Dr. Rim Samia Kaabi

Modèle OR: implémentation des associations

Manipulation des objets

1

Attribut REF valué avec une REF non contrainte

```
create type eleve_t as object ( nom varchar2(50), age int)
 /
create type capitaine_t as object (nas int, refEtudiant  REF eleve_t)
/

create table Secondaire of eleve_t;
            Insert into secondaire values  ( 'Louise', 16);
create table Cegep of eleve_t;
            Insert into cegep values  ( 'Jacques', 18);

create table Capitaine of capitaine_t ;

Valuation avec une Ref sur Secondaire:
Insert into Capitaine values ( 1231, (select ref(s) From Secondaire s
                                      Where s.nom = 'Louise'));
Valuation avec une Ref  sur Cegep:
Insert into Capitaine values ( 345, (select ref(c) From Cegep c
                                      Where c.nom = 'Jacques'));

2  références ajoutées
```

Sous table imbriquée: implantation et interrogation

3

Table imbriquée

- Varray: imbrication physique
- Nested table: stockée en dehors de la table mère
- Nested table plus rapide que le varray

4

Nested Tables

Création : 2 étapes

1. Créer le type de la table imbriquée

```
CREATE TYPE nom-type1 AS TABLE OF nom-type2;
```

nom-type2 : type usuel, type défini par l'utilisateur.

2. Créer la table maître avec la table imbriquée

```
Create une_table (
  Nom_col type, … ,
  nom_col_i nomtype1 )
  NESTED TABLE nom_col_i STORE AS nom_table
```

5

Nested Tables : exemple

Create type ty_passager (nom VARCHAR2 (35), pays VARCHAR2 (50));

1. Create type ty_liste_passagers as TABLE OF ty_passager ;

2. Create TABLE vol (

num char(6),

date DATE,

pilote varchar2 (35),

liste_passagers ty_liste_passagers)

NESTED TABLE liste_passagers STORE AS table_passagers ;

6

Association 1..*: table imbriquée

•Une table imbriquée dans une table parent est désignée par un attribut complexe à titre d'objet colonne

•Création du type de la table imbriquée:

```
(2): Create or replace type lesEtudiantsU_t AS TABLE OF etudiantU_t -- type de l'ensemble
/
(1) : Create Type etudiantU_t as Object ( matE int, nomE varchar(50), ageE int)
/
```

Type des éléments de la collection OC

OU si le type de l'élément de l'ensemble est simple ou atomique :

```
Create Type lesEtudiantsU_t AS TABLE OF varchar2(50)
/
```

7

Association 1..*: création de containeurs d'objets

•Création de la table typée StageU contenant la table imbriquée:

```
Create Table StageU of stageU_t (Constraint pk_stageU Primary Key (noS))
NESTED TABLE lesEtudiantsU STORE AS Table_LesEtudiantsU;
```

•STORE AS précise le nom de la nouvelle table physique externe Table_LesEtudiantsU

•Cette table n'est pas directement accessible par le LMD

8

Association 1..*: création de containeurs d'objets

Set Describe stageU_t depth 2;

sqlplus: describe stageU_t; (idem pour describe StageU)

Nom	NULL ?	Type
NOS		NUMBER(38)
SUJETS		VARCHAR2(35)
LESETUDIANTSU		LESETUDIANTSU_T
MATE		NUMBER(38)
NOME		VARCHAR2(50)
AGEE		NUMBER(38)

table imbriquée

Insert into StageU values (stageU_t(1,' base de données', lesEtudiantsU_t (etudiantU_t(1, 'Poulin', 28))));

Table imbriquée

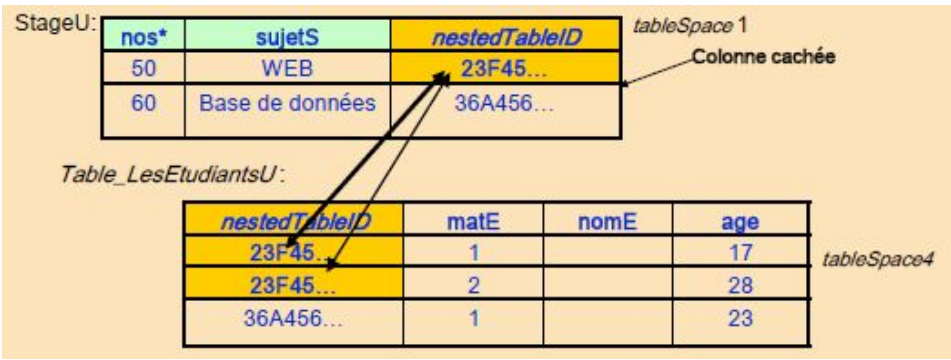
Structure logique :

StageU : stageU_t				
noS*:int	sujetS: varchar	lesEtudiantsU: { }		
		matE	nomE	ageE
50	WEB	1	Dupont	22
		2	Ardois	19
60	Base de données	1	Poulin	28

Une collection

10

Table imbriquée



11

Quelques opérateurs avec les ensembles imbriqués et Varra

Cardinality ()

Permet de compter le nombre d'objets imbriqués (nested) pour chaque objet de table.

