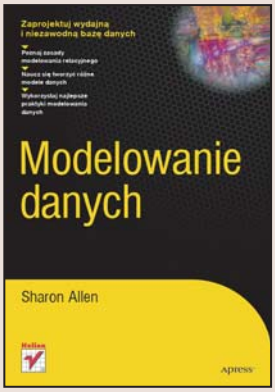
Modelowanie danych

Autor: Sharon Allen
Tłumaczenie: Bartłomiej Garbacz, Tomasz Walczak
ISBN: 83-246-0184-8
Tytuł oryginału: Data Modeling for Everyone
Format: B5, stron: 578



Modelowanie danych to umiejętność stosunkowo rzadko poszukiwana na rynku. Firmy chętniej zatrudniają programistów i administratorów baz danych. Jednak zaprojektowanie efektywnych mechanizmów przechowywania danych ma duże znaczenie przy tworzeniu korporacyjnych aplikacji bazodanowych. Dopiero w sytuacji, gdy systemy zaczynają działać niewłaściwie, okazuje się, że przyczyną jest niepoprawny projekt bazy danych. Odpowiednio przeprowadzony proces modelowania danych może ułatwić rozwiązywanie problemów z aplikacją.

Książka „Modelowanie danych" to wyczerpujące omówienie tego procesu i niezbędne źródło wiedzy dla każdego projektanta baz danych, który chce opracować wydajny i niezawodny system. Przedstawia modelowanie oparte na modelu relacyjnym, jego matematyczne podstawy i praktyczne wdrożenia. Czytając tę książkę, poznasz różne typy modeli. Dowiesz się, jaki poziom szczegółowości reprezentuje każdy z nich i jak je zaimplementować w konkretnych projektach informatycznych.

W książce omówiono także:

- Cykl istnienia danych
- Podstawowe pojęcia modelowania relacyjnego
- Reguły Codda dotyczące relacyjnych baz danych
- Normalizacja danych
- Analiza logiczna i fizyczna
- Modelowanie procesów biznesowych
- Tworzenie modelu logicznego
- Przekształcanie modelu logicznego w fizyczny
- Stosowanie metadanych
- Praktyki modelowania danych

Dzięki wiadomościom z tej książki staniesz się specjalistą w zakresie modelowania danych.

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Rozdział 2.

Modelowanie relacyjne

Aby móc tworzyć relacyjne modele danych i opracowywać prawdziwie relacyjne bazy danych, musimy zrozumieć, co składa się na taką bazę, opanować terminologię związaną z opisywaniem systemów relacyjnych oraz mieć świadomość istnienia różnych zestawów symboli umożliwiających reprezentowanie systemów relacyjnych w formacie graficznym. Dlatego też w niniejszym rozdziale zostaną poruszone następujące kwestie:

- Natura relacyjnych baz danych oraz istniejące różnice między nimi a hierarchicznymi oraz sieciowymi **systemami zarządzania bazami danych** (ang. *Database Management System — DBMS*).
- Terminologia dotycząca relacyjnych baz danych:
  - encje,
  - atrybuty,
  - klucze (kandydujące, główne i obce),
  - związki,
  - tabele i perspektywy,
  - kolumny,
  - klucze (główne, sztuczne, alternatywne, obce),
  - więzy.
- Składnia modelowania IDEF1X oraz inne notacje dostępne w arsenale modelarza, a używane w celu dokumentowania elementów danych oraz reguł biznesowych.

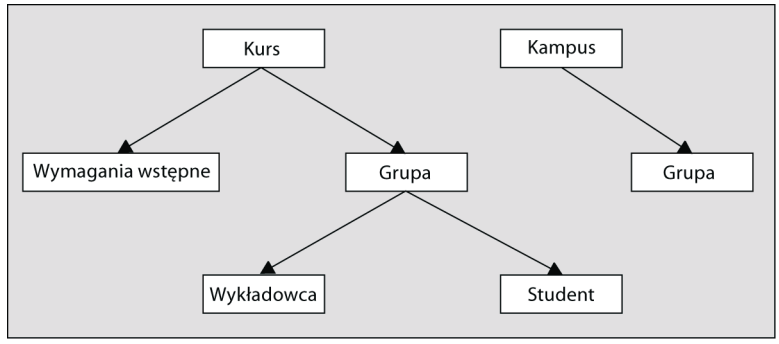
Modele baz danych

Zanim przyjrzymy się bardziej szczegółowo bazom danych, najpierw pobieżnie omówimy hierarchiczne oraz sieciowe systemy zarządzania bazami danych, gdyż poznanie wiążących się z nimi ograniczeń pomoże nam w zrozumieniu, rozwiązaniem jakich problemów jest podejście relacyjne.

Model hierarchiczny

Jednym z wciąż wykorzystywanych hierarchicznych systemów DBMS jest produkt firmy IBM o nazwie **IMS** (od ang. *Information Management System*). Zasadniczo stanowi on strukturę drzewiastą wykorzystującą serię łączy w celu zapewnienia możliwości nawigacji od jednego do drugiego typu rekordu (tabeli). Rekordy (tabele) zawierają jedno lub większą liczbę pól (kolumn). Każde drzewo musi posiadać pojedynczy **główny** (ang. *root*) typ rekordu. Jako przykład takiego systemu rozważmy poniższy system planowania zajęć używany w college'u przedstawiony na rysunku 2.1.

Rysunek 2.1.



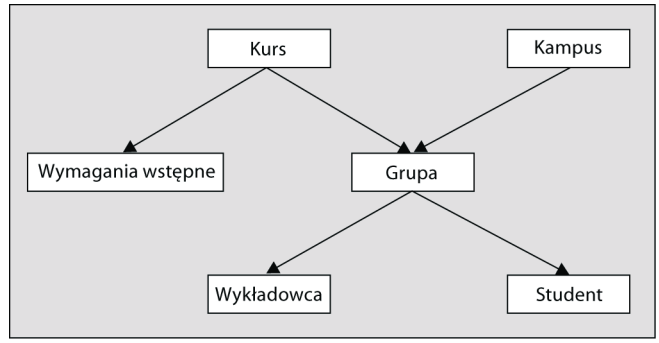
Problem dotyczący tego rodzaju systemów jest związany z wierzchołkami potomnymi posiadającymi więcej niż jednego rodzica (na przykład `Grupa` w powyższym przykładzie). Wiele rzeczywistych elementów danych charakteryzuje się właśnie taką cechą, ale niestety, hierarchiczny system DBMS nie potrafi obsłużyć tej sytuacji w wydajny sposób, gdyż dwa egzemplarze rekordu `Grupa` nie są ze sobą w prosty sposób powiązane. Rozwiązania umożliwiające obejście tego problemu zwykle wymagają tworzenia zduplikowanych rekordów lub tabel w celu spełnienia różnych wymagań, a to prowadzi do powstawania problemów z synchronizacją danych, ponieważ te same rekordy pojawiają się w wielu miejscach w bazie danych.

Model sieciowy

Jednymi z dużych baz sieciowych wciąż przetwarzających ogromne ilości danych są systemy **IDMS** (od ang. *Integrated Database Management Systems*). Sieciowe bazy danych rozwiązują problem wielu rodziców występujących w przypadku baz hierarchicznych. Jeżeli raz jeszcze posłużymy się przykładem bazy danych college'u, to w przypadku bazy sieciowej tabela `Kampus` będzie bezpośrednio połączona z tabelą `Grupa` poprzez łącze właściciel-składowa, co przedstawiono na rysunku 2.2.

Niestety, w celu przejścia z tabeli `Wykładowca` do tabeli `Kampus` musimy wrócić do tabeli `Grupa`. Można sobie z łatwością wyobrazić, że w przypadku dużej bazy danych liczącej wiele tabel może się to okazać bardzo długą ścieżką. Ponadto ścieżki te nie są proste w modyfikacji ani też nie jest prostą rzeczą dodawanie nowych związków, kiedy baza danych zostanie już utworzona, co sprawia, że ma ona mało elastyczny charakter.

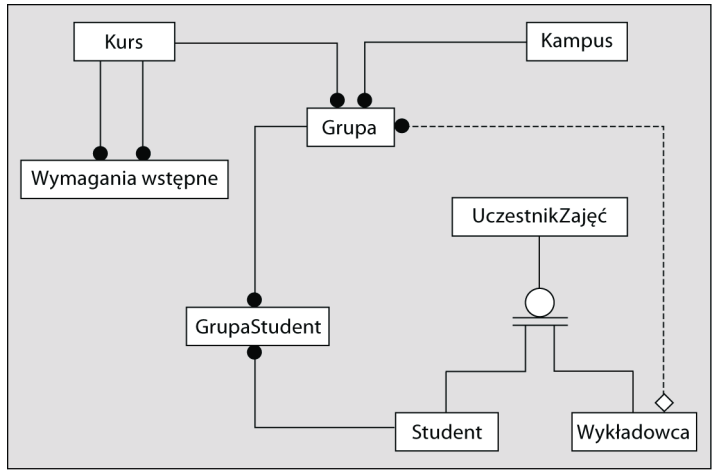
Rysunek 2.2.



Model relacyjny

Relacyjne bazy danych, takie jak Oracle, Microsoft SQL Server czy IBM DB2, stanowią rozwinięcie zarówno modelu sieciowego, jak i hierarchicznego poprzez umożliwienie dopasowywania pól danych na poziomie bazy danych. Wkrótce przyjrzymy się bardziej szczegółowo, w jaki sposób jest to osiągane, ale na razie omówimy model relacyjnej bazy danych przedstawionego już przykładu (rysunek 2.3).

Rysunek 2.3.



Jeżeli Czytelnik nie rozumie znaczenia powyższych symboli, nie powinien się tym na razie martwić — wkrótce wszystko zostanie wyjaśnione.

Powyższy model pozwala, aby tabela `Grupa` posiadała trzy związki rodzicielskie, co nie jest możliwe w przypadku hierarchicznych baz danych. Zapewnia to, że będzie występował tylko jeden zbiór o nazwie `Kurs`, który może być związany z samym sobą w przypadku, gdy wymagania wstępne pewnego kursu określają, że trzeba mieć ukończony inny kurs. W przypadku sieciowej bazy danych nie jest możliwe, aby występował taki rekord, który byłby zarówno swoim rodzicem, jak i potomkiem. Podobnie wykładowcy i studenci są rozpoznawani jako `Osoby`, co pozwala, aby Isaac Asimov był zarówno

Studentem, jak i Wykładowcą Grupy. Projekt relacyjny kładzie nacisk na zachowanie tylko pojedynczego rekordu dla podobnych informacji. Uzyskiwanie dostępu do rekordów także może być łatwiejsze w przypadku relacyjnej bazy danych. W omawianym przykładzie można natychmiast połączyć tabelę GrupaStudent z tabelą Kampus bez konieczności ich bezpośredniego łączenia.

Pojęcia z zakresu modelowania koncepcyjnego i logicznego

Modelowanie koncepcyjne i logiczne jest wykorzystywane w celu określania reguł danych w ramach działań biznesowych. Nacisk kładzie się na sposoby wykorzystania danych w środowisku biznesowym. Nie ma znaczenia, że bieżące narzędzie dopuszcza tylko jeden adres poczty elektronicznej, jeżeli przechowuje się wiele takich adresów producentów. Nie ma znaczenia, czy bieżący formularz nie posiada miejsca na adres witryny internetowej, jeżeli taka informacja okaże się potrzebna. Modelowanie koncepcyjne i logiczne skupia się na dokumentowaniu tego, co naprawdę dzieje się z danymi, jak są używane oraz do czego mają służyć w przyszłości.

Na podstawie takiej analizy można wówczas utworzyć narzędzia, które zapewniają łatwość i poprawność użycia danych w ramach danego rozwiązania biznesowego. Projekt bazy danych będzie posiadał reguły wbudowane w faktyczną strukturę, która promuje jakość, łatwość zarządzania danymi oraz wzrost. Zdobycie wiedzy o tym, jakie reguły rządzą danymi w określonym biznesie, stanowi pierwszy krok w zakresie zaprojektowania odpowiedniej struktury tabel w bazie danych.

Encje

Encje stanowią podstawę relacyjnego modelowania danych. Branżowa definicja encji brzmi następująco:

> Encja jest osobą, miejscem, rzeczą lub pojęciem, które posiada cechy interesujące z punktu widzenia przedsiębiorstwa i o którym chce się przechowywać informacje.

Należy zauważyć, że encje nie są tabelami. Często są one na poziomie fizycznym implementowane w postaci tabel i na diagramach modeli danych obrazuje się je w sposób podobny do tabel, jednak **nie** są one tabelami. Tabele są zawsze implementowane fizycznie, natomiast niektóre encje mają zbyt koncepcyjny charakter, aby mogły stać się faktycznymi tabelami.

Weźmy pod uwagę typowy dom. Na obszarach nawiedzanych trzęsieniami ziemi mieszkańcy od czasu do czasu zestawiają na ziemię różne przedmioty domowego użytku (książki, płyty CD, butelki wina, ołówki i tak dalej), kiedy ziemia zaczyna się bardziej trząść. Wszystko, co było zorganizowane, staje się stertą przedmiotów porozrzucanych

na podłodze. Kiedy stanie się oko w oko z takim bałaganem, naturalnym odruchem jest chęć poukładania wszystkiego: płyty CD są ustawiane w jednym rogu, książki w drugim i tak dalej. Modelowanie danych zapewnia organizację oddzielnych elementów danych w podobny sposób, grupując razem elementy, które reprezentują osoby, miejsca, rzeczy lub pojęcia. Na szczęście zwykle nie znajdują się one oryginalnie w stanie zupełnego chaosu, ale też nie posiadają mechanizmów pozwalających na ich odpowiednie ułożenie. Musimy określić wymagania systemu, którym się zajmujemy, i utworzyć od podstaw odpowiednie struktury składowania danych.

W przypadku sprzątania domu rozpoczynamy zwykle od szybkiego ułożenia drobnych przedmiotów, zwykle grupując elementy o podobnym charakterze. Wszystkie multimedia zostają umieszczone w jednym rogu, naczynia kuchenne w drugim, zaś potłuczone resztki wędrują do kosza. W przypadku modelowania danych stanowi to analogię do fazy **analizy koncepcyjnej**. Posiadamy elementy, które zamierzamy zachować na bardzo ogólnym poziomie szczegółowości, wciąż niezbadane fragmenty (**encje koncepcyjne**) oraz pewną liczbę elementów wykraczających poza zakres naszego zainteresowania. Po zapewnieniu organizacji na takim poziomie koncepcyjnym przechodzimy do bardziej szczegółowego schematu organizacji. Tym razem multimedia zostają posortowane w ramach bardziej szczegółowych grup, takich jak muzyka, filmy i gry. W kontekście modelowania danych oznacza to zidentyfikowanie encji koncepcyjnej `Multimedia`, reprezentującej ten zbiór przedmiotów, i pogłębienie swojego zrozumienia istoty problemu, określając encje logiczne w postaci `Muzyka`, `Film` oraz `Gra`. Każda z nich jest oddzielnym zbiorem o innych istotnych atrybutach. Dane o kompozytorze lub artyście są istotne pod względem sposobu składowania i pobierania muzyki, jednak w przypadku filmów największe znaczenie ma tematyka i tytuł.

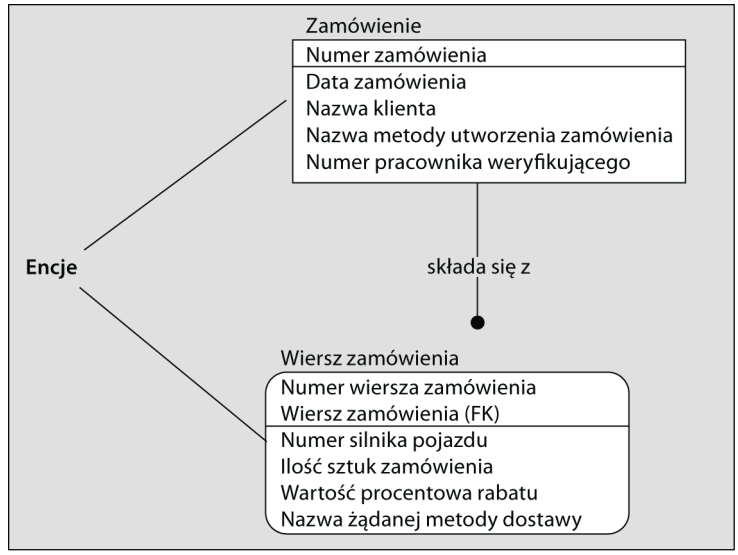
Modelowanie danych rozpoczyna się od zorganizowania, w ramach abstrakcji koncepcyjnej, zbiorów odpowiednich danych. Poprzez użycie pojęcia określanego mianem normalizacji, któremu przyjrzymy się bardziej szczegółowo w kolejnym rozdziale, oczyszczamy i rozdzielamy pod względem logicznym dane do postaci odrębnych, możliwych do ponownego użycia, zbiorów elementów, które mogą być wiązane i łączone w różnych celach. Wreszcie projektujemy fizyczne struktury składowania (tabele) w celu przechowywania danych i odpowiedniego zarządzania nimi.

Ogólnie rzecz biorąc, encje postrzega się jako zbiory klas encji reprezentujące osoby, miejsca, rzeczy, zdarzenia lub pojęcia. Dobrym pomysłem jest użycie tej listy jako formy sprawdzianu, kiedy bada się grupę elementów pod względem potencjalnego uznania ich za encje. Tak więc, jeżeli encję będziemy postrzegali jako zbiór, wówczas obie struktury przedstawione na rysunku 2.4 będą encjami systemu `Zamówienie`:

Analogicznie pojęcia, takie jak `Pracownik`, `Projekt`, `Dział`, `Część`, `Producent`, wszystkie można by uznać za encje.

