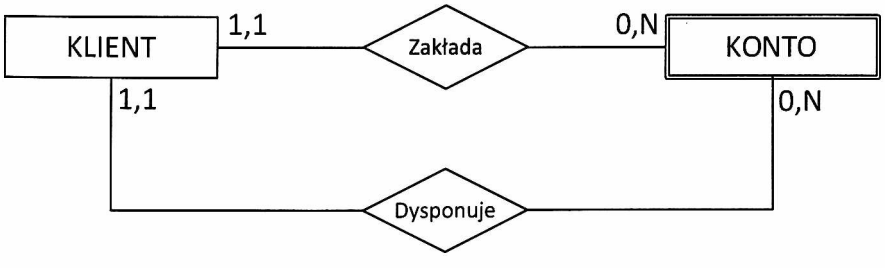
Rys. 3.12. Dwa związki 1:N pomiędzy dwiema encjami (obligatoryjny i opcjonalny po stronie 1)

Jeśli natomiast konto musi należeć dokładnie do dwóch osób, to należy zdefiniować dwa związki:

Zakłada(KLIENT(1,1) : KONTO(0:N))

Dysponuje(KLIENT(1,1) : KONTO(0:N))

czyli oba związki muszą być obligatoryjne po stronie 1 (rysunek 3.13).



Rys. 3.13. Dwa związki 1:N pomiędzy dwiema encjami, obligatoryjne po stronie 1

Decyzja, jakiego typu związki powinny być na ERD – z rysunku 3.12 czy z 3.13, zależy od analizowanej rzeczywistości.

Jeśli dla projektowanej bazy danych jest wymaganie, że dwie drużyny muszą grać w meczu piłkarskim, to należy utworzyć dwa związki, np. **Gra**, **Towarzyszy**, zachodzące pomiędzy encjami typu MECZ i DRUŻYNA. Decyzja o typie związku zależy o przeznaczenia bazy, kto i kiedy wprowadza dane oraz w jakim celu. Jeśli baza będzie wykorzystywana do ewidencji wyników, a więc dane są wprowadzane do bazy po meczu, to oba związki mogą być obligatoryjne po stronie *jeden* (do meczu muszą być przypisane dwie drużyny):

Gra(MECZ(0,N) : DRUŻYNA(1:1))

Towarzyszy(MECZ(0,N) : DRUŻYNA(1:1)).

Jeśli natomiast dane gromadzone w bazie mają wspomagać organizowanie meczu, czyli najpierw jest planowany mecz, potem przypisywana jest jedna drużyna, a po pewnym czasie dopiero druga, to oba związki powinny być opcjonalne po stronie *jeden*:

Gra(MECZ(0,N) : DRUŻYNA(0:1))

Towarzyszy(MECZ(0,N) : DRUŻYNA(0:1)).

Oczywiście możliwa jest także sytuacja, że mecz jest planowany tylko wtedy, gdy jest chociaż jedna drużyna chętna do gry (np. jako organizator meczu) i wówczas jeden związek jest opcjonalny, a drugi obligatoryjny po stronie *jeden*:

Gra(MECZ(0,N) : DRUŻYNA(1:1))

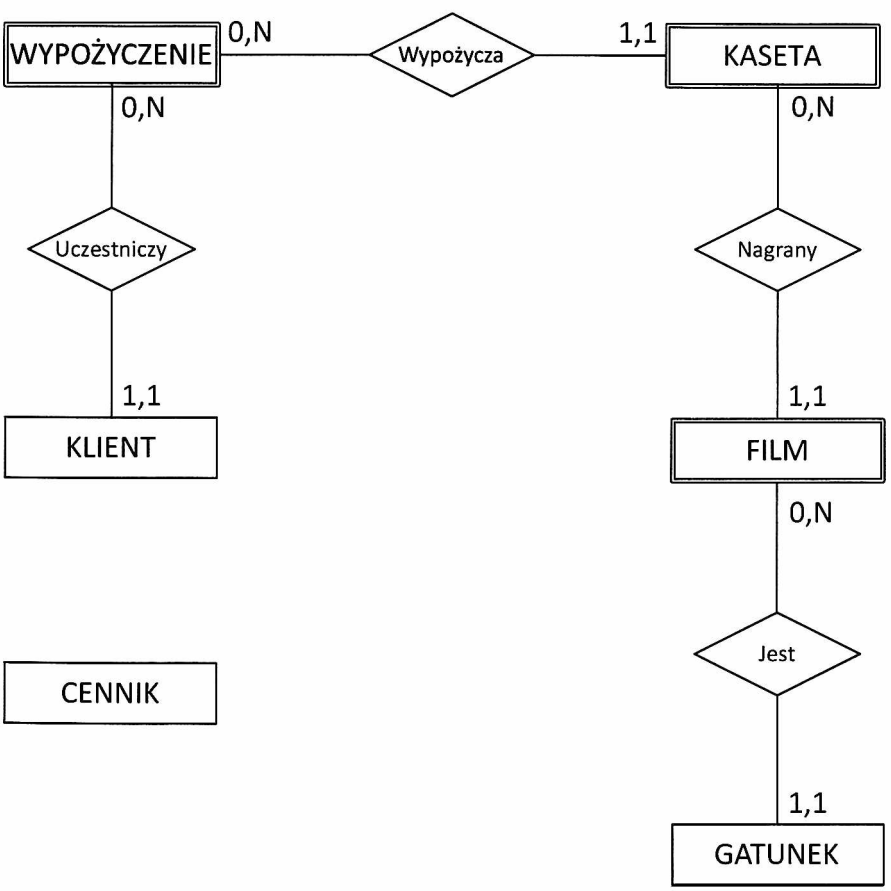
Towarzyszy(MECZ(0,N) : DRUŻYNA(0:1)).

Decyzję o wyborze typu związków musi podjąć projektant.

Aby sprawdzić poprawność diagramu pod kątem kompletności danych, należy dla każdego typu encji występującego na diagramie zweryfikować jego atrybuty oraz udział w transakcjach (operacjach na danych), czyli prześledzić cykl życia każdej encji, jakie dane są potrzebne podczas wykonywania operacji, które dane trzeba przechowywać i jak długo. Na przykład klient najpierw jest zapisywany do wypożyczalni, potem może wypożyczać kasety, może je zwracać, można modyfikować jego dane (np. zmienić adres, telefon, nazwisko), można sprawdzić, które kasety dotychczas wypożyczył, których kaset nie oddał, jak długo je przetrzymywał, czy ich nie uszkodził, ile zapłacił za wypożyczenia, czy korzystał z rabatów, można usunąć jego dane na jego prośbę (jeśli oddał wszystkie kasety lub zapłacił karę za kasety nieoddane), wysłać e-mail przypominający o przekroczeniu terminu oddania kasety, a po 10 latach bez żadnych aktywności można usunąć jego dane z bazy. Na każdym etapie należy sprawdzać, czy dzięki gromadzonym danym te operacje będą

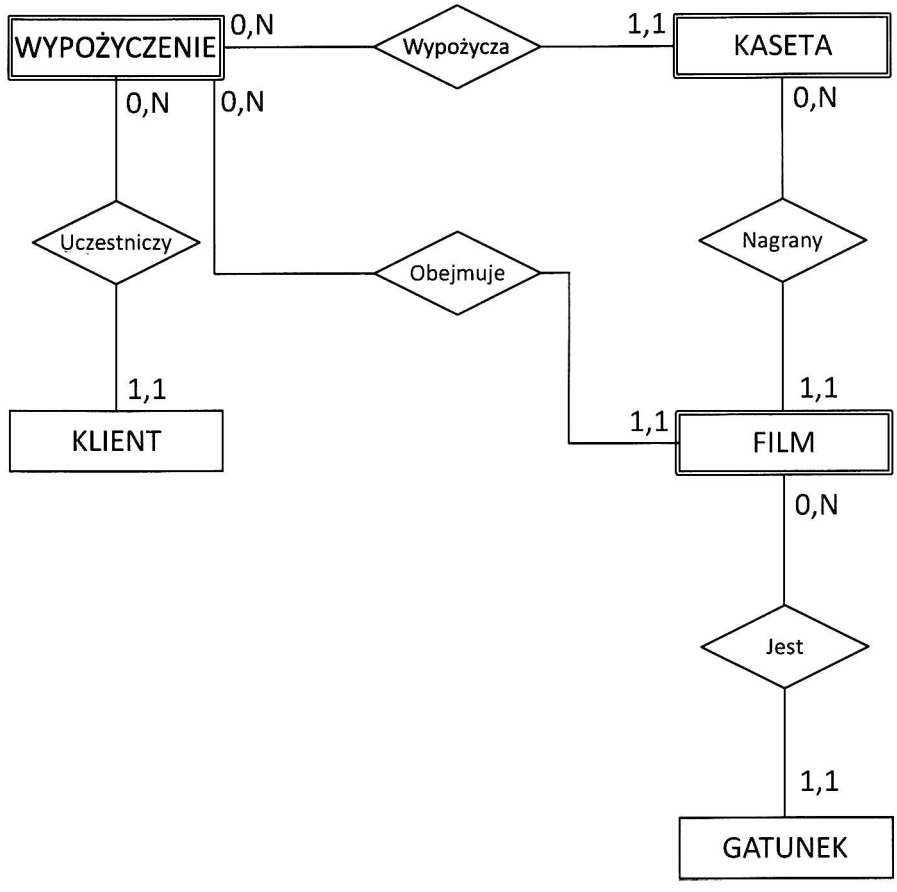
możliwe do wykonania. Być może w opracowanym projekcie np. nie uwzględniono kosztów wypożyczenia lub możliwości adnotacji przyczyny nieoddania kasety (uległa zniszczeniu, klient zmarł) i należy dodać odpowiednie atrybuty lub encje. W transakcjach (etap 6, rozdz. 3.6) każda operacja musi być precyzyjnie zaplanowana, np. czy usunięcie danych wypożyczającego polega na fizycznym usunięciu jego danych i danych wypożyczeń, czy dane są anonimizowane, dzięki czemu można wykorzystywać je do sporządzania statystyk i opracowywania raportów.

Zgodnie z zasadą, że każda encja występująca na diagramie musi uczestniczyć w co najmniej jednym związku, diagram przedstawiony na rysunku 3.14 należy uznać za niewłaściwy do zaprojektowania bazy danych, ponieważ encja **CENNIK** nie występuje w związku z żadną inną encją. Zdefiniowane encje i związki muszą być graficznie zapisane w postaci jednego diagramu, a nie kilku rozłącznych.



Rys. 3.14. Błędny diagram związków encji zawierający niepowiązaną encję CENNIK

Poprawny diagram związków encji nie powinien zawierać zbędnych związków, które mogą być wyprowadzone z innych związków, np. na diagramie przedstawionym na rysunku 3.15 związek **Obejmuje** między encjami WYPOŻYCZENIE i FILM jest zbędny, ponieważ może być wyprowadzony ze związku **Dotyczy**. Pozostawienie go prowadziłoby do utraty spójności danych (można przypisać do danego wypożyczenia film z innej kasety niż ta, która go zawiera).



Rys. 3.15. Błędny diagram związków encji zawierający zbędny związek **Obejmuje**

Jako ćwiczenie proponujemy opracowanie diagramu związków encji przedstawiającego przynależność pracowników uczelni wyższej do wydziału, instytutu, zakładu, katedry według reguł obowiązujących w danej uczelni.

Na zakończenie tego rozdziału podkreślmy, że przy projektowaniu bazy danych prawidłowa identyfikacja encji i związków powinna doprowadzić do utworzenia

schematów relacji będących w trzeciej postaci normalnej (rozdz. 2.3). Aby upewnić się, że w danym modelu konceptualnym prawidłowo wyodrębniliśmy encje i związki, należy sprawdzić:

- czy w encjach lub związkach nie ma powtarzających się atrybutów lub grup atrybutów; występowanie atrybutów postaci:

  ImięDziecka_1, *ImięDziecka_2*, *ImięDziecka_3*,
  DataUrDziecka_1, *DataUrDziecka_2*, *DataUrDziecka_3*,

  jest sygnałem, że należy utworzyć nową encję DZIECKO, która będzie w pierwszej postaci normalnej,

- czy nie ma atrybutów, które zależą tylko od części klucza głównego, jeśli tak jest, to ten atrybut i ta część klucza, od której atrybut zależy, jest podstawą do utworzenia nowego typu encji, która będzie w drugiej postaci normalnej, np. źle zaprojektowaną encję (nie jest w 2PN)

  ZAJĘCIA(*IdP*, *NazwaP*, *NazwiskoN*, *Email*),

  gdzie nauczyciel może prowadzić wiele przedmiotów, przedmiot może być prowadzony przez wielu nauczycieli i atrybut *Email* jest funkcyjnie zależny tylko od atrybutu *NazwiskoN*, a *NazwaP* zależy tylko od *IdP*, można zastąpić dwiema encjami, które są w drugiej postaci normalnej:

  NAUCZYCIEL(*IdN*, *NazwiskoN*, *Email*)
  PRZEDMIOT(*IdP*, *NazwaP*),

  ale wówczas encja NAUCZYCIEL jest w związku N:N z encją PRZEDMIOT, który określa zajęcia prowadzone przez nauczycieli. Aby tego uniknąć, należy zdefiniować trzy encje:

  NAUCZYCIEL(*IdN*, *NazwiskoN*, *Email*)
  PRZEDMIOT(*IdP*, *NazwaP*)
  ZAJĘCIA(*IdZ*, *IleGodz*)

- czy nie ma atrybutów zależnych od atrybutów, które nie są częścią klucza głównego, jeśli tak jest, to te atrybuty są podstawą do utworzenia nowej encji pozostającej w związku jeden do wielu z encją wyjściową (3PN), np. w encji

  UCZEŃ(*NrUcznia*, *Nazwisko*, *Klasa*, *Profil*),

  atrybut *Profil* zależy funkcyjnie od atrybutu *Klasa*, który nie jest częścią klucza głównego, dlatego należy utworzyć dwie encje

  UCZEŃ(*NrUcznia*, *Nazwisko*)
  KLASA(*Symbol*, *Profil*),

  przy czym encja KLASA będzie w związku 1:N z encją UCZEŃ.

Pamiętajmy także, aby po każdej korekcie diagramu związków encji sprawdzić projekt od początku i nanieść wszystkie konieczne poprawki w celu zapewnienia spójności i poprawności projektu jako całości.

Opracowanie diagramu związków encji kończy etap modelowania konceptualnego bazy danych.

Diagram związków encji w połączeniu z definicjami predykatowymi typów encji i typów związków umożliwia szybką, niemalże mechaniczną transformację modelu konceptualnego do modelu logicznego omówioną w rozdziale 4.

Konstruktor wie, iż osiągnął doskonałość nie wtedy, gdy nic już nie może dodać, lecz wtedy, gdy nic już nie da się ująć.

Antoine de Saint-Exupery

4. MODEL LOGICZNY

Zaprojektowanie modelu logicznego bazy danych wymaga wykonania transformacji opracowanego modelu konceptualnego bazy do schematu relacyjnego bazy danych zawierającego schematy relacji oraz tak wiele szczegółów dotyczących danych, jak to tylko możliwe. Każdy schemat relacji definiuje relacje (rozdz. 2.1), które można reprezentować w postaci tabel. Na etapie projektowania logicznego powinno się sprawdzić, czy utworzone schematy relacji są przynajmniej w trzeciej postaci normalnej, a jeśli nie są, to należy przeprowadzić ich normalizację. Jak już wspomnieliśmy wcześniej (rozdz. 2.3), normalizacja danych jest sformalizowaną procedurą, w wyniku której eliminuje się redundancje (powtarzanie się) danych, ich niespójności i anomalie oraz wszelkie nieprawidłowości występujące w bazie danych związane z wykonywanymi operacjami na danych takimi jak aktualizowanie, usuwanie i dopisywanie danych.

Projektowanie logiczne bazy danych ma na celu opracowanie schematu bazy danych, z którego wynika struktura danych i powiązania między nimi.

Otrzymany w rozdz. 3 konceptualny model danych jest zdefiniowany przez definicje typów encji i związków oraz definicje transakcji, a ponadto jest reprezentowany graficznie przez diagram związków encji (rozdz. 3.9).

Transformacja modelu konceptualnego bazy danych do modelu logicznego jest wykonywana według pewnych reguł, które są opisane w rozdziałach 4.1.1–4.1.4. W wyniku transformacji otrzymujemy schematy relacji, których definiowanie

omówiono w rozdziale 4.2. Model logiczny bazy danych można zwięźle przedstawić w postaci schematu bazy danych (rozdz. 4.3), którego dopełnieniem jest słownik atrybutów (rozdz. 4.4). W kolejnym (i zarazem ostatnim) etapie projektowania logicznego bazy danych należy dla każdej zdefiniowanej grupy użytkowników określić perspektywy (rozdz. 4.5).

4.1. TRANSFORMACJA MODELU KONCEPTUALNEGO DO MODELU LOGICZNEGO

W rozdziale tym omówimy reguły ogólne oraz szczególne przypadki transformacji modelu konceptualnego do relacyjnego modelu logicznego będącego zbiorem struktur danych reprezentowanych przez relacje. Podamy i omówimy wiele przykładów ilustrujących prezentowane zagadnienia. Jeśli typy encji i typy związków między nimi zostały zdefiniowane poprawnie, to na ogół otrzymane po transformacji schematy relacji są w trzeciej postaci normalnej.

4.1.1. REGUŁY TRANSFORMACJI

Transformację modelu konceptualnego do modelu logicznego przeprowadzamy według podanych reguł.

Reguły transformacji.

1. Dla każdego typu encji tworzymy schemat relacji. Najczęściej nazwa schematu relacji (i relacji) jest taka sama jak typu encji, tylko w liczbie mnogiej, i nie jest pisana wielkimi literami.
2. Atrybuty typu encji stają się atrybutami w schemacie relacji (o takich samych nazwach jak odpowiadające im atrybuty w typach encji).
3. Atrybuty odpowiadające kluczom głównym typów encji stają się kluczami głównymi schematów relacji.

