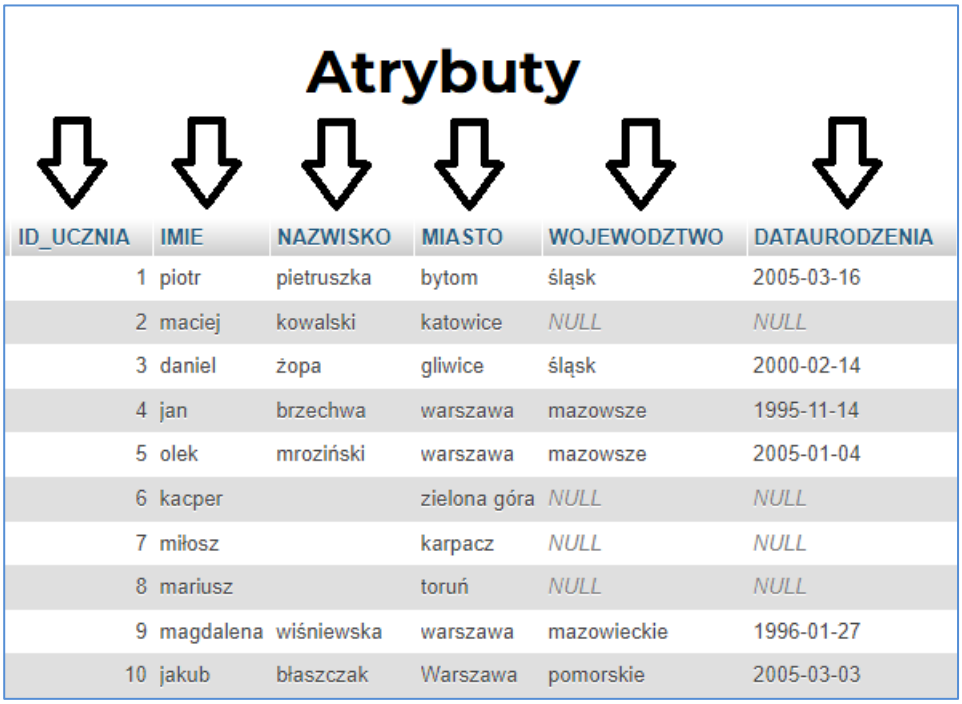
# ROZDZIAŁ 2

# RELACYJNY MODEL BAZ DANYCH

# 2 Relacyjny Model Baz Danych

## 2.1 Cechy relacyjnego modelu baz danych

W relacyjnym modelu baz danych, informacje grupowane są w dwuwymiarowych tabelach nazywanych inaczej relacjami. Każda tabela posiada swoją unikatową nazwę, przez którą jest identyfikowana. Kolumny tabeli, zwane także atrybutami reprezentują zbiór danych poszczególnego typu (np. wiek, numer telefonu, nazwisko itp.). Kolejność atrybutów w tabeli nie ma znaczenia. Kolumny mogą zawierać różne typy danych, m. in.: tekst, liczbę, datę a nawet obraz lub plik. Jednakże wszystkie wartości jednego atrybutu muszą być tego samego typu danych.




| ID_UCZNIA | IMIE | NAZWISKO | MIASTO | WOJEWODZTWO | DATAURODZENIA |
|---|---|---|---|---|---|
| 1 | piotr | pietruszka | bytom | śląsk | 2005-03-16 |
| 2 | maciej | kowalski | katowice | *NULL* | *NULL* |
| 3 | daniel | żopa | gliwice | śląsk | 2000-02-14 |
| 4 | jan | brzechwa | warszawa | mazowsze | 1995-11-14 |
| 5 | olek | mroziński | warszawa | mazowsze | 2005-01-04 |
| 6 | kacper | | zielona góra | *NULL* | *NULL* |
| 7 | miłosz | | karpacz | *NULL* | *NULL* |
| 8 | mariusz | | toruń | *NULL* | *NULL* |
| 9 | magdalena | wiśniewska | warszawa | mazowieckie | 1996-01-27 |
| 10 | jakub | błaszczak | Warszawa | pomorskie | 2005-03-03 |

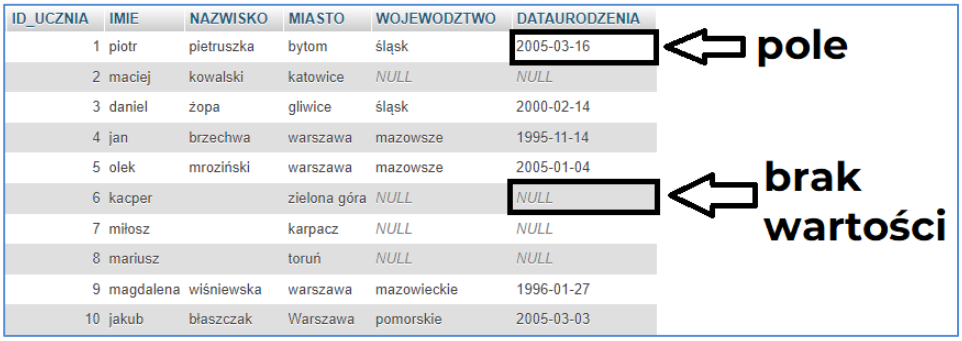
**Rysunek 2.1 Atrybuty w tabeli**

Wiersz nazywany także rekordem lub krotką, reprezentuje zbiór atrybutów danego obiektu w bazie danych (np. Piotr, Pietruszka, Bytom, Śląsk, 2005-03-16). Podobnie jak kolejność atrybutów, kolejność rekordów w tabeli nie ma znaczenia.


| ID_UCZNIA | IMIE | NAZWISKO | MIASTO | WOJEWODZTWO | DATAURODZENIA |
|---|---|---|---|---|---|
| 1 | piotr | pietruszka | bytom | śląsk | 2005-03-16 |
| 2 | maciej | kowalski | katowice | NULL | NULL |
| 3 | daniel | żopa | gliwice | śląsk | 2000-02-14 |
| 4 | jan | brzechwa | warszawa | mazowsze | 1995-11-14 |
| 5 | olek | mroziński | warszawa | mazowsze | 2005-01-04 |
| 6 | kacper | | zielona góra | NULL | NULL |
| 7 | miłosz | | karpacz | NULL | NULL |
| 8 | mariusz | | toruń | NULL | NULL |
| 9 | magdalena | wiśniewska | warszawa | mazowieckie | 1996-01-27 |
| 10 | jakub | błaszczak | Warszawa | pomorskie | 2005-03-03 |

