

## Modele danych. Integralność danych.

**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 między 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

W tym modelu wszystkie dane przechowywane są w jednym arkuszu, tabeli, pliku.

**Zaletami** tego modelu są łatwość i szybkość odczytywania interesujących nas danych- np. wystarczy tylko znaleźć rekord opisujący szukany zakup, żeby poznać wszystkie szczegóły operacji.

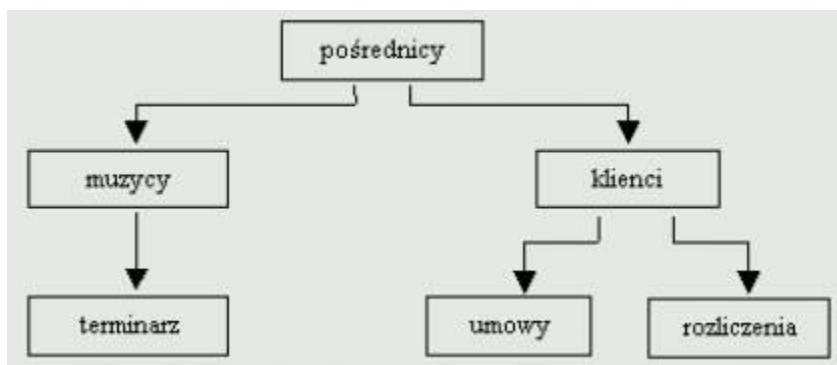
**Wadą** modelu jednorodnego jest duża liczba duplikatów np. nazwy dostawcy, jego adres, telefon wypisywane są tyle razy ile razy kupiliśmy u niego towar. Wielokrotnie zapisywanie tych samych danych nie tylko wymaga więcej miejsca na dysku i w pamięci, ale także utrudnia modyfikowanie danych(gdyby firma zmieniła adres) i zwiększa ryzyko wpisania błędnych danych.

## Hierarchiczny model danych

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:

- jeden korzeń (tabela nadrzędna)
- synowie (tabele podrzędne).

Hierarchiczny model danych (do końca lat 70.) opierał się na strukturze drzewa (jeden wyróżniony wierzchołek), np.



**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. Model hierarchiczny opiera się zatem 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.

Używany był, gdy bazy danych były tworzone w językach wysokiego poziomu jak C, C++, itd., kiedy nie było języków baz danych.

Aby uzyskać dostęp do danych użytkownik zaczyna od korzenia i przedziera się przez całe drzewo danych, aż do interesującego miejsca. Oznacza to zarazem, że użytkownik musi dobrze znać strukturę b.d. Struktura ta jest podobna do znanych wszystkim stron WWW.

Nie można dopisać żadnego z muzyków dopóki nie powiążemy go z którymś z pośredników

Nie istnieje tu relacja wiele-do-wielu ponieważ jeden muzyk gra dla wielu klientów, a jeden klient może zamówić wielu muzyków. Dane o klientach muszą być zawarte w tabeli terminarz (obok danych o muzykach), a w tabeli klienci będą dane o muzykach (naturalnie obok danych o klientach). Mamy więc do czynienia z **Nadmiarowością danych!!!**

## Sięciowy model:

Pewna modyfikacja modelu hierarchicznego - dane można przedstawić w postaci grafu. Ten model wyszedł z obiegu. Opierał się na systemie plików.

Sięciowy model bazy danych (SMBD) został stworzony głównie w celu rozwiązania problemów związanych z modelem hierarchicznym. Podobnie, jak w modelu hierarchicznym SMBD można sobie wyobrazić jako odwrócone drzewo. Różnica polega jednak na tym, że w przypadku SMBD, wiele drzew może dzielić ze sobą gałęzie, a każde z nich stanowi część ogólnej struktury bazy danych.



Diagram modelu sieciowego.

## Obiektowy model danych

Bazy opierające się na modelu obiektowym łączą cechy programów komputerowych tworzonych za pomocą nowoczesnych języków programowania obiektowego z cechami aplikacji bazodanowych. Aplikacje bazodanowe bazują na obiektach (zbiorach połączonych danych i procedur umożliwiających manipulowanie tymi danymi) i tzw. klasach obiektów.

