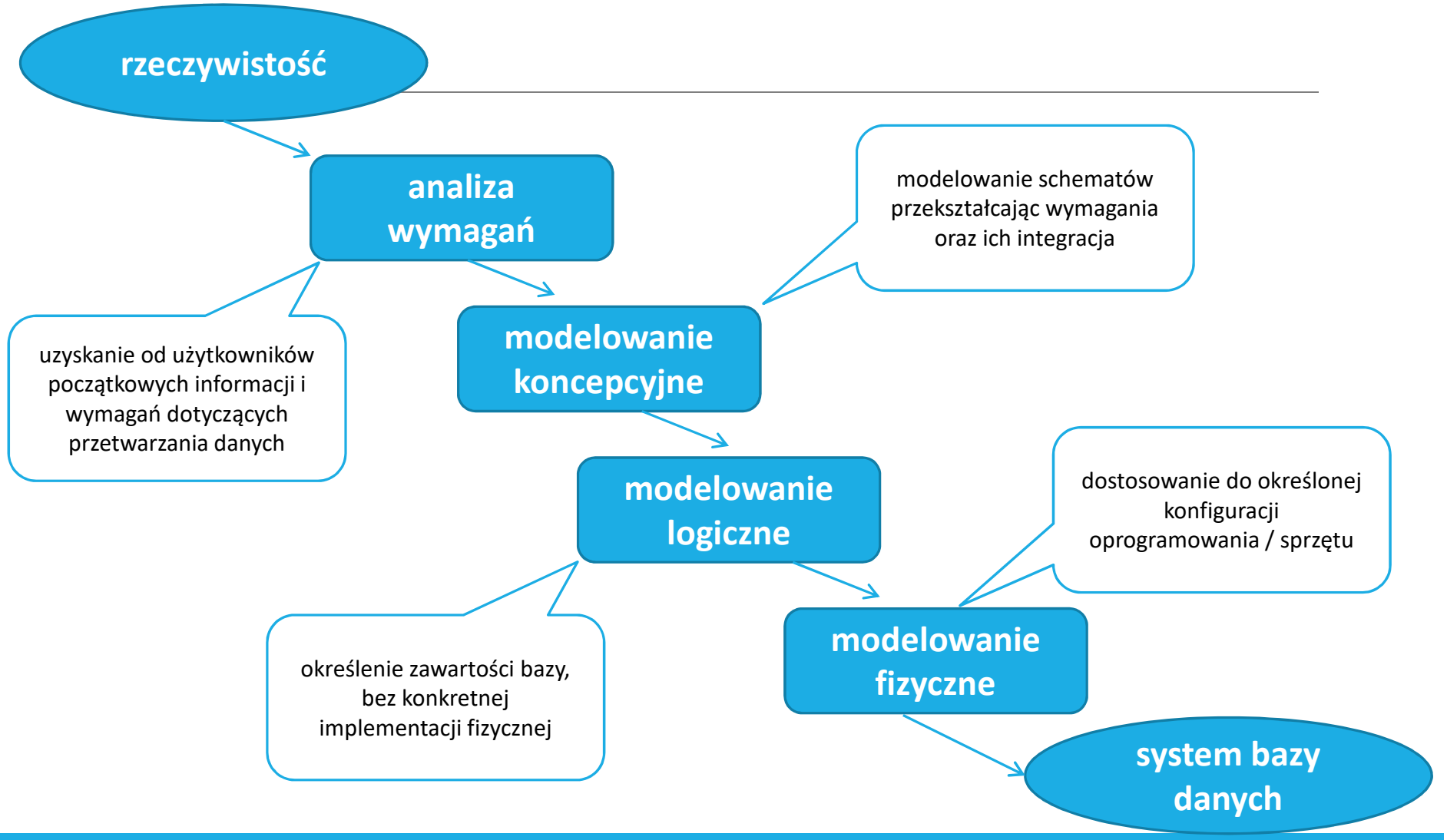
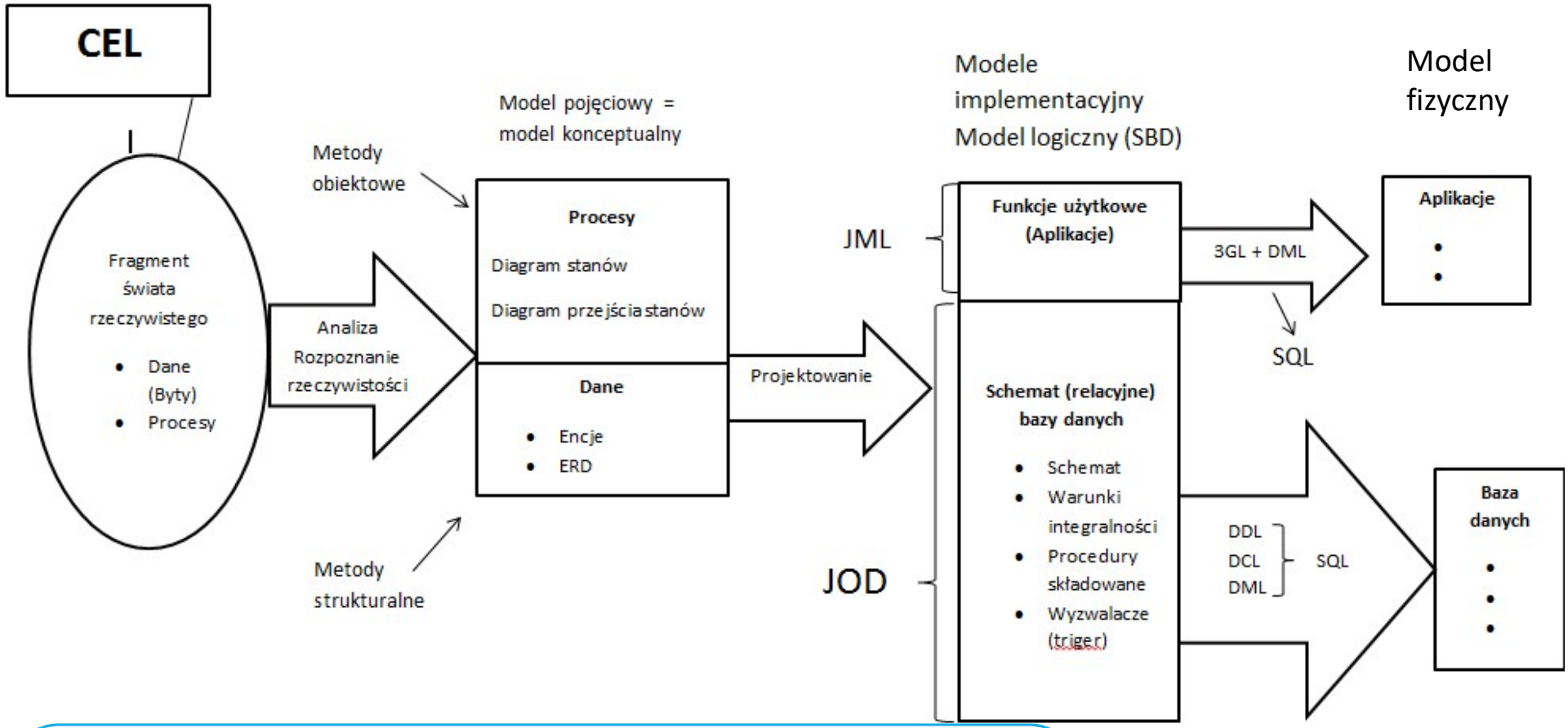
Zasady projektowania baz danych

