Objektově orientované databáze

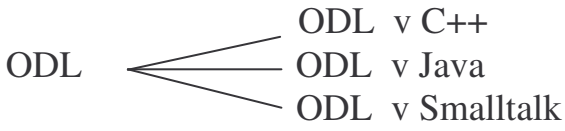
CORBA (Common Object Request Brooker Architecture)

ODL (Object Definition Language)

OQL (Object Query Language)

ODL

Účel - dovolit OO návrh DB a přímý překlad do OODBMS (obdoba DDL)



Objektově orientovaný návrh - modeluje svět složený z objektů

Objekty jsou seskupovány do tříd

Objekt má jedinečné OID
může mít i jméno použitelné k odkazování (vstupní body databáze)
OID vytváří systém - v centralizovaném prostředí z času vytvoření objektu
- v distribuovaném prostředí ---------“---------- +
z identifikace hostitele

Při popisu ODL tříd určujeme:

- atributy
- relace
- metody (signatury)

použití metod je součástí OQL
Metody jsou popsány v hostitelském jazyce

Deklarace objektových typů

interface - k popisu abstraktního chování
(použitelné jen k dědění operací)
class - k popisu abstr. chování i abstr. stavu (k dědění i k vytváření objektů). Class slouží k popisu databázového schéma

interface <jméno třídy> {<seznam vlastností>}
class <jméno třídy> {<seznam vlastností>}

Atributy v ODL

jsou nejjednodušším typem vlastností (atomické, složené)

Př.

```
interface Obrazec {
        attribute enum Tvar {Obdelnik, Kruh. Trojuhelnik} Druh;
        attribute struct Bod {short x, short y}  Ref_bod;
        float plocha();
};
class Obdelnik: Obrazec {
        attribute struct Bod {short x, short y}  Ref_bod;
        attribute  short  Delka;
        attribute  short  Sirka;
};
class Kruh: Obrazec {
        attribute struct Bod {short x, short y}  Ref_bod;
        attribute  short  Radius;
};
```

Př.

```
class  Film  {
                attribute string  titul;
                atribute short  rok;
                attribute short  delka;
                attribute enum  EFilm  {barevny, cernobily}  typfilmu;
        };
```

Př. složeného atributu

```
class  Herec  {
        attribute string  jmeno;
        attribute Struct  Adr  {string  ulice, string mesto} adresa;
};
```

Typ atributu nesmí být třídou

Relace v ODL prostředek spojování objektů

př. chceme přidat k třídě Film vlastnost - množinu jeho herců
=> je nutné vytvořit relaci mezi třídami Film - Herec

relationship Set <Herec> herci ;

a vložit jí do třídy Film

př. Chceme vyjádřit relaci jediného objektu s třídou

relationship Herec hlavníherec;

ODL zavádí jen binární relace (n-ární lze transformovat)

Binární relace musí být obousměrná

Inverzní relace

př.) chceme zaznamenat filmy, ve kterých hraje daný herec.

Přidáme do třídy Herec:
relationship Set <Film> hrajev

propojení herci a hrajev zajistíme formulí:
inverse <jméno propojované relace>

v případě propojování relace z jiné třídy, použijeme kvalifikace:
jméno třídy :: jméno relace

Př. Zavedení inverze mezi hrajev třídy Herec a herci třídy Film

```
class Herec {
        attribute string jmeno;
        attribute Struct Adr { string ulice, string mesto } adresa;
        relationship Set < Film > hrajev
                inverse Film :: herci ;
};
```

Relace Herci - Filmy je M : N

ODL vyjadřuje násobnost operátorem Set v deklaraci relace

Př. Zaveďme třídu Studio, reprezentující producentskou společnost

```
1)      class Film {
2)              attribute string titul;
3)              attribute short rok;
4)              attribute short delka;
5)              attribute enum Efilm {barevny, cernobily} typfilmu;
6)              relationship Set < Herec > herci
                        inverse Herec :: hrajev;
7)              relationship Studio vlastneny       /*ma jedineho vlastnika*/
                        inverse Studio :: vlastni ;

8)      class Herec {
9)              attribute string jmeno;
10)             attribute Struct Adr {string ulice, string mesto} adresa;
11)             relationship Set < Film > hrajev
                        inverse Film :: herci;
        };

12)     class Studio {
13)             attribute string jmeno;
14)             attribute string adresa;/*stejne jmeno atributu. ale v jine
                                        tride*/
15)             relationship Set < Film > vlastni
                        inverse Film :: vlastneny ;   /*muze vlastnit vice
                                                        filmu*/
        };
```

Relace mezi třídami:

M : N herci z třídy Film do třídy Herec
hrajev --"-- Herec --"-- Film

N : 1 vlastněný z třídy Film do třídy Studio

1 : N vlastní z třídy Studio do třídy Film

1 : 1 ?

