Objektově orientované vlastnosti v SQL a jejich implementace v Oracle

Pavel Kříž
Katedra informatiky a kvantitativních metod
Fakulta informatiky a managementu
Univerzita Hradec Králové
Pavel.Kriz@uhk.cz

esf evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Standardy a implementace

- SQL3 → SQL:1999, SQL:2003

- Oracle od verze 8i
- PostgreSQL
- …

- Uživatelsky definované typy (UDT)
- Předpřipravené knihovny (např. Oracle Spatial)

Uživatelsky definované typy (UDT)

- = třídy

- „struktura“ + metody
- použití
  - jako typ řádku v tabulce („objektová tabulka“)
  - jako typ sloupce (atributu) – hnízdění
  - pomocné objekty v procedurách apod.

Definice UDT

- SQL standard:

```
CREATE TYPE <typename> AS (
      <list of attribute-type pairs>
);
```

- Oracle:

```
CREATE TYPE <typename> AS OBJECT (
      <list of attribute-type pairs>
);
```

Příklad UDT v SQL:1999 vs. Oracle

- SQL standard

```
CREATE TYPE t_osoba AS (
    jmeno VARCHAR(30),
    prijmeni VARCHAR(30),
    narozeni DATE)
INSTANTIABLE
NOT FINAL
REF IS SYSTEM GENERATED
METHOD vek() RETURNS INTEGER;
```

- Oracle

```
CREATE TYPE t_osoba AS OBJECT (
  jmeno      VARCHAR2(30),
  prijmeni   VARCHAR2(30),
  narozeni   DATE,
  MEMBER FUNCTION vek RETURN INTEGER);
```

i v Oracle jsou k dispozici možnosti FINAL či NOT INSTANTIABLE

Příklad UDT – těla metod (Oracle)

- těla metod jsou oddělená v „TYPE BODY“

```
CREATE OR REPLACE TYPE BODY t_osoba AS

  MEMBER FUNCTION vek RETURN INTEGER IS
  BEGIN
    RETURN TO_NUMBER(SYSDATE - naroz)/365;
  END;

  -- pripadne dalsi metody

END;
```

lze i SELF.naroz

SELF reprezentuje instanci (jako *this* v Jave)

Objektové modifikátory tříd a metod

- [NOT] FINAL
  - z třídy (ne)lze dědit
  - metodu (ne)lze přepsat
- [NOT] INSTANTIABLE
  - z třídy (ne)lze vytvářet instance (není abstraktní)
  - metoda (ne) má definovanou implementaci (není abstraktní)
- [NOT] OVERRIDING
  - metoda (ne)přepisuje metodu z předka

Řazení instancí (SQL:1999)

- UDT typ definuje metody EQUAL a LESSTHAN:
  o1.EQUAL(o2)
  o1.LESSTHAN(o2)
  - jako parametr tyto metody obdrží jinou instanci téhož UDT typu
  - vrátí true/false v případě, že instance, na které se metoda volá je rovná (resp. menší než) instance předaná jako parametr
- díky tomu lze pak instance porovnávat (<,>,=) a řadit (ODER BY)

Řazení instancí (Oracle)

1. pomocí ORDER metody („compare“)
   - < 0 (-1)
   - = 0
   - > 0 (1)
2. pomocí MAP metody
   - mapuje instanci na ordinární typ (lze podle něj řadit, „projekce“)

- třída může mít definovanou nanejvýš jednu ORDER nebo MAP metodu

Řazení instancí (Oracle) 1.

```
CREATE TYPE t_osoba AS OBJECT (
  jmeno      VARCHAR2(30),
  prijmeni   VARCHAR2(30),
  narozeni   DATE,
  MEMBER FUNCTION vek RETURN INTEGER,
  ORDER MEMBER FUNCTION compare(o2 t_osoba) RETURN INT
);
```

```
CREATE TYPE BODY t_osoba AS
   ...
   ORDER MEMBER FUNCTION compare(o2 t_osoba) RETURN INT IS
   BEGIN
      IF SELF.jmeno < o2.jmeno THEN RETURN –1;
      ELSIF SELF.jmeno = o2.jmeno THEN RETURN 0;
      ELSE RETURN 1;
      END IF;
   END;
END;
```

