# Obiektowe i obiektowo-relacyjne bazy danych

# Plan wykładu

1. ODBMS
2. ORDBMS
3. SQL3
   - Typy
4. Źródła

# ODBMS

Obiektowy system zarządzania baza danych jest systemem opartym na modelu obiektowym wspomagającym definiowanie, zarządzanie, utrzymywanie, zabezpieczenia i dostępność obiektowej bazy danych.

# ODMG

Organizacja(Object Database ManagementGroup) powstała w1991 r. tworzyła standardy obiektowej bazy danych,z wykorzystaniem:

- języka specyfikacji obiektu ODL (Object Definition Language),
- obiektowego języka zapytań OQL (Object Query Language),
- wiązań do języków C++, Java Smalltalk

# ODL I

Specyfikacja obiektu:

Interface <name> {

attributes(<type> : <name>);

relationships(<range type> : <name>);

methods;

};

# ODL II

Przykłady specyfikacji obiektów:

```
class Obraz {
(extent Obrazy)
attribute Date utworzony;
attribute set<Figura>Elementy;
relationship set<Osoba> autorzy
inverse Osoba::utworzył;
relationship set<Obraz>jest Pierwowzorem
inverse Obraz::jest Modyfikacją;
relationship set<Obraz>jest Modyfikacją
inverse Obraz::jest Pierwowzorem;
void dodaj (in Figura fig)
raises(NiewłaściwaFigura);
Obraz utwórzKopię( )
raises(ZaDużoKopii); };
```

# OQL

Język deklaratywny oparty na SQL'u, przystosowany do współpracy z C++,java i Smalltalk, dla każdego zapytania wymagana jest obecność trwałego obiektu będącego instancja zdefiniowanej klasy.

```
SELECT struct(nazwisko:s.imNaz.nazwisko,
imie:s.imNaz.imie,
sredniaOcen:s.sredniaOcen)
FROM s in studenci
WHERE s.stopienStudiow= 2 AND s.semestr= 3
ORDER BY sredniaOcendesc, nazwisko,imie;
```

# JDO

Standard(Java Data Objects) stworzony przez firme Sun, wykorzystujący:

- model obiektu używany w języku Java,
- *persistencebyreachability*
- do specyfikacji obiektów - XML,
- JDOQL jako język zapytań.

```
Query query=pManager.new Query(Osoba.class,

"wiek >= 50 && wiek < 60");
```

# Obiektowo-relacyjne bazy danych ORDBMSI

Bazy tworzone w tym modelu są często nazywane rozszerzonymi bazami relacyjnymi,bądź postrelacyjnymi, czy hybrydowymi.

Zapewniają:

- kolekcje (zbiory,wielozbiory, sekwencje,zagnieżdżone tablice, tablice o zmiennej długości,tabele obiektów),
- metody użytkownika(funkcje i procedury definiowane przez użytkownika w różnych językach C++,VisualBasic, Java, PL/SQL)
- przeciążanie funkcji,
- typy referencyjne,
- perspektywy obiektowe,

# Obiektowo-relacyjne bazy danych ORDBMS II

- hierarchie zbiorów danych,
- przystosowanie do multimediów(duże obiekty BLOB, CLOBi pliki binarne),
- dane przestrzenne (spatial),
- abstrakcyjne typy danych (ADT)
- język zapytańSQL3 zwany tez ObjectSQL.

Z modelu relacyjnego pozostawiają:

- architekturę klient/serwer,
- mechanizmy buforowania i indeksowania,
- przetwarzanie transakcji,
- optymalizacje zapytań.

