Postacie normalne

W odróżnieniu od schematu procesu projektowania bazy danych „z góry do dołu” (ang. *top – down* – od ogółu do szczegółów), normalizacja jest uznawana niekiedy za odrębną metodologię projektowania typu „z dołu do góry” (ang. *bottom – up*, tzn. od szczegółów do uogólnień). W swojej pracy na temat relacyjnego modelu danych E.F.Codd sformułował reguły projektowania relacyjnych baz danych. Reguły te zostały pierwotnie nazwane *postaciami normalnymi*. Codd opisał trzy postacie normalne oznaczane często symbolami 1NF, 2NF, 3NF. Proces kolejnego przekształcania projektu bazy danych przez te trzy postacie normalne jest znany jako *normalizacja* bazy danych.

W połowie lat siedemdziesiątych spostrzeżono pewne niedostatki w trzeciej postaci normalnej Codd’a i zdefiniowano mocniejszą postać normalną, znaną jako postać normalna Boyce'a-Codda. Później Fagin przedstawił czwartą postać normalną, i piątą postać normalną.

Powodem wprowadzenia kolejnych postaci normalnych była chęć pełnej optymalizacji bazy. Baza jest tym „lepsza”, im jest w wyższej postaci normalnej. Zazwyczaj jednak wystarczająca jest normalizacja do trzeciej postaci normalnej.

W modelu implementacyjnym relacja ma postać tabeli. W zależności od organizacji SZBD wszystkie relacje (tabele) bazy wraz z danymi organizacyjnymi niezbędnymi do prawidłowego przetwarzania bazy przez SZBD przechowuje się albo w jednym pliku w pamięci zewnętrznej, albo dla każdej tablicy przeznacza się odrębny plik w pamięci zewnętrznej.

Normalizacja, czyli proces sprowadzania bazy do odpowiedniej postaci polega przede wszystkim na dzieleniu tabeli na kilka połączonych kluczem tabel. Głównym powodem, dla którego normalizuje się bazę jest występowanie problemów (zwanych dalej anomaliami) w przypadku źle zaprojektowanej struktury.

Proces normalizacji musi posiadać trzy własności:
- żaden atrybut nie zostanie zagubiony w trakcie procesu normalizacji,
- dekompozycja relacji nie prowadzi do utraty informacji,
- wszystkie zależności funkcyjne są reprezentowane w pojedynczych schematach relacji.

Wyróżniamy klucze proste i złożone. Jeżeli zbiór identyfikacyjny jest zbiorem jednoelementowym to tworzy klucz prosty, w przeciwnym wypadku klucz jest kluczem złożonym. W relacji możemy wyróżnić wiele kluczy, które nazywamy kluczami potencjalnymi. Wybrany spośród nich klucz nazywamy kluczem głównym (pierwotnym) (*primary key*), pozostałe kluczami drugorzędnymi (*secondary key*).
Atrybuty relacji dzielimy na dwie grupy:
- atrybuty podstawowe: atrybuty należące do klucza schematu relacji,
- atrybuty wtórne: atrybuty nie należące do żadnego klucza schematu.

Problemy te można zilustrować na następującym, prostym przykładzie: Przypuśćmy, że dla bazy danych dotyczących książek w bibliotece zaproponowano strukturę złożoną z jednej relacji:
R(TYTUŁ-KSIĄŻKI, AUTOR, POŻYCZAJĄCY, ADRES, DATA-WYPOŻYCZENIA)
W tak zaprojektowanej bazie danych mogą wystąpić anomalia:
przy aktualizacji - jeżeli wypożyczający zmienił adres, trzeba przeszukać całą bazę i we wszystkich komórkach, w których występuje należy zmienić ten adres,
przy usuwaniu - jeżeli pożyczający zwróci ostatnią książkę, zostanie utracona informacja na jego temat,
przy wstawianiu - gdy pożyczający chce zapisać się do biblioteki, należy go wpisać do tablicy, ale jednocześnie istnieje wymaganie, aby podana została książka, którą wypożycza - nowy użytkownik wcale nie musi pożyczać książki,

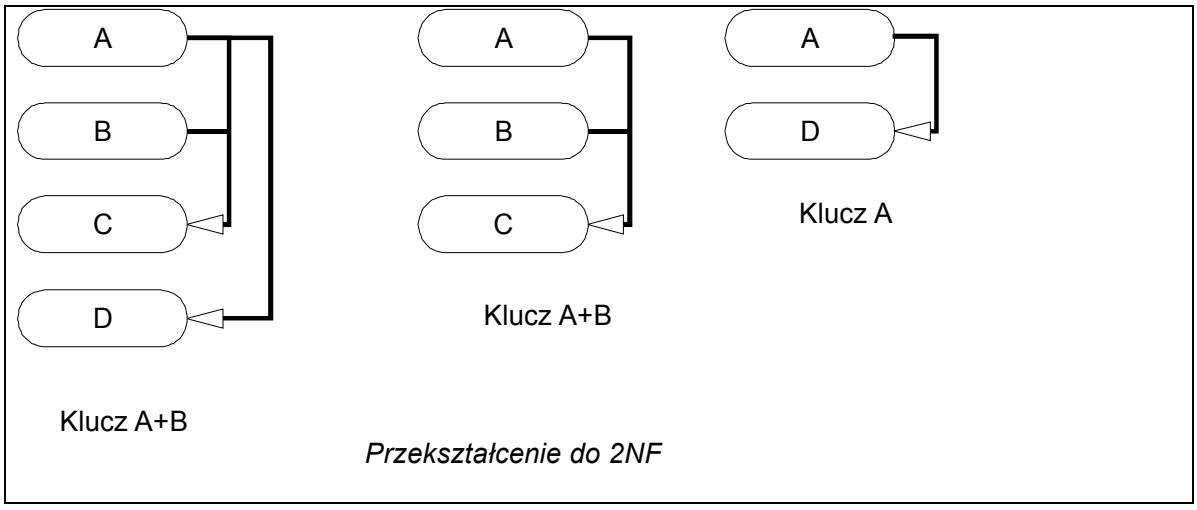
redundancja czyli powtarzanie tej samej informacji w kilku miejscach w bazie; powoduje to niepotrzebne zajmowanie pamięci przez tą samą informację. W przypadku, gdy jedna osoba pożycz dwie lub więcej książek niepotrzebnie powtarzana jest informacja na temat adresu czytelnika.

