Dr. Rim Samia Kaabi

Héritage

1

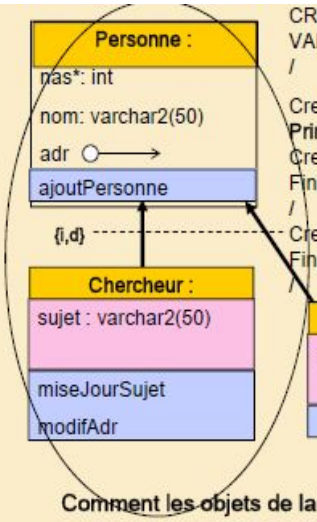
Comportement polymorphique dans une hiérarchie

•Classe Personnel: il est utile de faire une mise à jour de toutes les catégories d'employés en travaillant à la racine et en ayant accès a tous les employés
•Substitutionalité automatique
•Sauf:

```
CREATE TABLE employe_tech
(noET NUMBER, nomET poste ET char(1))
NOT SUBSTITUABLE AT ALL LEVELS;
```

2

Rangement des objets de cette hiérarchie d'héritage




```
CREATE TYPE personne_t as Object (nas NUMBER, nom
VARCHAR(20), adr  adr_t)  INSTANTIABLE   NOT FINAL
/
Create Table Personne of personne_t (Constraint pk_Personne
Primary key (nas)) ;
Create type chercheur_t  UNDER personne_t (sujet varchar2(50))
Final Instantiable;
/
Create type etudiant_t  UNDER personne_t (sujet varchar2(50))
Final Instantiable;
/
```

```
Create Table Chercheur of
chercheur_t (Constraint
pk_Chercheur Primary key (nas));
```

Comment les objets de la hiérarchie sont-ils rangés?

3

Rangement et substitution des types

1- Les objets sont rangés (par 3 clauses INSERT) dans leur table respective:

- les objets de type personne_t dans la table Personne
- les objets de type chercheur_t dans la table Chercheur
- les objets de type etudiant_t dans la table Etudiant

Trois tables distinctes et éventuellement dans des espaces de table distincts.

```
Insert into Personne  values (personne_t(….));
Insert into Chercheur values (chercheur_t (….);
Insert into Etudiant  values (etudiant_t( ….));
```

2- Les objets d'une hiérarchie peuvent être rangés dans une même table du type de la racine par une clause DML Insert:

-Tous les objets : personne, chercheur et etudiant cohabitent dans le classe Personne et donc dans le même *table-space*

```
Insert into Personne  values (personne_t(….));
Insert into Personne values (chercheur_t (….);
Insert into Personne values (etudiant_t( ….));
```

Sélection des objets de table à un niveau de la hiérarchie lorsqu'il ya substitution des types d'objets de table

•Fonction IS OF

Traite à un niveau de la hiérarchie les objets pour sélectionner que ceux des niveaux inférieurs de la hiérarchie (la racine étant le niveau 0)

Select value (p) from Personne p where value(p) IS OF (chercheur_t);

```
VALUE(P) (NAS, NOM)
CHERCHEUR_T(34, 'Ferron', 'physique')
```

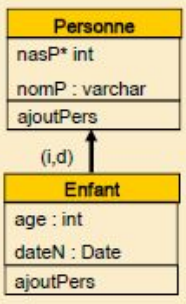
Select p.nom from Personne p where value(p) IS OF (chercheur_t);

•Le filtre se fait avec le type de l'objet p obtenu de la fonction value(p)

5

Héritage de type et non de table

Une table-objet typée avec un sous-type n'hérite pas des contraintes d'une table parent-objet typée avec un sur-type (dans la même hiérarchie).




```
Create table Personne of personne_t (
    Constraint  pk_personne Primary Key (nasP));

Create table Enfant of enfant_t (
---->   Constraint  pk_Enfant  Primary Key (nasP),
    Constraint c_age check(age < 16));
```

** Sans la contrainte de clé primaire dans la table Enfant permettrait d'ajouter jeux enfants (2 objets) ayant le même nasP.

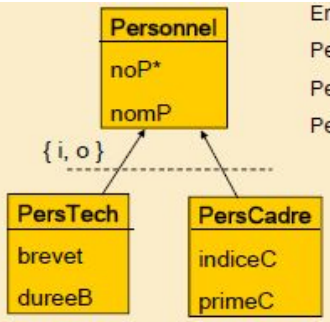
Cela est possible car la contrainte de clé primaire dans la table Personne n'est pas l'objet d'un héritage entre les tables.

