

Obiektowe bazy danych

Obiektowy model danych

Wykład prowadzi:

Tomasz Koszlajda

Plan wykładu

- Przestanki dla nowej generacji systemów baz danych
- Podstawowe elementy obiektowego modelu danych
- Konstruktory złożonych typów danych
- Abstrakcyjne typy danych
- Dziedziczenie
- Związki między danymi
- Hierarchie kolekcji obiektów
- Polimorfizm i późne wiązanie
- Tożsamość danych
- Trwałość danych

Nowe dziedziny zastosowań baz danych

- Systemy wspomagania projektowania CAD/CAM
- Systemy informacji przestrzennej
- Multimedialne bazy danych

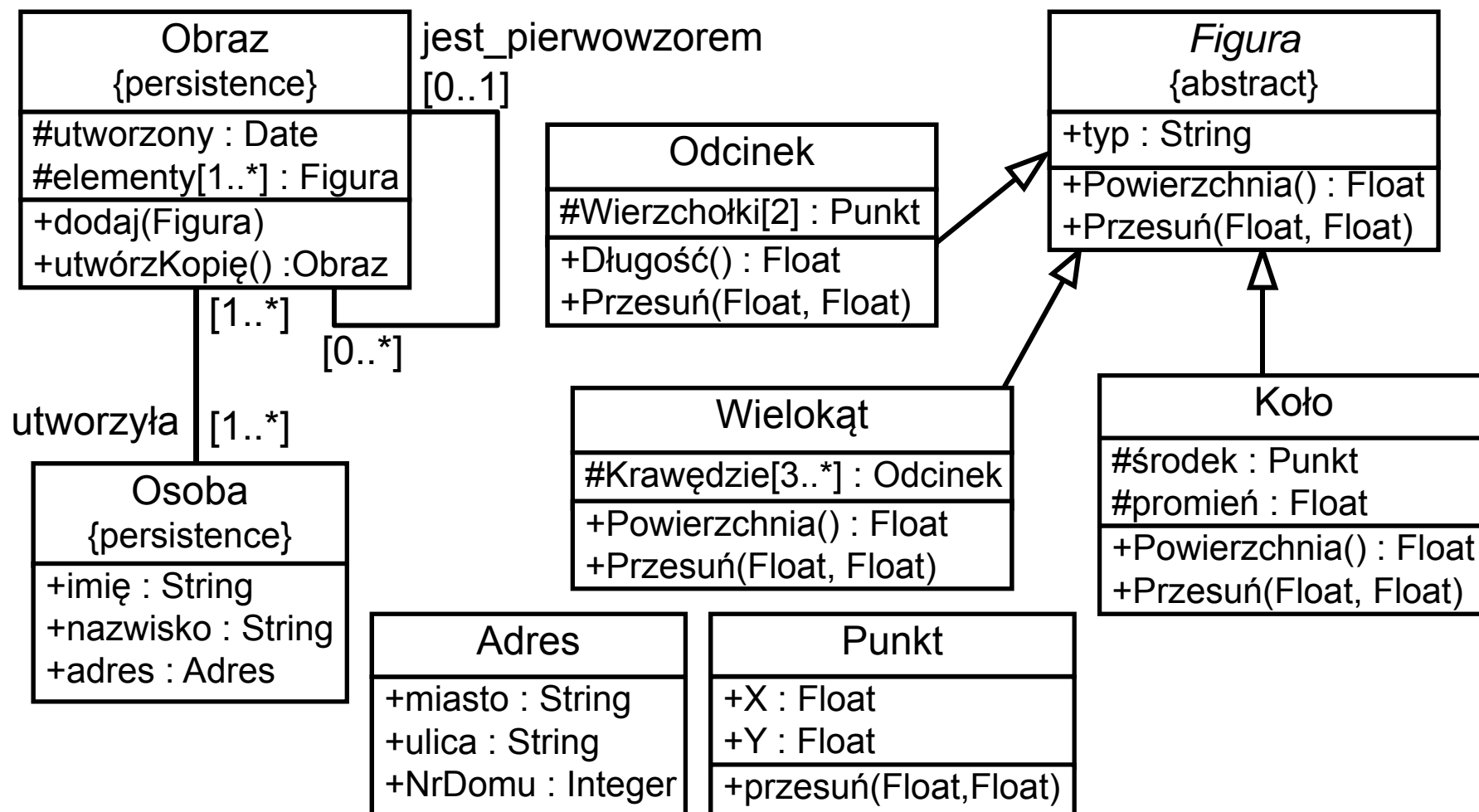
Charakterystyka nowych dziedzin zastosowań