4. Atrybuty, które odpowiadają kluczom kandydującym w typach encji, są kluczami kandydującymi w utworzonych schematach relacji (są atrybutami obligatoryjnymi o unikalnych wartościach).
5. Atrybuty opcjonalne są atrybutami o dopuszczalnych wartościach NULL, atrybuty obligatoryjne (wymagane) – NOT NULL.
6. Dla każdego związku binarnego *jeden do wielu*, obligatoryjnego po stronie *jeden*, wstawiamy klucz główny ze strony *jeden* do schematu relacji reprezentującego stronę *wiele* związku. W ten sposób otrzymujemy **klucz obcy** (ang. *foreign key*) w schemacie relacji ze strony *wiele* związku (NOT NULL).
7. Związek binarny typu *jeden do wielu*, opcjonalny po obu stronach, reprezentujemy zazwyczaj nowym schematem relacji, w którym umieszczamy klucze główne obu encji (ze związku opcjonalnego po stronie jeden otrzymujemy kolumny NULL). Jeżeli co najmniej 60% instancji encji pozostaje w związku, to uznaje się związek za prawie obligatoryjny i zaleca się zastosować transformację według reguły 6, z tym że klucz obcy w schemacie po stronie *wiele* jest atrybutem opcjonalnym.
8. Dla związku opcjonalnego *wiele do wielu* tworzymy nowy schemat relacji, ewentualnie ze sztucznym kluczem głównym. Klucze główne z typów encji po obu stronach związku stają się kluczami obcymi w utworzonym schemacie relacji. Jeśli związek posiadał atrybuty dodatkowe, to stają się one atrybutami w nowo utworzonym schemacie relacji. Kluczem głównym może być klucz złożony z obu kluczy obcych lub nowo wprowadzony klucz sztuczny.
9. Jeśli w wyniku transformacji w utworzonym schemacie relacji pojawia się naturalny klucz kandydujący, a schemat posiada sztuczny klucz główny, to można zbędny klucz sztuczny usunąć.
10. W utworzonych schematach relacji klucze główne zaznaczamy przez podkreślenie nazw atrybutów, a nazwy atrybutów, które są kluczami obcymi, poprzedzamy symbolem #.

Zauważmy, że podczas transformacji po raz pierwszy w projekcie pojawiają się atrybuty łączące z sobą schematy relacji i zapewniające integralność referencyjną, tzw. **klucze obce**.

Jeżeli w wyniku transformacji modelu konceptualnego do modelu logicznego powstanie nowy schemat relacji, to nie odpowiada mu żadna kategoria ani typ encji, a często także nie ma dla niego zdefiniowanych transakcji. Z tego względu oraz dla zachowania zgodności otrzymanego schematu bazy danych co do liczby i nazw schematów relacji z liczbą i nazwami kategorii i typów encji należy przeanalizować

i poprawić wykonany projekt od etapu 1. Należy zwrócić uwagę na zmiany konieczne we **wszystkich etapach**, m.in. nowe kategorie, reguły funkcjonowania i ograniczenia dziedzinowe, transakcje dla nowo utworzonych kategorii, typy encji i związki między nimi oraz diagram związków encji.

Przedstawione reguły ogólne transformacji na ogół dają dobre efekty, ponieważ uwzględniają dwa typy związków spotykane najczęściej:

Zw1(A(1,1) : B(0,N))

oraz

Zw2(A(0,N) : B(0,N)).

W szczególnych przypadkach podane zasady transformacji mogą się jednak okazać niewystarczające, ponieważ nie wszystkie uwzględniają typ uczestnictwa encji w związku (obligatoryjność / opcjonalność).

4.1.2. TRANSFORMACJA ZWIĄZKÓW

Przeanalizujemy dokładnie różne typy związków (rzadko lub często spotykanych oraz niemożliwych do realizacji), uwzględniając liczność i uczestnictwo encji w związku. W tabeli 4.1 (rozdz. 4.1.2) zestawiono wszystkie typy związków.

1. Dla związku binarnego ***jeden do jeden*** mogą zachodzić trzy przypadki:

a. Zw(A(1,1) : B(1,1))

Rys. 4.1. Graficzna reprezentacja związku Zw(A(1,1) : B(1,1))

Dla związku binarnego typu *jeden do jeden,* obligatoryjnego po obu stronach:

- Łączymy schemat relacji **RA** powstały z typu encji A oraz schemat **RB** powstały z typu encji B w jeden schemat, który zawiera atrybuty z obu schematów relacji.
- W utworzonym schemacie relacji oba klucze główne (z encji typu A i z encji typu B) są kluczami kandydującymi i jeden z nich (dowolny) wybieramy na klucz główny utworzonego schematu relacji.

Przykład 4.1

Studenci danego kierunku studiów mają jeden i tylko jeden indeks, indeks należy dokładnie do jednego studenta. Reguły te odzwierciedla związek **Ma** zachodzący między encjami typu STUDENT i INDEKS:

STUDENT(*NrS*, *Nazwisko*, *Imię*, *PESEL*, *DataUr*)
INDEKS(*NrIndeksu*, *DataWyd*)
Ma(STUDENT(1,1) : INDEKS(1,1))

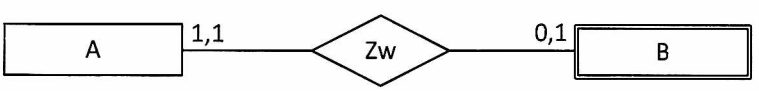
W wyniku transformacji powstanie jeden schemat relacji:

Studenci(*NrS*, *Nazwisko*, *Imię*, *PESEL*, *DataUr*, *NrIndeksu*, *DataWyd*)

W schemacie **Studenci** są trzy klucze kandydujące: *NrS*, *NrIndeksu*, *PESEL*, a na klucz główny wybrano *NrS*, przy czym *NrIndeksu* też może być kluczem głównym, natomiast *PESEL* nie, bo jest daną chronioną.

■

b. Zw(A(1,1) : B(0,1))



Rys. 4.2. Graficzna reprezentacja związku **Zw**(A(1,1) : B(0,1))

Związek binarny typu *jeden do jeden*, obligatoryjno-opcjonalny przekształcamy następująco:

- Dla typów encji A i B tworzymy schematy relacji **RA** i **RB**.
- Do schematu **RB** wstawiamy jako dodatkowy atrybut (klucz obcy) klucz główny ze schematu **RA**.
- Klucz obcy jest atrybutem obligatoryjnym i unikalnym, czyli jest kluczem kandydującym.
- Kluczem głównym w schemacie relacji **RB** może być klucz główny z typu encji A lub z typu encji B.

Przykład 4.2

Omawianą sytuację ilustruje związek **Ma** między typami encji: OSOBA (encja typu A) i NUMER (encja typu B):

OSOBA(*NrO*, *Nazwisko*, *Imię*, *DataUr*, *PESEL*)

NUMER(*NIP*, *NrWpisu*, *DataWpisu*, *Miasto*)

Ma(OSOBA(1,1) : NUMER(0,1))

Osoba może mieć (nie musi) co najwyżej jeden numer NIP, a dany numer musi mieć przypisaną dokładnie jedną osobę. W wyniku transformacji powstaną następujące schematy relacji:

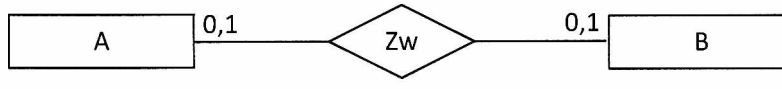
Osoby(*NrO*, *Nazwisko*, *Imię*, *DataUr*, *PESEL*)

Numery(*NIP*, *NrWpisu*, *DataWpisu*, *Miasto*, *#NrO*)

Zapis *#NrO* w schemacie relacji oznacza klucz obcy (czyli *NrO* jest kluczem głównym w innym schemacie relacji – **Osoby**). Kluczem głównym schematu relacji **Numery** jest atrybut *NIP*, ale może być również atrybut *NrO* – atrybut obligatoryjny i unikalny, czyli klucz kandydujący.

■

c. Zw(A(0,1) : B(0,1))



Rys. 4.3. Graficzna reprezentacja związku **Zw**(A(0,1) : B(0,1))

Dla związku binarnego typu *jeden do jeden*, opcjonalnego z obu stron:

- Tworzymy schematy relacji **RA** i **RB** z typów encji A i B.
- Związek binarny **Zw** reprezentujemy nowym schematem relacji **RC**, który zawiera dwa klucze obce: klucz główny ze schematu **RA** i klucz główny ze schematu **RB**.
- Kluczem głównym nowo utworzonego schematu **RC** jest dowolny z kluczy obcych – obydwa są atrybutami obligatoryjnymi i unikalnymi, czyli są kluczami kandydującymi.

Przykład 4.3

Rozważmy związek **Wybiera** zachodzący między encjami typu PANI (encja typu A) i PAN (encja typu B):

PANI(*NrPani, Nazwisko, Imię, KolorWłosów*)
PAN(*NrPana, Nazwisko, Imię, Wykształcenie, DataUr*)
Wybiera(PANI(0,1) : PAN(0,1); *DataWyboru*)

Związek **Wybiera** odzwierciedla dane o powstałych parach. Każda pani może (nie musi) być w parze tylko z jednym panem, każdy pan może (nie musi) być w parze tylko z jedną panią.

W wyniku transformacji zostaną utworzone trzy schematy relacji:

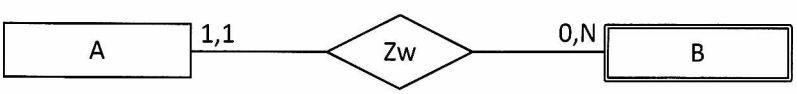
Panie(*NrPani, Nazwisko, Imię, KolorWłosów*)
Panowie(*NrPana, Nazwisko, Imię, Wykształcenie, DataUr*)
Pary(*#NrPani, #NrPana, DataWyboru*)

Kluczem głównym w schemacie relacji **Pary** może być zarówno klucz obcy *NrPani*, jak i *NrPana* – oba są obligatoryjne i unikalne, czyli są kluczami kandydującymi.

■

2. Dla związku binarnego ***jeden do wielu*** mogą zachodzić cztery przypadki:

a. Zw(A(1,1) : B(0,N))



Rys. 4.4. Graficzna reprezentacja związku **Zw**(A(1,1) : B(0,N))

Jest to bardzo często spotykany typ związku binarnego 1:N (*jeden do wielu*), obligatoryjny po stronie jeden: z każdą encją silną typu A może być, ale nie musi, związanych wiele encji typu B, a każda encja słaba typu B musi być związana z dokładnie jedną encją typu A.

- Dla typów encji A i B tworzymy schematy relacji **RA** i **RB**.
- Do schematu **RB** (reprezentującego stronę *wiele* związku) wstawiamy jako dodatkowy atrybut (klucz obcy) klucz główny ze schematu **RA** (ze strony *jeden*). Klucz obcy jest atrybutem obligatoryjnym (NOT NULL).

- Kluczem głównym w schemacie relacji **RA** jest atrybut odpowiadający kluczowi encji typu A, a kluczem głównym w schemacie **RB** jest atrybut odpowiadający kluczowi encji typu B.

Przykład 4.4

Załóżmy, że PRODUCENT (encja typu A) produkuje wiele towarów lub nie produkuje jeszcze żadnego, a TOWAR (encja typu B) jest produkowany przez dokładnie jednego producenta i musi mieć go przypisanego.

Omawianą sytuację ilustrują przykładowe typy encji PRODUCENT i TOWAR oraz związek **Produkuje**:

PRODUCENT(*NrP, NazwaP, AdresP, NIP*)

TOWAR(*NrT, NazwaT*)

Produkuje(PRODUCENT(1,1) : TOWAR(0,N))

Po przeprowadzeniu transformacji modelu konceptualnego do modelu logicznego otrzymamy dwa schematy relacji:

Producenci(*NrP, NazwaP, AdresP, NIP*)

Towary(*NrT, NazwaT, #NrP*)

Obligatoryjność związku po stronie wiele (TOWAR) ma określone konsekwencje. Dopisanie towaru do bazy nie jest możliwe dopóki nie ma wpisanych danych producenta, usunięcie danych producenta, który wycofał się z rynku, wiąże się z koniecznością usunięcia danych towarów przez niego produkowanych lub przydzielenia im innego producenta.

■

b. Zw(A(0,1) : B(0,N))

Rys. 4.5. Graficzna reprezentacja związku **Zw**(A(0,1) : B(0,N))

Związek binarny typu *jeden do wielu*, opcjonalny po obu stronach, można transformować na dwa sposoby:

Sposób 2b1:

- Dla typów encji A i B tworzymy schematy relacji **RA** i **RB**.
- Związek binarny **Zw** reprezentujemy nowym schematem relacji **RC**, który zawiera dwa klucze obce: klucz główny ze schematu **RA** i klucz główny ze schematu **RB**.
- Kluczem głównym nowo utworzonego schematu **RC** jest klucz główny ze schematu **RB**.

Sposób 2b2:

- Jeżeli ponad 60% encji typu B jest w związku z encją typu A, to uznaje się związek za prawie obligatoryjny i zaleca się przeprowadzenie transformacji według reguły 2a, przy czym klucz obcy jest atrybutem **opcjonalnym** (może przyjmować wartości NULL).

Przykład 4.5

Prześledźmy następujący przykład typów encji i związku między nimi:

PRODUCENT(*NrP, NazwaP, AdresP, NIP*)
TOWAR(*NrT, NazwaT*)
Produkuje(PRODUCENT(0,1) : TOWAR(0,N))

a) Jeśli wiele towarów nie ma określonego producenta, to transformację przeprowadzamy według sposobu 2b1.
W wyniku transformacji powstaną trzy schematy relacji:

Producenci(*NrP, NazwaP, AdresP, NIP*)
Towary(*NrT, NazwaT*)
Produkcje(*#NrT, #NrP*)

W schemacie **Produkcje** są numery tylko tych towarów, które mają określonych producentów. Oba klucze obce są atrybutami obligatoryjnymi.

Z pewnością Czytelnikowi nie sprawi kłopotu uzasadnienie, dlaczego *NrP* nie może być kluczem głównym w schemacie relacji **Produkcje**.

b) Jeśli większość towarów (ponad 60%) ma określonego producenta, to transformację przeprowadzamy według sposobu 2b2.
W wyniku transformacji powstaną dwa schematy relacji:

Producenci(*NrP, NazwaP, AdresP, NIP)*
Towary(*NrT, NazwaT, #NrP*)

Klucz obcy *NrP* w schemacie relacji **Towary** jest atrybutem opcjonalnym, ponieważ nie wszystkie towary mają przypisanego producenta.

c. Zw(A(0,1) : B(1,N))

Rys. 4.6. Graficzna reprezentacja związku **Zw**(A(0,1) : B(1,N))

Związek binarny typu *jeden do wielu*, opcjonalny po stronie *jeden* i obligatoryjny po stronie *wiele*, można transformować na dwa sposoby:

Sposób 2c1 (jeśli mniej niż 60% encji typu B uczestniczy w związku **Zw**):
- Z typu encji A (słabej) i z B (silnej) tworzymy schematy relacji **RA** i **RB**.
- Do schematu relacji **RA** jako klucz obcy wstawiamy klucz główny ze schematu **RB**.
- Tworzymy nowy schemat **RC**, w którym umieszczamy jako klucze obce klucze główne ze schematów **RA** i **RB** – oba atrybuty są obligatoryjne.
- Kluczem głównym schematu **RC** jest klucz obcy ze schematu **RB**.

Sposób 2c2:
- Z encji typu A i B tworzymy schematy relacji **RA** i **RB**.
- Do schematu relacji **RA** jako klucz obcy wstawiamy klucz główny ze schematu **RB** (atrybut obligatoryjny). Wartością tego atrybutu może być wartość klucza głównego dowolnej krotki z relacji o schemacie **RB** powiązanej z krotką o schemacie **RA**, np. pierwszej zapisywanej do relacji o schemacie **RB**.
- Do schematu relacji **RB** jako klucz obcy wstawiamy klucz główny ze schematu **RA** (atrybut opcjonalny).

Uwaga. Takie rozwiązanie nie jest możliwe do praktycznej realizacji na poziomie definiowania struktury tabel w relacyjnej bazie danych. Integralność referencyjną należy wówczas zrealizować jako transakcję.

Transformacja przeprowadzona w ten sposób ma wiele wad: nie zapewnia spójności danych i są możliwe anomalie przy przetwarzaniu danych, np. przy usuwaniu danych.

Związek tego typu zachodzi bardzo rzadko i często po ponownej analizie rzeczywistości okazuje się, że np. ze względu na zapewnienie trwałości danych (przechowywanie danych archiwalnych) powinien być to związek typu N:N lub powinien zostać zastąpiony związkiem **Zw**(A(0,1) : B(0,N)), opcjonalnym po stronie *wiele*. Zmiana typu związku wymaga powrotu do wcześniejszych etapów projektowania i poprawy projektu od etapu 1.

Przykład 4.6

Prześledźmy następujący przykład typów encji i związku między nimi:

PRODUCENT(*NrP, NazwaP, AdresP, NIP*)
TOWAR(*NrT, NazwaT*)
Produkuje(PRODUCENT(0,1) : TOWAR(1,N))

a) Jeśli wiele towarów nie ma określonego producenta, to transformację przeprowadzamy według sposobu 2c1.
W wyniku transformacji powstaną trzy schematy relacji:

Producenci(*NrP, NazwaP, AdresP, NIP, #NrT*)
Towary(*NrT, NazwaT*)
Produkcje(*#NrT, #NrP*)

W schemacie **Producenci** klucz obcy *NrT* jest atrybutem obligatoryjnym, np. numerem pierwszego wyprodukowanego towaru.

W schemacie **Produkcje** są numery tylko tych towarów, które mają określonych producentów. Oba klucze obce są atrybutami obligatoryjnymi, *NrT* jest atrybutem unikalnym, więc jest kluczem głównym, natomiast wartości atrybutu *NrP* mogą się powtarzać.

Z pewnością Czytelnik nie będzie miał problemu z uzasadnieniem, dlaczego *NrP* nie może być kluczem głównym w schemacie relacji **Produkcje**.

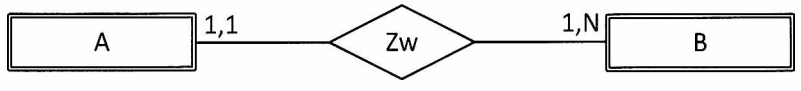
b) Jeśli większość towarów ma określonego producenta, to transformację przeprowadzamy według sposobu 2c2.
W wyniku transformacji powstaną dwa schematy relacji:

Producenci(*NrP, NazwaP, AdresP, NIP, #NrT*)
Towary(*NrT, NazwaT, #NrP*)

Klucz obcy *NrP* w schemacie relacji **Towary** jest atrybutem opcjonalnym, ponieważ nie wszystkie towary mają przypisanego producenta.