Należy zachować szczególną ostrożność w zakresie stosowanego nazewnictwa w przypadku modelowania. `Pracownika` być może lepiej będzie nazwać `Robotnikiem` lub nawet `Osobą obsługującą` — w zależności od tego, dla jakiego klienta tworzy się model.

Rysunek 2.4.



To, z czego składa się encja, zależy od modelowanego systemu, przykładowo:

- dla hodowcy psów encją może być `Owczarek`,
- dla weterynarza encją może być `Duży pies`,
- dla sklepu ze zwierzętami encją może być `Pies`,
- dla firmy wynajmującej mieszkania encją może być `Zwierzę domowe`,
- dla stacji epidemiologicznej encją może być `Zwierzę`.

Podobnie:

- dla kancelarii prawniczej `Porada` może być usługą fakturowaną lub encją,
- dla komisu samochodowego `Porada` może być atrybutem encji `Sprzedawca`.

Sztuka polega na podejmowaniu odpowiednich decyzji odnośnie do tego, czy znalazło się encję czy **egzemplarz** (jeden z elementów zbioru) albo też oba, oraz stosowaniu odpowiedniej metody określania tego faktu na podstawie zakresu projektu oraz dziedziny problemu.

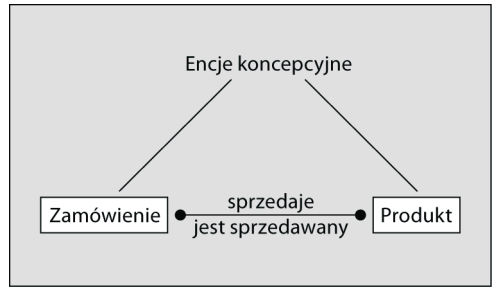
Encje koncepcyjne i logiczne

Warto pamiętać, że encje występują w dwóch głównych odmianach, które służą różnym celom. **Encje koncepcyjne** (ang. *conceptual entity*) w istocie reprezentują idee i nigdy nie podlegają fizycznemu wdrożeniu. Rzadko posiadają one jakieś dodatkowe cechy (znane jako **atrybuty** (ang. *attributes*)) i głównie są wykorzystywane w celu powiadamiania zespołów o obszarach zainteresowania oraz rozległych pojęciach, które mają być poddawane dalszej analizie w ramach projektu. Trudno wyróżnić jakiekolwiek reguły definiujące encje koncepcyjne. Jeżeli uzna się, że encja taka pomoże

w zrozumieniu istoty problemu oraz można nadać jej definicję opisową, to ogólnie rzecz biorąc, można jej użyć. Encje koncepcyjne niemal nigdy nie zostają tabelami lub innymi obiektami fizycznymi, gdyż zazwyczaj mają zbyt ogólny charakter.

Poniżej przedstawiono niewielki model zawierający encje koncepcyjne opisujące pewne przemyślenia dotyczące systemu obsługi zamówień (rysunek 2.5). Encje te znajdują się na bardzo wysokim poziomie. `Zamówienie` uwzględnia tu zapewne pojedyncze zamówienia, jak również kontrakty łańcucha dostaw. `Produkt` może być usługami, funkcjami śledzenia lub procesami pomocnymi w obsłudze zamówień. Wszystko zależy od przeprowadzonych analiz.

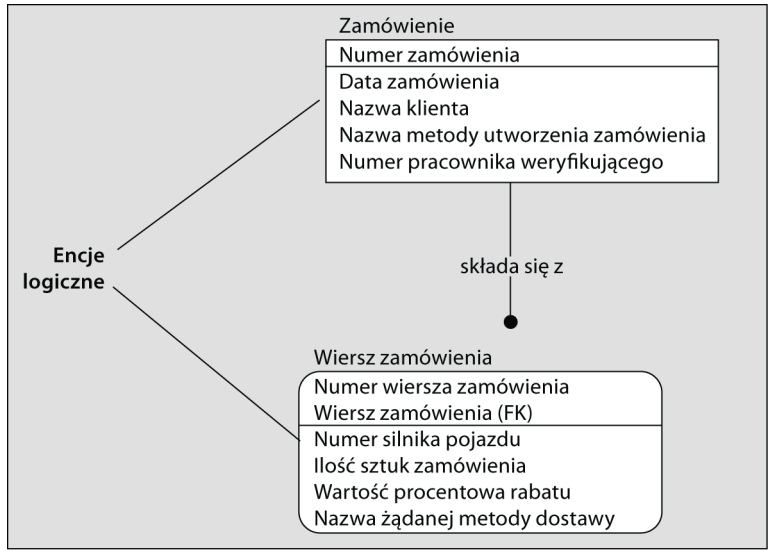
Rysunek 2.5.



Na poziomie koncepcyjnym szczegóły ulegają rozmyciu. Kiedy używa się tych struktur w ramach modelu, można nie mieć pełnego zrozumienia zakresu pojęć. Na dalszym etapie prac osoba znająca daną problematykę pomoże utworzyć i zweryfikować poprawność modelu.

Encja logiczna (ang. *logical entity*) to nieco bardziej złożony element, gdyż posiadając szczegółowy opis i dokumentację, często stanowi strukturę prowadzącą do projektu tabeli lub obiektu fizycznego. Kiedy zostaną opatrzone atrybutami, mają postać podobną do przedstawionej poniżej (rysunek 2.6):

Rysunek 2.6.



Od tego momentu, kiedy będzie mowa o encjach, będzie nam chodziło o encje logiczne.

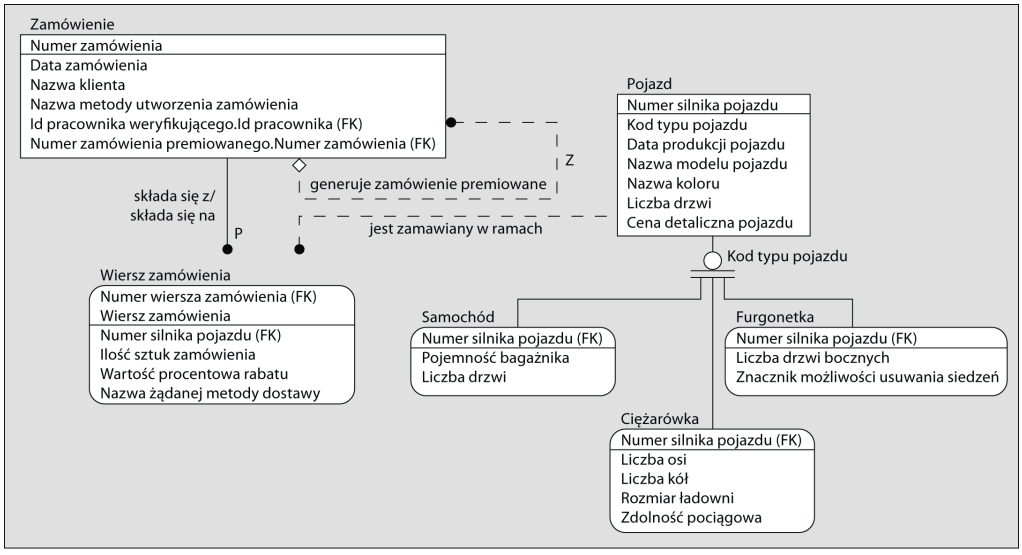
Egzemplarz czy encja?

Egzemplarz (element zbioru) zdefiniowany przez encję odpowiada poniekąd wierszowi lub rekordowi w tabeli — tak samo jak encja odpowiada tabeli — ale nie jest to jego ścisła definicja. Przykładowo, w przypadku encji `Płyty jazzowe` można wyróżnić takie egzemplarze, jak „Kind of Blue", „Porgy & Bess" lub „Time Out".

> Egzemplarz oznacza pojedynczą składową encji — element zbioru. Może to być wiersz lub rekord w tabeli, ale nie definiuje się go w ten sposób.

Encje kategorii

Struktury **kategorii**, inaczej **hierarchie uogólnień** (ang. *generalization hierarchies*), są używane w celu podziału encji na podzbiory. Każdy taki podzbiór stanowi część całości, więc na przykład `Ciężarówki`, `Samochody`, `Statki` i `Samoloty` to wszystko podzbiory encji `Pojazdy`. Na rysunku 2.7 przedstawiono schemat systemu zamówień. Jak widać, sprzedaży podlegają `Pojazdy`, które można rozbić na `Samochody`, `Ciężarówki` i `Furgonetki`.



Rysunek 2.7.

Encję, której zbiór zawiera wszystkie elementy (w omawianym przykładzie jest to encja `Pojazdy`) określa się mianem **nadtypu** (ang. *supertype*). Z kolei podziały (w omawianym przykładzie chodzi o encje `Samochód`, `Furgonetka` oraz `Ciężarówka`) określa się mianem **podtypów** (ang. *subtypes*). Wszystkie podtypy współużytkują elementy danych nadtypu, ale mogą również posiadać własne elementy danych, unikatowe tylko dla nich.

Takiej struktury modelowania (kategorii) używamy w celu dokumentowania zwartości schematu, która w przeciwnym wypadku zostałaby utracona.

Encje kategorii potomków niekiedy przypominają egzemplarze ze względu na fakt, że podziały stają się coraz bardziej szczegółowe. Czasem nawet wystąpi encja podtypu, która stanowi zbiór zawierający tylko jeden element, więc różnica między egzemplarzem a encją staje się bardzo cienka. Należy jednak pamiętać, że egzemplarz jest **elementem** zbioru, natomiast encja sama w sobie reprezentuje **ideę zbioru**.

Struktura kategorii zarządza pojedynczym zbiorem egzemplarzy, dzieląc je na typy. Ogólnie rzecz biorąc, dzieje się tak ze względu na różnice pod względem użycia lub pod względem atrybutów należących do każdego z typów.

Oczywiście `Owczarek` może być egzemplarzem encji `Pies`, natomiast `Ciężarówka`, `Samochód` i `Furgonetka` mogą być egzemplarzami encji `Pojazd`. Pojawia się zatem pytanie, skąd wiadomo, że czy chodzi o egzemplarz czy element składowy kategorii? W celu rozwiązania tego problemu należy obserwować lub odkryć „naturalne sposoby użycia" zbioru, innymi słowy zaobserwować, jak i dlaczego występują określone zachowania w ramach danego zbioru, wykorzystując wskazówki, które zostaną podane poniżej.

Definiowanie oznak występowania kategorii

Poniżej wymieniono najważniejsze wskazówki ułatwiające stwierdzenie, czy należy lub nie zapisać strukturę kategorii.

- **Co sądzi klient?** Jest to pierwsza i najlepsza metoda sprawdzenia, czy mamy do czynienia z egzemplarzem czy encją. Należy się dowiedzieć, co sądzi klient. Jeżeli zadaje się pytania dotyczące `Ciężarówki`, `Furgonetki` lub `Samochodu` i klient uważa, że dotyczą one „Pojazdu", to „Ciężarówki" są po prostu egzemplarzami zbioru `Pojazd`. Z drugiej strony, może się okazać, że „Ciężarówka" i jej zdolność pociągowa, możliwość posiadania więcej niż czterech kół oraz zdefiniowany rozmiar ładowni są wystarczająco unikatowymi cechami, że pojęcie „Ciężarówki" jest czymś odrębnym samym w sobie. Czy jest encją, której istnienie jest na tyle odrębne ze względu na jej użycie lub strukturę, że można poddać ją oddzielnej kategoryzacji wraz z przydatnymi atrybutami charakterystycznymi tylko dla niej?
- **Jak jest definiowana?** Należy zapisać definicje elementów, które potencjalnie mogłyby być encjami. Następnie należy opisać te pojęcia w formie tekstowej i dokonać ich przeglądu oraz wyszukać podobieństw. Być może znajdzie się pojęcia, które wyglądają na bardzo różne, ale w rzeczywistości oznaczają to samo lub posiadają encję uogólniającą, która uwzględnia istniejące podobieństwa, na przykład:
  - Samochód = **pojazd** kołowy, w szczególności samochód osobowy.
  - Ciężarówka = Duży, zwykle czterokołowy, **pojazd** używany w celach transportu drogowego wielkich ciężarów.
  - Furgonetka = Duży **pojazd** przykryty służący do transportu drogowego mebli i innych towarów.

Należy zwrócić uwagę na istniejące podobieństwa między wymienionymi trzema elementami. Wszystkie one posiadają w definicjach wspólny czynnik **pojazd**, posiadają koła, jeżdżą po drogach. Należy także zwrócić uwagę na występujące różnice: `Ciężarówka` jest używana w przypadku ciężkich ładunków, `Furgonetki` są przykryte, zaś `Samochody` to zwykle samochody osobowe. Jeżeli klienci podkreślają istniejące podobieństwa, prawdopodobnie mamy do czynienia z egzemplarzami. Jeżeli jednak podkreślają istniejące różnice, to zapewne mamy do czynienia z podtypami kategorii.

- **Przyjrzenie się przykładowym danym.** Te same działania wykonujemy w przypadku danych dotyczących potencjalnych encji, które klient będzie chciał przetwarzać. Należy utworzyć listę opisowych elementów danych. Jeżeli zauważy się powtarzalne wzorce atrybutów takich jak `Nazwa`, `Waga` lub `Opis`, to prawdopodobnie mamy do czynienia z kolekcją egzemplarzy, a nie encji. Nie będą one potrzebowały na tyle odrębnej struktury, aby ich utworzenie wymagało użycia różnych encji. Z drugiej strony, jeżeli tylko `Ciężarówki` wymagają prowadzenia rejestru przeglądów stanu technicznego, to posiadają one inny atrybut niż pozostałe `Pojazdy` i mogą wymagać utworzenia dla nich odrębnej kategorii.
- **Zliczanie.** Inna metoda polega na sprawdzeniu, czy istnieje więcej niż jeden egzemplarz encji kandydującej. Należy jednak pamiętać, że nie jest to sprawdzenie niezawodne. Pewne encje posiadają tylko jedną instancję w czasie dokonywania analizy, ale potencjalnie mogą później posiadać większą ich ilość. Przykładowo, encja `Firma`, która odnosi się do samego przedsiębiorstwa, może być w danym momencie pojedynczym egzemplarzem, ale w przyszłości możemy mieć do czynienia z fuzjami firm. Niekiedy nieuwzględnienie pojedynczego egzemplarza, który stanowi odrębne pojęcie, jako encji spowoduje znaczne ograniczenie elastyczności schematu w przyszłości.

Kategorie zupełne i niezupełne

Kategorie mogą być **zupełne** (ang. *complete*) lub **niezupełne** (ang. *incomplete*). Kategorie zupełne posiadają zdefiniowane wszystkie podtypy, natomiast w przypadku kategorii niezupełnych tak nie jest. Kategoria może być niezupełna, ponieważ nie są znane wszystkie kategorie, mogą się one rozszerzać wraz z upływem czasu lub zdecydowano się nie uwzględniać wszystkich. Decyzja dotycząca tego, czy należy uwzględnić wszystkie z nich, jest oparta na istotności identyfikowania podtypów z punktu widzenia klientów lub prac zespołu programistycznego. Jeżeli podtyp wymaga odmiennego traktowania, na przykład ze względów bezpieczeństwa lub o charakterze wydajnościowym, może okazać się konieczne przedstawienie ich w ramach modelu logicznego w celu umożliwienia odwzorowania takich typów wymagań.

Struktura kategorii zupełnej tworzy podkategorię, do której pasuje każdy egzemplarz z danego zbioru. Struktura kategorii niezupełna tworzy pewne, ale nie wszystkie podkategorie, do których pasują instancje.

Kategoria podrzędna może być kategorią nadrzędną dla kolejnego poziomu kategoryzacji i schemat ten może być dowolnie zagnieżdżony. Należy jedynie pamiętać, że każdy egzemplarz na każdym poziomie musi stanowić część zbioru zdefiniowanego przez najbardziej nadrzędną kategorię.

Kategorie zawierające i wykluczające

Kategorie można także podzielić na **zawierające** (ang. *inclusive*) oraz **wykluczające** (ang. *exclusive*). Zawieranie oznacza, że dowolny element nadtypu może należeć do dowolnego (lub wszystkich) z podtypów. Wykluczanie oznacza, że każdy element może należeć tylko do jednego z podtypów. Tak więc w przypadku kategorii zawierającej, gdybyśmy posiadali encję nadtypu o nazwie `Deser` podzieloną na cztery podtypy `Mrożony`, `Pieczony`, `Świeży` i `Flambé`[1], to egzemplarz encji `Deser` o nazwie `Wiśnie jubileuszowe` byłby zarówno elementem podtypu `Mrożony`, jak i `Flambé`. Z kolei w przypadku kategorii wykluczającej musielibyśmy wybrać, do którego z nich należy przypisać ten egzemplarz — prawdopodobnie byłby to podtyp `Flambé`, gdyż ogień jest tu najbardziej wyróżniającą cechą.

