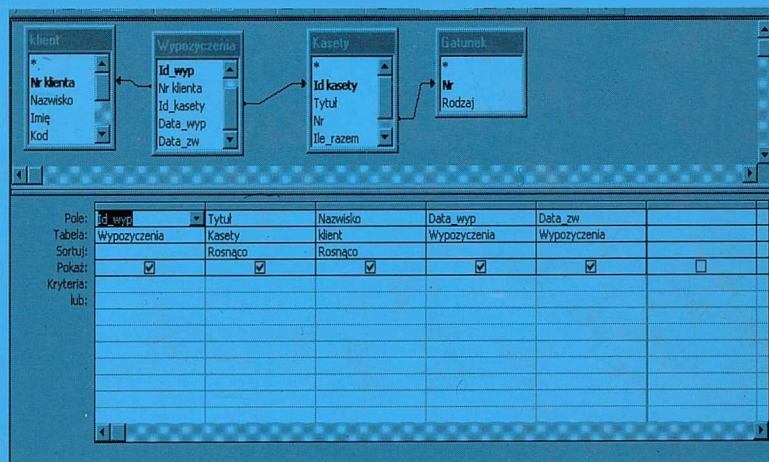


Hanna Mazur
Zygmunt Mazur

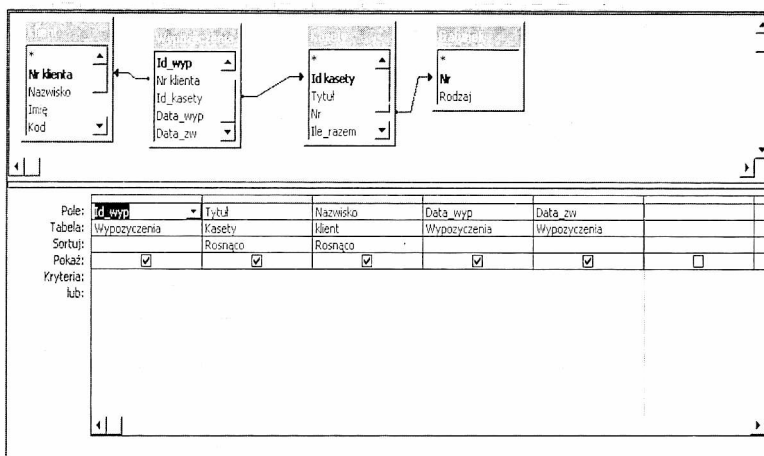
PROJEKTOWANIE BAZ DANYCH





Hanna Mazur
Zygmunt Mazur

PROJEKTOWANIE BAZ DANYCH



POLSKIE TOWARZYSTWO INFORMATYCZNE

Copyright © by Hanna Mazur, Zygmunt Mazur, 2002

Recenzja: Jacek Gruber

Redakcja: Hanna Mazur

Projekt okładki: Zygmunt Mazur

Adres do korespondencji:
Hanna Mazur, Zygmunt Mazur
Politechnika Wrocławska
Wydział Informatyki i Zarządzania
Wydziałowy Zakład Informatyki
Wybrzeże Wyspiańskiego 27, 50-370 Wrocław
h.mazur@ci.pwr.wroc.pl
z.mazur@ci.pwr.wroc.pl

Wydawca:
Polskie Towarzystwo Informatyczne
Oddział Dolnośląski
Pl. Strzelecki 20, p. 29
50-224 Wrocław

Druk:
Wydawnictwo POLIGRAF
54-210 Wrocław
ul. Kwiska 150
tel. (0-71) 351-73-52

Biblioteka Główna i OINT
Politechniki Wrocławskiej



001889531



ISBN 83-88330-05-5

309514 L/1

Wrocław 2002



POLSKIE TOWARZYSTWO INFORMATYCZNE

28/10/02

SPIS TREŚCI

WSTĘP	5
1. Wprowadzenie	7
2. Podstawowe pojęcia relacyjnych baz danych	11
2.1. Relacyjny model danych	11
2.2. Podstawowe operacje algebry relacji	14
2.3. Podstawowe pojęcia baz danych	16
3. Projektowanie konceptualne	19
3.1. Faza analizy	19
3.1.1. Analiza wycinka rzeczywistości	20
3.1.2. Analiza istniejącej bazy danych	23
3.1.3. Analiza wymagań funkcjonalnych	23
3.1.4. Analiza wymagań niefunkcjonalnych	23
3.2. Reguły funkcjonowania	24
3.3. Ograniczenia dziedzinowe	25
3.4. Definicje kategorii	26
3.5. Transakcje	27
3.6. Model konceptualny	29
3.6.1. Identyfikacja encji	29
3.6.2. Związki i typy związków	33
3.6.3. Definicje predykatowe encji i związków	37
3.6.3.1. Definicje predykatowe typów encji	37
3.6.3.2. Definicje predykatowe typów związków	37
3.6.4. Diagram związków encji	39
3.6.5. Normalizacja schematów relacji	42
4. Projektowanie logiczne	53
4.1. Transformacja modelu konceptualnego do modelu logicznego	53
4.1.1. Reguły transformacji	54
4.1.2. Definicja relacji (tabel)	58

4.1.3. Schemat relacyjnej bazy danych	64
4.2. Definiowanie perspektyw	63
5. Zakończenie	65
6. Zestawienie etapów projektowania danych	67
7. Literatura	68
Dodatek	71

Errare humanum est
- Starożytni

WSTĘP

W ostatnich latach obserwuje się dynamiczny rozwój systemów umożliwiających szybkie tworzenie oprogramowania dla baz danych. Nowoczesne systemy zarządzania bazami danych pozwalają na proste formułowanie zapytań do bazy danych i szybkie wyszukiwanie odpowiednich informacji z baz danych. W wielu publikacjach i dokumentacjach firmowych narzędzi główny nacisk kładzie się na opis składni języka definiowania (opisu) danych (*DDL* - ang. *Data Definition (Description) Language*), języka manipulowania danymi (*DML* - ang. *Data Manipulation Language*), języka zapytań (wyszukiwania) danych (np. *QBE* - ang. *Query By Example*) i języka nadzoru, który obejmuje nadawanie i odbieranie uprawnień użytkownikom, autoryzację (ang. *authorization*) dostępu do bazy danych (zajmuje się tym administrator bazy danej – *DBA* – ang. *Database Administrator*). Często pomija się problemy związane z właściwym zaprojektowaniem bazy danych lub etap projektowania bazy traktuje się bardzo pobieżnie. W systemach wykorzystujących źle zaprojektowane bazy danych napotyka się na spore trudności z utrzymaniem spójności bazy danych, często trzeba wykonywać dodatkowe czynności związane z obsługą anomalii aktualizowania danych, dopisywania danych do bazy i usuwania danych z bazy danych.

Nawet w bardzo prostych bazach danych można napotkać wielkie problemy. Rozważmy prostą bazę danych „PANORAMA FIRM” zawierającą tylko dane o unikalnych nazwach firm, ich adresach, dostarczanych towarach i aktualnych cenach towarów

PANORAMA (Firma, Adres, Towar, Cena).

Dla tak źle zaprojektowanej bazy wystąpią anomalie dopisywania danych, usuwania, aktualizacji i redundancji danych. Zauważmy, że adres firmy będzie

się powtarzał tyle razy, ile różnych towarów dostarcza dana firma. Zmiana adresu firmy będzie wymagała aktualizowania wielu rekordów w zależności od liczby oferowanych produktów. Kluczem relacji jest 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, gdyby firma przestała dostarczać towary, to nie można by nadal przechowywać danych o firmie itd. Wszystkie te anomalie mogą być usunięte po wykonaniu poprawnego projektu bazy danych.

Celem pracy jest zwrócenie uwagi Czytelnika na podstawowe problemy związane z wykonaniem prawidłowego projektu bazy danych. Jakość poszczególnych etapów projektowania ma bardzo istotny wpływ na efekt końcowy czyli na ostateczną postać bazy danych. Umiejętność właściwego zaprojektowania bazy danych jest równie ważna jak umiejętność programowania.

Wiele faktów świata rzeczywistego, niejednokrotnie pozornie nieistotnych z punktu widzenia początkującego projektanta (np. dostawca musi czy może dostarczać części, część jest dostarczana przez jednego lub wielu dostawców albo nie dostarczana przez nikogo) ma bardzo istotny wpływ na ostateczną postać bazy danych. 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.

Niniejsza książka jest przeznaczona dla osób projektujących bazy danych. Może być wykorzystana jako materiał dydaktyczny w ramach zajęć związanych z projektowaniem relacyjnych baz danych.

1. WPROWADZENIE

Niniejsza praca ma na celu omówienie najważniejszych pojęć z teorii relacyjnych baz danych oraz przedstawienie poszczególnych etapów projektowania baz danych. Obecnie jest dostępnych wiele publikacji dotyczących tematyki baz danych, ale są to na ogół pozycje bardzo obszerne (kilkaset stron), poruszające wiele zagadnień (nie zawsze bezpośrednio dotyczących projektowania baz danych), co utrudnia wydobycie informacji istotnych z punktu widzenia projektanta bazy danych.

Poprawnie zaprojektowana relacyjna baza danych, musi spełniać wszystkie oczekiwania przyszłego użytkownika a jednocześnie musi gwarantować bezbłędne, wydajne i przyjazne w obsłudze funkcjonowanie. Takie własności można zapewnić, jeśli poprawnie przeprowadzi się wszystkie etapy projektowania bazy danych.

Celem pracy jest przedstawienie w sposób systematyczny i uporządkowany, a zarazem prosty i zrozumiały, kolejnych etapów projektowania relacyjnych baz danych, przy czym autorzy dużą wagę przywiązują do znormalizowanego sposobu dokumentowania poszczególnych etapów w taki sposób, aby na każdym etapie przedstawić używaną notację i pojęcia.

Prawidłowo zaprojektowana baza danych nie powinna zawierać żadnych zbędnych danych, powinna natomiast umożliwiać efektywną pracę zaimplementowanego systemu oraz możliwości jego rozbudowy. W projektowaniu relacyjnej bazy danych można wyróżnić następujące etapy:

- projektowanie koncepcyjne (konceptualne, pojęciowe), polegające na zdefiniowaniu wymagań niezależnych od implementacji. Proces zbierania wymagań powinien doprowadzić do powstania konceptualnego modelu danych (CDM – *ang. Conceptual Data Model*), zwykle przedstawianego w postaci graficznej za pomocą tzw. **diagramów związków encji** (ERD – *ang. Entity Relationship Diagram*). Na tym etapie często przeprowadzany jest również **proces normalizacji**. W fazie tej nie bierze się pod uwagę docelową, implementacyjną platformę sprzętową przyszłego systemu bazy danych.
- projektowanie logiczne, polegające na transformacji modelu konceptualnego do modelu logicznego (w tym przypadku relacyjnego),

w którym określamy strukturę przechowywanych danych, czyli tzw. schematy relacji. Na tym etapie często przeprowadza się normalizację schematów relacji. Normalizacja danych jest sformalizowaną procedurą, w wyniku której eliminujemy redundancje (powtarzanie się) danych, ich niespójności i anomalie, czyli nieprawidłowości związane z aktualizacją, usuwaniem i dopisywaniem danych.

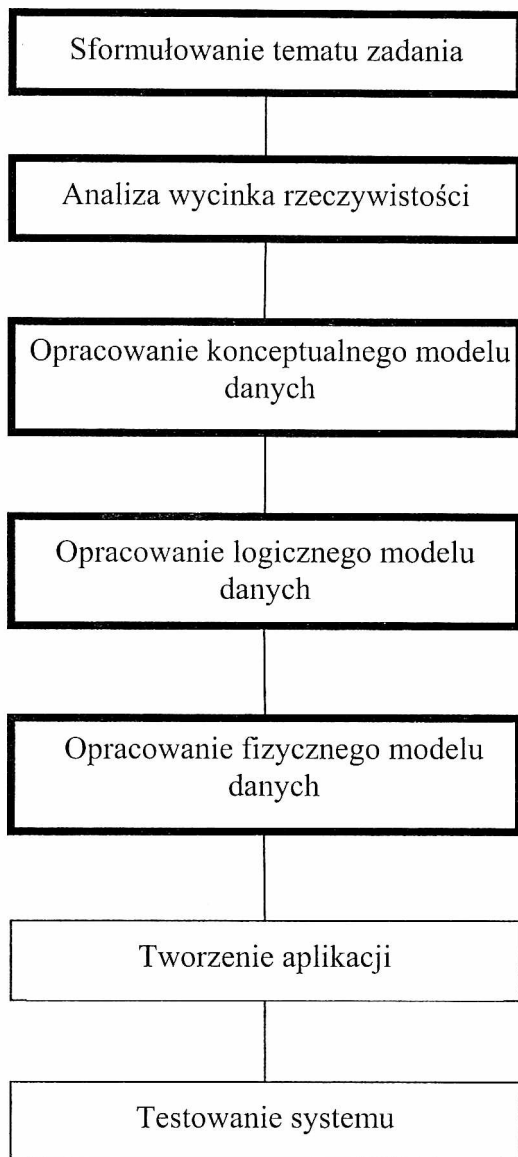
- projektowanie fizyczne, w wyniku którego powstaje fizyczny model danych (PDM – *ang. Physical Data Model*). Na tym etapie należy określić sprzęt, na którym ma działać docelowy system informatyczny, oraz narzędzie, jakie będzie wykorzystane do implementacji systemu. Elementy logicznego projektu bazy danych zamieniamy na rzeczywiste obiekty bazy danych. Etap ten jest ściśle związany z budową aplikacji bazodanowej.

W niniejszej pracy zostaną omówione dwa pierwsze etapy, nie będziemy natomiast zajmować się projektowaniem fizycznym. Wiadomo, że w różnych systemach bazy danych są przechowywane w odmienny sposób, tej samej dziedzinie atrybutów mogą odpowiadać różne typy itd. Ten poziom szczegółowości na ogół jest nieistotny dla projektanta bazy danych, którego interesuje jedynie opracowanie modelu danych oraz określenie sposobu dostępu do danych.

Po prawidłowo przeprowadzonym projekcie bazy danych można **zaprojektować aplikację** obsługującą bazę danych w sposób poprawny, efektywny i przyjazny. Wiedząc już, *co aplikacja ma robić*, należy zastanowić się nad tym *jak to zrobić*. W tym celu należy zaprojektować potrzebne formularze, zapytania, raporty, widoki. Należy zaprojektować menu główne i podrzędne, sposób przełączania się pomiędzy przewidywanymi opcjami, pliki pomocy, sposób komunikowania się z użytkownikiem, sposób wyświetlania komunikatów. Należy rozważyć możliwość zabezpieczenia systemu hasłami z różnymi poziomami dostępu do danych dla różnych użytkowników itp. Po całościowym zaprojektowaniu aplikacji można przejść do **zaimplementowania systemu**. W tym celu wykorzystuje się odpowiednio dobrane środowisko. Kolejnym etapem powinno być **testowanie** systemu i sporządzenie bardzo dokładnej **dokumentacji** obejmującej wszystkie etapy projektowania schematu bazy danych, opis zbudowanego systemu bazodanowego, omówienie sposobu **zainstalowania programu i rozpoczęcia pracy z systemem bazy danych** oraz instrukcja posługiwania się aplikacją przez przyszłego użytkownika.

1. WPROWADZENIE

Kolejne etapy tworzenia systemu bazodanowego, obejmujące etapy projektowania bazy danych jak i projektowania części aplikacyjnej systemu bazy danych można przedstawić jako kaskadę działań projektowych (Rys. 1).