**Rysunek 2.2 Rekordy w tabeli**

Każde pole, czyli przecięcie się rekordu z atrybutem zawiera niepodzielną wartość elementarną z dziedziny określonej przez atrybut. Brakowi wartości w danym polu odpowiada specjalna wartość NULL




| ID_UCZNIA | IMIE | NAZWISKO | MIASTO | WOJEWODZTWO | DATAURODZENIA |
|---|---|---|---|---|---|
| 1 | piotr | pietruszka | bytom | śląsk | 2005-03-16 |
| 2 | maciej | kowalski | katowice | NULL | NULL |
| 3 | daniel | żopa | gliwice | śląsk | 2000-02-14 |
| 4 | jan | brzechwa | warszawa | mazowsze | 1995-11-14 |
| 5 | olek | mroziński | warszawa | mazowsze | 2005-01-04 |
| 6 | kacper | | zielona góra | NULL | NULL |
| 7 | miłosz | | karpacz | NULL | NULL |
| 8 | mariusz | | toruń | NULL | NULL |
| 9 | magdalena | wiśniewska | warszawa | mazowieckie | 1996-01-27 |
| 10 | jakub | błaszczak | Warszawa | pomorskie | 2005-03-03 |

**Rysunek 2.3 Pola i Wartości Null w tabeli**

## 2.2 Klucze główne i klucze obce

W relacyjnym modelu baz danych występują tzw. klucze główne i klucze obce. Klucz Główny to minimalna kombinacja atrybutów identyfikujących każdy rekord w tabeli w jednoznaczny sposób. Bez klucza głównego niemożliwe byłoby zidentyfikowanie poszczególnych rekordów, dlatego niezbędnym jest, aby każda utworzona tabela go posiadała.

Może on składać się on z wielu atrybutów znajdujących się już w strukturze tabeli, jednakże najczęściej przyjmuje on postać osobnego atrybutu o nazwie ID. W takim wypadku nazywany jest on także kluczem sztucznym. Forma klucza głównego nie jest od górnie ustalona, podyktowana jest ona wyborem projektanta baz danych.

Przykładem klucza głównego w codziennym życiu może być np. numer pesel. Jest on unikatowy dla każdego obywatela, czyli pozwala on go jednoznacznie zidentyfikować, nawet w przypadku, kiedy 2 osoby miałby takie same imię i nazwisko. Do niezbędnych cech klucza głównego w bazie danych zaliczamy:

- Unikatowość - Wartość klucza głównego musi być unikatowa, co oznacza, że nie powtarza się w żadnym innym rekordzie w danej tabeli.
- Stabilność - Wartości klucza głównego nie mogą się zmieniać w czasie.
- Nie NULL - Klucz główny nie może przybierać wartości NULL, ponieważ w takim wypadku identyfikacja danego rekordu nie byłaby możliwa.

Inne ważne cechy klucza głównego, które nie są niezbędne to:

- Długość - Klucz główny powinien być jak najkrótszy i składać się z jednego atrybutu, jeśli tylko się da.
- Numeryczny Typ Danych - Wartość klucza głównego powinna przybierać postać liczby.


| ID_UCZNIA | IMIE | NAZWISKO | MIASTO | WOJEWODZTWO | DATAURODZENIA |
|---|---|---|---|---|---|
| 1 | piotr | pietruszka | bytom | śląsk | 2005-03-16 |
| 2 | maciej | kowalski | katowice | NULL | NULL |
| 3 | daniel | żopa | gliwice | śląsk | 2000-02-14 |
| 4 | jan | brzechwa | warszawa | mazowsze | 1995-11-14 |
| 5 | olek | mroziński | warszawa | mazowsze | 2005-01-04 |
| 6 | kacper | | zielona góra | NULL | NULL |
| 7 | miłosz | | karpacz | NULL | NULL |
| 8 | mariusz | | toruń | NULL | NULL |
| 9 | magdalena | wiśniewska | warszawa | mazowieckie | 1996-01-27 |
| 10 | jakub | błaszczak | Warszawa | pomorskie | 2005-03-03 |

**Rysunek 2.4 Klucz główny**

Klucz obcy jest niezbędnym elementem poprawnie zaprojektowanej bazy danych. Jest on atrybutem lub zbiorem atrybutów, który wskazuje na klucz główny w innej tabeli. Praktycznie mówiąc, dzięki kluczowi obcemu jesteśmy w stanie stworzyć relację pomiędzy dwoma tabelami. Możemy sobie wyobrazić tak utworzoną relację, jako połączenie nadrzędnej i podrzędnej tabeli. Tabela nadrzędna posiada wszystkie informacje o danym obiekcie, a tabela podrzędna posiadająca klucz obcy wskazuje tylko na odpowiedni rekord w tabeli nadrzędnej.


| ID_OCENA | STOPIEN | DATAWYSTAWIENIA | ID_UCZNIA | ID_NAUCZYCIELA |
|---|---|---|---|---|
| 1 | 1 | 2023-12-23 | 1 | 1 |
| 2 | 5 | 2023-05-11 | 1 | 1 |
| 3 | 4 | 2023-01-06 | 2 | 1 |
| 4 | 5 | 2023-04-12 | 3 | 1 |
| 5 | 6 | 2023-03-22 | 4 | 2 |
| 6 | 2 | 2023-11-15 | 5 | 2 |
| 7 | 2 | 2023-10-18 | 5 | 2 |
| 8 | 3 | 2023-09-23 | 6 | 3 |
| 9 | 4 | 2023-12-06 | 7 | 3 |
| 10 | 1 | 2023-11-15 | 7 | 4 |
| 11 | 2 | 2023-10-01 | 1 | 4 |