Encje powiązań i przecięcia

Encja **powiązania** (ang. *associative*) lub **przecięcia** (ang. *intersection*) stanowi połączenie między dwiema encjami. Ogólnie rzecz biorąc, encje takie są tworzone w celu odwzorowania związków typu wiele do wielu. W dalszej części rozdziału ten rodzaj związków zostanie omówiony bardziej szczegółowo. Na razie weźmy pod uwagę prosty przykład. Załóżmy, że w dziale sprzedaży znajdują się osoby, które muszą śledzić kwestie dotyczące zamówień. Każde zamówienie może wymagać przeprowadzenia więcej niż jednej rozmowy telefonicznej zwrotnej w celu rozwiązania problemu, jaki się pojawił. Każdy członek personelu sprzedaży może nadzorować rozwiązywanie problemów wielu zamówień. Posiadamy zbiór (encję) `Zamówień` oraz `Personelu sprzedaży`, ale także zbiór `Odebranych połączeń zwrotnych` służący do rejestrowania rozmów telefonicznych przeprowadzonych w celu rozwiązywania problemów. Zilustrowano to na rysunku 2.8.

Odpowiedni model miałby postać przedstawioną na rysunku 2.9.

Przykład encji identyfikującej

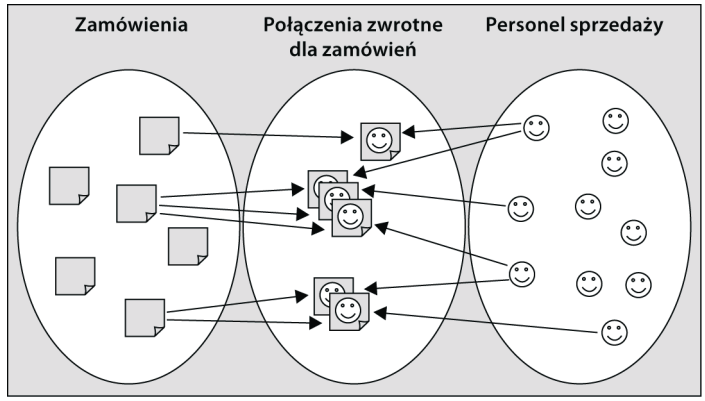
Określanie encji jest całkowicie uzależnione od okoliczności, zakresu i celu analizy. Weźmy pod uwagę przykładowy portfel i dokonajmy analizy znajdujących się w nim przedmiotów.

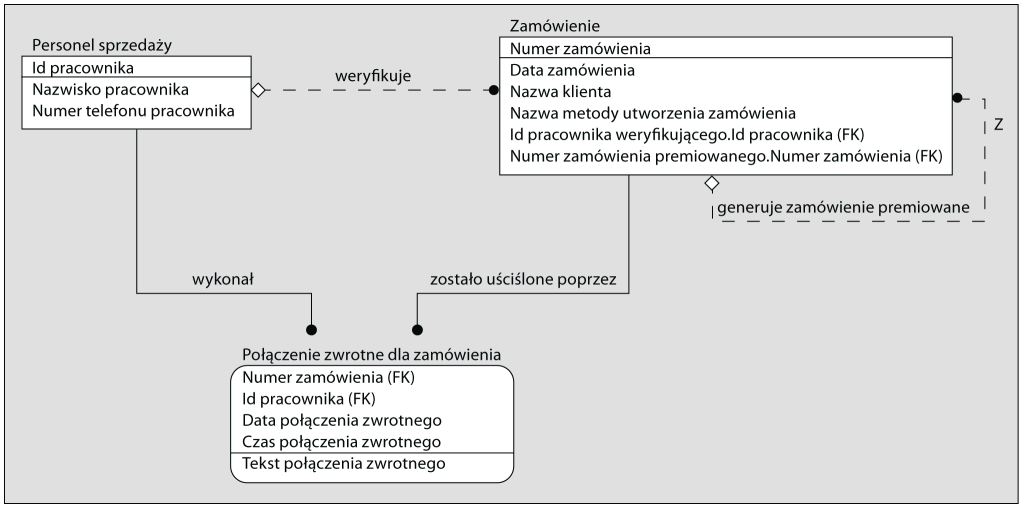
Wśród kart znajdują się:

- Dwie karty płatnicze, jedno prawo jazdy, dwie karty klubowe, jedna karta bankowa oraz cztery karty kredytowe (jedna firmowa, jedna sklepowa i dwie ogólnego przeznaczenia).

[1] Flambé — serwowany w płonącej brandy — *przyp. tłum.*

Rysunek 2.8.





Rysunek 2.9.

Wśród dokumentów znajdują się:

- Pięć rachunków, jeden anulowany czek, dwa dowody wpłaty oraz cztery potwierdzenia operacji wypłaty pieniędzy z bankomatu.

Wśród pieniędzy mamy:

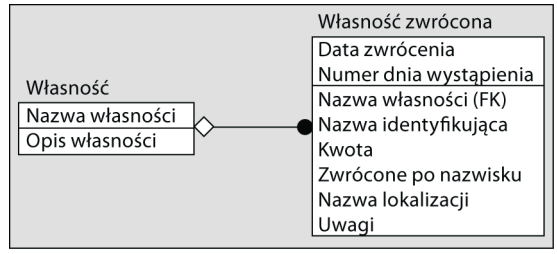
- 7 groszówek, 3 dwuzłotówki, 1 banknot pięćdziesięciozłotowy oraz 4 banknoty dziesięciozłotowe.

Niektóre z rzeczowników występujących w powyższym opisie oznaczają egzemplarze, inne encje, a dodatkowo występują tu inne encje, których nie nazwano lub jeszcze nie odkryto. Rozróżnienie między encjami a egzemplarzami będzie miało inną postać w przypadku archeologa, który odkopie taki portfel za 3000 lat, a inne w przypadku biura rzeczy znalezionych. W obu przypadkach istotne znaczenie ma fakt znalezienia portfela oraz w obu przypadkach będzie przydatne zebranie pewnych informacji o jego zawartości. Jednak w obu będzie wymagany inny poziom szczegółowości, a stąd

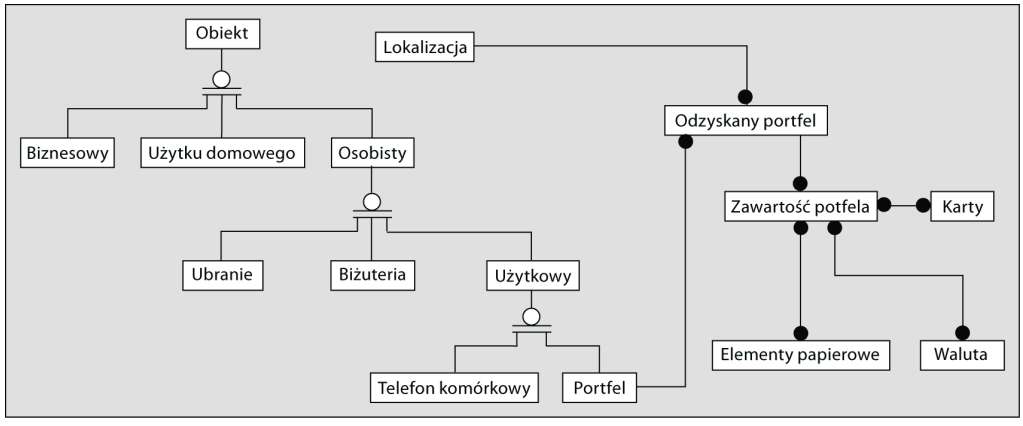
definicja encji zależy od określonych uwarunkowań. W obu przypadkach będziemy mieć do czynienia z encją odpowiadającą zdarzeniu — czemuś, co nie wystąpiło w opisie, ale ma bardzo duże znaczenie. Może to być encja zdarzenia `Własność zwrócona`.

W przypadku biura rzeczy znalezionych `Portfel` będzie zapewne egzemplarzem. Czy ma tu znaczenie fakt, że znaleziona własność jest portfelem, a nie parasolem? Tylko o tyle, że portfel będzie musiał być umieszczony w sejfie, a parasol w szafce. W przypadku takiego biura obiekty są postrzegane jako cudza własność i odpowiednio rejestruje się je w systemie. W tej sytuacji encją będzie zapewne `Własność`, zaś `Portfel` — jednym z rodzajów własności. Znalezione 96,07 zł oraz nazwisko znajdujące się na prawie jazdy mogą dla biura być wystarczającą informacją, więc `Własność` i `Własność zwrócona` to być może wszystkie informacje, jakie należy tu rejestrować. W rzeczywistości wymagania może spełnić tylko `Własność zrócona`, rejestrowana w książce rejestracji znajdującej się na biurku. Adekwatny model przedstawiono na rysunku 2.10.

Rysunek 2.10.



Z drugiej strony, dla archeologa `Portfel` może mieć istotne znaczenie i być na tyle odrębnym bytem, że będzie posiadać własną encję. Może być konieczne opisanie rozmiaru portfela, materiału, z jakiego go wykonano, oraz stylu, co sprawi, że będzie się on znacznie różnił od znajdowanych magnetowidów oraz płyt CD. Zawartość opisanego portfela może nawet nieść ze sobą wystarczającą ilość informacji, aby stać się przedmiotem czyjejś pracy doktorskiej poświęconej mieszkańcom Europy Środkowej z początków XXI wieku (rysunek 2.11).



Rysunek 2.11.

Atrybuty

Większość osób, mówiąc o danych, ma na myśli atrybuty. Występują one w postaci liczb, kodów, słów, wyrażeń, fragmentów tekstu, a nawet dźwięków lub obrazów, które są łączone do postaci egzemplarza w ramach encji. Atrybuty nie są kolumnami w tabeli, choć mogą ostatecznie być w ten sposób implementowane. Atrybuty są prostymi, oddzielnymi, odrębnymi, pojedynczymi cechami, które opisują lub identyfikują encję. Ujmując rzecz obrazowo, atrybuty są genami, kodem DNA encji. Atrybut można zdefiniować w sposób następujący:

> Atrybut jest odrębną cechą, z której powodu obsługuje się dane.

Ponownie należy dokonać rozróżnienia na świat logiczny i świat fizyczny. Atrybuty nie przechowują danych — one je opisują. Pewne atrybuty nigdy nie stają się odrębnymi kolumnami. Bardzo rzadko spotyka się oddzielnie zaimplementowane fizycznie atrybuty `Wiek`, `Rok`, `Miesiąc`, `Dzień`, `Godzina`, `Minuta` i `Sekunda`. Trzeba podkreślić, że wszystkie one są atrybutami, gdyż posiadają znaczenie różniące się od ich połączenia. Każdy oddzielny element danych może być ważny i z pewnego powodu zostać zdefiniowanym oddzielnie.

Czym więc jest atrybut? Jeżeli weźmiemy po uwagę przedstawioną wcześniej encję `Zamówienie`, to atrybutami będą wszystkie etykiety zapisane dla tej encji (`Numer zamówienia`, `Data zamówienia`, `Nazwa klienta`, `Nazwa metody utworzenia zamówienia` oraz `Numer pracownika weryfikującego`). Każdy atrybut musi być „własnością" encji. W świecie logicznego modelowania relacyjnego każdy oddzielny atrybut jest „własnością" jednej encji. Może podlegać „migracji" poprzez związki i być współużytkowany, ale jest tworzony i obsługiwany w jednym miejscu.

Atrybuty grupowe

Atrybut grupowy (ang. *group attribute*) można zdefiniować jako połączenie odrębnych atrybutów logicznych opisujących pojedynczą cechę encji. `Adres` składa się z wielu odrębnych elementów danych, podobnie jak `Numer telefonu`. Gdybyśmy każdy numer telefonu wymieniali w kontekście:

- numeru operatora,
- numeru kierunkowego kraju,
- numeru kierunkowego,
- numeru telefonu,
- rozszerzenia numeru,

musielibyśmy dodawać te atrybuty za każdym razem, gdy potrzebny nam będzie numer telefonu w modelu. Jeżeli możemy użyć definicji atrybutu grupowego `Numer telefonu`, pozwoli to zaoszczędzić czas w przyszłości. Pozwala to również zmniejszyć złożoność rozwiązania na początku, kiedy nie jest wskazane zasypywanie klienta szczegółami.

Należy w tym miejscu jednak wspomnieć, że jeśli atrybut jest częścią jednej grupy, to nie może być częścią innej. Jest on całkowicie absorbowany przez atrybut o większym rozmiarze.

Istnieje również możliwość, choć jest ona rzadziej wykorzystywana, użycia **atrybutów koncepcyjnych** (ang. *conceptual attributes*). Można je zdefiniować jako abstrakcję cech encji. Tak więc, na przykład, można dodać atrybut `Imię i nazwisko osoby` do encji `Klient` jako substytut atrybutu koncepcyjnego. Kiedy ostatecznie zostanie on w pełni opatrzony atrybutami logicznymi, może się okazać, że składa się z pięciu lub większej liczby różnych atrybutów logicznych, takich jak:

- `Kod tytułu`,
- `Imię`,
- `Drugie imię`,
- `Nazwisko`,
- `Kod kwalifikatora pokoleniowego`,

jak choćby w zapisie `dr Martin Luther King Jr`. W rzeczywistości może być konieczne użycie kilku imion lub tytułów w celu zapewnienia pełnego opisu osoby, w zależności od potrzeb klienta.

Warto w tym miejscu zauważyć, że można wygenerować model logiczny rozbijający wspólne elementy danych w celu późniejszego ich ponownego połączenia w ramach realizacji fizycznej jako `Pełne imię i nazwisko`. Jednak zachowanie rozbicia atrybutów logicznych w ramach realizacji fizycznej pozwala na zapewnienie prostego dostępu do danych, na przykład sortowania po `Nazwisku`. Połączenie wszystkiego w jedną całość znacznie utrudnia wykonywanie tego rodzaju zadań.

Jeżeli nie jest się gotowym do szczegółowego opisania atrybutu w postaci jego podstawowych składowych lub zamierza się uprościć wszystko na początkowych etapach działań, należy wykorzystać atrybuty koncepcyjne lub grupowe. Nie wszystkie programy modelowania danych mogą pomóc w zarządzaniu opcją przechodzenia między szczegółami a nazwą atrybutu grupowego. Może okazać się konieczne zarządzanie powiązaniem poza modelem.

Po omówieniu istoty atrybutów oraz ich szczegółowości zajmiemy się ich kolejną cechą — typem danych, jakie opisują.

Klasy atrybutów (typy)

Wszystkie atrybuty logiczne (koncepcyjne i grupowe nie są na tyle precyzyjne, aby wymagały tego poziomu definiowania) muszą pasować do jednego z typów danych. Jest to bardzo wysokopoziomowa klasyfikacja znaczenia danych, uwzględniająca sposoby użycia i definicję atrybutu. Różni się bardzo od dziedziny atrybutu, która określa docelowy typ danych w ramach realizacji fizycznej, na przykład `VARCHAR`, `DECIMAL` itd. Jednakże mimo to sprawdzenie wyglądu i wartości danych atrybutu jest dobrym testem poprawności analizy i definicji atrybutów. Tak więc na przykład w przypadku encji `Zamówienie` mamy do czynienia z takimi atrybutami jak `Numer zamówienia`, `Data`

`zamówienia` oraz `Nazwa klienta`. Możemy się spodziewać, że atrybut `Numer zamówienia` będzie zawierał wartość liczbową, zaś atrybut `Data zamówienia` powinien zawierać wartość daty. Podobnie atrybut `Nazwa klienta` powinien zawierać wartość tekstową.

Co dzieje się na przykład w sytuacji, kiedy wydaje się nam, że mamy do czynienia z listą kwot sprzedaży, którą ktoś utworzył na podstawie faktur, a okazuje się, że daty zostały pomylone z kwotami? Zadajemy pytania. Jest prawdopodobne, że pole użyte na formularzu było wykorzystywane w kilku celach i niesie ze sobą dwa lub więcej znaczeń. Może się okazać, że odkryliśmy nową regułę biznesową, o której nikt do tej pory nie wspomniał, lub jeden z obszarów, które można usprawnić, uściślając elementy danych. Proces biznesowy może obsługiwać niepraktyczną regułę, która określa, że jeżeli faktura jest faktycznym rachunkiem, pole zawiera kwotę fakturowaną, ale jeżeli została ona zwrócona w celach korekcyjnych, to pole to zawiera proponowaną datę zwrotu. Dziwne rzeczy się zdarzają.

Każdy atrybut musi być możliwy do zdefiniowania jako jedna i tylko jedna klasa. Jeżeli z jakiegoś powodu występuje wiele klas danego atrybutu o podobnym znaczeniu w jednym zbiorze, wówczas należy wybrać najbardziej specyficzny typ, który jest prawdziwy dla wszystkich wystąpień. Przykładowo, jeżeli scala się pola, które wydają się posiadać wiele znaczeń, wskazane może być zaklasyfikowanie nowego atrybutu po prostu jako pola tekstowego. Może się również okazać, że można uszczegółowić klasę z typu Numer do Ilość lub Wartość, ponieważ wartości danych są wystarczająco spójne, aby móc je zakwalifikować w ramach bardziej rozstrzygającej klasy.

W tabeli 2.1 przedstawiono krótką listę typów atrybutów, którą można uzupełnić we własnym zakresie:

Tabela 2.1.

Typ	Znaczenie	Dziedzina danych
Czas	Moment taki jak godzina, minuta, sekunda itd.	Czas
Data	Data kalendarzowa — miesiąc, dzień, rok.	Data
Dźwięk	Obiekt dźwiękowy.	Obiekt Blob
Identyfikator	Unikatowy znacznik rozpoznawczy w postaci znaku i (lub) cyfry, identyfikator generowany przez system lub identyfikator globalny. Może być generowany przez proces lub nie mieć żadnego znaczenia dla użytkownika danych.	Tekst
Ilość	Wartość liczbowa określająca pewną miarę w odpowiednich jednostkach.	Liczba
Kod	Alfanumeryczny skrót o zrozumiałym znaczeniu.	Tekst
Wartość	Wartość pieniężna.	Liczba
Numer	Wartość z sekwencji liczbowej.	Liczba
Nazwa	Etykieta tekstowa.	Tekst
Rysunek	Niejęzykowy obiekt wizualny.	Obiekt Blob
Opis	Tekstowe objaśnienie lub charakterystyka.	Tekst
Tekst	Niesformatowany element językowy.	Tekst
Znacznik	Zbiór logiczny pojedynczych liter lub liczb.	Tekst

Należy zwrócić uwagę, że tego rodzaju klasyfikacja nie określa dziedziny. Wielokrotnie używamy dziedziny danych dla wielu różnych klas lub typów. Taka klasyfikacja staje się bardzo ważna w przypadku standardów nazewnictwa atrybutów.