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 baz danych mającym obiektowe rozszerzenie jest **PostgreSQL**.

Obiektowe bazy danych do przechowywania danych używają obiektów posiadających swoją tożsamość (tożsamość obiektu – *identity* – która wyznacza jego identyfikator). W obiektowej bazie danych nie może być dwóch identycznych obiektów o identycznych identyfikatorach. Obiekty charakteryzujące się tymi samymi metodami i atrybutami są instancjami tej samej klasy stanowiącej dla nich model. Zwykle fragmentem definicji klasy są atrybuty, które w rozumieniu obiektowego modelu danych odpowiadają atrybutom (kolumnom) relacyjnej bazy danych. Podobnie jak w programowaniu obiektowym, klasy mają przypisane funkcje nazywane metodami, działające w obrębie obiektu. Obiektowy model danych zawiera również koncepcję hermetyzacji, hierarchii klas i dziedziczenia.

### **3. Relacyjny model:**

W latach 60. dr E F Codd pracując w IBM stworzył relacyjny model b.d. opierając się dwóch gałęziach matematyki - teorii mnogości i rachunku predykatów I-rzędu! W relacyjnych b.d. dane przechowujemy w tabelach. Każda z tabel składa się z rekordów oraz pól. Fizyczna kolejność pól i rekordów jest tutaj bez znaczenia. Każdy rekord jest wyróżniony przez unikatową wartość - klucz. W konsekwencji użytkownik nie musi znać fizycznego położenia rekordu, który chce odczytać. Odróżnia to model relacyjny od hierarchicznego czy sieciowego, gdzie b. duży nacisk kładziono na struktury, które użytkownik musiał opanować, aby mógł odczytać interesujące dane.

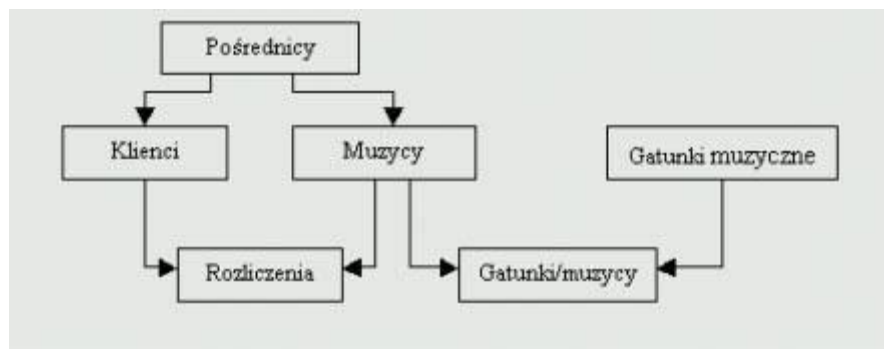


Diagram logicznego modelu relacyjnego.

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 Codd'a 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 Codd'a.

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. Podstawą relacyjnego modelu baz danych jest teoriomnogościowe pojęcie relacji. Dane przechowuje się w tabelach, nazywanych **relacjami**, składających się z **wierszy (krotek)** i **kolumn (atrybutów)**.

Frank Codd określił 12 podstawowych zasad, które musiał spełniać system, by mógł zarządzać relacyjnym modelem danych.

## Postulaty Codda

System musi być kwalifikowany jako relacyjny, jako baza danych i jako system zarządzania:

1. **Postulat informacyjny** – dane są reprezentowane jedynie poprzez wartości atrybutów wierszach tabel.
2. **Postulat dostępu** – każda wartość w bazie danych jest dostępna poprzez podanie nazwy tabeli, atrybutu i wartości klucza podstawowego.
3. **Postulat dotyczący wartości NULL** - dostępna jest specjalna wartość NULL dla reprezentacji zarówno wartości nieokreślonej, jak i nieadekwatnej, inna od wszystkich i podlegająca przetwarzaniu.
4. **Postulat dotyczący katalogu** – wymaga się, aby system obsługiwał wbudowany katalog relacyjny z bieżącym dostępem dla uprawnionych użytkowników używających języka zapytań,
5. **Postulat języka danych** – system musi dostarczać pełny język przetwarzania danych, który może być używany zarówno w trybie interaktywnym, jak i w obrębie programów aplikacyjnych, obsługuje operacje definiowania danych, operacje manipulowania danymi, ograniczenia związane z bezpieczeństwem i integralnością oraz operacje zarządzania transakcjami.
6. **Postulat modyfikowalności perspektyw** – system musi umożliwiać modyfikowanie perspektyw, o ile jest ono semantycznie realizowalne.
7. **Postulat modyfikowalności danych** – system musi umożliwiać operacje modyfikacji danych, musi obsługiwać operatory INSERT, UPDATE oraz DELETE.
8. **Postulat fizycznej niezależności danych** – zmiany fizycznej reprezentacji danych i organizacji dostępu nie wpływają na aplikacje. Postulat logicznej niezależności danych – zmiany wartości w tabelach nie wpływają na aplikacje.
9. **Postulat logicznej niezależności danych** – zmiany wartości w tabelach nie wpływają na aplikacje.
10. **Postulat niezależności więzów spójności** – więzy spójności są definiowane w bazie i nie zależą od aplikacji.
11. **Postulat niezależności dystrybucyjnej** – działanie aplikacji nie zależy od modyfikacji i dystrybucji bazy.
12. **Postulat bezpieczeństwa względem operacji niskiego poziomu** — operacje niskiego poziomu nie mogą naruszać modelu relacyjnego i więzów spójności.

## Integralność danych

**Integralność danych**, określana również mianem spójności danych, jest to funkcja SZBD, która gwarantuje, że dane nie zostaną usunięte lub zmienione w nieautoryzowany sposób.

**Integralność** (ang. data integrity) to formalna poprawność bazy danych, jej fizycznej organizacji, zgodność ze schematem bazy danych i regułami dostępu.

Ochrona integralności danych polega również na zapewnieniu, że dane nie ulegną zniekształceniu podczas wykonywania na nich operacji. Spójność danych związana jest z ich dokładnością – dane dokładnie odzwierciedlają modelowaną rzeczywistość. Oznacza ona również ich prawdziwość oraz aktualizowanie, gdy zmienia się rzeczywistość modelowana w bazie danych. Dane muszą być poprawne i zgodne ze schematem bazy danych. W SZBD powinny istnieć mechanizmy, które pozwolą zabezpieczyć dane przed skutkami awarii zasilania, sprzętu lub oprogramowania. W bazie danych powinny działać mechanizmy, których zadaniem jest zabezpieczenie danych przed następstwami błędów logicznych. Zachowanie spójności danych powinny gwarantować systemy chroniące dane też przed błędami pojawiającymi się w chwilach współbieżnego dostępu do tej samej informacji. Istotną rolę odgrywa również system kontroli danych wejściowych. Proces utrzymania integralności bazy danych obejmuje również kopie zapasowe danych.

**Zachowanie poprawności bazy danych** opiera się na utrzymaniu poprawności w obrębie semantycznym, encji i referencyjnym.

**Integralność semantyczna** polega na utrzymaniu ograniczeń nakładanych na dane, min.:

- w określonej kolumnie tabeli muszą znajdować się wyłącznie dane zgodne z typem danych kolumny, np. tylko liczby całkowite;
- w kolumnie nie mogą wystąpić braki wartości – puste miejsca NULL.

Obok integralności semantycznej wyróżniamy także integralność encji.

**Integralność encji** wprowadza się w trakcie definiowania schematu danych. Zgodnie z regułami integralności encji każda tabela powinna mieć **klucz główny** – kolumnę, w której nie mogą wystąpić wartości NULL oraz w której dane nie mogą się powtarzać.

**Integralność referencyjna** polega na wprowadzeniu i utrzymaniu powiązań pomiędzy tabelami. Związki te tworzy się przez umieszczenie kolumny pełniącej rolę **klucza głównego tabeli** w innej tabeli, co nadaje kolumnie funkcję klucza obcego. Takie rozwiązanie nazwę związków: klucz **podstawowy** – **klucz obcy**. Reguły integralności referencyjnej determinują, czy wartości klucza obcego mają odpowiadać wartościom klucza głównego w powiązanej tabeli, czy mogą przyjmować wartości **NULL**.

