Bases de Données Avancées

Le Relationnel Objet : SQL3 Modèle et Langage

– USTHB Master 1 IL–
Z. ALIMAZGHI – M. AZZOUZ
3ème version : Mars 2020

Sources du cours

- Bases de données Architecture, modèles relationnels et objets : Serge Miranda université de Nice Sofia Antipolis Dunod

- Introduction au langage SQL3 (SQL99) : Bernard Espinasse université Aix Marseille

- BDD Avancées : Richard Chbeir, Université de Pau et de des pays de l'Adour

- Les bases de données objet et relationnel : Georges Gardarin Ed Eyrolles 1999

PLAN

- Insuffisances du modèle relationnel
- Concepts objet
- Modèle Relationnel-Objet
- SQL3 (sous ORACLE) : LDD, LMD & LID
- Conclusion

Insuffisances du modèle relationnel

- Opérations séparées des données
  - procédures stockées non intégrées dans le modèle
- Limites du modèle de données
  - 1ère forme normale de Codd
  - inadapté aux objets complexes ((ex: identité numérique, documents structurés) et non structurés
  - introduction du BLOB (Binary large object): comment les manipuler ?
- les langages
  - Pas de manipulation d'ensembles dans les LP
- Mauvais support des applications non standards
  - CAO, CFAO
  - BD Géographiques
  - BD techniques

L'apport des modèles objets

- **Identité d'objets**
  - -Introduction de pointeurs invariants
  - -Possibilité de chaînage
- **Encapsulation des données**
  - -Possibilité d'isoler les données par rapport aux opérations
  - -Facilite l'évolution des structures de données (les interfaces restent inchangées)
- **Héritage d'opérations et de structures**
  - - Facilite la réutilisation des types de données
  - –Permet l'adaptation à son application
- **Possibilité d'opérations abstraites (polymorphisme)**
  - - Simplifie la vie du développeur

Extensions RICE du modèle Relationnel : le modèle relationnel Objet [Miranda]

Les modèles relationnels-objets permettent de tirer partie à la fois du modèle relationnel et des concepts objet.
La norme SQL2 a donc été étendue dans SQL3 pour intégrer les concepts objets caractérisés par:

- **R** Réutilisation
- **I** Identité d'objets
- **C** Complexité
- **E** Encapsulation

Evolution de SQL 3 [Miranda]

- SQL 3-99 hybride : objet relationnel

- SQL4- 2003 (SQL/OLB Object Langage Binding; OLAP

- SQL 5- 2006 (SQL/XML : Xquery)

- SQL 6 – 2008 (SQL/JRT pour fonctions Java)

- SQL7-2011 ISO/IEC 9095 : 2011

- SQL8 ? (Teradata, SQL/B cf Bigquery de Google etc.)

Remarque

Différents composants de SQL3 : (noyau, SQL/CLI, SQL/Temporal etc.)

Nous nous intéressons dans ce cours au noyau de spécification :

- **types abstraits de données,**
- **Objets**
- **opérations de définition, manipulation, interrogation.**

Types abstraits de données et tables objets

Il est possible de définir ses propres types : types abstraits de données ou **TAD**:

- Un TAD est une structure de données partagée
- qui inclut des méthodes (procédures où fonctions) pour manipuler les objets ou instances du TAD (lorsqu'il est inclu dans une relation)
- Il sera ensuite réutilisé dans la définition des relations.

  L'objectif est de considérer un groupe de colonnes dans une relation comme un tout, comme une seule entité.

1. Types abstraits de données et tables objets

1.1 Composition de types:

Déclaration

CREATE OR REPLACE TYPE TNOM AS OBJECT
(ELEMENT1,
ELEMENT2
); /

où un élément est

- un attribut défini avec son nom et son type :
  nom_attribut type_attribut
- ou une méthode :
  mamethode *member* ... ,
  mamethode *static* ... ,

member s'applique sur l'objet tandis que *static* correspond aux méthodes appelées par la classe (par exemple le constructeur).