W schemacie **Producenci** klucz obcy *NrT* (atrybut obligatoryjny) jest numerem dowolnego towaru produkowanego przez producenta, np. pierwszego wpisanego do bazy danych.

d. Zw(A(1,1) : B(1,N))



Rys. 4.7. Graficzna reprezentacja związku **Zw**(A(1,1) : B(1,N))

Związek binarny typu *jeden do wielu*, obligatoryjny po obu stronach, zachodzi bardzo rzadko. Często po głębszej analizie rzeczywistości okazuje się, że powinien to być związek opcjonalny po stronie *wiele*, czyli powinien zostać zastąpiony często spotykanym związkiem **Zw1**(A(1,1) : B(0,N)), co wymaga powrotu do wcześniejszych etapów projektowania bazy i poprawy projektu (od etapu 1).

Pozostawienie związku w niezmienionej postaci będzie przyczyną anomalii przy usuwaniu i dopisywaniu danych. Wszystkie encje typu A muszą być w związku **Zw** z co najmniej jedną encją typu B. Wszystkie encje typu B muszą być w związku **Zw** z dokładnie jedną encją typu A.

- Z typów encji A i B tworzymy schematy relacji **RA** i **RB**.
- Do schematu **RA** jako klucz obcy wstawiamy klucz główny ze schematu **RB**.
- Do schematu **RB** jako klucz obcy wstawiamy klucz główny ze schematu **RA**.
- Wstawione klucze obce są atrybutami obligatoryjnymi.
- Kluczem głównym w schemacie relacji **RA** jest klucz główny z typu encji A, a kluczem głównym w schemacie **RB** jest klucz główny z typu encji B.

Uwaga. Takiego rozwiązania nie można w praktyce zrealizować na poziomie definiowania struktury tabel w relacyjnej bazie danych. Integralność referencyjną należy wówczas zrealizować jako transakcję. Szczegółowe rozważania na temat tego typu związków zamieszczono w rozdz. 4.1.3.

Przykład 4.7

Rozważmy dwa typy encji i związek między nimi:

KLIENT(*NrKlienta*, *Nazwisko*, *Imię*)

BILET(*NrBiletu*, *Lot*, *Data*, *Godzina*)

Ma(KLIENT(1,1) : BILET(1,N))

W związku **Ma** biorą udział wszyscy klienci, którzy mają wykupiony co najmniej jeden bilet na przelot samolotem, mogą mieć kupionych wiele biletów, ale dany bilet jest przypisany dokładnie do jednej osoby.

Po transformacji związku otrzymamy schematy relacji:

Klienci(*NrKlienta*, *Nazwisko*, *Imię*, *#NrBiletu*)

Bilety(*NrBiletu*, *Lot*, *Data*, *Godzina*, *#NrKlienta*).

Atrybut *NrBiletu* w schemacie **Klienci** odnosi się do dowolnego biletu klienta, np. pierwszego lub ostatnio zakupionego. Klucze obce *NrBiletu* i *NrKlienta* są atrybutami obligatoryjnymi. Klucz obcy *NrBiletu* jest kluczem kandydującym w schemacie **Klienci** (klient musi mieć numer biletu i numer biletu nie może się powtarzać).

Tak zaprojektowana baza danych ma jednak wady: nie zapewnia spójności danych (klientowi można przypisać numer biletu, którego nie kupił), mogą wystąpić anomalie, np. usunięcie danych klienta powoduje konieczność usunięcia danych o biletach przez niego zakupionych, nie ma więc możliwości przechowywania danych o biletach do sporządzania zestawień statystycznych dotyczących sprzedaży biletów, bez jednoczesnego przechowywania danych o klientach. Ponadto trzeba kontrolować usuwanie danych o biletach danego klienta: usunięcie danych o ostatnim bilecie wymaga usunięcia danych klienta.

W praktyce taki związek jest niemożliwy do zrealizowania za pomocą więzów integralności i powinien być zastąpiony związkiem

Ma(KLIENT(1,1) : BILET(0,N)).

■

3. Dla związku binarnego ***wiele do wielu*** teoretycznie mogą zachodzić trzy przypadki, jednak w praktyce realizowany poprawnie jest tylko związek opcjonalny wiele do wielu, którego sposób transformacji opisuje reguła 8.

e. Zw(A(0,N) : B(0,N))

Rys. 4.8. Graficzna reprezentacja związku Zw(A(0,N) : B(0,N))

Dla związku binarnego typu *wiele do wielu,* opcjonalnego z obu stron:

- Tworzymy schematy relacji **RA** i **RB** z typów encji A i B.
- Związek binarny **Zw** reprezentujemy nowym schematem relacji **RC**, który zawiera dwa klucze obce: klucz główny ze schematu **RA** i klucz główny ze schematu **RB**. Oba klucze obce są atrybutami obligatoryjnymi.
- Kluczem głównym w schemacie relacji **RC** może być klucz złożony z obu kluczy obcych lub nowo wprowadzony klucz sztuczny (klucz prosty).
- Jeśli związek posiadał atrybuty dodatkowe, to stają się one atrybutami w utworzonym schemacie relacji **RC**.

Zatem zgodnie z ogólną regułą transformacji **związku obustronnie opcjonalnego *wiele do wielu*** tworzy się nowy schemat relacji ze złożonym (z kluczy obcych) kluczem głównym (jeśli para wartości kluczy obcych nie może się powtarzać) lub ze sztucznym kluczem głównym (jeśli para wartości kluczy obcych może się powtarzać).

Uwaga. Związek *wiele do wielu* opcjonalny po obu stronach zaleca się zastąpić dwoma związkami 1:N (rozdz. 3.7.2, rysunek 3.6).

Przykład 4.8

a) Załóżmy, że OSOBA (encja typu A) może napisać wiele książek lub nie być autorem żadnej książki, a KSIĄŻKA (encja typu B) może mieć wielu autorów lub żadnego (autor nieznany). Definicje typów encji OSOBA i KSIĄŻKA oraz związku **Jest** są następujące:
   OSOBA(*NrO*, *Nazwisko*, *Imię*)
   KSIĄŻKA(*NrK*, *TytułK*)
   Jest(OSOBA(0,N) : KSIĄŻKA(0,N); *Uwagi*)

Schematy relacji otrzymane w wyniku transformacji mają postać:

Osoby(*NrO*, *Nazwisko*, *Imię*)
Książki(*NrK*, *TytułK*)
Autorstwa(*#IdO*, *#IdK*, *Uwagi*)

Ponieważ dana osoba może tylko raz być przypisana jako autor do danej książki, więc zbiór wartości atrybutów {*#NrO*, *#NrK*} jest unikalny (nie może się powtórzyć), zatem para {*#NrO*, *#NrK*} jest kluczem głównym złożonym schematu **Autorstwa**.

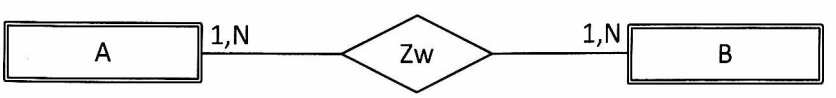
b) Załóżmy, że KLIENT (encja typu A) może wypożyczać wiele kaset lub nie mieć wypożyczonej jeszcze żadnej, a KASETA (encja typu B) może być wypożyczana przez wielu klientów (po oddaniu kasety danych o wypożyczeniu się nie usuwa) lub ani razu jeszcze nie być wypożyczona. Klient może wielokrotnie wypożyczać tę samą kasetę. Definicje typów encji KLIENT i KASETA oraz związku **Wypożyczył** są następujące:
KLIENT(*NrKlienta*, *Nazwisko*, *Imię*)
KASETA(*NrKasety*, *TytułFilmu*)
Ma(KLIENT(0,N) : KASETA(0,N); *DataW*, *DataZ*, *Uwagi*)

Schematy relacji otrzymane w wyniku transformacji mają postać:
Klienci(*NrKlienta*, *Nazwisko*, *Imię*)
Kasety(*NrKasety*, *TytułFilmu*)
Wypożyczenia(*NrW*, *#NrKlienta*, *#NrKasety*, *DataW*, *Uwagi*).

Ponieważ dany klient może wielokrotnie wypożyczyć tę samą kasetę (nawet tego samego dnia może ją dwukrotnie wypożyczyć i oddać), więc zbiór atrybutów {*#NrKlienta*, *#NrKasety*} nie może być kluczem głównym schematu **Wypożyczenia**, dlatego został wprowadzony klucz sztuczny *NrW*.

■

f. Zw(A(1,N) : B(1,N))



Rys. 4.9. Graficzna reprezentacja związku **Zw**(A(1,N) : B(1,N))

Związek *wiele do wielu* obligatoryjny po obu stronach, przedstawiony na rysunku 4.9, uważa się za niemożliwy do wystąpienia w praktyce. Związek ten oznacza, że encje po jednej stronie związku nie mogą istnieć bez encji po drugiej stronie związku. W praktyce po szczegółowej analizie takiego związku okazuje się, że nie oddaje on dobrze rzeczywistości, jest przyczyną powstawania anomalii (np. dopisywania i usuwania danych) i związek ten trzeba zastąpić odpowiednim związkiem innego typu. Gdyby zgodnie z zaleceniem, by związki *wiele* do *wielu* zastępować dwoma związkami prostym, utworzyć dwa związki, to będą one postaci **Z**(A(1,1) : B(1,N)), a ich transformacja została już omówiona w punkcie 2b, jako niemożliwa do zrealizowania w praktyce. Tak więc należy rozważyć zastąpienie związku **Zw** dwoma związkami typu Z(A(1,1) : B(0,N)) (rozdz. 3.7.2, rysunek 3.6).

g. Zw(A(0,N) : B(1,N))

Rys. 4.10. Graficzna reprezentacja związku **Zw**(A(0,N) : B(1,N))

Związek tego typu należy do bardzo rzadko spotykanych i nie jest możliwy do praktycznej realizacji na poziomie definiowania struktury tabel w relacyjnej bazie danych. Integralność referencyjną należy wówczas zrealizować jako transakcję. Pozostawienie go w takiej postaci jest także przyczyną powstawania anomalii w bazie danych przy wykonywaniu podstawowych operacji, np. dopisywania i usuwania danych. Zaleca się ponowną analizę rzeczywistości i zastąpienie związku związkiem obustronnie opcjonalnym.

Przykład 4.9

Załóżmy, że KLIENT (encja typu A) musi wypożyczyć co najmniej jedną kasetę wideo, ale może wypożyczyć wiele kaset. KASETA (encja typu B) może być wypożyczana przez wielu klientów (po oddaniu kasety dane o wypożyczeniu nie są usuwane) lub ani razu jeszcze nie być wypożyczona. Definicje typów encji i związku **Ma** są następujące:

KLIENT(*NrKlienta*, *Nazwisko*, *Imię*)
KASETA(*NrKasety*, *TytułFilmu*)
Ma(KLIENT(0,N) : KASETA(1,N); *DataW*, *DataZw*)

Schematy relacji otrzymane w wyniku transformacji mają postać:

Klienci(*NrKlienta*, *Nazwisko*, *Imię*, *#NrKasety*)
Kasety(*NrKasety*, *TytułFilmu*)
Wypożyczenia(*NrW*, *#NrKlienta*, *#NrKasety*, *DataW*, *DataZw*).

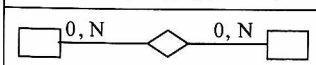
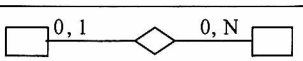
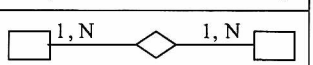
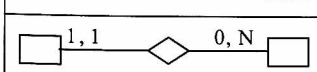
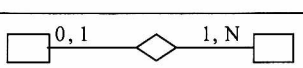
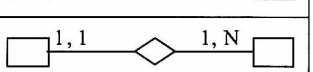
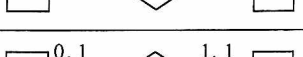
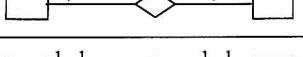
Ponieważ zbiór atrybutów {*#NrKlienta*, *#NrKasety*} nie może być kluczem głównym schematu **Wypożyczenia**, dlatego został wprowadzony klucz sztuczny *NrW*. Atrybut *NrKasety* w schemacie **Klienci** jest obligatoryjny (jest numerem pierwszej wypożyczonej kasety). W tak zaprojektowanej bazie danych nie można przechowywać danych klientów, którzy nie wypożyczyli żadnej kasety. Przy usuwaniu danych o wypożyczeniach (np. z ubiegłego roku) należy kontrolować, czy nie trzeba też usunąć danych klienta, co znacznie wydłuża czas wykonywania transakcji usunięcia. W praktyce związek tego typu jest niemożliwy do zrealizowania za pomocą więzów integralności.

■

W tabeli 4.1 przedstawiamy typy związków występujące w praktyce często, rzadko oraz związki trudne do zrealizowania za pomocą więzów integralności [Barker1996]. Związki wymienione jako trudne do realizacji należy rozumieć jako niemożliwe do realizacji na poziomie struktur tabel i więzów integralności.

Tabela 4.1

Występowanie różnych typów związków

Związki występujące często	Związki występujące rzadko	Związki trudne do realizacji
		
		
		
		
		

Takie związki są możliwe do realizacji za pomocą odpowiedniej implementacji (transakcji), ale mimo wszystko są one często przyczyną anomalii (np. dodawania i usuwania danych) oraz utraty spójności danych. Z tego powodu należy za każdym razem rozważyć zastąpienie takich związków (jeśli to możliwe i pożądane) innymi związkami.

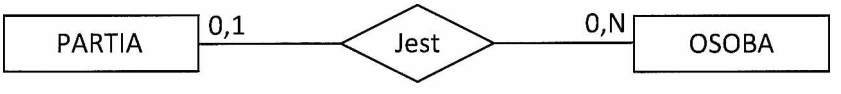
Przeprowadzając transformację modelu konceptualnego do modelu logicznego, należy to robić w ustalony, usystematyzowany sposób, tak by uwzględnić wszystkie występujące typy encji i związków oraz wszystkie ich atrybuty i klucze obce, klucze kandydujące i klucze główne.

4.1.3. PRZYKLADY TRANSFORMACJI ZWIĄZKÓW

W rozdziale 4.1.2 zostały podane przykłady zastosowania poszczególnych reguł transformacji dla wszystkich możliwych do wystąpienia typów związków binarnych wymienionych w tabeli 4.1. W celu lepszego zilustrowania reguł transformacji rozważmy kolejne przykłady związków.

Przykład 4.10

Na rysunku 4.11 przedstawiono rzadko występujący, ale jednak możliwy związek obustronnie opcjonalny typu 1:N.



Rys. 4.11. Przykład obustronnie opcjonalnego związku typu 1:N

Zakładamy, że każda PARTIA może mieć wielu członków lub jeszcze nie mieć żadnego i każda OSOBA może zapisać się do jednej partii lub być bezpartyjna (nie być członkiem żadnej partii).

Transformację możemy przeprowadzić na dwa sposoby:

a) Z encji typu OSOBA utworzyć schemat relacji **Osoby**, z encji typu PARTIA utworzyć schemat relacji **Partie** i do schematu **Osoby** wstawić jako klucz obcy (atrybut opcjonalny) klucz główny (*IdP*) ze schematu relacji **Partie**.

Jeżeli bardzo mało osób należy do jakiejś partii (np. w Polsce tylko ok. 0,8% uprawnionych), to w relacji **Osoby** dla atrybutu *IdP* bardzo mało osób będzie miało wpisany identyfikator partii, a dla pozostałych osób ten atrybut będzie miał wartość pustą (nieokreśloną). Wówczas korzystniejsza będzie transformacja według sposobu b).

b) Dla małej liczby encji typu PARTIA i OSOBA pozostających w związku **Jest** podczas transformacji związku korzystniej jest utworzyć trzy schematy relacji: **Osoby**, **Partie** oraz **Członkowie**, w którym kluczem głównym będzie klucz obcy ze schematu **Osoby**.

■

W podobny sposób jak w przykładzie 4.10 należy rozważyć każdy związek podczas transformacji. Wybór sposobu transformacji zależy także od innych uwarunkowań, na przykład, czy w rzeczywistości możliwe będzie rozwiązanie polegające na dopisaniu klucza obcego do danego schematu relacji. Internauta może na przykład korzystać z danych otwartych udostępnianych w internecie, ale nie może zmienić struktury udostępnianych danych, czyli dopisać klucza obcego do tabeli udostępnianej bazy danych. Taka sytuacja może mieć także miejsce w firmie, gdzie dane osobowe można pobierać z działu kadr, ale nie można w bazie kadr niczego zmieniać.

W rozdz. 4.1.2 w punkcie 2d omówiono transformację związku binarnego obustronnie obligatoryjnego typu 1:N:

Zw(A(1,1) : B(1,N))

wymienionego w tabeli 4.1 wśród związków trudnych do realizacji.

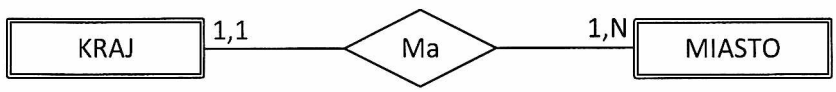
W przykładzie 4.11 przeanalizujmy dokładnie ten typ związku.

Przykład 4.11

Rozważmy dwie encje typu KRAJ i MIASTO oraz związek **Ma** (rysunek 4.12) wymieniony w tabeli 4.1 jako niemożliwy:

KRAJ(*IdK*, *NazwaK*)
MIASTO(*IdM*, *NazwaM*)
Ma(KRAJ(1,1) : MIASTO(1,N))

Dla uproszczenia pomijamy pozostałe możliwe atrybuty w typach encji KRAJ i MIASTO oraz w związku **Ma**.