- Złożone, hierarchiczne struktury danych
- Złożone behawioralne własności danych
- Nowe modele przetwarzania: ad hoc – dynamiczna struktura transakcji, nawigacja vs. selekcja, kooperujące i intensywnie przetwarzające dane transakcje

Nowe technologie budowy aplikacji

- Języki obiektowe

Przykład złożonej rzeczywistości



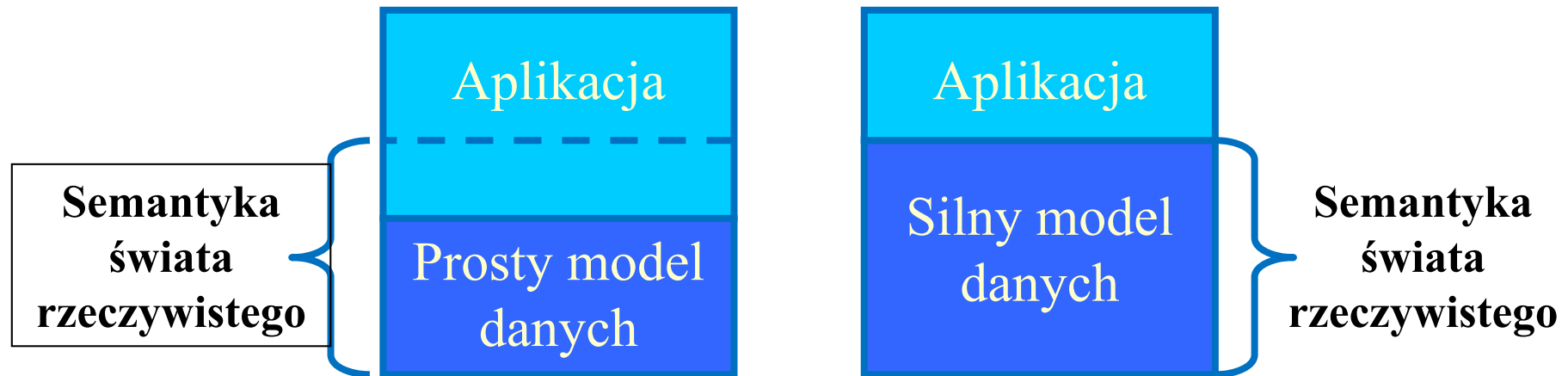
Ograniczenia relacyjnego modelu danych

- Płaskie jednowymiarowe struktury danych
- Identyfikacja danych jedynie za pomocą ich wartości
- Zamknięty zbiór predefiniowanych typów danych
- Brak pojęcia związków między danymi
- Brak hierarchii

```
Figury(id_f PK, typ, powierzchnia)
Odcinki(id_odc PK FK(Figury), typ, x1, y1, x2, y2)
Wielokąty(id_w PK FK(Figury), typ)
Krawędzie(id_k PK, x1, y1, x2, y2, id_w FK(Wielokąty))
Koła(id_k PK FK(Figury), typ, x, y, promień)
Obrazy(id_ob PK, utworzony)
Osoby(id_os PK, imię, nazwisko, miasto, ulica, numer_domu)
Autorstwo(id_ob FK(Obrazy), id_os FK(Osoby))
Modyfikacje(wzorzec
```