1. Types abstraits de données et tables objets

Exemple1 de TAD : attribut d'une table relationnelle

PERSONNE						
NUMP	NOMP	SALAIRE	ADRESSE			
			Num	Rue	Ville	Codepostal

Remarque

Une table qui contient un TAD est une table relationnelle objet

1. Types abstraits de données et tables objets

1.1 Composition de types:

Exemple1

```
CREATE TYPE TADRESSE AS OBJECT
(RUE VARCHAR2(50),
VILLE VARCHAR2(25),
CODE_POSTAL NUMBER
);/
```

Il est ensuite possible d'utiliser ce TAD :

pour le ré-utiliser dans une relation :

```
CREATE TABLE PERSONNE(
NUMP NUMBER,
NOMP VARCHAR2(25), SALAIRE NUMBER
ADRESSE TADRESSE);
```

N.B. Personne n'est pas une table d'objets. Il n'y a pas de mécanisme d'OID.

1. Types abstraits de données et tables objets

1.1 Composition de types:

Exemple1

```
CREATE TYPE TADRESSE AS OBJECT
(RUE VARCHAR2(50),
VILLE VARCHAR2(25),
CODE_POSTAL NUMBER
);/
```

Il est ensuite possible d'utiliser ce TAD :

pour créer une table d'objets

```
CREATE TABLE ADRESSE OF TADRESSE;
```

N.B. Adresse est alors une table d'objets avec gestion des OID associés.

1. Types abstraits de données et tables objets

1.1 Composition de types:

Exemple1

```
CREATE TYPE TADRESSE AS OBJECT
(RUE VARCHAR2(50),
VILLE VARCHAR2(25),
CODE_POSTAL NUMBER
);/
```

Il est ensuite possible d'utiliser ce TAD :

pour définir un autre TAD complexe :

```
CREATE TYPE TPERSONNE AS OBJECT
(NUMP NUMBER, NOMP VARCHAR2(8), SALAIRE
NUMBER, ADRESSE TADRESSE);/
CREATE TABLE OPERSONNE OF TPERSONNE ;
```

N.B. OPersonne est alors une table d'objets.

1. Types abstraits de données et tables objets

1.1 Composition de types:

Dictionnaire

- Pour connaître la description du TAD, il faudra utiliser les tables systèmes : ***user_types*** (type name, attributes, methodes, ...) et ***user_type_attrs*** pour connaître les attributs d'un TAD.
- Si l'on n'est pas propriétaire d'un TAD, il est nécessaire pour l'utiliser d'avoir les droits :

  GRANT EXECUTE ON NOM_TAD TO UTILISATEUR ;
- Ceci donne également accès aux différentes méthodes du TAD.
- De la même façon que pour les classes, il est associé au TAD un constructeur de même nom.

1. Types abstraits de données et tables objets

1.1 Composition de types:

Type forward ou Type Incomplet

- Dans le cas de TAD inter-dépendants (par exemple TVoiture qui utilise TPersonne et TPersonne qui utilise TVoiture), il est impossible de créer un TAD complet tant que l'autre n'est pas déjà déclaré. Oracle permet alors de définir un TAD incomplet sans attribut ni méthode qui est complété ensuite.

- Syntaxe : CREATE [OR REPLACE] TYPE NOM_TAD;

1. Types abstraits de données et tables objets

1.1 Composition de types:

Insertion

- Dans une table incluant un TAD

On utilise alors un constructeur pour le sous-type construit.

```
INSERT INTO PERSONNE VALUES (10, 'ADLI',
TADRESSE('10 RUE AHMED KATIB','ALGER',16140)
);
```

1. Types abstraits de données et tables objets

1.1 Composition de types:

Insertion

- Dans une table associée à un TAD

L'utilisation du constructeur n'est pas obligatoire :

a- INSERT INTO ADRESSE VALUES ('10 RUE AHMED KATIB',ALGER',16140));

b- INSERT INTO OPERSONNE VALUES (*TPERSONNE* (10 'ADLI',TADRESSE('10 RUE AHMED KATIB',ALGER',16140)));

Ou bien INSERT INTO OPERSONNE VALUES (10 'ADLI',TADRESSE('10 RUE AHMED KATIB',ALGER',16140));

N.B. Dans le dernier exemple ci-dessus, si le constructeur Tpersonne n'est pas obligatoire, le constructeur Tadresse est **obligatoire**.