Klucze

Atrybuty pełnią również inną rolę poza opisywaniem cech encji. Są one używane w celu identyfikowania unikatowych egzemplarzy encji jako albo elementów zbioru (opisywanych jako klucze kandydujące, główne lub alternatywne), albo odwołań do elementów zbioru (klucze obce). Atrybuty te, lub grupy atrybutów, określa się mianem **kluczy** (ang. *keys*).

Klucz w modelu logicznym to jeden lub większa liczba atrybutów używanych w celu jednoznacznego zidentyfikowania egzemplarza w encji jako części tej encji albo poprzez odwołanie do innej encji.

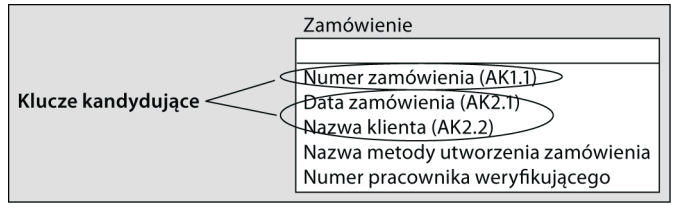
Poniżej po kolei omówimy wszystkie typy kluczy, rozpoczynając od kluczy kandydujących.

Klucze kandydujące

Klucz kandydujący (ang. *candidate key*) to jeden z wielu zbiorów pojedynczego lub wielu atrybutów używanych w celu jednoznacznego identyfikowania egzemplarzy encji.

Klucze kandydujące to **wszystkie** opcjonalne możliwości wyboru jednoznacznych sposobów identyfikowania encji (rysunek 2.12).

Rysunek 2.12.



Od kluczy kandydujących oczekuje się, że będą zawsze unikatowe, jak również stabilne i nigdy nie będą podlegać zmianom. W przedstawionym powyżej przykładzie mamy do czynienia z dwoma kluczami kandydującymi. Egzemplarze encji `Zamówienie` mogą być identyfikowane przez:

- Nadawany im unikatowy numer (`Numer zamówienia`).
- Kombinację atrybutów `Data zamówienia` oraz `Nazwa klienta` (zakładając, że klient może składać zamówienie tylko raz dziennie oraz każdy klient ma inną nazwę).

W programie Erwin 3.5.2 klucz kandydujący jest określany również mianem klucza alternatywnego (ang. *Alternate key*), skąd bierze się zapis „AK". Pierwsza liczba określa, do którego klucza alternatywnego należy dany atrybut. Druga liczba określa, którym z kolei z atrybutów tworzących klucz jest dany atrybut. Inne pakiety oprogramowania stosują inne konwencje.

Identyfikacja kluczy kandydujących daje pewne możliwości wyboru. Może się okazać, że istnieją spodziewane interfejsy, dla których lepszą metodą identyfikacji byłaby jedna zamiast drugiej. Warto w takiej sytuacji przyjrzeć się przykładowym danym i sprawdzić unikatowość oraz stabilność kluczy.

Klucze naturalne i sztuczne

Modelowanie logiczne skupia się na elementach danych, które występują w świecie rzeczywistym. Chodzi tu o liczby zapisywane na fakturach i czekach, nazwy miejsc i państw, współrzędne długości i szerokości geograficznej raf i pasów startowych lotnisk — to tylko kilka przykładów. Z reguły w ramach modelu logicznego nie tworzy się atrybutów. Klucze naturalne są tworzone z atrybutów, które istnieją w świecie rzeczywistym.

Klucz naturalny (ang. *natural key*) to jeden spośród jednego lub wielu zbiorów pojedynczych lub wielu atrybutów używanych w celu jednoznacznego identyfikowania egzemplarzy w encji, które mogą występować w świecie biznesu.

Jednak dość często klucze naturalne nie są wystarczająco stabilne lub spójne, aby móc ich użyć w projekcie fizycznym. Na szczęście platformy bazodanowe oferują możliwość automatycznego generowania wartości liczbowych i wiązania ich z rekordami tworzonymi w tabeli. Nie niosą one ze sobą żadnego znaczenia, ale zapewniają możliwość identyfikowania wierszy. Określa się je mianem **kluczy sztucznych**.

Klucz sztuczny (ang. *surrogate key*) to pojedynczy atrybut implementowany w systemach zarządzania relacyjnymi bazami danych w celu jednoznacznego identyfikowania egzemplarzy w ramach encji. Nie występuje on w sposób naturalny, który odzwierciedlałby realia biznesowe.

Prostym przykładem klucza sztucznego jest identyfikator zamówienia. Jeżeli zdecydujemy się na fizyczne zaimplementowanie takich kluczy, należy podjąć pewne działania wstępne w celu zapobieżenia duplikowaniu się wpisów danych, gdyż wszystko, do czego te klucze służą, to identyfikacja egzemplarzy wierszy (a nie elementów zbioru) jako unikatowych. Przykładowo, w przypadku identyfikatora zamówienia, nie przemyślawszy wstępnie problemu, można by wstawiać stale ten sam numer.

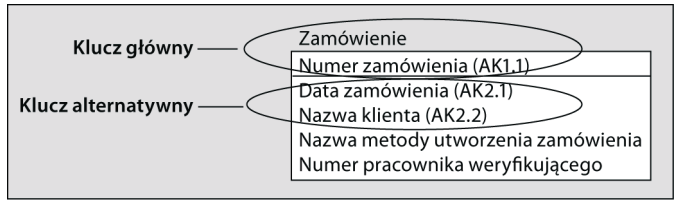
Klucze główne i alternatywne

Klucz główny (ang. *primary key*) jest kluczem kandydującym, który wybrano w celu jednoznacznego identyfikowania egzemplarzy w ramach encji.

Bez względu na dokonany wybór identyfikatora, każdy z nich potencjalnie może migrować w ramach systemu oraz do innych systemów. Taki identyfikator samodzielnie umożliwia dotarcie do reszty atrybutów należących do danej encji. Oznacza to, że bez względu na wybór klucza głównego, nigdy nie powinien on ulegać zmianom. W przeciwnym razie, wszystkie elementy odwołujące się do niego potencjalnie mogłyby stracić powiązanie.

W składni modelowania używanej w niniejszej książce klucze główne zawsze są zapisywane nad linią umieszczoną w symbolu encji, co wyróżnia je spośród innych atrybutów (rysunek 2.13).

Rysunek 2.13.



Wszelkie klucze kandydujące, które pozostaną po określeniu klucza głównego, przyjmują nazwę **kluczy alternatywnych** (ang. *alternate keys*). Należy jednak zauważyć, że choć klucza głównego nigdy nie określa się mianem klucza alternatywnego, wciąż można go do nich zaliczyć, gdyż wszystkie są kluczami kandydującymi.

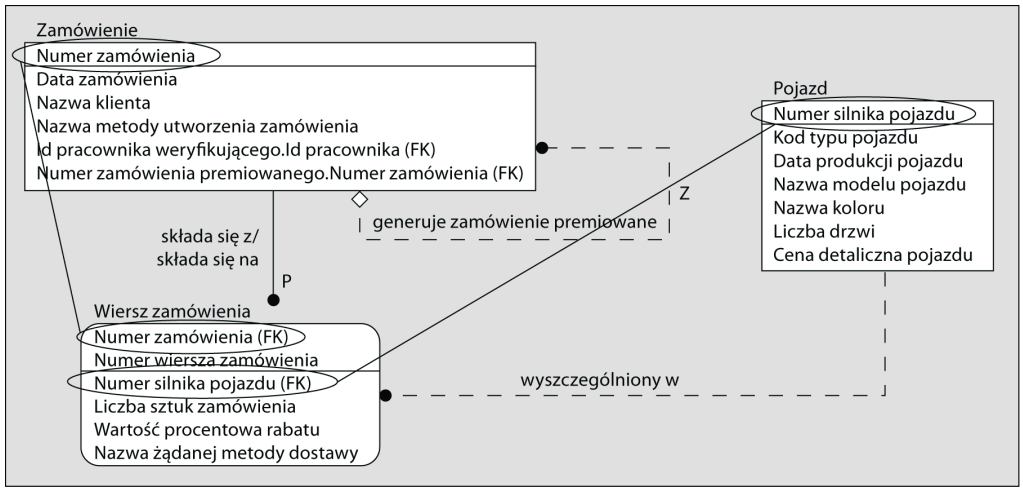
Klucz alternatywny (ang. *alternate key*) to jeden spośród pojedynczego lub wielu zbiorów pojedynczych lub wielu atrybutów niewybranych w celu podstawowego jednoznacznego identyfikowania egzemplarzy w ramach encji, ale który również mógłby jednoznacznie identyfikować te egzemplarze.

Klucze obce

Klucz obcy (ang. *foreign key*) to pełne odwzorowanie atrybutów określonych jako klucz główny jednej encji, które zostały poddane migracji (lub są współużytkowane) poprzez związek do nowej encji.

W przykładzie przedstawionym na rysunku 2.14 atrybut `Numer zamówienia` został poddany migracji do części klucza głównego encji `Wiersz zamówienia`. Atrybut `Numer silnika pojazdu` również został poddany migracji do encji `Wiersz zamówienia`. Nie ma żadnych ograniczeń co do liczby atrybutów poddawanych migracji do pewnej encji.

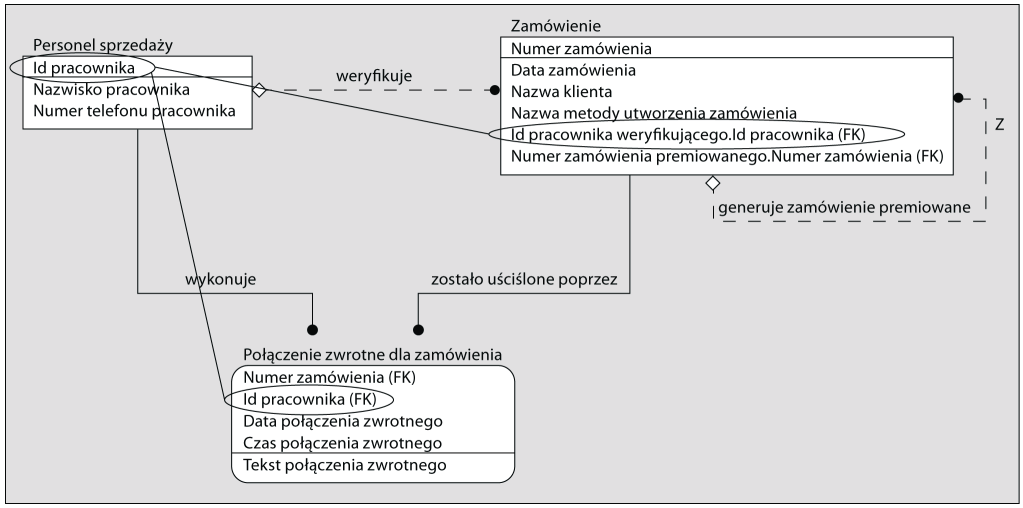
Klucze obce pozwalają „spojrzeć wstecz" na ich źródła i znaleźć inne atrybuty zawarte w danej encji. Encja `Wiersz zamówienia` może uzyskać dostęp do atrybutu `Nazwa klienta` poprzez odpowiednie przejście wstecz i dopasowanie `Numeru zamówienia` do `Zamówienia`. Encja `Wiersz zamówienia` może także uzyskać dostęp do `Nazwy modelu pojazdu`, przechodząc wstecz za pomocą klucza obcego i odpowiednio dopasowując jego wartość. Tak długo jak jeden atrybut jest identyfikowany w encji nadrzędnej jako klucz główny, klucz obcy (cały zbiór atrybutów identyfikujących wybranych do pełnienia roli klucza głównego) jest częścią zbioru atrybutów w powiązanej encji.



Rysunek 2.14.

Nazwy ról

Klucze obce nie muszą zachowywać nazwy atrybutu, od którego się wywodzą. Weźmy pod uwagę choćby rysunek 2.15. Atrybut `Id pracownika` staje się kluczem obcym o nazwie `Id pracownika weryfikującego` w encji `Zamówienie`, a jednocześnie zachowuje nazwę `Id pracownika` w encji `Połączenie zwrotne dla zamówienia`. Nazewnictwo ról zwykle uwzględnia także zmianę definicji. Użycie danych w nowym miejscu może całkowicie zmienić ich definicję. Jednak praktyczna reguła określa, że zachowują one swój rozmiar oraz dziedzinę. Jedyny przypadek, w którym mogą wystąpić różnice, to unifikacja kolumn (o czym będzie mowa w dalszej części książki).



Rysunek 2.15.

Związki

Związki (ang. *relationships*) to logiczne powiązania między encjami.

Związki są w modelu reprezentowane jako linie łączące symbole encji opatrzone dodatkowymi oznaczeniami uściślającymi ich znaczenie. Tak więc ponownie są to potencjalne więzy fizyczne tworzone w momencie przechodzenia od modelu logicznego do modelu fizycznego, ale tu oznaczają one tylko odwołanie do połączenia między encjami. Używane wyrażenia czasownikowe podkreślają ich dynamiczny charakter oraz proces biznesowy, który dane połączenie obsługuje. Niektórzy nazywają zdania utworzone przez wyrażenia rzeczownikowe i czasownikowe encji i związków **regułami biznesowymi** (ang. *Business Rules*).

Związki mają charakter dwustronny — z jednym wyjątkiem obrazują one powiązanie między jedną a drugą encją. To właśnie poprzez definiowanie związków następuje migracja kluczy obcych, przez utworzenie powiązań, dzięki którym można śledzić połączone egzemplarze. Ogólnie rzecz biorąc, strony te określamy mianem **nadrzędnej** (ang. *parent*), inaczej źródłowej, oraz **podrzędnej** (ang. *child*), inaczej docelowej. Wyróżniamy dwa rodzaje związków:

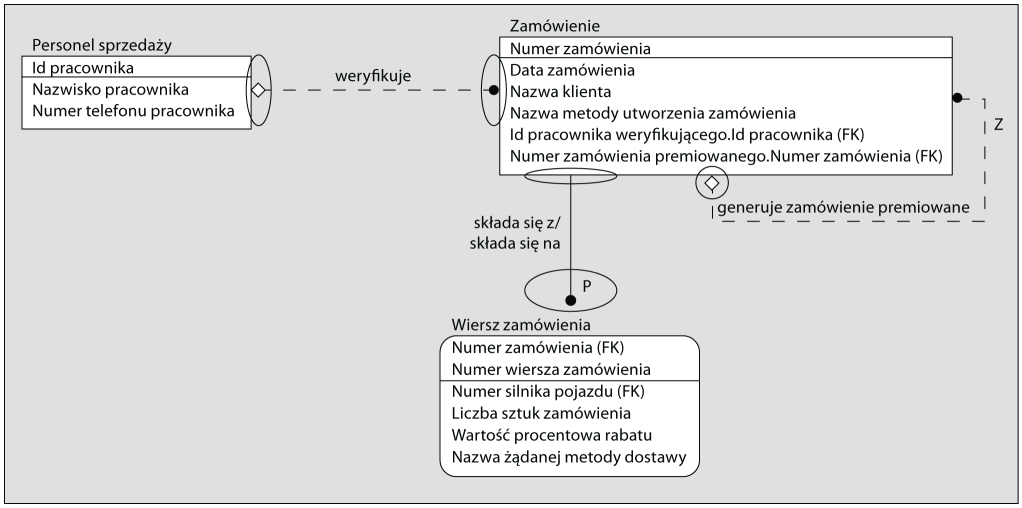
- **identyfikujące** (ang. *identifying*) — w ich przypadku klucz główny strony nadrzędnej migruje do pozycji klucza głównego strony podrzędnej,
- **nieidentyfikujące** (ang. *non-identifying*) — w ich przypadku klucz główny migruje do pozycji nienależącej do klucza.

Każda strona związku posiada określoną liczność oraz symbol możliwości przyjmowania wartości `NULL`. Liczność określa, ile egzemplarzy może być związanych z każdym egzemplarzem encji nadrzędnej. Jedna encja nadrzędna może być związana z encjami podrzędnymi z następującymi licznościami:

- zero lub jeden,
- jeden i tylko jeden,
- zero, jeden lub wiele,
- jeden lub wiele,
- konkretna `<Liczba>`,
- zakres `<Liczba>` – `<Liczba>`,
- dowolna wartość określona dla związku wielokrotnego.

Możliwość przyjmowania wartości `NULL` określa, czy egzemplarze podrzędne muszą być związane z egzemplarzem nadrzędnym. Jeżeli migrowany klucz w egzemplarzu podrzędnym **musi** występować, wówczas opcja ta jest nazywana **wymaganą** (ang. *mandatory*) i wartości `NULL` są niedozwolone. Jeżeli z kolei migrowany klucz w egzemplarzu podrzędnym **może** występować, opcja jest nazywana **niewymaganą** (ang. *nonmandatory*) i wartości `NULL` są dozwolone.

