

### **Bazy Danych 2 - Lab 3**

Wykonane ćwiczenia należy odpowiednio udokumentować, sporządzając krótkie sprawozdanie. Powinno ono zawierać:

- Zwięzłe omówienie przeprowadzonych zadań.
- Odpowiedzi na wszystkie postawione pytania.
- Zrzuty ekranu potwierdzające poprawną realizację kolejnych etapów ćwiczenia.

Diagramy ER (Entity-Relationship) to graficzna metoda modelowania danych używana do przedstawienia struktury systemu informacyjnego na poziomie konceptualnym. Służą one do wizualizacji encji, atrybutów tych encji oraz relacji między nimi.

[http://smurf.mimuw.edu.pl/external\\_slides/W3\\_Modelowanie\\_danych\\_Model\\_zwiazkow-encji/W3\\_Modelowanie\\_danych\\_Model\\_zwiazkow\\_encji.html](http://smurf.mimuw.edu.pl/external_slides/W3_Modelowanie_danych_Model_zwiazkow-encji/W3_Modelowanie_danych_Model_zwiazkow_encji.html)

[https://www.impan.pl/~kz/DB/KZ\\_BD\\_w04.pdf](https://www.impan.pl/~kz/DB/KZ_BD_w04.pdf)

<https://technikinformatyk.pl/programowanie/bazy-danych-diagram-erd>

[https://pl.wikipedia.org/wiki/Diagram\\_zwi%C4%85zk%C3%B3w\\_encji](https://pl.wikipedia.org/wiki/Diagram_zwi%C4%85zk%C3%B3w_encji)

## 1. Podstawowe Elementy Diagramu ER

### Encje

- **Definicja:** Encja reprezentuje konkretny obiekt lub zbiór obiektów o podobnych właściwościach. Przykłady: *Klient, Pracownik, Produkt, Zamówienie*.
- **Notacja:** W diagramach ER encje są zazwyczaj przedstawiane jako prostokąty. Nazwa encji umieszczona jest wewnątrz prostokąta.

➤ **Encja może reprezentować:**

- obiekt materialny, np.: pracownik, książka, samochód, towar
- obiekt niematerialny, np.: faktura, projekt, konto, zamówienie
- zdarzenie, np.: choroba pracownika, przyznanie nagrody, kontrola
- fakt, np.: znajomość języka obcego, stan magazynowy produktu

```
graph TD; Towar[Towar] --- nazwa((nazwa)); Towar --- cena((cena)); Towar --- kategoria((kategoria))
```

Rys. 10 Encja **Towar**

```
graph LR; subgraph Towar [Towar]; T1[Towar] --- A1[Nazwa = DELL Inspiron 1564 i5  
Kategoria = notebooki  
Cena = 2699.00]; T1 --- A2[Nazwa = Logitech OEM PC860  
Kategoria = periferia  
Cena = 42.90]; T1 --- A3[Nazwa = Canon EOS 7D  
Kategoria = lustrzanki  
Cena = 8299.00]; end
```

Rys. 11 Wystąpienia encji (instancje encji) **Towar**

## Atrybuty

- **Definicja:** Atrybuty opisują właściwości encji. Mogą to być dane, takie jak nazwa, adres, numer identyfikacyjny itp.
- **Notacja:** Atrybuty są przedstawiane jako elipsy połączone liniami z odpowiadającą im encją.

### Model E-R - Atrybut

Encja posiada **atrybuty** czyli opisujące jej szczegółowe własności  
Wartości atrybutów stanowią główną część danych składowanych w bazie danych

Definicja atrybutu encji:

- nazwa
- dziedzina:
  - typ danych i maksymalny rozmiar
  - zbiór dozwolonych wartości
  - zakres dozwolonych wartości
- dozwolone / niedozwolone wartości puste
- opcjonalnie - unikalność wartości

Każda encja posiada **identyfikator** - atrybut lub zbiór atrybutów jednoznacznie identyfikujący wystąpienie encji; pozostałe atrybuty nazywa się *deskryptorami*

## Relacje (Związki)

- **Definicja:** Relacja określa związek pomiędzy dwiema lub więcej encjami. Przykłady:  
*Klient składa Zamówienie, Pracownik pracuje w Dziale.*
- **Notacja:** Relacje przedstawiane są jako romby połączone liniami z encjami.

### Model E-R - Związek

Związek reprezentuje powiązania między obiektami świata rzeczywistego (np. *klienci kupują towary, pokój jest przydzielony pracownikowi*)

#### Krótkość związku (liczebność)

- jeden-do-jeden (1:1)
- jeden-do-wiele (1:N)
- wiele-do-wiele (M:N)

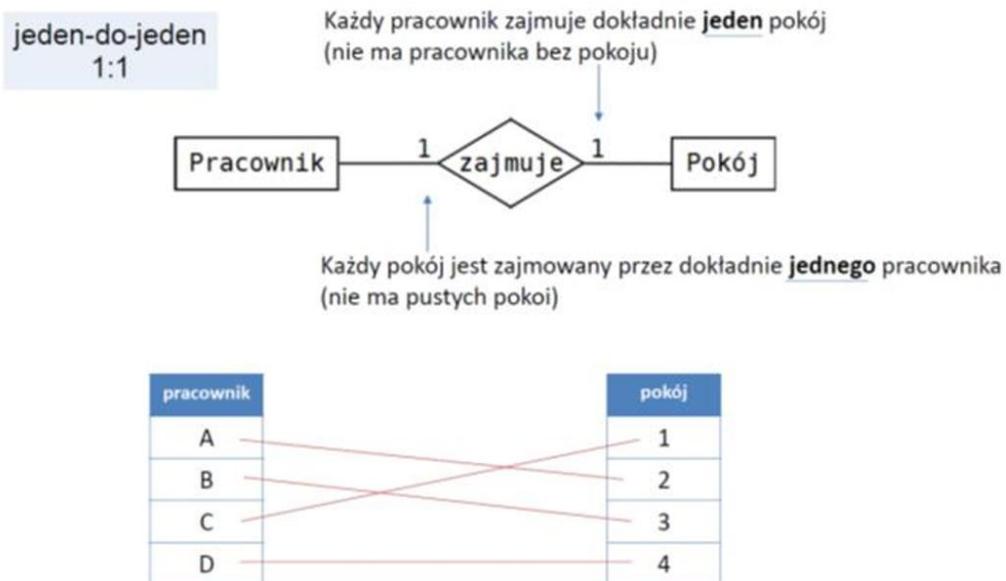
#### Stopień związku (kardynalność)

- unarny (binarny rekursywny)
- binarny
- ternarny
- n-arny (wieloczłonowy)

#### Istnienie

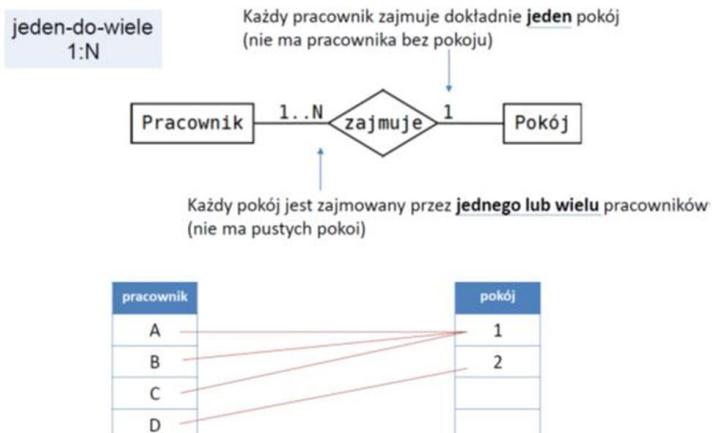
- opcjonalny (nieobowiązkowy)
- obowiązkowy (obligatoryjne)

