Model danych to zintegrowany zbiór zasad opisujących dane, relacje, powiązania (stosunki) pomiędzy danymi, dozwolone operacje i ograniczenia nakładane na dane i operacje.

Model danych jest próbą reprezentacji świata realnego i występujących w nim obiektów, zdarzeń oraz związków zachodzących miedzy nimi. Można go opisać jako konstrukcję składającą się z **trzech komponentów:**

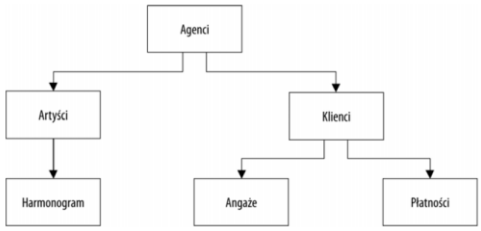
- **części strukturalnej** - składającej się z reguł określających budowę bazy danych:
- **części manipulacyjnej** - określającej, które operacje (transakcje) aktualizacji, pobierania i zmiany struktury można wykonywać na danych;
- **części zawierającej reguły integralności** - gwarantującej stabilność działania systemu.

Model jednorodny - to model, w którym wszystkie dane są umieszczone **w jednej tabeli**, **jednym arkuszu**. Przykładem takiego modelu jest książka telefoniczna. Cechuje go łatwość i szybkość odczytywania danych, jego wadą jest duża liczba duplikatów, które mogą się pojawiać i tym samym zwiększyć zużycie zasobów komputera. Dane w modelu jednorodnym nie zawsze będą łatwe do odnalezienia. Gdyby jednorodny model danych prezentować na przykładzie książki telefonicznej, to łatwe byłoby odnalezienie numeru telefonu na podstawie nazwiska, jednak wyszukanie imienia i nazwiska na podstawie numeru telefonu stwarzałoby problem.

W modelu jednorodnym **dane mogą być duplikowane (redundancja)**. Dzieje się tak właśnie ze względu na strukturę tego modelu. Na przykład, gdyby w Warszawie mieszkało 50 osób o nazwisku Nowak i każda z nich miała telefon, wówczas w jednorodnym modelu danych otrzymalibyśmy pięćdziesiąt duplikujących się nazwisk w kolejnych wierszach, różniących się adresami i numerami telefonów, przy czym nazwisko byłoby wielokrotnie powtórzone.

Hierarchiczny model danych - jego początki sięgają 1960 r., gdy rozpoczęto prace nad projektami IDS (integrated Data Store) oraz MIS (Management Information System). MIS rozwijany był przez IBM w ramach projektu kosmicznego Apollo. Hierarchiczny model bazy danych pod względem modelu przypomina strukturę odwróconego drzewa – istnieje jeden korzeń (tabela nadrzędna) oraz synowie (tabele podrzędne). Jeden ojciec może mieć wiele dzieci, ale każde dziecko ma tylko jednego ojca. Każdy rekord (z wyjątkiem głównego, który jest na szczycie) powiązany jest z jednym rekordem nadrzędnym.

Hierarchiczny model bazy danych



Baza danych agentów

W przykładzie pokazanym na rysunku 1.1 agent zajmuje się rezerwacjami kilku artystów estradowych, a każdy artysta ma swój harmonogram. Agent ma również kilku klientów, których potrzebami musi się zająć. Klient rezerwuje występ poprzez agenta i płaci mu za usługę.

Model hierarchiczny opiera się na dwóch strukturach danych - typach rekordów i związkach **nadrzędny-podrzędny.** Powiązanie nadrzędny - podrzędny to związek **„jeden do wielu**" pomiędzy dwoma typami rekordów. Model ten różni się od relacyjnego, ponieważ w modelu relacyjnym powiązania zachodzą przez klucze obce, a w hierarchicznym przez związek nadrzędny - podrzędny. W hierarchicznym modelu danych **nie można wstawić rekordu podrzędnego, dopóki nie zostanie powiązany z nadrzędnym. Usunięcie rekordu nadrzędnego powoduje automatycznie usunięcie wszystkich rekordów podrzędnych.**

Sieciowy model danych został ustandaryzowany w 1969 r. przez CODASYL (Conference on Data Systems Languages).