Tak więc związki łączą informacje o postaci (Identyfikujący/Nieidentyfikujący) + (Możliwość występowania wartości NULL) + (Liczność) + (Wyrażenie czasownikowe). Kiedy odczytuje się związek istniejący między dwiema encjami, otrzymuje się zdania, które z łatwością można określić jako prawdziwe lub fałszywe reguły dotyczące zachowania zbiorów danych. Weźmy raz jeszcze pod uwagę przykład Zamówienia (rysunek 2.16).



Rysunek 2.16.

Każdy Wiersz zamówienia może zostać zidentyfikowany tylko wówczas, gdy znane jest jego Zamówienie. Prawda czy fałsz? Prawda. A jak wygląda sytuacja w przypadku weryfikacji Nazwiska pracownika? Czy musimy znać pracownika, który zweryfikował Zamówienie w celu zidentyfikowania Zamówienia? Nie, ponieważ zamówienia są identyfikowane przez unikatowy numer (choć możemy wybrać mniejszy ich zbiór, ograniczając listę do osób, które je zweryfikowały). Gdyby zdanie to było prawdą, nie mielibyśmy możliwości użycia przyjmującego wartości NULL, nieidentyfikującego związku (w prezentowanej składni modelowania oznaczanego symbolem rombu). Klucz główny nie może zawierać wartości NULL, ponieważ migruje do niego klucz główny encji nadrzędnej w ramach związku identyfikującego. Pozwalamy, aby związek ten przyjmował wartości NULL, ponieważ przez pewien czas Zamówienie może istnieć bez przejścia przez proces weryfikacji. W czasie modelowania trzeba uważnie analizować tego rodzaju sytuacje.

W tabeli 2.2 przedstawiono reguły dotyczące przyjmowania wartości NULL oraz liczności.

Należy zwrócić uwagę, że tylko liczności dopuszczające wartość zero mogą być licznościami NULL. Tak więc łącząc informacje o możliwości przyjmowania wartości NULL z licznością, ściśle określamy regułę biznesową.

Wiele do wielu

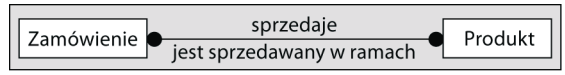
W przypadku modelowania koncepcyjnego możemy po obu stronach związku używać dowolnych wartości liczności. W przypadku takiego związku nie występują elementy

Tabela 2.2.

Reguła związku	Prawda lub fałsz
Każde Zamówienie może posiadać zero lub jeden Wiersz zamówienia.	Fałsz
Każde Zamówienie może posiadać zero, jeden lub wiele Wierszy zamówienia.	Prawda (jeżeli można tworzyć zamówienia przed podjęciem decyzji co do wyboru pojazdu)
Każde Zamówienie musi posiadać jeden i tylko jeden Wiersz zamówienia.	Fałsz
Każde Zamówienie musi posiadać jeden lub wiele Wierszy zamówienia.	Fałsz
Każde Zamówienie musi posiadać 7 Wierszy zamówienia (jednak nie mniej ani nie więcej).	Fałsz
Każde Zamówienie musi posiadać od 1 do 7 Wierszy zamówienia (ale nie więcej niż 7 i nie mniej niż 1).	Fałsz
Każde Zamówienie musi posiadać liczbę Wierszy zamówienia równą numerowi bieżącego dnia kalendarzowego (jednak nie mniej ani nie więcej).	Fałsz

nadrzędne ani podrzędne. Są one jedynie oznaczane jako powiązane i muszą zostać uściślone poprzez wykorzystanie encji powiązania lub przecięcia (tak jak zrobiliśmy to wcześniej w przypadku encji Wiersz zamówienia oraz Połączenie zwrotne zamówienia), kiedy wystąpi potrzeba uściślenia modelu (rysunek 2.17).

Rysunek 2.17.



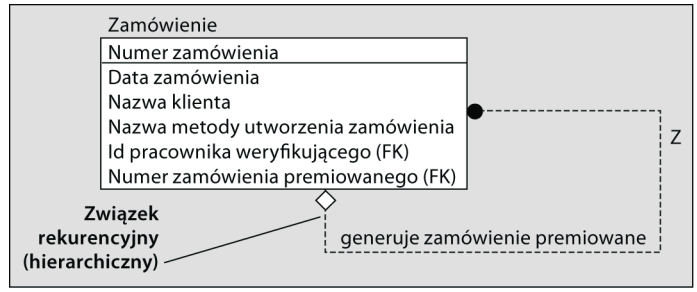
Przedstawiony na powyższym rysunku schemat należałoby zaklasyfikować jako pojedynczą encję koncepcyjną (choć jest to dopuszczalne także na poziomie związków encji modeli logicznych). Opisuje ona pojęcie o szerokim znaczeniu i wymaga dalszej analizy, aby stać się przydatną.

Związki rekurencyjne (hierarchiczne)

Jest to związek encji z samą sobą. Linia związku tworzy pętlę określaną potocznie „uchem świnki". Kontynuując przedstawiony wcześniej przykład, powiedzmy, że co dziesiąte Zamówienie o wartości ponad 50 złotych otrzymuje specjalną premię. Jednak ze względu na fakt, że premie są finansowane przez dział marketingu, wymagają one odrębnego Zamówienia w celu obsługi kwestii w rodzaju faksów oraz zapewnienia uniezależnienia finansowego od Klienta. Odpowiednia reguła określa, że Zamówienie premiowane musi być wynikiem kontynuowania procesu przetwarzania Zamówienia i tylko jedno Zamówienie premiowane może zostać utworzone dla każdego Zamówienia. Wynikiem tych działań jest schemat rekurencyjny prowadzący od Zamówienia do Zamówienia.

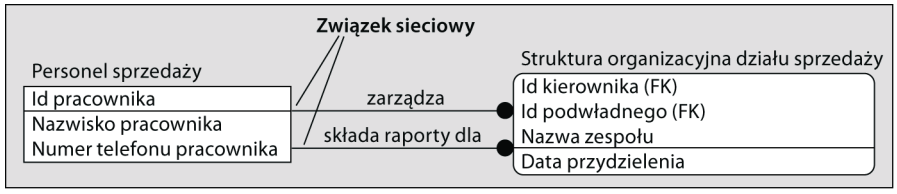
Rekurencja tworzy drzewo hierarchii, w którym każdy element podrzędny może posiadać tylko jeden element nadrzędny. Używając na rysunku 2.18 zapisu Z dla oznaczenia opcji „zero lub jeden", rekurencję oznaczamy jako pojedyncze powiązanie prowadzące w dół.

Rysunek 2.18.



Związki sieciowe

Związek sieciowy uwzględnia tę samą zasadę łączenia dwóch lub większej liczby encji, ale zapewnia znacznie większą elastyczność niż związek rekurencyjny, gdyż w tym przypadku element podrzędny może posiadać wiele elementów nadrzędnych. Używamy ich przez cały czas w celu tworzenia schematów organizacyjnych oraz schematów części składowych. Na rysunku 2.19 przedstawiono przykład schematu organizacyjnego.



Rysunek 2.19.

Załóżmy, że chcemy utworzyć siatkę obrazującą osoby kierujące zespołami oraz należące do nich. Związek rekurencyjny nie mógłby zapewnić wykonania tego zadania, gdyż dopuszcza on tylko jeden związek dla zbioru członków zespołu i jest zbyt restrykcyjny. W omawianym przypadku `Pracownik` może być zarówno `Kierownikiem`, jak i `Podwładnym`. Pracownicy mogą kierować wieloma zespołami i należeć do wielu zespołów.

Tego rodzaju dualny związek daje nieograniczone możliwości, jeśli chodzi o głębokość i szerokość sieci. Wymaga jednak uwzględnienia pewnego zagrożenia na poziomie fizycznej implementacji. Chodzi tu o zapewnienie, aby żaden rekord nie mógł być jednocześnie swoim elementem podrzędnym, jak i nadrzędnym. Spowodowałoby to utworzenie pętli nieskończonej, w której zapytanie pobierające szczegółowe dane nie mogłoby zakończyć działania.

Reguły biznesowe modelu relacyjnego

Reguły biznesowe łączą encje i związki w ramach pojedynczych zdań, które mogą być przeglądane przez klientów i ewentualnie odrzucane jako nieprawdziwe. Ze względu na fakt, że w ramach modelu wszystko staje się niepodzielne, niekiedy reguły mogą wydawać się bardzo naiwne. Przykładowo, można by uznać, że wszyscy wiemy, że „jeden tydzień musi mieć siedem dni". Jeśli jednak tydzień zostanie zdefiniowany jako „okres od niedzieli do soboty", to jest bardzo prawdopodobne, że każdego roku będą

występować dwa tygodnie, które nie mają siedmiu dni — pierwszy tydzień w styczniu oraz ostatni tydzień w grudniu. Kiedy autorka pierwszy raz spotkała się z 54 tygodniami w roku, była bardzo zaskoczona, a nie zostało to wychwycone w trakcie przeglądu reguł biznesowych, ponieważ wszyscy sądzili, że niektóre z nich nie są warte głębszej analizy. Owo drobne przeoczenie spowodowało tygodniowe dyskusje związane z potrzebą obsługi raportu analitycznego „tydzień po tygodniu". Oczywiście w tym przypadku złamano zasadę mówiącą o tym, że do prowadzonych analiz należy podchodzić bez odgórnych ustaleń i należy zachować obiektywizm.

Reguły biznesowe formułuje się tak, aby móc je odczytywać w dwóch kierunkach:

- „Każdy" || „Nazwa encji nadrzędnej" || wyrażenie czasownikowe związku || liczność || „Nazwa encji podrzędnej"
- „Każdy" || „Nazwa encji podrzędnej" || wyrażenie czasownikowe związku || liczność || „Nazwa encji nadrzędnej"

W składni modelowania, z której będziemy korzystać, standardowym sposobem zapisu wyrażeń czasownikowych jest stosowanie osoby trzeciej liczby pojedynczej. Następnie dodawana jest druga reguła biznesowa w celu uwzględnienia zasad dotyczących przyjmowania wartości `NULL`. Ma ona następującą postać:

Wartości `NULL` niedozwolone:

- Każdy || „Nazwa encji podrzędnej" || należy do || liczność || „Nazwa encji nadrzędnej"

Wartości `NULL` dozwolone:

- Każdy || „Nazwa encji podrzędnej" || może należeć do || liczność || „Nazwa encji nadrzędnej"

Weźmy pod uwagę przykład przedstawiony na rysunku 2.20.

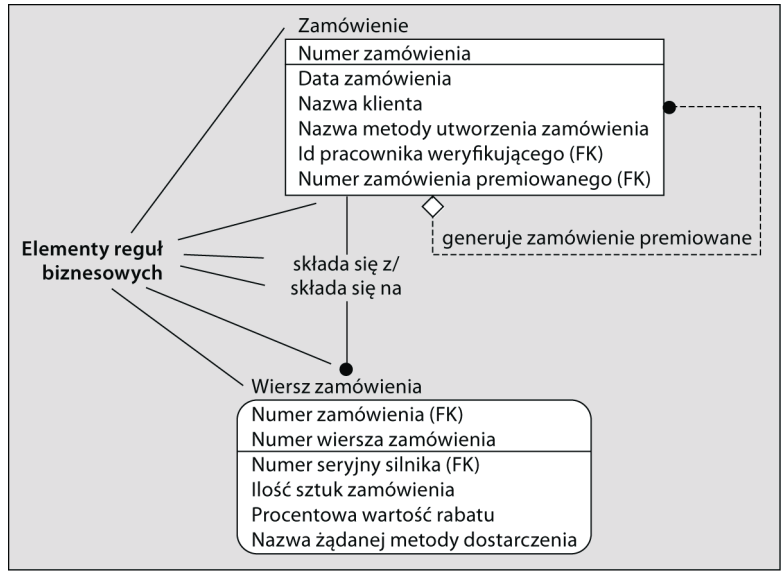
Możemy tu utworzyć następujące reguły biznesowe:

1. Każde `Zamówienie` składa się z zera, jednego lub wielu `Wierszy zamówienia`.
2. Każdy `Wiersz zamówienia` składa się na jedno i tylko jedno `Zamówienie`.
3. Każdy `Wiersz zamówienia` musi należeć do jednego i tylko jednego `Zamówienia`.

W niniejszej książce będziemy korzystać z zapisu nieco różniącego się od standardu IDEF1X, co pozwoli nam na przeglądanie reguł biznesowych z uwzględnieniem czynnika dotyczącego wartości `NULL`:

- „Każdy" || „Nazwa encji nadrzędnej" || określenie występowania wartości `NULL` || wyrażenie czasownikowe związku || liczność || „Nazwa encji podrzędnej"
- „Każdy" || „Nazwa encji podrzędnej" || określenie występowania wartości `NULL` || wyrażenie czasownikowe związku || liczność || „Nazwa encji nadrzędnej"

Rysunek 2.20.



W ten sposób możemy utworzyć następujące reguły dla powyższego przykładu:

1. Każde `Zamówienie` może składać się z zera, jednego lub wielu `Wierszy zamówienia`.
2. Każdy `Wiersz zamówienia` musi składać się na jedno i tylko jedno `Zamówienie`.

Czytając je, można wywnioskować, że `Wiersz zamówienia` nie może istnieć po usunięciu `Zamówienia`, którego jest częścią. Ponadto `Wiersz zamówienia` nie może być częścią więcej niż jednego `Zamówienia` oraz `Wiersze zamówienia` nie mogą być tworzone, dopóki nie zostanie utworzone `Zamówienie`.

Niekiedy wyciąganie wniosków na podstawie czytanych reguł jest ważniejsze od samych reguł, ponieważ widać wówczas, jak restrykcyjne w rzeczywistości są reguły i można sobie uświadomić, że występują sytuacje, w których nie są one zachowywane. Reguły biznesowe muszą uwzględniać wszystkie sytuacje możliwe do zaistnienia w miarę upływu czasu.

W świecie specjalistów od modelowania danych istnieje pewna niezgodność co do tego, czy wyrażenia encji i związków są regułami biznesowymi czy nie. Zdaniem autorki stanowią po prostu jeden z typów reguł biznesowych. Zdanie w rodzaju „Pracownik może być przydzielony do zera, jednego lub wielu Projektów" brzmi bardzo podobnie do reguły biznesowej. Jednak podobnie jest w przypadku zdania „Klient może uzyskać prawo do darmowego towaru premiowego, kupując produkty za kwotę 100 złotych w ciągu trzech miesięcy od czasu złożenia pierwszego zamówienia". Zakładając poprawność modelu logicznego, tę regułę biznesową można utworzyć na podstawie kilku decyzji typu prawda-fałsz, jednak nie jest to proste wyrażenie encji i związku. Reguły biznesowe to więcej niż tylko to, co jest modelowane, ale model musi je obsługiwać.

Pojęcia z zakresu modelowania fizycznego

Dopiero na etapie opracowywania modelu fizycznego zwraca się cały wysiłek włożony w dokonanie analizy elementów danych i procesów biznesowych. Encje stają się tabelami kandydującymi, atrybuty kolumnami kandydującymi, zaś związki — więzami kandydującymi. Wiele razy można spotkać się z niemal domyślnym procesem „bierzemy to, co jest", który występuje na pierwszym etapie opracowania modelu fizycznego. W tym momencie, ujmując rzecz obrazowo, teoria przechodzi test praktyczny. Proces strojenia, o którym sądziło się, że został przeprowadzony na etapie modelowania logicznego, musi zostać powtórzony, ale tym razem z dużo większym zrozumieniem stałego wpływu modelu na funkcjonowanie biznesu. Modele logiczne można tworzyć cały dzień, nie wpływając na pracę zbyt wielu osób. Jednak tym razem nie mamy do czynienia z taką sytuacją. Każda wybrana nazwa, popełniony błąd w zapisie, zbyt mały dobrany rozmiar, błędnie określony typ danych oraz niepoprawnie zdefiniowane więzy będą mieć wpływ na całe zespoły opracowujące i konserwujące system przez najbliższe lata. Na szczęście modelarz danych nie jest pozostawiony sam sobie w procesie tworzenia projektu fizycznego, gdyż administrator bazy danych oraz programiści zwykle biorą udział w burzy mózgów i przeglądzie projektu jeszcze przed faktycznym zaimplementowaniem czegokolwiek. Choć na tym etapie wciąż wprowadza się poprawki i zmiany, fundamentalne znaczenie ma posiadanie jak najlepiej opracowanego projektu, nawet w przypadku rozwojowych i testowych baz danych tworzonych w celu sprawdzenia projektu za pomocą odpowiedniego zbioru danych i zapytań testowych.

