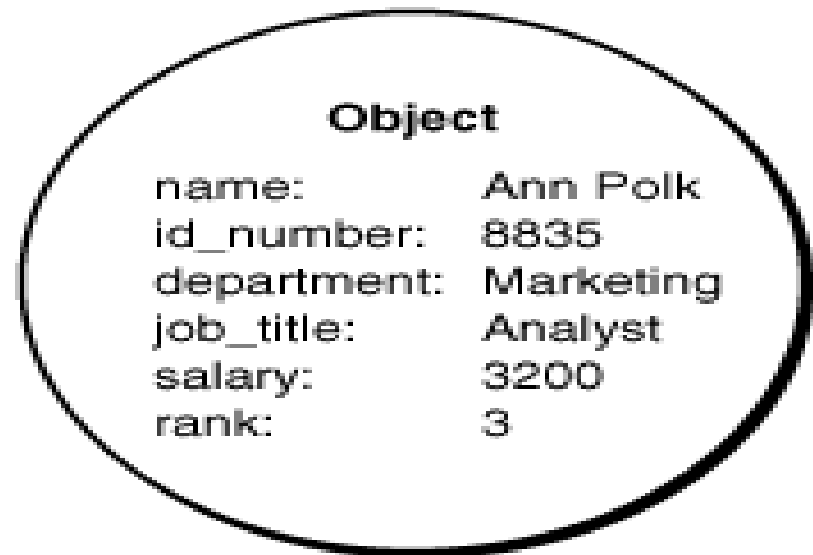
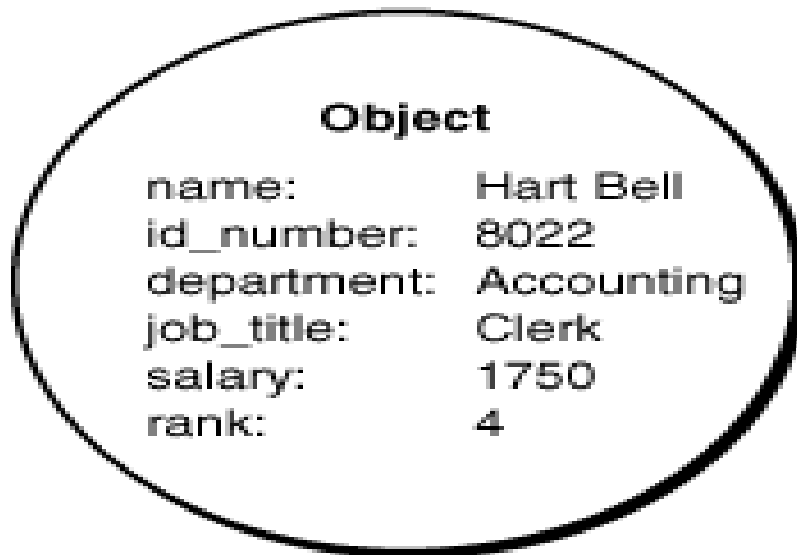
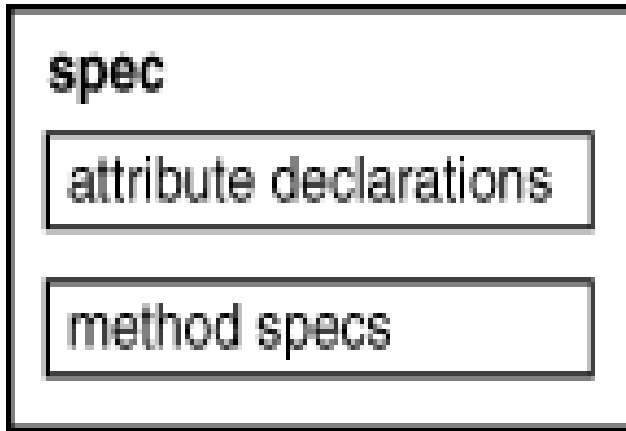


Object Type and Objects (Instances)

Object Type <i>Employee</i>	
Attributes name id_number department job_title salary rank	Methods calculate_bonus change_dept change_job_title change_salary change_rank



- Structura unui tip obiect



public interface



private implementation

Limbajul PL/SQL- Obiecte

- ```
CREATE TYPE tip_obiect AS OBJECT(
 var1 tip1,
 var2 tip2,
 ...
 MEMBER FUNCTION f1(lista1) RETURN tipf1,
 MEMBER FUNCTION f2(lista2) RETURN tipf2,
 ...
 MEMBER PROCEDURE p1(lista21),
 MEMBER PROCEDURE p2(lista22),
 ...
);
CREATE TYPE BODY tip_obiect AS
MEMBER FUNCTION f1(lista1) RETURN tipf1 IS
BEGIN
... END f1; ... END;
```

# Limbajul PL/SQL- Obiecte

- Componentele unui tip obiect

Atributele: nume si tip.

