Modelowanie i Analiza Systemów informacyjnych (MAS)

dr inż. Mariusz Trzaska
mtrzaska@pjwstk.edu.pl

Wykład 9

Wykorzystanie modelu relacyjnego w obiektowych językach programowania

Zagadnienia

- Model relacyjny
- Przyczyny popularności relacyjnych baz danych.
- Mapowanie poszczególnych konstrukcji z obiektowości
  - Klasy,
  - Asocjacje,
  - Dziedziczenie.
- Relacyjne bazy danych w obiektowych językach programowania (m. in. JDBC).

Wykorzystano fragmenty wykładu „ Relacyjne Bazy Danych” opracowanego przez L. Banachowski, J. Wierzbicki.

Model relacyjny

- Twórca: Edgar Codd (1970)
- Pojęcia
  - Tabele
  - Relacje
  - Klucze

Model relacyjny (2)

- Tabele
  - Liczba kolumn jest z góry ustalona.
  - Z każdą kolumną jest związana jej nazwa oraz dziedzina, określająca zbiór wartości, jakie mogą wystąpić w kolumnie.
  - Na przecięciu wiersza i kolumny znajduje się pojedyncza (atomowa) wartość należąca do dziedziny kolumny.
  - Wiersz reprezentuje jeden rekord informacji np. osobę.
  - W modelu relacyjnym abstrahujemy od kolejności wierszy (rekordów) i kolumn (pól w rekordzie).

Model relacyjny (3)

- Tabele – c. d.

IdWykładowcy	Imię	Nazwisko	Tytuł
237	Jan	Kowalski	Doktor
3245	Maciej	Jankowski	Docent
8976	Artur	Malinowski	Profesor

NazwaPrzedmiotu	Kod	IdWykładowcy
Bazy danych	BDA	1237
Projektowanie systemów informacyjnych	PSI	3245
Technologie internetowe	TIN	3245
Programowanie obiektowe	POB	8976
Systemy decyzyjne	SDE	1237

Model relacyjny (4)

- Znaczenie kolumny IdWykladowcy
  - Jej wartość nie opisuje cechy wykładu.
  - Reprezentuje związek danego przedmiotu z wykładowcą, o którym informacja znajduje się w innej tabeli i tylko korzystając z identyfikatora możemy rozpoznać w innej tabeli wiersz właściwego wykładowcy i odczytać o nim informacje.
  - Istotne jest więc, aby identyfikator ten jednoznacznie określał danego wykładowcę - w modelu relacyjnym nie ma innej możliwości identyfikacji wiersza tylko poprzez wartości kolumn, które jednoznacznie identyfikują wiersz.

Model relacyjny (5)

- Klucz
  - Główny. Jedna lub więcej kolumn, w których wartości jednoznacznie identyfikują cały wiersz.
  - Obcy
    - Klucz obcy jest to jedna lub więcej kolumn, których wartości występują jako wartości ustalonego klucza głównego w tej lub innej tabeli i są interpretowane jako wskaźniki do wierszy w tej drugiej tabeli.
    - Przykład: w tabeli Przedmioty kluczem obcym jest IdWykładowcy, którego wartości pochodzą z kolumny IdWykładowcy w tabeli Wykładowcy.

Model relacyjny (6)

- Wartość *Null*
  - Dziedziny kolumn (dopuszczalne wartości) są rozszerzane o specjalny obiekt `Null` - oznaczający brak wartości.
  - Brak ten może być
    - chwilowy,
    - wynikający z dziedziny biznesowej.
  - `NULL` jest czymś innym niż `0` lub „”
- Indeks
  - Umożliwia błyskawiczne wyszukiwanie wartości w kolumnach objętych indeksem.

Popularność modelu relacyjnego

- Popularność
  - modelu relacyjnego,
  - czy raczej popularność baz danych?
- Rola systemów spadkowych (*legacy*)
- Termin „baza danych” kojarzony jest głównie z relacyjną bazą danych.
- Prawdopodobnie w przyszłości ulegnie to zmianie.
  - Biblioteki zapewniające m. in. trwałość danych, np. Hibernate.
  - Obiektowe bazy danych.

Bazy danych

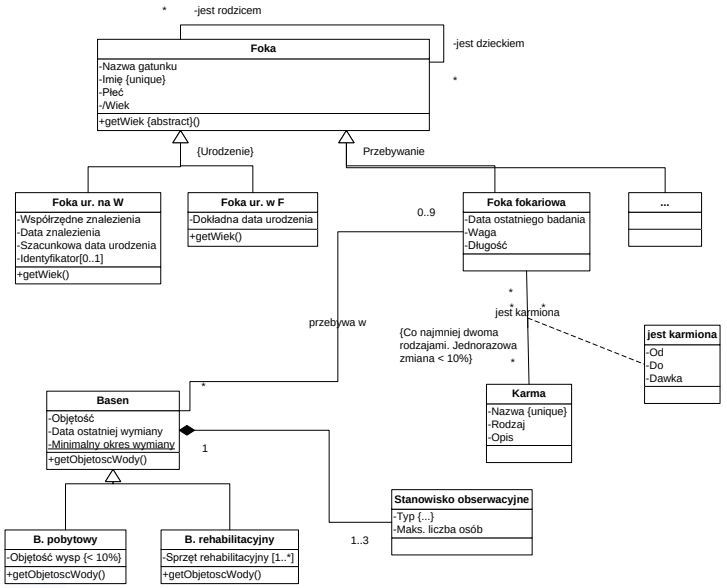
- Główne powody korzystania z baz danych:
  - Język zapytań,
  - Iteracyjny sposób zapisywania danych,
  - Szybkość działania,
  - Bezpieczeństwo danych,
  - Bezpieczeństwo dostępu do danych,
  - I oczywiście zdolność do przechowywania dużej ilości informacji.
- Wady relacyjnych baz danych
  - model relacyjny ☺
  - Inne…