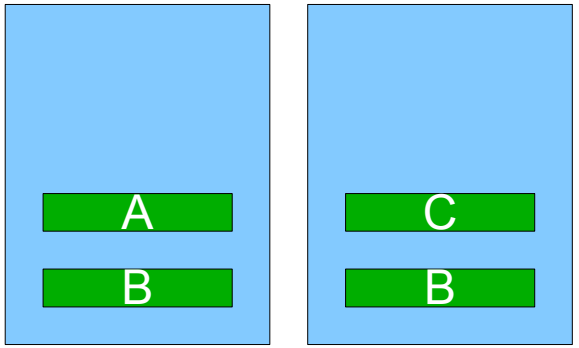
Řazení instancí (Oracle) 2.

```
CREATE TYPE t_osoba AS OBJECT (
  jmeno     VARCHAR2(30),
  prijmeni  VARCHAR2(30),
  narozeni  DATE,
  MEMBER FUNCTION vek RETURN INTEGER,
  MAP MEMBER FUNCTION weigth RETURN NUMBER
);
```

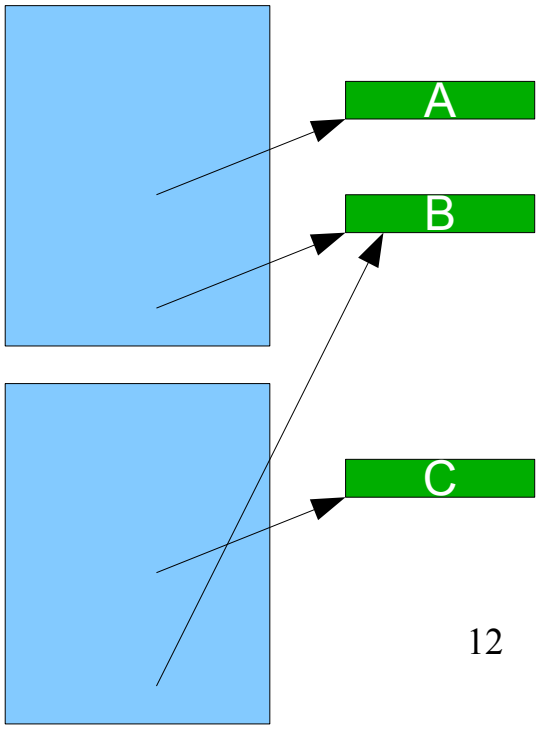
```
CREATE TYPE BODY t_osoba AS
   ...
   MAP MEMBER FUNCTION weight RETURN NUMBER IS
   BEGIN
      RETURN SELF.vek();
   END;
END;
```

Hnízdění vs. reference

- hnízdění (nested model)
  - jednodušší
  - možnost duplikace (redundance)



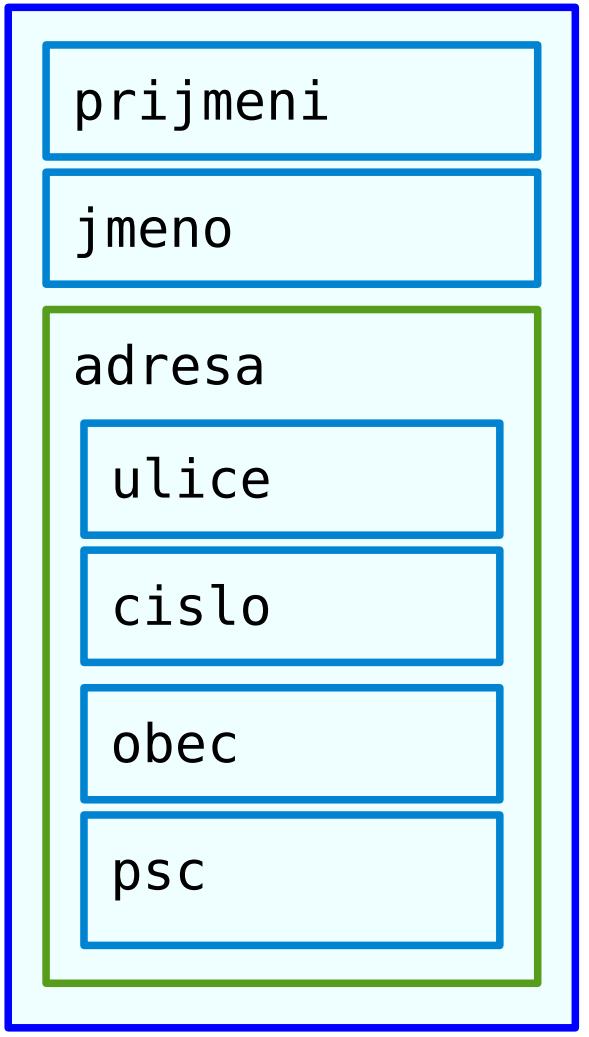
- reference (ukazatele)
  - sdílení objektů
    - nedochází ke zbytečnému kopírování dat
    - změny se provádí na jednom místě