Tworzenie baz danych





JOD – język opisu danych np. utwórz tabelę DDL
JML – język manipulacji danych np. dopisz do tabeli, skasuj z tabeli DML

DML (ang. Data Manipulation Language –„ język manipulacji danymi")
SQL DDL (ang. Data Definition Language – „język definicji danych")
SQL DCL (ang. Data Control Language - „język kontroli nad danymi")
SQL DQL (ang. Data Query Language – „strukturalny definiowania zapytań")
SQL (ang. Struetured Query Language- „strukturalny język zapytań")

Na SZBD składa się baza danych + aplikacja (najczęściej jakaś strona www)

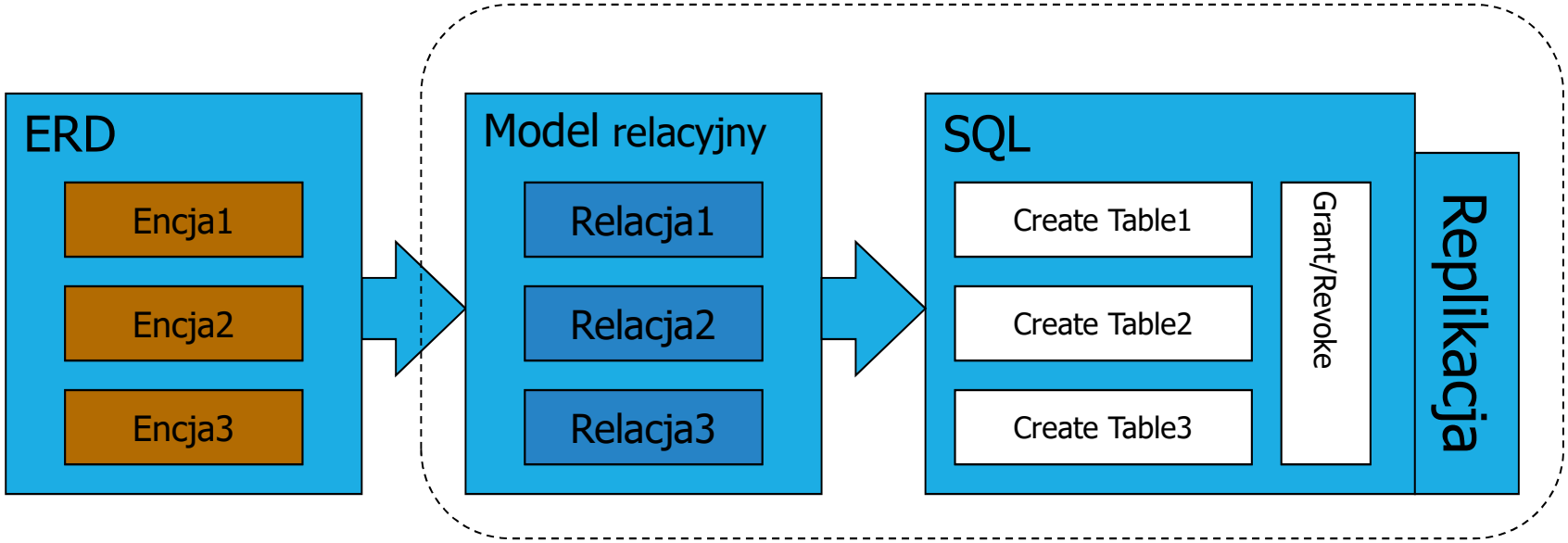
Cały proces projektowania bazy danych możemy podzielić na kilka etapów:

1. Planowanie bazy danych.
2. Tworzenie modelu konceptualnego (diagramu ERD).
3. Transformacja modelu konceptualnego na model relacyjny.
4. Proces normalizacji bazy danych.
5. Wybór struktur i określenie zasad dostępu do bazy danych.

Projektowanie modelu bazy danych powinno składać się z następujących działań:

1. Określenie występujących zbiorów encji.
2. Określenie atrybutów przypisanych do poszczególnych encji.
3. Określenie dziedziny poszczególnych atrybutów.
4. Ustalenie kluczy podstawowych.
5. Określenie typów występujących związków.
6. Zweryfikowanie utworzonego modelu.

Projektowanie baz danych



Projektowanie konceptualne, logiczne i fizyczne.

Projektowanie konceptualne, logiczne i fizyczne.

Konceptualne (pojęciowe) projektowanie bazy danych – to proces konstrukcji modelu danych, który jest niezależny od wszelkich aspektów fizycznych (reprezentacja obiektów w uniwersalnym modelu niezależnym od modelu implementacyjnego np. ERD, modele UML)

Logiczne projektowanie bazy danych – to proces konstrukcji modelu, który jest oparty na specyficznym modelu danych (np. model relacyjny, model obiektowy) ale niezależny od konkretnego BDMS i innych aspektów fizycznych.

Fizyczne projektowanie bazy danych – to proces tworzenia opisu implementacji bazy danych w pamięci zewnętrznej. Opis ten zawiera bazowe relacje oraz organizacje plików i indeksów zapewniających efektywny dostęp do danych, realizacje więzów integralności i środków bezpieczeństwa danych.