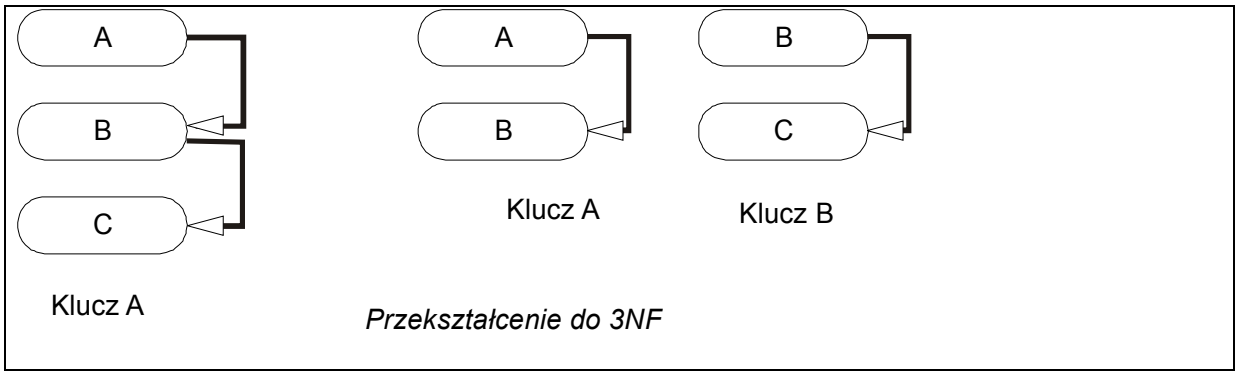
W poniższej tabeli zestawiono charakterystyki poszczególnych postaci normalnych bazy.

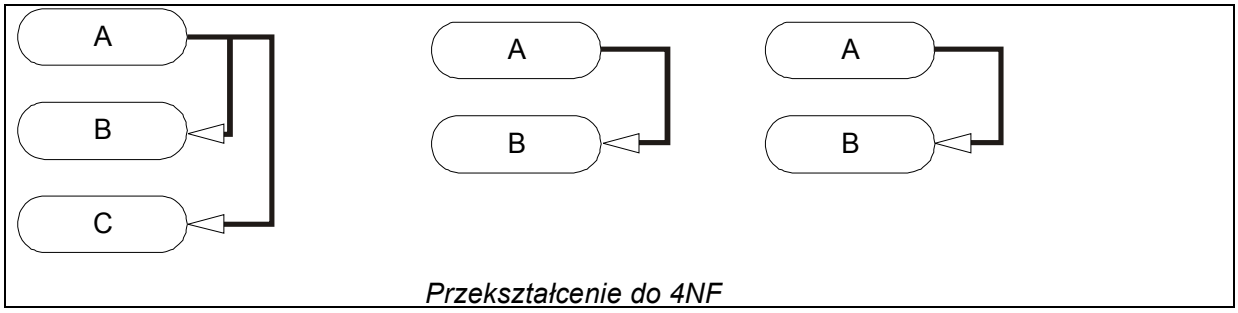
NF	Opis
1NF	W każdej komórce tabeli znajduje się tylko jedna wartość, inaczej: każdy atrybut niekluczowy (nie należący do żadnego klucza) jest funkcjonalnie zależny od klucza głównego. Schemat relacji R znajduje się w 1NF, jeżeli wartości atrybutów są atomowe.
2NF	Baza jest w 2NF, jeżeli jest w pierwszej postaci normalnej oraz kolumna nie należąca do klucza nie zależy od części klucza głównego - klucza wybranego przez projektanta (w ten sposób usuwa się niepełne zależności funkcjonalne), inaczej: każdy atrybut niekluczowy jest w pełni funkcyjnie zależny od klucza głównego. Aby stwierdzić, czy tabela jest w 2NF trzeba wyznaczyć funkcyjne zależności i klucz główny. Rozważając klucz główny i zależności funkcyjne w danej tabeli możemy stwierdzić czy kolumny zależą od całego klucza głównego czy jego części. Zbiór atrybutów Y jest w *pełni funkcyjnie zależny* od zbioru atrybutów X w schemacie R, jeżeli $X \rightarrow Y$ i nie istnieje podzbiór $X' \subset X$ taki, że $X' \rightarrow Y$. Zbiór atrybutów Y jest *częściowo funkcyjnie zależny* od zbioru atrybutów X w schemacie R, jeżeli $X \rightarrow Y$ i istnieje podzbiór $X' \subset X$ taki, że $X' \rightarrow Y$. Dana relacja r o schemacie R jest w 2NF, jeżeli żaden atrybut wtórny tej relacji nie jest częściowo funkcyjnie zależny od żadnego z kluczy relacji r. Dana relacja r o schemacie R jest w *drugiej postaci normalnej*, jeżeli każdy atrybut wtórny tej relacji jest w pełni funkcyjnie zależny od klucza podstawowego relacji r.
3NF	Baza jest w 3NF, jeżeli jest w 2NF oraz kolumna nie należąca do klucza nie zależy od innej kolumny nie należącej do klucza (w ten sposób usuwa się częściowe zależności funkcjonalne), inaczej: każdy niekluczowy atrybut jest bezpośrednio zależny od klucza głównego. Zbiór atrybutów Y jest *przechodnio funkcyjnie zależny* od zbioru atrybutów X w schemacie R, jeżeli $X \rightarrow Y$ i istnieje zbiór atrybutów Z, nie będący podzbiorem żadnego klucza schematu R taki, że zachodzi $X \rightarrow Z$ i $Z \rightarrow Y$. Zależność funkcyjna $X \rightarrow Y$ jest *zależnością przechodnią* jeżeli istnieje podzbiór atrybutów taki, że zachodzi $X \rightarrow Z$, $Z \rightarrow Y$ i nie zachodzi $Z \rightarrow X$ lub $Y \rightarrow Z$. Dana relacja r o schemacie R jest w 3NF, jeżeli dla każdej zależności funkcyjnej $X \rightarrow A$ w R jest spełniony jeden z następujących warunków: • X jest nadkluczem schematu R, lub • A jest atrybutem podstawowym schematu R. Dana relacja r o schemacie R jest w trzeciej postaci normalnej, jeżeli jest w drugiej postaci normalnej i żaden atrybut wtórny nie jest przechodnio zależny od podstawowego klucza schematu relacji R.
4NF	Baza jest w 4NF, jeśli jest w trzeciej 3NF oraz usunięto wielokrotne, wielowartościowe zależności funkcjonalne. *Zależności wielowartościowe* są konsekwencją wymagań 1NF, która nie dopuszcza, aby krotki zawierały atrybuty wielowartościowe. Wystąpienie zależności wielowartościowej $X \rightarrow\rightarrow Y$ w relacji o schemacie $R=XYZ$ wyraża dwa fakty:

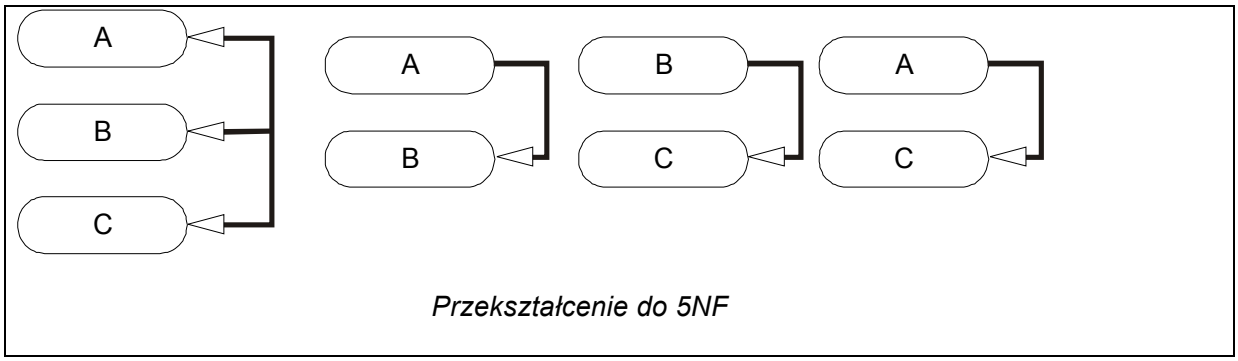
	• związek pomiędzy zbiorami atrybutów X i Y; • niezależność zbiorów atrybutów Y i Z – zbiory te są związane ze sobą pośrednio przez zbiór atrybutów X. Zależność wielowartościowa $X \rightarrow\rightarrow Y$ w relacji r (R) nazywamy *zależnością trywialną*, jeżeli: • zbiór Y jest podzbiorem X, lub $X \cup Y = R$ Relacja r o schemacie R jest w *czwartej postaci normalnej (4NF)* względem zbioru zależności wielowartościowych MVD, jeżeli jest ona w trzeciej postaci normalnej i dla każdej zależności wielowartościowej $X \rightarrow\rightarrow Y \in$ MVD zależność ta jest trywialna lub X jest nadkluczem schematu R.
5NF	Baza jest w 5NF, jeżeli jest w 4NF oraz usunięto zależności funkcjonalne, które nie wynikają z zależności od atrybutów klucza (tabele zostały podzielone na najmniejsze możliwe kawałki w celu eliminacji redundancji w tabeli). Niech R $\{R_1, R_2, ..., R_p\}$ oznacza zbiór schematów relacji zdefiniowany na zbiorem atrybutów $U = \{A_1, A_2, ..., A_n\}$ takich, że $R_1 \cup R_2 \cup ... \cup R_p = U$. Mówimy, że relacja $r(U)$ *spełnia zależność połączeniową,* oznaczaną przez JD $[R_1, R_2, ..., R_p]$, jeżeli można ją zdekomponować bez utraty informacji na podrelacje $r_1(R_1)$, $r_2(R_2)$ ..., $r_p(R_p)$. Zależność połączeniowa JD$[R_1, R_2, ..., R_p]$ jest *trywialna*, jeżeli: jeden ze schematów R_i, i= 1, 2, ..., p jest równy R. Schemat relacji R jest w 5NF, jeżeli dla każdej zależności połączeniowej JD w schemacie R zachodzi: • zależność jest trywialna, • każdy podschemat R_i i=1, 2, ..., p jest nadkluczem schematu R.

Schemat procesu normalizacji bazy danych









Postać normalna Boyce's-Codda

Dla każdej zależności : $X \to Y \in F^+$ ($X \subseteq R$, $A \in R$) zachodzi albo

1. $A \in X$ (zależność trywialna)
2. X jest kluczem.

Przykład:

1. $F=\{X \to R\}$ dla $X \subseteq R$ (jest w postaci normalnej)
2. $F=\{X_1 \to R; \ldots ; X_k \to R\}$ dla $X_1, \ldots, X_k \subseteq R$ (jest w postaci normalnej)
3. $F=\{MU \to K; K \to M\}$ dla $R=\{M, U, K\}$ (miasto,ulica,kod),

Są dwa klucze MU i KU. Ze względu na zależność $K \to M$ schemat nie jest w postaci normalnej Boyce'a-Codda. Schematu nie da się rozłożyć z zachowaniem funkcyjnych zależności.

Trzecia postać normalna

Dla każdej zależności $X \to Y \in F^+$ ($X \subseteq R$, $A \in R$) zachodzi albo

1. $A \in X$ (zależność trywialna)

2. X jest nadkluczem, albo
3. A jest atrybutem głównym.

Są dwa rodzaje zależności naruszające trzecią postać normalną.
Niech $A \in X$, A atrybut niegłówny oraz $X \to A \in F^{+}$.