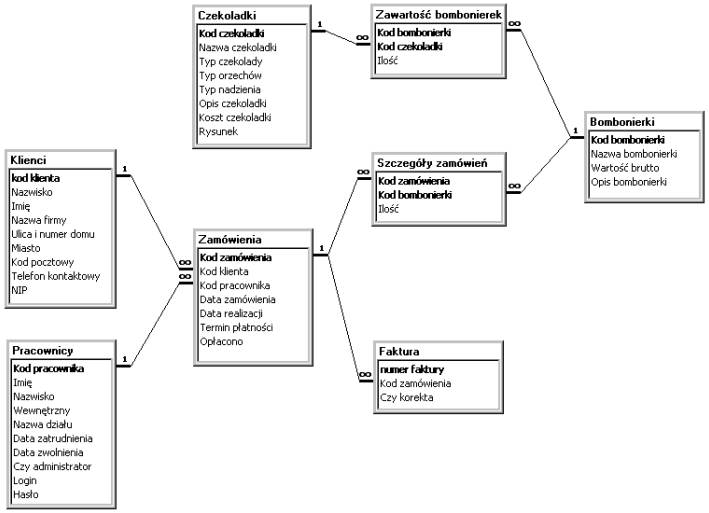
Wykorzystanie baz danych

- Przykrą konsekwencją wykorzystywania relacyjnych baz danych z poziomu obiektowych języków programowania jest ***niezgodność impedancji***.
- Polega na konieczności dopasowania do siebie „dwóch różnych światów”:
  - Relacyjnego,
  - Obiektowego.

Wykorzystanie baz danych (2)

- Niezgodność impedancji





Przejście z modelu obiektowego na relacyjny

- Klasy
  - Atrybuty,
    - proste i złożone
    - wymagane i opcjonalne
    - pojedyncze i powtarzalne
    - obiektu i klasowe
    - wyliczalne
  - Metody
    - obiektu,
    - klasowe,
    - przesłonięcie i przeciążenie.

Przejście z modelu obiektowego na relacyjny (2)

- Asocjacje
  - binarne,
  - z atrybutem,
  - kwalifikowane,
  - n-arne,
  - agregacje,
  - kompozycje.

Przejście z modelu obiektowego na relacyjny (3)

- Dziedziczenie
  - *overlapping*,
  - *complete*, *incomplete*,
  - *multi-inheritance*,
  - *multi-aspect*,
  - *dynamic*.

Mapowanie klas

- Klasy zastępujemy tabelami.
- Każdy atrybut jest przechowywany w oddzielnej kolumnie.
- Dla każdej tabeli dodajemy specjalny atrybut – klucz jednoznacznie identyfikujący „obiekt”.
  - Raczej nie wykorzystujemy atrybutów „biznesowych”.
- Specjalne rodzaje atrybutów
  - wymagane i opcjonalne,
  - pojedyncze i powtarzalne,
  - obiektu i klasowe,
  - wyliczalne.

Mapowanie klas (2)

- Specjalne rodzaje atrybutów
  - Prosty

    Kolumna w tabeli, np. *Nazwisko*.
  - Złożony
    - W tej samej tabeli biznesowej, jako wiele atrybutów (kolumn) powstałych z „rozgrupowania” atrybutu złożonego;
    - Jako jeden „spłaszczony atrybut”, np. adres: *„ul. Marszałkowska 12, 03-333 Warszawa, Polska”*;

Mapowanie klas (3)

- Specjalne rodzaje atrybutów
  - Złożony – c. d.
    - Jako jeden atrybut, ale z użyciem specjalnej notacji np. XML

```
<Adres>
        <Ulica>Marszałkowska</Ulica>
        <NrDomu>12</NrDomu>
        <KodPocztowy>03-333</KodPocztowy>
        <Kraj>Polska</Kraj>
</Adres>
```

    - Stworzenie oddzielnej tabeli *Adres* i połączenie jej z „główną”

IdAdres	Ulica	Nr Domu	Kod Pocztowy	Kraj
328	Marszałkowska	12	03-333	Polska

  - Opcjonalny

    Kolumna musi dopuszczać wartości `null`. Uwaga na problemy, np. z zapytaniami.

Mapowanie klas (4)

- Specjalne rodzaje atrybutów
  - Powtarzalny
    - Jako jedna kolumna ze specjalną składnią. Poszczególne wartości oddzielone np. przecinkami lub wykorzystanie XML – podobnie jak przy atrybucie złożonym.
    - Stworzenie oddzielnej tabeli i połączenie jej z „główną”.
  - Klasowy
    - Utworzenie specjalnej tabeli zawierającej wartości atrybutów klasowych z różnych klas.
    - Zapamiętanie go poza bazą danych, np. w kodzie programu.