Rys. 1. Etapy projektowania bazy danych oraz tworzenia systemu bazy danych
Elementy Rysunku 1 narysowane grubą linią odpowiadają etapom projektowania schematu bazy danych.

1. WPROWADZENIE

Zanim przejdziemy do przedstawienia i omówienia poszczególnych etapów projektowania baz danych, omówimy podstawowe pojęcia z teorii relacyjnych baz danych.

2. PODSTAWOWE POJĘCIA RELACYJNYCH BAZ DANYCH

Baza danych (ang. *database*) jest trwałym zbiorem tematycznie ze sobą powiązanych danych (zob. też Dodatek). Pojęcie bazy danych zostało sformułowane w latach 1962-63. Najbardziej abstrakcyjnym poziomem projektu bazy danych jest model danych, czyli opis pojęciowy przestrzeni zagadnienia [Riordan2000]. W modelu danych nie ma żadnych odniesień do fizycznej struktury systemu.

Wyróżnia się następujące podstawowe modele baz danych:

- hierarchiczny
- sieciowy
- relacyjny
- obiektowy

Obszerne informacje na temat tych podstawowych modeli można znaleźć w literaturze np. [Date1981], [Hernan1998], [Pankow1992], [UllWid2000]. W niniejszej pracy skupimy się tylko na **modelu relacyjnym**. Twórcą relacyjnego modelu bazy danych jest E.F.Codd, który w 1970 r. opublikował fundamentalną pracę na temat relacyjnych baz danych [Codd1970].

2.1. RELACYJNY MODEL DANYCH

Modele danych są opisywane w kategoriach encji, atrybutów, dziedzin i powiązań, których definicje podamy w dalszej części pracy.

Ze względu na bogatą dostępną literaturę na temat relacyjnych baz danych (np. [Date2000], [DelAdi1989]), ograniczymy się tutaj jedynie do podania podstawowych pojęć teorii relacyjnych baz danych.

Relacja R (ang. *relation*) jest podzbiorem iloczynu kartezyjańskiego n ($n > 0$) dowolnych zbiorów wartości (**dziedzin** - ang. *domain*)

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

gdzie D_i ($i=1, \dots, n$) jest dowolnym zbiorem wartości (dziedziną).

2. PODSTAWOWE POJĘCIA RELACYJNYCH BAZ DANYCH

Elementami iloczynu kartezjańskiego są n-krotki (zwane **krotkami** – *ang. tuples*) wartości $(d_1, d_2, \dots, d_n)$ przy czym

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

Relacja jest identyfikowana przez **nazwę relacji**.

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

Uwzględniając atrybuty możemy więc zapisać relację R następująco:

$$R(A_1, A_2, \dots, A_n) \subseteq D_1 \times D_2 \times \dots \times D_n = \{(d_1, d_2, \dots, d_n) : d_i \in D_i, i = 1, \dots, n\}$$

gdzie

D_i jest dowolnym zbiorem wartości (dziedziną) dla $i = 1, \dots, n$,
 $\text{dom}(A_i) = D_i$ dla $i = 1, \dots, n$.

Schemat relacji (*ang. relation schema*) – nazwa relacji oraz zbiór atrybutów relacji, $R(A_1, A_2, \dots, A_n)$.

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

Liczność relacji (*ang. cardinality*) – liczba dopuszczalnych krotek w relacji.

Stopień relacji (*ang. degree*) – liczba atrybutów relacji.

Instancja relacji (*ang. instance of relation*) – zbiór krotek relacji.

Przykład schematu relacji:

Załóżmy, że interesują nas cechy samochodów takie jak:

marka, kolor, rok produkcji, zużycie paliwa, numer rejestracyjny,
data rejestracji,

gdzie:

marka – przyjmuje wartości ze zbioru nazw marek, np. opel, BMW, Fiat 126p;

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

rok produkcji – jest liczbą całkowitą większą od 1900, np. 1995;

2. PODSTAWOWE POJĘCIA RELACYJNYCH BAZ DANYCH

zużycie paliwa – jest liczbą rzeczywistą dodatnią z jednym miejscem po przecinku, oznaczającą zużycie paliwa w litrach na 100 km, np. 8,5;
numer rejestracyjny – jest ciągiem 5 liter i 3 cyfr, np. NUMER123;
data rejestracji – jest datą w postaci dd-mm-rrrr, np. 02-02-1999.

Dane o samochodach można przedstawić w postaci tabeli **REJESTR** (zob. Tabela 1).

Tabela 1

Przykładowe dane tabeli REJESTR

REJESTR

Marka	Kolor	RokProd	Zużycie	NrRej	DataRej
Opel	Biały	1990	6	ABCDE123	10-02-1998
Fiat 126p	Czerwony	1995	7,4	TYCIK543	01-03-1995
Opel	Czerwony	1995	5,6	ZDREW876	15-02-1998

Schemat relacji **REJESTR** ma postać:

REJESTR (Marka, Kolor, RokProd, Zużycie, NrRej, DataRej)



Zbiór identyfikujący relacji (ang. *identifying set of relation*) – taki zbiór atrybutów danej relacji, których kombinacje wartości jednoznacznie identyfikują każdą krotkę relacji [CELL1988].

Klucz (ang. *key*) - jest to taki zbiór identyfikujący relacji, którego żaden podzbiór nie jest zbiorem identyfikującym, czyli jest to atrybut (lub kilka atrybutów), którego wartość jednoznacznie identyfikuje każdą krotkę w relacji.

Klucz kandydujący (ang. *candidate key*) - dla jednej relacji może istnieć kilka kluczy, które jednoznacznie identyfikują krotki relacji. Nazywa się je kluczami kandydującymi. Spośród kluczy kandydujących wybiera się jeden tzw. klucz główny. Klucz kandydujący nie może przyjmować wartości pustej (NULL).

Klucz główny (ang. *primary key*) - dla jednej relacji może istnieć kilka kluczy. Klucz, wybrany spośród kluczy kandydujących, nazywa się kluczem głównym.



Sztuczny klucz główny (*ang. artificial primary key*) - sztucznie stworzony klucz główny w przypadku, gdy klucze kandydujące są bardzo złożone lub, gdy brak jest klucza głównego. Słowo „sztuczny” oznacza, że klucz ten nie pojawił się w sposób naturalny i trzeba go dołączyć do atrybutów relacji, np. Id_wpisu.

Klucz obcy (*ang. foreign key*) - atrybut lub kilka atrybutów relacji, które są kluczem głównym w innej relacji.

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

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

Nadklucz (*ang. super key*) - zbiór atrybutów, który zawiera klucz (czyli zbiór atrybutów, z których można utworzyć klucze danej relacji).

Powiązanie (*ang. relationship*) - rodzaj związku zachodzący pomiędzy określonymi atrybutami krotek różnych relacji.

2.2. PODSTAWOWE OPERACJE ALGEBRY RELACJI

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

Operatory algebry relacyjnej mają za argumenty jedną lub więcej relacji, a wynikiem ich działania jest również relacja. Podstawowe operacje algebry relacyjnej to:

- selekcja (SELECT)

jest to operacja jednoargumentowa, oznacza wybór krotek z relacji, spełniających zadany warunek dotyczący wartości atrybutów danej relacji, np. dane osób o nazwisku „Nowak” z relacji **PRACOWNIK**(IdOsoby, Nazwisko, Imię, Pensja, IdDziału). Wynikiem jest relacja zawierająca te krotki z relacji **PRACOWNIK**, których odpowiednie atrybuty spełniają

2. PODSTAWOWE POJĘCIA RELACYJNYCH BAZ DANYCH

zadany warunek. Schemat relacji wynikowej jest taki sam jak schemat relacji wyjściowej (w tym przypadku PRACOWNIK).

- projekcja (PROJECT)

czyli rzut, jest to operacja jednoargumentowa, polegająca na wyborze z danej relacji określonych atrybutów, np. nazwisko, imię i pensja wszystkich pracowników. Relacja wynikowa zawiera określone atrybuty relacji wyjściowej i wszystkie krotki.

- złączenie (JOIN)

jest to operacja dwuargumentowa. Polega na połączeniu krotek (z dwóch relacji, posiadających kolumny o takich samych dziedzinach) spełniających zadany warunek. Z relacji wynikowej usuwane są powtarzające się krotki, nie są usuwane powtarzające się atrybuty. Operacje złączenia można wykorzystać na przykład w celu uzyskania danych o pracownikach i działach, w których pracują. Jeśli relacja PRACOWNIK(IdOsoby, Nazwisko, Imię, Pensja, IdDziału) zawiera dane o pracownikach a relacja DZIAŁ(IdDziału, Nazwa, Adres) zawiera dane o działach, to wykonujemy operację złączenia relacji PRACOWNIK i DZIAŁ przy czym atrybutem łączącym jest IdDziału.

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 relacyjnej. Ponadto wprowadza się dodatkowe operatory algebry relacyjnej takie jak:

- **suma** (operator określony dla relacji o takich samych atrybutach, wynikiem jest relacja, której wiersze są sumą teoriomnogościową wierszy relacji wyjściowych),
- **różnica** (operator określony dla relacji o takich samych atrybutach, wynikiem jest relacja, której wiersze są różnicą teoriomnogościową wierszy relacji wyjściowych),
- **iloczyn** czyli *przecięcie* (operator określony dla relacji o takich samych atrybutach, wynikiem jest relacja, której wiersze należą do obydwu relacji wyjściowych),
- **iloczyn kartezyjański** (operator dwuargumentowy, relacja wynikowa zawiera atrybuty jednej i drugiej relacji, zbiór krotek relacji wyjściowej

powstaje przez utworzenie iloczynu kartezjańskiego krotek obu relacji wyjściowych),

- **złączenie naturalne** (usuwane są powtarzające się atrybuty),
- **złączenie zewnętrzne** (złączenie zewnętrzne relacji L i P jest to złączenie, w którym są zachowywane wszystkie krotki: relacji L – *złączenie lewostronne*, relacji P – *złączenie prawostronne*, obu relacji – *złączenie obuustronne*),
- **Θ -złączenie** (łączenie krotek spełniających warunek Θ).

2.3. PODSTAWOWE POJĘCIA BAZ DANYCH

Podamy teraz podstawowe pojęcia związane z relacyjnymi bazami danych:

Tabela (*ang. table*) jest jedną z najczęstszych reprezentacji relacji. Tabela jest podzielona na wiersze i kolumny.

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

Wiersz (*ang. row*) odpowiada krotce relacji, czyli jest wystąpieniem (elementem) relacji.

Każda tabela ma swoją unikalną nazwę. Tabela składa się z **nagłówka**, zawierającego nazwy kolumn (atrybuty), oraz z wierszy (odpowiedników krotek), stanowiących treść tabeli. Nazwa kolumny jest jednoznaczna w obrębie jednej tabeli. Każdy wiersz tabeli odpowiada jednej krotce relacji i musi się różnić od pozostałych wierszy w tabeli chociaż jedną wartością. Tabela może być wypełniana przez dopisywanie wierszy. Dane w tabeli można zmieniać. Można usuwać wiersze z tabeli. Można wyszukiwać dane. Czynności takie jak dopisywanie kolumny do tabeli, modyfikowanie nazwy kolumny tabeli, usuwanie kolumny z tabeli również można wykonywać, ale w praktyce wykonuje się je bardzo rzadko i powinno się tego unikać.

Pole (*ang. field*) jest to przecięcie wiersza i kolumny. Wartość każdego pola powinna być wartością atomową (nierozkładalną). Odwołanie do pola następuje poprzez nazwę kolumny (atrybutu).

Zwróćmy uwagę [Date2000], że definicja relacyjnego modelu danych, podana przez Codda, która jest przytaczana do dziś, zawiera pewne nieścisłości (w teorii relacyjnych baz danych takich nieścisłości jest wiele). W tabeli mogą być dwie dziedziny atrybutów takie same, przy czym atrybuty (nazwy kolumn) muszą być różne (np. dla dwóch różnych atrybutów *Imię Ojca*, *Imię Matki*, może być ta sama dziedzina – np. łańcuch maksymalnie 15. literowy). Kolejność dziedzin w relacji matematycznej jest ściśle ustalona, ale w przypadku tabeli w modelu relacyjnym nie ma ona znaczenia (z kolei ma znaczenie w strukturalnym języku zapytań do zarządzania bazami danych **SQL** - ang. *Structured Query Language* [DateDar2000], str. 102). Następną nieścisłością jest zastrzeżenie, aby w tabeli nie było identycznych wierszy. Tymczasem w SQL dochodzi do sytuacji, w których wiersze mogą powtarzać ([DateDar2000], str. 101).

Plik (ang. *file*) jest zbiorem logicznie powiązanych informacji przechowywanych np. w pamięci dyskowej. Identyfikowany jest poprzez swoją nazwę.

Rekord (ang. *record*) jest podstawową jednostką informacji przechowywania danych w pliku. Odpowiednikiem rekordu jest krotka w relacji i wiersz w tabeli. Rozmiar rekordu w danym pliku jest stały.

Relacyjna baza danych (ang. *relational database*) to baza danych oparta na modelu relacyjnym. Jest to baza danych postrzegana jako zbiór nazwanych dwuwymiarowych tablic o określonej liczbie kolumn i nieograniczonej 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 język zapytań wysokiego poziomu (np. QBE, SQL). Na danych można wykonywać operacje algebry relacyjnej.

System zarządzania bazą danych – SZBD (ang. *Database Management System – DBMS*) jest to oprogramowanie narzędziowe służące do zakładania, przechowywania i przetwarzania dużych zbiorów informacji, np. ORACLE, INGRES, PROGRESS, ACCESS, dBASE.

System zarządzania relacyjną bazą danych – SZRBD (ang. *Relational Database Management System – RDBMS*) jest to oprogramowanie narzędziowe służące do zakładania, przechowywania i przetwarzania relacyjnych baz danych.

System bazy danych (ang. *database system*) jest to baza danych wraz z jej systemem obsługi bazy danych.

Transakcja (ang. *transaction*) jest to ciąg poleceń (operacji) przeznaczonych do wykonywania w całości albo wcale, np. 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, jeśli tak, to odjęcie danej kwoty z pierwszego konta oraz dodanie kwoty do stanu drugiego konta.

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

Integralność (ang. *integrity*) bazy danych oznacza, że baza danych jest stale zgodna z rzeczywistością, odzwierciedla aktualny stan danych w rzeczywistości oraz ich zmiany.

Trwałość (ang. *durability*) bazy danych oznacza, że dane są przechowywane i dostępne przez pewien czas.

Spójność (ang. *consistency*) bazy danych oznacza poprawność poszczególnych danych oraz poprawność bazy danych jako całości. Zapewnienie spójności bazy danych jest podstawowym problemem związanym z definiowaniem transakcji. Gdyby w trakcie wykonywania transakcji przelewania pieniędzy z jednego konta na drugie po wykonaniu czynności odjęcia danej kwoty z pierwszego konta 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.

3. PROJEKTOWANIE KONCEPTUALNE

Projektowanie konceptualne (pojęciowe) ma na celu zbudowanie modelu konceptualnego (CDM – *ang. Conceptual Data Model*). Aby opracowywany projekt bazy danych był poprawny i spełniał oczekiwania przyszłego użytkownika musimy uświadomić sobie, w jakim celu projektujemy bazę, kto będzie jej użytkownikiem, jakie są jego potrzeby, ograniczenia i wymagania. Nie wystarczy pobieżnie określić, że projektujemy bazę np. dla szkoły, ponieważ w zależności od tego, kto ma być jej użytkownikiem i do wykonywania jakich zadań dane będą wykorzystywane, projektowana baza może zawierać np. tylko dane pracowników (dział finansowy pracowników szkoły) albo tylko dane uczniów (księga uczniów), albo dane zarówno uczniów jak i pracowników (kompleksowa obsługa szkoły).