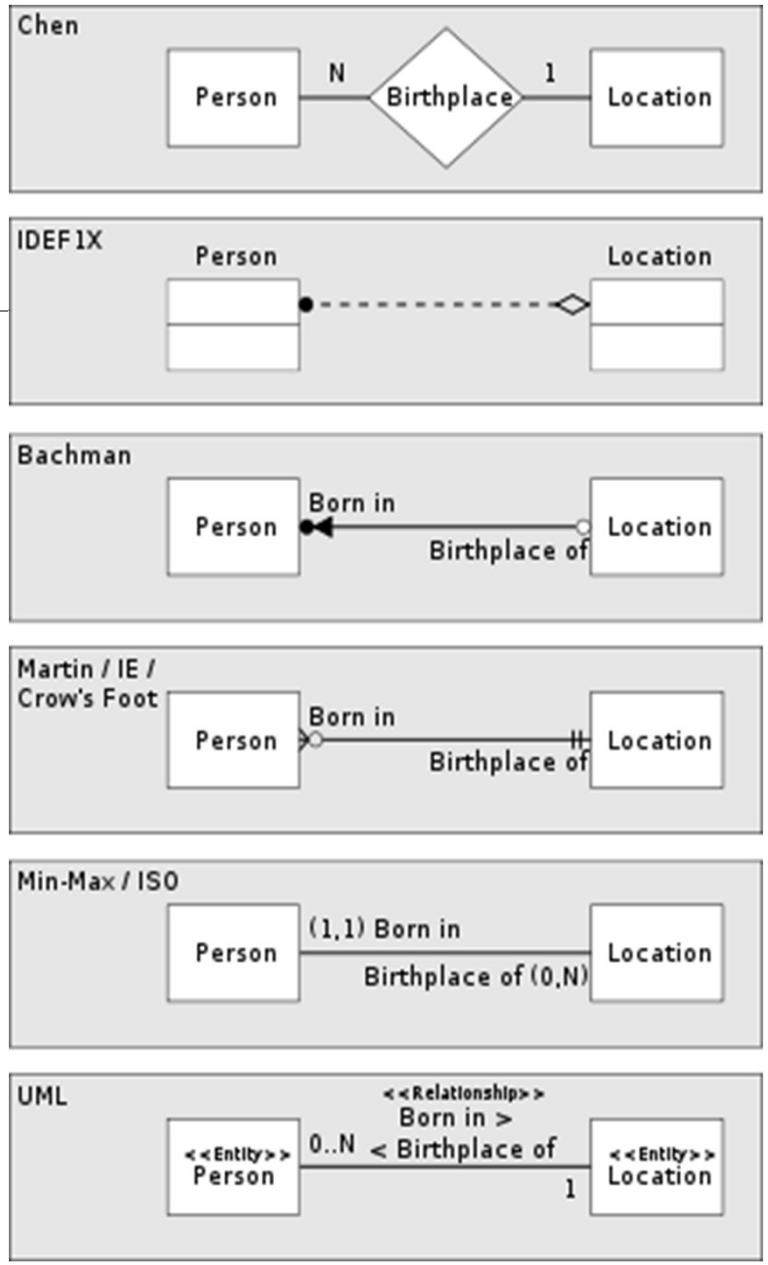
Projektowanie konceptualne

Konceptualne projektowanie bazy danych to konstruowanie schematu danych niezależnego od wybranego modelu danych, docelowego systemu zarządzania bazą danych, programów użytkowych czy języka programowania.

Do tworzenia modelu graficznego schematu bazy danych wykorzystywane są diagramy związków encji, z których najpopularniejsze są diagramy **ERD (ang. Entity Relationship Diagram).** Pozwalają one na modelowanie struktur danych oraz związków zachodzących między tymi strukturami. Nadają się szczególnie do modelowania relacyjnych baz danych, ponieważ umożliwiają prawie bezpośrednie przekształcenie diagramu w schemat relacyjny.

Model związków encji

Diagram związków encji lub Diagram ERD (od ang. Entity Relationship Diagram) –rodzaj graficznego przedstawienia związków pomiędzy encjami używany w projektowaniu systemów informacyjnych do przedstawienia konceptualnych modeli danych używanych w systemie.



Różne notacje ERD

ERD- Entity-Relationship Diagram

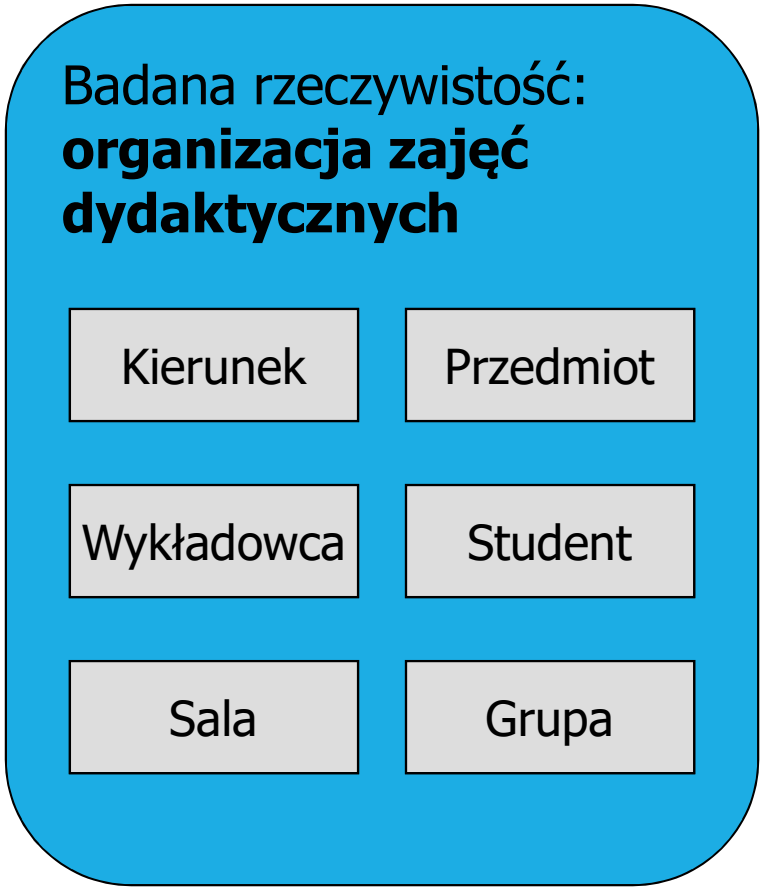
Diagramy ERD składają się z trzech rodzajów elementów:

- Zbiorów encji
- Atrybutów encji
- Związków zachodzących między encjami.

Narzędzia **CASE (ang. Computer Aided Software Engineering)** są wykorzystywane podczas projektowania różnego rodzaju oprogramowania, najczęściej wspomagają proces jego wytwarzania. Narzędzie te pozwalają tworzyć modele graficzne odpowiadające konstrukcjom programistycznym. Przykładem narzędzia typu CASE jest program DBDesigner4.

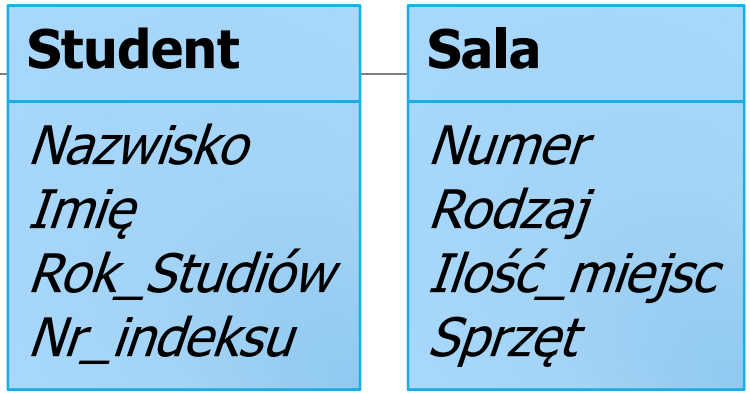
Pojęcie ENCJI

Encja to pewien wyodrębniony logicznie i jednoznacznie określony byt (obiekt), rozpoznawalny w badanej rzeczywistości i pełniący w niej określoną rolę. Encja może być zarówno obiektem fizycznym (takim jak np. samochód, drzewo, książka itp.) jak również zdarzeniem (np. sprzedaż samochodu, zasadzenie drzewa, zakup książki itp.). Każda encja jest jednoznacznie identyfikowana na podstawie swojej nazwy. Przyjęło się, że nazwy encji są *rzeczownikami w liczbie pojedynczej*. Graficznie każda encja jest reprezentowana przez prostokąt.