- Zależność funkcyjna $X \to A$ nazywa się zależnością częściową, jeśli X jest właściwym podzbiorem pewnego klucza
- Zależność funkcyjna $X \to A$ nazywa się zależnością przechodnią, jeśli X nie jest ani podzbiorem ani nadzbiorem żadnego klucza ($K \to X \to A$)

W związku z tym wyróżnia się dwa typy „złych" schematów relacji. Dla pierwszego, „gorszego" nazywanego schematem relacji w pierwszej postaci normalnej – nie ma żadnych ograniczeń na zależności funkcyjne. Drugi typ wprowadza ograniczenia. Schemat R ze zbiorem zależności F jest w drugiej postaci normalnej, jeśli nie zawiera zależności częściowych, tzn. żadna zależność częściowa nie wynika z logicznie ze zbioru zależności F.

Własność:
Schemat R ze zbiorem zależności F jest w trzeciej postaci normalnej, jeśli jest w drugiej postaci normalnej oraz nie zawiera zależności przechodnich.

12 praw CODD’a dla relacyjnych baz danych

Model relacyjnej bazy danych został opracowany na początku lat siedemdziesiątych przez Amerykanina E.F. Codd'a, który przy jego tworzeniu oparł się na matematycznej teorii relacji i zbiorów. Dr E. F. Codd, twórca modelu relacyjnego opublikował dwuczęściowy artykuł w *ComputerWorld* (Codd, 1985), w którym podał 12 praw określania, kiedy baza danych jest relacyjna. Dwanaście Reguł Codd'a brzmi następująco:

1. **Reguła Informacyjna**. Wszystka informacja w relacyjnej bazie danych jest reprezentowana wprost i tylko w jeden sposób, jako wartości w tabelach.
2. **Reguła Gwarantowanego Dostępu**. Każda i wszystkie dane w relacyjnej bazie danych są logicznie dostępne poprzez nazwę tabeli, wartość klucza pierwotnego i nazwę kolumny.
3. **Reguła Systematycznego Traktowania Wartości Pustych**. Wartości puste (różne od pustego łańcucha znaków, łańcucha spacji i różne od zera lub innej liczby) są całkowicie wpierane przez relacyjny system zarządzania bazą danych dla reprezentacji braku informacji w sposób systematyczny (konsekwentny) niezależnie od typu danej.
4. **Reguła Organizacji Dostępu w Modelu Relacyjnym**. Opis bazy danych jest przedstawiany na poziomie logicznym w ten sam sposób jak dane, tak że upoważniony użytkownik może zastosować ten sam język zapytań tak w celu poznania opisu bazy jak i danych.
5. **Reguła Pełności Danych Podjęzyka**. System relacyjny może wspierać wiele języków, jednakże musi istnieć przynajmniej jeden, którego instrukcje tworzą wyrażenia dla dobrze zdefiniowanej składni w postaci łańcuchów znaków i są zdolne do pełnego wspierania następujących elementów: definicji danych, definicji perspektyw, manipulacji danymi (interaktywnie lub przez program), więzów integralności danych i zakresów transakcji (begin, commit i rollback).
6. **Reguła Przeglądania Modyfikacji**. Wszystkie modyfikacje działające na perspektywach muszą wykonywalne przez System Zarządzania Bazą Danych.
7. **Reguła Wysokiego Poziomu Wstawiania, Aktualizacji i Usuwania**. System musi wspierać zespół jednoczesnych działań takich jak wstawianie, aktualizacja i usuwanie danych.
8. **Reguła Fizycznej Niezależności Danych**. Programy aplikacyjne lub akcje wykonywane na terminalu pozostają logicznie nienaruszone w przypadku zmian dokonywanych w reprezentacji pamięci fizycznej lub metod dostępu.
9. **Reguła Logicznej Niezależności Danych**. Modyfikacje w logicznej strukturze bazy danych mogą być wykonywane bez wyrejestrowywania się istniejących użytkowników czy zamykania istniejących programów.
10. **Reguła Niezależności Integralności**. Ograniczenia integralności specyficzne dla konkretnej relacyjnej bazy danych muszą być definiowalne w podjęzyku relacyjnym i przechowywane w schemacie bazy a nie w programie aplikacyjnym. Minimum dwa ograniczenia integralności muszą być wpierane:
    - integralność encji: żaden z elementów składowych klucza pierwotnego nie może zawierać wartości pustej,
    - integralność referencyjna: dla każdej różnej, nie pustej wartości klucza obcego musi odpowiadać odpowiednia wartość klucza pierwotnego z tej samej domeny

11. **Reguła Niezależności Dystrybucji**. Niezależność dystrybucji wymusza, że użytkownicy nie powinni martwić się, kiedy baza danych jest dystrybuowana
12. **Reguła Braku Podwersji**. Dostęp na niskim poziomie albo na poziomie rekordu nie może być zdolny do naruszenia systemu, ominięcia reguł integralności lub ograniczeń zdefiniowanych na wyższych poziomach

Do 12 reguł istnieje uzupełnienie znane jako **Reguła zero:** Dla każdego systemu, który uważany jest za relacyjny musi istnieć możliwość zarządzania danymi wyłącznie poprzez jego relacyjne możliwości

Przykład projektowania bazy danych

Na początek parę słów o terminologii. W zależności od kontekstu lub też autora opracowania te same rzeczy nazywane są innymi terminami. Poniższa tabela przedstawia częściowe zestawienie odpowiadających sobie terminów. Uwagi na temat różnej interpretacji pojęcia encji zawarte są w podrozdziale: „Określenie encji".

Teoria relacyjna	**Model ER**	**Relacyjne b.d.**	**Aplikacje**
Relacja	Encja	Tabela	—
Krotka	Instancja	Wiersz	Rekord
Atrybut	Atrybut	Kolumna	Pole
Dziedzina	Dziedzina / Typ	Dziedzina / Typ	—
Schemat relacji	—	Struktura tabeli	—

Ważnym punktem przed przystąpieniem do projektowania aplikacji wykorzystującej bazy danych jest zapewnienie jej przenoszalności. Przenośności aplikacji jest to nie tylko możliwość przeniesienia na inną platformę sprzętową, ale także możliwość pracy z innym relacyjnym systemem bazodanowym. Jeżeli mówimy o możliwości pracy aplikacji z innym relacyjnym systemem bazodanowym to mamy na myśli system, który obsługuje SQL zgodnie z międzynarodowym standardem. Niestety, często w praktyce w różnych SZBD istnieją odstępstwa od standardu. Wynika to z pozostałości historycznych, niedoskonałości standardu i rozszerzaniem możliwości języka wymuszone żądaniami twórców aplikacji i konkurencją.