1. Types abstraits de données et tables objets

1.1 Composition de types:

Sélection

```
SELECT * FROM OPERSONNE;
```

N.B. la syntaxe suivante est fausse.

```
SELECT NOM, VILLE FROM OPERSONNE;
```

Pour accéder à la valeur de la ville, il faut indiquer les attributs des TAD avec leur chemin complet avec notation pointée. On utilise alors une variable de corrélation pour accéder aux attributs des TAD :

```
SELECT NOM, P.ADRESSE.VILLE FROM OPERSONNE P;
SELECT NOM FROM OPERSONNE P WHERE
P.ADRESSE.VILLE LIKE 'ALGER' ;
```

1. Types abstraits de données et tables objets

1.1 Composition de types:

Update-delete

```
UPDATE OPERSONNE P
SET P.ADRESSE.VILLE = 'ALGER'
WHERE P.ADRESSE.CODE_POSTAL LIKE '16%';
```

La même syntaxe est valable pour les suppressions
Si une table d'objets utilise un TAD non imbriqué, les ordres UPDATE et DELETE respectent la même syntaxe que pour des tables relationnelles.

N.B. si le TAD n'appartient pas à l'utilisateur, il est nécessaire de spécifier le nom du propriétaire.
NOM_UTILISATEUR.CONSTRUCTEUR

1. Types abstraits de données et tables objets

1.1 Composition de types:

Index

- Il est possible de créer des index sur les attributs appartenant à un TAD. Il faut toujours spécifier le chemin complet. Il n'y a pas de besoin de variable de corrélation.

```
CREATE INDEX IND_VILLE_OPERSONNE
ON OPERSONNE(ADRESSE.VILLE);
```

- L'index ne concerne que la table spécifiée et non le TAD.

1. Types abstraits de données et tables objets

1.2 Collection:

- Deux types de collections :

  -Les tableaux de longueur variable (VARRAY). Le nombre maximum d'éléments EST CONNU
  -Ensemble Multiple ou Bag (NESTED TABLE). Le nombre maximum d'éléments N'EST PAS CONNU

- Une collection peut contenir :

  -Des types de données élémentaires.
  -Des types de données abstraits.
  -Des références vers des types de données abstraits
  -Des collections (VARRAY ou NESTED TABLE)

1. Types abstraits de données et tables objets

1.2 Collection:

tableaux Varray

Un tableau Varray est un tableau de taille limitée à longueur variable. Il est représenté par une seule valeur de type n-uplet dans la table.

```
CREATE OR REPLACE TYPE TTABVAR AS
VARRAY(NB_MAX_ELTS) OF TYPE_ELT;
/

CREATE OR REPLACE TYPE TLISTE_PRENOMS AS
VARRAY(4) OF VARCHAR2(15);
/
```

1. Types abstraits de données et tables objets

1.2 Collection:

tableaux Varray

On peut utiliser ces collections pour construire des types complexes.

```
CREATE OR REPLACE TYPE TPERSONNE AS OBJECT
(NOM VARCHAR2(15),
PRENOMS TLISTE_PRENOMS,
DATE_NAISSANCE DATE,
ADRESSE TADRESSE
);/

CREATE TABLE OPERSONNE OF TPERSONNE;
```

N.B. Un type ne peut être supprime ou recréé que si aucune table ne l'utilise. Sinon, il faut utiliser la clause ALTER TYPE. La clause CASCADE permet alors de faire le changement sur tous les types et objets dépendants.