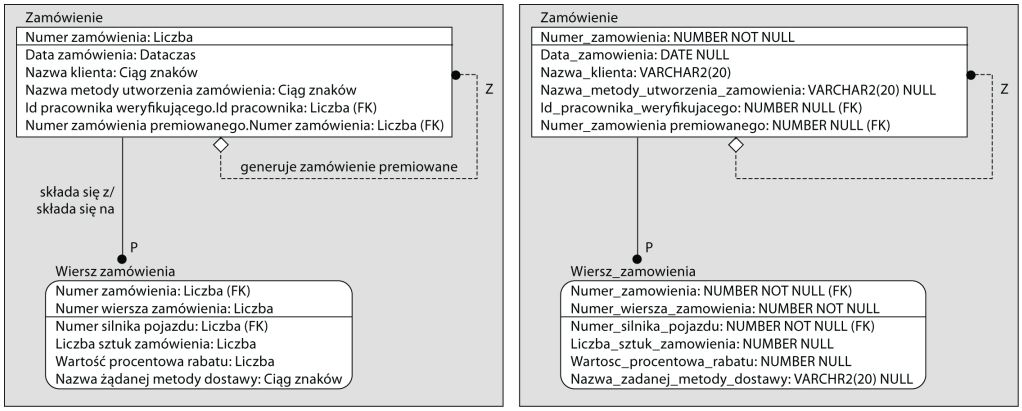
Tabele

W sieciowych i hierarchicznych systemach zarządzania bazami danych tabele określa się mianem **plików** (ang. *files*) lub **rekordów** (ang. *records*). Z fizycznego punktu widzenia są one odpowiednikiem encji. Kiedy rozmawia się z klientem nieposiadającym żadnej wiedzy z zakresu baz danych, tabele można porównać do arkuszy kalkulacyjnych lub księgi głównej. Tabela stanowi pojedynczy kontener danych zorganizowanych w postaci wierszy i kolumn. Podobnie jak w przypadku arkuszy kalkulacyjnych tabele mogą być ze sobą łączone w celu tworzenia połączeń kluczy obcych z rekordu znajdującego się w jednym arkuszu do rekordu znajdującego się w innym.

Zamiana encji na tabelę nie jest trudna. W przypadku narzędzi służących do modelowania danych stanowi to po prostu kwestię przejścia z widoku logicznego na fizyczny. Trzeba jednak pamiętać, że należy w tym momencie na wszystko spojrzeć z nieco odmiennej perspektywy, a konkretnie z perspektywy wybranej platformy systemu obsługi baz danych, na której zostanie wdrożony model fizyczny. System DB2 charakteryzuje się ograniczeniami co do długości nazw, zaś system SQL Server nie obsługuje komentarzy tabel i kolumn umożliwiających zapisywanie definicji na poziomie bazy danych. Każdy system charakteryzują pewne cechy szczególne, które mają wpływ na sposób tworzenia modelu fizycznego. Trzeba poznać używany system w celu uświadomienia sobie jego zalet i wad.

W poniższych przykładach będzie wykorzystywany system Oracle oraz notacja wykorzystująca znak podkreślenia w zastępstwie znaków spacji w nazwach kolumn.

W przypadku prostej transformacji modelu logicznego na fizyczny encje logiczne stają się tabelami, jak w przykładzie przedstawionym na rysunku 2.21.



Rysunek 2.21.

Przedstawiono tu także dziedziny danych atrybutów, tak aby można było porównać je z domyślnymi typami danych utworzonymi w tabelach fizycznych. Jak widać, w czasie transformacji zostało wprowadzonych także kilka innych zmian poza zamianą znaków spacji na znaki podkreślenia. Niekiedy otrzymanie modelu fizycznego jest właśnie tak proste. Należy jednak pamiętać, że istnieją pewne reguły, których należy przestrzegać, dotyczące słów zarezerwowanych w zależności od używanego języka programowania w celu zapisu kodu. Nie można na przykład stosować prostych nazw typu `Date` lub `Column` jako nazw kolumn. Może również być wskazane skrócenie długich nazw niektórych kolumn poprzez utworzenie standardowej listy skrótów. Różne firmy posiadają różne standardy i podejścia do kwestii konwencji nazewnictwa obiektów w bazach danych.

Więcej informacji na temat przekształcania modelu logicznego na fizyczny zostanie przedstawionych w rozdziale 8.

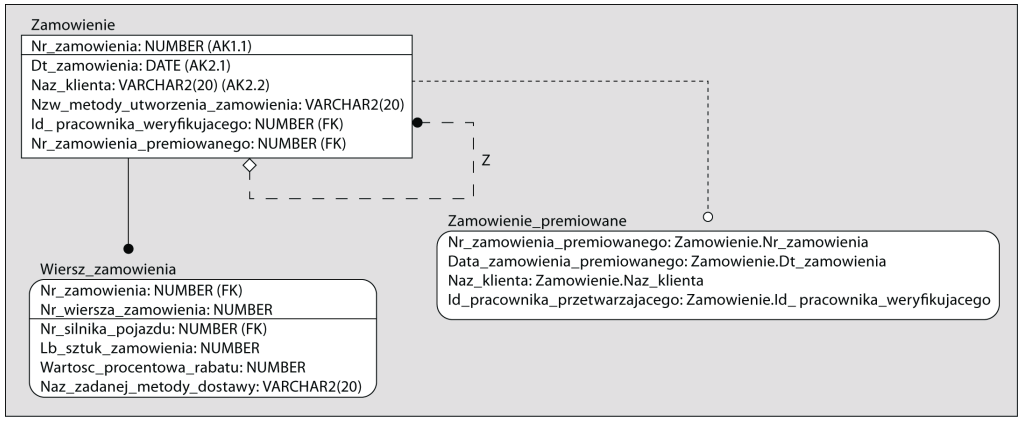
Tabele operacyjne

Encje logiczne mogą być niejedynymi tabelami, jakie trzeba utworzyć w celu zapewnienia działania docelowej aplikacji. Czasem potrzebne są tabele nieposiadające encji źródłowych. Tabele te można określić mianem **tabel operacyjnych** (ang. *operational tables*), choć określa się je również jako **tabele użytkowe** (ang. *utility tables*). Służą one po prostu pomocą w zakresie zarządzania procesami, które muszą występować w celu zapewnienia spełnienia wymagań systemu. Na przykład może okazać się konieczne uwzględnienie tabeli związanej z obsługą podsystemu zabezpieczeń, zawierającą na przykład zaszyfrowane nazwy użytkowników i hasła wymagane przez administratora. Może się okazać, że programista zechce przechowywać w tabeli wartości związane

z dzienną liczbą odwiedzin stron WWW lub z przechowywaniem błędów w celu ich późniejszej analizy. Prawdopodobnie wskazane byłoby również umożliwienie przechowywania historii procesu wsadowego ładowania danych przez kilka tygodni przed ich zarchiwizowaniem. Można wymienić setki tego typu wymagań w kontekście modelu fizycznego. Tabele te, ogólnie rzecz biorąc, nie są modelowane w ramach modelu logicznego, ale są dodawane w procesie opracowywania modelu fizycznego.

Perspektywy

Perspektywa to (ogólnie rzecz ujmując) struktura tymczasowa, bazująca na wynikach zapytania, która jednak może być używana w ten sam sposób (z kilkoma ograniczeniami) jak tabela. Perspektywy mogą stanowić część tworzonego projektu, wspierając obsługę wymagań i funkcjonalność aplikacji (rysunek 2.22).



Rysunek 2.22.

Należy zwrócić uwagę, że nazwy kolumn w perspektywie nie muszą odpowiadać nazwom kolumn występujących w oryginalnych tabelach. Wiele perspektyw tworzy się po prostu po to, by klienci mieli dostęp do nazw, które będą dla nich łatwiejsze do rozpoznania. Powyższą perspektywę utworzono w celu zapewnienia obsługi ekranu *Zamówienie premiowane* dla działu marketingu w celu umożliwienia tworzenia raportów dotyczących tego rodzaju zamówień generowanych przez wykorzystywane narzędzie definiowania zapytań ad hoc.

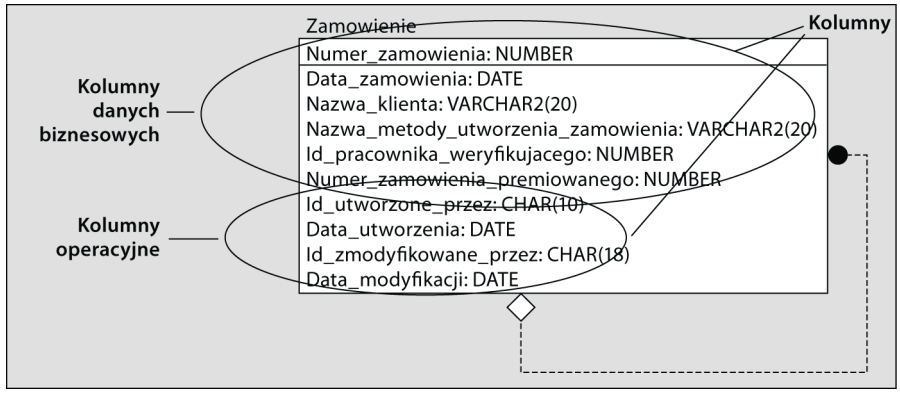
Kolumny

Wszystkie atrybuty, które udokumentowano jako reprezentujące elementy danych biznesowych, należy traktować jako kandydujące do roli kolumn. Można wyróżnić kilka dodatkowych czynników, które trzeba wziąć pod uwagę, zanim przeniesie się atrybuty do modelu fizycznego. Muszą one posiadać metodę tworzenia i konserwacji oraz powinny posiadać obiekt zarządzający w celu weryfikacji wartości. Jest to szczególnie istotne w przypadku encji niezależnych, takich jak `Lista wartości`, lub źródeł istotnych i często wykorzystywanych danych, takich jak kody pocztowe i nazwy państw stających się tabelami.

Należy starać się nie przekazywać do projektu bazy danych atrybutu, jeżeli nikt nie ma się zamiaru podejmować odpowiedzialności za jego jakość i obsługę. W tym kontekście modelarz jest w pewnym stopniu odpowiedzialny za zweryfikowanie faktu, czy dany atrybut ma szansę stać się zasobem firmowym, a nie tylko kolejnym zbiorem danych obciążających system zarządzania relacyjną bazą danych i wydłużającym czas diagnozowania, tworzenia kopii bezpieczeństwa lub odtwarzania. Może się wydawać, że o kwestii tej wypadało wspomnieć przy omawianiu tabel, jednak problem jest bardziej złożony. Dodatkowe przejrzenie atrybutów wraz z klientem biznesowym i określenie, czy są one na tyle ważne, aby pewna osoba zajmowała się ich nadzorem jakościowym, stanowi dobry test wykonywany przed faktycznym dodaniem danego atrybutu jako kolumny tworzonej tabeli.

Kolumny operacyjne

Kolumny operacyjne bardzo przypominają pod względem struktury tabele operacyjne. Są one potrzebne zespołowi programistów w celu zapewnienia prostoty i poprawności wykonywanych działań. Niemal wszystkie tabele mogą wykorzystywać grupę kolumn, która stanowi znacznik umożliwiający analizę w przypadku tworzenia lub modyfikowania wierszy. Niekiedy występuje kolumna znacznika, z której korzysta programista w celu śledzenia stanu złożonych procesów. Przykładowo, weźmy pod uwagę przedstawiony na rysunku 2.23 projekt fizyczny tabeli `Zamowienia`.



Rysunek 2.23.

Także w tym przypadku tego typu kolumny nie występują w modelu logicznym, lecz są dodawane w projekcie fizycznym.

Więzy

Prostym sposobem interpretacji więzów jest fizyczne wdrożenie logicznego związku, który reprezentuje regułę biznesową. Bardziej wyrafinowane programy wspomagające tworzenie oprogramowania mogą budować więzy bezpośrednio na podstawie definicji tworzonych w modelu fizycznym. Przydatną cechą więzów stosowanych w systemach sterowania relacyjnymi bazami danych jest to, że zachowują się one niczym prawa grawitacji — nie można ich naruszyć, chyba że się je dezaktywuje.

Istnieją pewne bardzo skomplikowane więzy, które także są tworzone w systemie zarządzania bazami danych. Ogólnie rzecz biorąc, nie mogą one być obrazowane po prostu jako linie między encjami na schemacie modelu danych. Więzy te, jako obiekty bazy danych, wykraczają poza zakres tematyczny prezentowanej tu dyskusji.

Poniżej przedstawiono prosty przykład więzów klucza obcego:

```
ALTER TABLE Wiersz_zamowienia
      ADD ( FOREIGN KEY (Numer_zamowienia)
            REFERENCES Zamowienie ) ;
```

Jeżeli zdefiniowane więzy określają, że element nadrzędny musi posiadać co najmniej jeden element podrzędny, system zarządzania relacyjną bazą danych zapobiegnie wstawieniu rekordu nadrzędnego bez odpowiadającego mu rekordu podrzędnego. Operacjom usuwania rekordów zapobiega się, jeżeli są one połączone poprzez klucze obce z innymi tabelami. Tak więc więzy takie zapobiegają usunięciu rekordu `Zamowienia`, jeżeli nie zostaną również usunięte wszystkie odpowiadające mu rekordy `Wiersz_zamowienia`. Chroni to przed występowaniem rekordów **osieroconych** (ang. *orphan*), niepowiązanych z żadną inną wartością w bazie danych, które naruszają spójność referencyjną.

Składnia modelowania

Po omówieniu elementów koncepcyjnych, takich jak encje, atrybuty, klucze i związki, przyjrzymy się symbolom graficznym używanym w celu graficznego reprezentowania modeli relacyjnych. Istnieje wiele różnych standardów modelowania i wybranie jednego z nich w celu przeprowadzania analiz danych może być jedną z najbardziej interesujących (a zarazem frustrujących) kwestii, jakimi trzeba się zająć. Choć wszystkie te style są do siebie podobne, występują tu różnice w szczegółach obrazowania poszczególnych elementów. Jak wynika z doświadczenia autorki, kiedy w danej formie raz wybierze się konkretny styl rysowania modeli, nie jest on już zmieniany. Często okazuje się konieczne nauczenie klientów i programistów odczytu modeli, więc wskazane byłoby zrobienie tego tylko dla jednego stylu. Jednak znajomość zalet i wad różnych stylów może pomóc w podjęciu decyzji, który z nich będzie najbardziej przydatny. Decyzji nie należy podejmować zbyt szybko. Chociaż poniżej skupimy się na standardzie IDEF1X, przedstawimy także trzy inne style modelowania.

Symbole standardu Integration DEFinition (IDEF1X)

Dyskusję rozpoczynamy od standardu IDEF1X, ponieważ jest to notacja, z której będziemy korzystać w pozostałej części książki. Jest on obsługiwany przez większość narzędzi CASE oraz służących do modelowania i jest językiem używanym w projektach rządowych Stanów Zjednoczonych. Został on opracowany w latach 70. XX w. przez US Air Force i uzupełniony w 1993 roku przez D. Appletona. Przeszedł już etap testowania i pozwala na obrazowanie większości informacji dotyczących związków istniejących między danymi. Jednak w przeciwieństwie do łaciny nie jest językiem martwym —

jest wciąż rozwijany w celu uwzględnienia niektórych nowych potrzeb związanych z językiem UML oraz technikami obiektowymi, których obsługę wprowadzono w jego najnowszej wersji IDEF1X97.

Wykorzystanie standardu IDEF1X stanowi kwestię osobistych preferencji — ogólnie rzecz biorąc, takie same analizy można zapisywać w wielu stylach. Niektóre z nich posiadają symbole uwzględniające pojęcia niewystępujące w innych. Przykładowo, notacja Barkera pozwala obrazować związki alternatywne, czego nie umożliwia standard IDEF1X. Bez względu na dokonany wybór należy uczyć się metod stosowania danego stylu do momentu osiągnięcia biegłości w posługiwaniu się nim.

Symbole używane w standardzie IDEF1X są na tyle podobne do symboli występujących w innych stylach, że łatwo je rozpoznać, a jednocześnie są na tyle różne, że mogą prowadzić do pewnych nieporozumień, jeśli się ich nigdy wcześniej nie używało.

- Encje obrazuje się za pomocą prostokątów o ostrych lub zaokrąglonych rogach. Rogi zaokrąglone oznaczają brak zależności identyfikacji względem innej encji. Rogi ostre oznaczają sytuację odwrotną.
- Związki są oznaczane za pomocą linii ciągłych, jeżeli klucz obcy jest identyfikujący, oraz przerywanych, jeżeli jest nieidentyfikujący. Linie ciągłe i przerywane są opatrywane symbolami końcowymi, które można łączyć ze sobą w celu uwzględnienia różnych reguł liczności i opcjonalności.
- Encje będące kategoriami posiadają symbole zupełności lub niezupełności.
- Atrybuty są wyświetlane w zależności od wybranego widoku.
- Klucze główne są oddzielane od innych atrybutów linią poziomą.

Bez względu na wybrany styl rysowania to, co podlega modelowaniu, zasadniczo nie ulega zmianie. Model można by zapisać w nieco odmienny sposób w zależności od stosowanej składni. Poniżej przyjrzymy się bliżej tej notacji.

Prostokąty

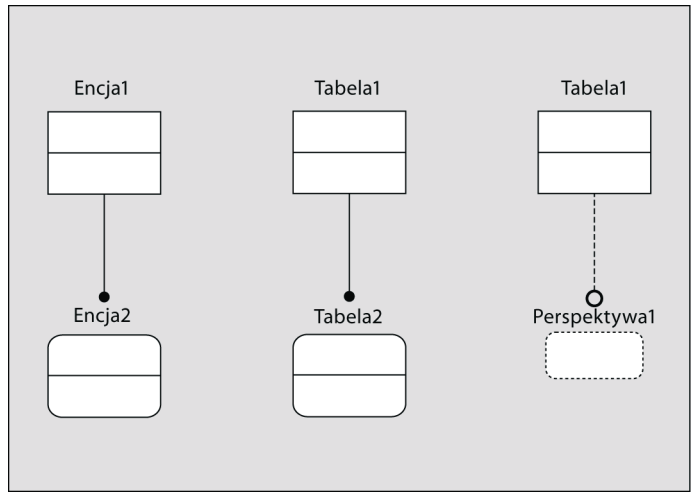
Prostokąty oznaczają trzy różne pojęcia. Modele logiczne zawierają encje, natomiast modele fizyczne zawierają tabele i perspektywy. Z tego powodu często trudno jest na pierwszy rzut oka określić, czy mamy przed sobą model fizyczny czy logiczny (rysunek 2.24).