Atrybuty ENCJI

Atrybuty encji to cechy (własności) charakteryzujące daną encję w badanej rzeczywistości, którym przypisywane są określone wartości. Wartości poszczególnych atrybutów pozwalają odróżniać encje od siebie.



Encje posiadające te same własności tworzą **typy (zbiory) encji**. W praktyce, dla uproszczenia przyjęto używać określenia *encja* zarówno w odniesieniu do typu encji, jak również do określonego wystąpienia encji (określonej instancji encji).

Typ Encji:
Studenci

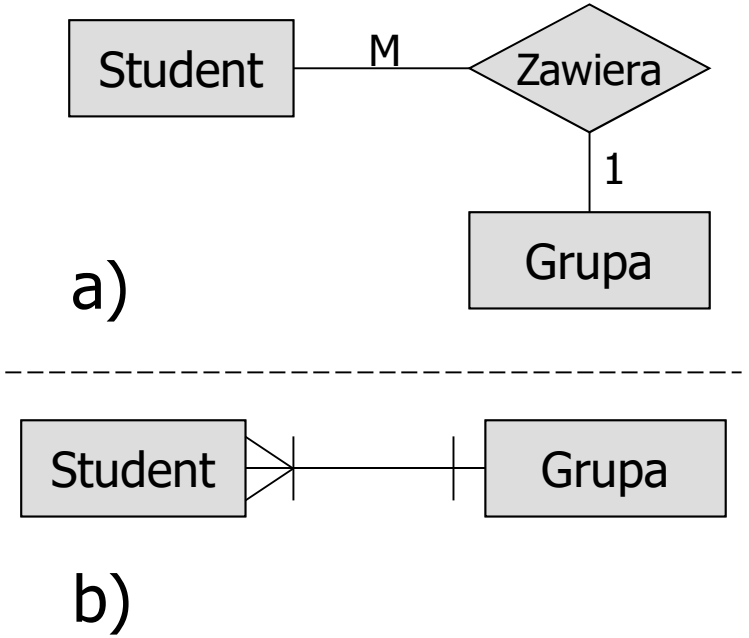
Instancja encji:
Nazwisko: Kowalski
Imię: Jan
Rok_Studiów: I
Nr_indeksu: R-10/03

Instancja encji:
Nazwisko: Nowak
Imię: Anna
Rok_Studiów: IV
Nr_indeksu: R-24/99

Związki ENCJI

Związki określają powiązania pomiędzy poszczególnymi encjami. W najprostszych modelach uwzględniane są związki występujące pomiędzy dwoma encjami. Pomiędzy dwoma różnymi encjami może zachodzić wiele związków, ale pomiędzy dwoma tymi samymi encjami może zachodzić tylko jeden związek.

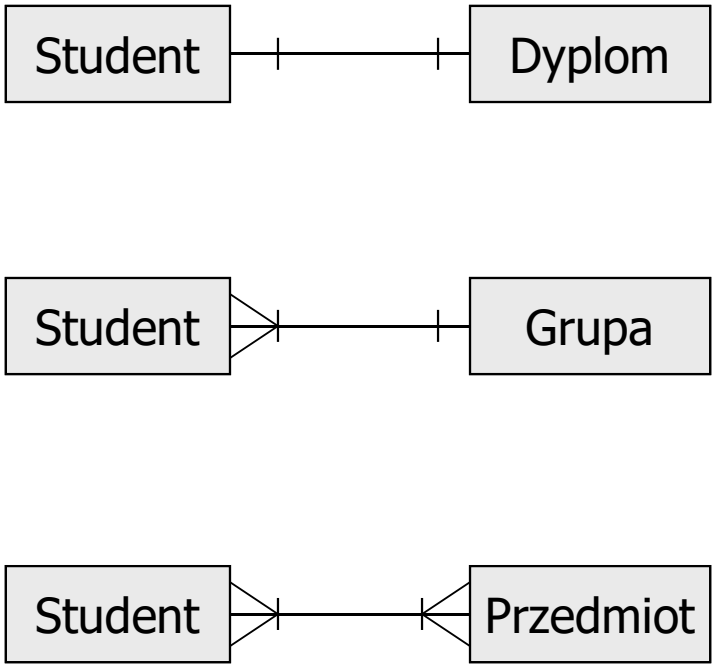
Każdy związek posiada swoją nazwę. Przyjęło się, że nazwy związków są czasownikami. Graficznie związek jest zwykle reprezentowany przez romb połączony liniami z encjami, pomiędzy którymi zachodzi wraz z oznaczeniem jego liczebności (a) lub też przez samą linię zakończoną symbolami określającymi jego liczebność (b). W niektórych notacjach nazwy związków się pomija.



Liczebność związku ENCJI

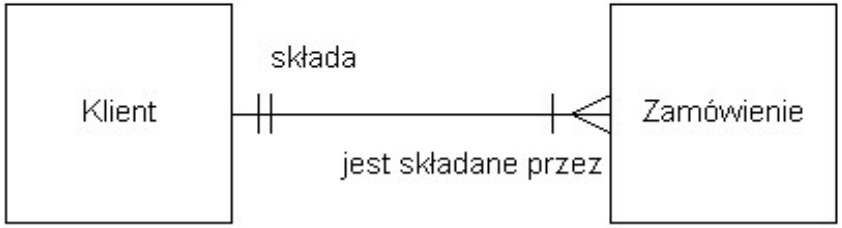
Liczebność (stopień) związku encji określa liczbę instancji biorących udział w danym związku. Rozróżnia się związki:

- **jednojednoznaczne (jeden-do-jeden, 1:1)** – każdej instancji pierwszej encji odpowiada dokładnie jedna instancja drugiej encji i odwrotnie;
- **jednoznaczne (jeden-do-wiele, 1:M)** – każdej instancji pierwszej encji odpowiada M instancji drugiej encji, ale każdej instancji drugiej encji odpowiada tylko jedna instancja pierwszej encji;
- **wieloznaczne (wiele-do-wiele, M:N)** – każdej instancji pierwszej encji odpowiada M instancji drugiej encji, a każdej instancji drugiej encji odpowiada N instancji pierwszej encji.