Mapowanie klas (5)

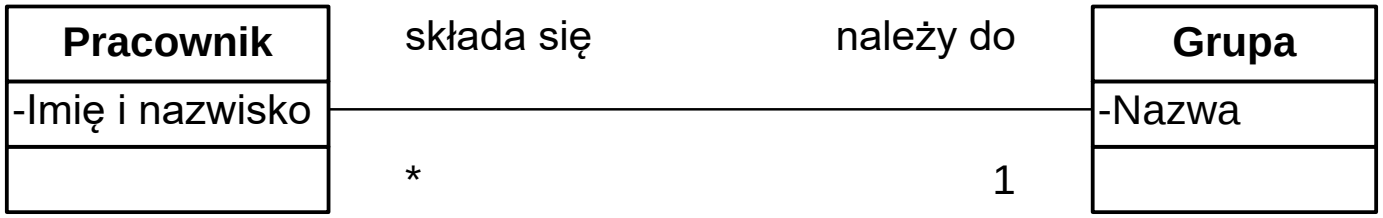
- Specjalne rodzaje atrybutów
  - Pochodny (wyliczalny)

    Utworzenie dedykowanej:

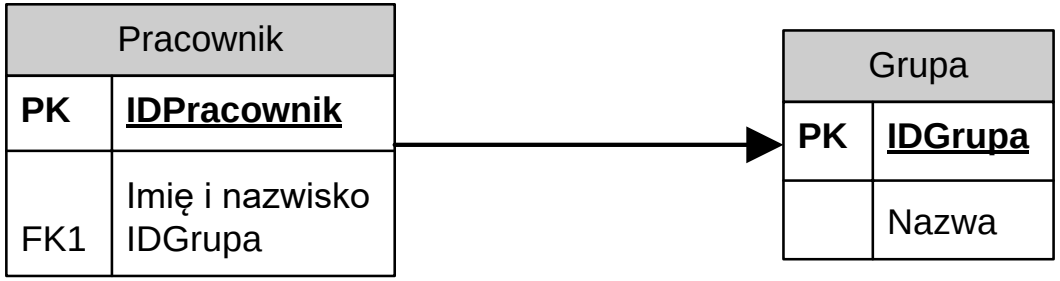
    - Perspektywy (jeżeli system zarządzania bazą danych na to pozwala),
    - Metody w języku programowania bazy danych, np. w SQL,
    - Metody w języku programowania (poza bazą danych).

Mapowanie asocjacji

- Asocjacje binarne
  - Liczność 1 – 1, 1 - *



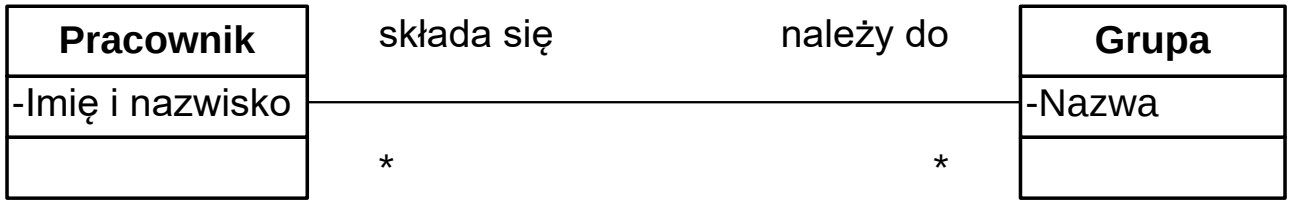
  - Zastępujemy je relacjami.
  - Dodajemy klucz obcy do odpowiedniej klasy.



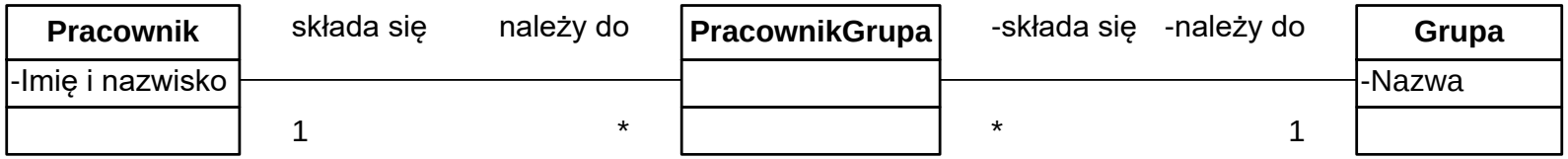
  - Aby odnaleźć pracowników należących do określonej grupy, trzeba przejrzeć wszystkich.

Mapowanie asocjacji (2)

- Asocjacje binarne
  - Liczność * - *

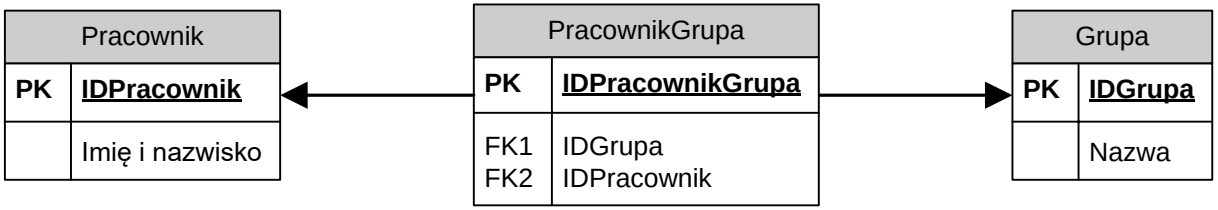


- Wprowadzamy klasę pośredniczącą (nowe nazwy ról?).
- Dzięki temu zamiast jednej asocjacji „* - *” mamy dwie asocjacje „1 - *”.



Mapowanie asocjacji (3)

- Asocjacje binarne
  - Liczność * - * - c. d.
    - Trzy klasy mapujemy na trzy tabele.
    - W „środkowej” tabeli umieszczamy klucze obce.