Poniżej opisano pewne podstawowe reguły dotyczące prostokątów występujących w modelu.

Składnia dualna

Wszystkie prostokąty w modelu reprezentują albo encje, albo tabele i perspektywy. Nie mogą one być ze sobą mieszane. Encje i tabele lub perspektywy nigdy nie są używane wspólnie. Choć mogą wyglądać tak samo, obiekty logiczne i fizyczne nigdy nie są umieszczane w ramach tego samego modelu.

Rysunek 2.24.



Pojedynczy cel

Prostokąty służą jednemu celowi — stanowią wizualne granice otaczające zbiory danych. W modelu relacyjnym oznacza to zasadniczo dwie rzeczy:

- Prostokąty nie reprezentują niczego innego niż encje i tabele lub perspektywy.
- Wszystkie encje i tabele lub perspektywy są reprezentowane jako prostokąty. Nie są reprezentowane w żaden inny sposób. Nawet kopie reprezentujące fakty odkryte w innych analizach logicznych lub faktyczne obiekty pochodzące z innych baz danych, które nie zostaną wdrożone, wciąż pozostają w ramach modelu prostokątami. Zwykle wyróżnia się je innym kolorem, tekstem, formatem lub nazwą w celu odpowiedniego ich zidentyfikowania.

Niezależność lokalizacji

W przypadku notacji IDEF1X prostokąty nie są ustawiane w żadnym konkretnym porządku. Nie są one ustawiane w żaden szczególny sposób i ich rozmieszczenie nie niesie ze sobą jakiegokolwiek znaczenia. Otrzymuje się zawsze ten sam kod bez względu na sposób ich ustawienia w ramach modelu. Ustawienie to ma służyć jedynie spełnieniu wymagań związanych z komunikowaniem odkrytych faktów na temat danych. Można spotkać osoby przyzwyczajone do określonych konwencji rozmieszczania elementów, reagujących negatywnie na ich przenoszenie. Jednak można usłyszeć też głosy, że porządek modelu powinien być określany przez porządek tworzenia obiektów fizycznych, tak aby wszystko, co musi zostać utworzone w pierwszej kolejności, znajdowało się nad innymi elementami. Spotyka się też głosy, że wszystkie linie związków powinny albo znajdować się „na górze i wychodzić od dołu", albo po stronach oznaczających migrowany klucz.

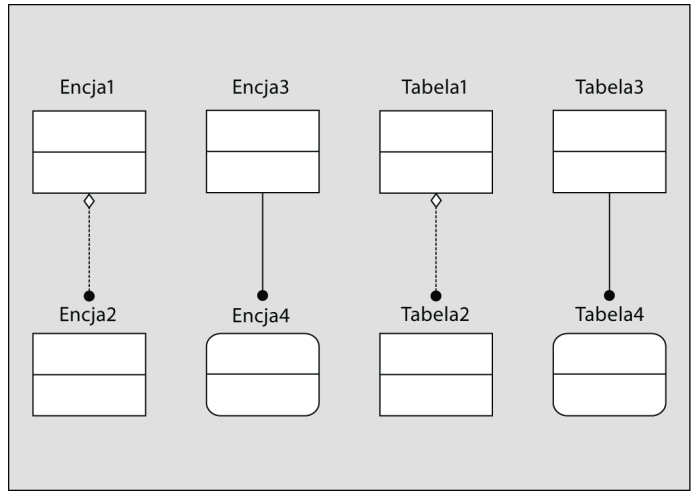
Warto skoncentrować się na zapewnieniu takiego ustawienia obiektów w modelu, aby utworzyć przydatny produkt dla swoich klientów. Na przykład autorka ma tendencję do podejmowania prób zajmowania jak najmniejszej ilości miejsca, tak aby ułatwić

drukowanie pełnego modelu. Nie zda się to na wiele, jeżeli podejmie się współpracę z kimś, kto wymaga wyraźnego określenia kolejności tworzenia obiektów poprzez umieszczenie encji niezależnych nad encjami zależnymi. Należy kierować się własnym osądem i być wyczulonym na potrzeby klientów.

Znaczenie stylu rogów — zależne i niezależne

Rogi prostokątów oznaczają rodzaj zależności między encjami. Prostokąty z rogami ostrymi oznaczają niezależne encje lub tabele, zaś prostokąty z rogami zaokrąglonymi oznaczają zależne encje lub tabele. Jeżeli zachodzi potrzeba zapewnienia migracji klucza głównego `Encji3` w celu identyfikacji `Encji4`, wówczas `Encja4` staje się encją **zależną** (ang. *dependent*). Podobnie jest w przypadku tabel. Perspektywy zawsze są zależne, ponieważ nie mogą istnieć bez tabel, na których są oparte (rysunek 2.25).

Rysunek 2.25.

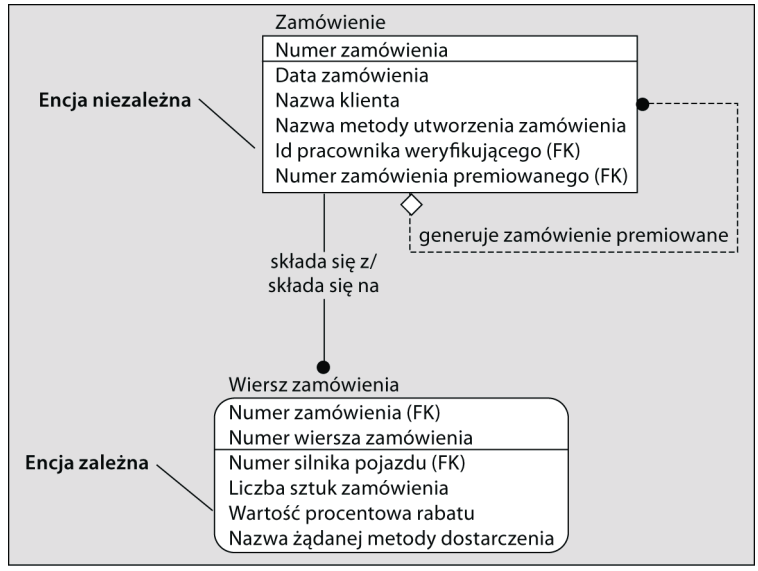


Na rysunku 2.26 przedstawiono adekwatny przykład.

Na podstawie powyższego przykładu można stwierdzić, że `Wiersz zamówienia` jest zależny od `Zamówienia` w zakresie części swojej identyfikacji. Potrzebny jest nam identyfikator `Numeru zamówienia` w celu uzupełnienia identyfikatora (klucza głównego) `Wiersza zamówienia`.

Należy również pamiętać, że zależność jest stanem posiadania części pełnego identyfikatora w innej encji lub tabeli i ma znaczenie dla kolejności wykonywania niektórych operacji tworzenia i usuwania danych. Chciałoby się powiedzieć, że może to być wizualną wskazówką co do wszystkich zależności sekwencyjnych, jednak nie jest to prawdą. Wpływ na sekwencyjne zadania tworzenia i usuwania obiektów, więzów i danych wywierają również określone związki wymagane, choć w notacji IDEF1X nie są przedstawiane jako prostokąty z zaokrąglonymi rogami.

Rysunek 2.26.



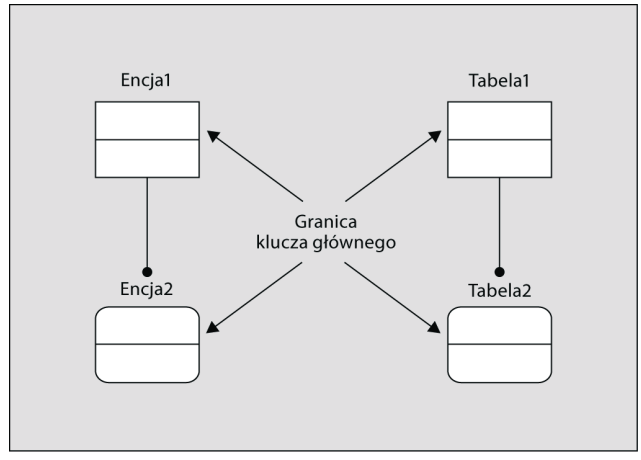
Linie

Linie także niemal w każdym aspekcie mają identyczną postać w modelu logicznym oraz fizycznym i służą do segregowania, łączenia i oznaczania przynależności.

Linie w prostokątach

Linie występujące w prostokątach segregują elementy danych na te, które funkcjonują jako klucz główny lub identyfikator, oraz te, które takiej roli nie pełnią. Każdy wiersz tekstu znajdujący się nad taką linią oznacza jeden odrębny element danych, który wybrano do pełnego lub jednoznacznego identyfikowania każdego elementu należącego do takiej encji. W prostokącie powinna się znajdować tylko jedna linia (rysunek 2.27).

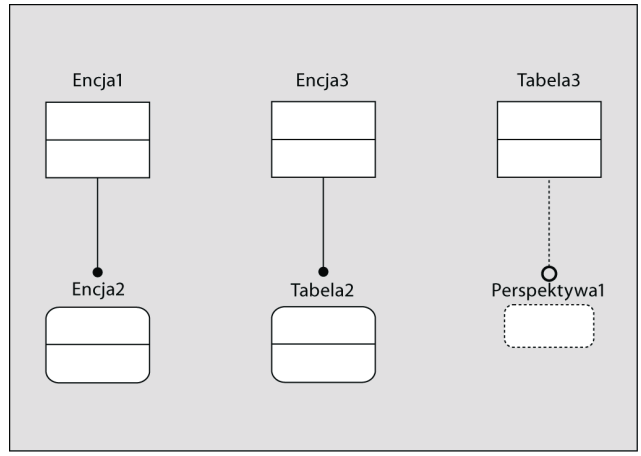
Rysunek 2.27.



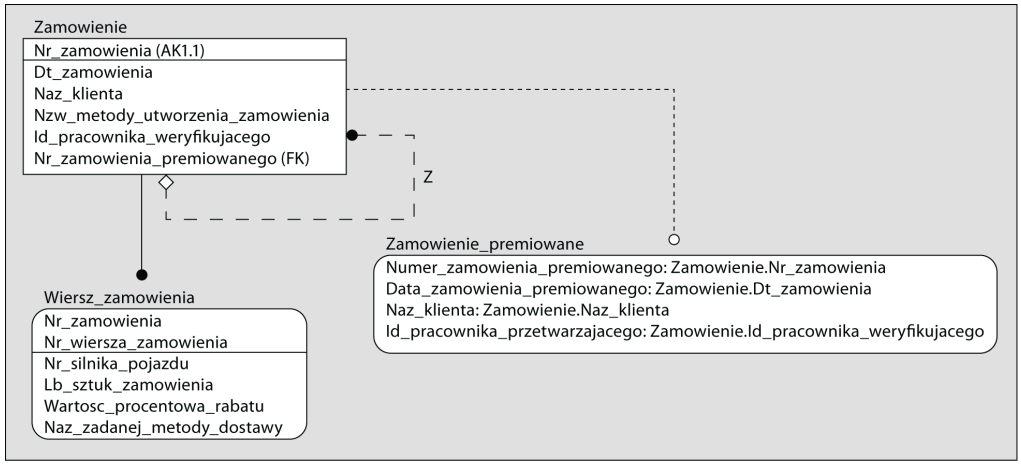
Związki między prostokątami

Jak wcześniej pokazano, związki niejako wiążą ze sobą encje oraz tabele i również są obrazowane w postaci linii. Linie te są opatrzone wyrażeniem czasownikowym, wyjaśniającym, dlaczego dane powiązanie występuje, ale są również rysowane tak, aby zobrazować informacje na temat związków (rysunek 2.28).

Rysunek 2.28.



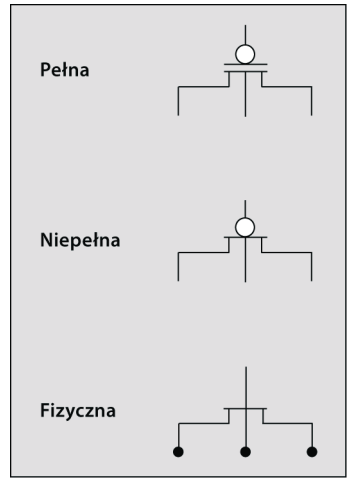
- Linie ciągłe oznaczają, że wszystkie elementy danych znajdujące się nad linią (klucz główny encji lub tabeli nadrzędnej) migrują do pozycji klucza głównego encji (lub tabeli), z którą są związane. Określa się to mianem **związku identyfikującego** (ang. *identifying relationship*). Przyjrzeliśmy się już jednemu z takich związków między encjami `Zamówienie` i `Wiersz zamówienia`. `Wiersz zamówienia` nie może istnieć bez `Zamówienia` (rysunek 2.26).
- Linie przerywane o długich odcinkach oznaczają, że wszystkie elementy danych znajdujące się nad linią encji lub tabeli nadrzędnej migrują do pozycji znajdującej się poniżej linii encji lub tabeli, z którą są związane. Określa się to mianem **związku nieidentyfikującego** (ang. *non-identifying relationship*). Związek między `Zamowieniem` i `Zamowieniem` oznaczający `Zamowienie_premiowane` jest nieidentyfikujący. `Zamowienie_premiowane` mogłoby istnieć bez `Zamowienia` i w celu swojej identyfikacji nie wymaga `Zamowienia`, które inicjowałoby jego utworzenie.
- Ostatnim stylem są linie przerywane o krótkich odcinkach. Jest to notacja dotycząca tylko modeli fizycznych, gdyż oznacza, że perspektywa korzysta z tabeli w celu uzyskania dostępu do danych źródłowych. Perspektywy mogą wykorzystywać dowolne (także wszystkie) kolumny tabeli, do której się odwołują. Przykładem takiej perspektywy jest perspektywa `Zamowienie_premiowane` (rysunek 2.29).



Rysunek 2.29.

Kolejny typ linii, o jakim należy wspomnieć, jest używany wyłącznie w przypadku modeli logicznych w celu oznaczenia grupowania kategorii. Gwoli przypomnienia, w takim przypadku posiadamy jedną encję nadrzędną o tylu encjach kategorii, ile jest wymagane. Linie ciągłe określają, że związek jest identyfikujący — innymi słowy, że wszystkie encje kategorii współużytkują klucz główny encji nadrzędnej. Linie pojedyncze lub podwójne w przecięciu mówią nam o tym, czy uwzględniany jest cały zbiór podziałów kategorii czy nie. Kategoria pełna uwzględnia wszystkie typy, zaś kategoria niepełna uwzględnia tylko niektóre z nich (rysunek 2.30).

Rysunek 2.30.



Notacja fizyczna jest notacją domyślną w przypadku przekształcania modelu logicznego na model fizyczny. Ta sama notacja jest wykorzystywana zarówno w przypadku kategorii pełnych, jak i niepełnych.

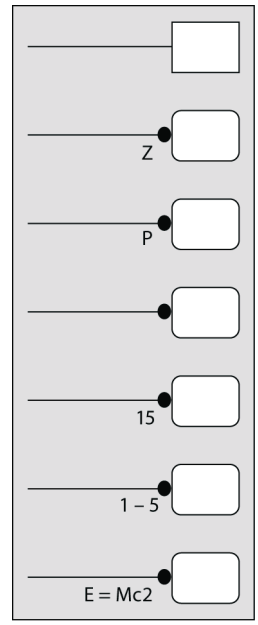
Symbole końcowe

Symbole te występują na końcach linii. Mówią nam one o tym, jaka liczba egzemplarzy w ramach encji lub tabeli może lub musi być związana z egzemplarzem w innej encji lub tabeli.

Symbol końcowy liczności

Liczność określa odpowiedź na pytanie „Ile?". Symbole te dają nam dużą swobodę w zakresie ich definiowania. Pomagają również administratorom baz danych w określeniu oszacowań rozmiaru w świecie fizycznym. Stosowany tu symbol stanowi połączenie znaku kropki oraz zapisu tekstowego (rysunek 2.31).

Rysunek 2.31.



W tabeli 2.3 opisano sposób odczytu tych symboli.

Należy zauważyć, że wymieniona lista zawiera dwie pozycje (na samym dole), które nie są zgodne ze standardem IDEF1X.

Symbol końcowy opcji NULL

Ten symbol określa, czy związek jest wymagany dla wszystkich egzemplarzy czy nie. Odczytuje się go w kierunku od elementu podrzędnego do nadrzędnego i w przypadku związku wymaganego taki odczyt będzie miał postać:

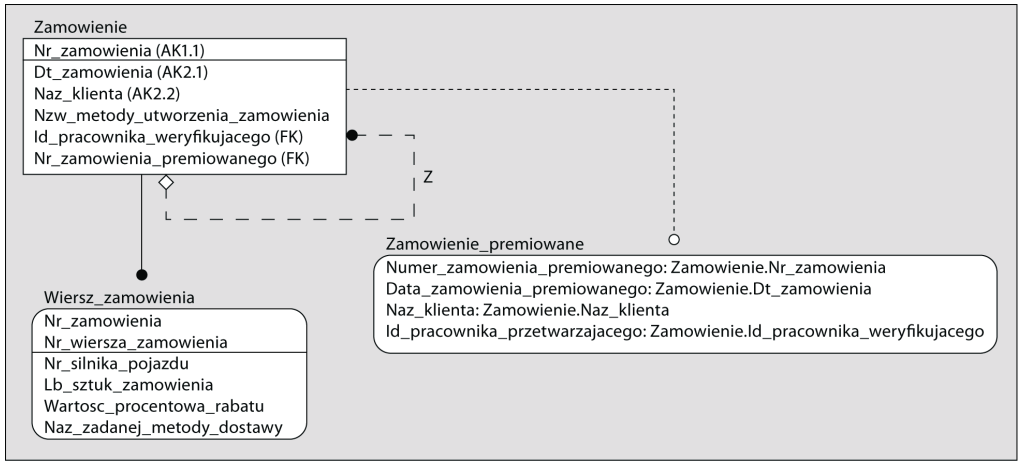
- Każdy egzemplarz <Encji/Tabeli> podrzędnej **musi** być związany z egzemplarzem <Encji/Tabeli> nadrzędnej.