Podczas omawiania zagadnienia integralności bazy danych stosuje się również podział więzów integralności na statyczne i dynamiczne.

**Ograniczenia integralnościowe statyczne** odnoszą się do bieżącego stanu bazy danych, np. na kolumnę wiek zostało nałożone takie ograniczenie jak **CHECK** (wiek < 200). Nałożenie tego ograniczenia podczas tworzenia tabeli spowoduje, że w kolumnie wiek wartość nie będzie większa niż 200 w trakcie nakładania ograniczenia i w przyszłości.

**Więzy integralności dynamiczne** to takie, które przeciwdziałają zmianom, ponieważ związane są z przejściem bazy danych z jednego stanu w drugi. Oznacza to, że odnoszą się do bazy danych w aspekcie temporalnym. Przykładem takich więzów może być wymaganie, aby wiek pracowników nigdy nie malał. Jeśli wiek pracownika po pewnym czasie wzrośnie, to wartość kolumny wiek nie może nigdy się zmniejszyć, ponieważ nikt nie staje się młodszy. Więzy dynamiczne nazywane są również więzami przejść.

### **SPRAWDŹ SWOJĄ WIEDZĘ**

1. Co to jest integralność danych?
  2. Jakich wyróżniamy rodzaje integralności?
  3. Czym różnią się ograniczenia integralnościowe statyczne od dynamicznych?
  4. Podaj przykład integralności encji w przypadku, gdy tabelą byłby spis uczniów w dzienniku szkolnym.
-

Notatka do wklejenia (opanu) zagadnienia):

**INTEGRALNOŚĆ DANYCH**- określana również mianem spójności danych, jest to funkcja SZBD, która gwarantuje, że dane nie zostaną usunięte lub zmienione w nieautoryzowany sposób.

**Integralność** (ang. data integrity) to formalna poprawność bazy danych, jej fizycznej organizacji, zgodność ze schematem bazy danych i regułami dostępu.

Zachowanie poprawności bazy danych opiera się na utrzymaniu poprawności w obrębie semantycznym, encji i referencyjnym.

- **Integralność semantyczna** polega na utrzymaniu ograniczeń nakładanych na dane, min.:
  - w określonej kolumnie tabeli muszą znajdować się wyłącznie dane zgodne z typem danych kolumny, np. tylko liczby całkowite;
  - w kolumnie nie mogą wystąpić braki wartości – puste miejsca NULL.
- **Integralność encji** wprowadza się w trakcie definiowania schematu danych. Zgodnie z regułami integralności encji każda tabela powinna mieć klucz główny – kolumnę, w której nie mogą wystąpić wartości NULL oraz w której dane nie mogą się powtarzać.
- **Integralność referencyjna** polega na wprowadzeniu i utrzymaniu powiązań pomiędzy tabelami. Związki te tworzy się przez umieszczenie kolumny pełniącej rolę klucza głównego tabeli w innej tabeli, co nadaje kolumnie funkcję klucza obcego. Takie rozwiązanie nazwę związków: klucz podstawowy – klucz obcy. Reguły integralności referencyjnej determinują, czy wartości klucza obcego mają odpowiadać wartościom klucza głównego w powiązanej tabeli, czy mogą przyjmować wartości NULL.