Zbieranie informacji

Tutaj robi się wywiad o rozwiązywanym problemie

Projekt logiczny

Jego realizacja składa się z kilku kroków (omawiany jest projekt oparty o diagramy związków encji).

Określenie encji

Encja (z ang. *entity* – jednostka) jest odzwierciedleniem rzeczywistego obiektu, o którym informacje należałoby przechowywać w bazie danych. Rozróżnianie encji jest możliwe dzięki temu, że ich odpowiedniki - rzeczywiste obiekty, mają tożsamość.
Encje o tych samych własnościach tworzą typy (zbiory) encji. Termin encja bywa często używany zarówno w znaczeniu „typ encji", jak i „instancja encji" (czyli reprezentant konkretnego obiektu).
Encje i typy encji są jednoznacznie określane przez nadanie im unikalnych nazw.
Atrybut encji danego typu jest to jej własność, reprezentowana przez pewną wartość (liczbę, tekst, ...).
Tabela - obiekt bazy danych, który jest odpowiednikiem encji – obiektu modelu baz danych (w tym miejscu encja rozumiana jest jako typ encji (zbiór encji)).

Spróbujmy zdefiniować encje główne (typy encji) w problemie archiwizowania danych związanych z wykonywaniem zamówień pewnych produktów przez klientów realizowanych w pewnej firmie. W wyniku analizy problemu wyróżnione zostały następujące encje (typy encji) wraz z parametrami:

Klienci i potencjalni klienci
Nazwa
Adres
Numer telefonu

Zamówienia
Zamówione towary
Data zamówienia
Informacja o przesyłce

Dane produktu
Opis
Cena zakupu
Cena sprzedaży
Kody paskowe

		Stan magazynu

Można dodać dokładniejszy opis parametrów encji

Dane produktu	Szczegóły
Opis	Tekst zawierający informacje o cechach fizycznych (np. długość), składający się z co najmniej 70 znaków.
Cena zakupu	Cena, jaką zapłacono dostawcy, bez kosztów i podatków
Cena sprzedaży	Cena dla klienta, bez kosztów i podatków
Kody paskowe	Kod w standardzie EAN13
Stan magazynu	Ilość towaru dostępna w sprzedaży, w tym wszystkie korekty wprowadzone w czasie inwentaryzacji

Konwersja encji na tabele

Etap ten zbliża projekt do implementacji.
Opisowe nazwy tabel zamieniane są na rzeczowniki w liczbie pojedynczej (najlepiej, jeśli jest to jedno słowo).
Opisowe nazwy parametrów zamieniane są na kolumny tabel, w których przechowywane będą wartości pojedynczych atrybutów (1NF).

klient
tytuł
nazwisko
imie
adresudm
miasto
kod
telefon

zamówienie
towary zamówione
ilość każdego towaru
data magazynowania
data dostarczenia
informacje spedycyjne

produkt
opis
cena zakupu
cena sprzedaży
kody paskowe
stan magazynu

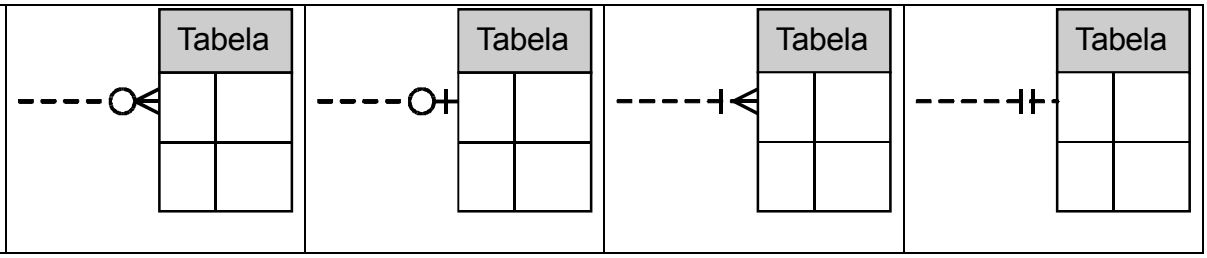
Określenie związków i liczebności

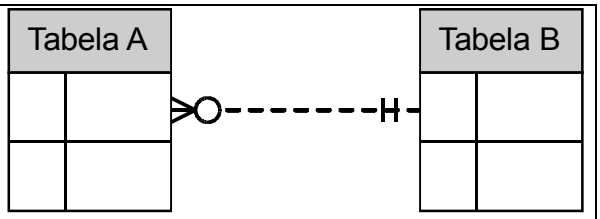
Chodzi o wyróżnienie tych atrybutów, pomiędzy którymi występują zależności funkcyjne oraz zdefiniowanie związków pomiędzy encjami oraz liczebności tych związków. Powstaje tzw. model kocepcyjny, który może być zamieniony w model relacyjny.

Rysowanie diagramów związków encji

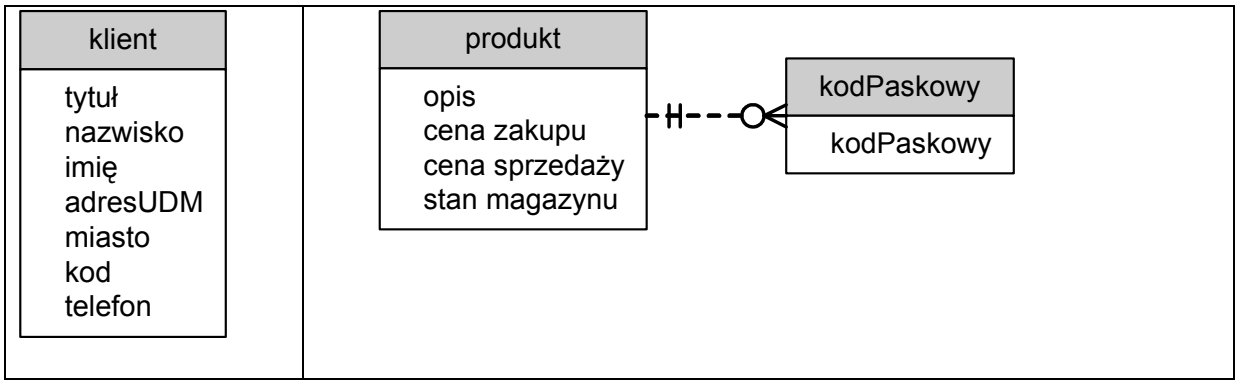
Jedną z metod rysowania diagramów związków encji jest metoda, w której za pomocą prostokątów rysowane są od razu typy (zbiory) encji wraz ze związkami pomiędzy typami (jak niżej). Ten sposób projektowania bliski jest modelowi relacyjnemu (z tabelami). Jednak w takim podejściu czasem trudno jest wychwycić niuanse, które ujawniają się dopiero na diagramach „bardziej pierwotnych", gdzie zbiory encji rysowane są za pomocą owali, w których wyróżnia się pojedyncze egzemplarze encji i ich indywidualne związki (taki model koncepcyjny jest zamieniany później na np. model relacyjny).