Rys. 4.12. Graficzna reprezentacja związku **Ma**

Zgodnie z regułami transformacji po sprowadzeniu modelu konceptualnego do modelu logicznego otrzymamy dwa schematy relacji:

Kraje(*IdK, NazwaK, #IdS*)

Miasta(*IdM, NazwaM, #IdP*)

Klucze obce *IdS* (identyfikator miasta, które jest stolicą) oraz *IdP* (identyfikator państwa, w którym leży miasto) są atrybutami obligatoryjnymi.

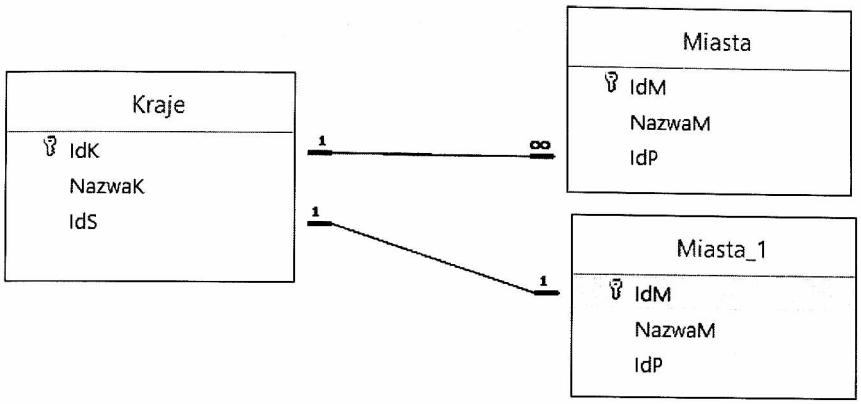
Wydaje się oczywiste, że każde miasto **musi** mieć przypisany **dokładnie jeden** kraj, do którego należy, a każdy kraj może mieć przypisanych **wiele** miast i **musi** mieć przypisaną dokładnie **jedną** stolicę.

Tak sformułowane reguły wydają się być zgodne z rzeczywistością, ale nie uwzględniają chronologii w tworzeniu danych, co jest częstą przyczyną definiowania złych związków. To, że kraj musi mieć przypisane miasto, które jest jego stolicą, nie oznacza, że to przypisanie trzeba zrobić jednocześnie z wpisywaniem danych kraju.

Dla tak zdefiniowanych schematów relacji **Kraje** i **Miasta** po zaimplementowaniu ich w fizycznej bazie danych jako tabele, nie można wprowadzać jednocześnie danych do obu tabel z zachowaniem więzów integralności. Utworzone referencje i wymuszone więzy integralności pomiędzy kluczami głównymi a kluczami obcymi nie pozwalają zapisać danych, gdyż nie można jedocześnie wpisywać danych do obu tabel, ponieważ fizycznie któraś z nich musi być zapisana jako pierwsza.

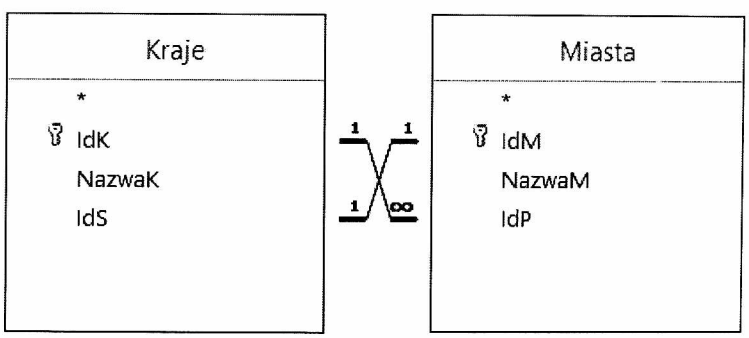
Tego typu związki są różnie nazywane, np. Date nazywa je **cyklami referencyjnymi** ([Date2000], s. 151, p. 3; s. 160 ćw. 5.4; s. 166), natomiast Celko – **sprzecznymi relacjami** ([Celko2016], s. 202).

Przykład powiązania między zaimplementowanymi schematami relacji w SZBD MS Access przedstawiono na rysunku 4.13 (widok okna relacji). Ze względu na problem z realizacją implementacji tabel Kraje i Miasta w systemie tworzona jest wirtualna kopia tabeli Miasta z nazwą Miasta_1, co umożliwia wizualizację powiązań, ale nie rozwiązuje problemów z wprowadzaniem danych do tabel.



Rys. 4.13. Implementacja związku **Ma**(KRAJ(1,1) : MIASTO(1,N)) w MS Access

Widok z projektu kwerendy w systemie MS Access przedstawiony na rysunku 4.14 odzwierciedla wzajemne powiązania między tabelami wynikające z cyklu referencyjnego.



Rys. 4.14. Wizualizacja cyklu referencyjnego

W praktyce wprowadzanie danych do tak zdefiniowanych tabel Kraje i Miasta wymaga zastosowania jednego z dwóch rozwiązań:

a) użycia transakcji (tzw. transakcji opóźnionej), czyli najpierw wpisania danych kraju bez przypisywania stolicy (czyli potraktowania atrybutu *IdS* jak opcjonalnego), potem wpisania danych miasta i przypisania jemu państwa, do którego należy, a na koniec przypisania do kraju odpowiedniego miasta jako stolicy, natomiast nie jest możliwe wpisanie danych bez użycia transakcji (bezpośrednio do tabel) – taka transakcja musi być zdefiniowana w etapie 6;

b) utworzenia dodatkowej tabeli Stolice, zawierającej klucze obce z tabel Kraje i Miasta (oba są obligatoryjne o unikalnej wartości, czyli są kluczami kandydującymi), umożliwia wprowadzanie danych bezpośrednio do tabel, ale nie zapewnia spójności danych.

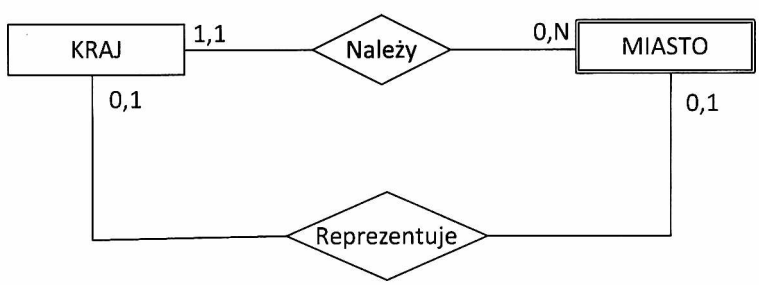
Oba rozwiązania **nie są** równoważne związkowi **Ma**, gdyż nie zapewniają spójności danych – jako stolicę można przypisać miasto, które nie leży w tym kraju. Ponadto nie gwarantują one, że dane miasto zostanie przypisane do tego samego kraju, do którego zostało już przypisane jako stolica, więc naruszają związek **Ma**, z którego wynika, że miasto musi być związane tylko z jednym krajem. Aby ten warunek był spełniony, trzeba go realizować odpowiednią transakcją.

Przedstawione rozwiązania są w rzeczywistości realizacją dwóch niezależnych związków (rysunek 4.15):

Należy(KRAJ(1,1) : MIASTO(0,N))

Reprezentuje(KRAJ(0,1) : MIASTO(0,1))

przy czym związek obustronnie opcjonalny **Reprezentuje** jest prawie obligatoryjny po stronie MIASTO (każdy kraj ma ostatecznie przypisaną stolicę).



Rys. 4.15. Graficzna reprezentacja związków **Należy** i **Reprezentuje**

Po transformacji tych związków otrzymujemy trzy schematy relacji

Kraje(*IdK*, *NazwaK*)

Miasta(*IdM*, *NazwaM*, *#IdP*)

Stolice(*#IdK*, *#IdM*)

gdzie wszystkie klucze obce są atrybutami obligatoryjnymi i klucz obcy *IdM* jest kluczem kandydującym w schemacie relacji **Stolice**. To rozwiązanie jest zgodne z regułami transformacji dla związków **Należy** i **Reprezentuje**.

Przy takiej definicji związków nie ma problemu z wprowadzaniem danych, ale nadal jest problem z niespójnością danych (można przypisać miasto do jednego kraju, a jako stolicę do innego).

Można także rozważyć inne możliwe transformacje związków **Należy** i **Reprezentuje**:

1) **Kraje**(*IdK*, *NazwaK*, *#IdS*)
Miasta(*IdM*, *NazwaM*, *#IdP*)

przy czym klucz obcy *IdP* jest atrybutem obligatoryjnym, a klucz obcy *IdS* jest atrybutem opcjonalnym o niepowtarzających się wartościach,

2) **Kraje**(*IdK*, *NazwaK*)
Miasta(*IdM*, *NazwaM*, *#IdP*, *#NrK*)

przy czym klucz obcy *IdP* jest atrybutem obligatoryjnym, a klucz obcy *NrK* (numer kraju, którego dane miasto jest stolicą) jest atrybutem opcjonalnym o niepowtarzających się wartościach. Przy takim rozwiązaniu wszystkie miasta, które nie są stolicami, nie będą miały określonej wartości atrybutu *NrK*.

Kolejnym rozwiązaniem umożliwiającym spełnienie określonych dla bazy wymagań jest dodanie do kategorii, a następnie do typu encji MIASTO atrybutu *CzyStolica* (np. o wartościach *True/False*). Otrzymalibyśmy wówczas prosty do transformacji związek **Ma**:

KRAJ(*IdK*, *NazwaK*)
MIASTO(*IdM*, *NazwaM*, *CzyStolica*)
Ma(KRAJ(1,1) : MIASTO(0,N))

W wyniku transformacji powstaną dwa schematy relacji:

Kraje(*IdK*, *NazwaK*)
Miasta(*IdM*, *NazwaM*, *CzyStolica*, *#IdP*)

To rozwiązanie **zapewnia spójność**, ale w relacji o schemacie **Miasta** niewiele krotek dla atrybutu *CzyStolica* ma wartość *True*.

Projektując bazę danych, należy zatem rozważyć różne możliwe rozwiązania i wybrać to najbardziej odpowiadające potrzebom i możliwościom realizacji.

■

Przykład 4.12

W rozdziale 3.9 w przykładzie bazy **WYPOŻYCZALNIA** przedstawionej na rysunku 3.7 występuje związek **Zawiera** obustronnie obligatoryjny

Zawiera(KASETA(1,N) : FILM(1,1))

zachodzący między encjami typu KASETA i FILM.

Pozornie może się wydawać, że jest to dobrze zdefiniowany związek – najpierw wpisujemy film, od razu podajemy numery kaset, na których ten film się znajduje (musi być co najmniej jedna kaseta z filmem, może być ich też więcej). Zakładamy, że dana kaseta zawiera tylko jeden film i nie ma kaset pustych. Obligatoryjność po stronie typu encji KASETA oznacza, że każda kaseta musi być powiązana z jednym filmem. Podobnie dla encji typu FILM – każdy film musi być powiązany chociaż z jedną kasetą. Usuwając dane o kasecie, która uległa zniszczeniu lub zgubieniu, należy kontrolować, czy to była ostatnia kaseta z tym filmem. Jeśli tak, to usuwając dane o kasecie, trzeba też usunąć dane o filmie. W praktyce na ogół nie usuwa się wtedy opisów filmów, bo jeśli dokupimy kasetę z takim filmem, to ponownie będziemy musieli wprowadzać opis filmu. Dlatego taki związek odpowiada sformułowanym regułom, ale formuły zostały źle sformułowane, co może być przyczyną anomalii usuwania i dopisywania danych, a w rzeczywistości znacznym utrudnieniem podczas korzystania z bazy. Po ponownej analizie rzeczywistości i wad takiego związku oraz ze względu na utrzymanie integralności i spójności danych, stwierdzamy, że należy zastąpić go związkiem

Zawiera (KASETA(0,N) : FILM(1,1))

Przy tak określonym związku szybciej wykonywana jest operacja usuwania danych – przy usuwaniu danych kasety z określonym filmem nie trzeba za każdym razem kontrolować, czy istnieje jeszcze jakaś kaseta z danym filmem, ponieważ mogą istnieć dane o filmie nie powiązanym z żadną kasetą znajdującą się na stanie wypożyczalni. Można więc przechowywać opisy filmów, które nie są dostępne na kasetach.

■

4.1.4. TRANSFORMACJA ZWIĄZKÓW TERNARNYCH I REKURENCYJNYCH

Na diagramach związków encji występują nie tylko związki binarne, ale także inne typy związków, np. ternarne (między trzema encjami) i rekurencyjne (związek między encjami tego samego typu).

Najczęściej spotykanym typem związku ternarnego jest *wiele do wielu do wielu*, opcjonalny z każdej strony

Zw(A(0,N) : B(0,N): C(0,N)).

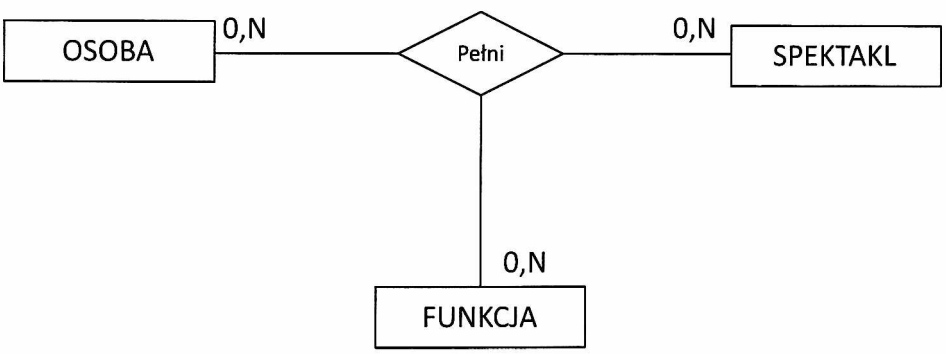
Wynikiem transformacji związku **Zw** jest nowy schemat relacji **RABC**, do którego wstawiane są klucze główne z encji typu A, B, C.

Związek ternarny powinien zostać zastąpiony encją asocjacyjną i trzema związkami typu 1:N.

Przykład 4.13

W bazie **TEATR** mają być zapisywane dane osób związanych ze spektaklem teatralnym i pełniących w nim różne funkcje, np. aktor, reżyser, charakteryzator. Każda osoba może pełnić w spektaklu kilka funkcji lub żadną. Zdefiniowano w tym celu trzy typy encji OSOBA, SPEKTAKL, FUNKCJA oraz związek ternarny między nimi **Pełni**:

OSOBA(*IdO*, *Nazwisko*, *Imię*)
SPEKTAKL(*IdS*, *Tytuł*, *RokPremiery*)
FUNKCJA(*IdF*, *NazwaF*)
Pełni(OSOBA(0,N) : SPEKTAKL(0,N): FUNKCJA(0,N); *Uwaga*).



Rys. 4.16. Graficzna reprezentacja związku ternarnego **Pełni**

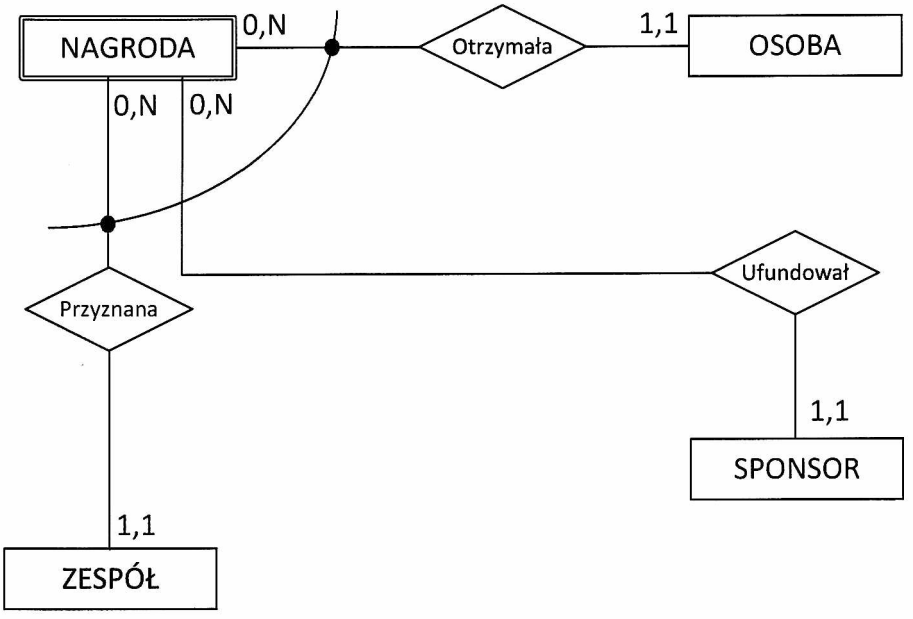
Po transformacji typów encji i związku powstaną cztery schematy:

Osoby(*IdO*, *Nazwisko*, *Imię*)
Spektakle(*IdS*, *Tytuł*, *RokPremiery*)
Funkcje(*IdF*, *NazwaF*)
Realizacje(#*IdO*, #*IdS*, #*IdF*, *Uwaga*)

Należy rozważyć, czy w schemacie relacji **Realizacje** klucz główny złożony z trzech atrybutów {#*IdO*, #*IdS*, #*IdF*} pozostawić, czy zastąpić prostym kluczem sztucznym *IdR*. Wprowadzenie klucza prostego jest wręcz konieczne w sytuacji,

gdy osoba jest aktorem w danym spektaklu i odgrywa dwie różne role, które mają być odnotowane w bazie jako dwa odrębne wpisy różniące się wartością atrybutu *Uwaga*.

W rozdziale 3.9 omówiono związki wzajemnie się wykluczające. Przeanalizujmy rysunek 4.17 przedstawiający na diagramie związków encji alternatywę wykluczającą oznaczoną łukiem wykluczającym po stronie *wiele* związku.



Rys. 4.17. Związki **Otrzymała** i **Przyznana** wzajemnie się wykluczające

Tworząc model logiczny, należy wziąć pod uwagę, czy w typach encji wzajemnie się wykluczających klucze główne mają takie same dziedziny, czy też nie.

Przypadek I

Dziedziny kluczy głównych typów encji pozostających w związkach wzajemnie się wykluczających są takie same. Wówczas do schematu relacji powstałego z typu encji powiązanej wstawiamy dwa dodatkowe (obligatoryjne) atrybuty:

- identyfikator encji (klucz główny z typu encji ze strony *jeden*),
- identyfikator związku, którego wartość służy do rozróżniania związków.