Przesłanki nowej generacji modelu danych

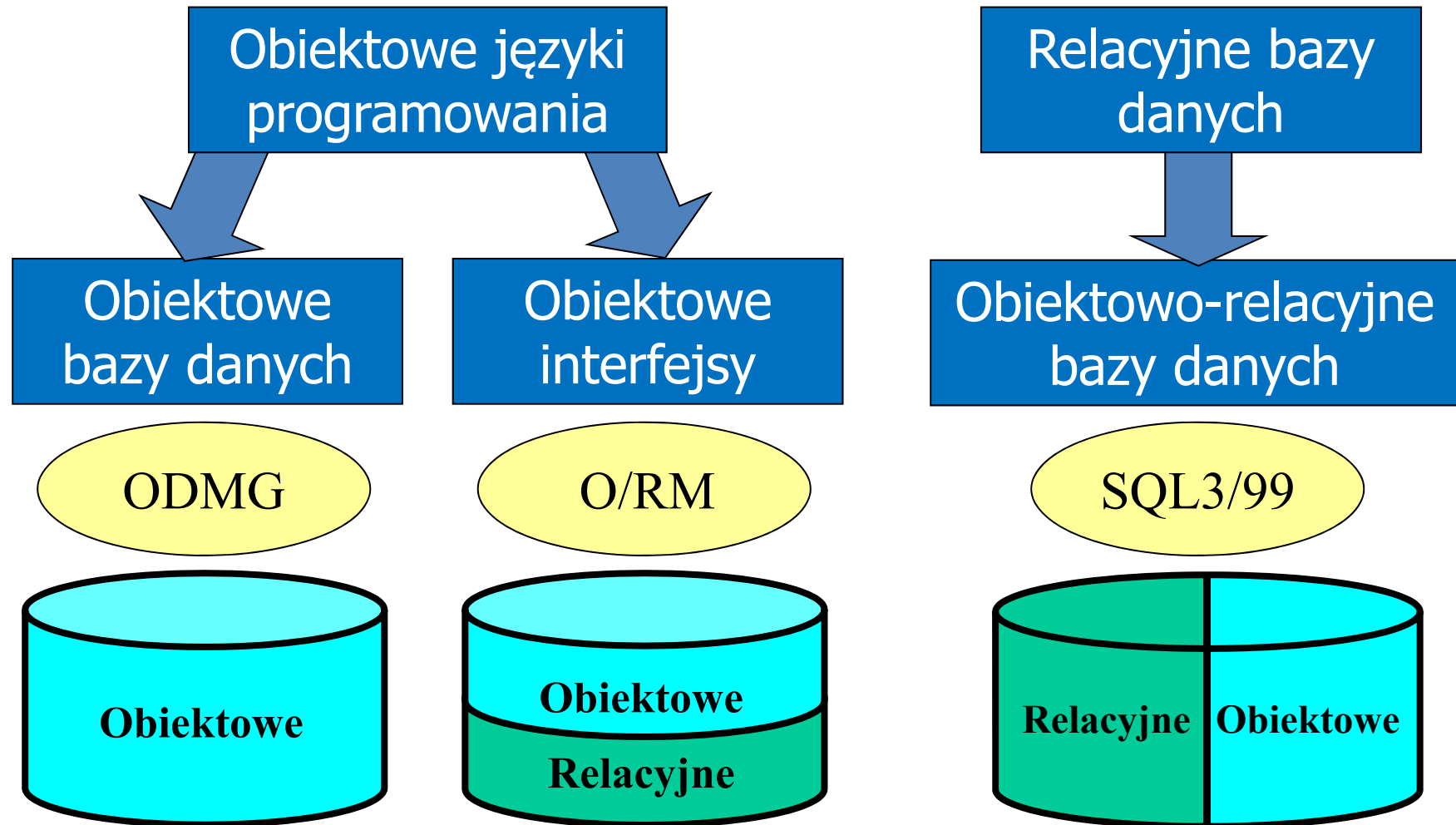
- Potrzeba bogatszego modelu danych
- Rozszerzalny model danych umożliwiający ścisłe dopasowanie dla dowolnych dziedzin zastosowań
- Ścisłejsza integracja z obiektowymi aplikacjami bazy danych