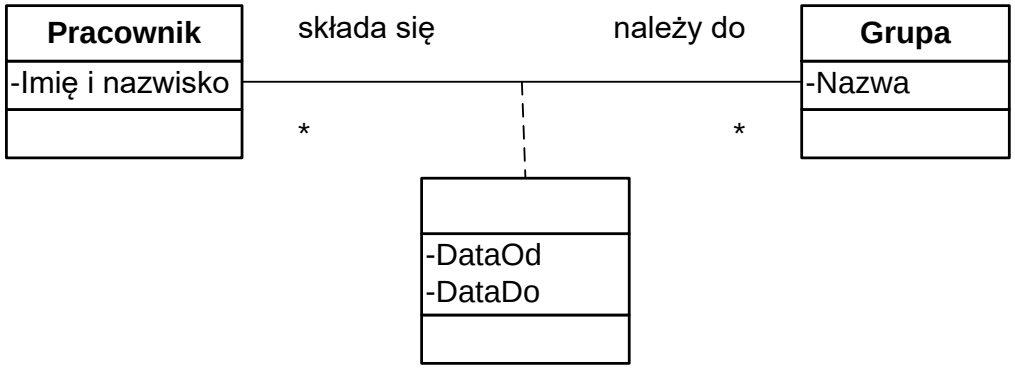


    - Niestety aby znaleźć
      - Pracowników należących do danej grupy,
      - Grupy do których należy konkretny pracownik

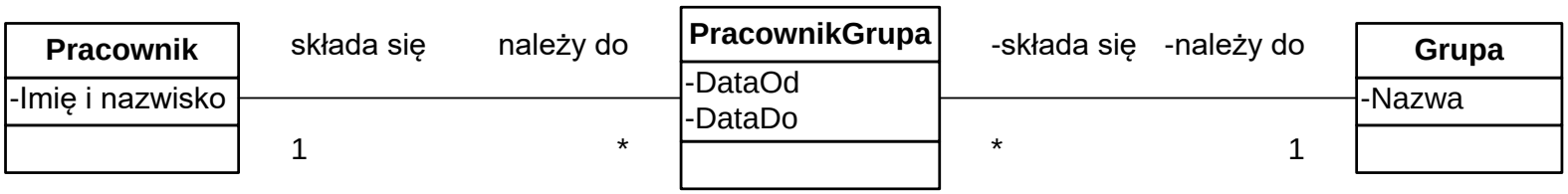
      musimy przejrzeć wszystkie krotki z tabeli *PracownikGrupa*.

Mapowanie asocjacji (4)

- Asocjacje z atrybutem (klasą asocjacji)

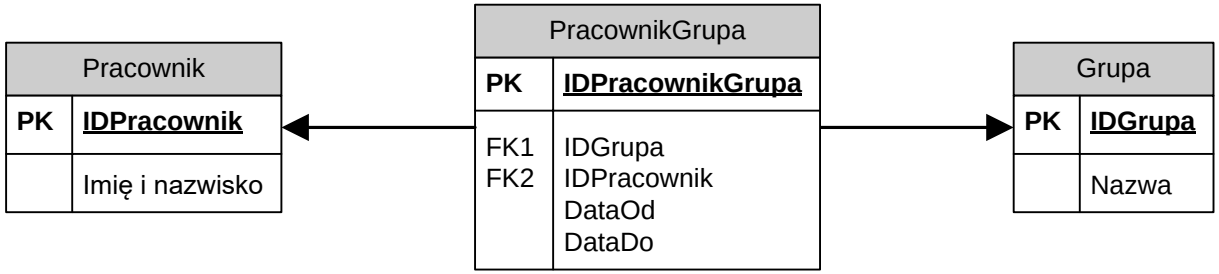


- Postępujemy analogicznie jak w przypadku asocjacji „* - *”
  - Klasa pośrednicząca z atrybutami pochodzącymi z klasy asocjacji.



Mapowanie asocjacji (5)

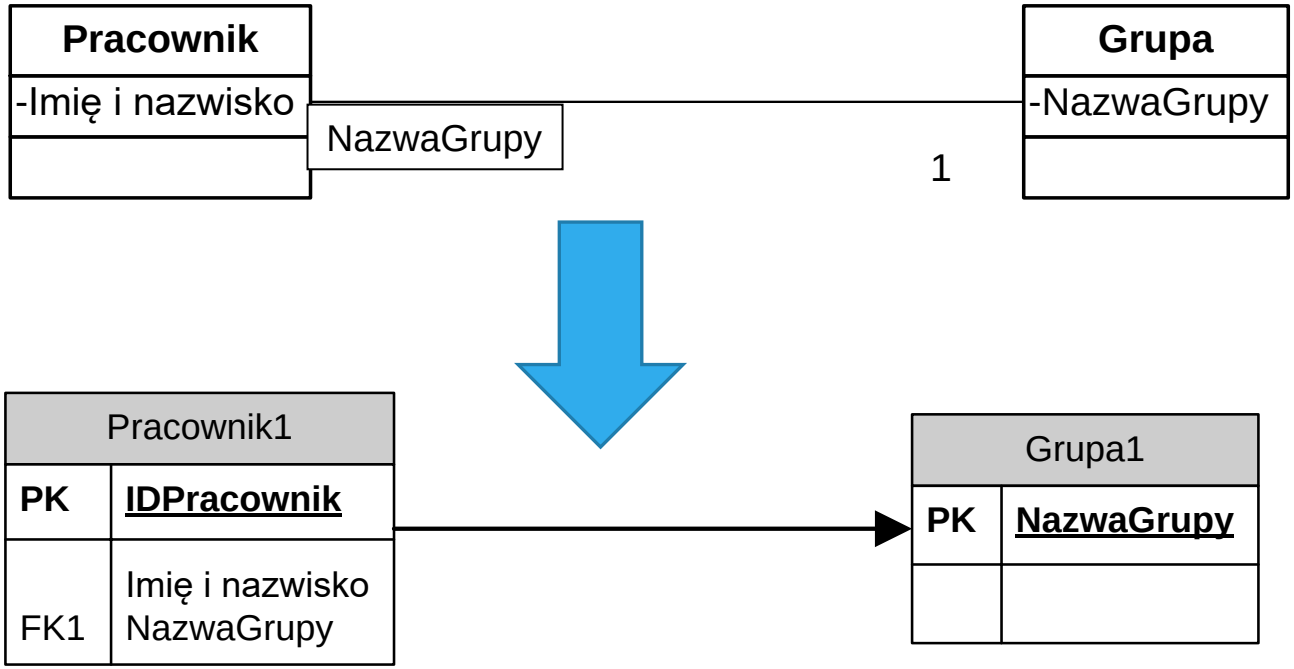
- Asocjacje z atrybutem (klasą asocjacji) – c. d.
  - Liczność * - * - c. d.
    - Trzy klasy mapujemy na trzy tabele.
    - W „środkowej” tabeli umieszczamy klucze obce oraz atrybuty pochodzące z klasy pośredniczącej (klasy asocjacji).