Pierwszym i zasadniczym elementem projektowania bazy danych jest więc zwięzłe i czytelne sformułowanie, w jakim celu projektujemy bazę danych. Jasno określony cel umożliwia przystąpienie do fazy analizy projektowania bazy danych.

3.1. FAZA ANALIZY

Dla danej rzeczywistości 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.

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

Celem fazy analizy jest:

- a) określenie celów przedsięwzięcia,
- b) przeprowadzenie wnikliwej analizy tematu,
- c) przeprowadzenie z ekspertem dziedzinowym szczegółowego wywiadu, na podstawie którego będzie można precyzyjnie opisać określony

wycinek rzeczywistości oraz zaplanować przyszłe funkcje systemu dla przewidywanych użytkowników systemu,

- d) ocena stanu ewentualnie istniejącej bazy danych,
- e) oszacowanie kosztów związanych z budową nowej bazy i nowego systemu zarządzania tą bazą.

W trakcie fazy analizy wyodrębniamy:

- obiekty (kategorie, encje) czyli rzeczy istotne (i nadajemy im nazwy, które są rzeczownikami),
- związki zachodzące pomiędzy encjami (i nadajemy im nazwy, które są czasownikami).

W encjach wyróżniamy cechy dla nich charakterystyczne, czyli atrybuty encji.

3.1.1 ANALIZA WYCINKA RZECZYWISTOŚCI

Pierwszym etapem projektowania bazy danych jest szczegółowa **analiza wycinka rzeczywistości**, który chcemy modelować w bazie danych [Górski2000], [Hernan1998]. W tym celu należy przeprowadzić wnikliwy wywiad z ekspertem dziedzinowym. Następnie trzeba 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 dane i uzyskane informacje prezentować,
- 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,

3. PROJEKTOWANIE KONCEPTUALNE

informatyków, psychologów, itp. aby wszechstronnie przeanalizować zagadnienie.

Dobrze jest zgromadzić wszystkie dostępne materiały związane z analizowaną rzeczywistością, np.:

- przepisy określające funkcjonowanie danej organizacji,
- druki i formularze papierowe używane dotychczas (np. faktury, zamówienia),
- postaci 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 lub są niekompletne albo nieczytelne, czy dopuszcza się brak pewnych 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. 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.

Dane przechowywane w bazie powinny być:

- pełne
- aktualne
- odpowiednie (jeśli interesuje nas wydobycie węgla w Polsce a ktoś poda nam pełne i aktualne dane dotyczące budowy dróg, to nie są to dane odpowiednie).

W tym miejscu chcielibyśmy podkreślić różnicę między pojęciami dana, informacja i wiedza. Ujmując to krótko można powiedzieć, że **dana** (ang. *data*, *datum*) to wartość, np. liczba całkowita, ciąg znaków, obrazek, i sama w sobie

nie ma większego znaczenia, nabiera sensu dopiero zinterpretowana dana czyli **informacja**. Angielskie słowo *datum*, wywodzące się z języka łacińskiego, oznacza dosłownie fakt, przesłankę.

Wartości przechowywane w bazie danych to **dane**, np. 5, Adam Nowacki, 12/12/25. Dane nie ulegają samoistnym zmianom aż do zmodyfikowania ich ręcznego lub przez jakiś proces automatyczny. Same w sobie są niezrozumiałe, można im nadać różne interpretacje. Nawet gdybyśmy wiedzieli, że 12/12/25 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. Informacje można przedstawiać na różne sposoby (jako formularz, raport, na ekranie komputera, 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 pliku liczba 12 nie nam nie mówi, można ją różnie zinterpretować (godzina, dzień, miesiąc, cena towaru, liczba studentów w laboratorium, numer domu itd.), liczba ta stanie się informacją, jeśli będziemy wiedzieli, że oznacza ona staż pracy pracownika Jana Kowalskiego. Kierownik, 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.

Wyczerpująco i poprawnie przeprowadzona analiza jest punktem wyjścia do prawidłowego zaprojektowania bazy danych. Brak umiejętności w przeprowadzaniu wywiadu bądź brak kompetencji osoby dostarczającej wiedzy o analizowanej rzeczywistości jest często przyczyną złego projektu, którego wady często są zauważane dopiero w trakcie eksploatacji bazy. Usunięcie wad systemu bazodanowego w trakcie jego eksploatacji jest bardzo kosztowne a często wręcz (z wielu względów) niemożliwe.

3.1.2. ANALIZA ISTNIEJĄCEJ BAZY DANYCH

W tej fazie należy przeprowadzić analizę być może istniejącej już bazy danych, związanej z rozważanym obszarem rzeczywistości. Należy wówczas zapoznać się ze sposobem gromadzenia danych i ich wykorzystywania oraz z ewentualnym funkcjonującym systemem informatycznym obsługującym dotychczasową bazę danych. Należy przeanalizować zalety i wady istniejącej bazy i systemu informatycznego, 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 systemu informatycznego a czy niezadowolenie użytkownika wynika na przykład z nieznajomości obsługi posiadanego programu.

3.1.3. ANALIZA WYMAGAŃ FUNKCJONALNYCH

Aby zaprojektowana baza danych umożliwiała wykonywanie określonych operacji należy je ustalić, wyszczególnić a następnie precyzyjnie opisać (rozdz. 3.3). W tym punkcie projektu wypisujemy funkcje, jakie dla zgromadzonych danych chcielibyśmy realizować, np.

- wprowadzanie, modyfikowanie, usuwanie danych,
- wyszukiwanie danych,
- sporządzanie statystyk,
- drukowanie raportów.

3.1.4. ANALIZA WYMAGAŃ NIEFUNKCJONALNYCH

Na etapie analizy wymagań niefunkcjonalnych opisujemy wszystkie pozostałe aspekty związane z opracowywaną bazą, na przykład, czy będziemy używać narzędzi CASE do projektowania bazy (np. EasyCASE Professional v.4.20), jakie metodyki projektowe będą wykorzystane do opracowania diagramów DFD (np. Yourdona) oraz diagramów ERD (np. Chena, Martina).

Dodatkowo, chociaż nie jest to związane z projektowaniem samej bazy danych, ale wiąże się z etapem tworzenia aplikacji, czyli tworzenia systemu bazy danych obsługującego zaprojektowaną bazę danych (Rysunek 1), określa się możliwości i oczekiwania użytkownika przyszłego systemu dotyczące

platformy sprzętowej, sposobu pracy z systemem (klawiatura, mysz), trybu pracy (tryb tekstowy, graficzny), czy program ma pracować w wersji sieciowej czy jednostanowiskowej itp. Szczególnie powinno określić się:

- platformę sprzętową (np. Pentium 3 z drukarką),
- platformę systemową (np. Windows 98/NT),
- środowisko implementacyjne (np. Borland C++Builder Professional v.4.0, ORACLE),
- rodzaj bazy danych (np. relacyjna),
- oszacowanie liczby danych wejściowych, tempo przyrostu danych,
- sposób archiwizowania danych.

Pomimo, że czasami opracowywany system bazy danych jest przeznaczony tylko do samodzielnej pracy (np. na danym komputerze instalowany jest tylko jeden system finansowo-księgowy F/K do pełnej obsługi małej firmy), to zdecydowana większość aplikacji powinna współpracować z istniejącym już systemem i innymi programami. Dokładna analiza środowiska jest wówczas niezbędną czynnością, aby nowy produkt mógł dobrze działać i współpracować z pozostałym oprogramowaniem.

Po bardzo precyzyjnej analizie wycinka rzeczywistości, która powinna być spisana w języku naturalnym, z uwzględnieniem wszelkich szczegółów rozpatrywanego miniświata i przedstawiona ekspertowi dziedzinowemu, mając określone założenia i wymagania użytkownika można przejść do etapu drugiego czyli do określenia kategorii.

3.2. REGUŁY FUNKCJONOWANIA

Na podstawie sporządzonej wcześniej analizy rzeczywistości należy wyodrębnić **reguły funkcjonowania** (reguły biznesowe) i **ograniczenia dziedzinowe**.

Prawidłowe określenie **reguł funkcjonowania**, a następnie ich przestrzeganie, ułatwi późniejsze modelowanie rzeczywistości, poprawne zaprojektowanie bazy danych i systemu informatycznego.

3. PROJEKTOWANIE KONCEPTUALNE

Reguły należy wypisywać w sposób systematyczny na podstawie dokonanej analizy fragmentu rzeczywistości, oznaczając je np.

REG/ x *sformułowanie reguły funkcjonowania w języku naturalnym*
gdzie x jest kolejnym numerem reguły.

Przykłady reguł funkcjonowania:

- | | |
|----------------|--|
| REG/001 | Dane towarów wpisuje kierownik magazynu |
| REG/002 | Każdy towar ma unikalny symbol nadany przez kierownika magazynu |
| REG/003 | Towar jest przyjmowany do magazynu na podstawie faktury zakupu |
| REG/004 | Faktury są przechowywane przez 5 lat |
| REG/005 | Przy sprzedaży towaru sprzedawca, za zgodą kierownika, może udzielić rabatu w określonej wysokości 3%, 5% lub 8% ceny netto towaru |
| REG/006 | Do wystawienia faktury niezbędny jest NIP klienta |

□

3.3. OGRANICZENIA DZIEDZINOWE

Ograniczenia dziedzinowe są nakładane na wcześniej określone kategorie. Wyszczególnione ograniczenia należy uwzględnić w dalszych etapach projektowania bazy danych i tworzonej implementacji. Wszystkie ograniczenia powinny wynikać z analizy wycinka rzeczywistości (rozdz. 3.1.1).

Pożądane jest wypisywanie ograniczeń w sposób systematyczny, np. dla poszczególnych kategorii.

Ograniczenia dziedzinowe można wypisywać numerując je kolejno, np.

OGR/00 x *sformułowanie ograniczenia w języku naturalnym*
gdzie x jest kolejnym numerem ograniczenia.

Przykłady ograniczeń dziedzinowych:

3. PROJEKTOWANIE KONCEPTUALNE

- OGR/001** Data urodzenia pracownika jest wcześniejsza niż data zatrudnienia
- OGR/002** Data zatrudnienia pracownika jest wcześniejsza niż data zwolnienia
- OGR/003** Kod pocztowy w adresie ma postać 99-999, gdzie 9 oznacza dowolną cyfrę
- OGR/004** Numer legitymacji pracownika jest postaci Numer pracownika /Rok wystawienia.
- OGR/005** Liczba pracowników filii jest liczbą dodatnią mniejszą od 25.

□

3.4. DEFINICJE KATEGORII

Ze szczegółowej analizy rzeczywistości powinniśmy wyodrębnić rzeczy istotne, o których chcemy przechowywać informacje, np. towar, uczeń, kaseta, lek, faktura, projekt itp.

Po wyodrębnieniu kategorii (obiektów) i nazwaniu ich rzeczownikami w liczbie pojedynczej, ustalamy ich atrybuty, czyli ich cechy charakterystyczne.

Następnie sporządzamy alfabetyczną listę kategorii opisując każdą następująco:

KAT/x Nazwa kategorii

Opis : Słowny opis kategorii

Atrybuty: Lista atrybutów i ich opisy

gdzie x jest kolejnym numerem kategorii.

Przykład opisu kategorii:

KAT/001 Towar

Opis: Kategoria towar służy do opisu towaru przechowywanego w magazynie.

Atrybuty:

Nazwa – nazwa grupy towarów np. cukier biały,

Symbol towaru – unikatowy symbol towaru, np. cuk001/99

Jednostka miary – jednostka miary towaru np.: kg, szt., metr

Cena zakupu – cena zakupu jednostki towaru wyrażona w zł, np. 3,50

Cena sprzedaży – cena sprzedaży jednostki towaru w zł, np. 3,70.

□

3.5. TRANSAKCJE

Transakcje są to operacje wykonywane na danych. Wynikają one z przeprowadzonej analizy i są opisywane przy pomocy pojęć języka naturalnego używanych w opisie rzeczywistości.

Transakcje powinny mieć cztery podstawowe własności:

- **niepodzielność, atomowość** (*ang. atomicity*) – transakcja jest wykonywana w całości albo wcale. Po przerwaniu transakcji musi być odtworzony stan bazy 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 niezależne, jedna transakcja od drugiej jest odizolowana,
- **trwałość** (*ang. durability*) – zmiany dokonane przez pomyślnie zakończoną transakcję są zachowywane na trwałe.

Te cztery istotne własności są często nazywane w skrócie *własnościami ACID*.

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

Etap wyodrębniania transakcji jest w zasadzie uszczegółowieniem etapu analizy wymagań funkcjonalnych (rozdz. 3.1.3). Opisuując poszczególne transakcje należy określić, które kategorie biorą udział w transakcji, co jest źródłem danych wejściowych oraz jakie informacje otrzymywane są na wyjściu.

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

- | | |
|----|----------------|
| U | - użytkownik, |
| BD | - baza danych, |
| S | - system. |

System jest najczęściej źródłem aktualnej daty (daty systemowej).

3. PROJEKTOWANIE KONCEPTUALNE

W etapie tym należy wymienić wszystkie przewidywane transakcje opisujące np. według schematu:

TRA/x Nazwa transakcji

Opis: Słowny opis transakcji

Uwarunkowania: Specyfikacja warunków dla prawidłowego wykonania transakcji

Wejście/Wyjście: Opis wejścia i wyjścia transakcji lub tabela transakcji gdzie x jest kolejnym numerem transakcji.

Przykład opisu transakcji:

TRA/001 Edycja danych dotyczących towaru

Opis: Zadaniem transakcji jest wyszukanie informacji o towarze i edycja tych danych np. ceny sprzedaży. Edycję może wykonać tylko kierownik magazynu.

Uwarunkowania: Dane towaru muszą istnieć w bazie. Jeśli dane towaru nie zostaną znalezione, zostanie wyświetlony komunikat *Brak danych*. Wyszukanie danych towaru może być wykonane po podaniu jego nazwy lub symbolu towaru.

Wejście:

- U - Nazwa towaru lub symbol towaru, nowe dane towaru
- BD - Dane towaru
- S -

Wyjście:

- U - Potwierdzenie edycji danych towaru
- BD - Dane towaru
- S -

Zamiast podanego wyżej sposobu zdefiniowania wejścia i wyjścia transakcji można sporządzić *tabelę transakcji* (Tabela 2):

Tabela 2

Wejście/Wyjście TRA/001

	WEJŚCIE	WYJŚCIE
Użytkownik	Nazwa towaru Nowe dane towaru	Potwierdzenie wykonania transakcji
Baza danych	Dane towaru	Dane towaru
System		



3.6. MODEL KONCEPTUALNY

Model konceptualny (konceptyjny) (*ang. Conceptual Data Model - CDM*) polega na zdefiniowaniu encji oraz związków między nimi. Po wyodrębnieniu encji i związków, przyjmując ustaloną notację (np. Chena [Barker1996, MurRyb1993], Martina [Beynon2000] lub Yourdona [Yourdon1996]) należy przedstawić graficznie opracowany model, czyli zbudować diagram związków encji (diagram obiektowo-związkowy).

Diagram związków encji (*ang. Entity Relationship Diagram - ERD*) – pozwala w sposób graficzny przedstawić dane i sposób widzenia ich struktury oraz związki między danymi [Barker1996, Date2000, Rodgers1995]. Aby prawidłowo opracować ERD należy najpierw zdefiniować encje oraz związki pomiędzy nimi a następnie przedstawić je w sposób graficzny używając określonej notacji.

