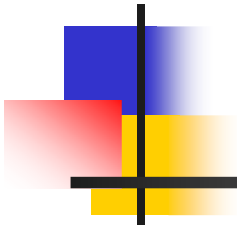


# CURS 7



## Substituirea tipurilor Funcții și predicate utile în lucrul cu obiectele



# Substituirea tipurilor în ierarhia de tipuri

- În ierarhia tipurilor, subtipurile sunt variante ale tipului de bază – rădăcina
- Ex. Tipurile *t\_angajat* și *t\_student* sunt variante ale tipului de bază *Persoana\_t*
- abilitatea de a selecta toate instanțele unui tip de bază și de a întoarce nu numai obiectele declarate în acest tip ci și pe acelea declarate ca subtipuri se numește *substitutabilitate*
- un supertip este substituibil dacă unul din subtipurile sale îl poate substitui sau poate fi pus în locul său într-un slot (o variabilă, o coloană, etc) al cărui tip declarat este supertipul
- în general tipurile sunt substituibile



# Substituirea atributelor

---

- Atributele, elementele colecție și REF-urile sunt substituibile
- Fie *tip1* un obiect tip:
  - Atributele definite ca REF *tip1* pot păstra o referință către o instanță a tipului *tip1* sau către orice subtip al tipului *tip1*
  - Atributele definite ca având tipul *tip1* pot conține o instanță a tipului *tip1* sau a oricăruia din subtipurile sale
  - Colecțiile de elemente de tip *tip1* pot păstra instanțe ale tipului *tip1* și instanțe ale oricărui subtip al lui *tip1*



# Exemple

- Atributul autor este substituibil în tipul *t\_carte* definit astfel:

```
CREATE TYPE t_carte AS OBJECT (  
titlu    VARCHAR2(30),  
autor    t_persoana);  /* substituibil */
```

- O instanță *t\_carte* poate fi creată prin specificarea unui șir pentru titlu și un autor de tip *t\_persoană* sau aparținând oricărui subtip al acestui tip.
- În următorul exemplu se specifică un autor de tip *t\_student*:

```
t_carte('Tutorial Oracle', t_student(2610723452213, 'Marius Buraga', 'Suceava', NULL))
```

- În general, attributele pot fi accesate prin intermediul notației *dot*.
- Attributele unui subtip al unui rând sau a unei coloane declarate ca tip pot fi accesate cu ajutorul funcției *TREAT*.
- Într-un obiect vedere *v\_carte* de tip *t\_carte* se poate utiliza *TREAT* pentru a atribui valoarea *cnp* din *autor* de tip *t\_student*. (coloana *autor* este de tip *t\_persoană*)

```
SELECT TREAT(author AS t_student).cnp FROM v_carte
```



## Substituirea unui rând sau a unei coloane

---

- Obiectele de tip coloana precum și obiectele de tip rând din obiecte tabel sau vedere sunt substituibile.
  - O coloană sau un rând definite ca având un tip T pot conține instanțe ale lui T sau ale oricăruia dintre subtipurile sale.



# Exemplu 1

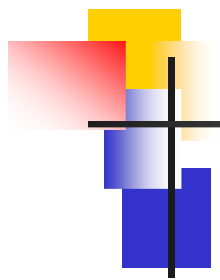
- Fie cele trei tipuri declarate anterior: *t\_persoana*, *t\_student* și *t\_licență*
- Un tabel de tip *t\_persoana* poate conține rânduri având oricare din tipurile enumerate
- Inserarea unei instanțe pentru un tip dat se face prin utilizarea constructorului corespunzător în clauza **VALUE** a instrucțiunii **INSERT**:

```
CREATE TABLE persoana OF t_persoana;
```

```
INSERT INTO persoana VALUES (t_persoana(1243, 'Mircea', 'Visinilor  
3'));
```

```
INSERT INTO persoana VALUES (t_student(3456, 'Ileana', 'Narciselor  
34', 12));
```

```
INSERT INTO persoana VALUES (t_licenta(5678, 'Tim', 'Visinilor 2', 13,  
'ELECTRONICA'));
```