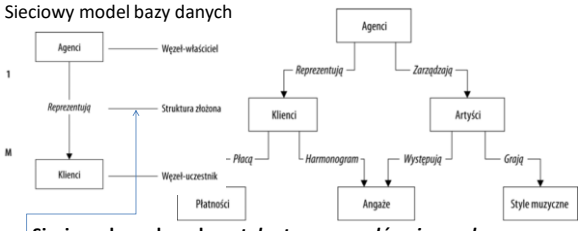
Jego twórcą jest Charles Bachman i mimo że nie znalazł szerszego zastosowania, to przyczynił się do powstania relacyjnego modelu baz danych. **Sieciowy model przyjmuje, podobnie jak hierarchiczny, strukturę przypominającą odwrócone drzewo z tą różnicą, że gałęzie jednego drzewa *mogą być wspólne z gałęziami innych drzew.***

SMD jest podobny do hierarchicznego, jednak z mniejszą liczbą ograniczeń: potomek(uczestnik) może mieć więcej rodziców(właścicieli).

Cechy sieciowego modelu danych:

- możliwość przeglądania danych w różnych kierunkach,
- wiele możliwych ścieżek dostępu,
- bardzo duża wydajność,
- duża złożoność systemu i trudność utrzymania.

Sieciowy model bazy danych



Sieciowe bazy danych *zostały stworzone głównie w celu rozwiązania niektórych problemów istniejących w hierarchicznych bazach* danych. Ich struktura reprezentowana jest przez **węzły** oraz **zestawy struktur**. Rysunek pokazuje diagram typowej sieciowej bazy danych. Węzeł przedstawia kolekcję rekordów, a **zestaw struktur ustala i prezentuje zależność w sieciowej bazie danych**. Wzajemny stosunek pary węzłów prezentuje przejrzysta konstrukcja, pokazując jeden węzeł jako właściciela (ang. owner), a drugi jako uczestnika(ang. member),

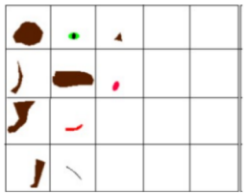
Model obiektowy

Relacyjną bazę danych można porównać do magazynu z meblami, w którym każda część mebla jest umieszczana w osobnym, przeznaczonym wyłącznie dla niej kontenerze (odpowiednikami kontenerów są tabele). Umieszczając meble w takim magazynie, rozbieramy je na części i osobno układamy drzwi, półki itd. Taki sposób przechowywania jest bardzo wydajny, ale wyjmując mebel z magazynu, każdorazowo musimy złożyć go w jedną całość.

Tymczasem model obiektowy przypomina magazyn, w którym meble są ustawiane bez rozbierania na części. W rezultacie korzystanie z magazynu jest prostsze (ale niekoniecznie szybsze), za to liczba mebli, jaką można zmieścić na tej samej przestrzeni, jest znacznie mniejsza.

Obiektowy model danych opiera się na koncepcji obiektów (podobnie jak w projektowaniu obiektowym - obiekt jest odwzorowaniem rzeczywistości lub abstrakcji).
Odwołania do określonego obiektu w tym modelu bazy danych są wykonywane za pomocą interfejsu, dzięki któremu są zachowane integralność i bezpieczeństwo danych. Obiektowe bazy danych korzystają z obiektowego języka zapytań **OQL (Object Query Language).** Każdy obiekt ma zaprojektowany interfejs określający metody dostępu do niego. Obecnie coraz popularniejszy staje się standard JDO (Java Data Object) stworzony przez firmę Sun Microsystems. W ramach tego standardu rozwinął się obiektowy język zapytań JDOQL (Java Data Object Query Language). Dość powszechnym niekomercyjnym i relacyjnym systemem bazodanowym mającym obiektowe rozszerzenie jest PostgreSQL

- Relacyjna baza danych kota:



•Obiektowa baza danych kota:

Potencjalnym polem zastosowań obiektowych baz danych są/były dziedziny wymagające bardziej rozbudowanych struktur danych, takie jak np. CAD (ComputerAidedDesign), oraz długie dokumenty tekstowe, obrazy, animacje, multimedia

Relacyjny model danych opracował w latach **osiemdziesiątych** XX w. Edgar Frank Codd. Opublikował on wówczas jedną z najważniejszych swoich prac pt. Relacyjny model logiczny dla dużych wielodostępnych baz danych. Przedsięwzięcie Codda znalazło entuzjastów na Uniwersytecie Kalifornijskim w Berkeley oraz w firmie IBM. Opracowane wówczas systemy baz danych były rozwijane nie tylko w celach komercyjnych. Larry Ellison, współfundator korporacji Oracle, projekt systemu zarządzania bazami danych oparł na założeniu Codda. Nazwa Oracle jest nie tylko określeniem DBMS, lecz także nazwą kodową projektu CIA, nad którym pracowali założyciele Oracle w korporacji Ampex Corporation.

Oprócz Oracle rozwijały się w tym czasie takie bazy danych, jak Sybase lub Informix, używające języka SQL. W 1985 r. Codd przedstawił 12 zasad opisujących model relacyjny baz danych. Zasady te rozwinął do 333 w książce wydanej w 1990 r. Dane przechowuje się w tabelach, nazywanych relacjami, składających się z wierszy (krotek) i kolumn (atrybutów)".

Obecnie spotykane bazy danych przystosowane są w większości do pracy wykorzystującej połączenia sieciowe (architektura KLIENT-SERWER). Aby sprostać rosnącemu zapotrzebowaniu na usługi oferowane przez bazy danych, wzrasta liczba zastosowań tego oprogramowania w **architekturze rozproszonej** (usługi chmurowe).

Postulaty Codda (12):

1. **Postulat informacyjny**. Informacje są reprezentowane w postaci logicznych tabel. Oznacza to, że fizyczny sposób organizacji i przechowywania danych przez serwer bazodanowy nie może mieć wpływu na działanie programów klienckich.

2. **Postulat dostępu**. Każda informacja musi być dostępna za pomocą nazwy tabeli, kolumny i wartości klucza podstawowego. Innymi słowy, znajomość struktury tabeli i wartości identyfikatorów rekordów musi wystarczyć do odczytania dowolnej informacji z bazy.

3. **Postulat fizycznej niezależności danych**. Sposób przechowywania danych i wewnętrzne mechanizmy dostępu do nich przez serwer nie mogą mieć wpływu na aplikacje klienckie. Na przykład to, czy dane są przechowywane w jednym pliku lub w wielu plikach, dla programów klienckich musi być całkowicie niewidoczne

4. **Postulat logicznej niezależności danych**. Zmiany w strukturze bazy danych, na przykład zmiana definicji tabeli, o ile tylko nie powodują utraty informacji i są poprawne semantycznie, nie mogą mieć wpływu na aplikację klienckа. ***Przykładowo, zgodnie z tym postulatem, dodanie kolumny do tabeli nie może zakłócać działania programów klienckich*.**

5. **Postulat niezależności dystrybucyjnej**. Odwołania do danych za pomocą języka SQL muszą być niezależne od fizycznej lokalizacji danych. Innymi słowy, ***aplikacje klienckie powinny mieć taki sam dostęp do danych znajdujących się na lokalnym dysku twardym, jak do danych rozproszonych pomiędzy różne lokalizacje***.

6. **Postulat zabezpieczenia przed modyfikacjami** przeprowadzanymi za pomocą języków proceduralnych. Jeśli serwer bazodanowy umożliwia bezpośrednie modyfikowanie poszczególnych rekordów za pomocą języków niższego poziomu, zmiany te nie mogą naruszać spójności danych, w szczególności nie mogą być sprzeczne z nałożonymi na tabele ograniczeniami.

7. **Postulat pełnego języka danych**. Serwer baz danych musi implementować jeden język pozwalający definiować tabele, widoki i ograniczenia (więzy spójności), zarządzać dostępem użytkowników i transakcjami oraz odczytywać i modyfikować dane.