Select Cardinality (attribut d'ensemble) from *nom_table_parent*

Select **Cardinality** (lesJoueursR) From Equipe;

Select Cardinality (lesJoueursR) From Equipe;

CARDINALITY(LESJOUEURSR)

2

Multiset Except (collection de scalaires seulement)

Accepte 2 tables imbriquées (identifiées par le nom Table_) et retourne les objets-colonnes qui sont dans la 1ère et non dans la 2ème .

Select lesjoueursR **Multiset Except** lesJoueursB From Equipe;

LESJOUEURSRMULTISETEXCEPTLESJOUEURSB(NOJ)

LESJOUEURS_T(JOUEUR_T(10), JOUEUR_T(12))

Quelques opérateurs sur les ensembles imbriqués

Multiset Intersect

Retourne les objets-colonnes communs à 2 tables imbriquées

Select lesJoueursR **Multiset Intersect** lesJoueursB from Equipe;

Exemple:

```
Create or Replace type joueur_t as object (noJ  int)
/
Create or Replace type lesJoueurs_t  as table of joueur_t
/
Create type equipe_t as object (noE int ,
            lesJoueursR lesJoueurs_t,
            lesJoueursB lesJoueurs_t)
/
```

13

Les opérateurs d'ensemble

```
Create table Equipe of equipe_t
        Nested table lesJoueursR store as Table_lesjoueursR,
        Nested table lesJoueursB store as Table_lesJoueursB;


Insert into Equipe values (equipe_t(1,
        lesjoueurs_t(joueur_t(10), joueur_t(12)),    -- joueurs réguliers
        lesJoueurs_t(joueur_t(12)));  -- joueurs blessés
```

NB Pour utiliser les deux tables imbriquées avec les opérateurs ensemblistes, il faut utiliser le même constructeur.

14

Les opérateurs d'ensemble

Set()

Retourne une table imbriquée en éliminant les doublons

Select **SET** (lesJoueursR) from Equipe ;

Equal (=) ou not Equal (<>)

Retourne un booléen si les conditions suivantes sont satisfaites:

1-Si les tables sont du même type

2- Même cardinalité

3- si les objets sont égaux par leurs valeurs d'attribut

```
…
res Boolean;
Begin
res :=  lesEtudiantsU = lesEtudiantsG;
…
End;
```

15

Augmentation de la taille d'un Varray

```
Create type liste_courriel_t  as Varray(50)  of varchar(80)
/
Create type employe_t  as Object ( nasE int, lesCourriels liste_courriel_t)
/
Alter Type liste_courriel_t MODIFY LIMIT 100 INVALIDATE ;
```

- Le type employe_t est rendu invalide.

```
Alter Type liste_courriel_t MODIFY LIMIT 100 CASCADE ;
```

- Le type employe_t est valide avec la nouvelle taille du varray.

16

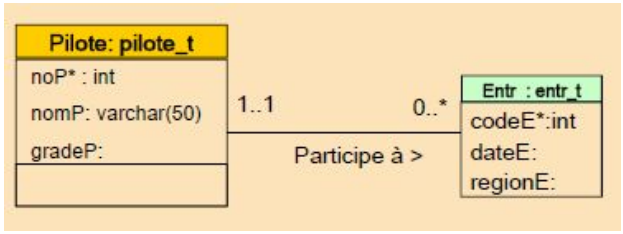
Unnesting une collection pour le résultat en FN1 avec la fonction TABLE()

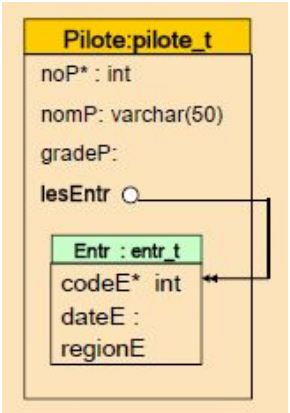
•Pour accéder aux OC, il faut faire un unnesting de la collection par la fonction TABLE()

•Une variable objet de type alias pour la table parent est obligatoire

17

Unnesting une collection pour le résultat en FN1 avec la fonction TABLE()





Quels sont les code et date d'entrainement concernant le région S

```
Select p.codeE, p.dateE    -attributs en provenance que de la table imbriquée
From Pilote p
Where p.regionE = 'S'      -- accès à la sous-table
```

ORA-00904: "P"."REGIONE" : identificateur non valide

Aucun accès possible à un attribut d'un objet-colonne sans faire Unnesting par la fonction Table().