Do sprawdzenia poprawności wyników analizy przedstawionych w formie diagramu związków encji zazwyczaj wykorzystuje się normalizację (rozdz. 3.6.4), dzięki której z opisanych encji być może zostaną wyłonięte nowe encje, wcześniej niedostrzeżone lub pominięte. Po znormalizowaniu, baza danych zajmuje mniej miejsca, dane się nie powtarzają, łatwiejsze jest dopisywanie, modyfikowanie i usuwanie danych. Dzięki normalizacji baza danych jest lepsza w tym sensie, że mniej podatna na anomalie, czyli nieprawidłowości związane z uaktualnianiem danych (*ang. update anomalies*), usuwaniem i dopisywaniem danych.

Poniżej podamy definicje podstawowych pojęć wykorzystywanych w tej fazie projektowania bazy danych.

3.6.1. IDENTYFIKACJA ENCJI

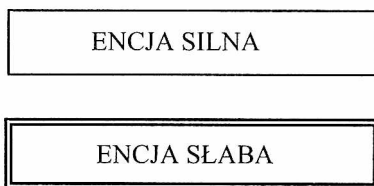
W tym etapie należy podać definicje **encji** – w pojęciach naturalnych wynikających z analizy rzeczywistości (opis słowny, atrybuty, klucze kandydujące, klucze główne). Opisywane encje są wyprowadzane z wyodrębnionych wcześniej kategorii (rozdz. 3.2).

Encja – (*ang. entity*) w języku łacińskim *esse* znaczy *być* [Boratyn1995]. Encja jest to pewien obiekt, rzecz, zjawisko, zdarzenie, pewien byt. Encje posiadają atrybuty, czyli cechy je opisujące, charakteryzujące, istotne z punktu widzenia eksperta dziedzinowego, przyjmujące wartości z ustalonego zbioru -

dziedziny. Encjom nadajemy nazwy, które powinny być rzeczownikami w liczbie pojedynczej. Nazwy encji piszemy wielkimi literami. W diagramie ER encję przedstawiamy w postaci prostokąta rysowanego linią pojedynczą (encja silna) lub podwójną (encja słaba).

Encja silna (regularna, właściwa) – oznacza obiekt, o którym informacja musi być znana i przechowywana. Istnieje niezależnie od innych obiektów. Reprezentowana jest przez prostokąt narysowany linią pojedynczą, ma nazwę, która jest rzeczownikiem w liczbie pojedynczej.

Encja słaba - oznacza obiekt, który może istnieć dopiero po zdefiniowaniu innych obiektów, np. dane dzieci pracownika można wprowadzać dopiero wtedy, gdy istnieją dane odpowiedniego pracownika. W tym wypadku encja DZIECKO byłaby encją słabą a encja PRACOWNIK encją silną. Encja słaba reprezentowana jest przez prostokąt narysowany linią podwójną. Nazwa encji jest rzeczownikiem w liczbie pojedynczej.



Rys. 2. Oznaczanie encji silnych i słabych w ERD

Atrybut – (*ang. attribute*) cecha charakterystyczna dla encji, szczegół służący do opisu, określania ilości lub wyrażenia stanu encji; dowolny opis mający znaczenie dla encji. Atrybut określa dokładnie jedną wartość, która z punktu widzenia projektanta jest niepodzielna (atomowa).

Atrybut opcjonalny (nieobowiązkowy) – (*ang. optional attribute*) atrybut, którego wartość może być nieokreślona, pusta, nieznana. Atrybut opcjonalny może przyjmować wartość NULL (wartość pustą, nieokreśloną).

Atrybut obligatoryjny (wymagany) – atrybut, którego wartość musi być zawsze określona. Nie może przyjmować wartości NULL.

Wartość NULL – wartość pusta interpretowana jako wartość nieznana, nieokreślona, niepewna, zastrzeżona. Dwa atrybuty o wartości NULL nie są traktowane jako sobie równe.

Uwaga. Date w pracy [Date2000], rozdz. 5. dokładnie omawia zagadnienia związane z wprowadzeniem pojęcia NULL do baz danych. Generalnie Date uważa ([Date2000], str.152), że problem jest nie w pełni zrozumiany i nierozwiązany a w związku z tym, pojęcie NULL nie powinno być wprowadzone do modelu relacyjnego. Stwierdza jednak, że Codd ma na ten temat zupełnie przeciwne zdanie.

Zwróćmy uwagę, że encja to nie to samo co typ encji. **Typ encji** jest kategorią a encja jest instancją danego typu encji [Beynon2000]. Typem encji (zbiorem encji) jest np. OSOBA a encją jest Jan Kowalski. Powszechnie używa się terminu encja zarówno na określenie typu encji (zbioru encji) jak i samej encji. Tam, gdzie nie prowadzi to do nieporozumień, będziemy używać pojęcia encja zamiast typ encji.

Wyodrębnione encje należy wypisać porządkując je np. alfabetycznie według nazw lub w inny uporządkowany logicznie sposób.

Opis encji można przedstawić w następujący sposób:

ENC/*x* NAZWA ENCJI

Semantyka encji – opis encji

Wykaz atrybutów – lista atrybutów, ich opis, dziedzina (typ). Listę atrybutów można przedstawić w formie *tabeli atrybutów* (Tabela 3).

Klucze kandydujące – zbiory atrybutów jednoznacznie identyfikujących encje

Klucz główny – wybrany klucz kandydujący identyfikujący encje

Charakter encji – encja silna lub słaba

gdzie *x* jest kolejnym numerem encji.

Precyzyjny opis encji, wszystkich atrybutów i ich dziedzin ułatwi zdefiniowanie modelu logicznego (struktur tabel).

Bardzo często na tym etapie wprowadza się **sztuczne klucze**, takie jak identyfikator encji (symbol, numer), jeśli klucze kandydujące są złożone z wielu atrybutów lub z jednego atrybutu, ale o dużej szerokości pola. Również dane osobowe szczególnie chronione nie powinny być polami kluczowymi np. NIP, PESEL, numer konta w banku itp. (szczególnie, gdy nie ma całkowitej pewności co do ich unikatowości).

Przykłady opisów encji:

ENC/001 OSOBA

Semantyka encji – encja zawiera dane dotyczące pracownika szkoły (nauczycieli, administracji, obsługi)

Wykaz atrybutów:

Tabela 3

Wykaz atrybutów encji OSOBA

Nazwa atrybutu	Opis atrybutu	Typ
NrOsoby	Identyfikator osoby	Liczba całkowita dodatnia
Nazwisko	Nazwisko osoby	Znakowy, max. 25 znaków
Imię	Imię osoby	Znakowy, max. 15 znaków
DataUrodzenia	Data urodzenia osoby	Data
NIP	Numer Identyfikacji Podatkowej	Znakowy {999-999-99-99}
PESEL	Numer identyfikacyjny PESEL	Ciąg 11 cyfr
Wykształcenie	Wykształcenie osoby	{podstawowe, średnie, wyższe}

Klucze kandydujące: NrOsoby, NIP, PESEL

Klucz główny: NrOsoby

Charakter encji: encja słaba

ENC/002 PRZEDMIOT

Semantyka encji – encja zawiera dane o przedmiocie nauczonym w szkole

Wykaz atrybutów:

Tabela 4

Wykaz atrybutów encji PRZEDMIOT

Nazwa atrybutu	Opis atrybutu	Typ
NrPrzedmiotu	Numer porządkowy	Liczba dwucyfrowa całkowita dodatnia
NazwaPrzedmiotu	Nazwa przedmiotu nauczanego w szkole	Znakowy, max. 30 znaków

Klucze kandydujące: NrPrzedmiotu

Klucz główny: NrPrzedmiotu

Charakter encji: encja silna



Aby stwierdzić, czy zdefiniowane encje są poprawne, trzeba odnieść się do analizy rzeczywistości. Gdybyśmy chcieli zdefiniować encję KSIĄŻKA, to z analizy rzeczywistości powinno wynikać, czy *wydawnictwo* powinno być postrzegane jako encja WYDAWNICTWO, czy raczej jako atrybut Wydawnictwo w 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 30 znakowych (nazwy wydawnictw) lub jako zbiór symboli (np. WNT, PWN) przypisanych poszczególnym wydawnictwom. Definiując WYDAWNICTWO jako encję można podać np. nazwę wydawnictwa, adres, telefon, e-mail, nazwisko i imię dyrektora, rok założenia wydawnictwa itp.

3.6.2. ZWIĄZKI I TYPY ZWIĄZKÓW

Związek (*ang. relationship*) jest to relacja (powiązanie) między encjami (najczęściej między dwiema). Związek binarny (między dwiema encjami) ma dwa końce. Związek musi mieć unikatową nazwę (niektóre notacje wprowadzają odrębne nazwy dla każdego końca). Nazwami związków powinny być czasowniki w 3. osobie liczby pojedynczej.

Opisując związki używamy pojęć takich jak **liczebność** oraz **typ uczestnictwa**.

Liczebność (stopień związku) (*ang. degree*) - określa liczbę encji uczestniczących w związku. Liczebność określamy dla każdego końca związku jako **jeden** lub **wiele** (przy czym jeden należy rozumieć jako jeden i tylko jeden a liczebność wiele należy rozumieć jako jeden lub więcej). Ze względu na liczebność wyróżniamy następujące typy związków:

1:1 – **jeden do jeden**, np.

jeden student ma tylko jeden indeks,
jeden indeks należy tylko do jednego studenta

1:N – **jeden do wiele**, np.

jeden pokój jest zamieszkiwany przez wielu studentów
jeden student mieszka tylko w jednym pokoju

N:N- **wiele do wiele**, np.

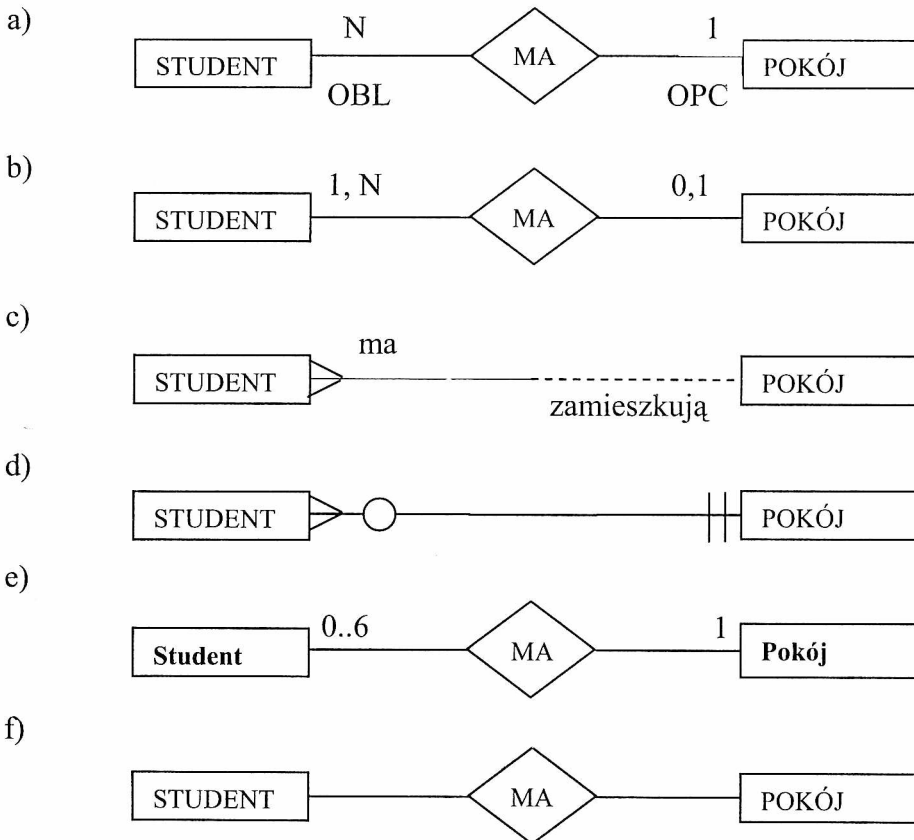
student może się zapisać na wiele wykładów
na wykład może się zapisać wielu studentów.

3. PROJEKTOWANIE KONCEPTUALNE

Uczestnictwo (opcjonalność/obligatoryjność) określa udział danej encji w związku. Uczestnictwo **obligatoryjne** (symbol 1 lub OBL obok specyfikowanej encji) oznacza, że wszystkie wystąpienia encji biorą udział w związku. Natomiast uczestnictwo **opcjonalne** (symbol 0 lub OPC obok encji) oznacza, że może istnieć encja nie uczestnicząca w związku.

Przykład związku:

Załóżmy, że w bazie danych studentów w akademiku obowiązują reguły przedstawione na Rys. 3.



Rys. 3. Różne sposoby przedstawienia liczebności i uczestnictwa

- a) notacja Chena (Date, Pankowski)
- b) notacja zmodyfikowana Chena
- c) notacja wg Barkera
- d) notacja Martina
- e) notacja UML
- f) notacja Yourdona

Student musi mieć przypisany numer pokoju – każdy student bierze udział w związku czyli STUDENT uczestniczy obligatoryjnie w związku MA.

Pokój może nie być przypisany do żadnego studenta – może być pokój, który nie uczestniczy w związku czyli encja POKÓJ uczestniczy opcjonalnie w związku MA.

Uwaga. Sposób interpretowania i zaznaczania opcjonalności i obligatoryjności na diagramach nie jest jednoznaczny. Różni autorzy w sposób odmienny definiują te pojęcia. Dla przykładu:

- a) Date wzorując się na pracy Chena ([Date2000] str. 398-400, [Chen1976]) definiuje uczestnictwo encji E jako obligatoryjne (OBL po stronie encji E, wymagane – *ang. total*) w związku R, jeśli każda instancja encji E bierze udział w co najmniej jednej instancji związku R; w przeciwnym razie jest ono opcjonalne (skrót OPC po stronie encji E, *ang. partial*). Notację Chena stosuje również Pankowski [Pankow1992] (Rys. 3a).
- b) Notacja wyprowadzona od Chena przez zastąpienie skrótu OPC oznaczającego opcjonalność przez symbol 0 oraz skrót OBL (obligatoryjność) przez symbol 1 (Rys. 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 A mogą istnieć niezależnie od B, to związek jest opcjonalny (Rys. 3c).
- d) Hernandez ([Hernan1998], str. 46, 62), który wykorzystuje metodykę Martina, definiuje uczestnictwo encji w związku jako obowiązkowe (symbol | przy encji), jeśli encja musi wystąpić, zanim określi się dane encji po drugiej stronie związku (uczestnictwo opcjonalne zaznacza się symbolem O (Rys. 3d).
- e) W **UML** (*Unified Modelling Language*) liczebność nazywa się krotnością ([Muller2000] str. 184, [Subieta1998] str. 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	oznacza jeden

(Rys. 3e).

- f) Edward Yourdon ([Yourdon1996], rozdz. 12) uważa, że na diagramach związków-encji (E-R) nie powinno umieszczać się żadnych dodatkowych informacji takich jak liczebność, 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”. Diagram E-R w notacji Yourdona przedstawia rysunek 3f.
- g) W pracy [Olle1982] jest podanych 13 różnych metod przedstawiania encji i związków na diagramach ERD.

□

Informacje zawarte na diagramach rysunku 3 można przeczytać następująco:

każdy student **musi** mieć przydzielony dokładnie jeden numer pokoju natomiast pokój **może** zamieszkiwać od 0 do n studentów, czyli pokój może być zamieszkanym przez wielu studentów lub nie zamieszkanym wcale.

Uwaga. Aby uprościć transformację modelu konceptualnego do modelu relacyjnego (logicznego) zaleca się zastępowanie związków N:N dwoma związkami 1:N, w wyniku czego pojawia się encja powiązania (asocjacyjna).

