

Obiektowo-relacyjne bazy danych

**Wykład prowadzi:
Tomasz Koszlajda**

Obiektowo-relacyjne bazy danych

Ewolucja rozszerzeń relacyjnego modelu danych:

- Składowanie kodu procedur w bazie danych
- Możliwość definiowania nowych typów danych
- Konstruktory złożonych typów danych
- Dziedziczenie typów danych
- Możliwość definiowania obiektowych typów danych
- Referencyjny typ danych
- Hierarchie zbiorów danych

Standard: SQL3/SQL99

Produkty komercyjne: Oracle, DB2

Definiowanie nowych typów danych

```
create type Punkt as object(  
    x Float,  
    y Float,  
    member procedure przesun(px Float, py Float));
```

Definiowanie
nowego typu

```
create type body Punkt as  
    member procedure przesun(px Float, py Float) is  
        begin  
            self.x := self.x + px;  
            self.y := self.y + py;  
        end przesun;  
end;
```

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

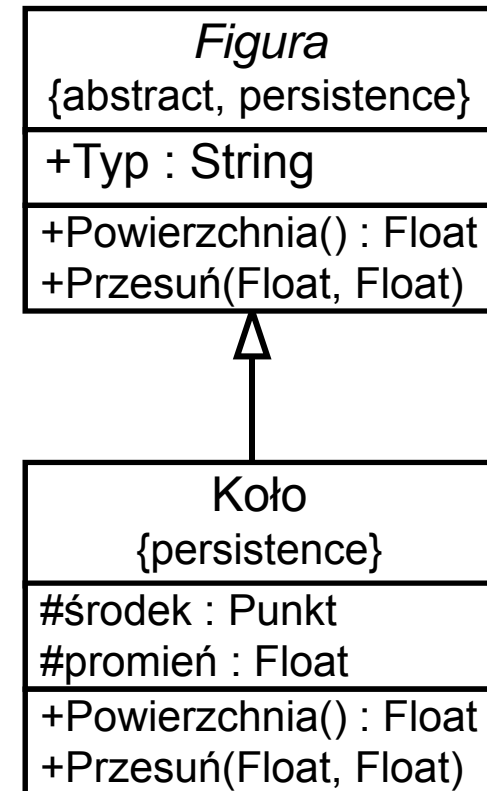
```
create type koło as object(  
    srodek Punkt,  
    Promien Float);
```

Stosowanie
nowego typu

Dziedziczenie

Nowe typy danych mogą być definiowane na bazie wcześniej zdefiniowanych typów danych.

```
create type figura as object (  
    typ varchar(10),  
    not instantiable member procedure  
        przesun(px Float, py Float),  
    -- definicja klasy abstrakcyjnej  
    not instantiable member function  
        powierzchnia return Float)  
not instantiable not final;  
  
create type koło under figura (  
    -- klasa koło dziedziczy po klasie figura  
    srodek Punkt,  
    promien Float,  
    overriding member procedure  
        przesun(px Float, py Float),  
    -- implementacja metody abstrakcyjnej  
    overriding member function  
        powierzchnia return Float);
```



Zbiory (relacje, tabele) obiektów

W obiektowo-relacyjnych bazach danych relacje mogą przechowywać krotki albo obiekty.

```
create type Koło as object(  
    środek Punkt,  
    promień Float,  
    member procedure  
        przesun(px Float, py Float),  
    member function  
        powierzchnia return Float);
```

Koło {persistence}
#środek : Punkt
#promień : Float
+Powierzchnia() : Float
+Przesun (Float, Float)

```
create table Koła of Koło;  
-- relacja obiektów  
insert into Koła values  
    (new Koło(new Punkt(10.5, 7.2), 10.0));  
-- konstruktor klasy Punkt
```