### 3.1.5. Model E-R – Związki binarne obowiązkowe 1:1



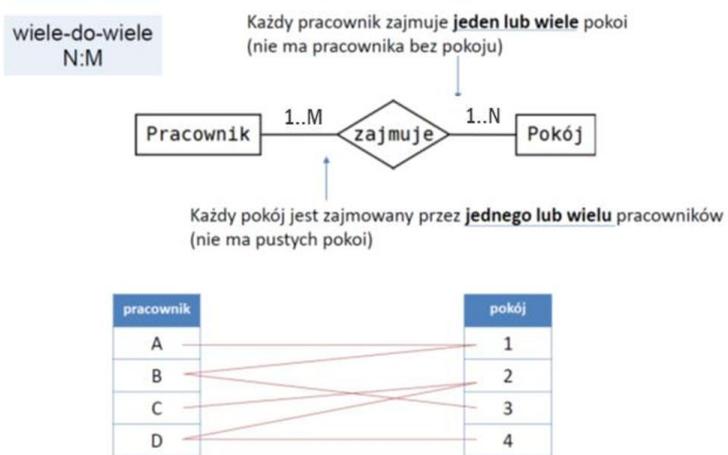
Rys. 12 Przykład związku jeden-do-jeden (1:1)

### 3.1.6. Model E-R – Związki binarne obowiązkowe 1:N



Rys. 13 Przykład związku jeden-do-wiele (1:N)

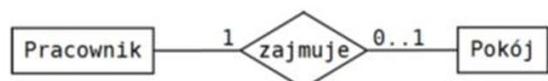
### 3.1.7. Model E-R – Związki binarne obowiązkowe M:N



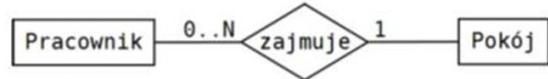
Rys. 14 Przykład związku wiele-do-wiele (M:N)

### Model E-R – Związki binarne opcjonalne

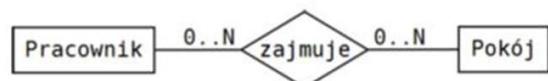
jeden-do-jeden



jeden-do-wiele

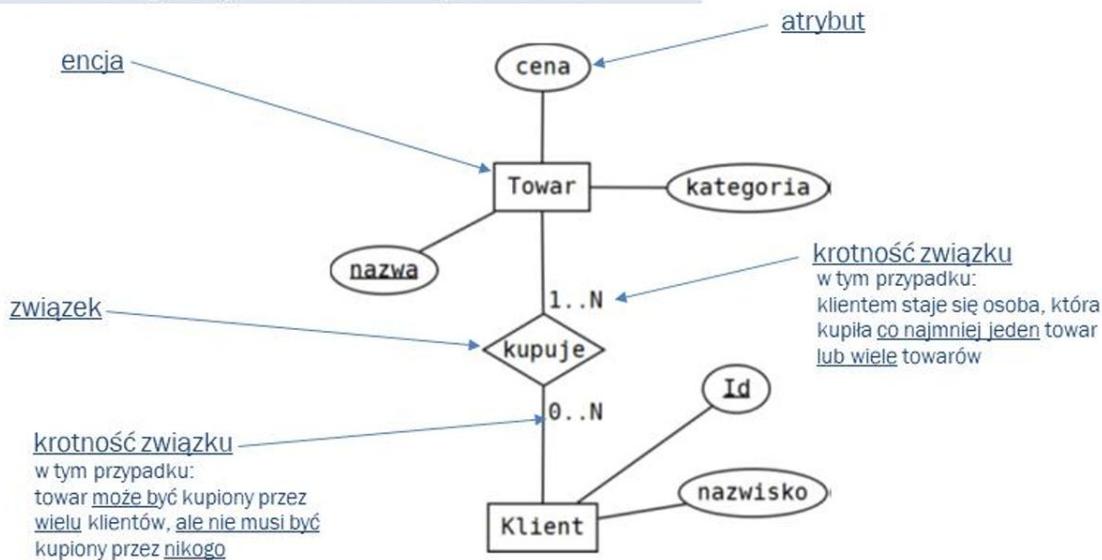


wiele-do-wiele



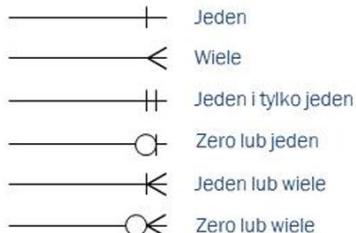
## Przykładowe Diagramy E-R:

### 3.1.01. Diagramy E-R – notacja Chen'a

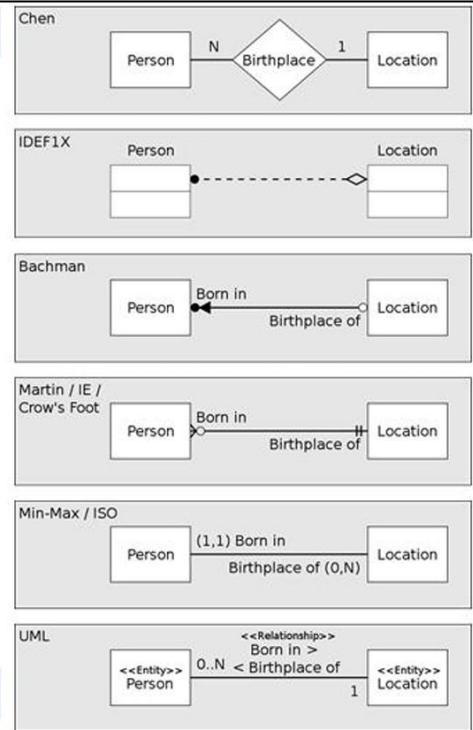


Rys. 7 Przykład schematu E-R w notacji Chen'a

### 3.1.02. Diagramy E-R – inne notacje



Rys. 8 Oznaczenia krotności w systemach CASE



Rys. 9 Przykłady diagramów E-R w różnych notacjach

### 3.1.1. Diagramy E-R – Encja

## 2. Rozszerzony Model ER (EER)

W miarę rozwoju systemów informacyjnych standardowy model ER bywa niewystarczający do oddania pełnej złożoności danych. Wprowadzono więc **rozszerzony model ER (EER)**, który dodaje nowe elementy:

### Generalizacja i Specjalizacja

- **Generalizacja:** Proces łączenia kilku encji szczegółowych w jedną encję ogólną.  
Przykład: encje *Samochód osobowy* i *Samochód ciężarowy* mogą być uogólnione do encji *Samochód*.
- **Specjalizacja:** Proces dzielenia encji ogólnej na podzbiory o bardziej szczegółowych właściwościach. Przykład: encja *Klient* może być wyspecjalizowana na *Klienta indywidualnego* i *Klienta biznesowego*.
- **Notacja:** Hierarchię tę przedstawia się zazwyczaj za pomocą trójkąta lub linii z etykietą "ISA" (od "is a"), łączącej encję ogólną z jej podklasami.

### Atrybuty Złożone i Wielowartościowe w EER

- Diagramy EER pozwalają na bardziej szczegółowe przedstawienie atrybutów, w tym atrybutów złożonych oraz wielowartościowych, co umożliwia dokładniejsze odwzorowanie rzeczywistych wymagań biznesowych.

### Dziedziczenie atrybutów

- W ramach specjalizacji, encje podrzędne dziedziczą wszystkie atrybuty encji nadzędnej, co pomaga uniknąć powtarzania tych samych danych w wielu miejscach.

<http://tadeusz.pankowski.pracownik.put.poznan.pl/06-model-eer.pdf>

[https://delibra.bg.polsl.pl/Content/78276/BCPS-87970\\_2013\\_Notacje-modelowania- 0000.pdf](https://delibra.bg.polsl.pl/Content/78276/BCPS-87970_2013_Notacje-modelowania- 0000.pdf)

### Przykłady Zastosowania Diagramów ER

#### System Zarządzania Biblioteką

- **Encje:** Książka, Autor, Czytelnik, Wypożyczenie.
- **Atrybuty:**
  - *Książka*: tytuł, ISBN, rok wydania.
  - *Autor*: imię, nazwisko, narodowość.
  - *Czytelnik*: imię, nazwisko, numer karty, adres.
- **Relacje:**
  - Czytelnik wypożycza Książkę (relacja M:N, którą rozwiązuje się tabelą asocjacyjną).
  - Książka napisana jest przez Autora (relacja 1:N – jeden autor może napisać wiele książek).
- **Diagram EER:** Można dodać atrybut wielowartościowy, np. słowa kluczowe dla książki lub dodatkową specjalizację w encji Czytelnik (np. Student, Nauczyciel).