6

Implémentation des contraintes de l'héritage

7

Héritage {i,o}



En relationel (avec 3 relations):

PersTech(*noP*, nomP* , brevet, dureeB)

PersCadre(*noP, nomP* , indiceC, primeC

Personnel (noP, nomP)

L'héritage des types est défini avec **UNDER** .

La table-objet Personnel stockera le personnel autre que technique et cadre. **Les 2 sous-tables peuvent représenter la même personne (même noP).**

```
Create Type personnel_t as Object(noP number, nomP varchar(50))
Instantiable Not Final /
Create Type persTech_t UNDER personnel_t ( brevet number, dureeB number(2)) Instantiable Final /
Create Type persCadre_t UNDER personnel_t (indiceC varchar(50), primeC int)) Instantiable Final /
```

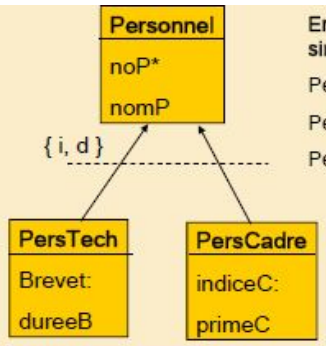
Héritage {i,o}: implémentation

```
Create Table Personnel of personnel_t (Constraint pk_Personnel Primary Key(noP));
Create Table PersTechnique of persTech_t;
Create Table PersCadre of persCadre_t;
```

- Lors d'un ajout d'une personne à la fois technique et cadre, la méthode doit faire deux insertions: une pour chaque classe
- Avantage de l'héritage de type: partage possible d'un type entre plusieurs tables

9

Héritage {i,d}



En relationel (3 relations): duplication des attributs pour simuler l'héritage

PersTech(***noP, nomP***, brevet, dureeB)

PersCadre(***noP, nomP***, indiceC, primeC)

Personnel (***noP, nomP***)

En Objet-Relationnel :

L'héritage des types est défini avec **UNDER** .

La table OR Personnel stockera le personnel qui est ni technique et ni cadre.

```
Create or Replace Type personnel_t as object(noP number, nomP varchar(50))
Instantiable Not Final /
Create Type persTech_t UNDER personnel_t ( brevet number, dureeB number(2))
Instantiable Final /
Create Type persCadre_t UNDER personnel_t (indiceC varchar(50), primeC int)
Instantiable Final /
```

Héritage {i,d}: implémentation

Create Table Personnel of personnel_t (Constraint pk_Personnel Primary Key(noP));

La contrainte de clé primaire doit être redéfinie car il n'y a pas d'héritage entre les table.

Create Table PersTech of persTech_t (constraint pk_PT primary key(noP));
Create Table PersCadre of persCadre_t (constraint pk_PC primary key(noP));

Insertion d'objets de structure différente par deux méthodes distinctes.

Exemple:

Insert into Personnel values (personnel_t(20, 'Marois'));
Insert into PersTech values (persTech_t(21, 'Gendron', 5, 20));

L'objet suivant **devrait être** interdit selon les contraintes de l'héritage {i, d}

Insert into PersCadre values (persCadre_t(21, 'Gendron', 25 , 2000));

11

Méthodes ajoutPT_id {i,d}

- ajoutPT_id doit vérifier qu'une personne insérée comme technicien ne le soit pas aussi comme cadre
- 2 objets ne pouvant pas avoir le même OID, il s'agit de personnes dont l'identifiant sémantique noP les distingue. La clé sert donc à cela dans une spécialisation d'héritage
- ajoutPT_id devra donc faire ce test avant d'ajouter un objet dans personnePT

Alter type persTech_t ADD Member Function ajoutPT_id return int cascade;

12

Body de la Méthode ajoutPT_id {i,d}

```
Create or replace type body persTech_t as
Member Function ajoutPT_id return int IS
v_nbc int;
Begin
/* aucune contrainte à différer */
Select count(*) into v_nbc From Personnel p Where p.noP = self.noP;
IF v_nbc = 1 Then return 0;
Else Begin
  Select count(*) into v_nbc from PersCadre pc Where pc.noP = self.noP;
  If v_nbc = 0 Then Begin
          Insert into PersTech values(self);
          Commit;
          return 1;
          End;
   Else  return 0;
 End IF;
End;
End IF;
End;   End;
/
```