Objekty a literály

1. Literály (konstanty) -atomické
-kolekce
-strukturované

2. Objekty -mají OID (a popř. i jméno)
-mohou mít komplexní strukturu danou konstruktorem

Atomické objekty se specifikují pomocí class
Většina objektů má komplexní strukturu s atributy, operacemi apod

Konstruktory pro objekty typu kolekce:

a) Množiny Set < libovolný_typ >
b) Multimnožiny Bag < libovolný_typ >
c) Seznamy List < libovolný_typ >
např. string je List < char >
d) Pole Array < T, i >
e) Slovník Dictionary<k, v>
vytváří asociaci dvojic key - value
o.bind(k,v) vytvoří asociaci k-v v objektu o
o.unbind(k,v) zruší "
v=o.lookup(k) přiřadí s k asociovanou hodnotu v o

Kolekce mají všechny elementy stejného typu

Konstruktor struktury má tvar:
Struct N {T1 F1, T2 F2, ... , Tn Fn}

Př. class Herec {
attribute string jmeno ;
attribute Set < Struct Adr {string ulice, string mesto } >
adresa;
relationship Set < Film > hrajev inverse Film :: herci ;
};

Podtřídy a dědičnost

- interface jméno_subinterface : jméno_interface1,: jméno_interface2…
{ seznam vlastností subinterface }

- class jméno_třídy : jméno interface1, : jméno_interface2 …
{ seznam vlastností třídy }

- class jméno_podtřídy extends jméno třídy { seznam vlastností
podtřídy }

Př. class Muzikál extends Film {

relationship Set < Herec > hlasy;

} ;

př.) class Krvák extends Film {

attribute long long početmrtvol;

} ;

Násobná dědičnost je možná pouze od interface

Deklarace klíčů

! OO model klíče nepotřebuje, umístění objektu určí z OID

ODL umožňuje definovat klíče pro návaznost na ER model a relační databáze

tvar:
(key seznam-klíčů) nebo (keys seznam-klíčů) před { ...

př.) class Herec (key jméno) { ...

př.) class Film (key (titul, rok)) { ...

př) alternativní klíč

class Zaměstnanec (key osobníčíslo, čísloobčprůkazu){ ...

složený klíč

class Zaměstnanec (key (osobníčíslo, čísloobčprůkazu)) { ...

Deklarace signatur metod v ODL

Signatura určuje - jméno metody sdružené s třídou
- vstupní / výstupní typy metody

Kód metody je zapsán v hostitelském jazyce, není součástí ODL

Syntax obdobná funkcím C mimo:
- specifikace parametrů in, out, inout
- fce může způsobit výjimky
raises (seznam výjimek)

Každá ODL třída může mít deklarován **extent**

extent = **(rozsah)** pojmenování současné množiny objektů této třídy

je obdobou jména relace

OQL dotazy se týkají extent, ne samotné třídy

Př.

```
1       class FILM
2               (extent Filmy
3               key (titul, rok))
        {
4               attribute string titul;
5               attribute short rok;
6               attribute short delka;
7               attribute enum (barevny, cernobily) typfilmu;
8               relationship Studio vlastneny
                        inverse  Studio  ::  vlastní;
9               relationship Set <Herec> herci
                        inverse Herec  ::  hrajev;
10              float naklady ( )  raises(nenalezenynaklady);
11              void jmenahercu(out  Set <String>);
12              void jinefilmy(in Herec,  out Set <Film>)
                                        raises(nenitakovyherec);
        };

        class Herec
                (extent Herci
                 key jmeno)
        {
                attribute string jmeno;
                attribute  Struct Adr
                                        {string ulice, string město} adresa;
                relationship Set <Film>  hrajev
                                inverse  Film  ::  herci;
                };

        class Studio
                (extent Studia
                 key  jmeno)
        {
                attribute string jmeno;
                attribute string adresa;
                relationship Set <Film>  vlastní
                                inverse Film :: vlastneny;
        };
```

OBJECT ORIENTED QUERY LANGUAGE

OQL přenos SQL do OO prostředí

SQL3 přenos OO do relačního prostředí

Základní konstrukce OQL

-OQL není úplným jazykem (Obdoba SQL92)

-Je chápán jako rozšíření hostitelského jazyka (C++, Smalltalk, Java)

-Poskytuje deklarativní přístup k objektům

-Umožňuje práci s objekty (mají OID) i s literály (identické se svou hodnotou)