    - Wady podobne jak dla asocjacji „* - *”.

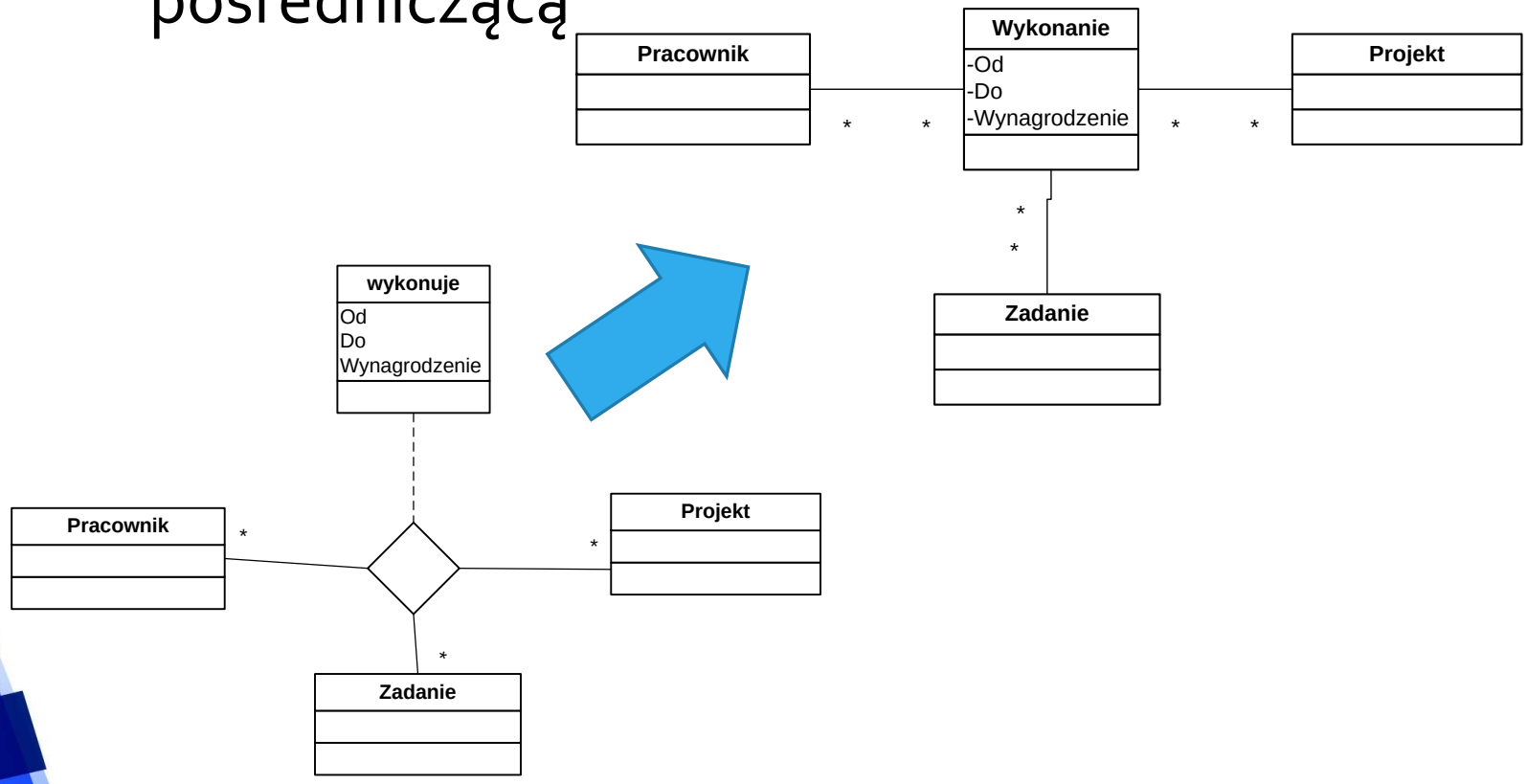
Mapowanie asocjacji (6)

- Asocjacje kwalifikowane
  - Brak odpowiednika w modelu relacyjnym.
  - Częściowo można je „symulować” stosując jako klucz obcy kwalifikator.