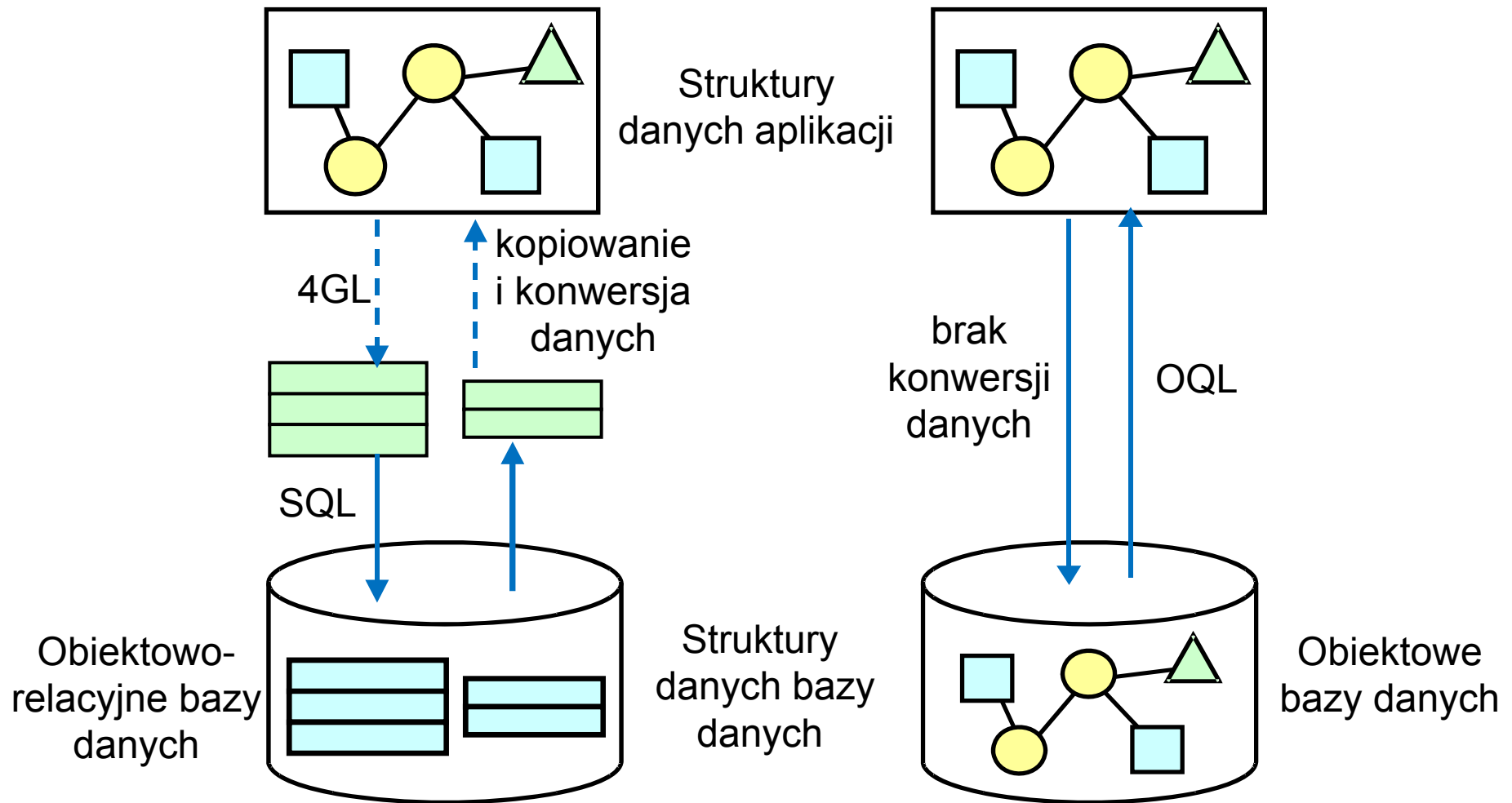
Podstawowe elementy obiektowego modelu danych

