Systemy Baz Danych

Wykład I

Podstawy relacyjnych baz danych

Powtórzenie wiadomości

Materiał wykładu

Wykład zawiera przegląd wiadomości podstawowych, dotyczących modelu relacyjnego danych baz danych zbudowanych według założeń tego modelu. Stanowi powtórzenie materiału wykładanego w ramach przedmiotu Relacyjne Bazy Danych dla studentów studiów inżynierskich Wydziału Informatyki PJWSTK.

Wykład jest przeznaczony dla studentów przedmiotu **Systemy Baz Danych** prowadzonego dla studiów inżynierskich Wydziału Informatyki PJWSTK.

Baza danych – informacje podstawowe

Dane stanowią jeden z zasobów firmy (instytucji, organizacji), obok kapitału, pracowników, infrastruktury. Umożliwiają działanie organizacji, kontakty z otoczeniem, ale tak jak i pozostałe zasoby, wymagają *zarządzania* i *konserwacji*.

Zarządzanie danymi jest realizowane przez **system informacyjny**, który zaspokaja zapotrzebowanie na informacje o pewnym fragmencie rzeczywistości.

Baza danych stanowi niemal zawsze część tego systemu, odpowiedzialną za przechowywanie, bezpieczeństwo, zarządzanie danymi i ich udostępnianie.

Baza danych – informacje podstawowe

Istnieje dualizm pojęciowy. Przez bazę danych rozumiemy albo

- **System Zarządzania Danymi** - ich strukturą, bezpieczeństwem, zasadami dostępu (**SZBD**),

albo

- **Kolekcja Danych (DB).**

Baza danych stała się standardową metodą wprowadzenia *struktury i reguł* do procesu zarządzania danymi. Te struktury i zasady ewoluują wraz z rozwojem technologii informatycznych.

W dalszej części wykładu, pod pojęciem baza danych będziemy (na ogół) rozumieli pierwszą z powyższych wersji, czyli *System Zarządzania Danymi.*

Relacyjny model danych

Koncepcja modelu relacyjnego została stworzona przez Edgara Codd'a i po raz pierwszy opublikowana w 1970 r. Teoretyczna koncepcja modelu opiera się na pojęciu relacji oraz na algebrze relacji.

W matematyce **relację** definiuje się jako podzbiór **iloczynu kartezjańskiego**.

Iloczynem kartezjańskim *zbiorów A i B nazywamy zbiór wszystkich par uporządkowanych (x,y) takich, że x należy do A i y należy do B i oznaczamy A×B.*

$$A \times B = \{(x,y): i \}$$

Iloczyn kartezjański nie jest przemienny: $A \times B \neq B \times A$

Pojedynczą parę (x, y) nazywamy krotką. Jeżeli iloczyn kartezjański zostanie określony na **n** zbiorach, każda krotka będzie się składała z **n** elementów.

Fizyczną interpretacją relacji jest tabela – jeśli relacja jest podzbiorem iloczynu kartezjańskiego określonego na n zbiorach, tabela będzie zbudowana z n kolumn, a w każdej z kolumn pojawią się elementy każdego ze zbiorów.

Relacyjny model danych – postulaty Codd'a

Definicję relacyjnej bazy danych Codd sprowadził do 13 reguł, zwanych postulatami Codd'a których spełnienie pozwalało określać model zapisu danych jako „bazę relacyjną".

Postulat 0.

System zarządzania bazą danych można uznać za system relacyjny (RDBMS) wówczas, gdy w celu zarządzania bazą danych wykorzystuje wyłącznie rozwiązania zgodne z modelem relacyjnym.

1. **Postulat informacyjny.**

Na poziomie logicznym wszystkie dane (łącznie z nazwami tabel i kolumn), reprezentowane są wyłącznie za pomocą wartości w tabelach.

2. **Postulat gwarancji dostępu**

Do każdej pojedynczej danej dostęp możliwy jest poprzez nazwę tabeli, nazwę kolumny oraz wartość (wartości) klucza głównego.