Transformacja modelu ER do modelu relacyjnego to proces, w którym koncepcyjny model danych (diagram ER) jest przekształcany w logiczny model relacyjny – czyli zestaw tabel, kolumn, kluczy głównych i obcych, które można wdrożyć w systemie zarządzania bazą danych.

## 1. Od Encji do Tabel

- **Encja → Tabela:**

Każdą encję z diagramu ER przekształcamy w tabelę relacyjną.

**Przykład:** Encja *Klient* staje się tabelą *Klient*, gdzie atrybuty (np. *imie*, *nazwisko*, *adres*) stają się kolumnami.

- **Klucz główny:**

Każda encja powinna mieć unikalny identyfikator – klucz główny, który staje się PRIMARY KEY w tabeli.

## 2. Od Atrybutów do Kolumn

- **Atrybuty proste:**

Są bezpośrednio przekształcane na kolumny w tabeli.

- **Atrybuty złożone:**

Mogliśmy rozbić je na poszczególne składniki.

**Przykład:** Atrybut *adres* może być rozdzielony na kolumny *ulica*, *miasto*, *kod\_pocztowy*.

- **Atrybuty wielowartościowe:**

Ponieważ relacyjna tabela nie może bezpośrednio przechowywać wielu wartości w jednej kolumnie, przekształcamy atrybut wielowartościowy w **osobną tabelę**.

**Przykład:** Jeśli encja *Osoba* ma atrybut *języki*, tworzymy tabelę np. *Osoba\_Jezyki* z kolumnami *osoba\_id* (klucz obcy wskazujący na *Osoba*) i *język*.

### 3. Od Relacji do Tabel i Kluczy Obcych

- **Relacje 1:1:**

Możemy zintegrować obie encje w jedną tabelę, lub umieścić klucz obcy w jednej z tabel.

**Przykład:** Relacja między *Osoba* a *Paszport* (każda osoba ma jeden paszport) – możemy dodać do tabeli *Osoba* kolumnę *paszport\_id* jako klucz obcy do tabeli *Paszport* lub odwrotnie.

- **Relacje 1:N:**

Odwzorowujemy, dodając klucz obcy do tabeli znajdującej się po stronie "wiele".

**Przykład:** Relacja *Dział – Pracownicy*: Tabela *Pracownicy* otrzymuje kolumnę *dzial\_id* wskazującą na tabelę *Dział*.

- **Relacje M:N:**

Aby odwzorować relację wiele-do-wielu, tworzymy **tabelę asocjacyjną** (łącznikową), która zawiera klucze obce do obu tabel.

**Przykład:** Relacja między *Student* a *Kurs*: tworzymy tabelę *Student\_Kurs* z kolumnami *student\_id* oraz *kurs\_id* jako kluczami obcymi. Zazwyczaj te kolumny razem tworzą klucz główny tabeli asocjacyjnej.

- **Relacje n-arne (wielokrotne):**

Jeśli relacja łączy więcej niż dwie encje, tworzymy oddzielną tabelę, która przechowuje klucze obce do wszystkich uczestniczących tabel oraz ewentualne atrybuty związku.

**Przykład:** W relacji ternarnej *Student – Kurs – Nauczyciel*, tabela *Zapis* zawiera kolumny *student\_id*, *kurs\_id* oraz *nauczyciel\_id*.

## **Normalizacja**

[https://www.impan.pl/~kz/DB/KZ\\_BD\\_w06.pdf](https://www.impan.pl/~kz/DB/KZ_BD_w06.pdf)

Normalizacja to proces organizowania danych w bazie danych. Obejmuje ona tworzenie tabel i ustanawianie relacji między tymi tabelami zgodnie z regułami zaprojektowanymi zarówno w celu ochrony danych, jak i zwiększenia elastyczności bazy danych poprzez wyeliminowanie nadmiarowości i niespójnej zależności.

Nadmiarowe dane zajmują dodatkowe miejsce na dysku i przyczyniają się do problemów z konserwacją. Gdy trzeba zmienić dane istniejące w więcej niż jednym miejscu, dane we wszystkich tych lokalizacjach należy zmienić w dokładne taki sam sposób. Zmiana adresu klienta jest łatwiejsza do zaimplementowania, jeśli te dane są przechowywane tylko w tabeli Customers i nigdzie indziej w bazie danych.

Co to jest „niespójna zależność”? Chociaż intuicyjne jest, aby użytkownik poszukał w tabeli Customers adresu określonego klienta, może nie mieć sensu szukać tam wynagrodzenia pracownika, który wywołuje tego klienta. Wynagrodzenie pracownika jest powiązane z pracownikiem (zależne od niego), więc powinno zostać przeniesione do tabeli Pracownicy. Niespójne zależności utrudniają dostęp do danych, ponieważ może brakować ścieżki do odnalezienia danych lub może być ona uszkodzona.

Istnieje kilka reguł normalizacji bazy danych. Każda reguła jest nazywana "formularzem normalnym". Jeśli zostanie zaobserwowana pierwsza reguła, mówi się, że baza danych ma "pierwszą normalną formę". Jeśli zostaną zaobserwowane pierwsze trzy reguły, baza danych jest uważana za "trzecią normalną formę". Chociaż możliwe są inne poziomy normalizacji, trzecia normalna forma jest uważana za najwyższy poziom niezbędny dla większości aplikacji.

Podobnie jak w przypadku wielu formalnych reguł i specyfikacji, rzeczywiste scenariusze nie zawsze pozwalają na doskonałą zgodność. Normalizacja wymaga zwykle dodatkowych tabel i niektórzy klienci mogą uważać to za niewygodne. W razie decyzji o naruszeniu jednej z trzech pierwszych reguł normalizacji należy upewnić się, że aplikacja przewiduje wszelkie ewentualne problemy, na przykład nadmiarowe dane i niespójne zależności.

### **Pierwsza postać normalna**

- Wyeliminuj powtarzające się grupy w poszczególnych tabelach.
- Utwórz osobną tabelę dla każdego zestawu powiązanych danych.
- Zidentyfikuj każdy zestaw powiązanych danych za pomocą klucza podstawowego.

Nie używaj wielu pól w jednej tabeli do przechowywania podobnych danych. Na przykład aby można było śledzić pozycję magazynową mogącą pochodzić z dwóch źródeł, rekord magazynu może zawierać pola na kod pierwszego dostawcy i na kod drugiego dostawcy.

Co się stanie po dodaniu trzeciego dostawcy? Dodawanie pola nie jest odpowiedzią; Wymaga on modyfikacji programu i tabeli i nie uwzględnia w sposób płynny dynamicznej liczby dostawców. Zamiast tego najlepiej jest umieścić wszystkie informacje o dostawcach w osobnej tabeli o nazwie Dostawcy, a następnie połączyć magazyn z dostawcami za pomocą klucza numeru pozycji albo połączyć dostawców z magazynem za pomocą klucza kodu dostawcy.

## **Druga postać normalna**

- Utwórz osobne tabele dla zestawów wartości dotyczących wielu rekordów.
- Powiąż te tabele za pomocą klucza obcego.

Rekordy nie powinny zależeć od niczego innego niż klucz podstawowy tabeli (klucz złożony, jeśli to konieczne). Rozważmy na przykład adres klienta w systemie księgowym. Adres jest potrzebny w tabeli Klienci, ale także w tabelach Zamówienia, Wysyłka, Faktury, Rozrachunki z odbiorcami i Pobory należności. Zamiast przechowywać adres klienta w osobnym wpisie w każdej z tych tabel, należy przechowywać go w jednym miejscu, w tabeli Klienci lub w osobnej tabeli Adresy.

## **Trzecia postać normalna**

- Eliminuj pola, które nie zależą od klucza.

Wartości w rekordzie, które nie są częścią klucza tego rekordu, nie należą do tabeli. Ogólnie w każdym przypadku, w którym zawartość grupy pól może dotyczyć więcej niż jednego rekordu w tabeli, należy rozważyć umieszczenie tych pól w osobnej tabeli.