Przykład 4.14

Załóżmy, że typy encji z rysunku 4.17 są zdefiniowane następująco:

ENC/001 NAGRODA(*IdN, TytułN*)

ENC/002 OSOBA(*IdOsoby, Nazwisko, Imię, PESEL, Telefon*)

ENC/003 ZESPÓŁ(*IdZespołu, NazwaZ, LiczbaOsób, DataPowstania*)

ENC/003 SPONSOR(*IdSponsora, NazwaS, Telefon*)

Zauważmy, że związek **Ufundował** występujący na diagramie 4.17 nie jest związkiem wykluczającym.

Jeśli dom(*IdOsoby*) = dom(*IdZespołu*), to po transformacji ERD do modelu logicznego otrzymamy schematy relacji:

Nagrody(*IdN, TytułN, #IdS, OsobaZespół, #IdNagrodzonego*)

Osoby(*IdOosoby, Nazwisko, Imię, PESEL, Telefon*)

Zespoły(*IdZespołu, NazwaZ, LiczbaOsób, DataPowstania*)

Sponsorzy(*IdSponsora, NazwaS, LiczbaOsób, DataPowstania*)

Dodatkowy obligatoryjny atrybut *OsobaZespół* w schemacie relacji **Nagrody** może przyjmować wartości:

"O" – nagrodzona została osoba i obligatoryjny atrybut *IdNagrodzonego* (klucz obcy) jest wartością klucza głównego ze schematu **Osoby**,

"Z" – nagrodzony został zespół i obligatoryjny atrybut *IdNagrodzonego* (klucz obcy) jest wartością klucza głównego ze schematu relacji **Zespoły**.

■

Przypadek II

Dziedziny kluczy głównych typów encji wykluczających się wzajemnie są różne. Wówczas do schematu relacji powstałego z encji powiązanej wstawiamy tyle dodatkowych (opcjonalnych) atrybutów, ile typów encji było połączonych łukiem wykluczającym (odpowiadają one ich kluczom głównym). W każdej krotce utworzonej relacji tylko jeden z tych atrybutów może mieć wartość różną od NULL, a w przypadku związków obligatoryjnych – dokładnie jeden musi mieć wartość różną od NULL.

Przykład 4.15

Jeśli dla typów encji z przykładu 4.14, zachodzi dom(*IdOsoby*) ≠ dom(*IdZespołu*), np. wartościami atrybutu *IdOsoby* są liczby dodatnie, a wartościami atrybutu *IdZespołu* są łańcuchy sześcioznakowe, to po transformacji modelu konceptualnego do modelu logicznego otrzymamy następujące schematy relacji:

Nagrody(*IdNagrody, TytułNagrody, #IdOsoby, #IdZespołu*)
Osoby(*IdOsoby, Nazwisko, Imię, PESEL, Telefon*)
Zespoły(*IdZespołu, NazwaZ, LiczbaOsób, DataPowstania*)
Sponsorzy(*IdSponsora, NazwaS, LiczbaOsób, DataPowstania*)

W każdej krotce relacji o schemacie **Nagrody** jeden z opcjonalnych atrybutów – kluczy obcych – *IdOsoby* lub *IdZespołu*, musi mieć wartość różną od NULL (wartość klucza głównego z odpowiedniego schematu relacji), a drugi z nich musi mieć wówczas wartość NULL. Kontrolę tego typu warunków można zapisać w regule poprawności, np.

IsNull(*IdOsoby*) or IsNull(*IdZespołu*) and Not IsNull(*IdOsoby & IdZespołu*)

Kolejnym typem związku występującym na diagramach związków encji jest związek unarny zwany też **rekurencyjnym**. Taki związek zachodzi między encjami tego samego typu. Po transformacji związku rekurencyjnego jednostronnie obligatoryjnego typu 1:1 lub 1:N do schematu relacji odpowiadającego typowi encji wstawiany jest dodatkowy atrybut – klucz obcy do tego schematu. Nazwa tego atrybutu musi zostać zmieniona, ponieważ w schemacie nie może być dwóch atrybutów o takiej samej nazwie.

Przykład 4.16

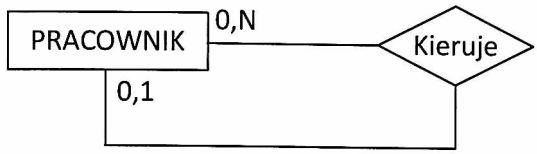
Dany jest typ encji

PRACOWNIK(*IdP, Nazwisko, Imię, Login, Hasło*)

Związek rekurencyjny **Kieruje** ma postać:

Kieruje(PRACOWNIK(0,1) : PRACOWNIK(0,N))

czyli każdy pracownik może mieć przypisanego jednego kierownika (pracownika), a kierownik może być przypisany do wielu pracowników (rysunek 4.18).



Rys. 4.18. Związek rekurencyjny 1:N

Otrzymany po transformacji schemat relacji **Pracownicy** ma postać:

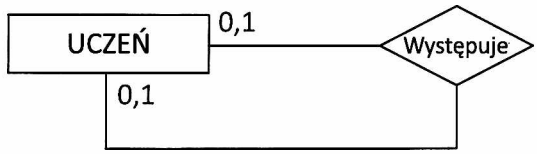
Pracownicy(*IdP*, *Nazwisko*, *Imię*, *Login*, *Hasło*, *#IdK*),

gdzie klucz obcy *IdK* (atrybut opcjonalny), który nie może mieć nazwy takiej samej jak klucz główny, jest numerem kierownika dla danego pracownika. W szczególności pracownik może być też swoim kierownikiem.

■

Przykład 4.17

Na konkurs piosenki należy utworzyć jedno lub dwuosobowe zespoły uczniów. W zespole dwuosobowym jeden uczeń śpiewa, a drugi gra na gitarze, w zespole jednoosobowym uczeń zarówno śpiewa, jak i gra na gitarze. Uczeń może zaśpiewać tylko jedną piosenkę i tylko jeden raz grać na gitarze (rysunek 4.19).



Rys. 4.19. Związek rekurencyjny typu 1:1

Dla typu encji

UCZEŃ(*IdU*, *Nazwisko*, *Imię*, *PESEL*)

i związku rekurencyjnego

Występuje(UCZEŃ(0,1) : UCZEŃ(0,1); *Piosenka*)

w wyniku transformacji otrzymamy dwa schematy relacji:

Uczniowie(*IdU*, *Nazwisko*, *Imię*, *PESEL*)

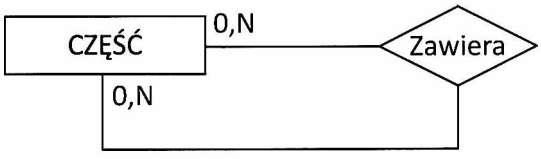
Zespoły(*#IdUŚpiew*, *#IdUGra*, *Piosenka*)

gdzie oba klucze obce *IdUŚpiew*, *IdUGra* w schemacie relacji **Zespoły** są kluczami kandydującymi i mogą mieć taka samą wartość, jeśli ta sama osoba śpiewa i gra.

■

Przykład 4.18

W firmie produkowane są części, które mogą składać się z wielu innych części. Dana część może być elementem wielu innych części (rysunek 4.20).



Rys. 4.20. Związek rekurencyjny N:N

Dla typu encji

CZĘŚĆ(*IdC*, *Nazwa*, *Opis*)

i związku rekurencyjnego

Zawiera(CZĘŚĆ(0,N) : CZĘŚĆ (0,N); *Liczba*)

w wyniku transformacji otrzymamy dwa schematy relacji:

Części(*IdC*, *Nazwa*, *Opis*)

Zestawy(#*IdC*, #*IdS*, *Liczba*),

W schemacie relacji **Zestawy** (zawierającym tylko części składane z innych części) klucz główny tworzą dwa klucze obce ze schematu **Części**, z których *IdC* jest numerem części, a *IdS* jest numerem jej części składowej.

Przeprowadzając transformację modelu konceptualnego do modelu logicznego, należy to robić w usystematyzowany sposób, tak by uwzględnić wszystkie występujące typy encji i związków oraz wszystkie ich atrybuty i klucze obce, klucze kandydujące i klucze główne.

Na zakończenie przeglądu reguł transformacji modelu konceptualnego do modelu logicznego rozważmy jeszcze zagadnienie **atrybutu wyliczanego**. Na ogół wartość atrybutu wyliczanego (np. przychód w firmie w poprzednim roku, średnia ocen studenta za poprzedni rok studiów, utarg w sklepie w ubiegłym tygodniu) nie jest przechowywana w bazie danych, tylko wyliczana za każdym razem, kiedy zachodzi taka potrzeba. Jeśli jednak ta wartość po wyliczeniu nie ulega zmianie, a jest wykorzystywana bardzo często, to optymalnie byłoby ją raz wyliczyć i przechowywać jako wartość określonego atrybutu. W takim przypadku należy do schematu relacji wprowadzić jeszcze dodatkowy atrybut wyliczany. Jeśli jednak istnieje możliwość, że ta wartość może ulec zmianie po jakimś czasie, to należy

w kolejnym dodatkowym atrybucie przechowywać datę wykonania wyliczenia (aktualizacji). Decyzja o tym, czy przechowywać wartość wyliczaną, czy też nie, zależy od rodzaju danej wyliczanej, od częstości jej zmian i wykorzystywania (potrzeby wyliczania). Umieszczenie w schemacie relacji atrybutu o wartości wyliczanej narusza jednak spójność bazy danych.

Jeśli dla typu encji **OCENA**(*IdWpisu*, *IdStudenta*, *IdPrzedmiotu*, *Ocena*, *Data*) na koniec roku jest wyliczana średnia ocen ze wszystkich przedmiotów dla każdego studenta i średnia ta po zakończonym roku nie ulega zmianie, natomiast jest wielokrotnie wykorzystywana, np. przy zapisach na kursy, przy przyznawaniu nagród, stypendiów, do sporządzania rankingów itd., to w schemacie relacji **Studenci** jako dodatkowy atrybut można dać *ŚrOcen*, *DataObliczeniaŚr*, ale należy to uwzględnić od początku projektu (m.in. podczas wyodrębniania kategorii i definiowaniu typu encji). Wartością atrybutu *ŚrOcen* może być np. średnia ocen za ostatni rok lub dotychczas uzyskana średnia ocen.

Jako ćwiczenie pozostawiamy Czytelnikowi przedstawienie modelu konceptualnego i przeprowadzenie transformacji do modelu logicznego odzwierciedlającego reguły: klient wypożyczalni kaset wideo musi wypożyczyć co najmniej jedną kasetę (może wiele), kaseta może być wypożyczona maksymalnie przez jednego klienta, a po zwróceniu kasety dane o wypożyczeniu są usuwane.

4.2. DEFINICJE SCHEMATÓW RELACJI

Po przeprowadzeniu transformacji (według podanych reguł), należy – podobnie do etapu 7, w którym definiowane były wszystkie typy encji – szczegółowo zdefiniować wszystkie otrzymane schematy relacji, według poniższego szablonu.

Definicja schematu relacji:

REL/*xxx* **Nazwa schematu relacji**/NAZWA TYPU ENCJI

lub

REL/*xxx* **Nazwa schematu relacji/Nazwa związku**, NAZWY TYPÓW ENCJI

gdzie *xxx* jest kolejnym numerem schematu relacji.

Szczegółowy opis atrybutów:

Nazwa atrybutu	Dziedzina	Maska	OBL	Wart. dom.	Ograniczenia	Unikalność	Klucz	Referencje	Źródło danych

Jeśli na przykład schemat relacji **Klienci** powstał z typu encji KLIENT, to po nazwie schematu relacji podajemy nazwę typu encji, z którego został utworzony, np.

REL/001 Klienci/KLIENT

Jeśli schemat relacji **Wypożyczenia** został utworzony na przykład ze związku **Wypożycza** typu N:N zachodzącego między encjami typów KLIENT i KASETA (których w projekcie nie powinno być, bo powinny zostać zastąpione związkami 1:N), to po nazwie schematu relacji należy podać nazwę związku i nazwy typów encji uczestniczących w transformowanym związku, np.

REL/002 Wypożyczenia/Wypożycza, KLIENT, KASETA

Dla wszystkich atrybutów schematów relacji należy ustalić odpowiednie formaty danych wynikające z wcześniejszych etapów projektowania.

Dla każdego schematu relacji należy podać szczegółowy opis atrybutów w postaci tabeli, uwzględniając elementy istotne dla projektowanej bazy danych:

- Atrybuty relacji
- Dziedziny atrybutów
- Maskę do wprowadzania danych
- Obligatoryjność / opcjonalność atrybutu
- Wartość domyślną atrybutu
- Unikalność wartości atrybutu
- Klucz relacji
  PK – klucz główny (ang. *primery key*)
  FK – klucz obcy (ang. *foreign key*)
- Referencje – nazwy schematów relacji, z których pochodzą klucze obce

- Źródło danych
  SZBD – System Zarządzania Bazą Danych
  BD – Baza danych
  U – Użytkownik

Informacje niezbędne do wypełnienia tabeli opisu schematu relacji powinny wynikać z wcześniejszych etapów (kategorii, ograniczeń dziedzinowych, definicji typów encji, transformacji). Wszelkie używane w opisach oznaczenia powinny być wyjaśnione i konsekwentnie stosowane.

Po zdefiniowaniu schematu relacji należy podać znaczenie wszystkich jego atrybutów.

Otrzymane po transformacji schematy relacji umożliwiają tworzenie relacji zawierających dane spełniające wszystkie wymagania i ograniczenia. Relacje te można zaimplementować w relacyjnej bazie danych w postaci tabel.

Dla każdego schematu relacji należy zamieścić **przykładowe dane**, jakie mogą wystąpić w relacji o tym schemacie (a docelowo – po implementacji – w tabeli relacyjnej bazy danych), co znacznie ułatwia analizowanie struktury bazy danych.

Dane przykładowe powinny być staranie dobrane tak, by ilustrowały sytuacje typowe, wyjątkowe, opcjonalność, ograniczenia itp. Umieszczanie przykładowych danych osobowych, w których wszystkie dane są zupełnie różne (czyli nie ma osób o takim samym nazwisku, nazwisku i imieniu, dacie urodzenia) nie jest właściwe, ponieważ nie oddaje rzeczywistości ani możliwych do wystąpienia danych. Z kolei umieszczenie wszystkich jednakowych danych różniących się tylko wartością klucza głównego także nie jest właściwe. Brak uwzględnienia danych rzadko występujących, ale jednak możliwych do wystąpienia, jest często przyczyną problemów podczas użytkowania bazy danych. Przygotowane dane przykładowe można wykorzystać do testowania bazy po jej zaimplementowaniu. Dlatego dane przykładowe w poszczególnych relacjach powinny stanowić spójną całość.

Przykład 4.19 (opis schematu relacji w modelu logicznym)

REL/001 Klienci/KLIENT

Tabela 4.2

Opis schematu relacji **Klienci**

Nazwa atrybutu	Dziedzina	Maska	OBL + OPC –	Wartość domyślna	Ograniczenia	Unikalność	Klucz	Referencje	Źródło danych
NrK	Int+		+			+	PK		SZBD
Nazwisko	String[30]		+						U
Imię	String[30]		+						U
DataUr	Date	____-__-__	–	1900-01-01	[1900-01-01, DATE()]				U
Płeć	String[1]		–		{K, M}				U
Email	String[15]		–			+			U
Hasło	String[10]		–						U
NrM	Int+		+				FK	Miasta	BD

Wykaz atrybutów schematu relacji **Klienci** i ich znaczenie przedstawiono w tabeli 4.3.

Atrybuty odpowiadające kluczom kandydującym w odpowiednim typie encji powinny mieć zapewnione unikalne wartości w relacji.

Opisy schematów relacji powinny być umieszczone w kolejności zgodnej z opisem typów encji, co ułatwia ich wyszukiwanie w dokumentacji projektu i weryfikowania poprawności. Jeśli podczas transformacji nie zostały utworzone żadne nowe schematy relacji ani nie połączono dwóch schematów w jeden, to końcowa liczba schematów powinna być równa liczbie typów encji.

Tabela 4.3

Znaczenie atrybutów w schemacie relacji **Klienci**

Nazwa atrybutu	Znaczenie atrybutu
NrK	Unikalny identyfikator klienta automatycznie nadawany przez system, klucz główny
Nazwisko	Nazwisko klienta
Imię	Pierwsze imię klienta
DataUr	Data urodzenia klienta
Płeć	Płeć klienta, K – kobieta, M – mężczyzna
Email	Email klienta pełniący rolę loginu do konta, jeśli je założył
Hasło	Hasło (szyfrowane) do konta klienta
NrM	Identyfikator miasta zamieszkania, klucz obcy

Przykładowe dane relacji o schemacie relacji **Klienci** zawiera tabela 4.4.

Tabela 4.4

Przykładowe dane relacji o schemacie relacji **Klienci**

NrK	*Nazwisko*	*Imię*	*DataUr*	*Płeć*	*Email*	*Hasło*	*NrM*
1	Koperski	Jan	1980-12-21	M	a@op.pl	62616B61736C	1
2	Us	Łukasz	1979-11-11	M	us@wp.pl	62616B61736C	1
3	Barycka	Beata	1977-03-18	K			2
4	Żółtaszek-Matacka	Maria		K			5
5	Barycka	Beata	1981-03-18	K			2
8	Koperski	Jan	1980-12-21	M			1
9	Barycka	Anna		K			1
10	Nguyen	Binh					6
11	Roman	Adam		M			2

■

4.3. RELACYJNY SCHEMAT BAZY DANYCH

Schemat bazy danych (ang. *database schema*) to zbiór obiektów (struktur) grupujących dane.

Relacyjny schemat bazy danych (ang. *relational database schema*) to zbiór schematów relacji o różnych nazwach powiązanych z sobą za pomocą kluczy głównych i kluczy obcych. Zgodnie z definicją podaną w rozdziale 2.1 **schemat relacji** (ang. *relation schema*) jest to nazwa schematu relacji oraz zbiór atrybutów relacji z zaznaczonymi atrybutami wchodzącymi w skład klucza głównego i klucza obcego.