1. Types abstraits de données et tables objets

1.2 Collection:

tableaux Varray

Un tableau de type Varray est stocké comme une unique valeur.

INSERT INTO OPERSONNE VALUES
('ADLI', *TLISTE_PRENOMS('AHMED','AMINE','FADI')*,
'18-03-1997', TADRESSE('10 RUE AHMED KATIB','ALGER', 16140)));

Pour accéder au contenu du Varray, on peut :

- utiliser la requête suivante : SELECT * FROM OPERSONNE

- utiliser un programme PL/SQL en utilisant une fonction count et la notation tab(i) pour accéder au ième élément du tableau

- utiliser la clause table :

SELECT NOM, Q.* FROM OPERSONNE P, *TABLE* (P.PRENOMS) Q;

1. Types abstraits de données et tables objets

1.2 Collection:

tables imbriquées

- Une table imbriquée est du type collection et est représentée sous la forme d'une colonne dans la table d'origine.
- **N.B.** une table imbriquée n'est pas une table d'objets.
- Pour définir une table imbriquée :
  1. créer le TAD représentant la structure d'un n-uplet de la table imbriquée,
  2. créer un TAD spécifiant la table imbriquée,
  3. définir la table utilisant un attribut du type de la table imbriquée.

1. Types abstraits de données et tables objets

Exemple2 de TAD : table imbriquée

DEPARTEMENT				
Numdpt	**Budget**	**PROJETS**		
		Nump	**Intitule**	**Duree**
D1	40500	n1	A1	2
		n2	A2	2
D2	50600	n3	A3	3
		n4	A4	2

1. Types abstraits de données et tables objets

1.2 Collection:

tables imbriquées

Exemple de déclaration:

```
CREATE TYPE  TPROJETS AS OBJECT
(NUMP VARCHAR2(50),
INTITULE VARCHAR2(25),
DUREE NUMBER);
/
CREATE TYPE NT_PROJETS AS TABLE
OF TPROJETS;
/
```

```
CREATE TABLE DEPARTEMENT
(
NUMDP VARCHAR2(25)
PRIMARY KEY, BUDGET
NUMBER, PROJETS
 NT_PROJETS
)
NESTED TABLE  PROJETS  STORE
AS NT_PROJETS_TAB ;
```

La clause nested crée une table avec une colonne projets qui est une table imbriquée.
NT_PROJETS_TAB a soit été créée avant, soit est créée automatiquement. Les valeurs de cette colonne seront stockées dans la table indiquée dans l'ordre nested. Oracle maintient les pointeurs adéquats.

1. Types abstraits de données et tables objets

1.2 Collection:

tables imbriquées Exemple3 Adresse table imbriquée

```
CREATE TYPE TADRESSE AS OBJECT
(RUE VARCHAR2(50),
VILLE VARCHAR2(25),
CODE_POSTAL NUMBER);
/
CREATE TYPE NT_ADRESSE AS
TABLE OF TADRESSE;
/
```

```
CREATE TABLE PERSONNE
(
NOMP VARCHAR2(25),
ADRESSES NT_ADRESSE
)
NESTED TABLE ADRESSES STORE
AS NT_ADRESSE_TAB ;
```

1. Types abstraits de données et tables objets

1.2 Collection:

tables imbriquées

Exemple d'insertion:

Il est possible d'insérer des n-uplets dans une table imbriquée. Il faut alors utiliser les constructeurs :

```
INSERT INTO PERSONNE
VALUES
('ADLI',
NT_ADRESSE(
TADRESSE('10 RUE AHMED KATIB','ALGER', 16140),
TADRESSE('BORDJ EL KIFFAN','ALGER', 16130)));
```

1. Types abstraits de données et tables objets

1.2 Collection:

tables imbriquées

Exemple d'insertion:

Il est également possible d'insérer de nouvelles valeurs dans la table imbriquée grâce au mot-clé TABLE (anciennement nommé THE):

La clause *TABLE* est une extension du LMD permettant de manipuler les objets (scalaires ou enregistrements) dans les collections implémentées sous forme de tables imbriquées.

```
INSERT INTO TABLE (SELECT ADRESSES FROM
PERSONNE WHERE NOM='ADLI')
VALUES (
TADRESSE('PLACE DU 1er Mai','ALGER',16450));
```

1. Types abstraits de données et tables objets

1.2 Collection:

tables imbriquées

Exemple d'insertion:

Pour effectuer des insertions groupées (par exemple deux personnes ayant les mêmes adresses), on utilise CAST et MULTISET.

- L'instruction *CAST (...) AS <type>* permet de renvoyer un type donné à partir d'une expression.
- L'instruction *MULTISET (...)* permet de renvoyer une collection à partir d'une liste de valeurs.

```
INSERT INTO PERSONNE VALUES ('MOMO',
CAST(MULTISET(SELECT * FROM THE (SELECT ADRESSES
FROM PERSONNE WHERE NOM='ADLI') ) AS NT_ADRESSE)
);
```

1. Types abstraits de données et tables objets

1.2 Collection:

tables imbriquées

Mise à jour et suppression:

On peut également utiliser UPDATE TABLE pour modifier et supprimer les n-uplets de la table imbriquée.

Exemple:

- UPDATE TABLE
(SELECT ADRESSES FROM PERSONNE
WHERE NOM = 'ADLI') *nttab*
SET NTTAB.VILLE='ORAN'
WHERE *nttab*.VILLE='ALGER';
- La suppression d'une collection complète est possible :
UPDATE PERSONNE SET ADRESSES = NULL WHERE NOM = 'ADLI';

1. Types abstraits de données et tables objets

1.2 Collection:

tables imbriquées

Interrogation d'une table imbriquée :

L'Interrogation est possible :

- En utilisant le mot-clé *table* :

```
SELECT nt.VILLE
FROM PERSONNE P, TABLE (P.ADRESSES) nt
WHERE NOM='ADLI'
AND nt.RUE LIKE '% AHMED K%';
```

1. Types abstraits de données et tables objets

1.2 Collection:

Comparaison tables imbriquées et Varray:

	Varray	Nested Table
Éléments ordonnés	oui	non
Stockage	1 valeur	1 table
Requêtes	table	table
Mise à jour	Difficile Utilisation de PL/SQL	update table

2. Identification d'objet OID

- Dans une table d'objet, chaque n-uplet possède un identifiant d'objet (OID ou Object Identifier).
- **N.B.** L'OID est affecté par le système, il permet :
  - d'identifier de manière unique ce n-uplet,
  - de référencer cet objet de manière explicite.
- Une table d'objet est créée comme étant associée à un TAD.
  - CREATE TABLE OPERSONNE OF TPERSONNE;
  - CREATE TABLE OADRESSE OF TADRESSE;

2. Identification d'objet OID

3.1 la fonction Ref

La fonction Ref permet d'accéder à l'OID des objets concernés.

Exemple:

```
SELECT REF(P)
FROM OPERSONNE P
WHERE P.NOM='ADLI';
```

N.B. La fonction *Ref* ne s'applique qu'aux objets et non aux colonnes associées à un TAD.