# SQL3 I

SQL3 jest kontynuacja linii SQL-92, rozwijana przez ANSI (American National Standard Institute), komitet X3H2,oraz ISO(International Standard Organization). Standard języka SQL powstały w 1999 rozszerzony o kolejne rozwinięcia modelu relacyjnego:

- nowe typy danych: BLOB, CLOB (brak możliwości narzucenia ograniczenia UNIQUE, niemożna go wykorzystać jako argumentu ORDER BY lub GROUP BY), BOOLEAN, ARRAY, ROW
- poszerzenie możliwości perspektyw (aktualizacja danych)
- wprowadzenie zapytań rekurencyjnych
- punkty zachowania w transakcjach
- wprowadzenie ról dla użytkowników
- rozbudowa wyzwalaczy

# SQL3 II

oraz o korzystanie z cech obiektowych modelu postrelacyjnego:

- strukturalne typy definiowane przez użytkownika(z metodami, dziedziczeniem, podtypami)
- funkcje i procedury pisane w innych językach
- tabele obiektów
- typy referencyjne -odnośniki do obiektów,niepowtarzalne i niemodyfikowalne

# SQL3 – nowe możliwości

SQL3 rozszerza poprzedni standard SQL-92 w następujących kierunkach:

- Obiektowość:SQL3 reprezentuje podejście“hybrydowe”, dodając niektóre cechy obiektowości (takie jak ADT)do tablic znanych z systemów relacyjnych.
- Rozszerzalność: umożliwienie użytkownikom deklarowania własnych typów.
- Niekonwencjonalne typy danych:multimedialne, przestrzenne, temporalne.
- Pełne możliwości uniwersalnego języka programowania dla definiowania i zarządzania trwałymi ,złożonymi obiektami.
- Rozszerzenia w zakresie aktywnych reguł, interfejsów do innych języków programowania, autoryzacji, procedur bazy danych, ewolucji

# SQL3 - rozszerzenia obiektowe

- Typy definiowane przez użytkownika(abstrakcyjne typy danych(ADT), nazwane typy wierszy, oraz typy rozróżnione (distinct)
- Konstruktory typów dla wierszy (rowtypes) i referencji (referencetypes)
- Konstruktory typów dla kolekcji: zbiorów, list (sekwencji)i wielozbiorów
- Definiowane przez użytkownika funkcje i procedury
- Wspomaganie dla dużych obiektów (Binary Large OBject-BLOB, Character Large OBject- CLOB)
- Wbudowane typy skalarne

# SQL3 - rozszerzenia typów I

- Rozszerzalność:definiowanie nowych typów “zrozumiałych” dla SZBD.
- Zwiększenie potencjału modelowania pojęciowego (złożone obiekty).
- Ponowne użycie istniejących bibliotek typów.
- Integracja pojęć obiektowych i relacyjnych w jednym języku.
- Abstrakcyjne typy danych (ADT)
- Mocna kontrola typów
- Procedury polimorficzne
- Kontrola typu podczas kompilacji
- Przywileje: kto może używać ADT

# SQL3 - rozszerzenia typów II

- Typy rozróżnione(distinct):zwiększenie skuteczności mocnej kontroli typów i wydajności.
- Typy wierszowe:
  - Są podobne do struktur lub zapisów (records) winnych językach
  - Umożliwiają określenie typów wierszy w tablicach
  - Umożliwiają zagnieżdżanie wierszy
- Typy kolekcji:
  - Wbudowane, parametryczne typy SET(T), LIST(T), MULTISET(T)
  - Odwzorowanie do tablic

# SQL3 - definicja typów

```
CREATE [OR REPLACE] TYPE nazwa_typu AS OBJECT (
atrybut1typ1 [atrybut2 typ2, ...,atrybutNtypN,]
[MEMBER PROCEDURE nazwa_procedury,]
[MEMBER FUNCTION nazwa_funkcji RETURN typ_zwracany]
)[NOT FINAL]
/
DROP TYPE nazwa_typu
/
DESCRIBE nazwa_typu
/
-- np.:
CREATE OR REPLACE TYPE adres AS OBJECT (
kodVARCHAR2(6),
miastoVARCHAR2(50),
ulicaVARCHAR2(100)
)
```

# Tabele obiektów, tabele zawierające obiekty

ROWOBJECT

- tabela “jednokolumnowa” - wiersze to obiekty
- OID - identyfikator obiektu (automatycznie nadawany)

COLUMN OBJECT

- tabela wielokolumnowa - obiekt jako atrybut

# Tabele obiektów, tabele zawierające obiekty cd. I

Tabele obiektów:

CREATE TABLE nazwa OF nazwa_typu;

np.:

CREATE TABLE adresy OF adres;

INSERT INTO adresy VALUES (adres('42-200',

'Czestochowa', 'Dabrowskiego69'));