- Neobsahuje explicitně UPDATE

Vytváření objektů

např. Film(jmeno: "Kolja", rok: 1998, delka: 125, typfilmu: barevny)
vytvoří objekt s OID, inicializovaný hodnotami

struct (a: 10, b: "Jan")
vytvoří objekt bez identity (konstantu) - strukturu se dvěma položkami

-pomocí SELECT

-pomocí konstruktorů
- Set
- Bag
- List
- Array
- jména tříd

př.)
```
x = Struct ( a : 1 , b : 2 ) ;
y = Bag ( x , x , Struct ( a : 3 , b : 4 ) ;
z = Film ( titul : ´´Titanic´´ , rok : 1997 , délka : 130 , . . . ) ;
```

Typový systém

-Proměnné užívané v OQL jsou většinou deklarovány v hostitelském jazyce

-Oproti ODL potřebuje OQL i konstanty

- základní typy (podobně v SQL)

  -atomické
  -vyjmenované

- složené typy - zaváděny pomocí konstruktorů

  -Set (...)
  -Bag (...)
  -List (...)
  -Array (...)
  -Struct (...)

př.) Struct (M : Bag (2, 1, 2) , R : ´´nic´´)

K odkazům na části objektů se užívá . notace, synonymem je ->

př.) Nechť a je objekt. Pak a . p značí:

- je-li p atribut pak hodnotu atributu
- je-li p relace pak objekt nebo kolekci objektů, které jsou v relaci p s objektem a
- je-li p metoda pak výsledek aplikace p na objekt a

př.) Nechť hostitelský jazyk deklaruje proměnnou můjoblíb typu Film

```
můjoblíb . délka
můjoblíb . náklady()
můjoblíb . jménaherců ( mojiherci )
můjoblíb . vlastněný . jméno
```

SELECT FROM WHERE výrazy v OQL

př. Dotaz na rok vzniku filmu Most u Remagenu

```
SELECT m . rok FROM Filmy m
    WHERE m . titul = ´´Most u Remagenu´´
```

Obecně:

1. SELECT seznam výrazů

2. FROM seznam deklarací proměnných v podobě

   a) výrazu jehož podoba je typu kolekce
      (nejčastěji extent třídy, může to být ale libovolný výraz produkující kolekci)
   b) nepovinně AS
   c) jméno proměnné

3. WHERE booleovský výraz

Dotaz produkuje multimnožinu - bag

př.) Dotaz na jména herců ve filmu Kolja

```
SELECT s . jméno
    FROM Filmy m , m . herci s          /*pořadí je důležité*/
    WHERE m . titul = ´´Kolja´´
```

Vyhodnocení lze znázornit cyklem:

```
FOR  všechny m IN Filmy DO
    FOR  všechny s IN m . herci DO
        IF  m . titul = ´´Kolja´´  THEN
                přidej s . jméno do výstupní multimnožiny ;
```

Eliminace duplikátů jako v SQL:

```
SELECT DISTINCT s . jméno FROM Filmy m , m . herci s
    WHERE m . vlastněné . jméno = ´´Paramount´´
```

Složené výstupní typy

Výrazy za SELECT mohou používat i konstruktorů typů

př.) Najdi množinu dvojic herců se stejnou adresou

```
SELECT DISTINCT Struct ( herec1 : s1 , herec2 : s2 )

        FROM Herci s1 , Herci s2

        WHERE s1 . adr = s2 . adr AND s1 . jméno < s2 . jméno
```

výsledek dotazu je typu:
Set < Struct N { herec1 : Herec , herec2 : Herec} >

Stejný efekt má příkaz:
SELECT DISTINCT herec1 : s1 , herec2 : s2 . . .

Poddotazy

SELECT FROM WHERE příkaz lze použít všude, kde může být kolekce

př.) Dotaz na herce filmů z produkce Paramount

```
SELECT DISTINCT s . jméno
        FROM ( SELECT m FROM Filmy m
            WHERE m . vlastněný . jméno = ´´Paramount´´ ) d,
                d . herci s
```

Uspořádání výsledku

Výsledkem OQL dotazu je **bag** nebo **set**

s použitím ORDER BY seznam_výrazů získáme **list**

př.) Nalézt filmy Paramount a uspořádat je dle délky

```
SELECT m FROM Filmy m
    WHERE m . vlastněný . jméno = ´´Paramount´´
    ORDER BY m . délka
```

Složitější formy OQL výrazů

- kvantifikátory
- agregační operátory
- množinové operátory
- operátor GROUP BY

Kvantifikované výrazy

FOR ALL x IN S : C(x)