- Obiekt: stan i funkcjonalność
- Cechy obiektów: atrybuty i związki
- Funkcjonalność obiektu: metody
- Tożsamość obiektu
- Hermetyczność obiektów
- Klasa: typ danych i moduł programowy
- Dziedziczenie: współdzielenie implementacji i relacja podtypu
- Przeciążanie i dynamiczne wiązanie funkcjonalności obiektów

Drogi rozwoju obiektowych baz danych



Architektury obiektowych i obiektowo-relacyjnych systemów baz danych



Baza danych jako zbiór kolekcji obiektów

- Stan obiektowej bazy danych jest zbiorem kolekcji trwałych i rozróżnialnych obiektów.
- Schemat obiektowej bazy danych jest zbiorem klas, które definiują strukturę i funkcjonalność obiektów.
- W obiektowych bazach danych rozróżnia się pojęcie klasy jako definicji własności obiektów od pojęcia rozszerzenia klasy będącego zbiorem trwałych obiektów.

```
class Figura { // nazwa klasy jako typu  
    (extent Figury) // nazwa rozszerzenia klasy jako  
                    // zbioru wystąpień
```

Abstrakcyjne typy danych

Możliwość definiowania nowych typów danych o dowolnej złożoności i funkcjonalności. Typy danych użytkownika mogą być podstawą definicji pojedynczych atrybutów klas jak również całych klas.

```
class Punkt {  
    attribute Float X, Y;  
    void przesun (in Float x,  
                  in Float y);};
```

Punkt
+X : Float
+Y : Float
+przesun(Float,Float)

```
class Odcinek {  
    attribute Punkt W1, W2;  
    void przesun (in Float a,  
                  in Float b) {  
        W1.przesun(a, b);  
        W2.przesun(a, b); }};
```

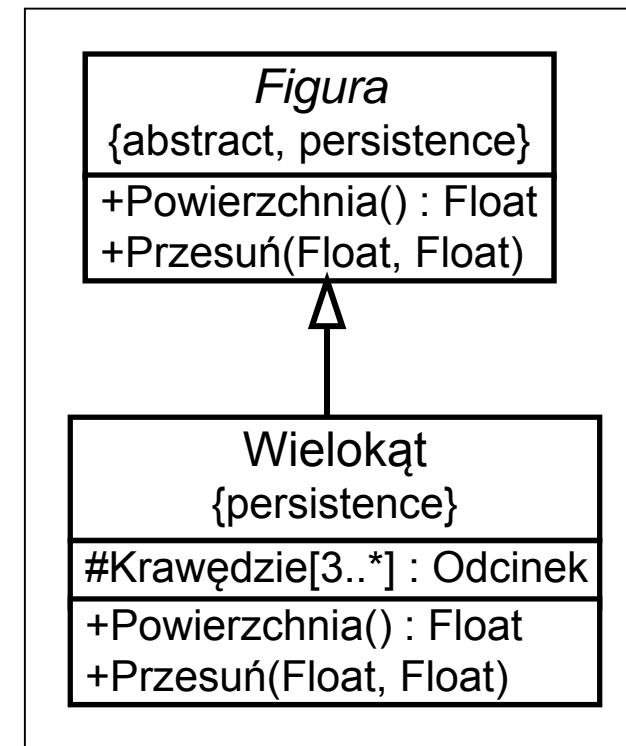
Odcinek {persistence}
#Wierzchołki[2]:Punkt
+przesun(Float,Float)