Tożsamość danych

Obiekty przechowywane w tabelach mają przypisany systemowy identyfikator OID gwarantujący rozróżnialność obiektów niezależnie od przechowywanych w nich wartości.

```
create table Koła of Koło;  
insert into Koła values  
    (new Koło(new Punkt(10.5, 7.2), 10.0));
```

```
select ref(k), value(k), k.promień  
from Koła k;
```

identyfikator
obektu

wartość
obektu

obekt

Konstruktory złożonych typów danych - atrybuty wielowartościowe

Konstruktory typów umożliwiają modelowanie złożonych i wielowartościowych atrybutów danych.

```
create type Punkt as object(  
    x Number,  
    y Number);  
create type TablicaWierzchołków  
    as varray(2) of Punkt;  
-- VARRAY – posortowany zbiór danych
```

```
create type Odcinek as object(  
    Wierzchołki TablicaWierzchołków);  
create type ZbiórKrawędzi  
    as table of Odcinek; -- zbiór danych  
create type Wielokąt as object(  
    Krawędzie ZbiórKrawędzi);  
create table Wielokąty of Wielokąt  
    nested table Krawędzie store as K_table;
```

Punkt
+X : Float
+Y : Float

Odcinek
#Wierzchołki[2] : Punkt

Wielokąt
{persistence}
#Krawędzie[3..*] : Odcinek

Tworzenie obiektów z atrybutami wielowartościowymi

```
create type TablicaWierzchołków as varray(2) of Punkt; // tablica
create type Odcinek as object(Wierzchołki TablicaWierzchołków);
create type ZbiórKrawędzi as table of Odcinek; // zbiór
create type Wielokąt under figura (
    Krawędzie ZbiórKrawędzi,
    overriding member procedure przesun(px Float, py Float),
    overriding member function powierzchnia return Float);

create table Wielokąty of Wielokąt
    nested table Krawędzie store as kraw_table;

insert into Wielokąty(typ, Krawędzie)
values ('Trójkąt', ZbiórKrawędzi(
    new Odcinek(TablicaWierzchołków(new Punkt(0,0), new Punkt(2,2))),
    new Odcinek(TablicaWierzchołków(new Punkt(2,2), new Punkt(4,3))),
    new Odcinek(TablicaWierzchołków(new Punkt(4,3), new Punkt(0,0)))
));
```


Przetwarzanie atrybutów wielowartościowych

Dodawanie, usuwanie i modyfikowanie elementów atrybutów wielowartościowych:

```
insert into table (  
    select krawędzie from Wielokąty w  
    where ref(w) = :x)  
values (new Odcinek(new Punkt(0,0),  
    new Punkt(7,7));  
update table (  
    select krawędzie from Wielokąty w  
    where ref(w) = :x) k  
set k.Wierzchołki = new Punkt(1,1)  
where ref(k) = &y;
```

Warunki logiczne na atrybutach wielowartościowych

- cardinality
- is a set
- submultiset
- multiset except
- set
- is empty
- multiset intersect
- powermultiset
- member of
- multiset union

Konstruowanie wyrażeń logicznych odwołujących się do własności atrybutów wielowartościowych

```
select k.imię, k.nazwisko
from Kandydaci k inner join OfertyPracy o
on o.wymag_języki submultiset of k.językiObce
   or cardinality (k.języki_obce
      multiset intersect o.wymag_języki) > 3
```

Przetwarzanie zagnieżdżonych kolekcji

"Rozpłaszczanie" zagnieżdżonych kolekcji

```
select d.imię  
from Zespoły z, z.Pracownicy p, p.dzieci d
```

Typ wyniku, zagnieżdżone kolekcje:

```
set<set<set<varchar>>>
```

```
select d.imię  
from Zespoły z, Table(z.Pracownicy p),  
      Table(p,dzieci d)
```