Relacyjny model danych – postulaty Codd'a

3. **Postulat obiektu null**

W systemie zarządzania bazą danych (SZBD) dostępny jest specjalny obiekt null reprezentujący stan braku wartości (tj. reprezentujący wartość brakującą, nieokreśloną lub nieznaną) – różny od każdej konkretnej wartości (jak 0 lub napis o zerowej długości), oraz niezależny od typu danych.

4. **Postulat struktury metadanych**

Informacje o obiektach bazy danych tworzących schemat bazy danych są na poziomie logicznym reprezentowane za pomocą tabel i dostępne tak jak każde inne dane.

5. **Postulat pełnego języka danych**

W SZBD zaimplementowany jest pełny język, obejmujący definiowanie tabel, perspektyw, więzów spójności, operowanie danymi (zarówno na poziomie interaktywnym jak też przez interfejs programistyczny), nadawanie uprawnień użytkownikom, przeprowadzanie na bazie danych operacji pogrupowanych w transakcje.

Relacyjny model danych – postulaty Codd'a

6. **Postulat modyfikowania danych poprzez perspektywy.**

SZBD umożliwia modyfikowanie danych przy użyciu perspektyw w sytuacji, gdy taka modyfikacja jest semantycznie sensowna.

7. **Postulat modyfikowania danych na wysokim poziomie abstrakcji.**

SZBD umożliwia modyfikowanie danych za pomocą operacji, których argumentami są tabele (perspektywy) – a więc nie tylko w sposób nawigacyjny, polegający na przejściu wszystkich wierszy (rekordów) w tabeli (perspektywie).

8. **Postulat fizycznej niezależność danych.**

Zmiany w metodach zapisu danych i dostępu do nich nie mają wpływu na aplikację.

9. **Postulat logicznej niezależność danych.**

Zmiany w tabelach zachowujące informacje i poprawne semantycznie, nie mają wpływu na aplikację.

Relacyjny model danych – postulaty Codd'a

10. Postulat niezależności więzów spójności.

Więzy spójności są definiowane w języku bazy danych (nie muszą być wyrażane w aplikacji).

11. Postulat niezależności dystrybucyjnej.

SZBD (i jego język) umożliwiają używanie danych zapisanych w różnych fizycznie miejscach (w różnych węzłach sieci).

12. Postulat zabezpieczenia przed operacjami na niższych poziomach abstrakcji.

Jeżeli system umożliwia operacje na niższych poziomach abstrakcji, nie mogą one naruszać relacyjnego modelu danych (w tym nie mogą omijać ograniczeń określonych przez więzy spójności).

Więzy spójności (constraints)

Więzy spójności to zbiór reguł, których zadaniem jest zagwarantowanie logicznej poprawności danych przechowywanych w bazie. Szczególną cechą więzów spójności jest to, że są definiowane na etapie projektowania bazy danych, a za ich realizację (przestrzeganie reguł) odpowiada SZBD.

Więzy integralności mogą być definiowane przez deklaracje języka SQL, wyzwalacze, procedury, indeksy.

Więzy integralności deklarowane w SQL

Więzy klucza głównego

Rolą klucza głównego jest jednoznaczna identyfikacja wiersza (rekordu). W tabeli relacyjnej bazy danych każdy wiersz musi być unikalny (to wynika z teorii). Ponieważ język SQL dopuszcza istnienie w tabeli identycznych rekordów, zatem istnieje konieczność dodatkowego zagwarantowania ich unikalności. Jeżeli żaden z atrybutów opisywanych przez dane w tabeli nie spełnia wymogu unikalności, stosuje się dodatkową kolumnę, na ogół wyposażając ją w mechanizm gwarantujący wprowadzanie wartości unikalnych. Klucz główny może zostać utworzony na więcej niż jednej kolumnie. Wówczas unikalne być muszą wartości poszczególnych jego kolumn składowych, traktowane jako całość. Taki klucz nazywa się kluczem złożonym. Klucz główny w tabeli może być tylko jeden.