## Exemplu 2

- In tabele sau vederi relaționale o coloană substituibilă de tipul *t\_persoana* poate conține instanțe din toate cele trei tipuri

```
CREATE TABLE carte (  
titlu varchar2(100),  
autor Person_typ);
```

```
INSERT INTO carte VALUES(Autobiografie', t_persoana(1243, 'Bob',  
    'Visinilor 3'));
```

```
INSERT INTO carte VALUES('Oracle avansat', t_student(3456, 'Joe',  
    'Visinilor 3',12));
```

```
INSERT INTO carte VALUES('Introducere in Oracle', t_licenta(5678,  
    'Tim', 'Visinilor 3',13, 'Calculatoare'));
```



# Subtipuri care au attribute supertip

Un subtip poate avea un atribut care este un supertip.

■ Ex.:

```
CREATE TYPE t_student UNDER t_persoana (... , tutor  
t_persoana);
```

- Coloanele de acest tip nu sunt substituibile.
- de asemenea, un subtip poate avea attribute de tip colecție ale  
căror tipuri pentru elemente pot fi supertipuri
  - Coloanele de acest tip NU sunt substituibile
- totuși se pot defini coloane substituibile de subtipuri care au  
attribute REF care referă supertipuri



# Funcții și predicate utile în lucrul cu obiectele



---

- VALUE
- REF
- Deref
- TREAT
- IS OF *tip*
- SYS\_TYPEID



# VALUE

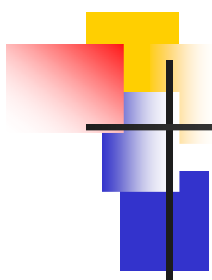
- într-o instrucțiune SQL, funcția VALUE are ca argument o variabilă de corelație (alias de tabel) pentru un obiect tabel sau obiect vedere și întoarce instanțele obiectului corespunzătoare rândurilor tabelului sau vederii.

**SELECT VALUE(p) FROM persoana p WHERE p.num = "X";**

- Întoarce toate persoanele al căror nume este X
- Funcția VALUE poate returna instanțe ale tipului declarat pentru rând sau ale oricăreia dintre subtipurile sale

**SELECT VALUE(p) FROM Person\_v p;**

- Întoarce toate persoanele, inclusiv studenții și angajații din obiectul vedere *Person\_v* de tip *t\_persoana*



# VALUE

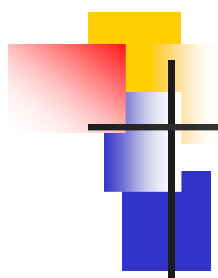
- Pentru a regăsi NUMAI persoane – adică instanțe al căror tip specific este t\_persoana, se utilizează cuvântul cheie ONLY

**SELECT VALUE(p) FROM ONLY(Person\_v) p;**

- VALUE poate fi folosit pentru a returna instanțe ale obiectelor rînd pentru actualizare:

## **UPDATE TABLE**

**(SELECT e.proiecte FROM employees e WHERE e.employee\_id = 100) p**  
**SET VALUE(p) = t\_proiect(1, 'Project Pluto')**  
**WHERE p.pno = 1;**



# REF

- Intr-o instrucțiune SQL funcția REF ia ca argument un nume de corelație pentru un obiect tabel sau vedere și întoarce o referință (REF) către un obiect instanța din tabel sau vedere
- funcția REF poate întoarce referințe către obiecte declarate de tip tabel/vedere sau oricare din subtipurile acestora.