Typ wyniku: **set<varchar>**

Heterogeniczne kolekcje danych

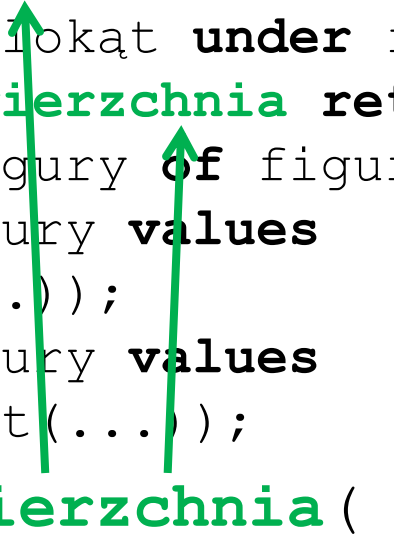
Ta sama tabela obiektów może przechowywać obiekty różnych typów danych o potencjalnie różnej strukturze, ale powiązanych związkiem dziedziczenia.

```
create type figura as object (typ Varchar(10));  
create type koło under figura (  
    środek Punkt,  
    promień Float);  
create type wielokąt under figura (...);  
create table Figury of figura;  
insert into Figury values  
    (new Koło(new Punkt(10.5, 7.2), 10.0));  
insert into Figury values(new Wielokąt(...));  
select value(f) from Figury f; -- wynikiem są koła i wielokąty  
select value(f) from Figury f  
    where value(f) is of (koło); -- wynikiem są koła
```

Dynamiczne wiązanie metod

Przetwarzanie heterogenicznych kolekcji obiektów wyzwala mechanizm dynamicznego wiązania

```
create type figura as object (  
    member function powierzchnia return Float);  
create type koło under figura (overriding member  
    function powierzchnia return Float);  
create type wielokąt under figura (overriding member  
    function powierzchnia return Float);  
create table Figury of figura;  
insert into Figury values  
    (new Koło(...));  
insert into Figury values  
    (new Wielokąt(...));  
select f.powierzchnia( ) from Figury f;
```

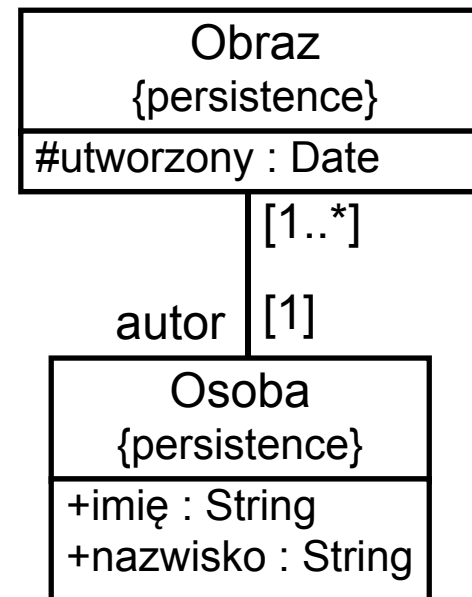


dynamiczne wiązanie
komunikatu "powierzchnia"

Referencyjny typ danych

Nowy systemowy typ danych REF służy do modelowania związków między obiektami.

```
create type Osoba as object (  
    imię varchar(10),  
    nazwisko varchar(20));  
create type Obraz as object (  
    utworzony Date,  
    autor REF Osoba  
-- atrybut autor służy do składowania OID);  
create table Obrazy of Obraz;  
create table Osoby of Osoba;  
insert into Osoby values ('Jan', 'Tarzan');  
insert into Obrazy  
    select '1-04-2006', ref(o)  
    from Osoby o where nazwisko = 'Tarzan';
```



Związki o krotności N

```
create type ZbiórAutorów as table of ref Osoba;  
create type Obraz as object(  
    utworzony Date,  
    autorzy ZbiórAutorów);  
create table Obrazy of Obraz  
    nested table autorzy store as aut;  
...  
insert into Obrazy  
    select '1-04-2006',  
        TypAutorzy(ref(o)) from Osoby o  
    where nazwisko = 'Tarzan';  
insert into Table (select autorzy  
                    from Obrazy)  
    (select ref(o) from Osoby o  
     where nazwisko = 'Nowak');
```

