Objektově-relační DB ORACLE 9.i

Petr Klemšinský xklemsin@fi.muni.cz

Objektově-relační databáze Oracle

- Je funkčním rozšířením relační databáze
  - Využití objektových metod (zapouzdření, dědičnost, polymorfismus)
- Umožňuje do atributu tabulky ukládat celé objekty (1 NF?)
- Objektové tabulky – pro ukládání objektů daného typu
- Možnost definování nových uživatelských typů

Objektové typy

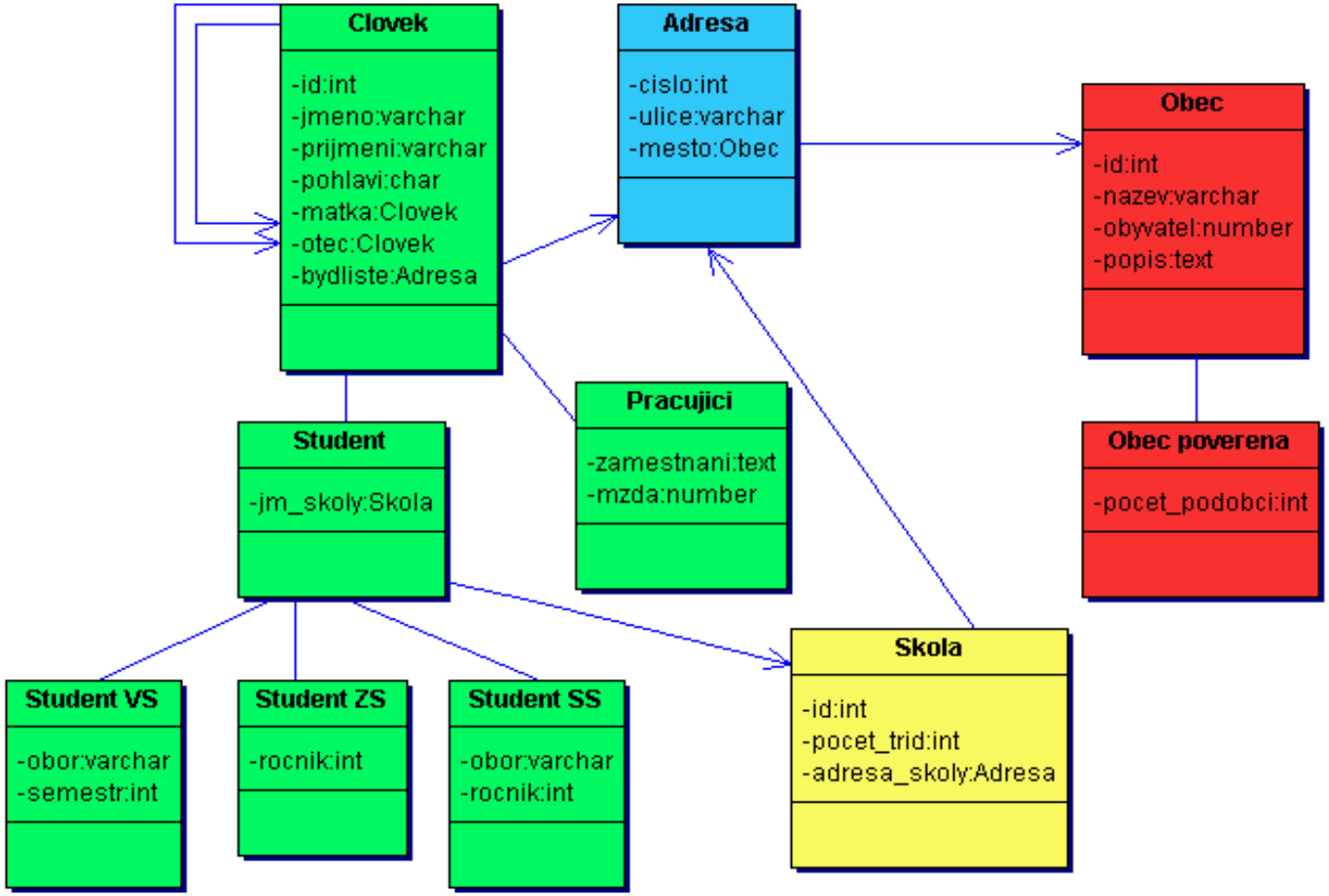
- Reálné objekty jsou definovány objektovými typy
- Definice typu se skládá:
  - specifikace
    - struktura dat (atributy)
    - hlavičky metod
  - tělo
    - plně definuje metody
- Př.:

```
CREATE TYPE person AS OBJECT (
id                  INT,
name                VARCHAR(20),
surname             VARCHAR(35),
MEMBER PROCEDURE add_person
(i_id  INT, i_name  VARCHAR, i_surname VARCHAR)
);
/
```

Atributy

- Deklarovány jménem a typem
- Mohou být libovolného typu (i objektového)
- Speciální atributy
  - Odkaz – typ REF
  - Kolekce – typ NESTED TABLE,VARRAY
- Díky ukazatelům REF můžeme vytvářet i značně sloužité systémy