Teoretycznie można wyrysować na kartce wszystkie możliwe kombinacje typów związków biorąc pod uwagę ich liczebność i uczestnictwo. W praktyce niektórych nie spotyka się wcale, niektóre rzadko a niektóre bardzo często (rozdz. 4.1.1).

Nie zawsze zakwalifikowanie danej (jako encję, atrybut czy związek) jest rzeczą jednoznaczną. 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. data rozwodu)
- rozwód może być związkiem między encją Kobieta i encją Mężczyzna.

Wybór odpowiedniej decyzji, jak zaklasyfikować dane, należy do projektanta bazy danych (po przeprowadzonej wnikliwiej analizie faktów wycinka świata rzeczywistego).

Po wyodrębnieniu encji i związków możemy przejść do kolejnego etapu polegającego na podaniu definicji predykatowych encji i związków.

3.6.3. DEFINICJE PREDYKATOWE ENCJI I ZWIĄZKÓW

W tym rozdziale omówimy sposób zwięzłego przedstawiania typów encji oraz związków między nimi.

3.6.3.1. Definicje predykatowe typów encji

Definicje predykatowe typów encji pozwalają w zwarty i czytelny sposób opisać encje występujące w diagramie związków encji oraz na szybką, niemalże mechaniczną transformację modelu konceptualnego do modelu logicznego.

Na podstawie opisu encji sporządzamy zestawienie typów encji. W tym celu wypisujemy kolejne nazwy typów encji (najlepiej uporządkowane alfabetycznie, co ułatwia odszukanie wybranego typu encji) oraz w nawiasach podajemy nazwy atrybutów. Klucz główny encji zaznaczamy przez podkreślenie i umieszczamy na początku listy atrybutów.

Opis encji można przedstawić w następujący sposób:

ENC/ x NAZWA TYPU ENCJI (lista atrybutów)

gdzie x jest kolejnym numerem typu encji.

Przykład definicji predykatowej typów encji:

ENC/001 OSOBA (Id osoby, Nazwisko, Imię, Data urodzenia)

ENC/002 JĘZYK (Id języka, Nazwa)

□

3.6.3.2. Definicje predykatowe typów związków

Podobnie jak w przypadku definicji predykatowych typów encji, tak i w przypadku definicji predykatowych typów związków, należy wypisać kolejne nazwy związków zachodzących między encjami (najlepiej uporządkowane alfabetycznie, co ułatwia odszukanie wybranego związku) oraz w nawiasach wypisać nazwy encji, między którymi związek zachodzi,

3. PROJEKTOWANIE KONCEPTUALNE

uczestnictwo i liczebność po stronie danej encji, kursywą nazwy atrybutów łączących oraz normalną czcionką nazwy atrybutów dodatkowych. Nazwa związku wraz z nazwami powiązanych z nią encji ma określone znaczenie semantyczne.

Ogólna postać definicji związków:

ZWI/*x* Nazwa związku(ENCJA1(*u*, *l*) : ENCJA2(*u*, *l*); lista)

gdzie:

x - kolejny numer związku,

u - uczestnictwo,

l - liczebność,

lista - lista atrybutów łączących i atrybutów dodatkowych związku.

Przykład definicji predykatowej związku:

ZWI/001 Zna((OSOBA(0,N) : JĘZYK(0,N); *Id osoby*, *Id języka*,
Stopień znajomości)

□

Przykład definicji predykatowej encji i związków:

ENC/001 OSOBA (*Id osoby*, Nazwisko, Imię, Data urodzenia, NIP,
PESEL)

ENC/002 WYKSZTAŁCENIE (*Id poziomu*, Nazwa)

ENC/003 PRZEDMIOT (*Numer przedmiotu*, Nazwa przedmiotu)

ZWI/001 Ma (OSOBA(0, N) : WYKSZTAŁCENIE(1, 1); *Id poziomu*)

ZWI/002 Naucza (OSOBA(0, N) : PRZEDMIOT(0, N); *Id wpisu*, *Id osoby*,
Numer przedmiotu, Liczba godzin, Klasa)

□

3.6.4. DIAGRAM ZWIĄZKÓW ENCJI

Model konceptualny najlepiej jest wyrazić w sposób graficzny za pomocą **diagramu związków encji** (ERD – *ang. Entity Relationship Diagram*). Dzięki ERD w czytelny i przejrzysty sposób przedstawiamy wyodrębnione we wcześniejszych fazach encje oraz związki między nimi. Często na diagramie w encjach są wyszczególniane ich atrybuty i klucze. Każda encja na diagramie powinna być powiązana z co najmniej jedną inną encją. Bywa jednak tak, że musimy rozstrzygnąć, czy dwie encje, które są już połączone w sposób pośredni, połączyć również w sposób bezpośredni. Aby podjąć właściwą decyzję, należy rozważyć rozmiar zajmowanej pamięci oraz czas dostępu do danych. Generalnie należy się kierować zasadą, aby uzyskany diagram był poprawny a jednocześnie czytelny.

Przykład diagramu związków encji:

Załóżmy, że projektujemy bazę danych **WYPOŻYCZALNIA_1** dla wypożyczalni kaset video i w wyniku analizy rzeczywistości wyodrębniliśmy cztery encje (**KLIENT**, **KASETA**, **FILM** i **GATUNEK**) i trzy związki między encjami (**Zawiera**, **Jest**, **Wypożycza**).

Definicje predykatowe wyodrębnionych encji i związków mają następującą postać:

ENC/001 KLIENT (NrKlienta, Nazwisko, Imię, Miasto, Ulica_nr)

ENC/002 KASETA (NrKasety, Opis)

ENC/003 FILM (KodFilmu, Tytuł, CzasTrwania)

ENC/004 GATUNEK (KodGatunku, Rodzaj)

ZWI/001 Zawiera (KASETA(1,N) : FILM(1,1); *KodFilmu*)

ZWI/002 Jest (FILM(0,N) : GATUNEK(1,1); *Kod_gatunku*)

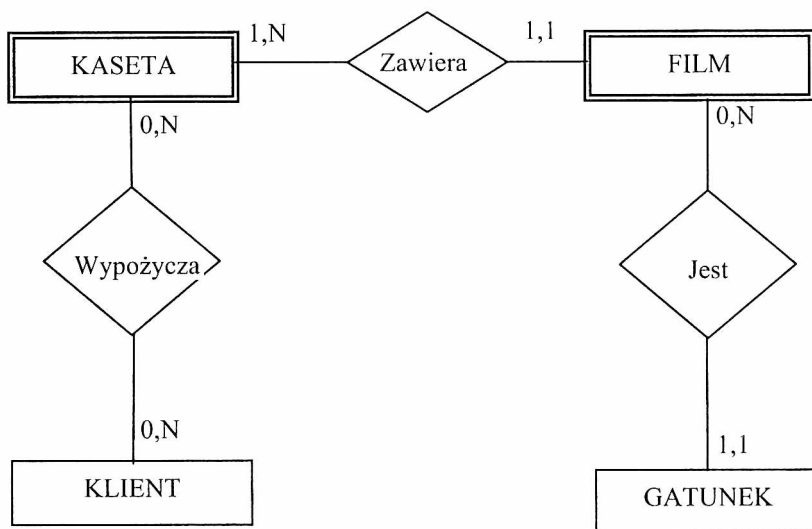
ZWI/003 Wypożycza (KLIENT(0,N) : KASETA(0,N); *NrKlienta*,
NrKasety, *NrWypożyczenia*, *DataWypożyczenia*, *DataZwrotu*, *Oplata*)

Zwróćmy uwagę, że związek **Wypożycza**, jest związkiem opcjonalnym wiele do wiele (jeden klient może wypożyczyć wiele kaset lub żadną, kaseta

3. PROJEKTOWANIE KONCEPTUALNE

z wypożyczalni może być nie wypożyczona dotychczas ani razu lub mogła już być wypożyczana przez wielu różnych klientów).

Rysunek 4a przedstawia diagram związków encji dla bazy **WYPOŻYCZALNIA_1**.



Rys. 4a. Diagram związków encji dla bazy WYPOŻYCZALNIA_1

Zgodnie z uwagą zawartą w rozdz. 3.6.2, zalecającą zastępowanie związków N:N dwoma związkami 1:N, należałoby przekształcić diagram z Rys. 4a i zastąpić związek **Wypożycza** encją WYPOŻYCZENIA (taką encję nazywamy *encją powiązania* lub *encją asocjacyjną*). Otrzymalibyśmy wówczas bazę danych **WYPOŻYCZALNIA_2** zawierającą następujące encje i związki:

ENC/001 KLIENT (NrKlienta, Nazwisko, Imię, Miasto, Ulica_nr)

ENC/002 KASETA (NrKasety, Opis)

ENC/003 FILM (KodFilmu, Tytuł, CzasTrwania)

ENC/004 GATUNEK (KodGatunku, Rodzaj)

ENC/005 WYPOŻYCZENIA (NrWypożyczenia, DataWypożyczenia,
DataZwrotu, Opłata)

ZWI/001 Zawiera(KASETA(1,N) : FILM(1,1); *KodFilmu*)

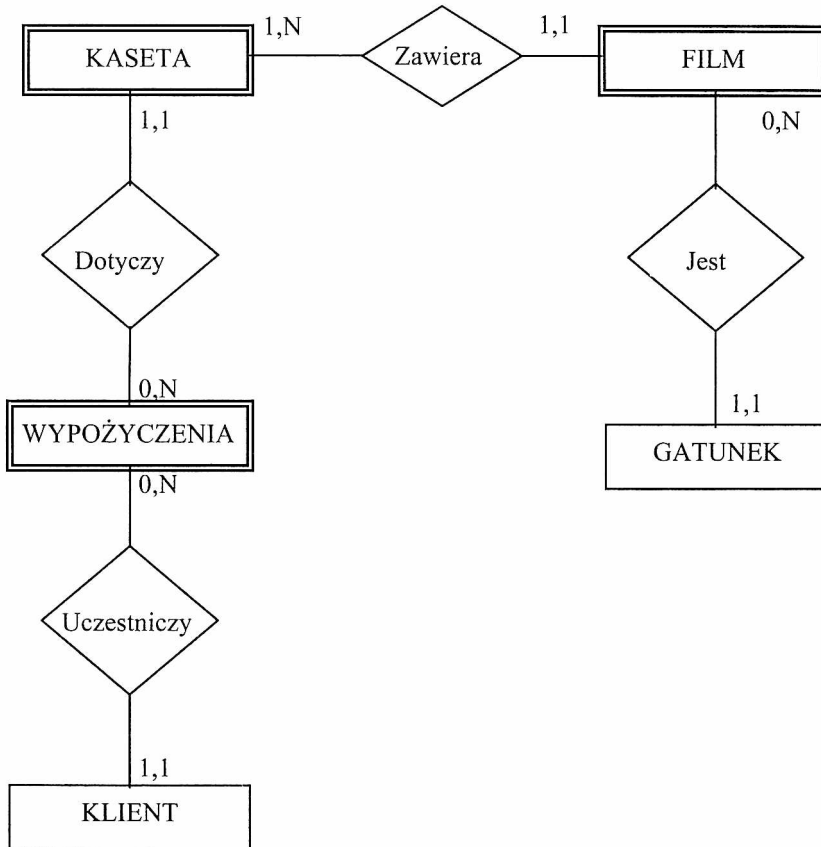
ZWI/002 Jest (FILM(0,N) : GATUNEK(0,N); *Kod_gatunku*)

3. PROJEKTOWANIE KONCEPTUALNE

ZWI/003 Uczestniczy (KLIENT(1,1) : WYPOŻYCZENIA(0,N);
NrKlienta)

ZWI/004 Dotyczy (WYPOŻYCZENIA (0,N) : KASETA(1,1); *NrKasety*)

Rysunek 4b przedstawia diagram związków encji dla bazy WYPOŻYCZALNIA_2.



Rys. 4b. Diagram związków encji dla bazy WYPOŻYCZALNIA_2

□

Czytelników zaintrygowanych związkiem **Zawiera**, który jest obligatoryjny po stronie filmu i kasyty odsyłamy do rozdz. 4.1.1 gdzie jest przeprowadzona jego analiza.

3.6.5. NORMALIZACJA SCHEMATÓW RELACJI

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

Normalizacja opiera się na idei funkcjonalnej zależności atrybutów i polega na dekompozycji początkowego schematu relacji na kilka innych schematów relacji. Postacie normalne są kolejno numerowane i 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.

Przeprowadzenie procesu **normalizacji** powoduje, że z początkowych schematów relacji wyłaniają się nowe schematy, wcześniej niedostrzegane lub pominięte. Na ogół normalizacja powoduje powstanie większej liczby schematów relacji i odpowiednio większej liczby tablic w relacyjnym logicznym modelu bazy danych. 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 zamieszczonych na końcu pracy pozycjach literatury (np. [Beynon1999, Beynon2000, Date2000, Harris1994, Pankow1992, Riordan2000, Roszko1998]).

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

Mówimy, że atrybut B relacji R jest *funkcyjnie zależny* od atrybutu A tej relacji, co zapisujemy

$$A \rightarrow B$$

jeśli zawsze każdej wartości *a* atrybutu A odpowiada nie więcej niż 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:

W schemacie relacji

OSOBA (PESEL, Nazwisko)

atrybut PESEL jednoznacznie określa 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).

□

Kolejnym ważnym pojęciem jest pełna funkcyjna zależność atrybutów.

Atrybut B relacji R jest w *pełni funkcyjnie zależny* od zbioru atrybutów A, co zapisujemy

$A \Rightarrow B$,

jeśli jest funkcyjnie zależny od A, ale nie jest funkcyjnie zależny od żadnego podzbioru zbioru A.

Przykład:

Dany jest schemat relacji

EGZAMINY (Student, Przedmiot, Data, Ocena),

Do przechowywania danych o ocenach końcowych studentów z różnych przedmiotów.

W schemacie relacji zachodzą zależności:

{Student, Przedmiot} $\Rightarrow$ Data

{Student, Przedmiot} $\Rightarrow$ Ocena

3. PROJEKTOWANIE KONCEPTUALNE

czyli atrybut Data (podobnie jak atrybut Ocena) jest w pełni funkcyjnie zależny od klucza złożonego {Student, Przedmiot} (nie jest funkcyjnie zależny ani od atrybutu Student ani od atrybutu Przedmiot).

□

W rozdziale 2 zdefiniowaliśmy pojęcie klucza relacji jako zbiór identyfikujący relacji, którego żaden podzbiór nie jest zbiorem identyfikującym, czyli jest to atrybut (lub kilka atrybutów), którego wartość jednoznacznie identyfikuje każdą krotkę w relacji. Czyli kluczem relacji jest taki zbiór identyfikujący relację, którego żaden podzbiór nie jest zbiorem identyfikującym relację (minimalny zbiór identyfikujący).

Zauważmy, że jeśli klucz główny relacji jest kluczem prostym, to wszystkie zależności funkcyjne względem klucza są zależnościami pełnymi.

Poniżej podamy definicje postaci normalnych schematów relacji:

Pierwsza postać normalna (1PN) – schemat relacji jest w pierwszej postaci normalnej wtedy i tylko wtedy, gdy każda wartość z dziedziny atrybutów jest elementarna (atomowa, nierozkładalna). Inaczej można to wyrazić, że nie ma powtarzających się grup wartości tzn. że każdemu elementowi relacji znajdującemu się na przecięciu dowolnej krotki i dowolnego atrybutu odpowiada pojedyncza wartość a nie zbiór wartości.