SELECT * FROM adresy;

# Tabele obiektów, tabele zawierające obiekty cd. II

Tabela z kolumnami obiektowymi:

```
CREATE TABLE osoba (
pesel NUMBER(11) PRIMARY KEY,
nazwisko VARCHAR2(50),
imieVARCHAR2(30),
adres_zam adres
);
```

# Tabele obiektów, tabele zawierające obiekty cd. I

```
CREATE OR REPLACE TYPE ulamek AS OBJECT (
licznik NUMBER(4),
mianownik NUMBER(4)
)
/
-- tabela zawierająca obiekty
CREATE TABLE ulamki(
atrybut ulamek,
CONSTRAINT ul_ch_mian CHECK(atrybut.mianownik<> 0)
);
INSERT INTO ulamki VALUES (ulamek(12,4));
SELECT * FROM ulamki;
```

# Tabele obiektów, tabele zawierające obiekty cd. II

```
-- tabela obiektowa

CREATE TABLE ulamkiOb OF ulamek;

ALTER TABLE ulamkiOb

ADD CONSTRAINT ulO_ch_mian

CHECK(mianownik<>0);


INSERT INTO ulamkiOb VALUES (12,4);

INSERT INTO ulamkiOb VALUES (ulamek(12,4));


SELECT * FROM ulamkiOb;

SELECT Value(u) FROM ulamkiObu;
```

# SQL3 - metody obiektów I

```
CREATE OR REPLACE TYPE ulamek AS OBJECT (
licznik NUMBER(4),
mianownik NUMBER(4),
MEMBER PROCEDURE skroc
)
/

CREATEOR REPLACE TYPE BODY ulamek
ASMEMBER PROCEDURE skroc IS
vr NUMBER;
BEGIN
r:=Nwd(licznik, mianownik);
licznik:= licznik/r;
mianownik:= mianownik/r;
END skroc;
END;
/
```

# Typy złożone

```
CREATE TYPE punkt AS OBJECT (
x NUMBER,
y NUMBER
);

CREATE TYPE tablicaWierzcholkow
AS VARRAY(2) OF punkt;

CREATE TYPE odcinek AS OBJECT (
wierzcholki tablicaWierzcholkow
);

CREATE TYPE zbiorKrawedzi
ASTABLE OF odcinek;

CREATE TYPE wielokat AS OBJECT (
krawedzie zbiorKrawedzi
);
```

# Dziedziczenie, typy abstrakcyjne, polimorfizm I

```
--klasa abstrakcyjna
CREATE TYPE figura AS OBJECT (
typ VARCHAR2(10),
NOT INSTANTIABLE  MEMBER FUNCTION
pole RETURN FLOAT
) NOT INSTANTIABLE NOT FINAL;

-- klasa dziedzicząca- UNDER
CREATE OR REPLACE TYPE kolo UNDER figura(
promien FLOAT,
OVERRIDING MEMBER FUNCTION
pole RETURN FLOAT
);

--redefinicja - przeładowanie - OVERRIDING
CREATE OR REPLACE TYPE BODY kolo AS
OVERRIDING MEMBER FUNCTION
pole RETURN FLOAT
```

# Dziedziczenie, typy abstrakcyjne, polimorfizm II

```
IS
BEGIN
RETURN3.14159265358979 *promien*promien;
END;
END;

-- klasa dziedzicząca
CREATE OR REPLACE TYPE kwadrat UNDER figura(
dl_boku FLOAT,
OVERRIDING MEMBER FUNCTION
pole RETURN FLOAT
);

-- redefinicja
CREATE OR REPLACE TYPE BODY kwadrat AS
OVERRIDINGMEMBER FUNCTION
pole RETURN FLOAT
IS
BEGIN
```

# Dziedziczenie, typy abstrakcyjne, polimorfizm III

```
RETURN dl_boku*dl_boku;
END;
END;

-- tabela obiektów
CREATE TABLE figury OF figura;
INSERT INTO figury VALUES(new kolo('kolo',10) );
INSERT INTO figury VALUES(new kwadrat('kwadrat',10));

SELECT * FROM figury;
-- koła i kwadraty
SELECT Value(f)  FROM figury f;
-- same kwadraty
SELECT Value(f) FROM figury f
WHERE Value(f) IS OF (kwadrat);
-- szczegółowe dane
SELEC Ttyp, Treat(Value(f) AS kwadrat).dl_boku AS bok,
Treat(Value(f) AS kolo).promien AS promien,f.pole()
FROM figury f;
```