8. **Postulat modyfikowania bazy danych przez widoki**. Zmiany danych przeprowadzane poprzez widoki muszą być odzwierciedlane w odpowiednich tabelach, a bezpośrednie zmiany danych w tabelach muszą być automatycznie widoczne poprzez widoki.

9. **Postulat modyfikowalności** danych - system musi umożliwiać operacje modyfikacji danych, musi
obsługiwać operacje INSERT, UPDATE oraz DELETE

10. **Postulat wartości NULL**. Serwer bazodanowy w spójny sposób przetwarza specjalną wartość NULL jak brakującą informację, a nie jak zero (0) czy pusty ciąg znaków („”).
11. **Postulat słownika danych**. Metadane (informacje o strukturze bazy danych) są przechowywane i udostępniane tak samo (czyli w postaci tabel), jak zapisane w bazie informacje.
12. **Postulat niezależności ograniczeń**. Ograniczenia (więzy spójności) muszą być definiowane w tym samym języku SQL i przechowywane po stronie bazy danych, a więc nie jest obowiązkowe implementowanie ich po stronie aplikacji klienckiej. Serwer baz danych musi umożliwiać zdefiniowanie przynajmniej dwóch typów ograniczeń:
a) **ograniczenia klucza podstawowego**, które gwarantują spójność danych w ramach tabel;
b) **ograniczenia klucza obcego**, które gwarantują spójność danych zapisanych w powiązanych tabelach.

Model relacyjno-obiektowy jest mieszanym modelem bazodanowym. a jego zastosowanie jest bardziej powszechne niż w przypadku modelu obiektowego. Dzieje się tak ze względu na trudną implementację i niezadowalającą wydajność (w niektórych zastosowaniach) typowego modelu obiektowego. Model ten pozwala w relacyjnych tabelach tworzyć kolumny, w których przechowywane są dane typu obiektowego, pozwala na definiowanie zmiennych oraz metod, które będą wykonywane na danych wprowadzanych do obiektu.

Rozproszona baza danych. Pracując z bazą danych typu Access lub używając narzędzi typu WAMP, LAMP lub XAMPP, mamy do czynienia z SZBD zainstalowanym na lokalnym komputerze. Zdarza się, że administrując serwerem, łączymy się zdalnie z bazą danych zainstalowaną na pojedynczym urządzeniu.

Możliwość łączenia komputerów za pomocą sieci, szyfrowania i zabezpieczania połączeń sprawia, że coraz częściej baza danych rozdzielana jest na węzły sieciowe, przez co jedna baza może występować w różnych konfiguracjach sprzętowych i programistycznych.

Taka baza może się znajdować na wielu komputerach położonych nie tylko w odległych od siebie geograficznie miejscach, lecz także w sieciach lokalnych.

Sieciową bazę danych definiujemy jako bazę danych zapisaną na kilku komputerach połączonych ze sobą w ten sposób, że korzystający z niej użytkownicy mają wrażenie, iż baza danych występuje w jednym miejscu.
W związku z coraz szybszymi połączeniami sieciowymi **podział jednej bazy danych pomiędzy kilka dedykowanych serwerów może przyśpieszyć operacje** wykonywane na zmiennych relacyjnych (tabelach). **Technologia ta zwiększa ilość zasobów sprzętowych przypadającą na określoną ilość danych**. Dane dzielone są na **fragmenty** (zbiory danych przechowywanych w jednym węźle sieci będące podzbiorami całej bazy danych). Sieciowe bazy danych wykorzystywane są również do zabezpieczania ważnych danych przed ich utratą. W tym celu tworzy się **repliki danych** (kopie całości bądź części danych przechowywane w innym miejscu niż oryginał). Rozproszenie danych poprawia wydajność szczególnie w przypadku dużych firm mających oddziały odlegle od siebie geograficznie.