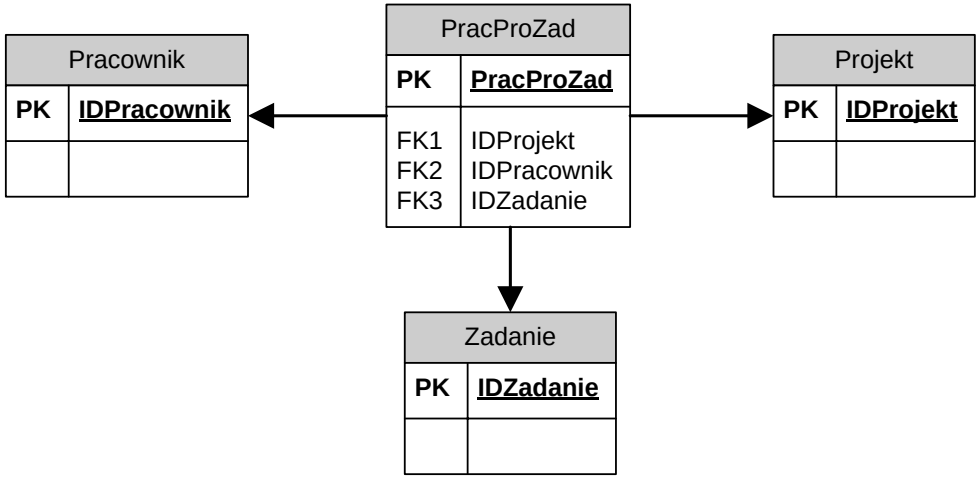
Mapowanie asocjacji (7)

- Asocjacje n-arne
  - Wprowadzamy dodatkową klasę pośredniczącą



Mapowanie asocjacji (8)

- Asocjacje n-arne – c. d.
  - Cztery klasy (n = 3) mapujemy na cztery tabele.
    - W „środkowej” tabeli umieszczamy klucze obce oraz ewentualnie atrybuty pochodzące z klasy pośredniczącej (klasy asocjacji). Nie uwzględniono tabel pośredniczących.



    - Wady podobne jak dla asocjacji „* - *”.

Mapowanie asocjacji (9)

- Agregacje
  - Analogicznie jak w przypadku przejścia z modelu pojęciowego na implementacyjny, agregacje realizujemy tak samo jak asocjacje.
- Kompozycje
  - Jeżeli system zarządzania bazą danych umożliwia to:
    - Tworzymy odpowiednią perspektywę (*db view*),
    - Stosujemy specyficzne *więzy integralności* (np. kaskadowe usuwanie) i/lub wyzwalacze (*trigger*'y).