Příklad UDT a hnízdění 1/2

```
CREATE TYPE t_adresa AS OBJECT (
  ulice      VARCHAR2(30),
  cislo      NUMBER(3),
  obec       VARCHAR2(30),
  psc        NUMBER(5)
);

CREATE TABLE obyvatele (
  prijmeni   VARCHAR2(30),
  jmeno      VARCHAR2(30),
  adresa     t_adresa
);
```



Příklad hnízdění 2/2

- insert, vytvoření inst. implicitním konstruktorem

```
INSERT INTO obyvatele (jmeno, prijmeni, adresa)
VALUES ('Pavel', 'Kříž',
        t_adresa('Lipová', 123, 'Stěžery',
50321));
```

- select, nutnost použít alias (o)!

```
SELECT o.jmeno AS jmeno, o.prijmeni AS prijmeni,
       o.adresa.obec AS obec, o.adresa.psc AS psc
FROM  obyvatele o;
```

- update
  - plně kvalifikované názvy atributů: `SET o.adresa.obec = ...`
  - nebo záměna celé adresy: `SET adresa = t_adresa(...)`

UDT jako typ řádku

- SQL standard i Oracle:

```
CREATE TABLE <table name> OF <type name>;
```

- vznikne „objektová tabulka“
  - každý řádek tabulky reprezentuje objekt (instanci)

- příklad:

```
CREATE TABLE osoby OF t_osoba;
```

  - (vycházíme z typu *t_osoba* z předešlých slidů)

Příklad: „objektová tabulka“ (Oracle)

- insert, nic zvláštního

```
INSERT INTO osoby (jmeno, prijmeni, naroz)
VALUES ('Pavel', 'Kříž', TO_DATE('1.1.1980', 'DD.MM.RRRR'));
```

- select, nic zvláštního, pokud chci volat metody, musím použít alias a za metodou vždy závorky

```
SELECT  o.prijmeni, o.jmeno, o.vek() AS vek
  FROM  osoby o;
```

Příklad: použití v PL/SQL

```
DECLARE
  clovek  t_osoba;
  c_ref   REF t_osoba;

BEGIN
  clovek := t_osoba('Pavel', 'Kříž',
                    TO_DATE('1.1.1980', 'DD.MM.RRRR'));

  DBMS_OUTPUT.PUT_LINE(clovek.vek());

  INSERT INTO osoby o VALUES(clovek)
    RETURNING REF(o) INTO c_ref;

  UPDATE osoby o SET jmeno = 'Jan'
   WHERE REF(o) = c_ref;

END;
```

Práce s atributy v SQL:1999

- v SQL:1999 má každý atribut tzv. mutator (setter) a generator (observer, getter)

```
SELECT o.prijmeni(), o.jmeno() FROM osoby o;
```

Podtypy (dědičnost)

- předek musí být NOT FINAL

```
CREATE TYPE t_urednik UNDER t_zamestnanec (
  prepazka NUMBER
)
```

Kolekce (Oracle)

- VARYING ARRAY (VARRAY)
  - uspořádané pole s definovanou maximální velikostí
  - uloženo v rámci řádku
- NESTED TABLE
  - neuspořádaná neohraničená kolekce prvků
  - uložena v další fyzické tabulce

Kolekce – VARRAY (Oracle)

- VARRAY (ekvivalent ARRAY z SQL:1999)
  - uspořádané pole s definovanou maximální velikostí

```
CREATE TYPE <typename> AS VARRAY(<max>) OF <element-type>
```

  - příklad:

```
CREATE TYPE t_tel_seznam AS VARRAY(5) OF VARCHAR2(30);
CREATE TYPE t_potomci AS VARRAY(10) OF t_person;
```

Př.: Kolekce – VARRAY (Oracle)

```
CREATE TABLE personal (
  jmeno    VARCHAR2(30),
  prijmeni VARCHAR2(30),
  telefony t_tel_seznam);
```

- insert

```
INSERT INTO personal (jmeno, prijmeni, telefony)
  VALUES ( 'Josef', 'Chroustal',
           t_tel_seznam('123', '124', '125'));
```

- select – jednotlivé záznamy z kolekce lze získat pomocí „přetypování“ na tabulku (*unnesting*)

```
SELECT p.*, t.* FROM personal p, TABLE(p.telefony) t;
```