Przykłady:

a) schematy relacji w pierwszej postaci normalnej:

KLIENT (NrKlienta, Nazwisko, Imię, Miasto, UlicaNr)

KASETA (NrKasety, Opis)

FILM (KodFilmu, Tytuł, CzasTrwania)

GATUNEK (KodGatunku, Rodzaj)

Każdemu atrybutowi powyższych relacji odpowiada zawsze dokładnie jedna wartość dla dowolnie wybranej krotki. Przykładowe dane relacji KLIENT przedstawia Tabela 5.

Tabela 5

Relacja KLIENT w pierwszej postaci normalnej

NrKlienta	Nazwisko	Imię	Miasto	UlicaNr
0001	Batycki	Jerzy	Opole	Polna 34

3. PROJEKTOWANIE KONCEPTUALNE

0002	Batycki	Marek	Opole	Polna 34
0003	Widecka	Barbara	Sopot	Rynek 1/1
0004	Artecka	Barbara	Opole	Rynek 2

b) schemat relacji nie będący w pierwszej postaci normalnej:

OSOBA (NrOsoby, Nazwisko, Imię, ImionaDzieci, DatyUrDzieci)

Tabela 6

Relacja OSOBA nieznormalizowana

NrOsoby	Nazwisko	Imię	ImionaDzieci	DatyUrDzieci
1	Adurska	Janina		
2	Kadurska	Genowefa	Adam	10-03-1960
3	Padurska	Maria	Adam	11-05-1950
			Ewa	01-01-1954
			Hanna	04-02-1960
4	Wadurska	Teresa	Monika	03-08-1962
			Roland	03-08-1962

Relacja nie jest w pierwszej postaci normalnej ponieważ np. atrybutowi ImionaDzieci dla trzeciej krotki odpowiada zbiór wartości {Adam, Ewa, Hanna} a nie pojedyncza wartość.

c) schemat relacji OSOBA w pierwszej postaci normalnej:

OSOBA (NrOsoby, Nazwisko, Imię, ImięDziecka, DataUrDziecka)

Tabela 7

Znormalizowana relacja OSOBA (w pierwszej postaci normalnej)

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

Analizując relację OSOBA można zauważyć, że pewne dane się powtarzają co znacznie wydłuża i utrudnia wykonywanie operacji na danych np. aktualizowanie.

□

Druga postać normalna (2PN) – schemat relacji jest w drugiej postaci normalnej, jeśli każdy atrybut schematu relacji, nie wchodzący w skład żadnego klucza kandydującego, jest w pełni funkcyjnie zależny od wszystkich kandydujących.

Zauważmy, że schematy relacji, które są w pierwszej postaci normalnej i których wszystkie klucze kandydujące są kluczami prostymi, są również w drugiej postaci normalnej.

W celu uzyskania drugiej postaci normalnej należy podzielić daną relację na kilka relacji (na ogół o mniejszej liczbie atrybutów) tak, aby wszystkie atrybuty w poszczególnych relacjach były w pełni funkcyjnie zależne od klucza. Symbolicznie możemy to zapisać następująco:

Schemat relacji $R(\underline{A}, \underline{B}, C, D)$, w którym zachodzą zależności

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

$$\{\underline{A}, \underline{B}\} \rightarrow D,$$

$$A \rightarrow D,$$

należy zastąpić dwoma schematami:

$R1(\underline{A}, \underline{B}, C)$, w którym $\{\underline{A}, \underline{B}\} \rightarrow C$,

oraz

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

Przykład:

Dany jest schemat relacji

OSOBA (PESEL, Nazwisko, Imię, ImięDziecka, DataUrDziecka) do przechowywania danych o dzieciach pracowników. Każda osoba, której dane przechowujemy, ma co najmniej jedno dziecko.

W schemacie relacji OSOBA istnieją następujące zależności funkcyjne:

$$\text{PESEL} \rightarrow \text{Nazwisko}$$

$$\text{PESEL} \rightarrow \text{Imię},$$

$$\text{PESEL}, \text{ImięDziecka} \rightarrow \text{DataUrDziecka}$$

Jedynym kluczem relacji OSOBA jest zbiór $\{\text{PESEL}, \text{ImięDziecka}\}$. Atrybuty niekluczowe Nazwisko, Imię nie są w pełni funkcyjnie zależne od klucza (bo zależą funkcyjnie od atrybutu PESEL) czyli schemat relacji OSOBA nie jest w drugiej postaci normalnej. Dlatego należy podzielić go na dwa schematy relacji:

3. PROJEKTOWANIE KONCEPTUALNE

DOROSŁY (PESEL, Nazwisko, Imię)

DZIECKO (PESEL, ImięDziecka, DataUrDziecka)

które są w drugiej postaci normalnej (dlaczego?).

□

Zanim przejdziemy do omówienia trzeciej postaci normalnej wprowadzimy pojęcie przechodniej zależności funkcyjnej.

Niech A, B, C będą trzema rozłącznymi podzbiorami atrybutów danej relacji. Podzbiór atrybutów C jest **przechodnio (tranzytywnie) funkcyjnie zależny** od podzbioru atrybutów A (*ang. transitively dependent*), co zapisujemy

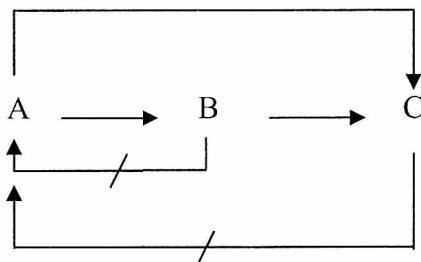
$$A \rightarrow C$$

jeśli

$$A \rightarrow B, B \rightarrow C, B \nrightarrow A, C \nrightarrow A$$

(B zależy funkcyjnie od A, C zależy funkcyjnie od B, A nie zależy funkcyjnie od B i A nie zależy funkcyjnie od C).

Graficznie przechodnią funkcyjną zależność przedstawia Rys. 8.



Rys. 8. Przechodnia zależność funkcyjna atrybutu C od A

Przykład:

W schemacie relacji

FAKTURA (NrFaktury, NazwaKlienta, AdresKlienta)

istnieją następujące zależności funkcyjne:

$\text{NrFaktury} \rightarrow \text{NazwaKlienta}$

$\text{NazwaKlienta} \rightarrow \text{AdresKlienta}$

Atrybut AdresKlienta jest przechodnio funkcyjnie zależny od NrFaktury ponieważ zależy od atrybutu NazwaKlienta, który jest funkcyjnie zależny od NrFaktury. Z kolei NrFaktury nie jest funkcyjnie zależny od atrybutu

NazwaKlienta ponieważ jeden klient może mieć wiele faktur. Również NrFaktury nie zależy funkcyjnie od atrybutu AdresKlienta ponieważ dany adres może być na wielu fakturach.

□

Trzecia postać normalna (3PN) – 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 funkcyjnie zależny od żadnego klucza kandydującego (czyli wszystkie atrybuty niekluczowe są wzajemnie niezależne).

Aby doprowadzić do trzeciej postaci normalnej schemat relacji, w którym atrybuty są w przechodniej zależności funkcyjnej od klucza, należy podzielić go na nowe schematy relacji, np.

ze schematu relacji $R(\underline{A}, B, C)$

gdzie

$\underline{A} \rightarrow B,$

$B \rightarrow C,$

czyli $\underline{A} \rightarrow C$

należy wyodrębnić dwa schematy relacji $R1(\underline{A}, B)$ oraz $R2(\underline{B}, C)$.

Przykład:

Aby schemat relacji **FAKTURA** (NrFaktury, NazwaKlienta, AdresKlienta) doprowadzić do trzeciej postaci normalnej należy podzielić go na dwa schematy relacji:

WYKAZ (NrFaktury, NazwaKlienta)

KLIENT (NazwaKlienta, AdresKlienta)

□

Przykład:

Założmy, że dla schematu relacji

ZAJĘCIA (Student, Kurs, Wykładowca),

zachodzą zależności funkcyjne:

$\{\underline{\text{Student}}, \underline{\text{Kurs}}\} \rightarrow \text{Wykładowca}$ (student może uczęszczać na dany kurs tylko do jednego wykładowcy);

$\text{Wykładowca} \rightarrow \text{Kurs}$ (każdy wykładowca prowadzi dokładnie jeden kurs).

W schemacie relacji **ZAJĘCIA**, który jest już w trzeciej postaci normalnej, występują nadal anomalie związane z usuwaniem danych (po usunięciu

3. PROJEKTOWANIE KONCEPTUALNE

informacji o ostatnim studencie uczęszczającym na dany kurs tracimy informacje o kursie) oraz anomalie dopisywania danych (nie można dopisać kursu i wykładowcy, zanim na kurs nie zapisze się co najmniej jeden student).

□

Poszczególne etapy normalizacji przedstawia Tabela 8.

Tabela 8

Etapy normalizacji	
Relacja nieznormalizowana	Wartości atrybutów nie są atomowe
Pierwsza postać normalna	Wszystkie atrybuty relacji są atomowe
Druga postać normalna	Atrybuty niekluczowe relacji są w pełni funkcyjnie zależne od jej klucza
Trzecia postać normalna	Relacja jest w drugiej postaci normalnej i każdy jej atrybut niekluczowy jest bezpośrednio zależny od klucza

Ideę procesu normalizacji można zawrzeć w punktach [Beynon2000]:

- Bez powtórzeń (1PN)
- Atrybuty zależą od klucza (2PN)
- Od całego klucza (3PN)
- I niczego innego, tylko klucza (PNBC).

W praktyce normalizację schematu relacji kończy się najczęściej na trzeciej postaci normalnej. W teorii relacyjnych baz danych definiuje się kolejne postaci, które krótko przedstawimy poniżej.

Postać normalna Boyce'a-Codda (PNBC) jest to poprawiona wersja trzeciej postaci normalnej. Schemat relacji R jest w postaci normalnej Boyce'a-Codda, jeżeli $A \rightarrow B$ i B nie zawiera się w A , to zbiór A zawiera klucz (tzn. A jest nadkluczem), przy czym A i B są zbiorami atrybutów relacji R . Wszystkie zależności funkcyjne w schemacie relacji determinowane są przez klucze (**wszystkie atrybuty zależą tylko i wyłącznie od klucza**).

W praktyce, aby można było schemat relacji sprowadzić do postaci Boyce'a-Codda muszą być spełnione trzy warunki:

1. w schemacie relacji muszą być co najmniej dwa klucze kandydujące,
2. co najmniej dwa klucze kandydujące muszą być kluczami złożonymi,

3. klucze kandydujące muszą mieć wspólne atrybuty.

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

Przykład:

Schemat 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),
nie jest w postaci normalnej Boyce'a-Codda (atrybut Kurs zależy od atrybutu niekluczowego Wykładowca) i dlatego występują anomalie usuwania i dopisywania danych.

Schemat ten można sprowadzić do postaci Boyce'a-Codda, ponieważ są spełnione warunki:

1. w schemacie relacji są dwa klucze kandydujące {Student, Kurs} i {Student, Wykładowca},
2. obydwa klucze kandydujące są kluczami złożonymi,
3. klucze kandydujące mają wspólny atrybut {Student}.

Schemat relacji **ZAJĘCIA** powinien zostać przekształcony do postaci:

PRZYDZIAŁ (Wykładowca, Kurs),

ZAJĘCIA (Student, Kurs).

□

Wielowartościowa zależność oznacza, że w schemacie relacji, który zawiera atrybuty A i B, atrybut B jest wielowartościowo zależny od atrybutu A, co zapisujemy

$A \rightarrow\rightarrow B$

wtedy i tylko wtedy, gdy jednej wartości atrybutu A odpowiada wiele wartości atrybutu B.

Przykład:

Schemat relacji

KLIENT (Nazwisko, Adres, ImięDziecka)

zawiera dwie wielowartościowe zależności:

Nazwisko $\rightarrow\rightarrow$ Adres

3. PROJEKTOWANIE KONCEPTUALNE

Nazwisko $\rightarrow \rightarrow$ ImięDziecka
czyli atrybutowi Nazwisko może odpowiadać wiele wartości atrybutu Adres lub ImięDziecka (Tabela 9).

Tabela 9

Przykładowe dane relacji KLIENT

Nazwisko	Adres	ImięDziecka
Adurska	Opole, Nyska 2	
Kadurska	Legnica, Stara 12/02	Adam
Kadurska	Wrocław, Pawia 1	Adam
Padurska	Sopot, Morska 123	Adam
Padurska	Warszawa, Stara 3	Adam
Padurska	Sopot, Morska 123	Ewa
Padurska	Warszawa, Stara 3	Ewa
Padurska	Sopot, Morska 123	Hanna
Padurska	Warszawa, Stara 3	Hanna
Wadurska	Wrocław, Rynek 1/1	Monika
Wadurska	Wrocław, Rynek 1/2	Monika
Wadurska	Wrocław, Rynek 1/1	Roland
Wadurska	Wrocław, Rynek 1/2	Roland



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

Zależność funkcyjna $\{A_1, A_2, \dots, A_n\} \rightarrow \{B_1, B_2, \dots, B_m\}$ jest **trywialna** wtedy i tylko wtedy, gdy prawa strona zależności jest podzbiorem lewej strony czyli $\{B_1, B_2, \dots, B_m\} \subseteq \{A_1, A_2, \dots, A_n\}$. Innymi słowy, zależność jest trywialna (oczywista), jeżeli zawsze jest spełniona (nie może być nie spełniona).

Sprowadzenie schematu relacji do czwartej postaci normalnej sprowadza się do usunięcia wielokrotnych wielowartościowych zależności funkcyjnych.

Aby zdefiniować piątą postać normalną wprowadzimy pojęcie zależnościłączenia.

Niech $A, B, \dots, Z$ będą dowolnymi podzbiorami zbioru atrybutów schematu relacji R . Mówimy, że relacja R spełnia **zależność złączenia** ($A, B, \dots, Z$) wtedy i tylko wtedy, gdy R jest równe złączeniu swoich rzutów na $A, B, \dots, Z$.

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. Inaczej można to wyrazić, że schemat, który jest w czwartej postaci normalnej jest w piątej postaci normalnej, jeżeli nie istnieje odwracalny rozkład na schematy relacji.

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

Przykład:

Aby schemat relacji

DOSTAWY (Dostawca, Towar, Odbiorca)

doprowadzić do piątej postaci normalnej należy wykonać dekompozycję na trzy schematy relacji

KTO_CO (Dostawca, Towar)

KTO_KOMU (Dostawca, Odbiorca)

CO_KOMU (Towar, Odbiorca)

□

Na zakończenie tego rozdziału pragniemy podkreślić, że normalizacja jest procesem tworzenia schematów relacji, które nie podlegają anomalii. Dane przechowywane w relacjach znormalizowanych zajmują mniej miejsca, nie powtarzają się, łatwiejsza jest ich modyfikacja, dopisywanie oraz ich usuwanie.

W praktyce często po przeprowadzonej normalizacji danych przeprowadza się denormalizację danych [Barker1996, Harris1994] polegającą na łączeniu niektórych znormalizowanych relacji w celu przyspieszenia operacji wykonywanych na danych. Jeśli projektant przewiduje, że w zaimplementowanym systemie często będą wykonywane operacje łączenia relacji, które są bardzo czasochłonne, to musi dokonać wyboru, czy lepiej dokonać denormalizacji (pojawią się znowu anomalie, które eliminowała normalizacja) czy pogodzić się z wolniejszym działaniem systemu.

4. PROJEKTOWANIE LOGICZNE

Projektowanie logiczne bazy danych polega na transformacji modelu konceptualnego bazy (rozdz. 3.6) do modelu logicznego (na przykład relacyjnego), w którym określamy strukturę przechowywanych danych, czyli tzw. schematy relacji. Na tym etapie powinno się przeprowadzić normalizację schematów relacji. Jak już wspomnieliśmy wcześniej (rozdz. 3.6.5) normalizacja danych jest sformalizowaną procedurą, w wyniku której eliminujemy redundancje (powtarzanie się) danych, ich niespójności i anomalie, czyli nieprawidłowości związane z aktualizacją, usuwaniem i dopisywaniem danych.