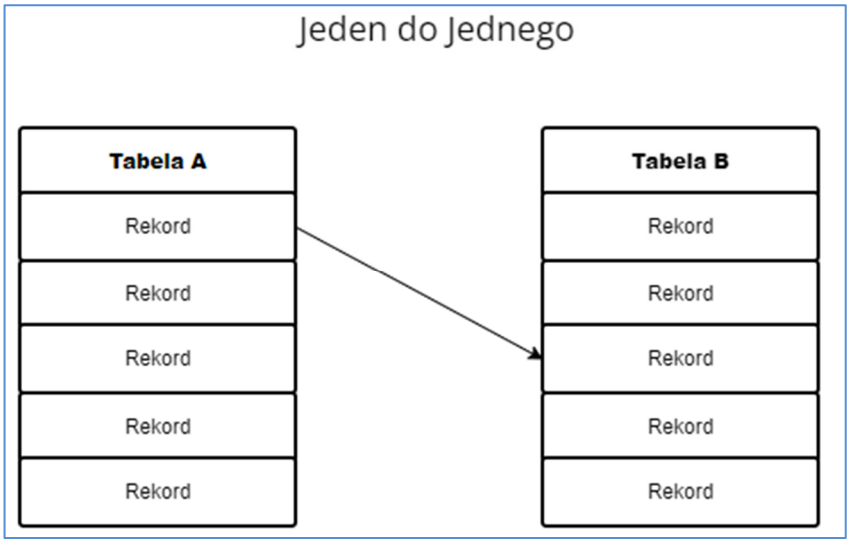
**Rysunek 2.5 Rodzaje kluczy w tabeli**

## 2.3 Typy Relacji

W relacyjnym modelu baz danych grupujemy informacje w odrębnych tabelach tematycznych. Jeśli chcemy zestawić ze sobą dane, musimy odpowiednio połączyć te tabele. Tabele możemy łączyć na 3 różne sposoby

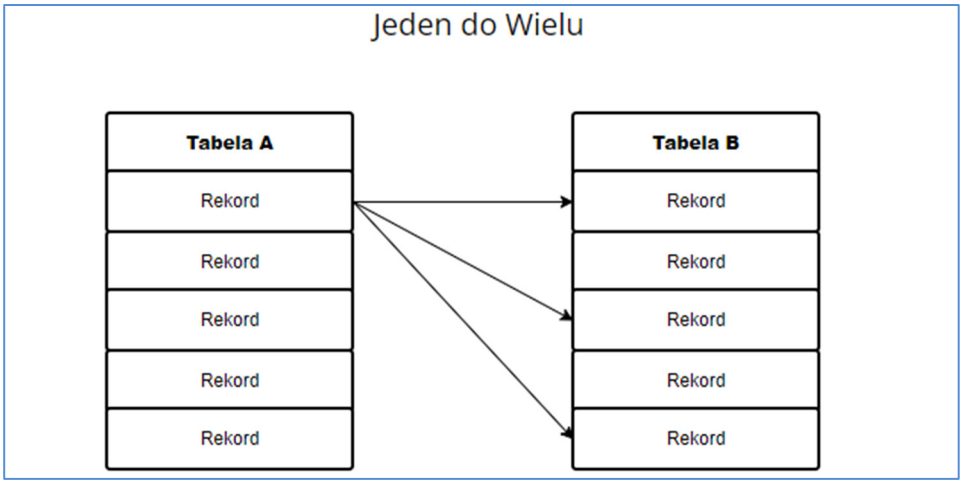
- Jeden do jednego (1:1) Przy takim połączeniu każdemu rekordowi z pierwszej tabeli może odpowiadać tylko jeden rekord z drugiej tabeli, a każdemu rekordowi z drugiej tabeli odpowiada jeden rekord

z pierwszej. Np. Jednemu pracownikowi przypada tylko jeden samochód służbowy. Relacje jeden do jednego są najrzadziej stosowanymi formami połączenia tabel, ponieważ w znacznej większości przypadków dane takie znajdują się w jednej tabeli.



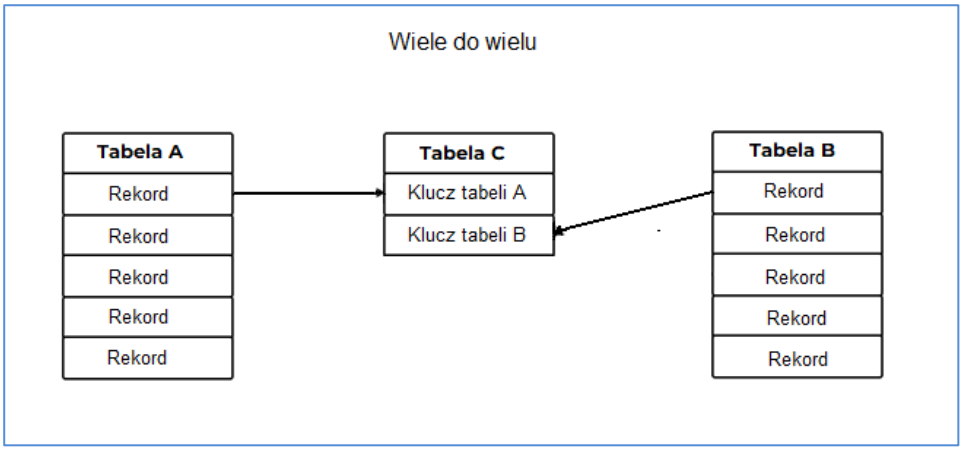

**Rysunek 2.6 Schemat relacji jeden do jednego**

- Jeden do wielu (1:n) - W takiej relacji każdemu rekordowi z pierwszej tabeli może odpowiadać wiele rekordów z drugiej tabeli, ale do każdego rekordu z drugiej tabeli, przypisany jest tylko jeden z pierwszej tabeli, np. jeden artysta może namalować wiele obrazów, ale każdy obraz ma jednego artystę. Taki sposób łączenia tabel jest bardzo użyteczny, dlatego relacje jeden do wielu wykorzystuje się najczęściej spośród wszystkich sposobów łączenia tabel.




**Rysunek 2.7 Schemat relacji jeden do wielu**

- Wiele do wielu (n:n) - Ta relacja charakteryzuje się tym, że każdy rekord z pierwszej tabeli może być połączonym z rekordem z drugiej tabeli, ale także każdy rekord z drugiej tabeli może być połączony z dowolnym rekordem z pierwszej, np. klient może złożyć zamówienie z wieloma produktami, a dany produkt może znajdować się w wielu zamówieniach. W praktyce, aby stworzyć taką relację należy utworzyć trzecią pośredniczącą tabelę, w której przechowywać będziemy informacje o kluczach głównych z tabel które chcemy połączyć.