Warunki konieczne klucza głównego:

- Unikalność wartości
- Niedopuszczalność wystąpienia NULL

Więzy integralności deklarowane w SQL

Więzy UNIQUE (nazywane też więzami klucza unikalnego)

- wszystkie wartości znaczące są unikalne,
- dopuszczalne jest wystąpienie NULL.

Nie wszystkie SZBD prawidłowo realizują postulat dopuszczalności NULL. W tabeli więzy UNIQUE mogą być zadeklarowanych wobec więcej niż jednej kolumny. Realizacją więzów UNIQUE jest utworzenie indeksu (na ogół nieposortowanego).

Więzy NOT NULL

Niedopuszczalność wystąpienia wartości NULL w zadeklarowanej kolumnie

Więzy CHECK

Istnieje możliwość zadeklarowania wartości, które są dopuszczalne dla kolumny. Mogą zostać zadeklarowane dla kolumny np. jako przedział wartości dopuszczalnych, lub na poziomie tabeli, przez porównywanie wartości pochodzących z różnych kolumn.

Więzy referencyjne (referential integrity)

Więzy referencyjne, to więzy wynikające z utworzenia powiązań pomiędzy tabelami w układzie **Klucz główny -> klucz obcy**. Istota tych więzów sprowadza się do stwierdzenia, że:

Wartości klucza obcego tabeli po stronie „wiele" związku tabel mogą pochodzić wyłącznie ze zbioru wartości użytych w roli klucza głównego tabeli po stronie „jeden" związku, lub (jeśli założenia biznesowe to dopuszczają) być NULL.

Więzy referencyjne są automatycznie przestrzegane przez SZBD po zadeklarowaniu związku pomiędzy tabelami.

Indeks

Indeks jest to dodatkowa struktura danych umożliwiająca szybki dostęp do wierszy tabeli na podstawie wartości w określonej kolumnie lub kolumnach. Inaczej – jest to mechanizm wiążący wartości logiczne z ich adresami fizycznymi (miejscem zapisu na dysku). Funkcja indeksu przypomina funkcję indeksu (skorowidza) w książce.

Np. indeks zbudowany na kolumnie Nazwisko umożliwia szybkie wyszukiwanie danych wykładowcy w oparciu o jego nazwisko.

W tabeli może znajdować się jeden indeks posortowany, oraz kilka indeksów niesortowanych.

Więcej informacji na temat funkcji i budowy indeksów w relacyjnych bazach danych zostanie przedstawione w kolejnych wykładach przedmiotu SBD.

Projektowanie baz danych - encja

Na etapie projektowania baz danych posługujemy się narzędziami i pojęciami pozwalającymi na stosowanie pewnych uproszczeń. Ponieważ baza danych ma z założenia przechowywać dane stanowiące opis pewnego fragmentu rzeczywistości, zatem jej model powinien z jednej strony stanowić obraz obiektów i zależności pomiędzy nimi, które w tej rzeczywistości występują, z drugiej strony powinien być odzwierciedleniem struktury przyszłych tabel w bazie danych.

Modelem bazy danych jest **diagram związków encji**. Encja to model klasy obiektów opisywanych atrybutami o zdefiniowanych typach danych, a jednocześnie model tabeli bazodanowej, w którym atrybuty encji stanowią model kolumn.

Instancja encji to pojedyncze wystąpienie obiektu (egzemplarz) danej klasy.

Zidentyfikowanie wszystkich obiektów, o których informacje będą przechowywane w bazie danych, to rola etapu analizy wstępnej. Rolą diagramu związków encji jest utworzenie modeli tych obiektów oraz istniejących pomiędzy nimi zależności. Diagram ma postać graficzną, ułatwiającą zrozumienie struktury bazy, która bywa skomplikowana. Dotyczy to zwłaszcza związków zachodzących pomiędzy obiektami.