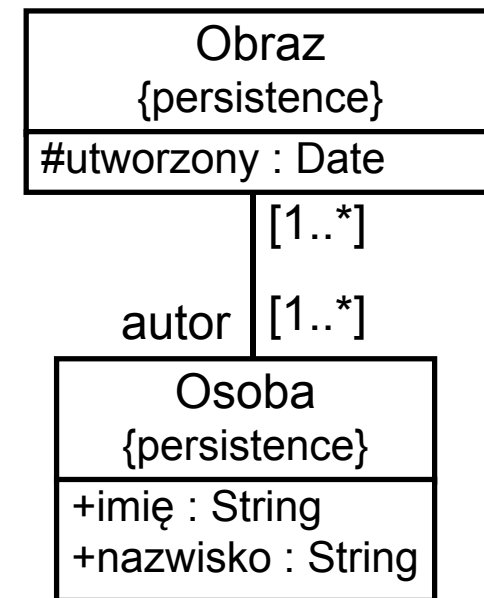


tabela
zagnieżdżona

związek
wielokrotny

nowe powiązanie

Przetwarzanie referencji

Operatory przetwarzające wartości przechowywane dla typów REF:

- Deref
- MAKE_REF
- REF
- REFTOHEX
- VALUE

```
create type Obraz as object (  
    utworzony Date,  
    autor REF Osoba);  
create table Obrazy of Obraz;  
select autor, deref(author) from obrazy;
```


Wyrażenia ścieżkowe

Możliwe jest tworzenie wieloczęłonowych wyrażeń ścieżkowych nawigujących wzdłuż powiązań między obiektami lub w głąb złożonych struktur obiektów.

ścieżka w głąb obiektu

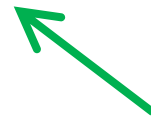
```
select o.utworzony, o.elementy.typ  
from Obrazy o
```



Połączenia strukturalne

Oprócz klasycznej operacji połączenia bazującej na warunkach połączeniowych, dostępny jest nowy typ ***połączenia strukturalnego***, bazujący na referencyjnych powiązaniach obiektów.

```
select o.utworzony, a.nazwisko, o.elementy.typ  
from Obrazy o, table(o.autorzy) a
```



nawigacja wzdłuż referencji

Przykłady zastosowań obiektowo-relacyjnego modelu danych

- Uniwersalne aplikacje baz danych – Oracle Financial, SAP
- Zastosowania w specyficznych dziedzinach
- Rozszerzenia systemowe: Oracle Media, Oracle Text, Spatial Oracle

Przykład klasy systemowej w Spatial Oracle

```
CREATE TYPE sdo_geometry AS OBJECT (  
    SDO_GTYPE NUMBER,                -- typ geometrii  
    SDO_SRID NUMBER,                 -- układ współrzędnych  
    SDO_POINT SDO_POINT_TYPE,       -- współrzędne punktu  
    -- interpretacja współrzędnych  
    SDO_ELEM_INFO SDO_ELEM_INFO_ARRAY,  
    -- współrzędne granic obiektu przestrzennego  
    SDO_ORDINATES SDO_ORDINATE_ARRAY);
```

Rzutowanie O-O

- Obiekty bazy danych Oracle nie są obiektami, żadnego z obiektowych języków programowania.
- Załadowanie obiektów bazy danych wymaga ich rzutowania na obiekty języka programowania.
- JDBC, SQLJ

Przykład w SQLJ:

```
CREATE TYPE person_t AS OBJECT
    EXTERNAL NAME 'Person' LANGUAGE JAVA
USING SQLData
    (ss_no number (9) external name 'ssn',
    name VARCHAR2(200) external name 'name',
    address Address_t external name 'address',
    member function length return number
        external name 'length () return int');
```

Podsumowanie własności

- Siła modelu obiektowo-relacyjnego jest całkowicie równoważna teoretycznemu homogenicznemu modelowi obiektowemu
- Rozszerzenia obiektowe są wykorzystywane do implementacji rozszerzeń systemu bazy danych – wdrożenie idei rozszerzalnego modelu danych
- Obiektowo-relacyjny model danych wymaga integracji z systemem typów danych aplikacji obiektowych