Tabela 2.3.

Symbol końcowy linii związku	Znaczenie	Znaczenie logiczne	Przykład
Linia zwykła	Jeden i tylko jeden	Ogólnie rzecz biorąc, element nadrzędny lub źródłowy.	Jest to symbol końcowy elementu nadrzędnego. Jest używany w przypadku reguł biznesowych dotyczących stosunku elementu podrzędnego do nadrzędnego takich jak „Każdy `Wiersz zamówienia` składa się na jedno i tylko jedno `Zamówienie`”.
Z plus kropka	Zero lub jeden	Decyzja typu prawda-fałsz.	Jest to symbol końcowy elementu podrzędnego. Jest używany w przypadku reguł biznesowych dotyczących stosunku elementu nadrzędnego do podrzędnego takich jak „Każde `Zamówienie` generuje zero lub jedno `Zamówienie`”.
P plus kropka	Jeden lub wiele	Wymagany co najmniej jeden element.	Jest to symbol końcowy elementu podrzędnego. Jest używany w przypadku reguł biznesowych dotyczących stosunku elementu nadrzędnego do podrzędnego takich jak „Każde `Zamówienie` składa się z jednego lub wielu `Wierszy zamówienia`”.
Kropka	Zero, jeden lub wiele	Najbardziej elastyczny przypadek.	Jest to symbol końcowy elementu podrzędnego. Jest używany w przypadku reguł biznesowych dotyczących stosunku elementu nadrzędnego do podrzędnego takich jak „`Zamówienie` składa się z zera, jednego lub wielu `Wierszy zamówienia`”.
<Liczba> plus kropka	Konkretnie <L>	Najbardziej ograniczony przypadek. Liczba (zawsze) równa określonej wartości.	Jest to symbol końcowy elementu podrzędnego. Jest używany w przypadku reguł biznesowych dotyczących stosunku elementu nadrzędnego do podrzędnego takich jak „Każdy `Rok kalendarzowy` składa się z 12 `Miesięcy`”.
<L – L> plus kropka	Zakres wartości od Liczba do Liczba	Reguła zakresu. Wartość musi należeć do podanego przedziału.	Jest to symbol końcowy elementu podrzędnego. Jest używany w przypadku reguł biznesowych dotyczących stosunku elementu nadrzędnego do podrzędnego takich jak „Każdy `Miesiąc kalendarzowy` składa się z 28 do 31 `Dni`”.
<opis> plus kropka	Wymieniony opis związku wielokrotnego	Odpowiedź na bardzo skomplikowane pytania o liczbę, której nie da się wyrazić w inny sposób.	Jest to symbol końcowy elementu podrzędnego. Jest używany w przypadku reguł biznesowych dotyczących stosunku elementu nadrzędnego do podrzędnego takich jak „Każdy `Klient` w wieku poniżej 12 oraz powyżej 65 lat otrzymuje `Rabat na bilet`”.

W przypadku związku dopuszczającego wartości NULL będzie to:

- Każdy egzemplarz <Encji/Tabeli> podrzędnej **może nie** być związany z egzemplarzem <Encji/Tabeli> nadrzędnej. Symbol oznaczający ten ostatni typ związku to niewypełniony romb umieszczony na końcu linii najbliższej elementowi nadrzędnemu. Tak więc, na przykład, w przypadku opisywanego modelu Zamówienia otrzymujemy diagram przedstawiony na rysunku 2.32.



Rysunek 2.32.

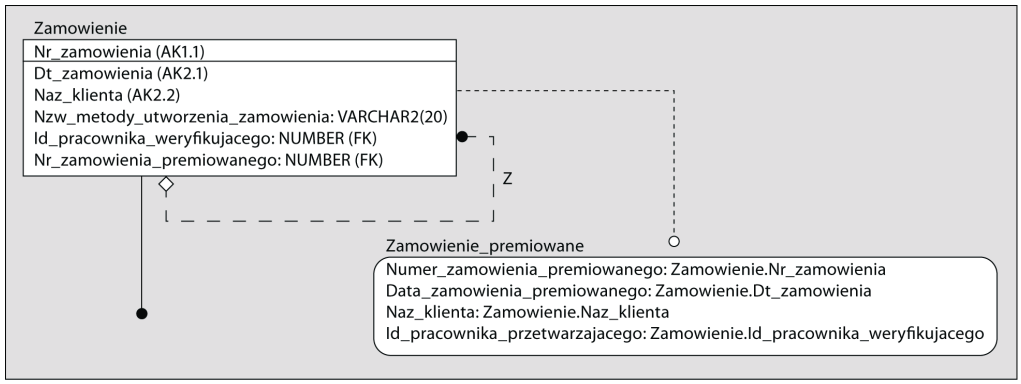
Pojazdy sprzedajemy w ten sposób, że każdy Wiersz_zamowienia **musi** posiadać wartość Numer_silnika_pojazdu, z którym jest związany. Jednak nie każde Zamowienie generuje Zamowienie_premiowane, więc choć może tak być, nie jest to wymagane.

Symbol końcowy perspektywy

Ostatnim typem symbolu końcowego jest linia drobno przerywana i niewypełnione kółko, co oznacza perspektywę (rysunek 2.33). Gwoli przypomnienia, perspektywy występują tylko w modelach fizycznych i zawierają przetworzone lub przefiltrowane dane pochodzące z tabel źródłowych. W całości są one zależne od istnienia oryginalnych obiektów źródłowych. Niewypełnione kółko można traktować jako symbol bytu niematerialnego, gdyż perspektywy modeluje się z wielu różnych powodów, ale same w sobie nie mają one charakteru materialnego. Są obrazowane jako puste elementy końcowe, gdyż zasadą tej notacji jest ukazywanie tylko zależności między perspektywą a tabelą (tabelami) źródłową (źródłowymi). Można także spotkać perspektywy, które nie posiadają żadnych związków, jeżeli odwołują się do tabel nieujętych w modelu.

Diagramy związków encji (ER, diagramy Chena)

Standard ER (od ang. *Entity-Relationship*) zyskał ogromną popularność, szczególnie w zakresie modelowania koncepcyjnego, i jest jednym z najstarszych stylów modelowania. Został opracowany przez doktora Petera P. Chena w 1976 roku. W ostatnich



Rysunek 2.33.

latach w przypadku niektórych narzędzi służących do modelowania zarzucono obsługę tego standardu. Charakteryzuje się on wieloma podobieństwami do innych omawianych tu stylów:

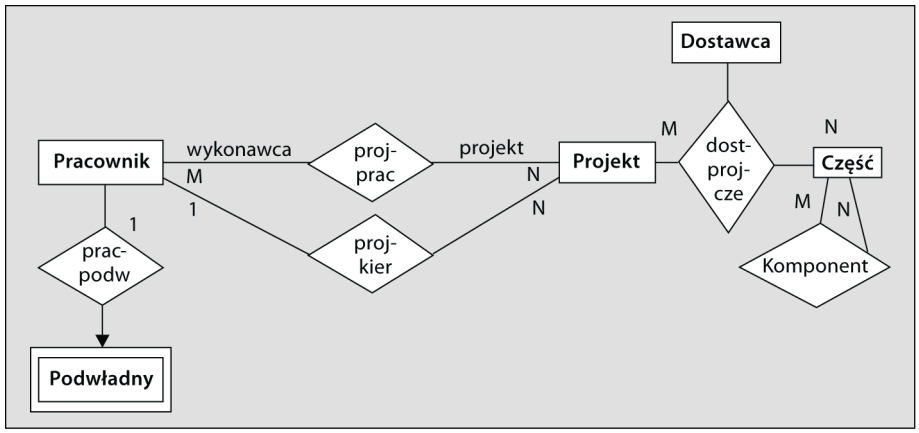
- Encje są obrazowane jako prostokąty z ostrymi rogami.
- Związki łączą prostokąty za pomocą linii.
- Liczność jest określana za pomocą litery lub kodu liczbowego.
- Nazwa encji jest dołączana do prostokąta.

Diagramy ER charakteryzują się też unikatowymi cechami:

- Prostokąty z ramką rysowaną podwójną linią oznaczają encje o typie słabym, czyli takie, których klucz jest częściowo uzupełniany danymi poprzez związek identyfikujący z inną encją i nie mogą one istnieć samodzielnie.
- Atrybuty nie są obrazowane.
- Związki są rzeczownikami, a nie wyrażeniami czasownikowymi.
- Reguły powinny być odczytywane w określonym kierunku (od strony lewej do prawej, z góry na dół).

Poniżej przedstawiono przykład diagramu Chena oraz pewne wskazania na informacje, które obrazuje (rysunek 2.34).

- `Pracownik` może wykonywać pracę projektową (`proj-praca`) na wielu (M) `Projektach`.
- `Pracownik` może być kierownikiem projektu (`proj-kier`) tylko jednego (1) `Projektu`.
- `Podwładny` jest podwładnym (`prac-podw`) tylko jednego (1) `Pracownika`. Ze względu na fakt, że `Podwładny` to encja typu słabego, jej każdy egzemplarz musi posiadać egzemplarz `Pracownika`, który go identyfikuje.
- `Komponent` może być `Częścią` wielu (M) komponentów.



Rysunek 2.34.

Standard Chena (ER) wciąż bywa używany i bardzo dobrze sprawdza się w zakresie analiz podsumowujących oraz koncepcyjnych.

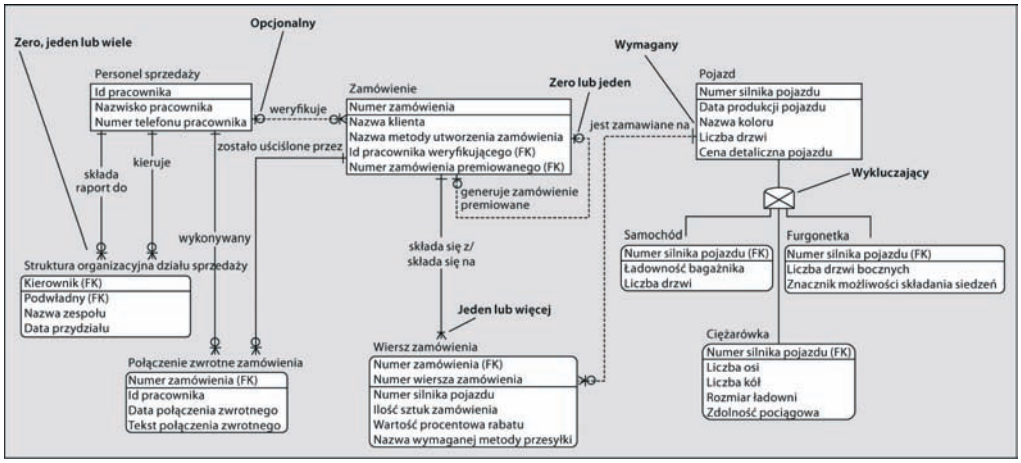
Standard Information Engineering (I/E)

Standard I/E, opracowany przez Clive'a Finkelsteina i Jamesa Martina w 1976 roku, jest dziś bardzo popularny i stanowi zwykle jeden z możliwych do wyboru stylów w przypadku popularnych programów wspomagających modelowanie danych.

- Są tu wykorzystywane konwencje z prostokątami o rogach ostrych i zaokrąglonych, tak jak miało to miejsce w przypadku encji standardu IDEF1X. Prostokąty z rogami ostrymi oznaczają encje niezależne, zaś z rogami zaokrąglonymi — encje zależne.
- Atrybuty są obrazowane. Prostokąt encji jest dzielony w celu ukazania różnicy między kluczem prywatnym a innymi atrybutami. Istnieje również możliwość wybrania widoku nieprezentującego atrybutów.
- Związki są liniami między prostokątami i zawierają na końcach symbol „kurzej stopki", która określa liczność.

Cechą charakterystyczną standardu I/E są symbole końcowe linii związków. Standard I/E wykorzystuje znany symbol „kurzej stopki" w celu oznaczenia wielokrotności, co pozwala obrazować wiele różnych kombinacji liczności i opcjonalności elementów nadrzędnych i podrzędnych. Interesującym faktem jest ten, że w rzeczywistości występują cztery różne wersje standardu I/E, które cechują subtelne różnice składniowe. Niektóre z nich dopuszczają stosowanie notacji dla związków wykluczających, a niektóre nie. Należy bliżej zapoznać się z wersją obsługiwaną przez używane oprogramowanie.

Na rysunku 2.35 przedstawiono przykład diagramu I/E. Jak widać, stosowana notacja w dużej mierze pokrywa się ze stylem IDEF1X. Są one do siebie bardzo podobne poza związkami kategorii. Tam, gdzie notacja IDEF1X przedstawia kategorie zupełne i niezupełne, notacja I/E przedstawia je jako wykluczające (gdzie jeden egzemplarz może



Rysunek 2.35.

zostać przydzielony tylko do jednego podtypu) oraz zawierające (gdzie egzemplarz może zostać określony jako więcej niż jeden podtyp).

Notacja Barkera

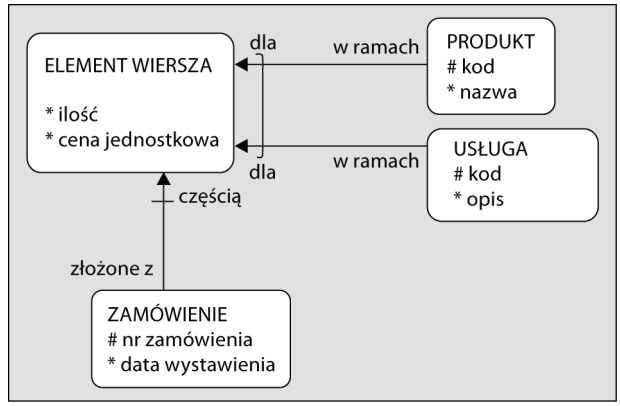
Notacja Richarda Barkera, opracowana w 1990 roku, jest wykorzystywana w narzędziach CASE firmy Oracle.

- Prostokąty encji zawsze posiadają zaokrąglone rogi.
- Przerywana linia związku oznacza opcjonalność — nie jest wykorzystywany symbol końcowy.
- Atrybuty są obrazowane, ale nie występuje linia oddzielająca klucz główny. Oznacza się go znakiem #.
- Związki są ograniczone do postaci binarnej (prawda lub fałsz).
- Wykluczanie (więzy alternatywy) oznacza się za pomocą symbolu łuku obejmującego linie związków.
- Związki kategorii i podtypu-nadtypu obrazuje się poprzez zagnieżdżanie prostokątów.

Poniżej przedstawiono adekwatny przykład oraz kilka reguł biznesowych, które można na jego podstawie odczytać (rysunek 2.36).

- Każdy `ELEMENT WIERSZA` dotyczy albo jednego `PRODUKTU`, albo jednej `USŁUGI`.
- Każdy `PRODUKT` występuje w wielu `ELEMENTACH WIERSZA`.
- `kod` jest identyfikatorem głównym `PRODUKTU`.
- `nazwa` jest atrybutem opcjonalnym `PRODUKTU`.

Rysunek 2.36.



- Każde ZAMÓWIENIE składa się z wielu ELEMENTÓW WIERSZA.
- Każdy ELEMENT WIERSZA jest przynajmniej częściowo identyfikowany przez ZAMÓWIENIE.
- Każdy ELEMENT WIERSZA jest częścią ZAMÓWIENIA.

Powyższy diagram pochodzi ze strony *http://www.inconcept.com/JCM/December2000/halpin.html*, gdzie można znaleźć więcej informacji na temat notacji Barkera.

Podsumowanie

W niniejszym rozdziale przyjrzeliśmy się różnicom występującym między systemami zarządzania relacyjnymi bazami danych (ang. *Relational Database Management Systems — RDBMS*) oraz systemami sieciowymi i hierarchicznymi w zakresie wykorzystywania mechanizmu dopasowywania pól danych (kluczy obcych) w celu tworzenia związków między tabelami zamiast wykorzystywania wskaźników danych stosowanych w innych systemach. Pojęcie to, wraz z pojęciem pojedynczego egzemplarza każdego elementu każdego zbioru, stanowi podstawę relacyjnego modelowania danych.

Przedstawiliśmy także podstawową terminologię z zakresu modelowania relacyjnego oraz pojęcia, których używamy na co dzień w celu tworzenia modeli koncepcyjnych, logicznych i fizycznych. Omówiliśmy:

- encje (niezależne, zależne),
- atrybuty (pojedyncze, grupowe),
- klucze (kandydujące, główne, obce),
- związki (identyfikujące, nieidentyfikujące, kategorii, perspektywy),
- symbole końcowe związków (liczności, opcjonalności),
- tabele i perspektywy (zbiory danych klienta, operacyjne zbiory danych),

- kolumny (elementy danych klienta, elementy danych operacyjnych),
- klucze (główne, sztuczne, alternatywne, obce),
- więzy.

Wreszcie przyjrzeliśmy się pewnym alternatywnym metodom graficznej reprezentacji modeli danych, skupiając się na notacji IDEF1X, z której będziemy korzystać we wszystkich modelach prezentowanych w książce.