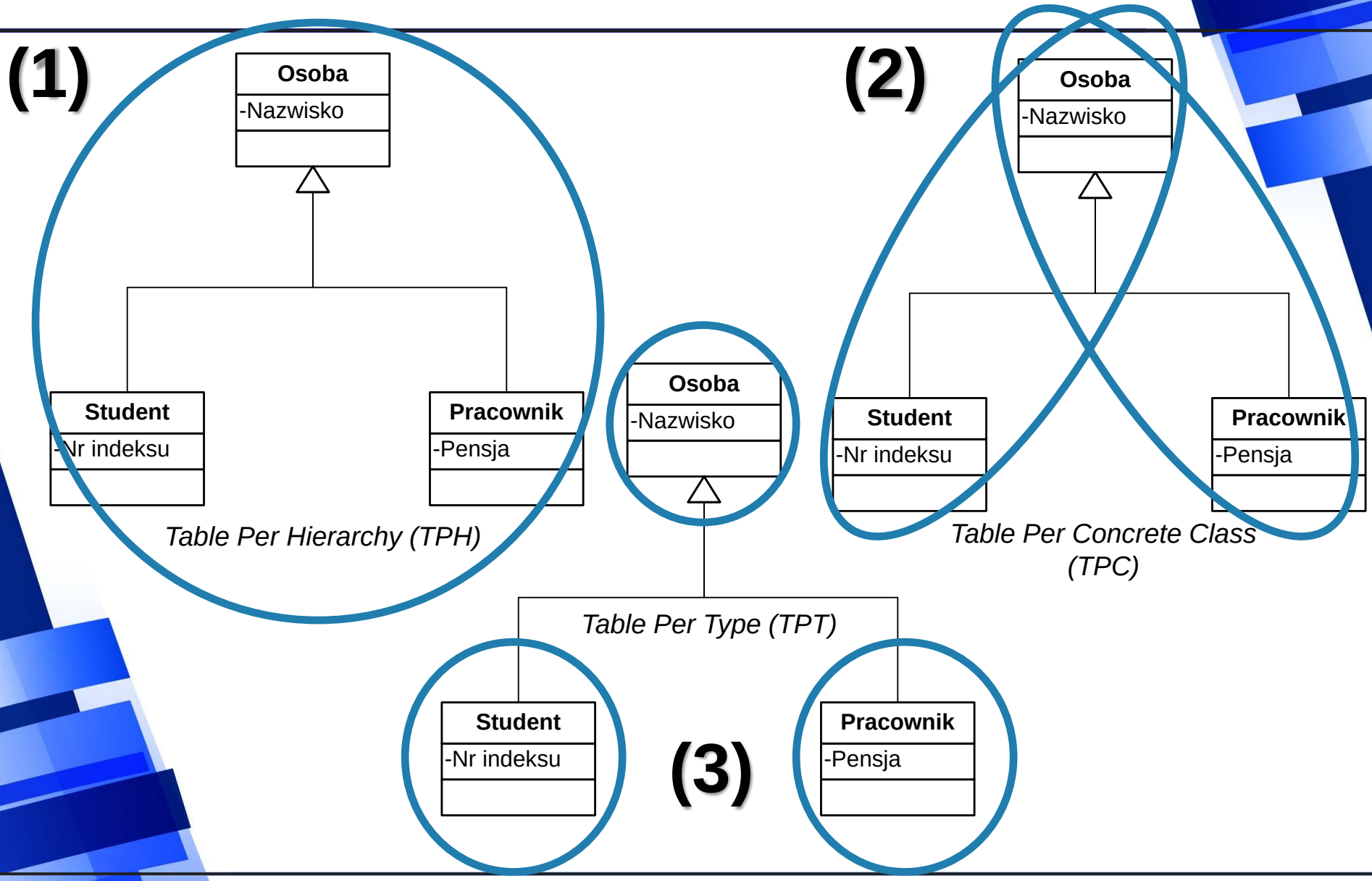
Mapowanie dziedziczenia

- W tradycyjnych relacyjnych bazach danych dziedziczenie nie występuje.
- W nowszych systemach, dziedziczenie występuje (przeważnie najprostsze: rozłączne, pojedyncze).
- Oczywiście, w przypadku jego braku, można próbować różnych konstrukcji, które problem „obchodzą”.

Mapowanie dziedziczenia (2)

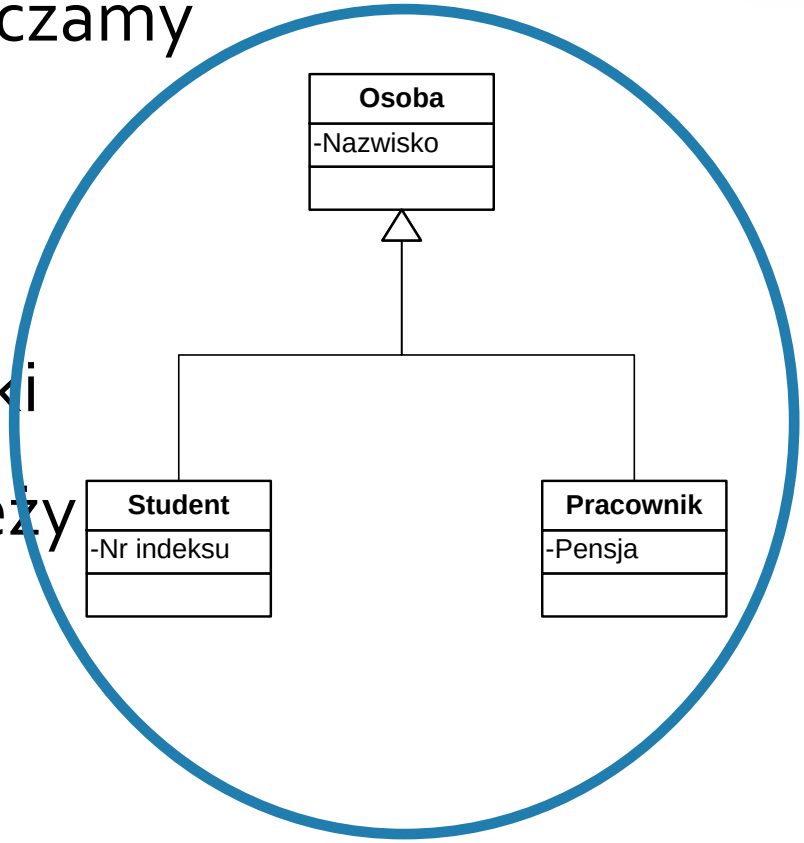
- Można powiedzieć, że sposoby obejścia braku dziedziczenia w RBD są zbliżone do tych wykorzystywanych przy przejściu z modelu pojęciowego do implementacyjnego.
  - Wykorzystanie relacji (asocjacji) do opisania zależności pomiędzy tabelami (klasami),
  - Spłaszczenie hierarchii.

Mapowanie dziedziczenia (3)



Mapowanie dziedziczenia (4)

- Sposób nr 1 (TPH - *Table Per Hierarchy*)
  - Całą zawartość wszystkich klas hierarchii dziedziczenia umieszczamy w jednej tabeli.
  - Dodajemy kolumnę określającą typ krotki („do jakiej klasy należy obiekt”)
  - Wady i zalety.

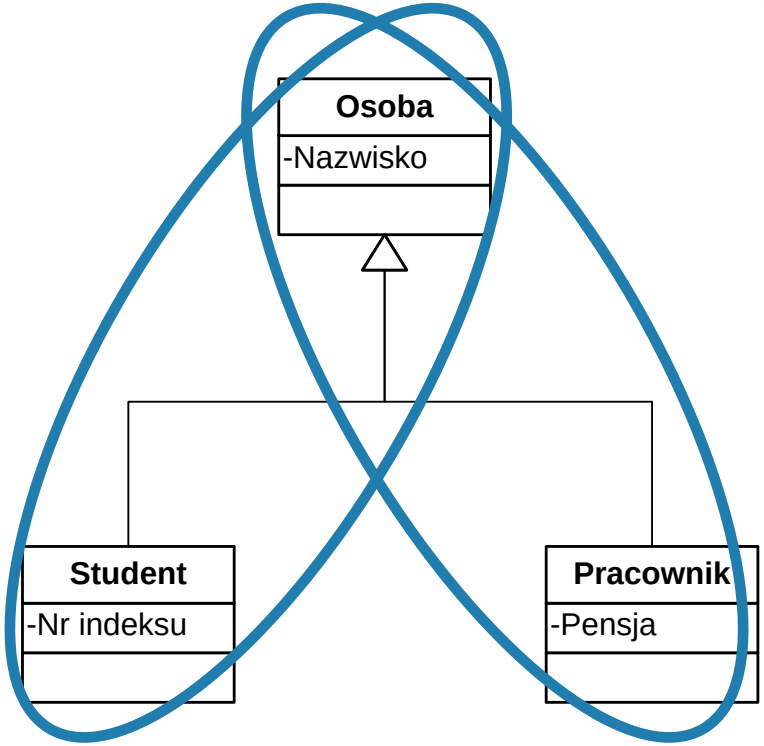


Mapowanie dziedziczenia (5)