La fonction de unnesting exige une variable de table-objet (alias)

Variable de table obligatoire:

```
Select e.codeE, e.dateE
From Table(Pilote.lesEntr) e
Where e.regionE = 'S' ;
```

ORA-00904: "PILOTE"."LESENTR" : identificateur non valide

Opération de Unnesting mal formulée! Cette opération exige l'usage d'une variable de table.

Unnesting et jointure

```
Select p.noP, e.codeE, e.dateE
From Pilote p, Table(p.lesEntr) e
Where e.regionE = 'S' ;
```

L'utilisation d'une variable objet de la table parent (alias) implique sa définition sous la forme d'une jointure implicite.

NOP	CODEE	DATEE
50	1	98-12-02
50	2	04-12-07

2.1 Quelles sont les dates d'entraînement du pilote 50 dans la région S?

Les données de la réponse proviennent que de la sous-table:

```
Select Unique e.dateE
From Pilote p, TABLE (p.lesEntr) e;
Where p.noP =50 and e.regionE= 'S';
```

La variable objet doit être associée à la table Pilote par une jointure implicite.

DATEE
98-12-02
98-12-07

Unnesting et jointure

2.2 Quelles sont les dates d'entraînement du pilote 50 dans la région S?

```
Select   Unique  e.dateE
From Pilote p, Table( p.lesEntr) e
Where e.regionE = 'S' and p.noP = 50;
```

Accès à tous les attributs dans la condition de restriction.

21

Accès aux attributs des deux niveaux: table et sous table

2.3 Listez les noP de pilote et leurs entraînements (en FN1):
les attributs demandés proviennent de la table objet et de la sous-table imbriquée; cela exige le Unnesting par la fonction Table() suivi de la joiture et que l'entraînement ait eu lieu (is not null).

```
Select    p.noP, e.dateE
From Pilote p, TABLE ( p.lesEntr)  e
Where e.dateE  is not null;
```

NOP	DATEE
50	98-12-02
50	98-12-07
60	99-11-17

La sous-table de Pilote est *Unnested* par Table() qui exige une variable objet de table définie par une jointure de la table parent et de la sous table.

Autres interrogations

- Combien de pilotes ont eu au moins un entraînement?

```
Select  Count(*)
From   Pilote p,  TABLE(p.lesEntr) e
Where e.codeE is not null ;
```

- Quels sont les noms de pilote de grade 1 ou 2 qui ont eu un entraînement?

```
Select  Unique p.nomP
From   Pilote p,  TABLE(p.lesEntr) e
Where p.gradeP = 1 or  p.gradeP= 2 ;
```

23

Ajout et mise à jour d'un élément dans une collection

•L'ajout d'un élément OC dans la sous table suppose en premier le repérage de l'objet parent et ensuite le repérage de l'objet colonne dans la collection de ce parent

Par exemple: ajout d'un OC entraînement [5, null, 'E'] dans l'ensemble des entraînements du pilote 60. Ensuite modifier son grade de pilote pour 3.

Calcul de l'ensemble lesEntr pour le pilote 60 et ensuite ajout d'un objet :

```
Insert into Table (Select  p.lesEntr  From Pilote p Where  p.noP = 60)
      Values (entr_t(5, null, 'E'));

Update Pilote p  SET p.gradeP = 3 Where p.noP =60;
```

24

Mise à jour d'un élément dans une collection

Mise à jour du pilote 60 dont l'entraînement no 1 a été effectué au 'S' et non au 'N'. Mise à jour d'un attribut de la sous-table (collection imbriquée)

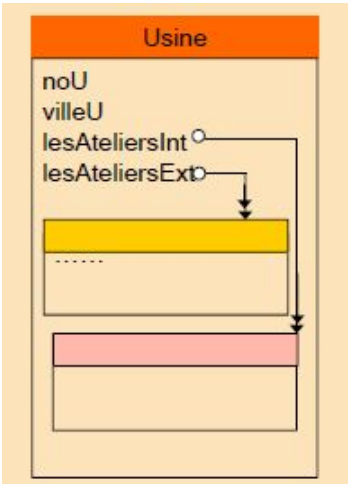
```
Update  Table(Select p.lesEntr From Pilote p Where p.noP = 60)  e
        SET e.regionE = 'N'   Where e.codeE = 1 ;
```

25

Plusieurs sous tables imbriquées dans une même table parent

26

Mnav



27

……..