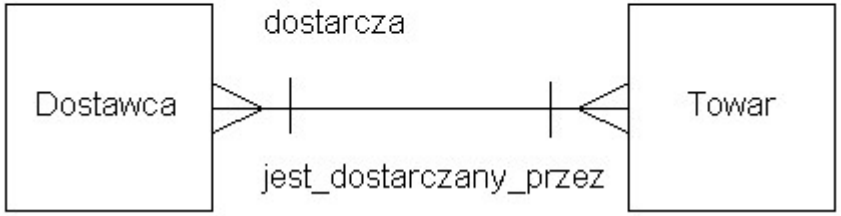
PRZYKŁADY ZWIĄZKÓW

KLIENT *składa* **ZAMÓWIENIE**
Arność: ***Jeden*** **do** ***Jeden-lub-Wiele***



KLIENT *posiada* **NIP**
Arność: ***Jeden*** **do** ***Jeden*** **(opcjonalnie)**

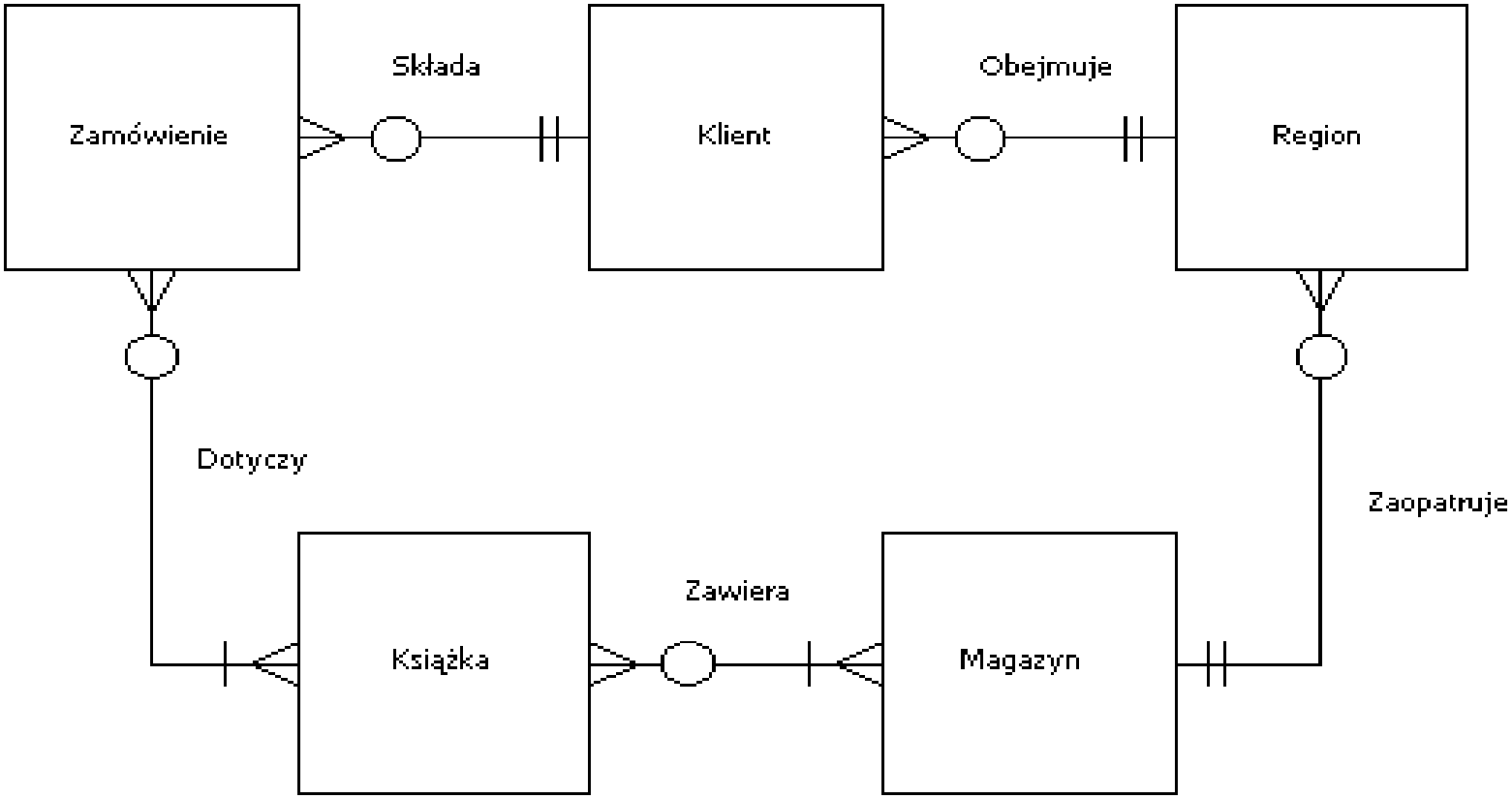
DOSTAWCA *dostarcza* **TOWAR**
Arność: ***Wiele*** **do** ***Wiele***




Opcjonalność (może)

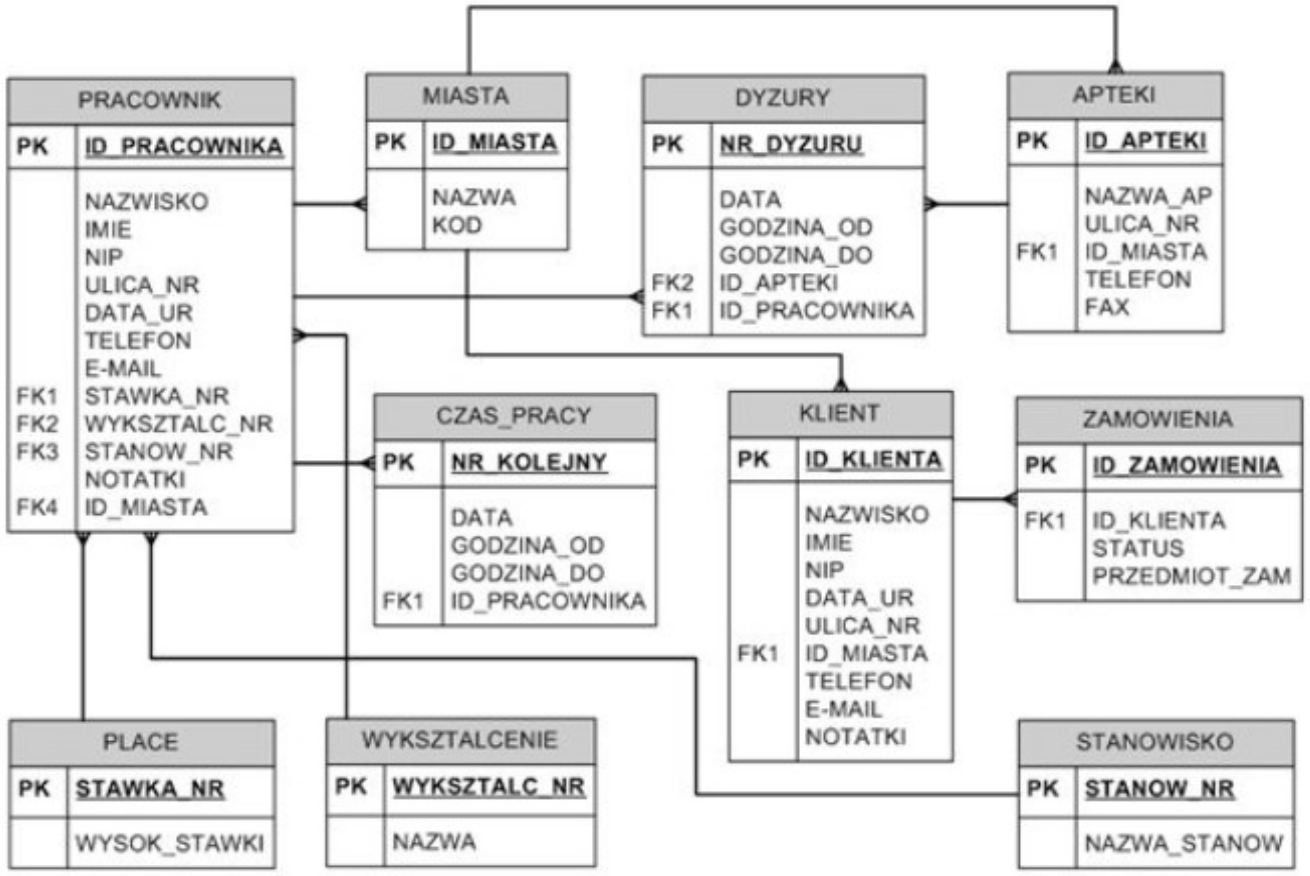

wymagany (musi)