Schemat bazy danych jest konstrukcją logiczną opisującą strukturę danych w pojęciach abstrakcyjnych. Obejmuje opisy schematów relacji (model logiczny), reprezentowane w bazie danych jako tabele, o nazwach takich samych jak schematy relacji, umożliwiające (po zaimplementowaniu) przechowywanie i przetwarzanie. Według definicji zawartej w pracy ([Subieta1999], s. 23), schemat bazy danych to „obraz zawartości bazy danych, wyrażony w sformalizowanym języku. Schemat bazy danych jest niezbędny dla jej użytkowników i programistów do zrozumienia, co baza danych zawiera i jak dane są zorganizowane".

Definicja relacyjnego schematu bazy danych:

NAZWA BAZY DANYCH
Nazwa_schematu_relacji_1(*lista_atrybutów_schematu_relacji_1*)
Nazwa_schematu_relacji_2(*lista_atrybutów_schematu_relacji_2*)
....

Poprawny relacyjny schemat bazy danych jest zbiorem schematów relacji, z których każdy jest co najmniej w trzeciej postaci normalnej.

W schematach relacji podkreślamy atrybuty, które są kluczami głównymi, a przed atrybutami, które są kluczami obcymi, wpisujemy symbol #.

Opis schematu bazy danych (metadane) jest przechowywany w tej samej bazie danych co dane właściwe.

Przykład 4.20 (przykłady schematów baz danych)

a) Po transformacji modelu konceptualnego bazy danych z rysunku 3.8 (rozdział 3.9) do modelu logicznego (zgodnie z regułami podanymi w rozdziale 4.1), otrzymalibyśmy następujący schemat relacyjnej bazy danych **WYPOŻYCZALNIA**:

WYPOŻYCZALNIA
Klienci(*NrKlienta*, *Nazwisko*, *Imię*, *Miasto*, *UlicaNr*)
Kasety(*NrKasety*, *Opis*, *#KodF*)
Filmy(*KodF*, *Tytuł*, *CzasTrwania*, *#KodG*)
Gatunki(*KodG*, *NazwaG*)
Wypożyczenia(*NrW*, *#NrKlienta*, *#NrKasety*, *DataW*, *DataZw*, *Opłata*)

Jako ćwiczenie pozostawiamy Czytelnikowi sprawdzenie, w której postaci normalnej są powyższe schematy relacji.

b) Relacyjny schemat bazy danych **FIRMA**:

FIRMA
Pracownicy(*IdP*, *Nazwisko*, *Imie*, *DataUr*, *#IdM*, *Login*, *Haslo*, *Pensja*, *Tel*, *Email*, *DataZatr*, *DataZw*)
Miasta(*IdM*, *NazwaM*)
Stanowiska(*IdS*, *NazwaS*)
Funkcje(*#IdP*, *#IdS*, *Od*, *Do*)
Klienci(*IdK*, *Nazwa1*, *Nazwa2*, *Adres*, *Tel*, *Email*, *#IdP*)
Oferty(*IdO*, *Opis*, *#IdK*)

■

Relacyjna baza danych jest to baza oparta na modelu relacyjnym. Jest zbiorem relacji o podanych schematach. Relacje są reprezentowane przez tabele (niekoniecznie fizycznie, ale tak są postrzegane). W danej chwili baza danych składa się z określonego zbioru relacji i ich instancji (zawartości), czyli jest w określonym stanie (ma określone dane). Stan bazy danych można zmienić przez wykonanie transakcji, np. aktualizację, dopisanie czy usunięcie danych. Baza danych w każdym stanie powinna spełniać wszystkie reguły funkcjonowania, czyli powinna być poprawna w sensie zgodności z rzeczywistością. Wykonywane transakcje powinny zachowywać stan integralności (ang. *state of integrity*) bazy danych.

4.4. SŁOWNIK ATRYBUTÓW

W celu ułatwienia wyszukiwania informacji o wybranym atrybucie sporządzany jest ***słownik atrybutów***. Słownikiem (ang. *dictionary*) ogólnie nazywa się wykaz terminów (pojęć) i ich objaśnień.

Słownik atrybutów – lub inaczej słownik danych (ang. *data dictionary*) – tworzy się w celu zestawienia nazw atrybutów i ich dziedzin oraz referencji, czyli nazw schematów relacji, w których występują.

Definicja słownika atrybutów:

Słownik atrybutów

Nazwa atrybutu	Dziedzina	Przynależność do schematu relacji

Słownik atrybutów przedstawiamy w postaci 3-kolumnowej tabeli, w której wyszczególniamy nazwy atrybutów relacji reprezentujących odpowiednie schematy relacji, dziedziny atrybutów oraz przynależność atrybutu do schematu relacji. Jeśli nazwa atrybutu występuje w kilku schematach relacji, to należy ją wpisać tyle razy, ile występuje ona w różnych schematach relacji. Dane w słowniku należy uporządkować alfabetycznie według nazw atrybutów, a następnie – w przypadku powtarzających się atrybutów – według nazw schematów relacji.

Słownik atrybutów pełni funkcję pomocniczą na etapie projektowania i implementacji bazy danych.

Przykład 4.21 (słownik atrybutów)

Tabela 4.5

Słownik atrybutów

Nazwa atrybutu	Dziedzina	Przynależność do schematu relacji
Cena	Int+	**Towary**
DataS	Date	**Faktury**
DataUr	Date	**Klienci**

FormaP	String[10]	**Faktury**
IdM	Int+	**Miasta**
IdTowaru	Int+	**PozycjeFaktur**
IdTowaru	Int+	**Towary**
Imie	String[15]	**Klienci**
Jednostka	String[10]	**PozycjeFaktur**
LiczbaSzt	Int+	**PozycjeFaktur**
NazwaM	String[15]	**Miasta**
NazwaT	String[25]	**Towary**
NazwiskoK	Nazwisko	**Klienci**
NazwiskoP	Nazwisko	**Pracownicy**
NrK	Int+	**Faktury**
NrK	Int+	**Klienci**
NrM	Int+	**Klienci**
Uwagi	Memo	Faktury

4.5. DEFINIOWANIE PERSPEKTYW

Ostatnim etapem projektowania bazy danych jest określenie **perspektyw** dla jej przyszłych użytkowników. W tym celu należy najpierw wyszczególnić grupy użytkowników bazy danych (zgodnie z etapem 2, rozdz. 3.2.3) oraz sprecyzować, które dane i w których transakcjach będą przez nich wykorzystywane oraz jakie transakcje (operacje) na tych danych będą mogli wykonywać (na podstawie etapu 6, rozdz. 3.6). Następnie w sposób systematyczny należy wyodrębnione perspektywy opisać.

Perspektywa (ang. *view*) w modelu relacyjnym jest to wirtualna nazwana relacja tworzona dynamicznie na każde żądanie za pomocą wyrażenia algebry relacji na

podstawie danych, do których użytkownik ma określone uprawnienia dostępu. Na podstawie uprawnień użytkownika można utworzyć wirtualny podschemat bazy danych zdefiniowany za pomocą jednego lub wielu wejściowych schematów relacji (bazowych) lub perspektyw, ich wszystkich lub niektórych atrybutów [Date2000]. Utworzony podschemat bazy danych w odróżnieniu od bazowych (wejściowych) schematów relacji nie może istnieć samodzielnie.

W wyrażeniu algebry relacji, na podstawie którego tworzona jest perspektywa, można wykorzystywać operacje algebry relacji opisane w rozdziale 2.2, np. rzutowanie, selekcję, złączenie naturalne itd.

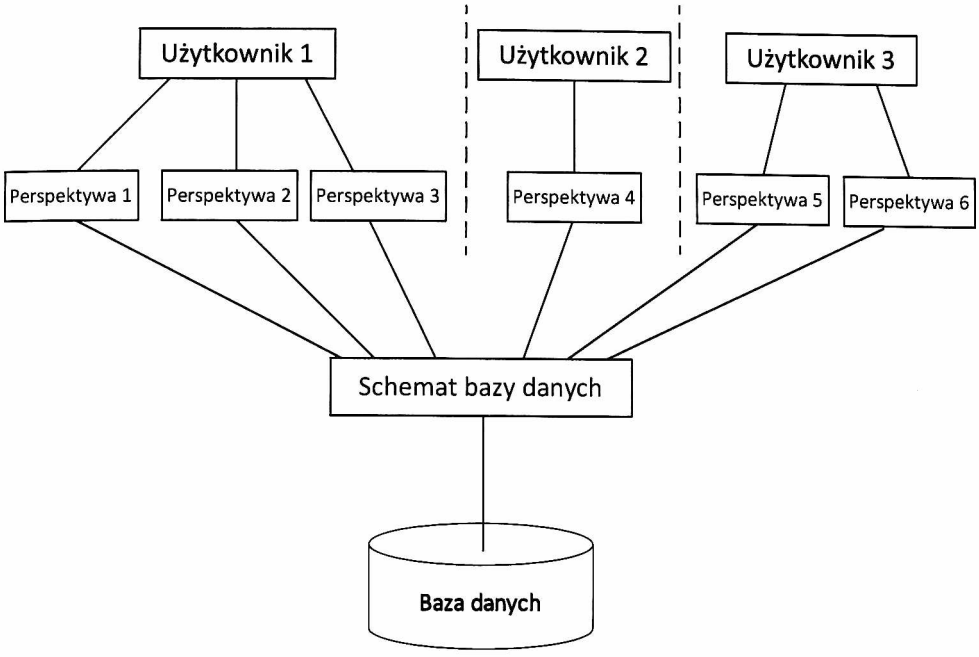
Perspektywa w szerszym znaczeniu jest to pojęciowy obraz danych, który będziemy przypisywali do jednej określonej grupy użytkowników wyodrębnionej w projekcie. Tak rozumiana perspektywa dostosowuje sposób widzenia i przedstawienia danych do konkretnych użytkowników (operujących na różnych danych z tej samej bazy danych), zapewniając jednocześnie bezpieczeństwo i niezależność przetwarzania danych. Niezwykle ważne jest zapewnienie przezroczystości (ang. *transparency)*, aby użytkownik miał wrażenie, że operuje na danych zapamiętanych, a nie wirtualnych [Subieta1999].

W zaimplementowanej bazie danych perspektywa jest to nazwana tabela (wirtualna) przechowywana w postaci zapytania tworzącego ją na każde żądanie użytkownika.

W uproszczeniu (interpretacja intuicyjna) perspektywę można traktować jako widok na dane poprzez okno (formularz), w którym są udostępniane odpowiednie atrybuty z jednej lub kilku relacji, wszystkie lub wybrane krotki, na których można wykonywać odpowiednie operacje (przeglądać, dopisywać, aktualizować, usuwać dane). Utworzone perspektywy ograniczają dostępność danych (określonych atrybutów lub całych schematów relacji) dla poszczególnych użytkowników.

Perspektywa daje możliwość zapewnienia bezpieczeństwa danych poprzez nadawanie użytkownikom uprawnień dostępu i uprawnień do wykonywania operacji na określonych schematach relacji lub wybranych atrybutach schematów relacji, czyli na utworzonym podschemacie bazy. Perspektywy są mocnym narzędziem służącym do ograniczania dostępu do danych.

Postrzeganie danych poprzez perspektywy zdefiniowane dla użytkowników przedstawiono na rysunku 4.20. Wszystkie osoby należące do określonej **grupy użytkowników** mają te same uprawnienia i dostęp do danych określonych przez perspektywy dla tej grupy, np. grupę użytkowników Pracownik stanowią wszyscy użytkownicy należący do tej grupy.



Rys. 4.21. Niezależność danych w perspektywach użytkowników

Z definicji perspektywy dla danego użytkownika wynika, do których danych ma on dostęp i jakie ma uprawnienia do tych danych (zapis, odczyt, modyfikacja).

W etapie definiowania perspektyw należy:

- wyszczególnić użytkowników bazy danych,
- wyszczególnić perspektywy dla każdej grupy użytkowników,
- określić schematy relacji i atrybuty udostępniane w danej perspektywie,
- wyszczególnić transakcje dla każdej perspektywy,
- określić dane uczestniczące w transakcji – nazwę schematu relacji, nazwę atrybutu oraz dopuszczalne operacje. Opisy przedstawiane są w formie tabeli posiadającej nagłówek:

  Nazwa schematu relacji, Atrybut, Zapis, Odczyt, Modyfikacja

  Dla każdego atrybutu symbolem + zaznaczana jest możliwość wykonania danej operacji (zapis, odczyt, modyfikacja).

Definicje perspektyw powinny być uporządkowane według zdefiniowanych grup użytkowników, czyli najpierw powinny być definicje wszystkich perspektyw dla pierwszej grupy użytkowników, potem dla drugiej itd. Dla każdej grupy użytkowników można zdefiniować wiele perspektyw, ale każda perspektywa

jest przypisana tylko do jednej grupy użytkowników. Takie podejście ułatwia implementację aplikacji dla poszczególnych grup użytkowników.

Nazwy perspektyw muszą być unikalne.

Definicja perspektywy:

PER/*xxx* **Nazwa perspektywy**
Użytkownik: Nazwa użytkownika
Podschemat bazy danych dla perspektywy:
Nazwa schematu relacji_1(lista atr)
Nazwa schematu relacji_2(lista atr)
Transakcje: Lista transakcji postaci TRA/*xxx*
Tabela transakcji (dla każdej transakcji wymienionej w danej perspektywie):
TRA/*xxx* Nazwa transakcji

Nazwa schematu relacji	Atrybut	Zapis	Odczyt	Modyfikacja

Przykład 4.22

Załóżmy, że użytkownikiem bazy danych jest sekretarka, która może dopisywać, edytować i wyświetlać dane pracowników i miast, w których oni mieszkają. Opis perspektywy dotyczącej obsługi danych pracownika przez sekretarkę może mieć postać:

PER/001 **Dane pracownika**
Użytkownik: Sekretarka
Podschemat bazy danych dla perspektywy:
Osoby(*IdP, Nazwisko, Imie, DataUr, #IdM, Login, Haslo, DataZatr, DataZw*)
Miasta(*IdM, NazwaM*)
Dzieci(*IdDz, NazwiskoDz, ImieDz, DataUr, #IdO*)
Transakcje: TRA/001, TRA/002, TRA/005

TRA/001 Wprowadzenie danych pracownika

Tabela 4.6

Tabela transakcji: Wprowadzenie danych pracownika

Nazwa schematu relacji	Atrybut	Zapis	Odczyt	Modyfikacja
Osoby	*IdP*	+		
Osoby	*Nazwisko*	+		
Osoby	*Imie*	+		
Osoby	*DataUr*	+		
Osoby	*IdM*	+		
Osoby	*Login*	+		
Osoby	*Haslo*	+		
Miasta	*IdM*	+	+	
Miasta	*NazwaM*	+	+	

TRA/002 Edycja danych pracownika

Tabela 4.7

Tabela transakcji: Edycja danych pracownika

Nazwa schematu relacji	Atrybut	Zapis	Odczyt	Modyfikacja
Osoby	*IdO*		+	
Osoby	*Nazwisko*		+	+
Osoby	*Imie*		+	+
Osoby	*DataUr*		+	+
Dzieci	*IdDz*	+	+	
Dzieci	*NazwiskoDz*	+	+	+
Dzieci	*ImieDz*	+	+	+
Dzieci	*DataUrDz*	+	+	+
Dzieci	*IdO*	+	+	+

Uwaga. Symbol + w kolumnie **Zapis** powinien występować tylko dla atrybutów opcjonalnych, które mogą mieć nadawane wartości dopiero na etapie edycji danych. Atrybuty obligatoryjne muszą mieć przypisane wartości przy wprowadzaniu danych, a podczas edycji mogą być ewentualnie modyfikowane.

TRA/005 Wyświetlenie danych pracownika

Tabela 4.8

Tabela transakcji: Wyświetlenie danych pracownika

Nazwa schematu relacji	Atrybut	Zapis	Odczyt	Modyfikacja
Osoby	*IdO*		+	
Osoby	*Nazwisko*		+	
Osoby	*Imie*		+	
Osoby	*DataUr*		+	
Dzieci	*IdDz*		+	
Dzieci	*NazwiskoDz*		+	
Dzieci	*ImieDz*		+	
Dzieci	*DataUrDz*		+	
Dzieci	*IdO*		+	

■

Przykład 4.23

Dla danego schematu bazy danych **KADRY**

Pracownicy(*IdP, Nazwisko, Imie, DataUr, #IdM, Login, Haslo, Pensja, Tel, Email, DataZatr, DataZw*)
Miasta(*IdM, NazwaM*)
Stanowiska(*IdS, NazwaS*)
Funkcje(*#IdP, #IdS, Od, Do*)
Klienci(*IdK, Nazwa1, Nazwa2, Adres, Tel, Email, #IdP*)
Oferty(*IdO, Opis, #IdK*)

i użytkowników:

- Kierownik
- Pracownik
- Sekretarka

zdefiniujmy przykładowe perspektywy.

PER/001 Dane pracowników
Użytkownik: Sekretarka
Dane bazowe dla perspektywy:

Pracownicy(*IdP*, *Nazwisko*, *Imie*, *DataUr*, *#IdM*, *Login*, *Haslo*, *DataZatr*, *DataZw*)
Miasta(*IdM*, *NazwaM*)

Transakcje: TRA/002, TRA/004, TRA/007

TRA/002 Dopisz dane pracownika

Tabela 4.9

Tabela transakcji: Dopisz dane pracownika

Nazwa schematu relacji	Atrybut	Zapis	Odczyt	Modyfikacja
Pracownicy	*IdP*	+		
Pracownicy	*Nazwisko*	+		
Pracownicy	*Imie*	+		
Pracownicy	*DataUr*	+		
Pracownicy	*IdM*	+		
Pracownicy	*Login*	+		
Pracownicy	*Haslo*	+		
Miasta	*IdM*	+	+	
Miasta	*NazwaM*	+	+	

TRA/004 Edytuj dane pracownika

Tabela 4.10

Tabela transakcji: Edytuj dane pracownika

Nazwa schematu relacji	Atrybut	Zapis	Odczyt	Modyfikacja
Pracownicy	*IdP*		+	
Pracownicy	*Nazwisko*		+	+
Pracownicy	*Imie*		+	+
Pracownicy	*DataUr*		+	+
Pracownicy	*IdM*	+	+	+
Miasta	*IdM*	+	+	
Miasta	*NazwaM*	+	+	

Uwaga. Symbol + w kolumnie Zapis dla atrybutów *Miasta.IdM* i *Miasta.NazwaM* oznacza, że na etapie edycji danych pracownika dopuszcza się dopisanie nowego miasta, jeśli należy przypisać je pracownikowi, a nie ma tego miasta w bazie.