13

Contrainte {c, d}: ajouterPT_cd

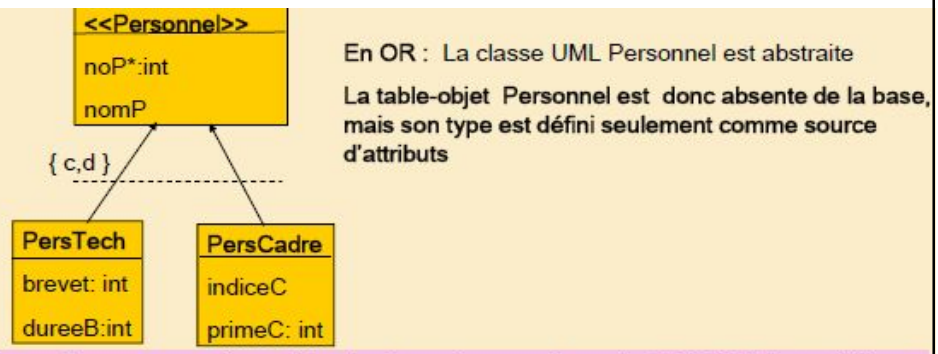
•Pour valider le disjoint, il faudra lors de l'ajout d'un technicien ou d'un cadre vérifier s'il n'est pas déjà présent à titre de membre du personnel (complete) et ensuite vérifier qu'il ne l'est pas aussi dans l'autre sous classe (disjoint)

•Puisque la classe Personnel est abstraite (Final et non intantiable), la 1 ère vérification n'a pas lieu d'être

```
Alter type persCadre_t ADD Member Function ajoutPC_cd return int CASCADE ;
Alter type persCadre_t ADD Member Function ajoutPT_cd return int CASCADE ;
```

14

Héritage {c, d}



En OR : La classe UML Personnel est abstraite

La table-objet Personnel est donc absente de la base, mais son type est défini seulement comme source d'attributs

Create Type personnel_t as object(noP number, nomP varchar(50)) **NOT Instantiable** not Final /

Create Type persTech_t **UNDER** personnel_t (brevet number, dureeB number(2)) Instantiable Final /

Create Type persCadre_t **UNDER** personnel_t (indiceC varchar(50), primeC int) Instantiable Final /

Contrainte {c, o}

- **{c, o}**

La superclasse est encore abstraite) et donc contrainte par le NOT INTANTIABLE, tandis que le o (pour *overlapping ou recouvrement*) sous-tend que les sous-classes ne sont pas contraintes:

Deux objets dans 2 sous-classes distinctes peuvent ou pas avoir le même :noP . C'est le minimum de contraintes.

Les méthodes ajoutPC_co et ajoutPT_co ne comprendront chacune qu'une seule clause DML Insert.

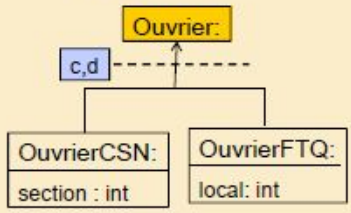
16

Table d'un type non instantiable pour y ranger des objets du sous-type

Modélisation de l'appartenace syndicale des ouvriers

```
Create or Replace type ouvrier_t as Object (nasO int, nomO varchar(50), ageO int)
not Final  not Instantiable;
/
Create type OuvrierCSN_t under ouvrier_t  (section int)
/
Create type OuvrierFTQ_t under ouvrier_t (local int)
/
Create table Ouvrier of ouvrier_t ;

Insert into Ouvrier values (ouvrierCSN_t( 10, 'Jacques', 25, 2345));
```

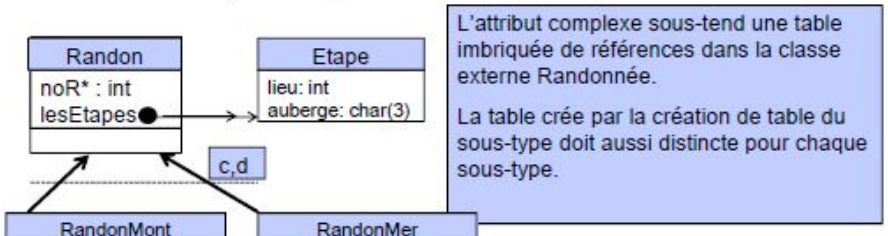


1 objet de type ouvrierCSN créé et placé dans la table Ouvrier. Un type non instantiable peut-être utilisé pour créer une table où des objets des sous-types peuvent être rangés.