Na przykład w tabeli Rekrutacja pracowników można uwzględnić nazwę i adres uniwersytetu kandydata. Ale do wysyłek grupowych potrzebna jest pełna lista uniwersytetów. Jeśli informacje o uniwersytetach są przechowywane w tabeli Kandydaci, nie można utworzyć listy uniwersytetów bez bieżących kandydatów. Należy utworzyć osobną tabelę Uniwersytety i połączyć ją z tabelą Kandydaci za pomocą klucza kodu uniwersytetu.

**WYJĄTEK:** Przestrzeganie trzeciej normalnej formy, choć teoretycznie pożądanej, nie zawsze jest praktyczne. Jeśli jest używana tabela Klienci i trzeba wyeliminować wszelkie możliwe zależności między polami, należy utworzyć osobne tabele dla miast, kodów pocztowych, przedstawicieli handlowych, klas klientów oraz wszelkich innych czynników, które mogą zostać zduplikowane w wielu rekordach. Teoretycznie warto dążyć do normalizacji. Jednak stosowanie wielu małych tabel może pogorszyć wydajność lub spowodować przekroczenie dozwolonej liczby otwartych plików lub pojemności pamięci.

Niekiedy lepszym rozwiązaniem jest stosowanie trzeciej postaci normalnej tylko do danych często zmienianych. Jeśli pozostają jakieś pola zależne, projekt aplikacji powinien wymagać od użytkownika zweryfikowania wszystkich powiązanych pól po zmianie jednego z nich.

## **Inne postacie normalizacji**

Czwarta normalna forma, nazywana również Boyce-Codd formą normalną (BCNF), i piąta normalna forma istnieją, ale rzadko są brane pod uwagę w praktycznym projekcie. Pominięcie tych reguł może spowodować mniej niż doskonały projekt bazy danych, ale nie powinno mieć wpływu na funkcjonalność.

**Przykład:**

1. Tabela nieznormalizowana:

Nr studenta	Opiekun	Pokój opiekuna	Zajęcia 1	Zajęcia 2	Zajęcia 3
1022	Czarnecki	412	101-07	143-01	159-02
4123	Borkowski	216	101-07	143-01	179-04

2. Pierwsza normalna forma: brak powtarzających się grup

Tabele powinny mieć tylko dwa wymiary. Ponieważ jeden student może mieć kilka rodzajów zajęć, zajęcia powinny być wymienione w osobnej tabeli. Pola Zajęcia 1, Zajęcia 2 i Zajęcia 3 w powyższych rekordach sygnalizują problemy z projektem.

Arkusze kalkulacyjne często używają trzeciego wymiaru, ale tabele nie powinny. Innym sposobem przyjrzenia się temu problemowi jest relacja jeden do wielu, nie umieszczaj jednej strony i wielu stron w tej samej tabeli. Zamiast tego utwórz inną tabelę w pierwszej normalnej formie, eliminując powtarzaną grupę (Class#), jak pokazano w poniższym przykładzie:

Nr studenta	Opiekun	Pokój opiekuna	Nr zajęć
1022	Czarnecki	412	101-07
1022	Czarnecki	412	143-01
1022	Czarnecki	412	159-02
4123	Borkowski	216	101-07
4123	Borkowski	216	143-01
4123	Borkowski	216	179-04

### 3. Druga normalna forma: wyeliminowanie nadmiarowych danych

Zwróć uwagę na wiele wartości **Class#** dla każdej wartości **Student#** w powyższej tabeli. Klasa# nie jest funkcjonalnie zależna od studenta# (klucz podstawowy), więc ta relacja nie jest w drugiej normalnej formie.

W poniższych dwóch tabelach pokazano drugą formę normalną:

Studenci:

Nr studenta	Opiekun	Pokój opiekuna
1022	Czarnecki	412
4123	Borkowski	216

Rejestracja:

Nr studenta	Nr zajęć
1022	101-07
1022	143-01
1022	159-02
4123	101-07
4123	143-01
4123	179-04

4. Trzecia normalna forma: wyeliminowanie danych zależnych od klucza

W ostatniej tabeli wartości Pokój opiekuna są funkcjonalnie zależne od atrybutu Opiekun. Rozwiązaniem jest przeniesienie tego atrybutu z tabeli Studenci do tabeli Wykładowcy, jak pokazano poniżej:

Studenci:

Nr studenta	Opiekun
1022	Czarnecki
4123	Borkowski

Wykładowcy:

Name (Nazwa)	Pokój	Wydział
Czarnecki	412	42
Borkowski	216	42

## **Denormalizacja**

Dobrze wytłumaczony proces denormalizacji można znaleźć pod linkiem:  
<https://codegym.cc/pl/quests/lectures/pl.questhibernate.level17.lecture07>

### **Co to jest denormalizacja?**

Denormalizacja w kontekście relacyjnych baz danych odnosi się do procesu strategicznego organizowania danych w mniej ustrukturyzowany lub redundantny sposób w celu optymalizacji wydajności zapytań, zmniejszenia kosztów wyszukiwania danych i zwiększenia wydajności operacyjnej. W przeciwieństwie do normalizacji, która ma na celu zminimalizowanie nadmiarowości i zależności w schemacie bazy danych poprzez podzielenie danych na mniejsze, powiązane tabele, denormalizacja celowo wprowadza nadmiarowości w celu konsolidacji danych i minimalizowania potrzeby wykonywania skomplikowanych operacji łączenia, które mogą potencjalnie obniżyć wydajność systemu. Chociaż normalizacja jest niezbędna do poprawy integralności i spójności systemu baz danych, często odbywa się kosztem wydajności zapytań. W wysoce znormализowanych schematach dostęp do pełnego zestawu danych zazwyczaj wymaga wielu operacji łączenia w różnych tabelach w celu ponownego złożenia informacji prezentowanych użytkownikom końcowym, co pochłania więcej zasobów i czasu. W rezultacie można zastosować techniki denormalizacji w celu zrównoważenia kompromisów między spójnością danych, integralnością i wydajnością zapytań. Denormalizację przeprowadza się poprzez scalanie tabel, dodawanie zbędnych kolumn lub utrzymywanie wstępnie obliczonych danych podsumowujących w celu uproszczenia i przyspieszenia operacji pobierania danych. Aby to zilustrować, rozważ wysoce znormализowany schemat bazy danych handlu elektronicznego, w którym informacje o klientach, zamówieniach i produktach są przechowywane w oddzielnych tabelach. Podczas sprawdzania listy zamówień wraz z odpowiednimi szczegółami klienta i produktu konieczne jest wykonanie wielu operacji łączenia w celu uzyskania niezbędnych informacji. W zdenormalizowanym schemacie do tabeli zamówień można dodać nadmiarowe kolumny, takie jak nazwa\_klienta i nazwa\_produktu, aby wyeliminować potrzebę operacji łączenia i zwiększyć wydajność zapytań. Należy zauważyc, że denormalizacja nie ma uniwersalnego zastosowania i do jej wdrożenia należy podchodzić rozsądnie. Ponieważ nadmiarowość z natury zwiększa poziom złożoności schematu bazy danych i zarządzania nią, denormalizacja może zwiększyć ryzyko niespójności i anomalii danych. Wymaga zatem czujnego monitorowania i odpowiednich mechanizmów egzekwowania integralności danych, aby zapewnić spójność i dokładność danych. Co więcej, denormalizacja nie zawsze może skutkować poprawą wydajności, a w niektórych przypadkach może prowadzić do pogorszenia wydajności systemu ze względu na zwiększone zużycie pamięci masowej i koszty zapisu.

### **Dlaczego stosujemy denormalizację?**

- Poprawa wydajności odczytu:**

W systemach, gdzie operacje odczytu są o wiele częstsze niż zapisy (np. systemy raportowe, hurtownie danych), łączenie danych w jednej tabeli eliminuje kosztowne operacje JOIN między wieloma tabelami.

- Zmniejszenie liczby zapytań łączących dane:**

Jeśli aplikacja często musi pobierać dane z kilku tabel, denormalizacja pozwala na dostęp do informacji w jednym zapytaniu, co redukuje czas przetwarzania.