Projektowanie logiczne bazy danych ma na celu skonstruowanie bazy, która określi sposób zgrupowania danych i powiązań między nimi (rozdz.4.1.2). Transformacja modelu konceptualnego bazy danych do modelu logicznego jest wykonywana według pewnych reguł, które są opisane w rozdziale 4.1.1. Model logiczny bazy danych można zwięźle przedstawić podając schemat bazy danych (rozdz. 4.1.3). W kolejnym etapie projektowania logicznego bazy danych należy określić grupy użytkowników bazy danych i wyodrębnić perspektywy (rozdz. 4.2).

4.1. TRANSFORMACJA MODELU KONCEPTUALNEGO DO MODELU LOGICZNEGO

W rozdziale tym omówimy, w jaki sposób przeprowadzamy transformację modelu konceptualnego do relacyjnego modelu logicznego będącego zbiorem struktur danych – najczęściej tabel. Jeśli mamy poprawnie zdefiniowane encje i związki między nimi, to na ogół otrzymujemy schematy logiczne w trzeciej postaci normalnej.

4.1.1. REGUŁY TRANSFORMACJI

Transformację modelu konceptualnego do modelu logicznego przeprowadzamy według niżej podanych reguł.

Podstawowe reguły transformacji modelu konceptualnego do modelu logicznego:

1. Dla każdej encji z diagramu ER tworzymy relację (tabelę). Dobrym zwyczajem jest nazwać ją tak jak encję, tylko w liczbie mnogiej ze względu na to, że zawiera ona wiele wystąpień obiektu.
2. Atrybuty encji stają się kolumnami tabeli o tej samej nazwie. Dla kolumn dobieramy odpowiednie formaty danych. Kolumny odpowiadające kluczom głównym encji stają się kluczami głównymi tabel. Atrybuty opcjonalne stają się kolumnami o dopuszczalnych wartościach NULL, atrybuty obligatoryjne (wymagane) stają się kolumnami NOT NULL.
3. Dla każdego związku binarnego *jeden do wiele* (*jeden do jeden*) obligatoryjnego po stronie *wiele*, wstawiamy klucz główny ze strony *jeden* do tabeli reprezentującej stronę *wiele* linii związku. W ten sposób otrzymujemy klucz obcy (*ang. foreign key*) w tabeli ze strony *wiele* linii związku.
4. Związek binarny typu *jeden do wiele*, opcjonalny po stronie *wiele*, reprezentujemy zazwyczaj nowym schematem relacji, w którym umieszczamy klucze główne obu encji. Jeżeli co najmniej 60% instancji encji pozostaje w relacji, to uznaje się związek za prawie obligatoryjny i zaleca się zastosować transformację według reguły 3.
5. Dla związku *wiele do wiele* tworzymy tabelę (ewentualnie ze sztucznym kluczem głównym). Klucze główne z encji po obu stronach związku stają się kluczami obcymi. Jeśli związek posiadał atrybuty dodatkowe, to stają się one kolumnami tabeli.

Powyższe zasady ogólne transformacji na ogół dają dobre efekty, ponieważ uwzględniają dwa typy związków spotykane najczęściej:

- a) związek *jeden do wiele* opcjonalny po stronie *wiele* (Rys. 9),
np. pokój (encja A) zamieszkiwany jest przez żadnego lub wielu studentów, student (encja B) zamieszkuje jeden pokój,



Rys. 9. Przykład związku 1:N, opcjonalnego po stronie wiele

- b) związek *wiele do wiele* opcjonalny po obu stronach (Rys. 10), np. klient (encja A) mógł dotychczas nie wypożyczyć jeszcze żadnej kasety video (encja B) lub wypożyczyć wiele kaset video; kaseata mogła być wypożyczana przez wielu klientów (po oddaniu kasety danych o wypożyczeniu się nie usuwa) lub ani razu jeszcze nie wypożyczona.



Rys. 10. Przykład opcjonalnego związku N:N

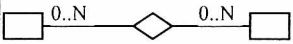
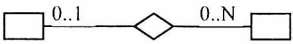
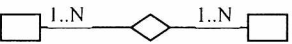
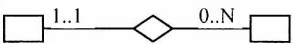
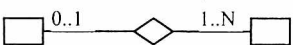
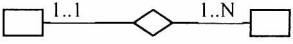
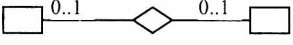
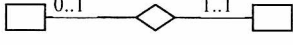
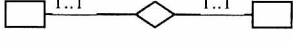
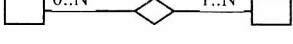
W szczególnych przypadkach podane zasady transformacji mogą się okazać jednak nie wystarczające, ponieważ nie wszystkie uwzględniają uczestnictwo encji w związku (obligatoryjność/opcjonalność). W Tabeli 10 przedstawiamy związki występujące w praktyce często, rzadko oraz związki niemożliwe [Barker1996].

Przypadek związku *wiele do wiele* obligatoryjny po obu stronach uważa się w praktyce za niemożliwy. Związek ten oznacza, że żadna encja po jednej stronie związku nie może istnieć bez encji po drugiej stronie związku. W praktyce, po szczegółowej analizie takiego związku okazuje się, że taka konstrukcja nie oddaje dobrze rzeczywistości i jest przyczyną powstawania anomalii (np. usuwania).

Tabela 10

Występowanie różnych typów związków

4. PROJEKTOWANIE LOGICZNE

Związki występujące często	Związki występujące rzadko	Związki niemożliwe
		
		
		
		
		
		
		

Rozważmy na przykład (Rys. 11) rzadko, ale jednak możliwą do wystąpienia, sytuację pierwszą z Tabeli 10:



Rys. 11. Przykład obustronnie opcjonalnego związku 1:N

Na przykład w bibliotece czytelnik (encja A) może wypożyczyć jednorazowo zero lub kilka książek, książka (encja B) może być wypożyczona tylko przez jednego czytelnika lub może aktualnie nie być wypożyczona. Gdybyśmy zastosowali regułę 3. transformacji, to należałoby z encji A utworzyć tabelę TAB_A, z encji B utworzyć tabelę TAB_B i do TAB_B wstawić klucz główny z TAB_A. Ponieważ związek po obu stronach ma charakter opcjonalny, więc nie dla każdego wiersza z TAB_B musi istnieć odpowiadający mu wiersz z TAB_A. Wracając do naszego przykładu z biblioteką, może być taka sytuacja, że w TAB_B jest 1 000 000 wierszy

opisujących książki a w danej chwili tylko 5 osób wypożyczyło po jednej książce. Tylko 5 książek będzie miało wpisaną wartość identyfikującą czytelnika, pozostałe 999 995 będzie miało wartości nieokreślone. W takim przypadku korzystniej jest wprowadzić trzecią tabelę zawierającą klucze główne z obu tabel, przechowującą informacje o książkach aktualnie wypożyczonych i o osobach, które je wypożyczyły (czyli zastosować regułę 4). W podobny sposób należy zawsze rozważyć każdą zaistniałą sytuację.

W rozdziale 3.6.4 podaliśmy przykład bazy WYPOŻYCZALNIA_1 (Rys. 4a) i WYPOŻYCZALNIA_2 (Rys. 4b). Występuje tam związek obustronnie obligatoryjny

Zawiera (KASETA(1,N) : FILM(1,1); *KodFilmu*)

zachodzący między encjami KASETA i FILM. W Tabeli 10 związek tego typu występuje w grupie związków zachodzących rzadko. Pozornie może się wydawać, że znaleźliśmy dobry przykład ilustrujący taki związek – najpierw wpisujemy film, od razu podajemy numery kaset (co najmniej jedna kaset z filmem musi być ale może być ich też więcej), na których ten film się znajduje. Zakładamy, że dana kaset zawiera tylko jeden film i że nie ma kaset pustych. Obligatoryjność po stronie KASETY oznacza, że każdy film musi być powiązany chociaż z jedną kasetą. Gdybyśmy więc usuwali dane o kasecie, która np. uległa zniszczeniu lub zgubieniu, to musielibyśmy jeszcze kontrolować, czy była to już ostatnia kaset z tym filmem. Jeśli tak, to trzeba wówczas, usuwając dane o kasecie, usunąć również dane o filmie. W praktyce na ogół nie usuwa się wtedy danych (jeśli dokupimy kasetę z takim filmem, to ponownie będziemy musieli wprowadzać opis filmu). Dlatego taki związek jest teoretycznie możliwy ale po głębszej analizie lepiej byłoby zastąpić go związkiem często występującym

Zawiera (KASETA(0,N) : FILM(1,1); *KodFilmu*)

Przy tak określonym związku przy usuwaniu danych kasety z określonym filmem nie musimy kontrolować, czy istnieje jeszcze jakaś kaset z danym filmem ponieważ zakładamy, że może istnieć film nie powiązany z żadną kasetą. Można więc przechowywać opisy filmów nie nagranych na kasetach.

4.1.2. DEFINICJA RELACJI (TABEL)

Utworzone według powyższych zasad relacje (tabele) opisujemy według następującego schematu:

REL/00x **Nazwa relacji/NAZWA ENCJI**
lub

REL/00x **Nazwa relacji/Nazwa związku, NAZWY ENCJI**
gdzie *x* jest kolejnym numerem relacji.

Jeśli tabela powstała z encji, to po nazwie relacji podajemy nazwę encji, z której tworzymy tabelę, np.

REL/001 Klienci/KLIENT

Jeśli tabelę tworzymy ze związku i encji, to po nazwie tabeli podajemy nazwę związku i nazwy encji uczestniczących w transformowanym związku, np.

REL/002 Wypożyczenia/Wypożyczza, KLIENT, KASETA

Po utworzenie niezbędnych relacji podajemy ich opisy, które można przedstawić w dowolnej czytelnej postaci, np. w postaci tabel, uwzględniając elementy istotne dla tworzonej bazy danych, np.

- Atrybuty relacji (nazwy kolumn tabeli)
- Dziedziny atrybutów
- Maskę do wprowadzania danych
- Obligatoryjność/opcjonalność atrybutu
- Wartość domyślną atrybutu
- Klucz relacji
 - PK - klucz główny – *ang. primary key*
 - FK - klucz obcy – *ang. foreign key*
- Referencje - nazwy relacji, z których pochodzą klucze obce
- Źródło danych

Wszelkie używane w opisach oznaczenia powinny być wyjaśnione i konsekwentnie stosowane.

Po opisie relacji powinniśmy podać opis poszczególnych atrybutów i, jeżeli zachodzi taka potrzeba, wymienić dodatkowe warunki (ograniczenia) nałożone na poszczególne atrybuty.

Następnie należy zamieścić **przykładowe dane**, jakie mogą wystąpić w danej relacji (tabeli), co znacznie ułatwia analizowanie struktury bazy

4. PROJEKTOWANIE LOGICZNE

danych. Dane przykładowe powinny być starannie dobrane tak, by ilustrowały sytuacje typowe, wyjątkowe, ograniczenia itp. Umieszczanie np. przykładowych danych osobowych, w których wszystkie dane są zupełnie różne (czyli nie ma osób np. o takim samym nazwisku, nazwisku i imieniu, dacie urodzenia) nie jest właściwe.

Opisy relacji można ułożyć alfabetycznie według nazw relacji, co ułatwi ich odszukiwanie.

Przykład opisu relacji w modelu logicznym:

REL/001 Klienci/KLIENT

Tabela 11

Struktura tabeli **Klienci**

Nazwa atrybutu	Dziedzina	Maska	OBL + OPC -	Wartość domyślna	Ograniczenia	Klucz	Referencje	Źródło danych
NrKlienta	Int+		+			PK		SZBD
Nazwisko	String[15]		+					USER
Imię	String[15]		+					USER
DataUr	Date	____-__-__	-	1900-01-01	> {1899-01-01}, < DATE()			USER
NrMiasta	Int+		+			FK	Miasta	BD

Znaczenie atrybutów w tabeli **Klienci** przedstawia Tabela 12.

Tabela 12

Znaczenie atrybutów w tabeli **Klienci**

Nazwa atrybutu	Znaczenie
NrKlienta	Unikalny identyfikator klienta, automatycznie nadawany przez system, klucz główny
Nazwisko	Nazwisko klienta, dowolny ciąg znaków, dozwolone litery polskie
Imię	Pierwsze imię klienta, dowolny ciąg znaków, dozwolone litery polskie
DataUr	Data urodzenia klienta
NrMiasta	Identyfikator miasta, klucz obcy z tabeli Miasta

Przykładowe dane relacji **Klienci** zawiera Tabela 13.

Tabela 13

Przykładowe dane relacji **Klienci**

NrKlienta	Nazwisko	Imię	DataUr	NrMiasta
1	Koperski	Jan	1980-12-21	1
2	Walicki	Łukasz	1979-11-11	1
3	Barycka	Beata	1977-03-18	2
4	Matacka	Maria		5
5	Barycka	Beata	1981-03-18	2
8	Koperski	Jan	1980-12-21	1

□

Aby ułatwić wyszukiwanie informacji o wybranym atrybucie sporządzamy **słownik atrybutów**. Słownikiem (*ang. dictionary*) ogólnie nazywa się miejsce przechowywania terminów i ich objaśnień.

Słownik atrybutów (*ang. data dictionary*) tworzy się w celu zestawienia nazw kolumn i domen (dziedzin), określających typ danych, jakie mogą być przechowywane w danej kolumnie oraz ograniczenia, którym podlegają. Jest to pewnego rodzaju baza danych projektanta bazy danych.

Słownik danych można przedstawić np. w postaci 3-kolumnowej tabeli, w której zostaną wyszczególnione nazwy kolumn poszczególnych tabel (relacji), dziedziny oraz przynależność do relacji. Nazwy kolumn powinny być uporządkowane alfabetycznie. Na bazie tego samego typu można zdefiniować wiele domen poprzez określenie ich cech indywidualnych takich jak np. maksymalna szerokość kolumny, liczba miejsc po przecinku, wzorzec (maska) wprowadzania danych, wartości domyślne, ograniczenie zakresu dozwolonych wartości, czy dopuszczalne są wartości NULL itp. Można określić np. domenę Nazwisko, która będzie oznaczała dziedzinę dla atrybutu NazwiskoKlienta jako łańcuch co najwyżej 25 znakowy, zawierający litery, myślnik, apostrof, spację.

Przykład słownika atrybutów:

Tabela 14

Słownik atrybutów

Nazwa atrybutu	Dziedzina	Przynależność do tabeli
Cena	Logical	Towary
DataSprzedaży	Date	Faktury
DataUr	Date	Klienci
IdMiasta	Int+	Miasta
IdTowaru	Int+	Towary DetaleFaktur
Ilość	Int	DetaleFaktur
Jednostka	String[10]	DetaleFaktur
Imię	String[15]	Klienci
NazwaMiasta	String[15]	Miasta
NazwaTowaru	String[25]	Towary
NazwiskoKlienta	Nazwisko	Klienci
NrKlienta	Int+	Klienci Faktury
NrMiasta	Int+	Klienci
Platność	String[10]	Faktury
Uwagi	Memo	Faktury

□

4.1.3. SCHEMAT RELACYJNEJ BAZY DANYCH

Zdefiniowane w poprzednim rozdziale relacje (tabele) relacyjnej bazy danych przedstawiamy w postaci schematów relacji. Zgodnie z definicją podaną w rozdziale 2.1, **schemat relacji** (*ang. relation schema*) to nazwa relacji oraz zbiór atrybutów relacji.