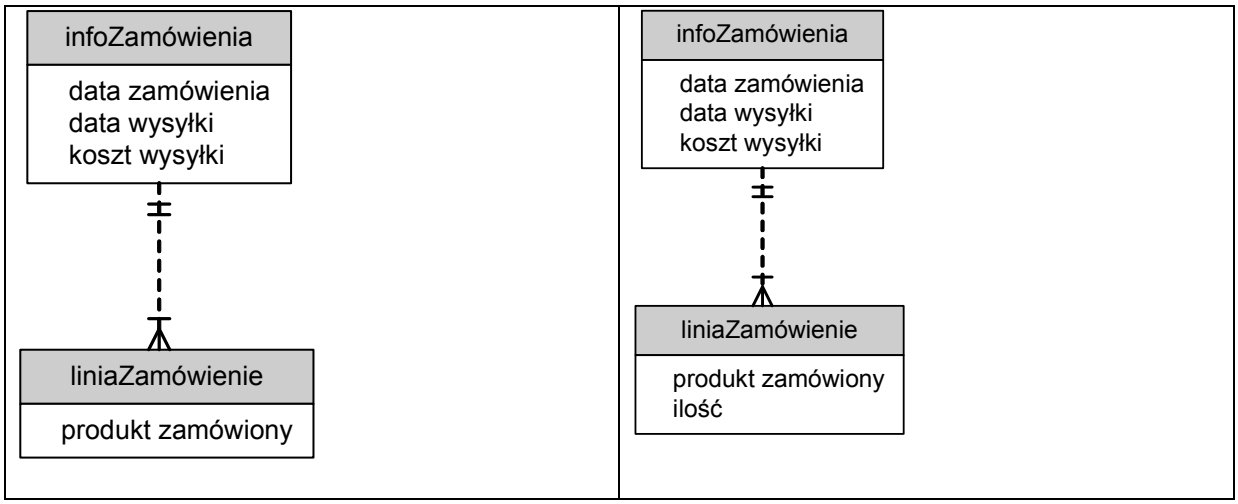
związek	zero lub wiele	zero lub jeden	jeden lub wiele	dokładnie jeden