# Dziedziczenie, typy abstrakcyjne, polimorfizm IV

- TREAT- funkcja umożliwiająca rzutowanie w dół
- ALTER TYPE nazwa_typu NOT INSTANTIABLE; -zmiana na typ abstrakcyjny
- IS OF – operator pozwalający na rozpoznanie typu obiektu najniższego w hierarchii dziedziczenia

# NOT FINAL,NOT INSTANTIABLE

FINAL vs NOT FINAL:

- typy – na końcu definicji
- metody – na początku definicji, przed nazwa

INSTANTIABLE vs NOT INSTANTIABLE

- typy – na końcu definicji
- metody – na początku definicji, przed nazwa
  - nie podany - wymagana definicja metody, nawet jeżeli typ jest abstrakcyjny
  - podany - typ musi być abstrakcyjny

# Typy referencyjne I

- OID
  - identyfikator obiektu wierszowego
- REF
  - logiczny wskaźnik do obiektowych wierszy
  - umożliwia stworzenie relacji pomiędzy obiektami różnych typów
  - umożliwia tworzenie relacji jeden do wielu
- DEREF
  - operator umożliwiający dostęp do obiektu wskazywanego przez REF

# Modelowanie związków- typy referencyjne I

```
CREATE TYPE osoba AS OBJECT (
imieVARCHAR2(10),
nazwiskoVARCHAR2(20)
);

-- tabela realizujaca związek do tabeli osoby
CREATE TYPE obraz AS OBJECT (
utworzony DATE,
autor REF osoba
);

-- tabele obiektowe
CREATE TABLE osoby OF osoba;
CREATE TABLE obrazy OF obraz;
```

# Modelowanie związków- typy referencyjne II

```
INSERT INTO osoby VALUES (’Jan’,’Kowalski’);


-- wstawienie powiązania

INSERT INTO obrazy

SELECT Current_date, Ref(o)

FROM osoby o

WHERE nazwisko LIKE ’Kowalski’;


SELECT * FROM obrazy;
```

# Modelowanie związków- typy referencyjne III

```
-- typ zawierający referencje do osób

CREATE TYPE zbiorAutorow AS TABLE OF REF Osoba;


CREATE TYPE dzielo AS OBJECT (

utworzone DATE,

autorzy zbiorAutorow

);


--tabela zzagnieżdżona tabela

CREATE TABLE dziela OF dzielo

NESTEDTABLE autorzy STORE AS aut;
```

# Modelowanie związków- typy referencyjne IV

```
INSERT INTO osoby VALUES (’Piotr’,’Nowak’);

-- wstawienie dzieła i pierwszego autora
INSERT INTO dziela
SELECT Current_date, zbiorAutorow(Ref(o))
FROM osoby o
WHERE nazwisko LIKE ’Kowalski’;

-- wstawienie kolejnego autora
INSERT INTO TABLE (SELECT autorzy FROM dziela)
(SELECT Ref(o) FROM osoby o
WHERE nazwisko = ’Nowak’);
```

# Źródła

W wykładzie wykorzystano materiały:


- Wykład dr inż.Olga Siedlecka-Lamch
- http://www.eyedb.org/wp-content/uploads/documentation/manual/html/ODL/node2.html
- http://db.apache.org/jdo/index.html
- fatcat.ftj.agh.edu.pl/~mariosh/DB_II/ObjectRelationalOracle.pdf
- http://www.cs.put.poznan.pl/mmorzy/
- http://www.ipipan.eu/staff/k.subieta/prezentacje/Obiektowe%20bazy%20danych%20kontra%20relacyjne%201997.ppt
- C.Zaniolo,S.Ceri,Ch.Faloutsos, R.T.Snodgrass, V.S.Subrahmanian,R.Zicari, Advanced DatabaseSystems, Morgan Kaufmann, 1997