Projektowanie baz danych - związek

Związek to uporządkowana lista encji. Poszczególne encje mogą na niej występować wielokrotnie. Każdy związek określa pewną zależność między zbiorami egzemplarzy (instancji) encji wchodzącymi w skład związku - instancję związku.

Związek można formalnie zapisać przy użyciu notacji relacyjnej:

Z(E1 ,...,En)

co oznacza: encje **E1** ,...,**En** wchodzą w skład związku **Z**

Związek binarny to uporządkowana para encji – np. encji **Student** i **Miasto** (urodzenia).

Instancja związku binarnego jest dwuargumentową relacją na iloczynie kartezjańskim zbiorów instancji encji - w naszym przykładzie zbioru studentów, ze zbiorem miast.

Związek jednoznaczny

Gdy instancja związku binarnego jest funkcją, związek nazywa się jednoznaczny.

W przytoczonym przykładzie związek ten opisuje miejsca (miasta) urodzenia studentów. Instancja związku jednoznacznego między encjami **Student** i **Miasto** jest **funkcją częściową** ze zbioru studentów w zbiór miast, jeśli nie znamy miejsc urodzenia wszystkich studentów. Przy założeniu, że ta informacja jest znana dla każdego studenta, będziemy mieli do czynienia z funkcją zupełną ze zbioru studentów w zbiór miast.

Część związku odpowiadająca dziedzinie funkcji jest nazywana **stroną wiele** związku (lub **encją podrzędną**), a część odpowiadająca przeciwdziedzinie funkcji **stroną jeden** związku (lub **encją nadrzędną**).

Miasto jest encją po stronie jeden (nadrzędną), a **Student** encją po stronie wiele (podrzędną).

Powszechnie używaną nazwą związku jednoznacznego jest **związek jeden – do – wiele**.

Implementacja związku jednoznacznego

Dwie encje, połączone związkiem jednoznacznym, są implementowane odpowiednio przez dwie tabele.

W encji po stronie wiele jest dodany nowy atrybut - klucz obcy, określający powiązanie z instancją encji po stronie jeden. W tabeli zostanie on zaimplementowany jako dodatkowa kolumna.

W naszym przykładzie, do encji **Student** jest dodany klucz obcy, określający powiązanie z instancją encji **Miasto** - przez wartość jej klucza głównego.

Właściwości związku

Związek posiada właściwości:

- Nazwę związku
- Liczność związku (cardinality)
- Typ związku (*identyfikujący* lub *nieidentyfikujący*)
- Opcjonalność
- Akcje referencyjne

Związek niejednoznaczny

Związek niejednoznaczny to taki związek, który nie jest jednoznaczny (!), czyli nie jest funkcją. Definiując związek binarny użyliśmy sformułowania „uporządkowana para encji”. W przypadku związku niejednoznacznego nie jesteśmy w stanie takiej pary uporządkować, czyli wskazać poprzednika i następnika.

Relacyjny model danych nie przewiduje innej reprezentacji związków pomiędzy encjami niż ta, którą dotychczas poznaliśmy, zatem każdy związek niejednoznaczny trzeba sprowadzić do związków jednoznacznych.

Transformacja związku niejednoznacznego

Dla niejednoznacznego związku binarnego **Z(E1,E2)** wprowadzamy nową encję **E0** i dwa związki jednoznaczne **Z1(E0,E1)** oraz **Z2(E0,E2)** łączące nowy zbiór encji ze starymi. Klucz encji **E0** jest sumą kluczy encji **E1** i **E2.**

Dla związku o liczbie argumentów większej niż dwa **Z(E1 ,...,En), n > 2** wprowadzamy nową encję **E0** i **n** jednoznacznych związków binarnych **Zi(E0,Ei)** łączących nową encję z encjami już istniejącymi. Klucz encji **E0** jest sumą kluczy encji **E1,...,En**.