- **Uproszczenie zapytań:**

Zapytania SQL stają się prostsze i bardziej czytelne, gdy dane są zgrupowane w jednej strukturze, zamiast rozproszonej po wielu tabelach.

### Przykłady denormalizacji

1. **Scenariusz e-Commerce:**

W znormalizowanym modelu relacyjnym dane o produkcie mogą być przechowywane w tabeli *Produkty*, a informacje o kategorii w osobnej tabeli *Kategorie*. Aby przyspieszyć zapytania dotyczące prezentacji produktów na stronie sklepu, można denormalizować i dodać kolumnę nazwa\_kategorii bezpośrednio do tabeli *Produkty*.

2. **System raportowy:**

Jeśli raporty muszą często agregować dane z wielu tabel (np. sprzedaż, koszty, marże), można przygotować **materializowany widok**, który łączy dane z różnych źródeł w jedną strukturę i jest aktualizowany okresowo.

Zestawienie porównawcze cech metamodeli:

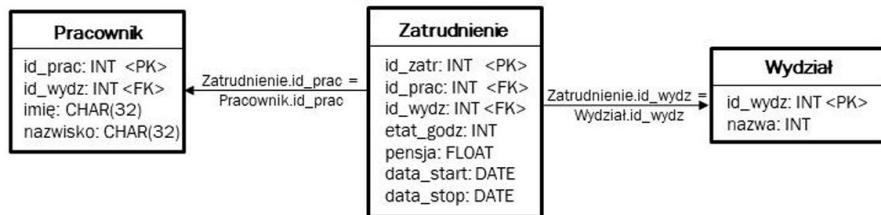
## GŁÓWNE KONCEPCJE SYSTEMÓW BAZODANOWYCH – ZESTAWIENIE PORÓWNAWCZE CECH METAMODELI

- Metamodel relacyjny (RDB)<sup>(2)</sup>
  - pojęcia pierwotne: {atrybut, schemat relacji, relacja, krotka (n-tka), domena aktywna, klucz główny, klucz obcy, rola}
  - podstawowe własności: RDB jest metamodellem baz danych opartym o relacje (tabele), nie rozdziela danych i związków między nimi, modeluje tylko związki binarne, krotność związków: jeden-do-jednego i jeden-do-wielu, związki między relacjami mają charakter niejawny, nie posiada ograniczenia uczestnictwa
  - graficzna notacja modeli: diagram relacyjny (ang. *relational diagram*, RD)
- Metamodel związków-encji (E-R)<sup>(3)</sup>
  - pojęcia pierwotne: {encja, związek, zbiór związków, wartość, zbiór wartości, rola, atrybut, klucz encji, zależność istnienia}
  - podstawowe własności: E-R rozdziela dane od związków między nimi, modeluje związki n-arne, krotności związków: jeden-do-jednego, jeden-do-wielu i wiele-do-wielu, związki między encjami mają charakter jawnego, ograniczenie uczestnictwa jest jednoznacznie określone
  - graficzna notacja modeli: diagram związków-encji (ang. *entity-relationships diagram*, ERD)
- Metamodel obiektowy (OM)<sup>(4)</sup>
  - pojęcia pierwotne: {obiekt, literal (stan obiektu, właściwości, zachowanie, interfejs, klasa, struktura, typy proste, typy definiowane, generalizacja-specjalizacja, związek rozszerzenia, zasięg typu, klucz, nazwa obiektu, czas życia obiektów, obiekt zbiorowy, obiekt strukturalny, atrybuty, związki, operacja, metadane)}
  - podstawowe własności: OM integruje w sposób przeźroczysty możliwości bazy danych z językiem programowania, przechowuje obiekty, umożliwia ich współdzielenie przez wielu użytkowników i wiele aplikacji, bazuje na schemacie określonym w ODL i zawiera instancje typów zdefiniowanych przez schemat, modeluje tylko związki binarne, częściowo rozdziela dane od związków między nimi, modeluje związki n-arne, krotności związków: jeden-do-jednego, jeden-do-wielu i wiele-do-wielu, nie oferuje ograniczenia uczestnictwa w związku
  - graficzna notacja modeli: zdobudowany ze związków, klas i interfejsu

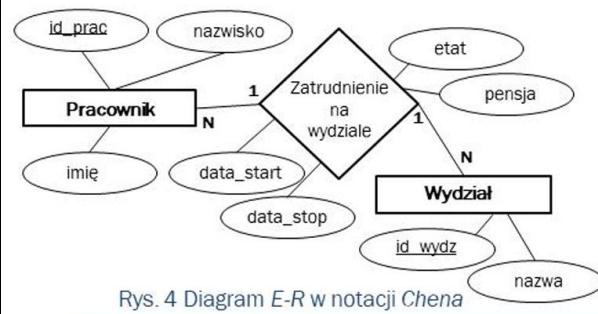
## PRZYKŁADY GRAFICZNYCH NOTACJI MODELI

Studium przypadku

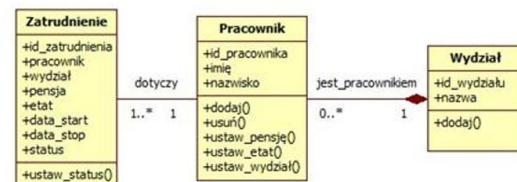
Pewna firma zatrudnia pracowników na wydziałach. Pracownik jest zatrudniony na cały, bądź część etatu (czas pracy) i ma określone uposażenie (pensja). Zarówno czas pracy, jak i pensja mogą ulegać zmianom.



Rys. 3 Diagram RDB



Rys. 4 Diagram E-R w notacji Chena



Rys. 5 Diagram OM

Etapy projektowania aplikacji bazodanowej:



## **Zadanie 1 – Odpowiedz na pytania**

1. Czym są diagramy ER i jakie podstawowe elementy w nich występują? Opisz encje, atrybuty i relacje oraz podaj przykłady dla każdej z tych kategorii.
2. Co oznaczają akronimy RDB oraz E-R?
3. Czym jest ERD?
4. Co to jest atrybut wielowartościowy w diagramie ER? Jakie oznaczenia stosujemy dla atrybutów wielowartościowych? Podaj przykład z życia codziennego, gdzie zastosowanie atrybutu wielowartościowego jest uzasadnione.
5. Wyjaśnij różnice między relacjami binarnymi a n-arnymi (wielokrotnymi) w diagramie ER. Podaj przykład relacji ternarnej i omów, dlaczego w niektórych sytuacjach lepiej jest modelować relację jako n-arną, a nie rozbijać ją na kilka relacji binarnych.
6. Co to jest generalizacja i specjalizacja w modelu ER? Jakie korzyści płyną z zastosowania mechanizmów dziedziczenia (ISA)? Podaj przykład, w którym encja ogólna została wyspecjalizowana na podklasy, oraz omów, jakie atrybuty są dziedziczone.
7. Jakie są główne zasady przekształcania encji z diagramu ER w tabele relacyjne? Co powinno znaleźć się w tabeli, aby prawidłowo odwzorować encję?
8. Jak przekształcamy atrybuty proste, złożone oraz wielowartościowe z modelu ER do modelu relacyjnego? Podaj przykład, jak rozbić atrybut złożony (np. adres) oraz jak odwzorować atrybut wielowartościowy (np. języki) przy użyciu dodatkowej tabeli.
9. W jaki sposób odwzorowujemy relacje 1:1, 1:N i M:N w modelu relacyjnym? Podaj przykłady tworzenia kluczy obcych oraz tabel asocjacyjnych dla relacji M:N.
10. Co to jest denormalizacja i dlaczego może być stosowana w projektowaniu baz danych? W jakich sytuacjach denormalizacja przynosi największe korzyści?
11. Jakie są główne zalety denormalizacji (np. szybszy odczyt, uproszczenie zapytań) i jakie potencjalne problemy mogą się z nią wiązać (np. ryzyko niespójności, zwiększone zużycie pamięci)?

## Zadanie 2 - Tworzenie relacyjnego modelu bazy danych

### 1. Cel ćwiczenia

Celem zadania jest zapoznanie się z procesem tworzenia relacyjnego modelu bazy danych w języku SQL. W szczególności studenci nauczą się:

- Definiowania tabel w bazie danych (tworzenie schematów tabel, typów danych, kluczy podstawowych).
- Tworzenia relacji między tabelami (klucze obce).
- Wstawiania danych do tabeli i odczytywania ich zawartości.