Dziedziczenie

Klasy mogą być specjalizowane przez mechanizm dziedziczenia. Klasa pochodna dziedziczy funkcjonalność i implementację klasy bazowej. W klasie pochodnej można dodać nową funkcjonalność lub redefiniować funkcjonalność odziedziczoną.

```
class Wielokąt extends Figura
// dziedziczenie
{...};
class Wielokąt : Figura
// relacja podtypu
{...};
```

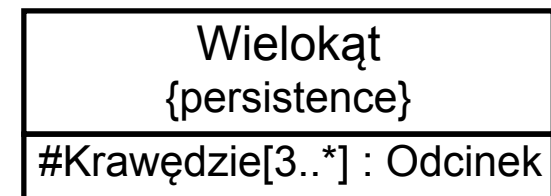
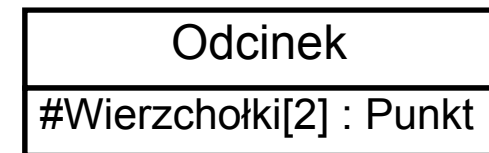
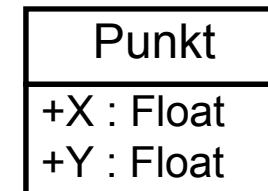
Relacja podtypu \supseteq Dziedziczenie



Złożone struktury danych

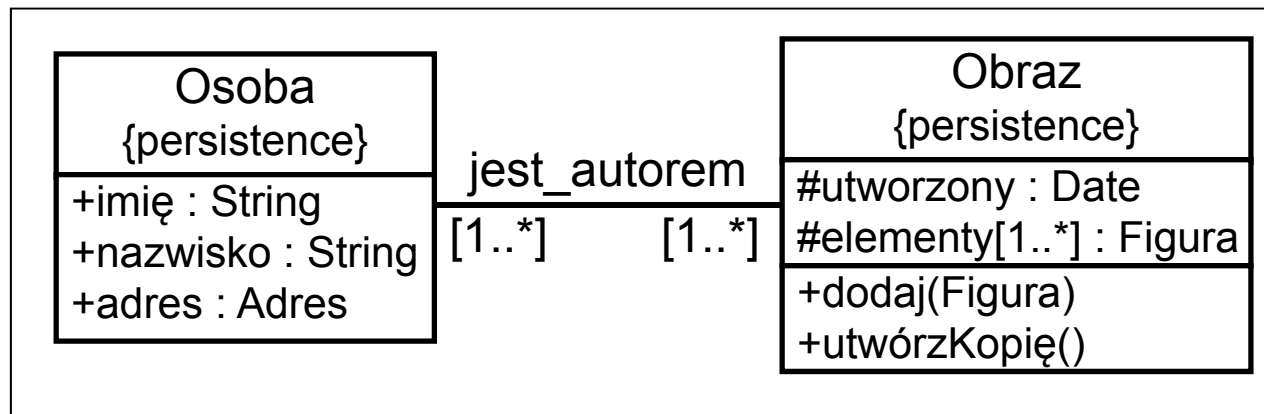
Możliwość naturalnego modelowania atrybutów **złożonych** i **wielowartościowych**. Przykład zastosowania konstruktorów typów złożonych w języku ODL.

```
class Wielokąt {  
    struct Punkt { //typ  
        Float X, Y; }  
    struct Odcinek { //typ  
        Punkt Wierzchołek_1;  
        Punkt Wierzchołek_2; }  
    //atrybut  
    attribute set<Odcinek> krawędzie; };
```