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

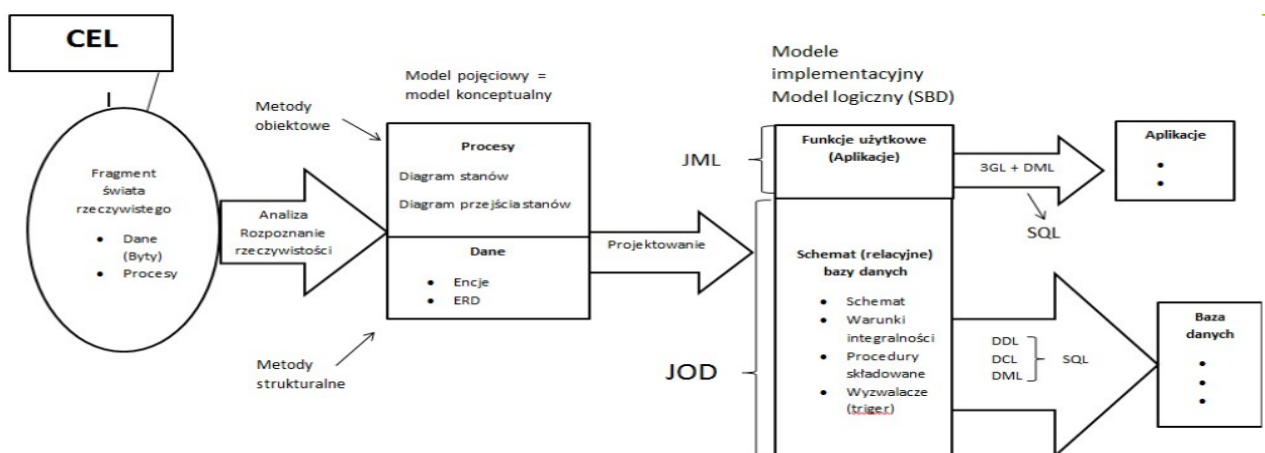
**Model danych** (ang. data base model) to zestaw pojęć do opisu świata rzeczywistego. Jest to swego rodzaju metajęzyk, w którym analityk systemu formułuje swoją wizję systemu informatycznego.

**Model danych** to notacja służąca do opisu danych lub informacji. Zwykle składa się z trzech części:

- **Struktura danych.** (Struktury danych używane do implementowania informacji w komputerach są w kontekście systemów baz danych nazywane czasem fizycznym modelem danych. W świecie baz danych modele danych znajdują się nieco na wyższym poziomie niż struktury danych i czasem nazywane są modelami pojęciowymi).
- **Operacje na danych.**
- **Wiązy danych.** (Modele danych umożliwiają nakładanie więzów na informacje. Istnieją różne rodzaje więzów np. nakładane na dzień tygodnia od 1 do 7).

Model danych określa dostęp i sposób zapisu danych. W podstawowej klasyfikacji modeli danych wyróżnia się:

- Modele pojęciowe (ang. conceptual model). Wśród modeli pojęciowych można wyróżnić model związków encji (ERM-Entity-Relationship Model), unifikowany język modelowania UML, oraz język definicji obiektów ODL).
- Modele logiczne (ang. logical model).



**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. Structured Query Language – „strukturalny język zapytań”)

Drugi podział modeli danych można wyróżnić następujące kategorie:

- **Koncepcyjne modele danych** -są to modele najbardziej zbliżone poziomem abstrakcji do wymagań projektantów BD, stosowane w pierwszych etapach projektów, w celu weryfikacji wyróżnionych w nim obiektów i związków między nimi
- **Implementacyjne modele danych**, stosowane do transformacji wcześniej przygotowanego modelu koncepcyjnego do konkretnego modelu BD. Wśród modeli implementacyjnych wyróżniamy modele:
  - Jednorodny
  - Hierarchiczny
  - Sieciowy
  - Relacyjny
  - Obiektowy
  - Obiektowo-relacyjny
- **Fizyczne modele danych**, określające sposoby organizacji danych w pamięci zewnętrznej komputerów. Operuje się tu pojęciami takimi jak: rekord, plik, adres.

**Model relacyjny**(RDBMS, relacyjny system zarządzania bazą danych):W latach 60. dr E F Codd pracując w IBM stworzył relacyjny model b.d. opierając się dwóch gałęziach matematyki - teorii mnogości i rachunku predykatów 1-rzędu! W relacyjnych b.d. dane przechowujemy w dwuwymiarowych tabelach (wiersze i kolumny). Każda z tabel składa się z rekordów oraz pól .Fizyczna kolejność pól i rekordów jest tutaj bez znaczenia. Każdy rekord jest wyróżniony przez unikatową wartość – klucz.