Model databáze



Atributy typu REF

- Obsahují ukazatele na existující objekty (nebo NULL)
- Používají pro přístup k atributům tečkovou notaci
- Částečně mohou nahrazovat integritní omezení
- Umožňují jednoduché přecházení mezi objekty
- Lze pomocí nich získat kopii objektu
- Může být použit pro změnu atributů jiného objektu

Atributy typu REF

Př.:

```
CREATE TYPE address_t AS OBJECT (
id                          INT,
street                      VARCHAR(20),
city                        VARCHAR(35),
number                      INT
);
/

CREATE TYPE person AS OBJECT (
id                          INT,
name                        VARCHAR(20),
surname                     VARCHAR(35),
address                     REF t_address,
MEMBER PROCEDURE add_person
(i_id INT,i_name VARCHAR,i_surname VARCHAR,i_address_id INT)
);
/
```

Pokud X je typu `person` , tak výraz `X.address.city` ukazuje na město, ve kterém osoba žije

Kolekce

- Slouží k modelování vztahů 1:N

- Typy kolekcí
  - VARRAY
    - Reprezentují množiny s omezeným počtem elementů
    - Mají definováno uspořádání na elementech
  - NESTED TABLE
    - Reprezentují množiny elementů
    - Počet elementů neomezen, nemá definováno uspořádání

- Všechny elementy musejí být stejného typu

- Příklad definice kolekce (typu NESTED TABLE)

```
CREATE TYPE nt_typ AS TABLE OF element_type;
```

Kolekce

- Schéma uložení vnořené tabulky

DATA1	DATA2	DATA3	DATA4	NT_DATA
…	…	…	…	A
…	…	…	…	B
…	…	…	…	C
…	…	…	…	D
…	…	…	…	E

Storage Table

NESTED_TABLE_ID	Values
B	B21
B	B22
C	C33
A	A11
E	E51
B	B25
E	E52
A	A12
E	E54
B	B23
C	C32
A	A13
D	D41
B	B24
E	E53

Metody

- Používány pro manipulaci s objekty a jejich atributy, pro vytváření objektů, jejich porovnávání a pod.

- Je možné jim předávat IN a OUT parametry
  - Implicitně je předáván parametr SELF

Př.:

```
CREATE OR REPLACE TYPE BODY person AS
   MEMBER PROCEDURE add_person
   (i_id INT,i_name VARCHAR,i_surname VARCHAR,i_address_id INT) AS
   BEGIN
        id := i_id;
        jmeno := i_jmeno;
        prijmeni := i_prijmeni;
        pohlavi := i_prijmeni;

        SELECT REF(o) INTO address
        FROM address_tab o
                  WHERE o.id = i_address_id;

        INSERT INTO person_tab VALUES(SELF);
        END add_person;
   END;
   /
```

Dědičnost typů

- Implementována od verze Oracle 9.i
- Používány při popisu podmnožin speciálních instancí objektu **(person – student, employee)**
- Rodičovský typ musí být deklarován jako NOT FINAL
- Podtyp dědí všechny atributy a metody od rodiče

Dědičnost typů

- Podtyp má definovány další parametry a případně i metody
- Lze předefinovat těla zděděných metod
- Polymorfismus typů ("subclass" polymorfismus)

Př.:

```
ALTER TYPE person NOT FINAL;

CREATE TYPE student UNDER person (
school                    VARCHAR(30),
year            INT
);
/
```

Manipulace s daty - DML

- Vkládání objektů
  - Přímo do tabulky

    Relačně

    ```
    INSERT INTO person_tab VALUES(1,'Jiří','Novák',null)
    ```

    Užitím implicitního konstruktoru

    ```
    INSERT INTO person_tab VALUES(person(2,'Josef','Novák',null))
    ```

  - Pomocí PL/SQL bloku
- Do tabulky můžeme vkládat instance podtypů

```
INSERT INTO person_tab VALUES(student(3,'Jan','Novák',null,’FI MUNI’,3))
```

Manipulace s daty - DML

- Získávání dat
  - Relačně

```
SELECT * FROM person_tab
```

!!vrací atributy typu person i u instancí student

  - Objektově

```
SELECT value(p) FROM person_tab p
```

výsledek dotazu:

```
VALUE(P)(ID, NAME, SURNAME, ADDRESS)
------------------------------------------------------------
PERSON(1, 'Jiří', 'Novák', NULL)
PERSON(2, 'Josef', 'Novák', NULL)
STUDENT(3, 'Jan', 'Novák', NULL, 'FI MUNI', 3)
```

Manipulace s daty - DML

- Vnořené tabulky
  - V definici tabulky nutné specifikovat jméno vnořené tabulky

```
 CREATE TYPE nested_sites AS TABLE OF VARCHAR(20);
 /
 CREATE TYPE Animal as object (
CAS       VARCHAR(20),
td50      NUMBER,
sites     nested_sites,
 );
 /

 CREATE TABLE animal_tab OF Animal
NESTED TABLE store as nt_sites
 ;
```

  - Přístup do vnořené tabulky jen přes instance typu

```
SELECT * FROM TABLE (

    SELECT o.site from animal_tab where o.CAS='12/12/2001-25');
```