**Rysunek 2.8 Schemat relacji wiele do wielu**

## 2.4 Normalizacja

W codziennym życiu informacje zwykle podawane są jako jedna całość. Wprowadzanie takich danych do bazy, skutkowałoby trudnością wyszukiwania poszczególnych rekordów lub konieczności wpisywania jednej informacji kilkukrotnie w wielu miejscach, co zajmuje więcej przestrzeni dyskowej i utrudnia edycję danych. Aby uniknąć takich problemów stosuje się metodę, zwaną normalizacją.

Normalizacja ma na celu zmniejszenie nadmiarowości danych oraz zapewnienie ich integralności. Polega ona na dzieleniu dużych tabel zawierających ogół informacji, na mniejsze tematyczne tabele i łączenie ich za pomocą relacji. Istnieje kilka zasad normalizacji baz danych, nazywane są one postaciami normalnymi. Każda postać normalna, dzieli tabelę na bardziej elementarne części. Tabelę uznajemy za znormalizowaną, jeśli spełnia ona zasady trzeciej postaci normalnej.

### 2.4.1 Tabela Nieznormalizowana.

Taka tabela może przyjmować dowolną strukturę. Wydobywanie z niej danych jest bardzo utrudnione. Silnik bazy danych nie pozwoli nam stworzyć takiej tabeli

| Model | Cena (zł) | Lokalizacja | Rodzaj |
|---|---|---|---|
| Hyundai i20 | 30 000 zł | Polska | Auto Miejskie |
| Hyundai i20 | 32 000 zł | Warszawa | Auto Miejskie |
| Opel Corsa | 25000 | Gliwice Polska | 3 |
| Volkswagen Passat | 60 tyś | 1 | Auto duże |
| Toyota Rav4 | 20 500 zł | Włochy | Suv |
| Kia Rio | 10000$ | Paryż | 2 |
| Kia Rio | 30000 | Łódź | Auto Miejskie |

**Rysunek 2.9 Tabela Nieznormalizowana**

**Pierwsza Postać Normalna (1NF)**

Tabela przybierająca pierwszą postać normalną, wygląda już jak tabela, z której możemy w prosty sposób wydobyć dane. Jednak musi ona spełniać określone wymagania. Między innymi są nimi:

- Musi posiadać klucz główny
- Dane zawarte w każdej kolumnie muszą być jednakowego typu

- Kolejność rekordów czy atrybutów nie może wpływać na znaczenie danych

| Nr ogłoszenia | Model | Cena (zł) | Lokalizacja | Rodzaj |
|---|---|---|---|---|
| 1 | Hyundai i20 | 30000 | Bytom | Auto Miejskie |
| 2 | Hyundai i20 | 32000 | Warszawa | Auto Miejskie |
| 3 | Opel Corsa | 25000 | Gliwice | Auto Miejskie |
| 4 | Volkswagen Passat | 60000 | Gdańsk | Auto duże |
| 5 | Toyota Rav4 | 20500 | Rzym | Suv |
| 6 | Kia Rio | 35000 | Paryż | Auto Miejskie |
| 7 | Kia Rio | 30000 | Łódź | Auto Miejskie |

**Rysunek 2.10 Tabela w 1NF**

W tym przypadku kluczem głównym jest kolumna Nr ogłoszenia

**Druga Postać Normalna (2NF)**

Tabela jest w drugiej postaci normalnej, jeżeli zachowuje zasady pierwszej postaci normalnej i żaden atrybut tej tabeli nie jest zależny od niczego, poza kluczem głównym danej tabeli. Zauważ, że gdybyśmy chcieli usunąć ogłoszenie o Toyocie z powyżej tabeli, stracilibyśmy również informacje na temat rodzaju tego samochodu. Taki sposób zapisywania rodzaju jest również nieoptymalny, ponieważ przy zmianie rodzaju danego samochodu, musielibyśmy edytować każdy rekord tego modelu.

Takich sytuacji możemy uniknąć sprowadzając tabelę do drugiej postaci normalnej. A robimy to następująco:

- Tworzymy osobnych tabele, dla wartości które wykorzystywane są w wielu rekordach
- Łączymy nowo utworzonych tabel za pomocą relacji

| Nr ogłoszenia | Model | Cena (zł) | Lokalizacja |
|---|---|---|---|
| 1 | Hyundai i20 | 30000 | Bytom |
| 2 | Hyundai i20 | 32000 | Warszawa |
| 3 | Opel Corsa | 25000 | Gliwice |
| 4 | Volkswagen Passat | 60000 | Gdańsk |
| 5 | Toyota Rav4 | 20500 | Rzym |
| 6 | Kia Rio | 35000 | Paryż |
| 7 | Kia Rio | 30000 | Łódź |

| Model | Rodzaj |
|---|---|
| Hyundai i20 | Auto Miejskie |
| Opel Corsa | Auto Miejskie |
| Volkswagen Passat | Auto duże |
| Toyota Rav4 | Suv |
| Kia Rio | Auto Miejskie |

**Rysunek 2.11 Tabela w 2NF**

### 2.4.2 Trzecia Postać Normalna (3NF)

Tabela jest w drugiej postaci normalnej, jeżeli zachowuje zasady pierwszej i drugiej postaci normalnej oraz wszystkie atrybuty w danej tabeli zależne są tylko

i wyłącznie od klucza głównego. Poszerzmy naszą tabelę z ogłoszeniami o kolumnę kraj.

| Nr ogłoszenia | Model | Cena (zł) | Lokalizacja | Kraj | | Model | Rodzaj |
|---|---|---|---|---|---|---|---|
| 1 | Hyundai i20 | 30000 | Bytom | Polska | | Hyundai i20 | Auto Miejskie |
| 2 | Hyundai i20 | 32000 | Warszawa | Polska | | Opel Corsa | Auto Miejskie |
| 3 | Opel Corsa | 25000 | Gliwice | Polska | | Volkswagen Passat | Auto duże |
| 4 | Volkswagen Passat | 60000 | Gdańsk | Polska | | Toyota Rav4 | Suv |
| 5 | Toyota Rav4 | 20500 | Rzym | Włochy | | Kia Rio | Auto Miejskie |
| 6 | Kia Rio | 35000 | Paryż | Francja | | | |
| 7 | Kia Rio | 30000 | Łódź | Polska | | | |

**Rysunek 2.12 Tabela po dodaniu kolumny Kraj**

Tabela ta znajduje się w drugiej postaci normalnej. Ale nie spełnia już wymogów 3NF. To dlatego ponieważ atrybut miasto zależny jest od atrybutu kraj. Wyobraźmy sobie, że chcemy zmienić lokalizację wystawienia samochodu.