Tip diferit de: LONG, LONG RAW, NCHAR, NCLOB, NVARCHAR2, MLSLABEL, ROWID, BINARY\_INTEGER, RECORD, REF CURSOR, %TYPE, %ROWTYPE, tipuri definite in pachete.

- Nu se poate initializa un atribut prin asignare sau DEFAULT.
- Nu putem impune NOT NULL pe un atribut.
- Obiectele pot fi memorate in tabele pe care se pot impune NOT NULL>

Metodele = subprograme precedate de MEMBER sau STATIC.

Fiecare metoda are o specificare si un corp.

Intr-un tip obiect, metodele pot referi atribute si alte metode fara a folosi calificarea.

# Limbajul PL/SQL- Obiecte

- Parametrul SELF:

Orice metoda a unui tip obiect accepta o instanta a acelui tip drept prim parametru. Numele acestui parametru este SELF. Indiferent daca SELF este sau nu declarat, el este mereu primul parametru de apel al metodei.

In functii membru SELF nedeclarat este de tip IN, in proceduri nedeclarat el este de tip IN OUT. Nu se specifica tipul lui SELF.

In corpul metodei SELF specifica obiectul pentru care a fost apelata metoda.

Metodele statice nu accepta sau refera SELF, ele se adreseaza tipului obiect si nu unui obiect.

```
CREATE FUNCTION cmmdc(x INTEGER, y INTEGER) RETURN
INTEGER AS
```

```
 z INTEGER;
```

```
 BEGIN
```

```
 IF(y<=x) AND (x MOD y =0) THEN z:=y;
```

```
 ELSIF x<y THEN z:=cmmdc(y,x); END IF; RETURN z; END; 162
```

# Limbajul PL/SQL- Obiecte

```
CREATE TYPE Rational AS OBJECT(
 numarator INTEGER,
 numitor INTEGER,
 MEMBER PROCEDURE normalizare,
 ...) /
```

```
CREATE TYPE BODY Rational AS
 MEMBER PROCEDURE IS
 g INTEGER;
 BEGIN
 g:=cmmdc(SELF.numarator,SELF.numitor); --sau
 g:=cmmdc(numarator,numitor);
 numarator:=numarator/g;
 numitor:=numitor/g;
 END normalizare;
 ... END;
```

# Limbajul PL/SQL- Obiecte

- Supraincarcarea.

Metodele de acelasi tip(proceduri sau functii) pot fi supraincarcate.

Nu se pot supraincarca doua metode daca parametrii formali difera numai in modul parametru.

Nu putem supraincarca doua functii membru ce difera numai in tipul returnat.

- Metodele ORDER si MAP

Tipurile de date scalare (CHAR, REAL,..) au o ordine predefinita, ce permite compararea lor. Instantele obiectelor nu au o ordine predefinita. Vom folosi metoda MAP.

```
CREATE TYPE Rational AS OBJECT (
 numarator INTEGER,
 numitor INTEGER,
 MAP MEMBER FUNCTION conversie RETURN REAL;
...);
```

# Limbajul PL/SQL- Obiecte

```
CREATE TYPE BODY Rational AS
```

```
MAP MEMBER FUNCTION conversie RETURN REAL IS
```

```
-- conversie obiect Rational la un numar REAL
```

```
BEGIN
```

```
 RETURN numarator/numitor;
```

```
END conversie;
```

```
...
```

```
END;
```

Ordonarile sunt necesare in realizarea clauzelor DISTINCT, GROUP BY, ORDER BY.

Un tip obiect poate contine numai o metoda MAP, cu tipul returnat unul din: DATE, NUMBER, VARCHAR2, CHARACTER, REAL.

Putem folosi metoda ORDER, cu doi parametri: primul SELF, al doilea de acelasi tip.

# Limbajul PL/SQL- Obiecte

O comparare a doua obiecte implica automat invocarea metodei definite de ORDER.

```
CREATE TYPE Student AS OBJECT (
 marca NUMBER,
 nume VARCHAR2(15),
 adresa VARCHAR2(20),
 ORDER MEMBER FUNCTION comparare (par2 Student) RETURN
 INTEGER);

CREATE TYPE BODY Student AS
 ORDER MEMBER FUNCTION comparare (par2 Student)
 RETURN INTEGER) IS
 BEGIN
 IF marca<par2.marca THEN RETURN -1;
 ELSIF marca > par2.marca THEN RETURN 1;
 ELSE RETURN 0; END IF; END; END;
```

# Limbajul PL/SQL- Obiecte

- Metoda constructor

Fiecare tip obiect are o metoda constructor, folosita pentru initializarea si returnarea unei instante de tipul respectiv. Parametrii formali ai constructorului au acelasi tip cu attributele obiectului.