Przykład ERD

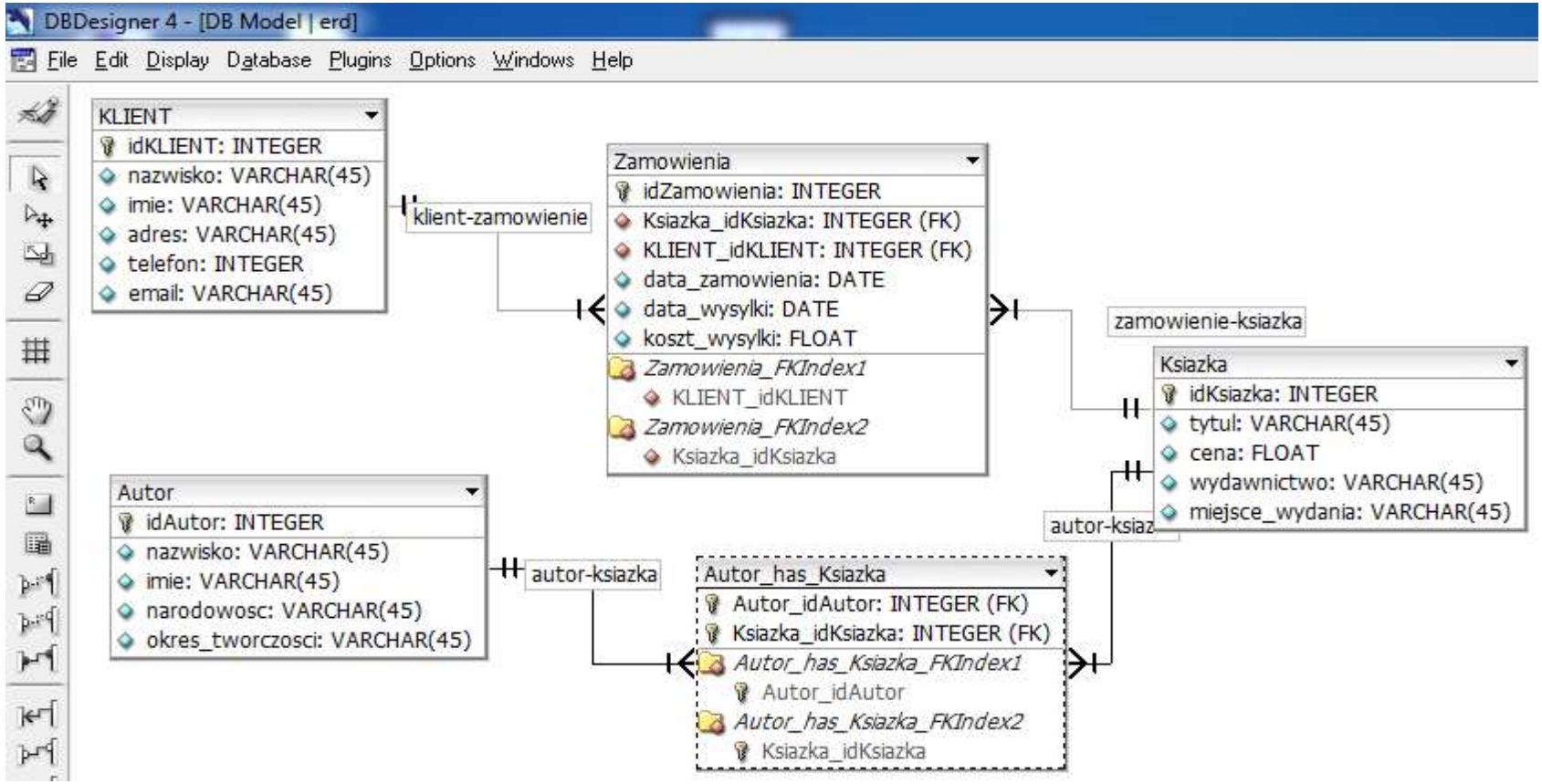


Przykład ERD

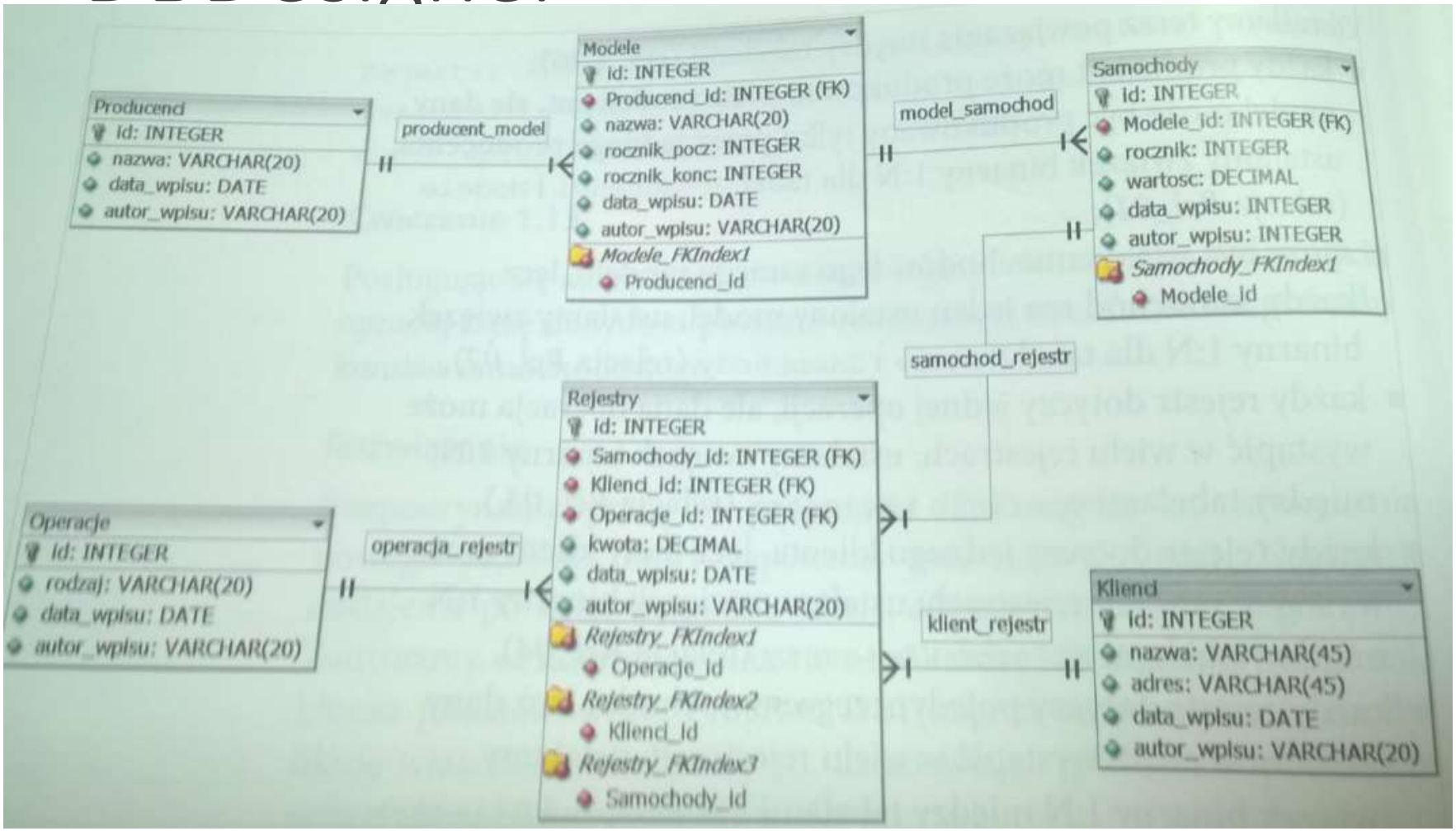
PK – klucz główny
FKx – klucz obcy



Przykład ERD



Przerysuj do programu DBDesigner



Projekt i implementacja bazy danych obejmuje następujące fazy:

I. **projekt koncepcji, założenia**
II. **projekt diagramów (projekt konceptualny)**
III. **projekt logiczny**
IV. **implementacja**

projekt koncepcji, założenia

1. **Sformułowanie zadania projektowego:** podanie przedmiotu projektowania, jego celów, przeglądu zadań, specyfiki i uwarunkowań.
2. **Analiza stanu wyjściowego**; analiza stanu zastanego, uwarunkowań prawnych, przyjętego obiegu istniejącej dokumentacji, analiza występujących problemów, etc. pomocne mogą być scenariusze postępowania i ich analiza (elementy/obiekty, charakterystyki/atrybuty, struktura/przepływ danych, powiązania/relacje).
3. **Analiza wymagań użytkownika** (wstępna); na tym etapie należy określić podstawowe cele, zadania i funkcjonalność jakie mają być realizowane przez projektowaną bazę danych oraz ew. wymagania dotyczące projektu i dokumentacji.
4. **Określenie scenariuszy użycia.** Scenariusze użycia pozwolą na konstrukcję diagramów DFD i STD oraz hierarchii funkcji.
5. **Identyfikacja funkcji.** Określenie podstawowych funkcji realizowanych w bazie danych.

projekt diagramów (projekt konceptualny)

1. **Analiza hierarchii funkcji projektowanej aplikacji**; określenie struktury zależności hierarchicznych pomiędzy jednostkami analizowanego systemu, zwłaszcza w zakresie specyfikacji wymagań funkcjonalnych.
2. **Budowa i analiza diagramu przepływu danych (DFD);** ma na celu określenie przepływu danych (wejścia, wyjścia, operacje, przechowywanie) oraz elementów sterowania tym przepływem, co może być pomocne dla tworzenia aplikacji. Specyfikacja danych wejściowych i wyjściowych.
3. **Wybór encji (obiektów) i ich atrybutów.**
4. **Projektowanie powiązań (relacji) pomiędzy encjami. Konstrukcja diagramu ERD;** jest to zasadniczy etap procesu projektowania struktury bazy danych. Identyfikacja klas encji, ich atrybutów, zdefiniowanie (określenie) kluczy. Tablica krzyżowa powiązań, eliminacja powiązań wiele-do-wielu. Konstrukcja diagramu ERD.
5. **Projekt diagramów STD** (diagramy przejść pomiędzy stanami). Wykonanie w oparciu o scenariusze użycia i strukturę bazy danych.