Aby to zrobić musielibyśmy zaktualizować pola miasto oraz kraj w naszej tabeli. Jednak, gdyby pomiędzy aktualizowaniem pola miasto a kraj operacja uległaby przerwaniu zostalibyśmy z nieprawdziwymi danymi. Np. Bytom jest we Francji. Aby nie dopuścić do takiego przypadku musimy sprowadzić tabelę do trzeciej postaci normalnej.

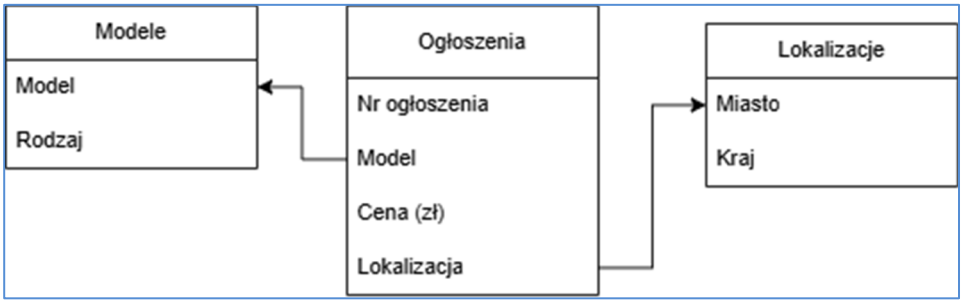
Robimy to poprzez:

- Utworzenie osobnych tabel dla każdej kolumny zależnej od kolumny, która nie jest kluczem głównym

- Połączenie nowo powstałych tabel relacjami

| Nr ogłoszenia | Model | Cena (zł) | Lokalizacja | | Model | Rodzaj | | Kraj | Miasto |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Hyundai i20 | 30000 | Bytom | | Hyundai i20 | Auto Miejskie | | Polska | Bytom |
| 2 | Hyundai i20 | 32000 | Warszawa | | Opel Corsa | Auto Miejskie | | Polska | Warszawa |
| 3 | Opel Corsa | 25000 | Gliwice | | Volkswagen Passat | Auto duże | | Polska | Gliwice |
| 4 | Volkswagen Passat | 60000 | Gdańsk | | Toyota Rav4 | Suv | | Polska | Gdańsk |
| 5 | Toyota Rav4 | 20500 | Rzym | | Kia Rio | Auto Miejskie | | Włochy | Rzym |
| 6 | Kia Rio | 35000 | Paryz | | | | | Francja | Paryż |
| 7 | Kia Rio | 30000 | Łódź | | | | | Polska | Łódź |

**Rysunek 2.13 Tabela w 3NF**




**Rysunek 2.14 Tabele połączone relacjami**

## 2.5 Diagramy związków encji

### 2.5.1 Diagramy ERD

Mając tylko informację o nazwach tabel i położeniu kluczy głównych i obcych, możemy mieć problem z wyobrażeniem sobie struktury bazy danych, szczególnie jeśli jest ona złożona. Aby w prosty i intuicyjny sposób zwizualizować taką bazę danych możemy posłużyć się tzw. diagramem ERD (Entity relationship diagram).

Jak sama nazwa wskazuje, diagram ten przedstawia sposób w jaki encje są ze sobą powiązane. Aby to zrozumieć, musimy dowiedzieć się najpierw czym jest encja.

Encja to każda rzecz lub pojęcie, o których informacje chcemy przechować. W odniesieniu do baz danych encja przyjmuje postać tabeli.

Encje mogą posiadać również, przypisane im atrybuty, czyli cechy, które je określają. Atrybuty przyjmują postać kolumn w tabeli.

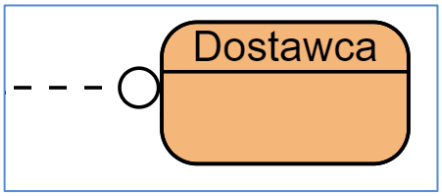
Aby pokazać istotne połączenie pomiędzy dwiema encjami, używamy tzw. związków. Związki opisują zależności zachodzące pomiędzy encjami. Związki w bazach danych reprezentowane są przez relacje.

### 2.5.2 Sposób zapisywania diagramów ERD

Istnieje wiele sposobów zapisu diagramu ERD, ale każdy z nich przedstawia zasadniczo to samo. Najszerzej używaną notacją jest tzw. Notacją kurzej stopki (Crow's foot notation). W tej notacji encje zapisuje się w formie prostokątów z nazwa i atrybutami, a relacje przedstawiane są za pomocą linii zakończonych określonymi symbolami.

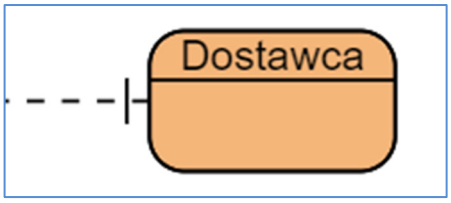
Gdzie:

Kółko symbolizuje zero.



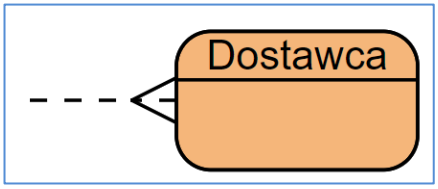

**Rysunek 2.15 Zapis 'Zero' w notacji kurzej stopki**

Linia przecinająca relacje pod katem prostym symbolizuje jeden




**Rysunek 2.16 Zapis 'Jeden' w notacji kurzej stopki**

Kurza stopka symbolizuje wiele




**Rysunek 2.17 Zapis 'Wiele' w notacji kurzej stopki**

Symbole te mogą być łączone w pary, gdy tak się dzieje element zewnętrzny określa maksymalną liczbę, a wewnętrzny minimalną:

Kółko i kreska oznaczają minimum zero i co najwyżej jedno wystąpienie. Taka relacja nie jest opcjonalna i nie musi zawsze zachodzić. Dwie kreski oznaczają minimum jeden i co najwyżej jeden, czyli w prostych słowach dokładnie jedno wystąpienie. Jest to relacja 1:1

Kółko i kurza stopka reprezentują minimum zero i brak wartości maksymalnej. Jest to również relacja opcjonalna. Kreska i kurza stopka oznaczają minimum jeden i maksimum wiele, czyli relacja 1 do wielu.