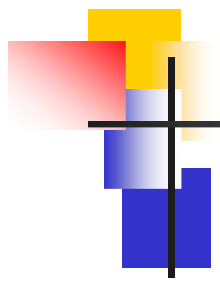
Exemple:

```
SELECT REF(p) FROM Person_v p;
```

- întoarce referințe către toate persoanele inclusiv studenți, samd

```
SELECT REF(p) FROM Person_v p WHERE p.id = 0001 ;
```

- întoarce o referință( REF) către persoana (sau studentul) al cărui atribut ID este 0001:



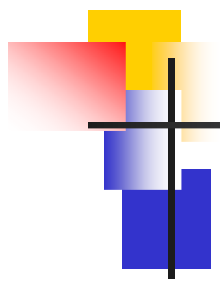
# DEREF

- Functia Deref folosita într-o interogare SQL intoarce instanta obiectului corespunzătoare referinței
  - Aceasta poate fi de tipul declarat pentru referinta sau oricare din subtipurile acestuia.

Exemplu:

```
SELECT Deref(Ref(p)) FROM Person_v p;
```

- Instrucțiunea întoarce obiectele *persoana* din obiectul vedere *Person\_v*, inclusiv persoanele care sunt studenți



# TREAT

- Funcția TREAT încearcă să modifice tipul declarat al unei expresii într-un alt tip specificat- de regulă un subtip al tipului declarat
  - Funcția încearcă să trateze (TREAT) instanța unui supertip ca instanță a unui subtip
- Ex. sa trateze o persoana ca pe un student. Acest lucru este posibil, pentru un anumit caz numai dacă persoana respectivă este student (sau în ciclul de licență). În acest caz persoana este returnată ca student cu atributele și metodele corespunzătoare tipului student. Dacă persoana nu este student funcția întoarce NULL
- TREAT este util în următoarele situații:
  - Pentru atribuiri *narrowing* – se modifică tipul unei expresii astfel încât aceasta să poată fi asignată unei variabile declarată cu un tip mai specializat în ierarhie
    - pentru a atribui o valoare corespunzătoare unui supertip unui subtip.
  - Pentru a accesa atribute sau metode ale unui subtip al tipului declarat pentru un rând sau o coloană



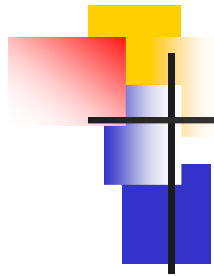
# TREAT- Exemple 1

```
UPDATE T set angcol = TREAT(perscol AS t_angajat);
```

- TREAT este utilizat într-o asignare – o coloana de tip *persoana* este atribuită unei coloane de tip *angajat* la actualizarea unui tip
  - Pentru fiecare rând în *perscol*, TREAT întoarce un tip *angajat* sau *NULL*.

```
SELECT TREAT(VALUE(p) AS t_student) FROM Person_v p;
```

- TREAT întoarce toate (și numai) instanțele *t\_student* din obiectul vedere *Person\_v*, de tip *t\_persoana*, care este un supertip pentru *t\_student*.
  - Instrucțiunea folosește TREAT pentru a modifica tipul lui *p* de la *t\_persoana* la *t\_student*.



## TREAT Example 2

- Cea mai importantă utilizare pentru TREAT- posibilitatea de a accesa attributele sau metodele unui subtip al unui tip declarat pentru un rand sau o coloana.

```
SELECT nume, TREAT(VALUE(p) AS t_student).codfac codfac FROM  
persoana p;
```

```
NUME CODFAC
```

```
----
```

```
Bob      null
```

```
Joe      12
```

- A fost afișat atributul *codfac* pentru toate persoanele care au acest atribut (adică studenții și studenții din ciclul de licență) Pentru persoanele care nu au calitatea de student se întoarce NULL.

- Formularea

```
SELECT nume, VALUE(p).codfac codfac FROM persoana p;
```

- este eronată deoarece *codfac* este un atribut al tipului *t\_student* dar nu și al tipului *t\_persoana*, care este tipul declarat pentru tabelul *persoana*





## TREAT Example 3

- Un obiect tabel sau coloană substituibilă de tip T are câte o coloană ascunsă pentru toate attributele tuturor subtipurilor definite pentru acest tip T.
  - aceste coloane nu apar în rezultatul comenzii DESCRIBE dar conțin datele atributelor subtipului.
  - TREAT permite accesul la aceste coloane ascunse.
- Următorul exemplu arată cum poate fi folosit TREAT pentru a accesa metodele unui subtip:

```
SELECT nume, TREAT(VALUE(p) AS t_student).metoda1() nume_col  
FROM persons p;
```



# IS OF *tip*

- IS OF *tip* este un predicat de selecție care testează instanțele obiectelor relativ la nivelul de specializare al tipului lor.
- Se poate folosi doar în SQL, nu și în PL/SQL.