Pour connaître le rang des attributs hérités, il suffit de faire le DESCRIBE. En générale, les attributs précèdent les attributs du sous-type.

Cas de l'héritage d'un attribut d'ensemble (table imbriquée)

Si les attributs sont obtenus par héritage, la table imbriquée associée à un attribut d'ensemble, ne l'est pas:



L'attribut complexe sous-tend une table imbriquée de références dans la classe externe Randonnée.

La table crée par la création de table du sous-type doit aussi distincte pour chaque sous-type.

```
Create type randon as object (noR int, lesEtapes lesEtapes_t) /

Create type lesEtapes as Table of etape_t  /

Create type etape_t as Object (lieu int, auberge char(3)) /
```

Cas de l'héritage d'un attribut d'ensemble (table imbriquée)

Create type randonMont under Randon (noMont int, alt int, km int) /

Création des tables:

Create table RandonMont of randonMont_t NESTED TABLE lesEtapes Store as **Table_lesEtapesMont;**

Create table RandonMer of randonMer_t NESTED TABLE lesEtapes Store as Table_lesEtapesMer;

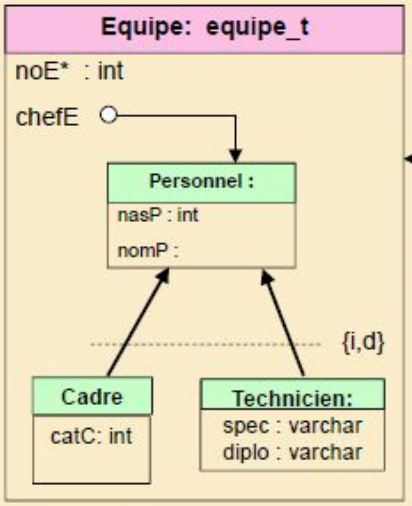
Les 2 sous-tables imbriquées doivent avoir des noms distincts même si elles **sont associées au même attribut d'ensemble.**

Insert into **RandonMont** values (randonMont_t(25, lesEtapes_t(etape_t('Qc', 'Au 2 Cœurs'), etape_t('Malbaie', 'Au Pigeon')), 4, 1110, 178));

Héritage entre les OC et leur substitution de type

20

Mnav avec spécialisation d'un OC



Modèle Mnav avec un lien interne vers une classe interne spécialisée (type de objet-colonne), donc dotée de sous-classes.

Projet:
nomP* :
refEquipe:

Substitution d'un objet-colonne (cohabitation des objets de type cadres_t et de techniciens_tdans la table-objet Equipe

Quel est le schéma OR de ce modèle ?

Mnav avec spécialisation d'un OC

La définition du schéma de ce modèle et plsu simple que sa représentation avec le Mnav!

```
Create Type personnel_t as Object ( nasP int, nomP varchar2(50))
    NOT FINAL  INSTANTIABLE
/

Create Type cadre_t UNDER Personnel_t (catC int) FINAL INSTANTIABLE
/

Create Type technicien_t UNDER Personnel_t (spec varchar2(25), diplo varchar2(50))
    FINAL INSTANTIABLE
/

Create Type equipe_t as Object (noE int, chefE  personnel_t)
/
Create Type projet_t as Object (nomPr varchar2(50), refEquipe REF equipe_t)
/
```

22

Exemple

```
Create Table Equipe of equipe_t;

Create Table Personnel of personnel_t;
Create Table Cadre of cadre_t;
Create Table Technicien of technicien_t;
```

Insertion d'objets dans la table Equipe: tout objet du type ou de son sous-type par substitution des types d'une même hiérarchie (ou par remplacement)

```
Insert into Equipe values(equipe_t(100, personnel_t(2222, 'Paul')));

Insert into Equipe values (equipe_t (101, cadre_t(3333, 'Renée', 1)));

Insert into Equipe values (equipe_t(102, technicien_t (4444, 'Louis',
    'informaticien', 'bac')));
```

(il y a alors substitution des objets-colonnes)

TREAT(): recherche sélective des chefs qui sont des cadres

```
SQL> Select Treat (chefE as cadre_t)   From Equipe;

TREAT (CHEFEASCADRE_T)(NASP, NOMP, CATC)
CADRE_T(3333, 'Renée', 1)
```