- update – k VARRAY lze vždy přistupovat pouze jako k celku, nelze jednoduše přidávat či mazat

Kolekce v PL/SQL obecně

- metody kolekcí
  - EXISTS(k) – TRUE, pokud existuje k-tý prvek pole
  - COUNT – počet existujících prvků v poli
  - LIMIT – maximální možný počet prvků v poli
  - FIRST, LAST – indexy prvního a posledního ex. prvku
  - PRIOR(i), NEXT(i) – indexy předchozího a následujícího prvku vzhledem k i-tému
  - EXTEND[(n[,i])] – přidá 1 (n) prázdných prvků, resp. kopií i-tého
  - TRIM[(n)] – odebere 1 (n) prvků z konce pole

Příklad: VARRAY v PL/SQL

```
DECLARE
  p  personal%ROWTYPE;
BEGIN
  SELECT * INTO p
    FROM personal
   WHERE prijmeni = 'Chroustal';

   DBMS_OUTPUT.PUT_LINE('Pocet telefonu: '||p.telefony.count);

   DBMS_OUTPUT.PUT_LINE('Cisla :');
   FOR i IN 1..p.telefony.count LOOP
     DBMS_OUTPUT.PUT_LINE(p.telefony(i));
   END LOOP;
END;
```

Kolekce – NESTED TABLE (Oracle)

- hnízděná (vnořená) tabulka
  - neuspořádaná neohraničená kolekce prvků

```
CREATE TYPE <typename> AS TABLE OF <element-type>;
CREATE TABLE <parent-table> (
  ...,
  <colectionname> <typename>
) NESTED TABLE <colectionname> STORE AS <phys-table>;
```

```
CREATE TYPE t_zamestnanec AS OBJECT (
  jmeno    VARCHAR2(30),
  plat     NUMBER(5)
);
/
CREATE TYPE t_zamestnanci AS TABLE OF t_zamestnanec;
/
CREATE TABLE katedry (
  cislo_kat   NUMBER(5),
  nazev       VARCHAR2(50),
  zamestnanci t_zamestnanci
) NESTED TABLE zamestnanci STORE AS ntab_zamestnanci;
```

Př.:Kolekce – NESTED TABLE (Oracle)

- fyzická tabulka (ntab_zamestnanci) je vidět v katalogu (datovém slovníku), ale není přístupná pro DML
- pracuje se s ní podobně jako s VARRAY, s tím že jsou možné aktualizace „po řádcích“

- insert – stejně jako u VARRAY

```
INSERT INTO katedry (cislo_kat, nazev, zamestnanci)
  VALUES (1, 'KIKM',
          t_zamestnanci(
            t_zamestnanec('Pavel Kříž', 12345),
            t_zamestnanec('Tomáš Kozel', 12345),
            t_zamestnanec('Filip Malý', 12345)));
```

Př.:Kolekce – NESTED TABLE (Oracle)

- select – opět *unnesting* jako u VARRAY

```
SELECT  z.*
  FROM  katedry k, TABLE(k.zamestnanci) z
 WHERE  k.cislo_kat = 1;
```

- update

```
INSERT INTO TABLE(SELECT zamestnanci
                  FROM katedry WHERE cislo_kat = 1)
  VALUES ('Pavel Pražák', 12345);

UPDATE TABLE(SELECT zamestnanci
             FROM katedry WHERE cislo_kat = 1)
  SET plat = 99999 WHERE jmeno like 'Pavel%';

UPDATE TABLE(SELECT zamestnanci
             FROM katedry WHERE cislo_kat = 1) z
  SET z = t_zamestnanec('Martin Kocour', 1234)
WHERE z.jmeno = 'Tomáš Kozel';
```

Reference

- místo hnízděného typu *t_adresa* lze použít referenční typ (ukazatel) *REF t_adresa*

- „funkce“ REF aplikovaná na instanci („záznam“) objektové tabulky nebo view, vrací **odkaz** (ukazatel) na instanci

```
SELECT REF(p) FROM person_obj_table p
  WHERE p.idno = 12;
```