Usine : usine_t						
noU*	villeU	lesAteliersIn : lesAteliersIn_t			lesAteliersEx :	
		tel	chef	specialite	chefE	villeE
100	Québec	456	Dion	soudure argon	Jean	Québec
					Lise	Hull
200	Montréal	327	Joliot	pliage acier	Kevin	Lévis
		678	Curie	structure	null	null

28

Schéma

```
Create or Replace type atelierIn_t as Object (tel varchar2(12), chef
    varchar2(50), specialite varchar2(50))
/
Create type lesAteliersIn_t as TABLE OF atelierIn_t
/
Create type atelierEx_t as object (chefE varchar2(12), villeE varchar2(50))
/
Create type lesAteliersEx_t as TABLE of atelierEx_t
/
Create type Usine_t as object (noU int, villeU varchar2(50),
    lesAteliersIn   lesAteliersIn_t, lesAteliersEx   lesAteliersEx_t)
/
```

29

Schéma

```
Create table Usine of Usine_t    (Constraint pk_Usine Primary Key(noU))
NESTED TABLE lesAteliersIn STORE AS Table_LesAteliersIn
    NESTED TABLE lesAteliersEx STORE AS Table_LesAteliersEx;
```

Absence de virgule entre les clauses Nested Table?

Insertion d'un objet de table Usine:

```
Insert Into Usine values(usine_t(100, 'Québec',
            lesAteliersIn_t (atelierIn_t('456', 'Dion', 'soudure argon')),
            lesAteliersEx_t ( atelierEx_t('Jean', 'Québec'),
                        atelierEx_t('Lise', 'Hull'))));
```

30

Insertion….

Insertion d'un 2è objet de type Usine:

```
Insert Into Usine values (usine_t(200, 'Montréal',
  lesAteliersIn_t (atelierIn_t('327', 'Joliot', 'pliage acier'),
                   atelierIn_t('678', 'Curie', 'structure')),
  lesAteliersEx_t ( atelierEx_t('Kevin', 'Lévis'), atelierEx_t(null, null))));
```

Objet Null

31

Insertion d'un objet dans une sous table existante

Repérage de la collection spécifique et insertion de l'atelierEx:

Insertion d'un atelier externe pour l'usine no 100:

```
Insert into TABLE(Select  u.lesAteliersEx From Usine u
                                                Where u.noU = 100)
       Values (atelierEx_t('Jacques', 'Rimouski'));
```

Insertion d'un atelier interne à l'usine 100:

```
Insert into TABLE (Select  u.lesAteliersIn From Usine u
                    Where  u.noU = 100)
       Values (atelierIn_t ( 999, 'Julia', 'soudure'));
```

32

Table objet avec un attribut complexe et un autre d'ensemble

```
Create type person_t as object (noP int, nomP varchar(50), age int) /
Create type lesEmp_t as Table of person_t /
Create or Replace type dep_t as Object (noD int, nomD varchar(50),
gerant person_t, lesEmp lesEmp_t) /
```

```
Create Table Dep of dep_t NESTED TABLE lesEmp STORE AS
Table_LesEmp;
```

NB avec une table objet aucune assignation avec DEFAULT n'est permise pour un attribut typé.

Dep:

noD	nomD	Gerant			lesEmp		
		noP	nomP	age	noP	nomP	age
25	Optique	1	Gagnon	29	100	Boivin	35
					101	Daneau	42
20	Laser	null	null	null	nil	nil	nil

Jointure

```
Insert into Dep values (dep_t( 25, 'optique', person_t (1, 'Gagnon', 29),
lesEmp_t (person_t(100, 'Boivin', 35), person_t(101, 'Daneau', 42))));
```

```
Insert into Dep values (dep_t (20, 'Laser', null, lesEmp_t()));
```

Création d'un objet nil

Pour obtenir les no de département et leurs employés:

```
Select d.noD, e.*
From Dep d , Table(d.lesEmp) e ;
```

34

Implantation du lien multiple interne faisant usage d'un ensemble de référence!!!!!!

35

Imbrication de table à 2 niveaux
Agrégation
réflexivité

36