### 2.5.3 Rodzaje diagramów ERD

Diagramy ERD możemy podzielić na 3 główne kategorie:

- Diagramy Konceptowe

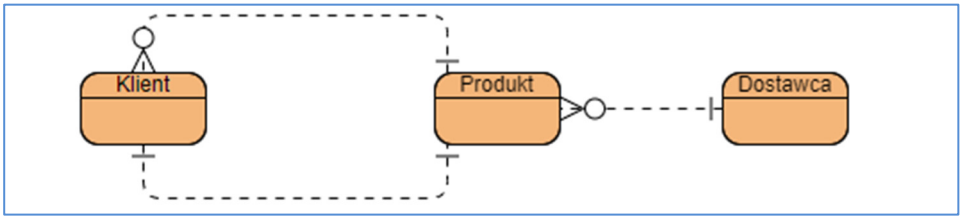
- Diagramy Logiczne

- Diagramy Fizyczne

Wszystkie trzy rodzaje diagramów składają się z encji, atrybutów i związków. Jednak pełnią one różne role i przedstawiają strukturę bazy danych, na różnych poziomach złożoności.

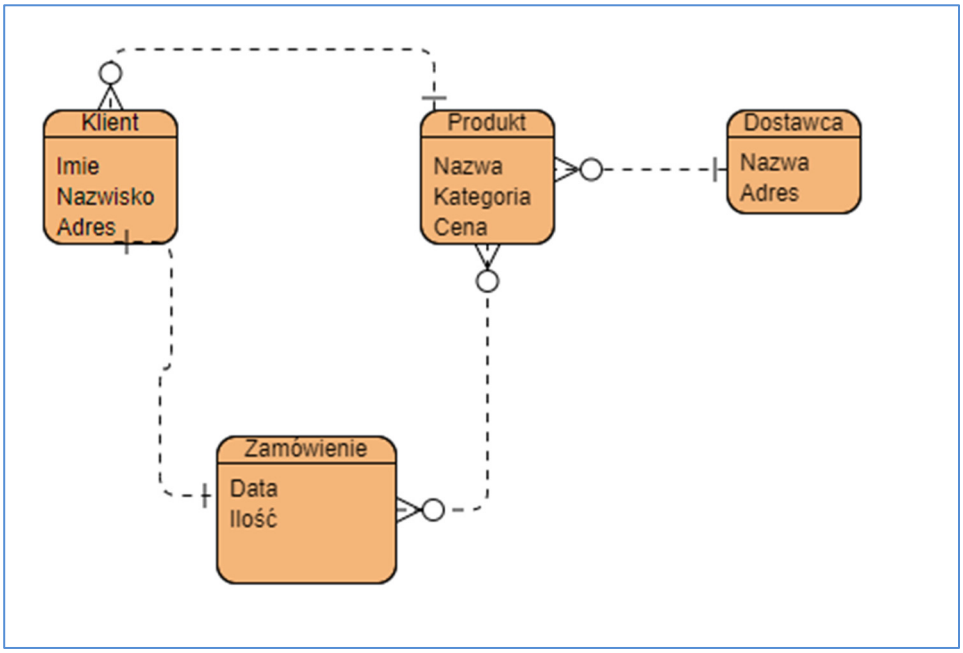
Najprostszym z nich jest diagram konceptowy. Ma on na celu zarys całokształtu bazy danych i określenia jakie obiekty będą się w niej znajdować. Nie ma w nim

ścisłego podziału na tabele z atrybutami. Jest najbardziej przejrzystym modelem diagramu.



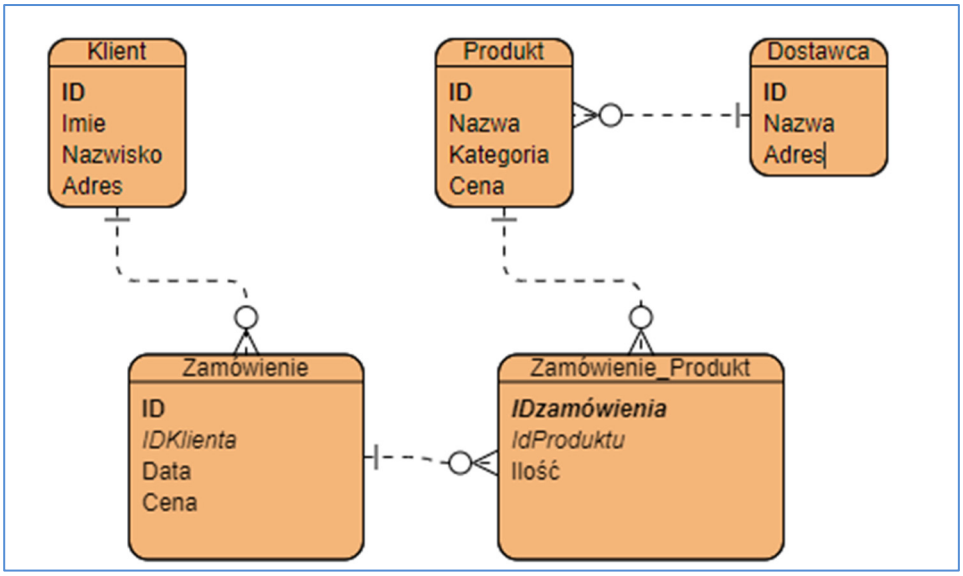

**Rysunek 2.18 Schemat diagramu konceptowego**

Drugim pod względem złożoności jest diagram logiczny. Jest on uszczegółowioną wersją diagramu konceptowego. Obiekty posiadają w nim już swoje atrybuty, jednak brakuje im wszystkich danych potrzebnych do stworzenia w pełni sprawnej bazy danych.