Podstawowym powodem zainteresowania projektowaniem baz danych powinien być fakt, że jest ono kluczowe dla zachowania konsekwencji, integralności oraz dokładności danych w bazie. Jeśli baza danych zostanie zaprojektowana nieprawidłowo, trudno będzie wydobyć z niej określone typy informacji, wzrośnie także ryzyko uzyskania niedokładnych informacji. Niedokładne informacje to prawdopodobnie najbardziej szkodliwy rezultat nieprawidłowego projektowania bazy danych, który może mieć negatywny wpływ na działanie organizacji. Jeżeli baza danych ma bezpośrednie przełożenie na sposób przeprowadzania codziennych operacji w firmie lub jeśli będzie miała wpływ na kierunek rozwoju firmy w przyszłości, trzeba zainteresować się projektowaniem baz danych.

- *Baza danych wspiera zarówno wyszukiwanie wymagane, jak i doraźne.* Baza danych musi przechowywać dane niezbędne do wsparcia wymagań dotyczących informacji określonych w trakcie procesu projektowego oraz doraźnych kwerend wpisywanych przez użytkownika.
- *Tabele są skonstruowane w sposób poprawny i wydajny.* Każda tabela w bazie reprezentuje pojedynczy temat, jest skomponowana z relatywnie odrębnych pól, minimalizuje ilość zbędnych danych, a także można ją zidentyfikować w bazie danych na podstawie pola zawierającego unikatowe wartości.
- *Integralność danych jest narzucona na poziomie pól, tabel oraz zależności.* Te poziomy integralności pomagają zagwarantować ważność i trafność struktur danych oraz ich wartości.

integralność definiowana jest jako połączenie trzech koncepcji: dokładność (ang. accuracy), prawdziwość (ang. correctness), oraz aktualność (ang. validity).

- *Baza danych nadaje się do dalszego rozwoju.* Struktura bazy danych powinna być prosta do modyfikacji lub rozwoju w miarę zmieniania się i rozrostu wymagań firmy.

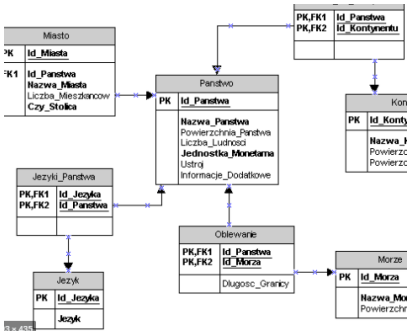
Etapy projektowania bazy danych

Konceptualne projektowanie bazy danych - to proces konstrukcji modelu danych, który jest niezależny od wszelkich aspektów fizycznych (specyficzny model danych, docelowy SZBD, programy użytkowe, języki programowania, platforma sprzętowa)
Logiczne projektowanie bazy danych - to proces konstrukcji modeli który jest oparty na specyficznym modelu danych (np. model relacyjny, obiektowy) ale niezależny od konkretnego SZBD 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.

Model	Konceptualny	Logiczny	Fizyczny
Nazwy encji	Tak	Tak	
Związki encji	Tak	Tak	
Atrybuty		Tak	
Klucze główne		Tak	Tak
Klucze obce		Tak	Tak
Nazwy tabel			Tak
Nazwy kolumn	-		Tak
Typy danych			Tak

Rys. 8.1. Proces projektowania bazy danych

Miasta



Relacja w ujęciu formalnym to **tabela** posiadająca kolumny i wiersze, w ujęciu matematycznym, każdy podzbiór iloczynu kartezjańskiego zbiorów, w ujęciu intuicyjnym - powiązanie pomiędzy tabelami. Związek między dwoma relacjami to relacja miedzy dwoma tabelami

Atrybut (pole) - kolumna relacji(tabeli) opatrzona nazwą. Atrybuty mogą się pojawić w relacji w dowolnej kolejności bez wpływu na znaczenie relacji. Z każdym atrybutem związana jest para (nazwa atrybutu, typ danych). Przyjęło się, że klucz główny (podstawowy) jest pierwszym atrybutem od lewej strony, a klucz obcy pierwszym od prawej.

Dziedzina - zbiór dopuszczalnych wartości dla jednego lub większej liczby atrybutów.