Schemat bazy danych (*ang. database schema*) jest konstrukcją logiczną opisującą strukturę danych w pojęciach abstrakcyjnych. Obejmuje opis relacji (tabel), które będą przechowywały dane (i które będą ewentualnie zaimplementowane w przyszłym systemie bazy danych). Według definicji [Subieta1999] 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.”

Postać ogólna schematu bazy danych jest następująca:

NAZWA BAZY DANYCH

Nazwa_relacji_1(*lista atrybutów relacji_1*)

Nazwa_relacji_2(*lista atrybutów relacji_2*)

....

Schemat bazy danych jest więc zbiorem schematów relacji (w praktyce będących najczęściej w trzeciej postaci normalnej).

W schematach relacji podkreślamy atrybuty, które są kluczami głównymi a przed atrybutami, które są kluczami obcymi, wpisujemy symbol #.

Przykład schematu relacyjnej bazy danych:

Po transformacji modelu konceptualnego bazy danych z przykładu zamieszczonego w rozdziale 3.6.4 do modelu logicznego (zgodnie z regułami podanymi w rozdziale 4.1.1), otrzymalibyśmy następujący schemat bazy danych WYPOŻYCZALNIA_2:

WYPOŻYCZALNIA_2

Klienci (NrKlienta, Nazwisko, Imię, Miasto, Ulica_nr)

Kasety (NrKasety, Opis, #KodFilmu)

Filmy (KodFilmu, Tytuł, CzasTrwania, #KodGatunku)

Gatunki (KodGatunku, Rodzaj)

Wypożyczenia (NrWypożyczenia, #NrKlienta, #NrKasety,
DataWypożyczenia, DataZwrotu, Opłata)

Jako ćwiczenie pozostawiamy Czytelnikowi sprawdzenie, w której postaci normalnej są powyższe schematy relacji.

□

4.2. DEFINIOWANIE PERSPEKTYW

Ostatnim etapem projektowania bazy danych jest określenie **perspektyw** dla jej przyszłych użytkowników. W tym celu należy najpierw wyszczególnić grupy użytkowników bazy danych oraz sprecyzować, które dane, w których transakcjach będą przez nich wykorzystywane oraz jakie operacje na tych danych będą mogli wykonywać. Następnie w sposób systematyczny należy wyodrębnić perspektywy opisać.

Perspektywa (*ang. view*) jest to nazwana tabela (wirtualna), która w odróżnieniu od tabeli bazowej (wyjściowej) nie może istnieć samodzielnie i definiowana jest za pomocą jednej lub wielu tabel wyjściowych (bazowych lub perspektyw), ich wszystkich lub niektórych atrybutów [Date2000]. W szerszym znaczeniu jest to pojęciowy obraz danych, który jest przypisany do pojedynczego użytkownika lub do grupy użytkowników i abstrahuje od danych dla nich nieistotnych. Tak rozumiana perspektywa dostosowuje sposób przedstawienia danych do przyzwyczajeń użytkownika zapewniając jednocześnie wygodę i przyjazność w przetwarzaniu danych. Niezwykle ważne jest zapewnienie przezroczystości (*ang. transparency*), aby użytkownik miał wrażenie, że operuje na danych zapamiętanych a nie wirtualnych [Subieta1999]. Perspektywy są mocnym narzędziem służącym do ograniczania dostępu do danych. Największym problemem jest aktualizacja perspektyw i w relacyjnych systemach zarządzania bazami danych na ogół możliwa jest jedynie aktualizacja perspektyw będących wynikiem operacji rzutu lub selekcji tablic bazowych (z zachowaniem klucza głównego).

W etapie definiowania perspektyw należy więc:

- wyszczególnić grupy użytkowników bazy danych,
- wyszczególnić perspektywy dla każdej grupy użytkowników,
- wyszczególnić transakcje dla każdej perspektywy;
- określić dane uczestniczące w danej transakcji; dane te można opisać podając nazwę relacji i atrybut, gdzie dana jest przechowywana oraz dopuszczalne operacje wykonywane na tej danej. Opisy można przedstawić np. w formie tabeli o następującym nagłówku:

(Nazwa relacji, Atrybut, Zapis, Odczyt, Modyfikacja)
zaznaczając symbolem + w odpowiedniej kolumnie dla odpowiedniego atrybutu możliwość wykonania danej operacji (zapis, odczyt, modyfikacja).

Przykład opisu perspektywy:

PER/001 Dane pracownika

Użytkownik: Administrator

Transakcje: TRA/001, TRA/002, TRA/005

TRA/001 - Wprowadzenie danych pracownika

Tabela 15

Tabela transakcji wprowadzenie danych pracownika

Nazwa relacji	Atrybut	Zapis	Odczyt	Modyfikacja
Osoby	Id	+		
Osoby	Nazwisko	+		
Osoby	Imię	+		
Osoby	DataUrodzenia	+		
Dzieci	ImięDziecka	+		
Dzieci	DataUrDziecka	+		

TRA/002 - Edycja danych pracownika

Tabela 16

Tabela transakcji edycja danych pracownika

Nazwa relacji	Atrybut	Zapis	Odczyt	Modyfikacja
Osoby	Id		+	
Osoby	Nazwisko		+	+
Osoby	Imię	+	+	+
Osoby	DataUrodzenia	+	+	+
Dzieci	ImięDziecka	+	+	+
Dzieci	DataUrDziecka	+	+	+

TRA/005 - Wyświetlenie danych pracownika

Tabela 17

Tabela transakcji wyświetlenie danych pracownika

Nazwa relacji	Atrybut	Zapis	Odczyt	Modyfikacja
Osoby	Nazwisko		+	
Osoby	Imię		+	
Osoby	DataUrodzenia		+	
Dzieci	ImięDziecka		+	
Dzieci	DataUrDziecka		+	



5. ZAKOŃCZENIE

W pracy został przedstawiony pełny cykl tworzenia projektu relacyjnej bazy danych. Aby wykonać poprawny projekt bazy danych należy poprawnie wykonać poszczególne etapy, dokładnie je dokumentując. Projekt możemy uznać za poprawny, jeśli jego opis jest zgodny z przyjętą notacją, która musi być na początku projektu przedstawiona i omówiona. 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 sprzecznych ze sobą
- **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 fazy projektowania bazy danych należy nieraz wykonywać wielokrotnie, poprawiać opisy, uzupełniać i uszczegóławiać. Opracowaną dokumentację należy analizować wielokrotnie zwracając uwagę na cztery wymienionej wyżej cechy (kompletność, niesprzeczność, spójność i zgodność).

Do projektowania baz danych można wykorzystać narzędzia CASE, które automatyzują i przyspieszają proces projektowania [FugStąTr1995] a których opis wykracza poza przyjęte ramy niniejszej pracy.

W pracy zostały opisane kolejne etapy projektowania bazy danych opartego głównie na identyfikacji encji i związków. Do tworzenia schematu bazy danych można podejść również w nieco inny sposób [Pankow1992]. Na podstawie analizy wycinka rzeczywistości określamy jeden globalny (początkowy) schemat relacji $R(A_1, A_2, \dots, A_n)$, w którym uwzględniamy wszystkie atrybuty występujące w bazie danych. Bazę danych traktujemy więc jako jedną (być może wielką) relację. W związku z tym wszystkie atrybuty

należy określić jednoznacznie. Równolegle z określaniem atrybutów należy określić zależności funkcyjne między atrybutami. W kolejnym kroku normalizujemy początkowy schemat relacji uzyskując docelowy (końcowy) schemat bazy danych, który jest zbiorem schematów relacji i 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 w [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, ograniczanie się do wartości prostych jako atrybutów, prowadzi do zaprojektowania znormalizowanej bazy danych.

Dysponując poprawnym projektem bazy danych można przystąpić do opracowania projektu aplikacji bazodanowej a następnie do implementacji systemu informatycznego [Banach1998, CasPal2000, Górski2000, Mazur1997, Riordan2000, Roman2001, W&W1997].

6. ZESTAWIENIE ETAPÓW PROJEKTOWANIA BAZY DANYCH

Model konceptualny

- Etap 1.** Zdefiniowanie celu i założeń wstępnych.
- Etap 2.** Analiza rzeczywistości (wywiad z ekspertem dziedzinowym).
- Etap 3.** Analiza istniejącej bazy danych (wady, zalety).
- Etap 4.** Wyodrębnienie reguł funkcjonowania
- Etap 5.** Wyodrębnienie ograniczeń dziedzinowych.
- Etap 6.** Zdefiniowanie kategorii.
- Etap 7.** Zdefiniowanie transakcji.
- Etap 8.** Identyfikacja encji.
- Etap 9.** Identyfikacja związków.
- Etap 10.** Definicje predykatowe encji i związków.
- Etap 11.** Diagram związków encji.

Model logiczny

- Etap 12.** Transformacja modelu konceptualnego do modelu logicznego
- Etap 13.** Zdefiniowanie relacji.
- Etap 14.** Utworzenie słownika atrybutów.
- Etap 15.** Opracowanie schematu bazy danych.
- Etap 16.** Określenie użytkowników i zdefiniowanie perspektyw.

7. LITERATURA

- [Banach1998] Banachowski Lech, *Bazy danych. Tworzenie aplikacji*, Akademicka Oficyna Wydawnicza PLJ, Warszawa, 1998
- [Barker1996] Barker Richard, *CASE*Method, Modelowanie związków encji*, WNT, Warszawa, 1996
- [BarLong1996] Barker Richard, Longman Cliff, *CASE*Method, Modelowanie funkcji i procesów*, WNT, Warszawa, 1996
- [Beynon1999] Beynon-Davies Paul, *Inżynieria systemów informacyjnych*, WNT, Warszawa, 1999
- [Beynon2000] Beynon-Davies Paul, *Systemy baz danych*, WNT, Warszawa, 2000
- [Bora1995] Boratyn Dariusz, *MS ACCESS 2.0*, Croma&Msoft&DK, Wrocław, 1995
- [Brooks2000] Brooks Frederick P. Jr., *Mityczny osobomiesiąc. Eseje o inżynierii oprogramowania*, WNT, Warszawa, 2000
- [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*, CACM 13, 1970
- [Date1981] Date C. J., *Wprowadzenie do baz danych*, WNT, Warszawa, 1981
- [DateDar2000] Date C. J., Darwen Hugh, *SQL. Omówienie standardu języka*, WNT, Warszawa, 2000
- [Date2000] Date C. J., *Wprowadzenie do systemów baz danych*, WNT, Warszawa, 2000

7. LITERATURA

- [DelAdi1989] Delobel Claude, Adiba Michel, *Relacyjne bazy danych*, WNT, Warszawa, 1989
- [FugStąTr1995] Fuglewicz Piotr, Stapor Katarzyna, Trojnar Andrzej, *CASE dla ludzi*, LUPUS, Warszawa, 1995
- [Górski2000] Praca zbiorowa pod redakcją Janusz 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
- [Hend1998] Henderson Ken, *Bazy danych w architekturze klient/serwer*, ROBOMATIC, Wrocław, 1998
- [Hernan1998] Hernandez Michael J. *Bazy danych dla zwykłych śmiertelników*, MIKOM, Warszawa, 1998
- [Jakub1998] Jakubowski Arkadiusz, *Delphi3. Tworzenie komponentów*, Lynx-SFT, Warszawa, 1999
- [Jakub1999] Jakubowski Arkadiusz, *Delphi4. Księga eksperta*, Helion, Gliwice, 1999
- [Jaszk1997] Jaskiewicz Andrzej, *Inżynieria oprogramowania*, Helion, Gliwice, 1997
- [Kierklo1994] Kierklo Jacek, *Język SQL w dBASE IV*, MIKOM, Warszawa, 1994
- [Kułak1995] Kułakowski Jacek, *Delphi. Pierwsze kroki*, MIKOM, Warszawa, 1995
- [MataCus2000] Mata-Toledo Ramon A., Cushman Pauline K., *Fundamentals of Relational Databases*, Schaum's Outline Series, USA, 2000
- [Mazur1997] Mazur Hanna, Mazur Zygmunt, *Programowanie w dBASE IV*, PSZI, Wrocław, 1997
- [Muller2000] Muller Robert J., *Bazy danych. Język UML w modelowaniu danych*, MIKOM, Warszawa, 2000
- [MurRyb1993] Muraskiewicz 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
- [Subieta1998] Subieta Kazimierz, *Obiektość w projektowaniu i bazach danych*, Akademicka Oficyna Wydawnicza PLJ, Warszawa, 1998
- [Subieta1999] Subieta Kazimierz, *Słownik terminów z zakresu obiektości*, Akademicka Oficyna Wydawnicza PLJ, Warszawa, 1999
- [TsiLoch1990] Tsichritzis Dionysios C., Lochovsky Frederick H., *Modele danych*, WNT, Warszawa, 1990
- [Turski1971] Turski Władysław M., *Struktury danych*, WNT, Warszawa, 1971
- [UllWid2000] Ullman Jeffrey D., Widom Jennifer, *Podstawowy wykład z systemów baz danych*, WNT, Warszawa, 2000
- [W&W1997] Strużińska-Walczak Anna, Walczak Krzysztof, *Delphi. Nauka programowania systemów baz danych*, Wydawnictwo W&W, Warszawa, 1997
- [Wremb1997] Wrembel Robert, Wiczyrzecki Waldemar, *Projektowanie aplikacji bazy danych Oracle*, Nakom, Poznań, 1997
- [Yourdon2000] Yourdon Edward, *Marsz ku kłębce*, WNT, Warszawa, 2000
- [Yourdon1996] Yourdon Edward, *Współczesna analiza strukturalna*, WNT, Warszawa, 1996

DODATEK

Niniejsza praca omawia projektowanie bazy danych. Termin „baza danych” jest więc podstawowym pojęciem spotykanym 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 adresowych, telefonicznych, kont bankowych, szkół i uczelni, bezrobotnych, dla Zakładu Ubezpieczeń Społecznych itd. Intuicyjnie każdy rozumie co to jest baza danych lecz w literaturze brak jest jednoznacznej definicji pojęcia bazy danych. Poniżej przytaczamy kilka określeń bazy danych zaczerpniętych z prac dostępnych na polskim rynku. Cytaty przytoczono w kolejności chronologicznej lat, biorąc pod uwagę rok wydania książki, z której pochodzi cytat.

Cytat 1 [OleńStan1984, str. 11]

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

Cytat 2 [Cell1988, str. 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, str. 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, str. 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, str.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, str. 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, str. 225]

„**Baza danych** – dowolny zbiór tabel lub plików, będący pod kontrolą systemu zarządzania bazą danych.”

Cytat 8 [Banach1998, str. 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 nierozzerwalnie 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 odpowiedni zmieniać. Zapewnienie zgodności z rzeczywistością jest podstawowym procesem we współczesnych systemach informacyjnych korzystających z baz danych.”

Cytat 9 [Subieta1999, str. 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, str. 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, str. 31]

„**Baza danych** jest to zbiór danych trwałych, które są wykorzystywane przez system aplikacji danej organizacji (lub przedsiębiorstwa).”

Cytat 12 [UllWid2000, str. 19]

„Baza danych nie jest w istocie rzeczą niczym więcej niż zbiorem danych istniejącym przez długi czas, często przez wiele lat. W potocznym rozumieniu terminu 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 13 [Harrin2001, str. 253]

„**Baza danych** – kolekcja obiektów zarządzanych w taki sposób, że mogą być dostępne dla wielu użytkowników”.

W niniejszej pracy bazę danych zdefiniowano jako trwały zbiór tematycznie ze sobą powiązanych danych [rozdz. 2, str. 11].





BIBLIOTEKA GŁÓWNA

3095 14 L/1