### 2. Opis zadania

Należy utworzyć 4 tabele: **Tutor**, **Student**, **Lesson** oraz **Course**. Każda tabela powinna zawierać klucz główny (PRIMARY KEY). W wybranych tabelach należy zdefiniować klucze obce (FOREIGN KEY) tak, aby poprawnie odzwierciedlać relacje między rekordami.

Do tego celu można wykorzystać:

<https://dbdiagram.io/d>

<https://app.quickdatabasediagrams.com/#/>

<https://paiza.io/projects/hjPZpaaa3WfiFWbU47tDpg?language=mysql>

### 3. Struktura modelu

- **Tutor**

Kolumny:

- TutorID (INT, NOT NULL, klucz główny)
- Name (VARCHAR(30), NOT NULL)
- Title (VARCHAR(30), NOT NULL)

- **Student**

Kolumny:

- StudentID (INT, NOT NULL, klucz główny)
- Name (VARCHAR(30), NOT NULL)
- DepartmentID (INT, NOT NULL)
- Address (VARCHAR(50), NOT NULL)
- DateOfBirth (DATE, NOT NULL)

- **Lesson**

Kolumny:

- LessonID (INT, NOT NULL, klucz główny)
- Description (VARCHAR(30), NOT NULL)

- TutorID (INT, NOT NULL) — klucz obcy odwołujący się do **Tutor(TutorID)**
- **Course**  
Kolumny:
  - CourseID (INT, NOT NULL, klucz główny)
  - StudentID (INT, NOT NULL) — klucz obcy odwołujący się do **Student(StudentID)**
  - LessonID (INT, NOT NULL) — klucz obcy odwołujący się do **Lesson(LessonID)**
  - Description (VARCHAR(50), NOT NULL)
  - StartDate (DATE, NOT NULL)
  - EndDate (DATE, NOT NULL)

4. Utwórz powyższy model

Przykład:

```
CREATE TABLE Tutor (
    TutorID INT NOT NULL,
    Name VARCHAR(30) NOT NULL,
    Title VARCHAR(30) NOT NULL,
    CONSTRAINT PK_Tutor PRIMARY KEY (TutorID)
);
```

Resztę tabel utworzyć analogicznie do przykładu

## 5. Wstawianie danych przykładowych

Na potrzeby testów należy wstawić co najmniej jeden rekord do każdej tabeli. Poniżej przedstawiono przykładowe wiersze:

### Tutor

```
INSERT INTO Tutor (TutorID, Name, Title)  
VALUES (1, 'Mateusz', 'Test');
```

### Student

```
INSERT INTO Student (StudentID, Name, DepartmentID, Address, DateOfBirth)  
VALUES (100, 'Jan Nowak', 1, 'ul. Kwiatowa 5, Warszawa', '1990-05-15');
```

### Lesson

```
INSERT INTO Lesson (LessonID, Description, TutorID)  
VALUES (10, 'Język SQL – wprowadzenie', 1);
```

### Course

```
INSERT INTO Course (CourseID, StudentID, LessonID, Description, StartDate,  
EndDate)  
VALUES (200, 100, 10, 'Kurs baz danych', '2025-01-01', '2025-02-01');
```

## 6. Weryfikacja danych

Aby upewnić się, że dane zostały wstawione prawidłowo, wyświetl zawartość każdej tabeli:

```
SELECT * FROM Tutor;  
SELECT * FROM Student;  
SELECT * FROM Lesson;  
SELECT * FROM Course;
```

### Zadanie 3 - Transformacja modelu konceptualnego do relacyjnego

[https://www.impan.pl/~kz/DB/KZ\\_BD\\_w05.pdf](https://www.impan.pl/~kz/DB/KZ_BD_w05.pdf)

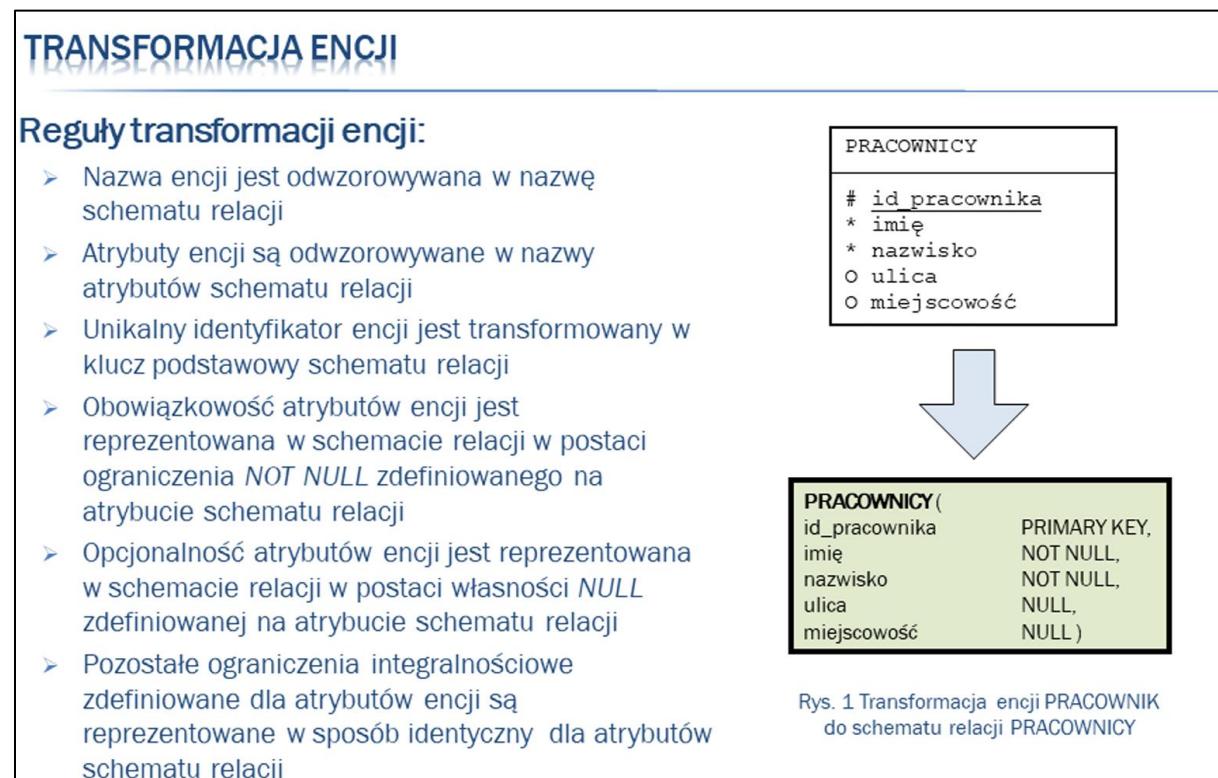
[http://smurf.mimuw.edu.pl/external\\_slides/W4\\_Transformacja\\_modelu\\_ER\\_do\\_modelu\\_relacyjnego/W4\\_Transformacja\\_modelu\\_ER\\_do\\_modelu\\_relacyjnego.html](http://smurf.mimuw.edu.pl/external_slides/W4_Transformacja_modelu_ER_do_modelu_relacyjnego/W4_Transformacja_modelu_ER_do_modelu_relacyjnego.html)