24

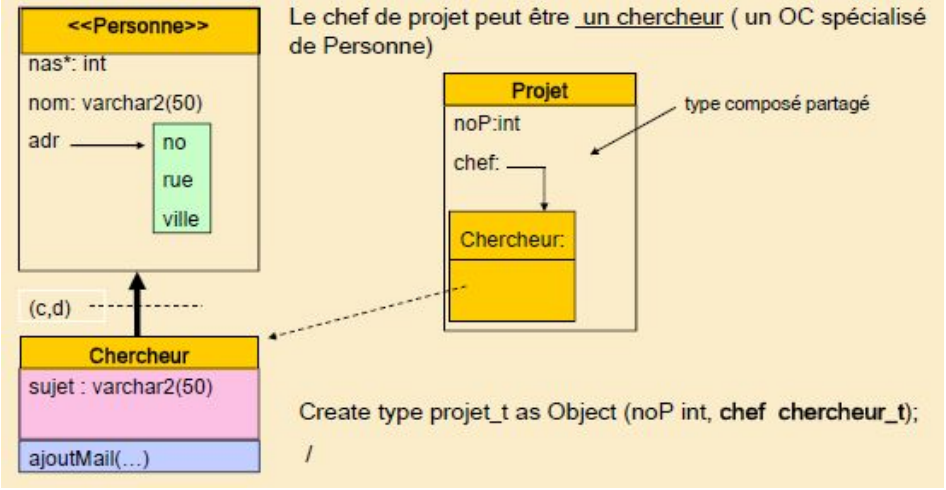
Recherche sélective des objets de la table d'un même type

•La substitution des objets de table dans une hiérarchie d'héritage signifie qu'un objet dans une table typée niv i peut être aussi un objet dans la table typée de niv i-1, i-2, …
•Les objets de T2 peuvent être obtenus en fouillant la super table racine T1 et en utilisant la fonction IS OF():

```
Select value(p)
from Personnel p
where value(p) IS OF (cadre_t) ;
```

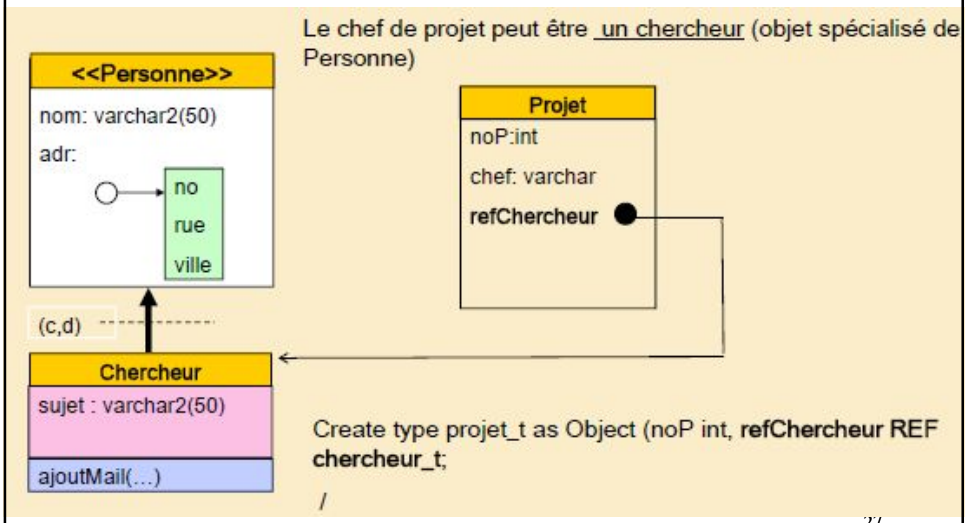
25

Exemple: substitution des types avec un OC



26

Exemple: usage d'une référence vers un sous type



27

Exemple: usage d'une référence vers un sous type

```
Create type  adr_t  as object (no int, rue varchar(50), ville varchar(50))
/
Create type personne_t  as object (nas int, nom varchar(50) , adr    adr_t)
/
Create type   chercheur_t  under Personne_t ( sujet varchar(50))
/
Create  table  Chercheur of type chercheur_t;
Create table   Projet  of type projet_t;

Création  d'un objet avec 3 constructeurs:

Insertion into Projet values (projet_t (34, chercheur_t (23, 'renée', adr_t (2,
    'lesLilas', null), 'infection résistante')));
```