Wprowadzana encja reprezentująca związek nazywa się encją asocjacyjną. Jest ona zawsze encją słabą (zależną), bo związki łączące ją z encjami - argumentami związku niejednoznacznego są związkami identyfikującymi.

Szczególne przypadki związków

Związek rekurencyjny

Jeżeli w związku jedna encja występuje wielokrotnie, mamy do czynienia ze związkiem rekurencyjnym. Ten typ związku nie może być związkiem identyfikującym, musi być również związkiem opcjonalnym, czyli musi dopuszczać NULL jako wartość klucza obcego.

Związek jedno-jednoznaczny

Związek ten, zwany powszechnie **związkiem jeden–do–jeden**, to szczególny przypadek związku jednoznacznego. Jest to taki związek jednoznaczny, którego instancja jest różnowartościową funkcją częściową 1 - 1 . Inaczej mówiąc, każdej instancji encji po stronie „jeden” odpowiada co najwyżej jedna instancja encji po stronie „wiele”.

Implementacja takiego związku na tabele może odbywać się na kilka sposobów – może być reprezentowany przez jedną, dwie lub trzy tabele.

Postulat normalizacji

Każdy fakt przechowywany w bazie danych powinien być wyrażalny w niej tylko na jeden sposób.

Niespełnienie tego postulatu prowadzi do istnienia **redundancji**, czyli sytuacji w której informacja dotycząca jednego faktu odnotowywana jest więcej niż jeden raz, co z kolei może skutkować powstaniem błędów i wewnętrznej niespójności danych.

Przedstawienie normalizacji wymaga zastosowania formalnego, matematycznego modelu relacji. W tym wykładzie przedstawiamy go w wersji uproszczonej.

Przeprowadzenie procesu normalizacji bazy danych sprowadza się do stworzenia takiego jej schematu, w którym nie pojawią się struktury mogące powodować przechowywanie danych redundantnych, także anomalii przy wstawianiu, usuwaniu i modyfikacji danych.

Pierwsza postać normalna

Pojęcie **pierwszej postaci normalnej** sprowadza się do wymogu, aby na przecięciu wiersza i kolumny tabeli (czyli w pojedynczej komórce) znajdowała się jedna, niepodzielna wartość (atomowa), nie będąca ciągiem ani zbiorem.

Wymóg ten wynika wprost z definicji relacji, i zawarty jest w postulacie gwarancji dostępu zakładającym, że znajomość nazwy tabeli, kolumny i identyfikatora wiersza (klucza) pozwala na jednoznaczne odczytanie zapisanej danej.

Zależność funkcyjna

Relacja r o schemacie **R = {A1, A2,..., An}** spełnia zależność funkcyjną

X => Y (**X, Y** - podzbiory **R**)

jeśli dla każdych dwóch krotek t, u relacji r zachodzi warunek:

Jeśli t|**X** = u|**X** to t|**Y** = u|**Y**

tzn. w ramach krotek relacji r wartości atrybutów zbioru **X** determinują jednoznacznie wartości atrybutów zbioru **Y.**

Można to też opisać w pewnym uproszczeniu: zależność funkcyjna to sytuacja, w której wartość jednego zbioru atrybutów determinuje inny zbiór atrybutów.

Nadklucz relacji

Nadkluczem relacji r o schemacie **R** = {***A1, A2,..., An***}

nazywamy każdy dowolny zbiór atrybutów

X z **R**

taki, że zachodzi zależność funkcyjna

X => **R**

Inaczej mówiąc, wartość każdego atrybutu jest jednoznacznie zdeterminowana przez wartości atrybutów zbioru ***X***. Jednym z nadkluczy jest zawsze zbiór wszystkich atrybutów relacji **R**.