<https://tsadowski.pl/lekcja/905>

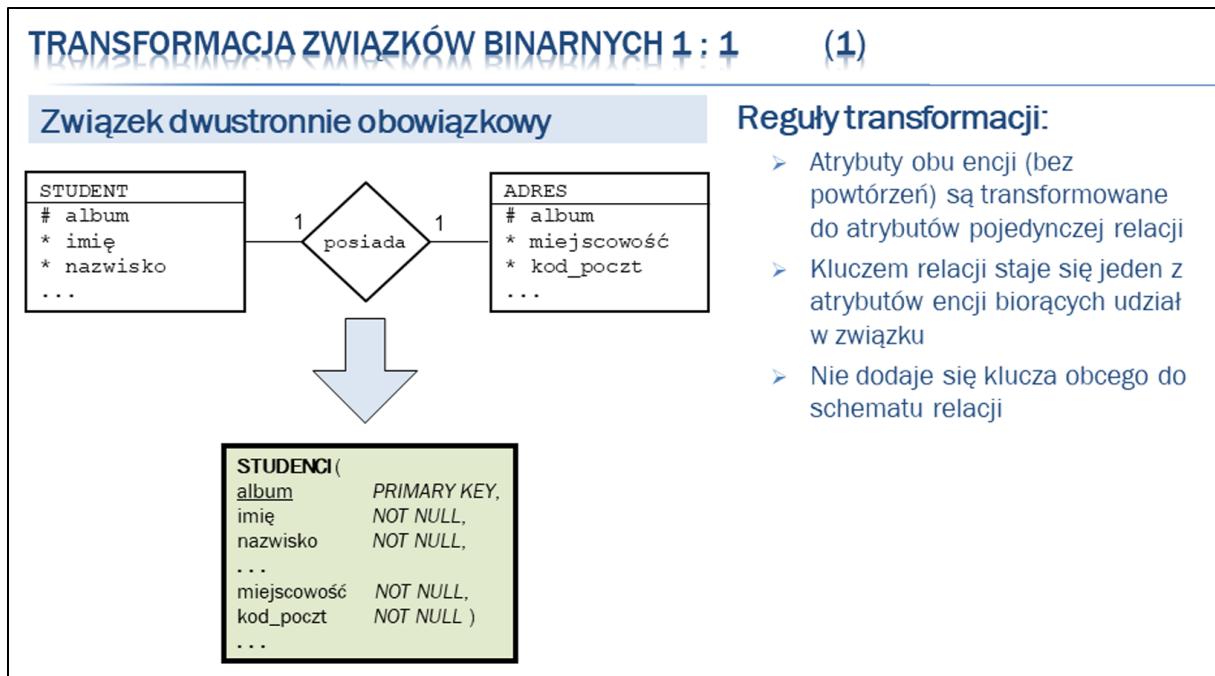
Transformacja modelu konceptualnego do relacyjnego to kluczowy etap w projektowaniu bazy danych, który polega na przekształceniu encji, atrybutów i relacji z modelu konceptualnego w tabele, kolumny i relacje w modelu relacyjnym. Ten proces formalizuje projekt bazy danych i przygotowuje go do implementacji w systemie zarządzania bazą danych (DBMS).

Przykłady:

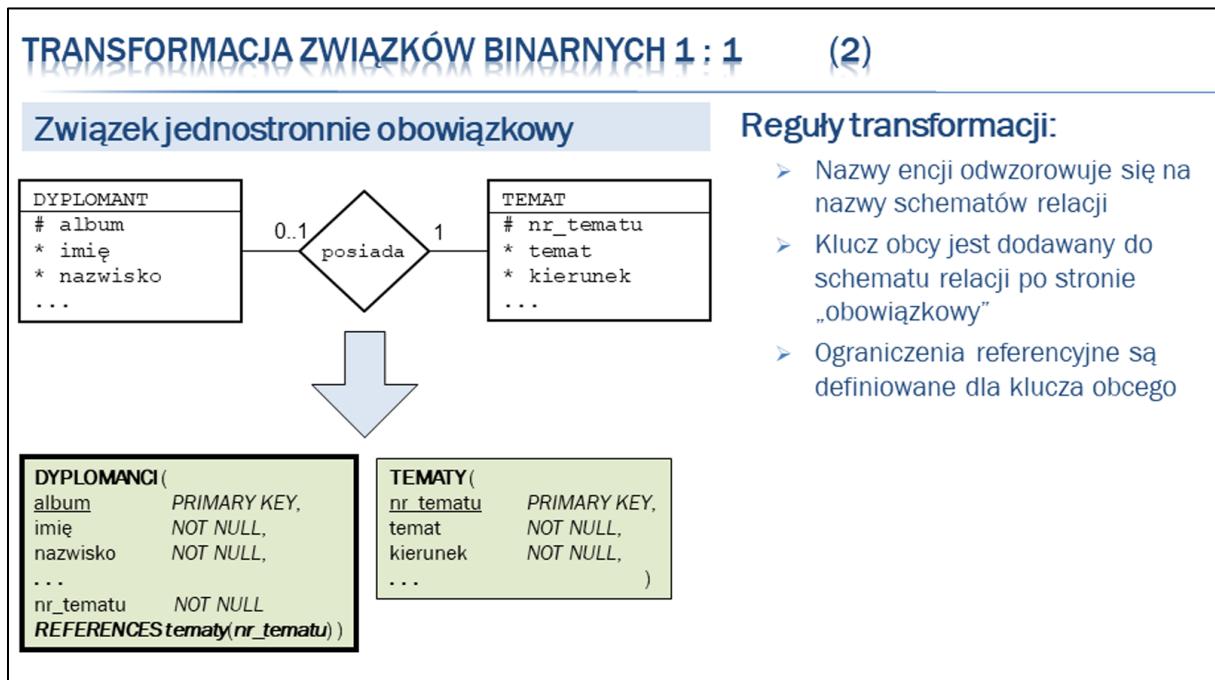
Przykład 1)



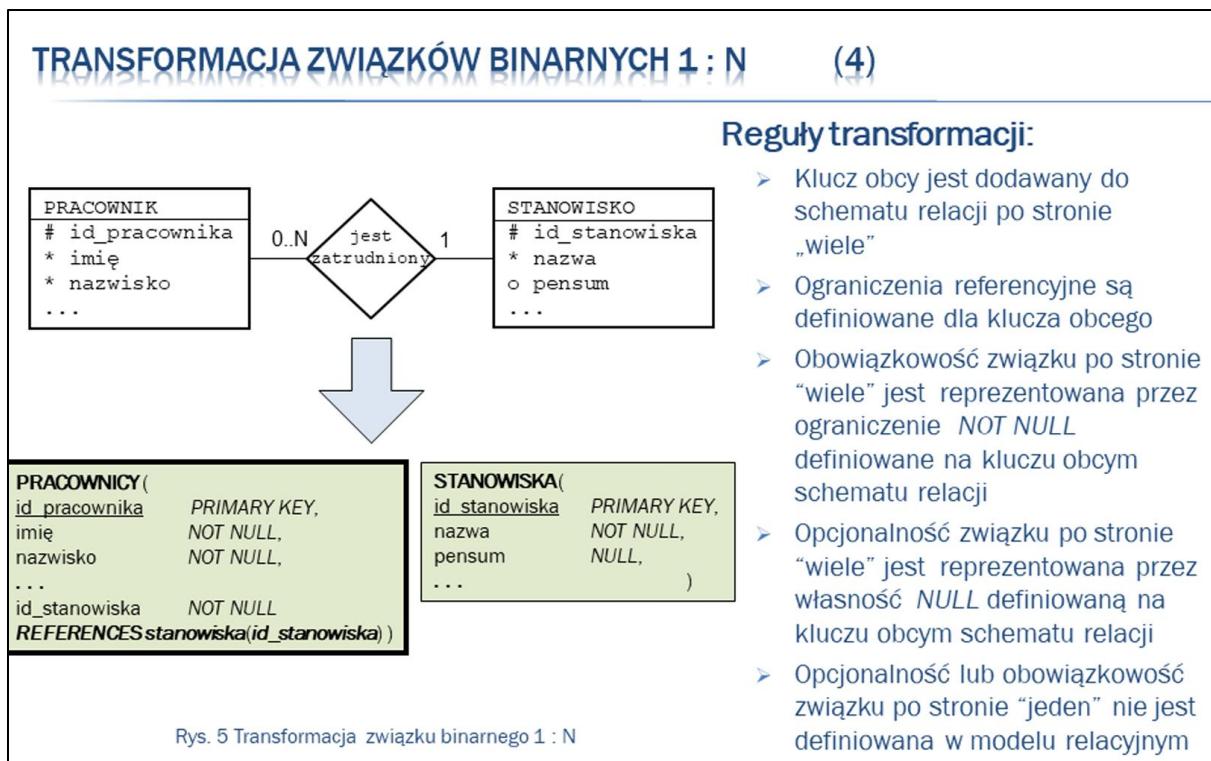
Przykład 2)