projekt logiczny

1. **Projektowanie tabel, kluczy, indeksów, w oparciu o zdefiniowany diagram ERD;** na tym etapie następuje „sprecyzowanie” struktury bazy danych wraz ze szczegółami technicznymi. Projekt bazy w języku SQL.

2. **Słowniki danych.** Specyfikacja słownika danych. Specyfikacja dziedzin i ograniczeń.

3. **Analiza zależności funkcyjnych i normalizacja tabel (dekompozycja do 3NF, BCNF, 4NF, 5NF);** na tym etapie dokonuje się sprawdzenia, czy tabele spełniają warunki zakładanych postaci normalnych i ew. dekompozycji w celu normalizacji.

4. **Denormalizacja struktury tabel;** ma ona na celu optymalizację przetwarzania, przechowywania danych archiwalnych, dostosowanie do specyficznych wymagań użytkownika.

5. **Projektowanie operacji na danych: zdefiniowanie kwerend dla realizacji funkcji wyspecyfikowanych w projekcie;** (zgodnie z wymaganiami użytkownika; na tym etapie mogą one zostać uszczegółowione bądź zmodyfikowane). Projekt w języku SQL.

projekt logiczny

1. **Zdefiniowanie interfejsów graficznych do prezentacji, edycji i obsługi danych** (formularze; należy zaprojektować strukturę każdego formularza oraz powiązania miedzy nimi).
2. **Zdefiniowanie dokumentów graficznych do przetwarzania i prezentacji danych** (raporty; informacje generowane w raportach i ich struktura powinna odpowiadać dokumentów wymaganym w danej firmie.).
3. **Zdefiniowanie panelu sterowania aplikacji** (należy pamiętać o dostosowaniu do potrzeb użytkownika).
4. **Zdefiniowanie makropoleceń dla realizacji typowych operacji;** (określonych dla panelu sterowania i ew. innych formularzy).
5. **Uruchamianie i testowanie aplikacji** (on-site, przy współpracy użytkownika)**.**

Normalizacja bazy danych

Tabele w bazie danych powinny być logicznie uporządkowane i funkcjonalne. W każdym polu i rekordzie powinna być jak najmniejsza porcja informacji tak, aby dane nie były powielane i można było je łatwo znaleźć.

Normalizacja to proces sprowadzania bazy danych do odpowiedniej postaci. Polega on przede wszystkim na dzieleniu tabeli na kilka połączonych kluczem tabel. Głównym powodem, dla którego normalizuje się bazę danych jest uniknięcie anomalii, które mogą wystąpić przy nieprawidłowo skonstruowanej strukturze bazy.