Putem defini metode constructor proprii prin supraincercarea constructorului sistem sau definind noi functii cu signatura diferita.

Apelul constructorului trebuie facut explicit de catre program.

- Modificarea atributelor si metodelor unui tip existent:

`ALTER TYPE` pentru adaugare, modificare, stergere attribute,  
adaugare, stergere metode pentru un tip existent.

`ALTER TYPE tip_obiect`

`ADD ATTRIBUTE (nume1 tip1, ...) CASCADE;`

`ALTER TYPE tip_obiect`

`DROP ATTRIBUTE nume1,.. ;`

# Limbajul PL/SQL- Obiecte

- Declararea obiectelor

Dupa definirea tipului obiect, acesta poate fi utilizat sa declare obiecte in blocuri PL/SQL, subprograme, pachete. In bloc sau subprogram, la intrare se creaza obiecte, care se sterg la iesire. In pacete obiectele se instantiaza la prima referire si se sterg la sfirsitul sesiunii.

DECLARE

```
nume_obiect tip_obiect;
```

Putem declara obiectele ca parametri formali ai functiilor sau procedurilor:

```
PROCEDURE nume_p (par1 IN OUT tip_obiect) IS...
```

```
FUNCTION nume_f (lista) RETURN tip_obiect IS...
```

- Initializarea obiectelor:

Inainte de initializare un obiect este atomic null. Compararea unui obiect null cu altul produce NULL.

```
nume_ob1:=nume_ob2; nume_ob:=NULL;
```

```
nume_obiect tip_obiect:= constructor...
```

# Limbajul PL/SQL- Obiecte

In expresii attributele unui obiect neinitializat au valoarea NULL.

nume\_obiect.atribut:=expresie; cind nume\_obiect este neinitializat produce ACCES\_INTO\_NULL.

Apel de metoda cu obiect neinitializat este permis, SELF este NULL.

nume\_obiect neinitializat ca parametru formal IN – attributele NULL.

Cind OUT sau IN OUT apare exceptia daca se asigneaza.

Referinta la attribute: nume\_obiect.atribut

Apelul constructorului implica definirea de valori initiale atributelor (nu se poate folosi DEFAULT). Putem folosi notatia pozitionala.

Apelul metodelor:

nume\_obiect.metoda(a1,...); -- sau inlanturi de chemari de metode.

In instructiuni procedurale lista vida este optionala, dar in apel inlantuit trebuie dat () pentru functii.

o.f1().F2() prima trebuie sa returneze un obiect.

# Limbajul PL/SQL- Obiecte

- Tipuri obiecte imbricate

```
CREATE TYPE Adresa AS OBJECT (
 strada VARCHAR2(20), oras VARCHAR2(15), codpostal
 INTEGER)
```

```
/
```

```
CREATE TYPE persoana AS OBJECT (
 nume VARCHAR2(20), prenume VARCHAR2(20), datanasterii
 DATE, adresa_domiciliu Adresa, telefon VARCHAR2(12))
```

Un obiect are un identificator de obiect. Vom utiliza REF care este pointer la un obiect. Procesul –"sharing". Avantaje: nereplicare si modificarea unui obiect implica actualizarea referintelor.

```
CREATE TYPE persoana AS OBJECT (
 nume VARCHAR2(20), prenume VARCHAR2(20), datanasterii
 DATE, adresa_domiciliu Adresa, telefon VARCHAR2(12) ,
 tata REF persoana, mama REF persoana)
```

# Limbajul PL/SQL- Obiecte

Putem declara REF pentru variabile, parametri, cimpuri, attribute, variabile de intrare/iesire in instructiuni SQL.

Nu putem naviga prin aceste elemente REF:

nume\_referinta.atribut --nu poate specifica obiectul referit. DEREf

- Definitii forward de tip:

```
CREATE TYPE Emp AS OBJECT (
 nume VARCHAR2(15), dept REF departament, --incorect
 ...)
```

```
CREATE TYPE departament AS OBJECT (
 numar INTEGER, manager Emp,...)
```

```
CREATE TYPE departament; -- definitie forward –tip obiect
incomplet.
```

```
CREATE TYPE Emp AS OBJECT...
```

Putem folosi tip de obiect in CREATE TABLE, INSERT, nume de Metode in UPDATE

```
CREATE TABLE Tab1 OF Rational --tabela de obiecte
```