TRA/007 Przeglądaj dane pracowników

Tabela 4.11

Tabela transakcji: Przeglądaj dane pracowników

Nazwa schematu relacji	Atrybut	Zapis	Odczyt	Modyfikacja
Pracownicy	*IdP*		+	
Pracownicy	*Nazwisko*		+	
Pracownicy	*Imie*		+	
Pracownicy	*DataUr*		+	
Pracownicy	*IdM*		+	
Pracownicy	*Login*		+	
Pracownicy	*DataZatr*		+	
Pracownicy	*DataZw*		+	
Miasta	*IdM*		+	
Miasta	*NazwaM*		+	

PER/002 Dane klientów dla sekretariatu
Użytkownik: Sekretarka
Dane bazowe dla perspektywy:
Pracownicy(*IdP*, *Nazwisko*, *Imie*, *Tel*, *Email*)
Klienci(*IdK*, *Nazwa1*, *Nazwa2*, *Adres*, *Tel*, *Email*, *#IdP*)
Transakcje: TRA/006

TRA/006 Przeglądaj dane klientów

Tabela 4.12

Tabela transakcji: Przeglądaj dane klientów

Nazwa schematu relacji	Atrybut	Zapis	Odczyt	Modyfikacja
Klienci	*IdK*		+	
Klienci	*Nazwa1*		+	
Klienci	*Nazwa2*		+	
Klienci	*Adres*		+	
Klienci	*Tel*		+	
Klienci	*Email*		+	
Klienci	*IdP*		+	
Pracownicy	*IdP*		+	
Pracownicy	*Nazwisko*		+	
Pracownicy	*Imie*		+	
Pracownicy	*Tel*		+	
Pracownicy	*Email*		+	

PER/003 Dane klientów

Użytkownik: Kierownik

Dane bazowe dla perspektywy:

Pracownicy(*IdP*, *Nazwisko*, *Imie*, *DataUr*, *#IdM*, *Login*, *Haslo*, *DataZatr*, *DataZw*)

Klienci(*IdK*, *Nazwa1*, *Nazwa2*, *Adres*, *Tel*, *Email*, *#IdP*)

Oferty(*IdO*, *Opis*, *#IdK*)

Transakcje: TRA/0015, TRA/017, TRA/018, TRA/020

TRA/015 Usuń dane klienta

Tabela 4.13

Tabela transakcji: Usuń dane klienta

Nazwa schematu relacji	Atrybut	Zapis	Odczyt	Modyfikacja
Klienci	*IdK*		+	
Klienci	*Nazwa1*		+	
Klienci	*Nazwa2*		+	
Klienci	*Adres*		+	
Klienci	*Tel*		+	
Klienci	*Email*		+	
Oferty	*IdO*		+	
Oferty	*Opis*		+	
Oferty	*IdK*		+	

Uwaga. Przed usunięciem danych klienta są one odczytywane w celu upewnienia się, że są usuwane odpowiednie dane, i następuje sprawdzenie, czy klient nie ma powiązanych ofert. W zależności od opisu transakcji (etap 6) usuwane są oferty klienta (usuwanie kaskadowe) lub nie można usunąć danych klienta, który ma oferty (usuwanie ograniczone).

■

Po zdefiniowaniu perspektyw dla wszystkich użytkowników należy zweryfikować poprawność i kompletność przypisania transakcji do perspektyw użytkowników.

Każda transakcja zdefiniowana w etapie 6 musi być przypisana co najmniej do jednego użytkownika.

Każda transakcja może być związana z wieloma użytkownikami, ale z jednym użytkownikiem może być związana tylko jeden raz w ramach jego wszystkich perspektyw.

Za pomocą macierzy powiązań transakcji z perspektywami użytkowników (tabela 4.14) należy sprawdzić, czy każda transakcja zdefiniowana w etapie 6 występuje chociaż w jednej perspektywie i czy jest to perspektywa użytkownika, który jest uprawniony do jej wykonywania.

Tabela 4.14

Macierz powiązań transakcji z perspektywami użytkowników

	U1			U2	U3	
	PER/001	**PER/002**	**PER/003**	**PER/004**	**PER/005**	**PER/006**
TRA/001	x			x	x	
TRA/002		x		x		
TRA/003	x					
TRA/004				x		
TRA/005					x	
TRA/006	x					
TRA/007		x		x		
TRA/008			x			x
TRA/009			x			
TRA/010			x			

Poprawnie wykonany relacyjny model logiczny w postaci relacyjnego schematu bazy danych i zbioru perspektyw dla wszystkich użytkowników uwzględniających zdefiniowane transakcje kończy projektowanie relacyjnej bazy danych.

Wszystko, co jest już gotowe, wygląda tak bardzo prosto.

Autor nieznany

5. ZAKOŃCZENIE

W książce omówiono relacyjny model danych, podstawowe operacje algebry relacji i normalizację schematów relacji. Przedstawiono także pełny cykl tworzenia projektu relacyjnej bazy danych w autorskiej strukturalnej metodyce projektowej. Aby wykonać poprawny projekt bazy danych, należy prawidłowo wykonać poszczególne etapy, dokładnie je dokumentując. Projekt można uznać za poprawny, jeśli jego opis jest zgodny z przyjętą notacją, która musi być na początku projektu przedstawiona, a następnie konsekwentnie stosowana. Poprawność projektu nie oznacza i nie zapewnia, że system, który na bazie tego projektu zbudujemy, będzie spełniał oczekiwania użytkownika.

Poprawny projekt powinien być [Jaszk1997]:

- **kompletny** – wszystkie składowe projektu są zdefiniowane w sposób odpowiedni i wyczerpujący,
- **niesprzeczny** – każdy element jest zdefiniowany jednoznacznie, nie ma definicji wzajemnie sprzecznych,
- **spójny** – informacje zawarte w kolejnych etapach projektu (definicjach, diagramach) są semantycznie zgodne,
- **zgodny z regułami składniowymi notacji** – opis projektu zależy od przyjętej notacji, z którą musi być zgodny.

Aby osiągnąć projekt wysokiej jakości, poszczególne etapy projektowania bazy danych należy nieraz wykonywać wielokrotnie, poprawiać opisy, uzupełniać je

i uszczegóławiać. Opracowaną dokumentację należy analizować wielokrotnie, zwracając uwagę na cztery wymienione wyżej cechy (kompletność, niesprzeczność, spójność i zgodność). W celu zapewnienia poprawności modelu danych w sensie zgodności często podaje się *metamodel danych*, czyli model modelu. Metamodel obejmuje pojęcia i reguły dotyczące pojęć wykorzystywanych w modelu, na przykład metamodel może zawierać następującą regułę: jeśli model zawiera co najmniej dwie encje, to każda encja musi być co najmniej w jednym związku z dowolną inną encją.

Za pomocą modelu relacyjnego definiuje się logiczną strukturę danych (w postaci schematów relacji i powiązań między nimi realizowanych za pomocą kluczy obcych) oraz transakcje określające operacje wykonywane na danych przez uprawnionych użytkowników.

Do projektowania baz danych można wykorzystać narzędzia CASE, np. Visual Paradigm, Enterprise Architect, IBM Rational Rose, które automatyzują i przyspieszają proces projektowania, a których opis wykracza poza przyjęte ramy niniejszej pracy.

W książce opisano modelowanie danych, przedstawiono kolejne etapy projektowania bazy danych opartego głównie na identyfikacji encji i związków.

Do tworzenia schematu bazy danych można podejść również w nieco inny sposób – metodą normalizacji, w której usuwa się anomalie występujące w schematach relacji przez odpowiednie operacje projekcji. Na podstawie analizy wycinka rzeczywistości można określić jeden globalny (początkowy, wyjściowy) schemat relacji $\mathbf{R}(A_1, A_2, \ldots, A_n)$, w którym uwzględnia się wszystkie atrybuty występujące w bazie danych. Baza danych traktowana jest więc jako jedna (być może bardzo duża) relacja. W związku z tym wszystkie nazwy atrybutów muszą być jednoznaczne. Równolegle z wyodrębnianiem atrybutów należy określić zależności funkcyjne między atrybutami. W kolejnych krokach normalizuje się początkowy schemat relacji, uzyskując docelowy (końcowy) schemat bazy danych, który jest zbiorem schematów relacji. Schemat ten powinien być lepszy od początkowego schematu relacji.

Jako kryterium oceny schematów baz danych przyjmuje się stopień normalizacji wynikowych schematów relacji oraz liczbę schematów relacji. W praktyce wymaga się, aby schematy relacji były co najmniej w trzeciej postaci normalnej, a ich liczba była jak najmniejsza. Istnieje kilka metod projektowania schematu bazy danych, np. algorytm dekompozycji, Bernsteina, Niekludowej-Calenki, Rissanena, których opisy można znaleźć na przykład w pracy [Pankow1992].

Schemat bazy danych otrzymany w wyniku normalizacji powinien być w zasadzie taki sam jak schemat bazy otrzymanej z transformacji diagramu

związków encji do modelu logicznego. Wynika to stąd, że wyodrębnianie encji i związków odpowiada intuicyjnie normalizacji schematów relacji. Sprowadzanie związków wieloznacznych do binarnych i ograniczanie się do wartości prostych dozwolonych w dziedzinach atrybutów prowadzi do zaprojektowania znormalizowanej bazy danych. W praktyce częściej używa się modelowania danych niż normalizacji, a normalizację wykorzystuje się zazwyczaj do sprawdzenia poprawności otrzymanych schematów relacji. Według C.J. Date większa część procesu projektowania bazy danych jest bardziej sztuką niż procesem naukowym [WhMa2002]. W tym bardzo subiektywnym procesie elementem naukowym jest normalizacja.

Dysponując poprawnym projektem bazy danych, można przystąpić do jej implementacji oraz wykonania projektu i implementacji współpracujących z bazą aplikacji [Banach1998, CasPal2000, Górski2000, MazMaz1997, Riordan2000, Roman 2001, W&W1997].

6. LITERATURA

[Allen2003] Allen Sharon, *Modelowanie danych*, Helion, Gliwice, 2003.

[Banach1998] Banachowski Lech, *Bazy danych. Tworzenie aplikacji*, Akademicka Oficyna Wydawnicza PLJ, Warszawa 1998.

[Barker1996] Barker Richard, *CASE*Method, Modelowanie związków encji,* WNT, Warszawa 1996.

[Beynon1999] Beynon-Davies Paul, *Inżynieria systemów informacyjnych*, WNT, Warszawa 1999.

[Beynon2000] Beynon-Davies Paul, *Systemy baz danych*, WNT, Warszawa 2000.

[Boratyn1995] Boratyn Dariusz, *MS ACCESS 2.0*, Croma&Msoft&DK, Wrocław 1995.

[CasPal2000] Cassel Paul, Palmer Pamela, *Access 2000 PL dla każdego*, Helion, Gliwice 2000.

[Cell1988] Cellary Wojciech, Królikowski Zbyszko, *Wprowadzenie do projektowania baz danych, dBase III*, WNT, Warszawa 1988.

[Chen1976] Chen Peter Pin-Shan, *The Entity-Relationship Model – Toward a Unified View of Data*, ACM TODS 1, No. 1, March 1976. Przedruk [w:] M. Stonebraker (ed.), *Reading In Database Systems*, San Mateo, Calif.: Morgan Kaufmann, 1988.

[Codd1970] Codd E.F., *A Relational Model of Data for Large Shared Data Banks*, Comm. ACM, vol. 13, No. 6, June 1970.

[Codd1972] Codd E.F., *Relational Completeness of Data Base Sublanguages*, [w:] *Data Base Systems, Courant Computer Science Symposia Series 6. Englewood Cliffs, N.J.*, Prentice-Hall, 1972.

[Codd1985a] Codd E.F., *Is Your DBMS Really Relational*? [w:] *Computerworld*, 14 October 1985.

[Codd1985b] Codd E.F., *Does your DBMS run by the rules*? [w:] *Computerworld*, 21 October 1985.

[ConBeg2004] Connolly Thomas, Begg Carolyn, *Systemy baz danych. Praktyczne metody projektowania, implementacji i zarządzania*, tom 1, RM, Warszawa 2004.

[Celko2016] Celko Joe, *Praktyki mistrza SQL. Programowanie zaawansowane*, Helion, Gliwice 2016.

[Date1981] Date C.J., *Wprowadzenie do baz danych*, WNT, Warszawa 1981.

[Date2000] Date C.J., *Wprowadzenie do systemów baz danych*, WNT, Warszawa 2000.

[DateDar2000] Date C.J., Darwen Hugh, *SQL. Omówienie standardu języka*, WNT, Warszawa 2000.

[DelAdi1989] Delobel Claude, Adiba Michel, *Relacyjne bazy danych*, WNT, Warszawa 1989.

[ElNav2005] Elmasri Ramez, Navathe Shamkant B., *Wprowadzenie do systemów baz danych*, Helion, Gliwice 2005.

[FugStąTr1995] Fuglewicz Piotr, Stąpor Katarzyna, Trojnar Andrzej, *CASE dla ludzi*, LUPUS, Warszawa 1995.

[GaUlWi2003] Garcia-Molina Hector, Ullman Jeffrey D., Widom Jennifer, *Implementacja systemów baz danych*, WNT, Warszawa 2003.

[GaUlWi2006] Garcia-Molina Hector, Ullman Jeffrey D., Widom Jennifer, *Systemy baz danych. Pełny wykład*, WNT, Warszawa 2006.

[Górski2000] Praca zbiorowa pod redakcją Janusza Górskiego, *Inżynieria oprogramowania w projekcie informatycznym*, MIKOM, Warszawa 2000.

[Harrin2001] Harrington Jan L., *Obiektowe bazy danych dla każdego*, MIKOM, Warszawa 2001.

[Harris1994] Harris Wayne, *Bazy danych nie tylko dla ludzi biznesu*, WNT, Warszawa 1994.

[Hernan1998] Hernandez Michael J. *Bazy danych dla zwykłych śmiertelników*, MIKOM, Warszawa 1998.

[Jaszk1997] Jaszkiewicz Andrzej, *Inżynieria oprogramowania*, Helion, Gliwice 1997.

[MazMaz1997] Mazur Hanna, Mazur Zygmunt, *Programowanie w dBASE IV*, PSZI, Wrocław 1997.

[MazMaz2004] Mazur Hanna, Mazur Zygmunt, *Projektowanie relacyjnych baz danych*, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2004.

[MaNoRaSz2020] Mazur Hanna, Nowak Mikołaj, Raczycki Kamil, Szenborn Filip, *Internetowy system do obsługi konkursu na najlepsze prace magisterskie z informatyki*, https://kpm.pti.org.pl, Wrocław 2020.

[Mazur2017] Mazur Paweł, *Strona WWW konkursu prac magisterskich*, http://pti.wroc.pl/html/konkurs_poprzednia_edycja.php, Wrocław 2017.

[Muller2000] Muller Robert J., *Bazy danych. Język UML w modelowaniu danych*, MIKOM, Warszawa 2000.

[MurRyb1993] Muraszkiewicz Mieczysław, Rybiński Henryk, *Bazy danych*, Akademicka Oficyna Wydawnicza, Warszawa 1993.

[OleńStan1984] Oleński Józef, Staniszkis Witold, *Projektowanie bazy danych*, seria „Informatyka w praktyce", Państwowe Wydawnictwo Ekonomiczne, Warszawa 1984.

[Olle1982] Olle T.W., Sol H.G., Verrijn-Stuart A.A. (ed), *Information Systems Design Methodologies. A Comparative Review*, Amsterdam, Netherlands: North-Holand / New York, NY: Elsevier Science, 1982.

[Pankow1992] Pankowski Tadeusz, *Podstawy baz danych*, PWN, Warszawa 1992.

[Riordan2000] Riordan Rebecca M., *Projektowanie systemów relacyjnych baz danych*, RM, Warszawa 2000.

[Rodgers1995] Rodgers Ulka, *ORACLE. Przewodnik projektanta baz danych*, WNT, Warszawa 1995.

[Roman2001] Roman Steven, *Access. Baza Danych – Projektowanie i programowanie*, Helion, Gliwice 2001.

[Roszko1998] Roszkowski Jerzy, *Analiza i projektowanie strukturalne*, Helion, Gliwice 1998.

[Sacha2010] Sacha Krzysztof, *Inżynieria oprogramowania*, Wydawnictwo Naukowe PWN, Warszawa 2010.

[Subieta1998] Subieta Kazimierz, *Obiektowość w projektowaniu i bazach danych*, Akademicka Oficyna Wydawnicza PLJ, Warszawa 1998.

[Subieta1999] Subieta Kazimierz, *Słownik terminów z zakresu obiektowości*, Akademicka Oficyna Wydawnicza PLJ, Warszawa 1999.

[TsiLoch1990] Tsichritzis Dionysios C., Lochovsky Frederick H., *Modele danych*, WNT, Warszawa 1990.

[UllWid2000] Ullman Jeffrey D., Widom Jennifer, *Podstawowy wykład z systemów baz danych*, WNT, Warszawa 2000.

[UllWid2011] Ullman Jeffrey D., Widom Jennifer, *Podstawowy kurs systemów baz danych*, Helion, Gliwice 2011.

[Ustawa2001] *Ustawa z dnia 27 lipca 2001 r. o ochronie baz danych*, Dz. U. 2001 Nr 128 poz. 1402.

[W&W1997] Struzińska-Walczak Anna, Walczak Krzysztof, *Delphi. Nauka programowania systemów baz danych*, Wydawnictwo W&W, Warszawa 1997.

[WhMa2002] Whitehorn Mark, Marklyn Bill, *Relacyjne bazy danych*, Helion, Gliwice 2002.

[Yourdon1996] Yourdon Edward, *Współczesna analiza strukturalna*, WNT, Warszawa 1996.