**Rysunek 2.19 Schemat diagramu logicznego**

Ostatnim modelem diagramu, jest diagram fizyczny. Można traktować go jako kompletny schemat bazy danych, według którego może być utworzona. Posiada on informacje o podziale na tabele z atrybutami, a także wskazane są klucze główne i obce.




**Rysunek 2.20 Schemat diagramu fizycznego**

## 2.6 Zasady projektowania bazy danych

Każda baza danych jest unikatowa i znajduje zastosowanie w innej dziedzinie życia. Nie istnieje przepis na zaprojektowanie idealnej bazy danych do każdego zadania. Jednak jako projektant baz danych, możesz podążać za dobrymi praktykami projektowania, które zagwarantują, że twoja baza danych będzie działała w wydajny i optymalny sposób.

Istnieje kilka zasad, które musimy brać pod uwagę projektując bazę danych. Pierwszą i za razem najważniejszą z nich jest to, żeby unikać zduplikowanych danych, gdzie tylko to możliwe. Posiadanie zduplikowanych danych zużywa więcej przestrzeni dyskowej i czyni naszą bazę o wiele bardziej podatną na wystąpienia błędów i niespójności. Druga zasada polega na tym, że dane zawarte w naszej bazie muszą być kompletne, a dostęp do nich ma być prosty. Tworzenie niepotrzebnie skomplikowanej bazy, do prostych zastosowań, doda pracy na każdym kolejnym kroku jej użytkowania.

Aby w poprawny sposób zaprojektować bazę danych, można podążać tzw. procesem projektowania. Składa się on z następujących kroków:

- Określenie przeznaczenia bazy danych

- Wyszukiwanie i organizowanie informacji

- Dzielenie informacji na tabele
- Przekształcanie informacji w tabele i kolumny
- Określanie kluczy głównych
- Łączenie tabeli w relacje
- Weryfikacja projektu

### 2.6.1 Określenie przeznaczenia bazy danych

Określenie do jakich celów, posłużyć ma nasza baza danych jest pierwszym i najważniejszym krokiem w jej projektowaniu. Dwie bazy danych stosowane w tym samym miejscu, jednak do różnych celów mogą wyglądać zupełnie odmiennie (np. baza danych zamówień sklepu, baza danych pracowników sklepu). Źle zdefiniowany cel, może niekorzystnie wpłynąć na każdy kolejny krok projektu.

### 2.6.2 Wyszukiwanie i organizowanie informacji

Następnym krokiem jest zebranie wszystkich informacji jakie chcemy zawrzeć w naszej bazie. Wymagane dane zależne są od przeznaczenia naszej bazy. Poza podstawowymi informacjami, które od razu przychodzą na myśl, należy przewidzieć również jakie dane mogą przydać się w przyszłości.

Warto zadać sobie pytanie, w jaki sposób informacje przedstawiane będą użytkownikowi. Jeżeli chcielibyśmy, aby były sortowane po regionie lub wieku musimy również uwzględnić te dane. Ważne jest, żeby wszystkie informacje rozdzielić na jak najmniejsze kawałki. Zamiast ceny końcowej, warto rozbić ją na cenę bazową i rabat. Podczas użytkowania bazy danych łatwo będzie obliczyć końcową cenę, ale odwrócenie procesu może być bardzo ciężkie lub niemożliwe.

### 2.6.3 Przekształcanie informacji w tabele

To jak będą wyglądać tabele w naszej bazie danych, ściśle zależy od tego, jakie informacje chcemy w niej zawrzeć. Tabela powinna reprezentować główne podmioty znajdujące się w naszej bazie.

Jeżeli poprawnie podzieliliśmy informacje, na najmniejsze możliwe części możemy posłużyć się zasadą, że gdy ponad 2 informacje opisują dany obiekt powinna zostać utworzona dla niego tabela. Kolumny w tabeli powinny

reprezentować każdą niepodzielną informację, która opisuje dany obiekt. Drugą zasadą, którą powinniśmy się kierować podczas tworzenia kolumn, jest unikanie obliczonych danych. Posługując się bazą danych obliczenie sumy lub średniej staje się banalnie łatwe Jednak, gdy do bazy wprowadzimy już wcześniej obliczone dane odwrócenie działań staje się niemożliwe.

### 2.6.4 Określanie kluczy głównych

Każda tabela powinna posiadać kolumnę, która pozwala na identyfikacje danego rekordu. Jeżeli posiadamy już taką kolumnę, np. numer seryjny produktu, to ona może posłużyć nam jako klucz główny. Musimy pamiętać, że kolumna kandydująca na bycie kluczem głównym, musi mieć niezmienne i unikatowe informacje. Jeżeli jednak nie posiadamy takiej kolumny, możemy ją zwyczajnie utworzyć. Taka kolumna, będzie nazywana wtedy kluczem sztucznym i zwykle zawierać będzie kolejne liczby całkowite.

### 2.6.5 Łączenie tabeli w relacje

Gdy mamy już określone tabele, powinniśmy połączyć je we wspólną całość, tak aby informacje w nich się uzupełniały. Tabele łączymy ze sobą za pomocą relacji, jednak musimy określić jakiego typu relacji użyć. Gdy łączymy ze sobą tabele zawierające informacje o jednym obiekcie, robimy to za pomocą relacji jeden do jednego. W takim przypadku połączeniu ulegają klucze główne obu tabel. Gdy informacje z jednej tabeli, pasują do wielu rekordów z drugiej, powinniśmy zastosować relację jeden do wielu. Tworząc taką relacje musimy w tabeli po stronie wiele dodać kolumnę, która posłuży jako klucz obcy. Gdy wiele rekordów z pierwszej tabeli może odpowiadać wielu rekordom z drugiej, używamy relacji wiele do wielu. Aby takowa utworzyć, musimy zdefiniować trzecią tabelę, która przechowywać będzie klucze obce obu pierwotnych tabel.