Związki między danymi

Możliwość definiowania i składowania w bazie danych związków między danymi.



```
class Osoba {  
    relationship set<Obraz> jest_autorem  
    inverse Obraz::jest_utworzony_przez;  
    ...  
}
```

nazwa
związku

typ
związku

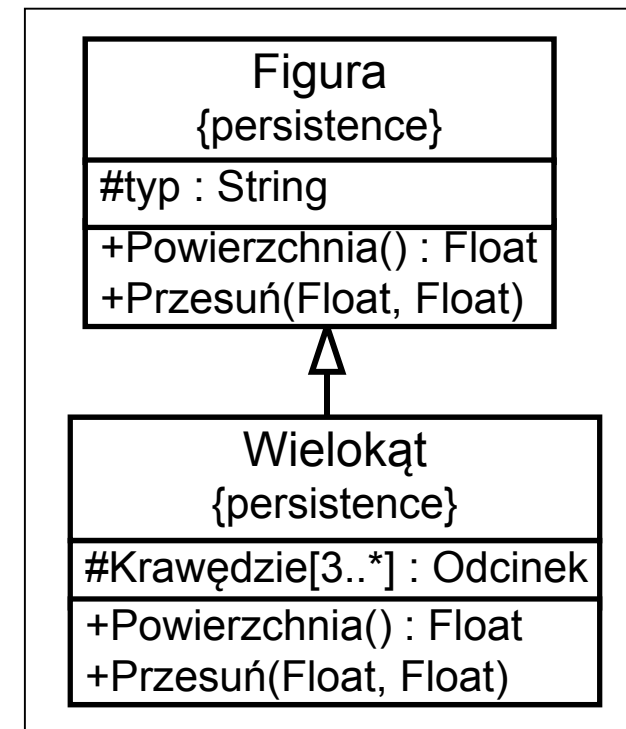
związek
odwrotny

Hierarchia kolekcji obiektów

Związek dziedziczenia między klasami, których rozszerzenia są składowane w bazie danych implementuje związek zawierania się podzbiorów obiektów. Rozszerzenie klasy pochodnej jest podzbiorem rozszerzenia klasy bazowej.

```
class Figura {  
    (extent Figury)  
    ...};  
class Wielokąt extends Figura{  
    (extent Wielokąty)  
    ...};
```

$\text{Wielokąty} \subseteq \text{Figury}$



Polimorfizm i późne wiązanie

Relacja podtypu łącząca klasy i interfejsy umożliwia podstawienia polimorficzne polegające na podstawieniu pod zmienną typu klasy bazowej obiektu, który jest wystąpieniem klasy pochodnej.

Podstawienia polimorficzne umożliwiają dynamiczne wiązanie nazw metod.

zmienna polimorficzna

podstawienie polimorficzne

Figura f = new Koło(10,6,5);

Float p = f.powierzchnia(); ← Koło::powierzchnia()

f = new Wielokąt(p1,p2,p3); ← wiązanie dynamiczne

p = f.powierzchnia(); ← Wielokąt::powierzchnia()

Tożsamość danych

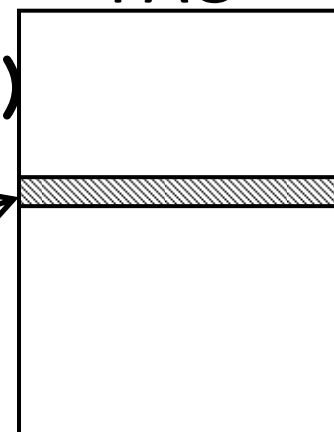
Obiektowe języki programowania: identyfikacja danych przez symboliczną nazwę i fizyczny adres w pamięci operacyjnej.

```
class Pracownik {...};
```

```
Pracownik Nowak ("Jan", "Nowak")
```

0c78:89ad

PAO

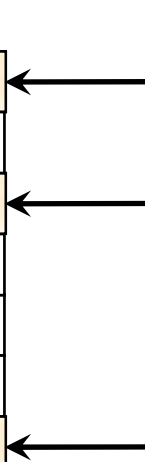


```
Nowak.nazwisko = "Kowalski";
```

Tożsamość danych

Model relacyjny: identyfikacja danych przez ich wartości

```
SELECT * FROM płatnicy  
WHERE Nazwisko = 'NOWAK';
```