EXISTS x IN S : C(x)

př.) Najdi všechny herce z filmů Paramount

```
1.  SELECT s
2.      FROM Herci s
3.      WHERE EXISTS m IN s . hrajev :
4.          m . vlastněný . jméno = ´´Paramount´´
```

př.) Dotaz na herce, kteří hrají ve všech Paramount filmech

```
1.  SELECT s
2.      FROM Herci s
3.      WHERE FOR ALL m IN s . hrajev :
4.          m . vlastněný . jméno = ´´Paramount´´
```

Agregační výrazy

AVG, SUM, COUNT, MIN, MAX

- v SQL je lze aplikovat jen na označený sloupec tabulky

- v OQL "----------" na kolekce, jejichž členy jsou vhodného typu.

COUNT	na libovolnou kolekci
SUM, AVG	na kolekce aritmetických typů
MAX, MIN	na kolekce libovolných porovnatelných typů

př.) AVG (SELECT m . délka FROM Filmy m)

Výrazy GROUP BY

Tvar: GROUP BY f1 : e1 , f2 : e2 , . . . , fn : en

Navrací: Set<Struct (f1 : v1 , f2 : v2 , . . . , fn : vn , partition : P)>

P je multimnožina struktur tvaru Struct (x : i)
kde x je proměnná z klazule FROM

př.) Vyhledej tabulku celkových délek filmů pro každé studio a pro každý rok

```
SELECT std , r , sumdélka : SUM ( SELECT  p . m . délka
                                  FROM  partition p )

        FROM  Filmy  m

        GROUP  BY  std : m . vlastněný ,  r : m . rok
```

Konstruuje multimnožinu struktur s komponentami studio, rok, celkovádélka

Výsledkem bude: Struct (std : ´´DEFA´´ , r : 1900 , sumdélka : 1200)
. . .
Struct (std : ´´ČSFILM´´ , r : 1918 , sumdélka : 200)
. . .

Klauzule HAVING

Uvedena za GROUP BY ve tvaru : HAVING podmínka

př.) Výběr jen těch studií, která v příslušném roce vyprodukovala alespoň jeden z filmů delších než 120

SELECT std , r , sumdélka : SUM (SELECT p . m . délka FROM partition p)

FROM Filmy m

GROUP BY std : m . vlastněný , r : m . rok

HAVING MAX(SELECT p . m . délka FROM partition p) >120

Množinové operátory

UNION , INTERSECT , EXCEPT

př.) Najdi množinu filmů s Henri Fondou, které nevyprodukoval Paramount

```
( SELECT  DISTINCT  m
      FROM  Filmy  m ,  m . herci  s
      WHERE  s . jméno  =  ´´Henri Fonda´´ )
EXCEPT
  ( SELECT  DISTINCT  m
      FROM  Filmy  m
      WHERE  m . vlastněný . jméno  =  ´´Paramount´´ )
```

Je-li alespoň jeden z operandů multimnožina, jsou operátory chápány multimnožinově

Nechť: -B1 je multimnožina s n1 výskyty objektu x

-B2 je multimnožina s n2 výskyty objektu x

- v B1 ∪ B2 je x n1 + n2 krát
- v B1 ∩ B2 je x min (n1 , n2) krát
- v B1 − B2 je x při n1 <= n2 0 krát
  při n1 > n2 n1 − n2 krát

Přiřazování objektů v OQL

Dotazy OQL produkují objekty => ty lze přiřadit proměnným hostit. jazyka

př.) staréfilmy = SELECT DISTINCT m

FROM Filmy m WHERE m . rok < 1945 ;

Výběr prvku z kolekce

-Konverze jednoprvkové kolekce na prvek (operátor ELEMENT)

```
př.)    x = ELEMENT ( SELECT  m
                      FROM  Filmy  m
                      WHERE  m . titul = ´´Kleopatra´´ ) ;
```

-Získání každého prvku z množiny či z multimnožiny

a) změníme set / bag na list

b) na prvek odkazujeme pořadovým číslem (0, 1, . . .)

př.) Vytisknout titul, rok, délku každého filmu

```
seznamfilmů = SELECT  m  FROM  Filmy  m
                      ORDER  BY  m . titul , m . rok ;

počtefilmů = COUNT ( Filmy ) ;

for ( i = 0 ; i < počtefilmů ; i++ ) {
            film = seznamfilmů[ i ] ;
            cout << film . titul << ´´   ´´ <<  film . rok << ´´   ´´
                  << film . délka << ´´\n´´ ; }
```

Vazba na C++

Knihovna tříd C++ poskytuje třídy a operace pro implementaci ODL konstrukcí
Způsob práce s objekty v C++ popisuje jazyk OML