- Sposób nr 2 (TPC - *Table Per Concrete Class*)
  - Tworzymy jedną tabelę dla każdej z podklas.
  - W każdej tabeli będą znajdowały się elementy z nadklasy.
  - Wady i zalety.
  - W celu połączenia wyników

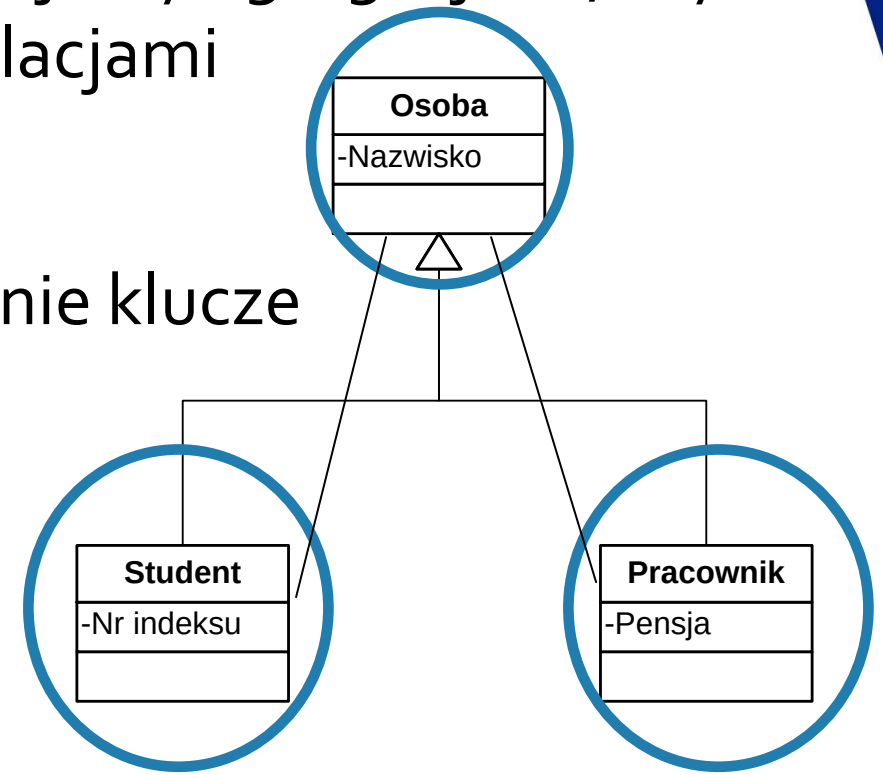
    można skorzystać

    z operatora `Union` (SQL).



Mapowanie dziedziczenia (6)

- Sposób nr 3 (TPT - *Table Per Type*)
  - Każda klasa ma swoją własną tabelę.
  - Dziedziczenie zastępujemy agregacjami, czyli w przypadku RBD, relacjami pomiędzy tabelami.
  - Dodajemy odpowiednie klucze główne i obce.
  - Wady i zalety.



Relacyjne bazy danych w obiektowych językach programowania

- Każdy z popularnych języków programowania udostępnia biblioteki do obsługi RBD.
- Java
  - JDBC
  - Natywne rozwiązania dla konkretnych baz,
- Microsoft C#
  - ADO,
- C++
  - ADO (MS),
  - ODBC.

RBD w języku Java

- Najpopularniejszym sposobem łączenia się z bazą danych w języku Java jest wykorzystanie funkcjonalności udostępnianej przez JDBC.

```
// Próba wczytania driver'a
try
{
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver").newInstance();
}
catch (ClassNotFoundException cnfe)
{
        // Błąd
        System.err.println ("Blad: " + cnfe);
        System.exit(0);
}

String url = "jdbc:odbc:" + args[0];

Connection db_connection = DriverManager.getConnection (url, "dba", "sql");

Statement db_statement = db_connection.createStatement();

db_statement.executeUpdate("create table employee { int id, char(50) name };");
db_statement.executeUpdate("insert into employee values (1, 'Jan Kowlski');");
db_connection.commit();
```

http://www.javacoffeebreak.com/articles/jdbc/

RBD w języku Java (2)

- JDBC – c. d.

```
// [...]

// Wykonaj zapytanie
ResultSet result = db_statement.executeQuery("select * from employee");

// Przetwarzanie wynikow
while (result.next() )
{
        System.out.println ("ID : " + result.getInt("ID"));

        System.out.println ("Name : " + result.getString("Name"));
        System.out.println ();
}
```

http://www.javacoffeebreak.com/articles/jdbc/

- Jak widać nie jest to najłatwiejszy sposób pracy z danymi.
- Szczególnie gdy porównamy go z podejściem obiektowym, np. *ObjectPlus*.

- Ciąg dalszy na następnym wykładzie…