DODATEK A
DEFINICJE BAZY DANYCH

W niniejszej książce omówiono zagadnienia związane z projektowaniem baz danych. Termin „baza danych" jest więc podstawowym pojęciem występującym zarówno w tej książce, jak i w literaturze związanej z tematyką baz danych. Coraz częściej słyszymy o istniejących lub tworzonych bazach danych w różnych dziedzinach życia, np. baza danych pracowników, szkół i uczelni, bezrobotnych, klientów banku, świadczeniobiorców Zakładu Ubezpieczeń Społecznych itd. Intuicyjnie każdy rozumie, co to jest baza danych, lecz w literaturze brak jest jednoznacznej definicji pojęcia bazy danych. Przytoczmy cytat z pracy E.F. Codda [Codd1970], który bazę danych nazywał bankiem danych (ang. *data bank*):

„The totality of data in a data bank may be viewed as a collection of time-varying relations. These relations are of assorted degrees",

czyli baza danych została tu zdefiniowana jako zbiór relacji zmieniających się w czasie, a relacje mają różne stopnie.

Poniżej podajemy różne określenia bazy danych zaczerpnięte z prac dostępnych na polskim rynku. Cytaty przytoczono w kolejności chronologicznej według roku wydania pracy, z której pochodzi dany cytat.

Cytat 1 [OleńStan1984, s. 11]

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

Cytat 2 [Cell1988, s. 7]

„Przez *bazę danych* rozumiemy uporządkowany zbiór danych przechowywanych w pamięci komputera, a przez *system bazy danych* – bazę danych wraz ze środkami programowymi umożliwiającymi operowanie na niej, w szczególności wyszukiwanie i aktualizowanie zawartych w niej informacji".

Cytat 3 [DelAdi1989, s. 16]

„Bazą danych nazywamy zbiór danych o określonej strukturze, zapisany na zewnętrznym nośniku pamięciowym komputera, mogący zaspokoić potrzeby wielu użytkowników korzystających z niego w sposób selektywny w dogodnym dla siebie czasie.

W bazie danych są zapisywane fakty i zdarzenia zachodzące w pewnym wycinku rzeczywistości, po to, by możliwe było ich odtworzenie i analiza w dowolnej chwili".

Cytat 4 [TsiLoch1990, s. 23]

„W konkretnym zastosowaniu modelu danych nazwy kategorii wraz z ich cechami nazywamy *schematem*. (...)

Kolekcja danych umieszczonych w określony sposób w strukturach odpowiadająca schematowi jest powszechnie nazywana *bazą danych*".

Cytat 5 [Pankow1992, s. 17]

„Aktywna rola systemu informatycznego, w systemie informacyjno-decyzyjnym wymaga aby:

- był w nim odwzorowany pewien zakres wiedzy o środowisku (miejsce tego odwzorowania nazywamy *bazą danych*, BD, systemu informatycznego);
- zapewnione były środki umożliwiające utrzymanie, aktualizowanie i udostępnianie odwzorowanej wiedzy (oprogramowanie realizujące te funkcje nazywamy systemem zarządzania bazą danych, SZBD).

Zagadnienia odwzorowania wiedzy są rozległą dziedziną badań naukowych i wykraczają poza zakres zagadnień poruszanych w tej książce. Zagadnienia te są przedmiotem badań sztucznej inteligencji, teorii języków i in. My ograniczymy się tylko do takiego zakresu wiedzy, który da się odwzorować w tzw. *sformatowanych*

bazach danych, a więc do takiej wiedzy, dla przekazania której można wcześniej ustalić pewien skończony zbiór formatów (wzorców). W dalszym ciągu mówiąc o bazach danych, będziemy mieli na myśli sformatowane bazy danych".

Cytat 6 [MurRyb1993, s. 6]

„Bazy danych, którymi zajmujemy się w tej książce, są komputerowymi reprezentacjami fragmentów istniejącego (niekoniecznie fizycznie) świata rzeczywistego, który jest przedmiotem zainteresowania użytkowników baz".

Cytat 7 [Barker1996, s. 225]

„**Baza danych** – dowolny zbiór tabel lub plików, będący pod kontrolą systemu zarządzania bazą danych".

Cytat 8 [Banach1998, s. 5]

„Informacje (dane) są takim samym zasobem firmy jak każdy inny (np. pracownicy, materiały, urządzenia) i wymagającym, tak samo jak one, zarządzania. Informacja może mieć wartość dla firmy tylko wtedy, gdy jest dokładna i dostępna wtedy, kiedy trzeba. *Baza danych* stała się standardową metodą strukturalizacji zarządzania informacją w większości organizacji.

Baza danych dotyczy zawsze pewnego fragmentu rzeczywistości związanego z firmą, organizacją lub innym działającym systemem. Stanowi kolekcję danych, których zadaniem jest reprezentowanie tego fragmentu rzeczywistości. Baza danych jest na ogół częścią systemu informacyjnego, obsługującego zapotrzebowania informacyjne pewnego fragmentu rzeczywistości. Bardziej technicznymi terminami używanymi w zastępstwie terminu system informacyjny w celu podkreślenia jego implementacyjnego charakteru są:

- *aplikacja bazy danych*, gdy chcemy zwrócić szczególną uwagę na charakter programistyczny i szczególne miejsce danych w całej aplikacji;
- *system informatyczny*, gdy chcemy zwrócić uwagę na ogólne środki informatyczne a więc także używany sprzęt.

Z pojęciem bazy danych są nierozerwalnie związane dwie cechy:

1. *trwałość* – dane mają być przechowywane przez pewien okres czasu, na ogół nieokreślony z góry, w odróżnieniu od danych chwilowych – istniejących tylko w momencie działania programu;
2. *zgodność z rzeczywistością* – dane w bazie danych muszą stanowić wierne odzwierciedlenie modelowanego fragmentu rzeczywistości, baza danych też

musi się odpowiednio zmieniać. Zapewnienie zgodności z rzeczywistością jest podstawowym procesem we współczesnych systemach informacyjnych korzystających z baz danych".

Cytat 9 [Subieta1999, s. 23]

„**Baza danych** to zestaw danych, metadanych (katalogów), programów i innych środków pozwalających na utrzymywanie, zabezpieczanie, przetwarzanie i udostępnianie danych dla użytkowników. Bazy danych są podtrzymywane przez systemy zarządzania bazą danych (SZBD). Mogą one być oparte o pewien model danych (np. hierarchiczny, sieciowy, relacyjny, funkcjonalny, obiektowy) lub mogą stosować rozwiązanie własne, nie mieszczące się w ramach zdefiniowanych modeli. Często bazą danych określa się także pewien system plików, pewien zestaw stron WWW, repozytorium dokumentów (np. pod LotusNotes) oraz inne środki przechowywania i udostępniania danych".

Cytat 10 [Beynon2000, s. 25]

„**Bazę danych** możemy przyrównać do kartoteki z aktami, a ściślej do zbioru kartotek. Baza danych jest magazynem danych z nałożoną na niego wewnętrzną strukturą. Ogólnym celem takiego magazynu jest przechowywanie danych związanych z pewnym zbiorem zadań organizacyjnych. Zwykle takie zadania dotyczą administracji. Celem większości systemów baz danych jest przechowywanie danych potrzebnych do wykonywania codziennej działalności organizacji. Dlatego na uniwersytecie baza danych jest potrzebna przy takiej czynności, jak rejestrowanie liczby studentów zapisanych na poszczególne moduły.

Struktura powoduje zwykle pewien logiczny podział, zazwyczaj hierarchiczny. Dlatego mówimy o bazie danych jak o zbiorze plików. Zbiór plików zawierających informacje na temat studentów na uniwersytecie stanowiłby bazę danych. Każdy plik w bazie danych jest również strukturalnym zbiorem danych. Odwołując się do porównania, takie pliki byłyby teczkami trzymanymi w kartotece. Każdy plik w kartotece składa się ze zbioru rekordów. Mogą to być karty z informacjami, na przykład o każdym studencie. Każdy rekord jest z kolei podzielony na zbiór obszarów nazywanymi polami, np. NrStudenta, NazwiskoStudenta, DataUrodzenia itd. W każdym polu jest wpisana konkretna wartość.

Pytanie o strukturę systemu bazy danych jest więc sprawą największej wagi. Z bazą danych są także powiązane inne właściwości: współdzielenie danych, integracja danych, integralność danych, bezpieczeństwo danych, abstrakcja danych i niezależność danych".

Cytat 11 [Date2000, s. 31]

„**Baza danych** jest to zbiór danych trwałych, które są wykorzystywane przez system aplikacji danej organizacji (lub przedsiębiorstwa)".

Cytat 12 [Górski2000, s. 252]

„Zbiór danych zorganizowany zgodnie z określonym *modelem danych* nazywamy *bazą danych*. Jeśli jeden system DBMS zarządza wieloma takimi zbiorami, mówimy o *systemie baz danych*".

Cytat 13 [UllWid2000, s. 19]

„**Baza danych** nie jest w istocie rzeczy niczym więcej niż zbiorem danych istniejącym przez długi czas, często przez wiele lat. W potocznym rozumieniu termin baza danych odnosi się do zbioru danych zorganizowanego przez *system zarządzania bazą danych*, który nazywa się również skrótowo DBMS (*database management system*) lub po prostu systemem baz danych".

Cytat 14 [Ustawa2001, Art.2.1]

„W rozumieniu ustawy: 1) **baza danych** oznacza zbiór danych lub jakichkolwiek innych materiałów i elementów zgromadzonych według określonej systematyki lub metody, indywidualnie dostępnych w jakikolwiek sposób, w tym środkami elektronicznymi, wymagający istotnego, co do jakości lub ilości, nakładu inwestycyjnego w celu sporządzenia, weryfikacji lub prezentacji jego zawartości",

Cytat 15 [Harrin2001, s. 253]

„**Baza danych** – kolekcja obiektów zarządzanych w taki sposób, że mogą być dostępne dla wielu użytkowników".

Cytat 16 [ConBeg2004, s. 14]

„**Baza danych** – Dostępny dla wielu użytkowników zbiór powiązanych logicznie danych wraz z definicją ich struktury, zaprojektowany dla zaspokojenia potrzeb przetwarzania danych przez instytucję".

Cytat 17 [ElNav2005, s. 22]

„**Baza danych** jest zbiorem powiązanych ze sobą danych".

W niniejszej książce bazę danych zdefiniowano jako trwały zbiór tematycznie z sobą powiązanych danych (rozdz. 2).

DODATEK B
WYKAZ ETAPÓW W STRUKTURALNEJ METODYCE PROJEKTOWANIA BAZ DANYCH

Model konceptualny

Etap 1. Zdefiniowanie tematu, celu, zakresu i użytkowników bazy danych
Etap 2. Analiza rzeczywistości
Etap 3. Zdefiniowanie kategorii
Etap 4. Wyodrębnienie reguł funkcjonowania
Etap 5. Wyodrębnienie ograniczeń dziedzinowych
Etap 6. Zdefiniowanie transakcji
Etap 7. Identyfikacja typów encji i typów związków
Etap 8. Definicje predykatowe typów encji i typów związków
Etap 9. Diagram związków encji

Model logiczny

Etap 10. Transformacja modelu konceptualnego do modelu logicznego
Etap 11. Zdefiniowanie schematów relacji i podanie danych przykładowych
Etap 12. Opracowanie schematu bazy danych i słownika atrybutów
Etap 13. Zdefiniowanie perspektyw dla wszystkich użytkowników bazy

DODATEK C
SZABLON DOKUMENTACJI PROJEKTU BAZY DANYCH

TEMAT PROJEKTU BAZY DANYCH

. . . . . .

Autorzy projektu

. . . . . .

Data

. . . . . .

SPIS TREŚCI W DOKUMENTACJI PROJEKTU

Etap 1. Temat, cel, zakres i użytkownicy bazy danych

1.1. Temat

1.2. Cel

1.3. Zakres

1.4. Użytkownicy

Etap 2. Analiza wycinka rzeczywistości

2.1. Szczegółowy opis wycinka rzeczywistości

2.2. Słownik pojęć

2.3. Użytkownicy i zakres uprawnień

2.4. Wymagania funkcjonalne

2.5. Wymagania niefunkcjonalne

2.6. Analiza istniejących baz danych

2.7. Analiza kosztów wykonania projektu bazy danych

Etap 3. Kategorie

KAT/001 Nazwa *rzeczownik w l. poj, bez polskich liter*

Opis: Semantyka kategorii i związek z innymi kategoriami

Atrybuty:

Atrybut1 – opis, np. przykładowa wartość

Atrybut2 – opis, np. przykładowa wartość

Etap 4. Reguły funkcjonowania

4.1. Reguły ogólne

REG/001 Tekst

REG/002 Tekst

4.2. Reguły dotyczące KAT/001 Nazwa

REG/003 Tekst

REG/004 Tekst

4.3. Reguły dotyczące KAT/002 Nazwa

REG/005 Tekst

REG/006 Tekst

Etap 5. Ograniczenia dziedzinowe

5.1. Ograniczenia ogólne

OGR/001 Format daty: dd-mm-rrrr (dd – dzień, mm – miesiąc, rrrr – rok)
OGR/002 Tekst

5.2. Ograniczenia dla KAT/001 Nazwa

OGR/002 Tekst
OGR/003 Tekst

5.3. Ograniczenia dla KAT/002 Nazwa

OGR/004 Tekst
OGR/005 Tekst

Etap 6. Transakcje

TRA/001 Nazwa transakcji

Opis: Opis i użytkownicy
Uwarunkowania:

Wejście/Wyjście TRA/*xxx*

	Wejście	**Wyjście**
Użytkownik		
Baza danych		
System		

Etap 7. Definicje typów encji i typów związków

7.1. Typy encji

ENC/001 NAZWA_TYPU_ENCJI

Semantyka encji: opis
Wykaz atrybutów:

Nazwa atrybutu	**Opis atrybutu**	**Typ**	**OBL (+)** **OPC (–)**

Klucze kandydujące:
Klucz główny:
Charakter encji:

7.2. Typy związków

ZWI/001 Zw(ENCJA1(0,N) : ENCJA2(1,1))

REG/001 Encja1 musi być powiązana z Encją2
REG/002 Encja1 może być powiązana co najwyżej z jedną Encją2
REG/003 Encja2 nie musi być powiązana z Encją1
REG/004 Encja2 może być powiązana z wieloma Encjami2

Etap 8. Definicje predykatowe typów encji i typów związków

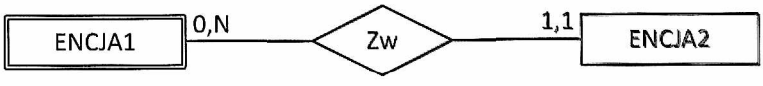
8.1. Definicje predykatowe typów encji

ENC/001 ENCJA1(*Lista atrybutów*)
ENC/002 ENCJA2(*Lista atrybutów*)

8.2. Definicje predykatowe typów związków

ZWI/001 Zw(ENCJA1(1,1) : ENCJA2(0,N))

Etap 9. Diagram związków encji



Etap 10. Transformacja modelu konceptualnego do modelu logicznego

ZWI/001 Zw(ENCJA1(1,1) : ENCJA2(0,N))
ENC/001 ENCJA1(*Lista atrybutów*)
ENC/002 ENCJA2(*Lista atrybutów*)

Po przekształceniu otrzymujemy:

REL/001 Nazwa_schematu_relacji(*Lista atrybutów*)
REL/002 Nazwa_schematu_relacji(*Lista atrybutów*)

Etap 11. Definicje schematów relacji

REL/001 Nazwa_schematu_relacji/NAZWA_TYPU_ENCJI
lub
REL/xxx Nazwa_schematu_relacji/Nazwa_związku, NAZWY_TYPÓW_ENCJI

Nazwa atrybutu	Dziedzina	Maska	OBL	Wart. dom.	Ograniczenia	Unikalność	Klucz	Referencje	Źródło danych

Znaczenie atrybutów w schemacie relacji **Nazwa_schematu_relacji**

Nazwa atrybutu	Znaczenie atrybutu

Przykładowe dane relacji o schemacie relacji **Nazwa**

Etap 12. Schemat bazy danych i słownik atrybutów

12.1. Relacyjny schemat bazy danych

NAZWA_BAZY_DANYCH
Nazwa_schematu_relacji_1(*lista_atrybutów_schematu_relacji_1*)
Nazwa_schematu_relacji_2(*lista_atrybutów_schematu_relacji_2*)
....

12.2. Słownik atrybutów

Nazwa atrybutu	Dziedzina	Przynależność do schematu relacji

Etap 13. Perspektywy

13.1. Użytkownicy

- U1
- U2
- ....

13.2. Perspektywy i transakcje

PER/001 Nazwa perspektywy
Użytkownik: Nazwa użytkownika
Podschemat bazy danych dla perspektywy:
Nazwa_schematu_relacji_1(*lista atr*)
Nazwa_schematu_relacji_2(*lista atr*)
...
Transakcje: Lista transakcji postaci TRA/*xxx*

Tabela transakcji (dla każdej transakcji wymienionej w danej perspektywie):

TRA/xxx Nazwa transakcji

Nazwa schematu relacji	Atrybut	Zapis	Odczyt	Modyfikacja

13.3. Weryfikacja przypisania transakcji

Macierz powiązań transakcji z perspektywami użytkowników

	U1	U1	U1	U2		
	PER/001	**PER/002**	**PER/003**			
TRA/001						
TRA/002						
TRA/003						

Wydział Informatyki i Zarządzania Politechniki Wrocławskiej

Przedstawiona w książce strukturalna metodyka projektowania baz danych jest aktualna i często stosowana, zwłaszcza w małych i w średnich projektach. Na Wydziale Informatyki i Zarządzania Politechniki Wrocławskiej wykonano w tej metodyce wiele tysięcy studenckich projektów. Książka jest drugim wydaniem – poprawionym, uzupełnionym i rozszerzonym – publikacji z 2004 roku. Przy zachowaniu zasadniczych założeń metodyki wzbogacono opisy, wprowadzono dodatkowe zalecenia i wskazówki do poszczególnych etapów projektowych w celu szybszego wytwarzania projektów lepszej jakości. Zamieszczono także szablon dokumentacji, który może być wykorzystywany do realizacji projektów w proponowanej metodyce. Celem wszystkich zmian było ułatwienie samodzielnego analizowania przedstawionych zagadnień i maksymalne podniesienie poziomu czytelności zawartego materiału. Autorzy mają nadzieję, że książka w obecnym kształcie będzie jeszcze bardziej zrozumiała, jej lektura sprawi Czytelnikowi przyjemność i umożliwi wykonywanie poprawnych projektów baz danych.

ISBN 978-83-958341-0-3