JAN	NOWAK	
TADEUSZ	KOWALSKI	
MACIEJ	NOWAK	
JAN	RZEPA	
KUBA	TARZAN	
JÓZEF	MALINIAK	
JAN	NOWAK	

Jednoznaczna identyfikacja danych wymaga zdefiniowania dla każdej relacji – klucza podstawowego

Tożsamość danych w obiektowej bazie danych

- Identyfikacja na podstawie wartości identyfikatora obiektu nazywanego OID
- Identyfikacja przez OID jest niezależna od wartości atrybutów obiektu

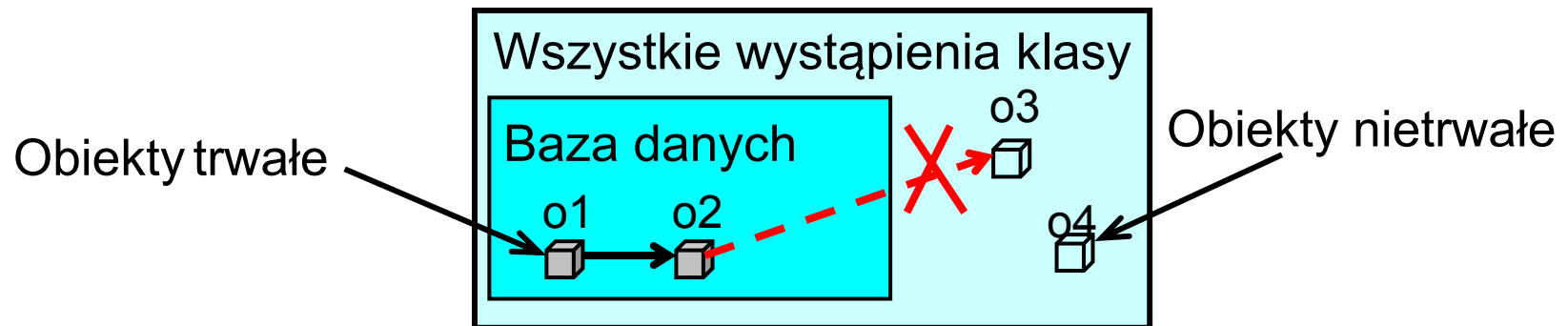
```
Osoba *kowalski, *kowalska;  
kowalski = new Osoba("Jan", "Kowalski");  
kowalska = new Osoba("Janina", "Kowalska");  
Kowalska->małżonek = kowalski; //podstawienie oid  
Kowalski->nazwisko = "Nowak";  
Kowalski->pesel = 68040102766;  
Kowalski->płeć = "K";  
Kowalska->małżonek->dochody->pokaż();
```

Trwałość danych

- Obiekty mogą być tworzone w trwałym lub ulotnym obszarze składowania
- Trwałość powinna być własnością poszczególnych obiektów, a nie klas obiektów
- Trwałość powinna być własnością obiektów dowolnej klasy
- Sposób operowania na obiektach trwałych i ulotnych powinien być taki sam
- Obiekty w ciągu cyklu życia mogą być przenoszone między trwałym i nietrwałym obszarem składowania

Trwałość danych

Obiekty stają się trwałe przez jawne utrwalenie lub przez bycie osiągalnymi przez inne obiekty trwałe.



```
pm = pmf.getPersistenceManager  
Punkt p = new Punkt(10.5, 7.1); // obiekt nietrwały  
Koło k = new Koło(p, 4); // obiekt nietrwały  
transaction = pm.currentTransaction();  
pm.makePersistent(k); // utrwalenie obiektów p i k w bazie danych  
transaction.commit();
```