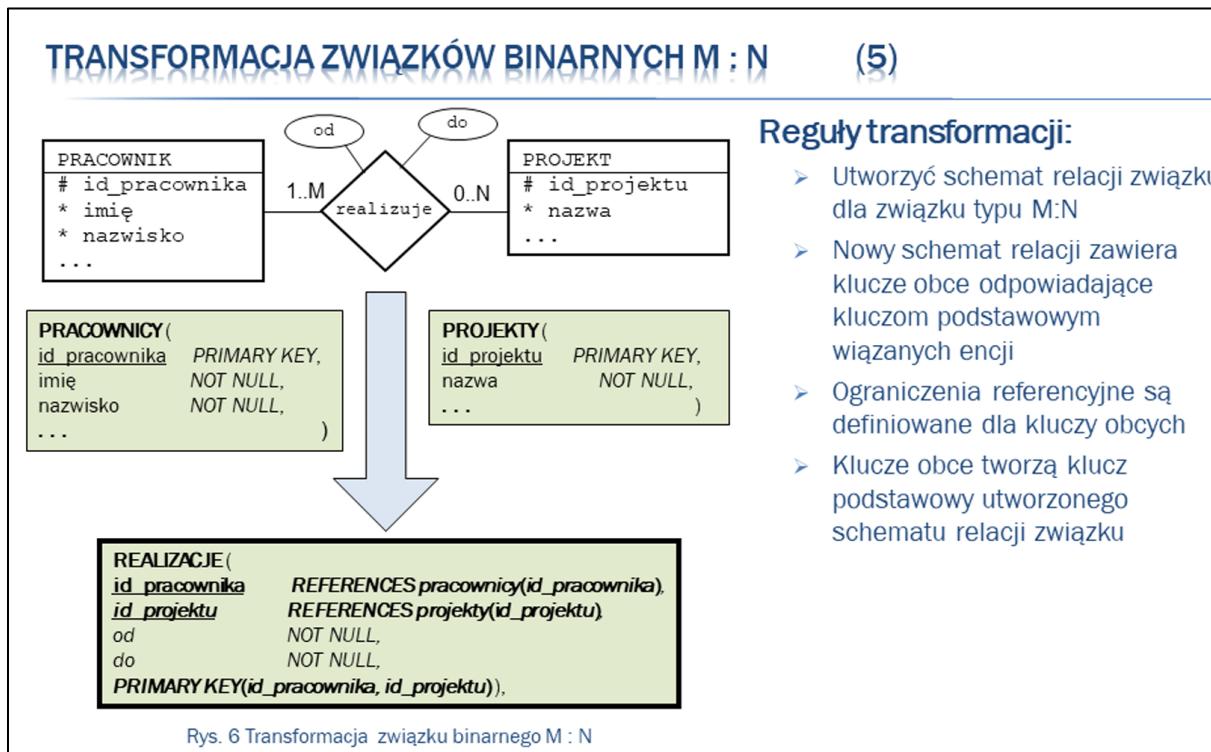
Przykład 3)



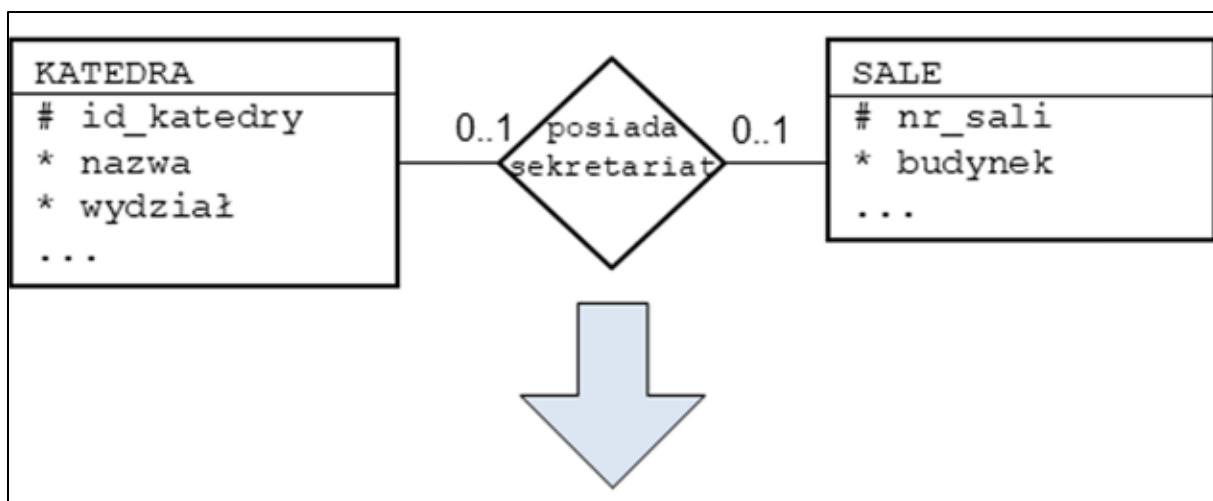
Przykład 4)



Przykład 5)



Przetransformuj poniższy diagram do schematu relacyjnego.



#### **Zadanie 4 – Podsumowanie**

W ramach powtórki proszę o przeczytanie:

- <https://codegym.cc/pl/quests/lectures/pl.questhibernate.level17.lecture00>
- <https://codegym.cc/pl/quests/lectures/pl.questhibernate.level17.lecture01>
- <https://codegym.cc/pl/quests/lectures/pl.questhibernate.level17.lecture02>
- <https://codegym.cc/pl/quests/lectures/pl.questhibernate.level17.lecture03>
- <https://codegym.cc/pl/quests/lectures/pl.questhibernate.level17.lecture04>
- <https://codegym.cc/pl/quests/lectures/pl.questhibernate.level17.lecture07>

## Zadanie Dodatkowe 😊

### Opis Scenariusza

W systemie rejestracji studenckiej mamy trzy podstawowe encje:

- **Student:**
  - Atrybuty: *student\_id, imię, nazwisko, email, oraz numery\_telefonu* (atrybut wielowartościowy).
- **Course (Kurs):**
  - Atrybuty: *course\_id, nazwa, opis.*
- **Registration (Rejestracja):**
  - Reprezentuje relację M:N między studentem a kursem (student może zapisać się na wiele kursów, a kurs może mieć wielu studentów).

### Etapy Zadania

#### 1. Tworzenie Diagramu ER:

- Narysuj prosty diagram ER, w którym:
  - Encja **Student** zawiera atrybuty: *student\_id, imię, nazwisko, email, numery\_telefonu.*
  - Encja **Course** zawiera atrybuty: *course\_id, nazwa, opis.*
  - Relacja między **Student** a **Course** jest relacją M:N, którą odwzorujesz w modelu relacyjnym za pomocą tabeli asocjacyjnej (np. o nazwie **Registration**).

#### 2. Transformacja do Modelu Relacyjnego:

Napisz polecenie SQL które utworzą wszystkie tabele:

- **Tabela Student:**

Utwórz tabelę, w której atrybuty *student\_id, imię, nazwisko i email* stanowią kolumny.

Przykład:

```
CREATE TABLE Student (
    student_id INT PRIMARY KEY,
    imie VARCHAR(50),
    nazwisko VARCHAR(50),
    email VARCHAR(100)
);
```

Analogicznie utwórz resztę tabel.

- **Tabela Student\_Telefony:**

Ponieważ *numery\_telefonu* to atrybut wielowartościowy, utwórz oddzielną tabelę zawierającą kolumny: *student\_id* (klucz obcy) oraz *telefon*. Klucz główny może być kombinacją obu kolumn.

- **Tabela Course:**

Utwórz tabelę dla kursów z kolumnami: *course\_id, nazwa, opis*.

- **Tabela Registration:**

Utwórz tabelę asocjacyjną, która łączy studentów z kursami. Tabela ta powinna zawierać kolumny: *student\_id* oraz *course\_id*, które razem tworzą klucz główny oraz są kluczami obcymi do tabel Student i Course.

Aby zweryfikować działanie modelu, można wstawić przykładowe rekordy do każdej z tabel.