Klucz relacji

Kluczem relacji r o schemacie **R** = {***A1***, ***A2***,..., ***An***}

nazywamy każdy **minimalny nadklucz** (nie zawierający w sobie żadnego innego nadklucza), tzn. zbiór atrybutów ***X*** jest kluczem, jeżeli wartość każdego atrybutu w **R** jest jednoznacznie zdeterminowana przez wartości atrybutów zbioru ***X*** i żaden podzbiór zbioru ***X*** nie ma już tej własności.

Zawsze istnieje co najmniej jeden nadklucz - całe **R**, stąd wniosek, że istnieje co najmniej jeden minimalny nadklucz, czyli klucz. Jednak kluczy może być więcej.

Zależność funkcyjna *od klucza*

Zależność funkcyjna X => Y jest zależnością od klucza jeśli zbiór atrybutów X jest nadkluczem.

Zależność funkcyjna ***X*** -> ***Y*** jest zależnością **nie od klucza** jeśli:

- **jest nietrywialna**, tzn. zbiór ***Y*** nie jest podzbiorem ***X***,
- **nie jest zależnością od klucza**.

Istnieją dwa typy zależności nie od klucza:

- **częściowa** - od części klucza,
- **przechodnia** - od czegokolwiek niebędącego kluczem, ani niewchodzącego w jego skład.

Istnienie w schemacie bazy danych zależności funkcyjnych „nie od klucza” może prowadzić (i na ogół prowadzi) do istnienia redundancji i anomalii przy operacjach na danych.

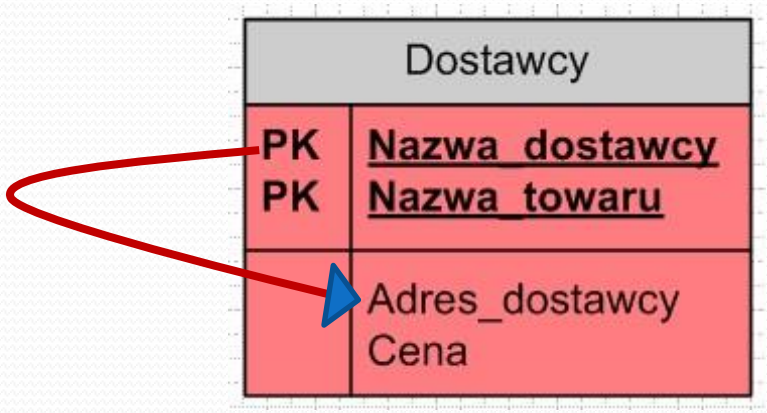
Zależność funkcyjna częściowa - przykład

Mówimy, że wartość atrybutu **Adres_dostawcy** zależy częściowo od klucza:

Nazwa_dostawcy ⟹ **Adres_dostawcy**

a samą zależność nazywamy **zależnością częściową**. Inaczej mówiąc, zależność częściowa to taka sytuacja, w której jesteśmy w stanie określić wartość atrybutu w oparciu o znajomość wartości innego atrybutu, który wchodzi w skład klucza, ale stanowi tylko jego część.

Wystąpienie zależności częściowej określamy jako **brak drugiej postaci normalnej**.

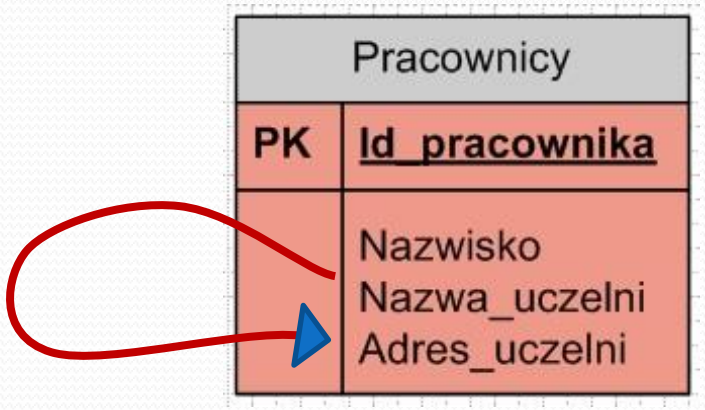


Zależność funkcyjna przechodnia - przykład

Mówimy, że wartość atrybutu **Adres_uczelni** zależy przechodnio od klucza:

Id_Pracownika ⟶ **Nazwa_uczelni** ⟶ **Adres_uczelni**

a samą zależność nazywamy **zależnością przechodnią**. Inaczej mówiąc, zależność przechodnia to sytuacja, w której jesteśmy w stanie określić wartość atrybutu na podstawie znajomości wartości innego atrybutu, nie wchodzącego w skład klucza (niekluczowego).



Zależności przechodnia i częściowa

Reasumując, istnienie zależności częściowych i przechodnich wskazuje na fakt, że schemat tabel ma niepoprawne właściwości. Jak widać, poprawne są tylko zależności funkcyjne od całego klucza.

Stwierdzenie istnienia w schemacie bazy danych zależności funkcyjnych przechodnich lub częściowych wskazuje na konieczność rozłożenia istniejących schematów na takie, które opisują pojedyncze klasy (typy) obiektów oraz wprowadzenie związków pomiędzy nimi.

Eliminowanie „złych” zależności funkcyjnych

Można podać algorytm eliminowania „złych” zależności funkcyjnych ze schematów relacji:

1. Atrybuty będące w „złej” zależności funkcyjnej przenieś do nowego schematu. Atrybuty znajdujące się po lewej stronie zależności będą kluczem nowego schematu.
2. Ze schematu wyjściowego usuń atrybuty, które znajdowały się po prawej stronie „złej” zależności funkcyjnej. Atrybuty z lewej strony tej zależności będą pełniły teraz rolę klucza obcego.

W praktyce należy przyjąć zasadę, że każdy rodzaj (klasa) obiektów jest opisywana przez osobną relację, czyli w modelu bazy danych przez osobną encję.

Postać normalna Boyce’a-Codda

Relacja o schemacie R znajduje się w **postaci normalnej Boyce'a-Codda**, jeśli nie zawiera zależności nie od klucza tj. dla każdej zależności

X => ***A*** w schemacie relacji **R**

(gdzie ***X*** podzbiór R, ***A*** atrybut w **R**)

zachodzi albo

- ***A*** należy do ***X*** (zależność trywialna),

albo

- ***X*** jest nadkluczem**.**

Jeśli schemat relacji znajduje się w postaci normalnej Boyce'a-Codda, nie można w tabeli przewidzieć jednych wartości w oparciu o inne, chociaż jak to będzie pokazane dalej, nie mamy gwarancji, że nie będzie innego rodzaju redundancji niż zależność funkcyjna.

Trzecia postać normalna

Relacja o schemacie **R** znajduje się w **trzeciej postaci normalnej**, jeśli wszystkie zależności nie od klucza są między atrybutami kluczowymi, tj. dla każdej zależności ***X*** => ***A*** w schemacie relacji **R** (gdzie ***X*** podzbiór **R**, ***A*** atrybut w **R**) zachodzi:

Albo

- ***A*** należy do ***X*** (zależność trywialna),

Albo

- ***X*** jest nadkluczem,

Albo

- ***A*** jest atrybutem kluczowym.

Atrybut kluczowy jest to atrybut wchodzący w skład jednego z kluczy relacji (tabeli).

Postać normalna IV i V

Brak "złych" zależności funkcyjnych nie gwarantuje jeszcze braku redundancji i anomalii. Definiowane są jeszcze dwa rodzaje zależności:

Wielowartościowe oznaczające brak **Czwartej Postaci Normalnej**

Złączeniowe oznaczające brak **Piątej Postaci Normalnej**

Ich istnienie implikuje redundancje i anomalie. Przykłady ilustrujące te ułomności schematów baz danych znajdują się w wykładzie nr 5 dla przedmiotu RBD.