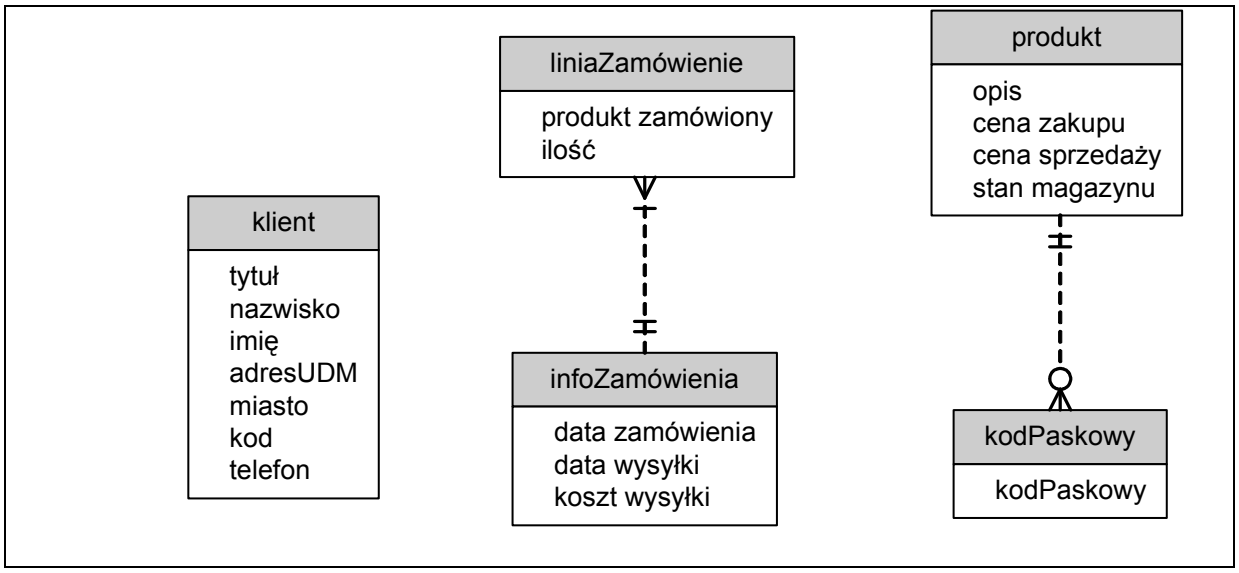
symbol	

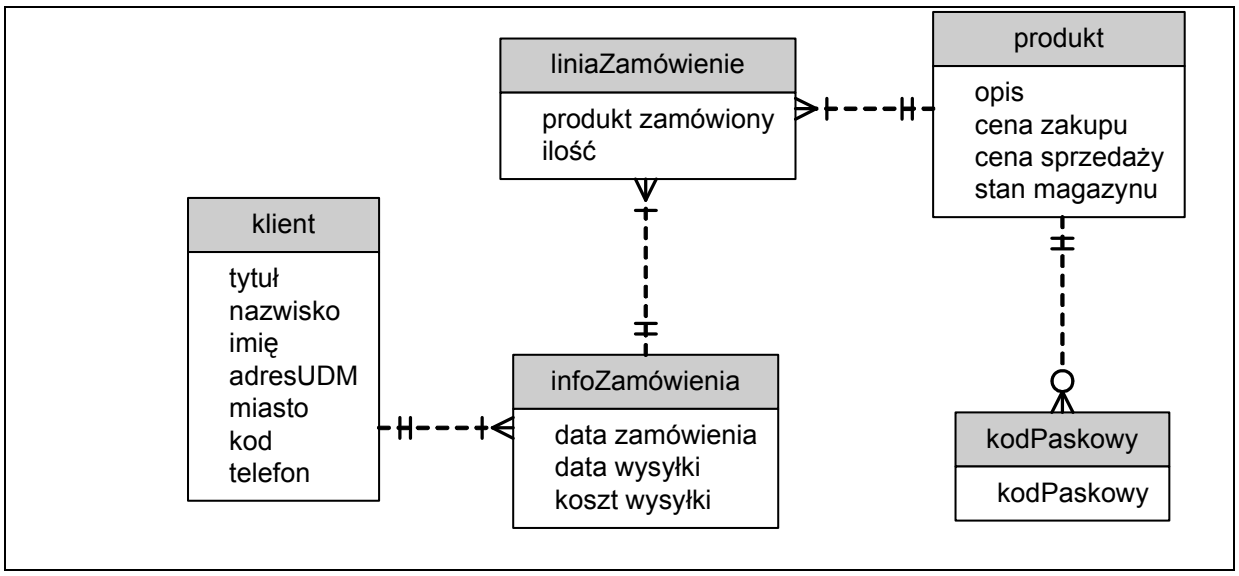
Dla każdego wiersza w tabeli A musi istnieć co dokładnie jeden wiersz w tabeli B Dla każdego wiersza w tabeli B może istnieć zero lub wiele wierszy w tabeli A	

Wracając do zaproponowanych wcześniej tabel:









Poprawność diagramów

Kompletny model ER

- Encje
  - dobrze nazwane
  - mają atrybuty i unikalne identyfikatory
  - pozostają w związkach z innymi encjami
- Atrybuty
  - dobrze nazwane
  - mają określony typ i długość
- Związki
  - dobrze określony stopień, opcjonalność i transferowalność

Poprawność związków

- Związki 1 do 1 są podejrzane
- Związki obustronnie obowiązkowe są podejrzane
- Związki rekurencyjne muszą być obustronnie opcjonalne

- W modelu implementacyjnym związki *n* do *m* powinny zostać rozbite

Konwersja do modelu fizycznego

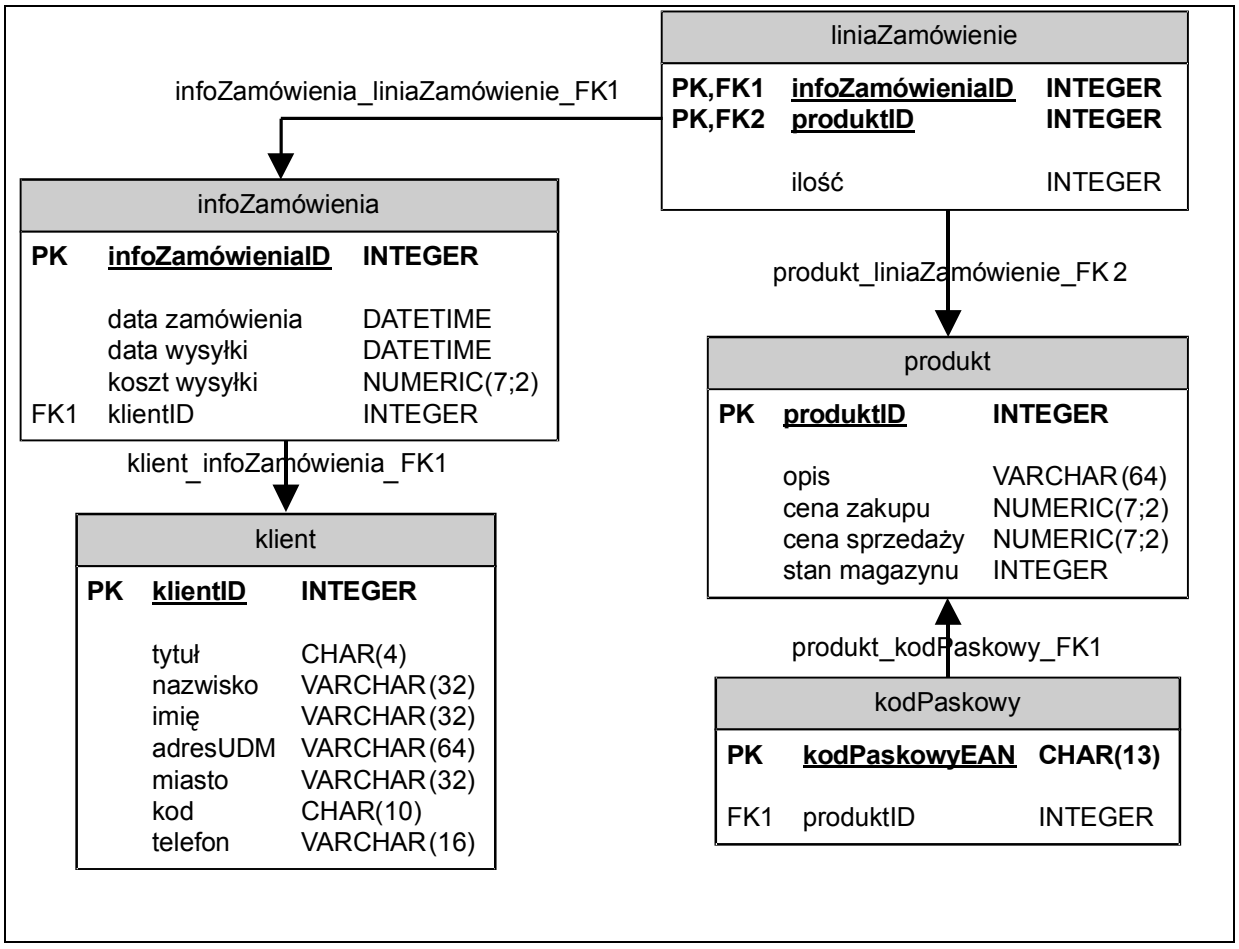
Korzysta się tu ze znalezionych wcześniej związków.