**SELECT VALUE(p) FROM persoana p WHERE VALUE(p) IS OF  
(t\_student);**

VALUE(p)

-----

t\_student(123456781, 'Mircea', 'Visinilor 2', 12)

t\_licenta(12345671876, 'Tim', 'Alexandru cel Bun 12', 13, 'Calculatoare')

- Interogarea regăsește toate instanțele student (inclusiv toate subtipurile acestuia) din tabelul *persoana*
- Pentru orice obiect care nu este în tipul specificat sau subtipurile acestuia IS OF întoarce FALSE



# IS OF *tip*

- Dacă se doresc excluse subtipurile se folosește cuvântul cheie ONLY, care va determina ca IS OF să întoarcă FALSE pentru toate tipurile cu excepția celui specificat în comandă

```
SELECT c.titlu titlu, c.autor autor FROM carte c WHERE c.autor IS OF (ONLY  
t_student);
```

TITLU	AUTOR
-------	-------

-----

-----

Oracle	t_student (12345617891, 'Alex', 'Narciselor 2', 12)
--------	---

- sunt excluse cartile scrise de alte tipuri de persoane decât cele aparținând tipului *t\_student*.

```
SELECT REF(p) FROM Person_v p WHERE VALUE(p) IS OF (t_angajat,  
t_student);
```

- În obiectul vedere Person\_v, sunt testate instanțele care cințin persoane, angajați și studenți și se întorc referințele pentru obiectele care sunt exclusiv de tipurile menționate și subtipurilor acestora, dacă există



## IS OF *tip*

```
SELECT TREAT(VALUE(p) AS t_student) FROM Person_v p
WHERE VALUE(p) IS OF(ONLY t_student);
```

- Întoarce numai studenții care au *cel mai specializat tip* t\_student. Sunt excluse orice obiecte care aparțin subtipurilor lui t\_student.
  - Funcția TREAT este folosită pentru a converti obiectele student la tipul t\_student pornind de la tipul declarat al vederii care este t\_persoana
- Pentru a testa tipul unui obiect referit de un atribut REF se utilizează funcția Deref pentru a dereferența referința înainte de a testa predicatul IS OF *tip*

```
SELECT * FROM view WHERE Deref(PersRefCol) IS OF
(t_student);
```

- dacă *PersRefCol* este declarata ca **REF t\_person** se pot obține numai rândurile pentru studenți



# SYS\_TYPEID

- Funcția SYS\_TYPEID poate fi utilizată într-o interogare pentru a întoarce valoarea **typeid** a *celui mai specific tip al instanței obiectului* folosit ca argument.
- ***Cel mai specific tip*** al instanței unui obiect este tipul căruia îi aparține instanța care este cel mai depărtat de tipul rădăcină
  - Ex. dacă Tim este student în ciclul de licență el este de asemenea student și persoana. Cel mai specific tip este însă *t\_licență*.
- Funcția întoarce identificatorii de tip din colana **type discriminant** care este asociată cu fiecare coloană substituibilă
  - Pentru tipul rădăcină funcția întoarce null
- Sintaxa funcției:

SYS\_TYPEID( *object\_type\_value* )



# SYS\_TYPEID

---

SYS\_TYPEID poate fi utilizată numai cu argumente de tip obiect.

- Scopul – posibilitatea de a construi un index al coloanei ascunse *type discriminant*
- Toate tipurile care se găsesc într-o ierarhie au asignate un identificator de tip *typeid* non- null care este unic în ierarhie.
  - Tipurile care nu aparțin unei ierarhii au în aceasta coloana NULL.