### 2.6.6 Weryfikacja projektu

Ostatnim krokiem, jaki musimy wykonać przed skończeniem projektu jest jego weryfikacja. Na tym etapie powinniśmy zweryfikować, czy wszystkie nasze tabele stosują się do zasad normalizacji. Znormalizowane tabele gwarantują brak powtarzalności się danych. Na tym etapie możemy również wypełnić bazę danych testowymi danymi. Po wypełnieniu łatwiej możemy zauważyć, czy nie zapomnieliśmy dodać jakieś kolumny lub czy któreś kolumny okazują się zbędne. Przy wstawianiu danych wyjdą na jaw, także błędy jakie mogliśmy popełnić przy

tworzeniu relacji. Powinniśmy zweryfikować także, że nasze tabele nie zawierają złożonych danych, których nie da się bardziej podzielić. Gdy jesteśmy pewni, że nasza baza danych nie zawiera błędów, oraz jest w pełni funkcjonalna nasz projekt możemy uznać za skończony.

## 2.7 Kontrola spójności danych

W świecie baz danych spójność danych to nie tylko pojęcie techniczne, ale podstawa zaufania do systemu. Oznacza ona, że dane przechowywane w bazie są poprawne, logicznie spójne, wolne od sprzeczności i zgodne z ustalonymi regułami biznesowymi. Bez kontroli spójności nawet najbardziej zaawansowana baza danych staje się niepewnym źródłem informacji, a decyzje podejmowane na jej podstawie mogą być błędne, czasem kosztowne.

Kontrolę spójności bazy danych możemy podzielić na kilka kategorii

### 2.7.1 Reguły spójności danych

Reguły spójności to logiczne ograniczenia, które definiują, co jest dozwolone, a co niedozwolone w danej bazie. Są one wyrażeniem rzeczywistych wymagań biznesowych, które muszą być odzwierciedlone w strukturze bazy.

Przykłady reguł:

- Uczeń nie może mieć oceny bez przypisania do przedmiotu.
- Data zakończenia zatrudnienia nie może być wcześniejsza niż data rozpoczęcia.
- Klient nie może mieć dwóch kont z tym samym numerem PESEL.

Te reguły implementuje się za pomocą ograniczeń, kwerend walidacyjnych, trigerów lub aplikacji po stronie użytkownika. Kluczowe jest, by były one zdefiniowane na poziomie bazy danych, a nie tylko w aplikacji w przeciwnym razie łatwo je ominąć.

### 2.7.2 Integralność referencyjna

Integralność referencyjna to mechanizm zapewniający, że relacje między tabelami są zawsze poprawne. Jest to jedna z najważniejszych form kontroli spójności.

Przykład: mamy tabelę Klienci (z kluczem głównym ID_Klienta) i tabelę Zamówienia (z kluczem obcym ID_Klienta). Integralność referencyjna gwarantuje, że:

- nie można dodać zamówienia dla klienta, który nie istnieje,
- nie można usunąć klienta, jeśli ma przypisane zamówienia (chyba że zdefiniujemy odpowiednią akcję),
- nie można zmienić ID_Klienta na wartość, która nie istnieje w tabeli Klienci.

### 2.7.3 Unikatowość danych

Powtarzające się dane to nie tylko marnowanie miejsca, ale przede wszystkim źródło błędów. Dwa rekordy o tym samym numerze PESEL lub adresie e-mail prowadzą do dezinformacji i trudności w identyfikacji użytkownika.

Unikatowość zapewnia się na dwa sposoby:

- Klucz główny automatycznie wymusza unikatowość i niepustość.
- Ograniczenie UNIQUE - pozwala na unikatowość w kolumnach niebędących kluczem głównym (np. Email, NumerLegitymacji).

### 2.7.4 Kontrola wartości

Domena kolumny to zbiór dozwolonych wartości. Kontrola domeny zapobiega wprowadzaniu danych „poza zakresem”.

Przykłady:

- Płeć może być tylko M lub K.
- Ocena w szkole od 1 do 6.
- Status zamówienia: Nowe, W realizacji, Wysłane, Anulowane.

### 2.7.5 Transakcje

Wyobraź sobie, że pracujesz w banku i musisz przelać 1000 zł z jednego konta na drugie. Aby to zrobić, trzeba wykonać dwie operacje:

Odjąć 1000 zł od salda konta nadawcy. Dodać 1000 zł do salda konta odbiorcy.

Wygląda to prosto, ale co się stanie, jeśli po pierwszej operacji (odjęciu pieniędzy) nastąpi awaria systemu? Wtedy pieniądze zniknęłyby z konta nadawcy, ale nigdy nie dotarłyby do odbiorcy. To katastrofa! W świecie baz danych taka sytuacja nazywana jest niespójnym stanem danych. Dane są częściowo zmienione, co prowadzi do błędów i utraty zaufania.

Transakcja to mechanizm, który chroni nas przed takimi wypadkami. To jakby "pudełko", do którego wkładamy wszystkie powiązane operacje. Baza danych obiecuje, że wszystkie operacje wewnątrz tego pudełka zostaną wykonane całkowicie albo wcale.

### 2.7.6 Czym jest transakcja?

Transakcja to logiczna jednostka pracy, która składa się z jednej lub więcej operacji na bazie danych. Kluczową zasadą jest, że transakcja musi być atomowa. To znaczy, że nie można jej podzielić. To oznacza, że:

- Jeśli wszystko przebiegnie pomyślnie, wszystkie zmiany są trwale zapisywane w bazie danych. Mówi się wtedy, że transakcja jest zatwierdzona (committed).
- Jeśli podczas wykonywania transakcji wystąpi błąd (np. brak środków, awaria sieci, naruszenie reguły), wszystkie zmiany są cofane. jakby nigdy się nie wydarzyły. Wtedy mówimy, że transakcja została cofnięta (rolled back).

W naszym przykładzie z przelewem:

- Jeśli dodanie pieniędzy do konta odbiorcy się powiedzie, to odjęcie z konta nadawcy również zostanie zatwierdzone.
- Jeśli dodanie pieniędzy się nie powiedzie, to odjęcie pieniędzy również zostanie cofnięte. Konto nadawcy pozostaje bez zmian.

### 2.7.7 Dlaczego transakcje są tak ważne?

- Zapobiegają utracie danych: Gwarantują, że nie dojdzie do sytuacji, w której pieniądze "znikną w próżni".
- Zachowują spójność bazy danych: Baza danych zawsze pozostaje w logicznie poprawnym stanie. Nie ma "półprac".
- Zwiększają niezawodność aplikacji: Programy mogą być pewne, że operacje albo się powiodły, albo nie, bez pozostawiania po sobie bałaganu.