Tworzenie kluczy głównych

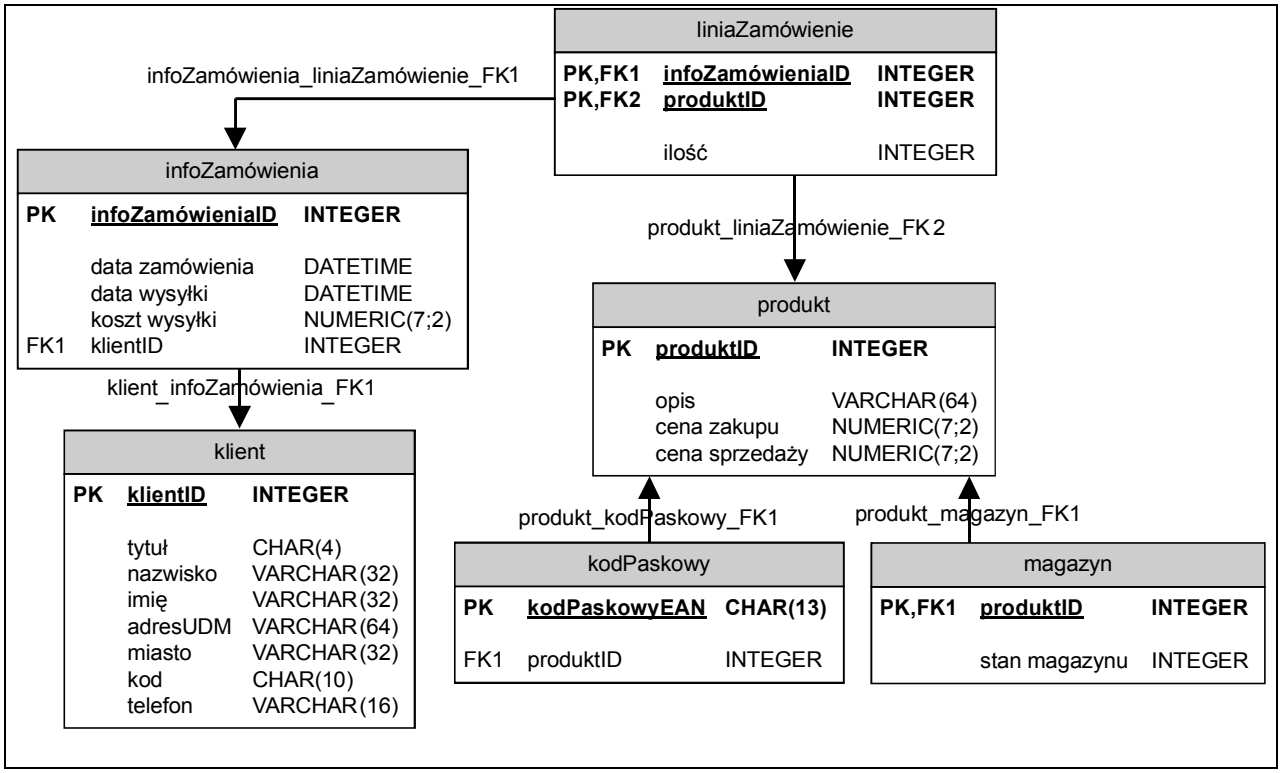
Poszukuje się tu najpierw kluczy-kandydatów, a potem wyznacza się klucze główne. Rozpoczyna się od unikatowej kolumny (nie zawierającej NULL), potem szuka się unikatowej kombinacji kolumn. Najlepsze tu są kolumny o „najkrótszych typach argumentów". Jeśli brak klucza, to może trzeba przeredagować model, albo wprowadzić automatyczną kolumnę klucza głównego. Propozycje mogą być następujące:

kodPaskowy – istnieje tylko jedna kolumna, wiec jest ona kluczem głównym
klient – trudno wybrać, więc niech to będzie dodatkowy, generowany numer
infoZamówienia – trudno wybrać, więc niech to będzie dodatkowy, generowany numer
produkt – dodajemy klucz, bo opis może być zbyt długi lub powtarzający się
liniaZamówienia – można byłoby wybrać produkt zamówiony, ale co się stanie, gdy dwóch klientów zamówi ten sam towar (i będzie on występował w dwóch różnych wierszach liniaZamówienia)? Wymyślimy ten klucz później.

Mając klucze główne łączymy tabele. Powstają klucze obce.
W tabeli liniaZamówienia pojawiły się klucze dwa klucze obce, które posłużyć mogą za klucz główny (produkt zamówiony z tabeli koncepcyjnej zamienia się na produktID).



Wydaje się, że poprawić schemat bazy danych może modyfikacja tabeli produkt. Jeśli bowiem produkt jest w magazynie, to może być on dodatkowo opisany przez numer półki, okres ważności, etc. Wprowadźmy więc nową tabele odpowiedzialną za informacje magazynowe i połączmy ją z produktem.



Ustalanie typów danych

Na powyższych diagramach założono już typy danych. Być może wnikliwa ich analiza pozwoli osiągnąć bardziej optymalne rozwiązanie. Ważne też jest, dla jakiego motoru bazy danych przygotowywany jest projekt.

Dokończenie tabel

W tym kroku sprawdza się, czy wszystkie zdefiniowane encje i artybuty znajdują się w modelu. Jeśli tak, to być może dołączenie dodatkowych tabel słownikowych lub tabel z danymi statycznymi ułatwi później wpisywanie danych. Tabele słownikowe składają się z dwóch kolumn – unikatowego klucza i danych (jak np. nazwy miast).

Testowanie bazy danych

Po wygenerowaniu bazy danych dobrze jest wykonać zestaw operacji wstawiania, odczytywania i usuwania danych.