Anomalie w bazie danych

Załóżmy, że mamy następującą strukturę bazy książek w bibliotece:

Tytuł książki	Autor	Wypożyczający	Adres wypożyczającego	Data wypożyczenia

W tej bazie wystąpią następujące anomalie:

Anomalia	**Opis**
przy aktualizacji	Jeżeli wypożyczający zmienił adres, trzeba przeszukać całą bazę i we wszystkich komórkach, w których występuje, zmienić ten adres
przy usuwaniu	Jeżeli wypożyczający zwróci ostatnią książkę, zostanie utracona informacja na jego temat (adres i inne dane osobowe)
przy wstawianiu	Nowa osoba nie może zapisać się do biblioteki, jeżeli nie wypożyczy książki (a nie musi od razu wypożyczać)
redundacja	Redundacja, czyli powtarzanie tej samej informacji w kilku miejscach w bazie, powoduje niepotrzebne zajmowanie pamięci (wypożyczenie dwóch książek powoduje, że niepotrzebnie adres jest powtarzany dwa razy)

cd. Anomalie w bazie danych

Aby osiągnąć właściwe uporządkowanie danych i zminimalizować ich powtarzalność należy każdą tabelę poddać co najmniej trzem etapom normalizacji (w sumie jest ich sześć). Każda z następnych postaci normalnych jest bardziej wymagająca od poprzedniej i jest zgodna z poprzednimi postaciami normalizacji.

Anomalie powstają gdy próbujemy w jednej tabeli (w żargonie matematycznym - relacji) umieścić zbyt wiele danych.

Rozwiązaniem tych problemów są **postacie normalne**, które gwarantują, że takie anomalie nie będą miały miejsca.

Implementacja

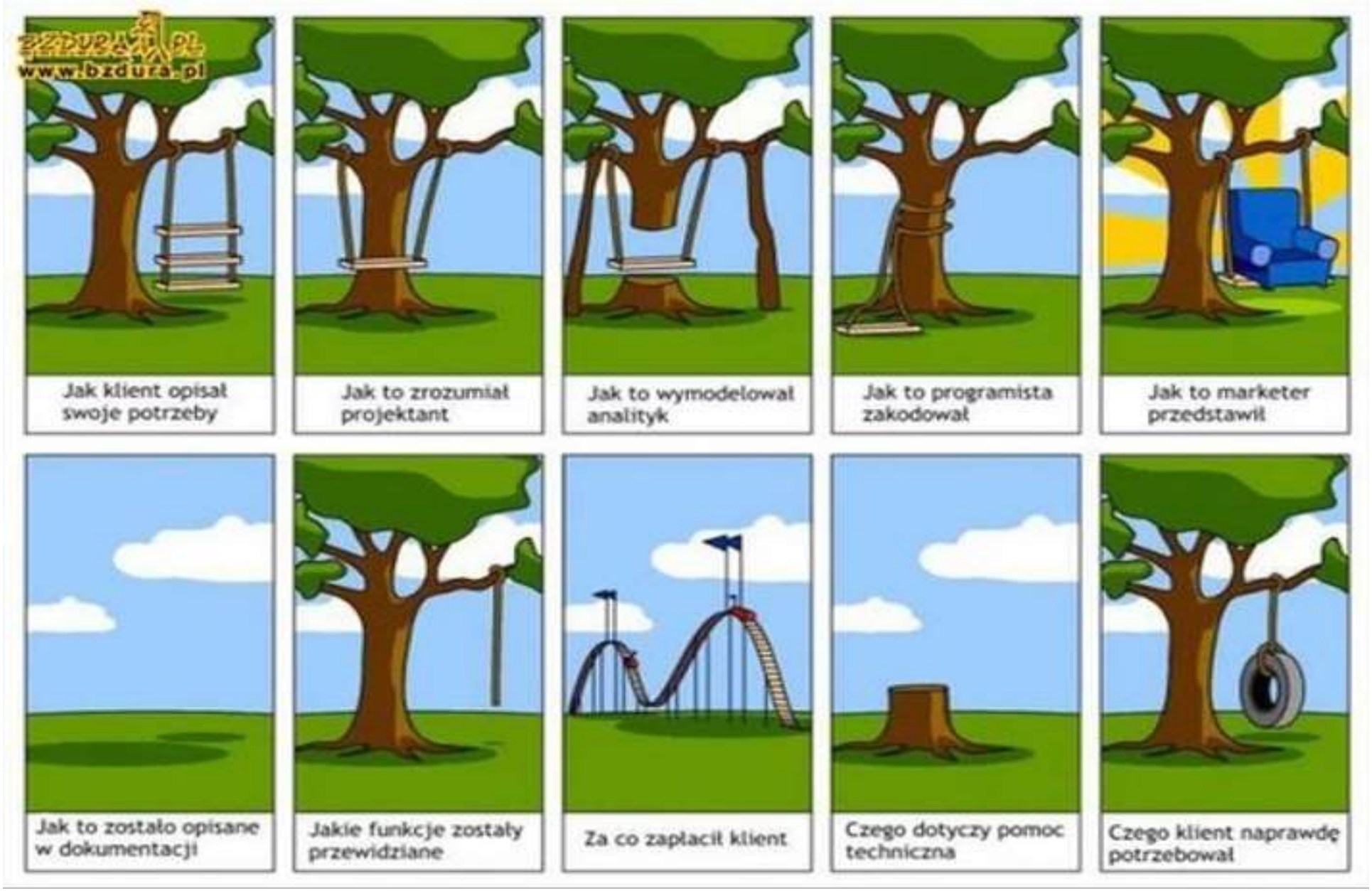
1. **Wprowadzanie danych (**ręczne, automatyczne, import, on-line).
2. **Wdrażanie systemu do użytkowania** (stopniowe, modułami, z możliwością wycofania się, z dublowaniem danych i obliczeń)**.**
3. **Przeprowadzenie szkolenia użytkowników.**
4. **Zapewnienie dokumentacji technicznej i użytkowej** (ten wymóg powinien postawić użytkownik!).

Wybór środowiska implementacji

Implementacja bazy danych obejmuje realizację projektu z zastosowaniem określonej strategii i przy użyciu wybranego środowiska implementacji (systemu zarządzania bazami danych). Dokonując wyboru środowiska implementacji należy kierować się kilkoma aspektami, determinującymi jego użycie:

- ogólnym przeznaczeniem projektowanej bazy danych;
- postacią danych;
- sposobami ich użycia i prezentacji;
- zakresem dostępu do danych (przewidywaną ilością użytkowników oraz zakresem ich uprawnień);
- wymaganiami bezpieczeństwa i możliwościami oferowanymi w tym zakresie przez system.

Istotnym etapem kończącym procedurę implementacji jest testowanie bazy danych, pozwalające na wyeliminowanie błędów.



Rysunek 1.Praca zespołowa. Dlaczego duże projekty kończą się sprawami w sądzie?

Pojęcia:

Modelowanie – odwzorowanie rzeczywistych obiektów świata rzeczywistego w systemie informatycznym (bazie danych).

Unified Modeling Language (UML zunifikowany język modelowania) – język pół-formalny wykorzystywany do modelowania różnego rodzaju systemów.

Przypadków użycia (ang. use case diagram)- graficzne przedstawienie przypadków użycia, aktorów oraz związków między nimi, występujących w danej dziedzinie przedmiotowej.

Przykładowy schemat USE CASE