Příklad: Reference 1/2

```
CREATE TYPE t_kancelar AS OBJECT (
  zkratka  VARCHAR2(3),
  nazev    VARCHAR2(50));

CREATE TABLE kancelare OF t_kancelar;

CREATE TYPE t_nabytek AS OBJECT (
  nazev       VARCHAR2(30),
  ev_cislo    NUMBER(5),
  pracoviste  REF t_kancelar);
```

```
CREATE TABLE nabytek OF t_nabytek;
```

omezení reference na typ t_pracoviste

```
CREATE TABLE nabytek OF t_nabytek
(SCOPE FOR (pracoviste) IS kancelare);
```

omezení reference na tabulku pracoviste

```
CREATE TABLE nabytek OF t_nabytek
(FOREIGN KEY (pracoviste) REFERENCES kancelare);
```

omezení reference na tabulku pracoviste + FK IO

Příklad: Reference 2/2

- insert – „funkce“ REF vrátí ukazatel na objekt

```
INSERT INTO nabytek (nazev, ev_cislo, pracoviste)
  VALUES ('Židle', 555,
          (SELECT REF(k) FROM kancelare k
                         WHERE k.zkratka = 'KIKM'));
```

- select
  - explicitní dereference pomocí „funkce“ DEREF

```
SELECT n.nazev, n.ev_cislo, DEREF(n.pracoviste).zkratka
  FROM nabytek n;
```

  - implicitní dereference pomocí operátoru „*tečka*“

```
SELECT n.nazev, n.ev_cislo, n.pracoviste.zkratka
  FROM nabytek n;
```

Reference – ukazatele

- pokud nezajistíme referenční integritu (FOREIGN KEY), je možné „ztratit“ odkazované objekty, ukazatele pak ukazují na „neexistující objekty“
  - seznam „slepých ukazatelů“ lze získat pomocí predikátu IS DANGLING

```
SELECT p.nazev
  FROM nabytek p
 WHERE p.pracoviste IS DANGLING;
```

- dereference v SQL:1999
  - místo tečkové notace (která se syntakticky překrývá s přístupem k „položkám v záznamu“) je použito **->**

Reference – vztahy

- One-to-one
  - reference (REF)
  - hnízděný objekt
- Many-to-one
  - reference (REF)
  - (hnízdění)
- Many-to-many vztah
  - obvykle realizován jako nested table of references
  - (nebo stejně jako u cizích klíčů, jen se místo nich použijí reference)

Objektové predikáty a operátory

- <*objekt*> IS OF <*typ*>
  - objekt je instance typu <*typ*> nebo jeho potomka
- <*objekt*> IS OF (NOLY <*typ*>)
  - objekt je instance typu <*typ*>
- TREAT (<*objekt*> AS <*typ*>)
  - přetypování

Objektové tabulky vs. pohledy

- objektové tabulky
  - každý řádek tabulky reprezentuje objekt (instanci)
  - již samotná tabulka nese informaci, že používáme objektový model
- objektové pohledy
  - umožňují vytvořit objektovou nadstavbu nad stávajícím relačním modelem, bez nutnosti do něj zasahovat

```
CREATE VIEW OF <type> AS SELECT …
CREATE VIEW OF <type> AS SELECT <type>(…) FROM …;
```

```
CREATE VIEW ov_zamestnanci OF t_osoba WITH OBJECT OID
(jmeno,prijmeni) AS SELECT
t_osoba(first_name,last_name,NULL) FROM employees;
```

Oracle Spatial

- Knihovna funkcí, procedur a datových typů
- Určena pro efektivní ukládání a zpracování prostorových dat
- Aplikace v oblasti geografických informačních systémů (GIS)
- Hlavní pilíř: typ SDO_GEOMETRY
  - Může reprezentovat různé geom. objekty (bod, obdélník, polygon,...)
- Funkce: SDO_CONTAINS, SDO_WITHIN_DISTANCE, SDO_NN,...
- Příklad:

```
select
    ST.NAME, UT.UNIVERSITY_NAME
from
    STATE_TABLE ST
    JOIN UNIVERSITY_TABLE UT
      ON SDO_CONTAINS(ST.GEOM, UT.GEOM) = 'TRUE'
```

Zdroje

- Connoly, T., Begg, C.: Database Systems, Pearson Education, 2005
- http://service.felk.cvut.cz/courses/X36SQL/Prednasky/SQL-OR.pdf
- http://download.oracle.com/docs/cd/B19306_01/appdev.102/b14260/toc.htm
- http://download.oracle.com/docs/cd/B19306_01/appdev.102/b14261/objects.htm
- http://www.kiv.zcu.cz/~zima/vyuka/db2/cviceni12.html
- http://www.odbms.org/download/001.02%20Ullman%20CS145%20Object-Relational%20DBMS%20Fall%202004.ppt
- http://computing.unn.ac.uk/staff/cgma2/cg096/Week5/ORNRdatabases.ppt