Krotka - wiersz relacji (rekord najczęściej używany w acces).

Id_m int	n_m var	l_m	
1			
2			

Relacyjna baza danych - zbiór znormalizowanych relacji(tabel) o różnych nazwach (relacyjny system zarządzania bazą danych określa jedynie, że taka baza jest dla użytkownika zbiorem relacji (tabel), czyli odnosi się do warstwy konceptualnej i zewnętrznej modelu ANSI).

Encja (ang. *entity*) jest rzeczą lub obiektem mającym dla nas znaczenie rzeczywiste bądź wyobrażone, o którym informacje muszą być znane lub przechowywane(inaczej wczesna wersja tabeli). Nazwa encji powinna być podana w liczbie pojedynczej i zapisana WIELKIMI LITERAMI.

Encją może być:
- obiekt fizyczny (np. samochód, bilet lotniczy)
- obiekt niematerialny (np. konto bankowe, zamówienie)
- zdarzenie (np. urlop pracownika, sprzedaż samochodu)
- istniejący fakt (np. znajomość języków obcych)

Relacyjna baza danych jest zbiorem relacji(tabel**) o następujących własnościach:**

- każda relacja (tabela) w bazie danych jest **jednoznacznie określona przez swoją nazwę**.
- każda kolumna w relacji (atrybut) ma **jednoznaczną nazwę** w ramach tej relacji.
- kolumny relacji tworzą **zbiór nieuporządkowany**, mogą być wpisywane **w dowolny sposób**.
- wszystkie wartości w danej kolumnie muszą być **tego samego typu** (pochodzić z tej samej dziedziny).

- każdy wiersz (krotka) **jest inny** - nie ma duplikatów krotek.
- teoretycznie, kolejność wierszy **nie ma znaczenia** (w praktyce może mieć to wpływ na efektywność wyszukiwania odpowiednich grup krotek).
- każde pole (przecięcie kolumny z wierszem) zawiera **wartość elementarną** (**atomową**) z dziedziny określonej przez kolumnę. Brakowi wartości odpowiada wartość specjalna NULL,
- każda relacja zawiera **klucz główny** (**podstawowy**) - kolumnę (lub kolumny), której wartości **jednoznacznie identyfikują wiersz** (a więc w szczególności **nie powtarzają się**). Wartością klucza głównego nie może być NULL. Każda tabela musi mieć dokładnie jeden klucz główny.

Rodzaje kluczy

superklucz (ang. *superkey*) - kolumna lub zestaw kolumn jednoznacznie identyfikujących każdą krotkę w tabeli
klucz podstawowy (**główny**) (PRIMARY KEY) - klucz wybrany przez projektanta bazy danych, aby unikatowo identyfikować krotki tabeli
klucz kandydujący (**potencjalny**), zwany w skrócie *kluczem*
klucz obcy - klucz w podrzędnej tabeli powiązany z kluczem głównym w nadrzędnej tabeli celem zapewnienia integralności referencyjnej bazy danych. Nigdy nie sam. Zawsze musi być powiązany z kluczem podst.
klucz prosty - klucz oparty na jednej kolumnie
klucz złożony - klucz oparty na wielu kolumnach
klucz sztuczny - pole sztucznie wprowadzone do relacji, aby stał się kluczem podstawowym
naturalny – występujący naturalnie w encji

Właściwości klucza głównego

trwałość - pole kluczowe nie może być puste i nie może zawierać wartości NULL
unikatowość - dane w polu kluczowym nie mogą się powtarzać (indeksowanie bez powtórzeń)

stabilność - dane w polu kluczowym nie mogą się zmienić; nie powinno się jako klucz głównych używać kolumn przechowujących dane nietrwałe (np. numer telefonu - może się zmienić lub zostać przypisany innej osobie)

niereduko walność - żaden podzbiór właściwy pól tworzących pole klucza nie jest kluczem

Atrybuty, które należą do klucza nazywać będziemy **atrybutami kluczowymi**, a te, które nie należą do klucza - **atrybutami niekluczowymi**.