2. Identification d'objet OID

PROBLEMATIQUE

PERSONNE				
NUMP	ADRESSE	SALAIRE	NUMVOITURE	NUMPROJET

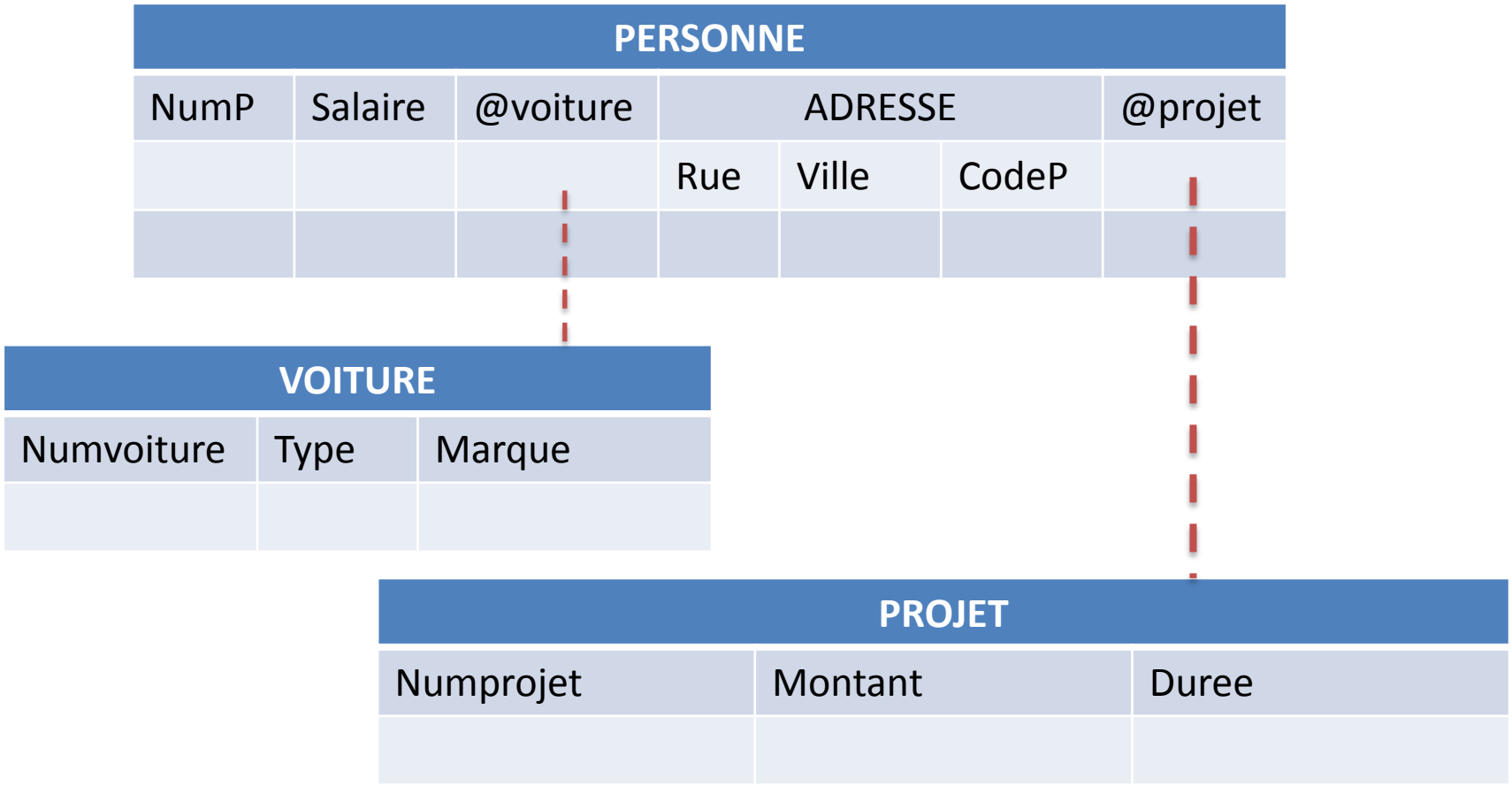
VOITURE		
NUMVOITURE	TYPE	MARQUE

PROJET		
NUMPROJET	MONTANT	DUREE

Modèle relationnel

2. Identification d'objet OID

Modèle Relationnel Objet



2. Identification d'objet OID

3.2 référencement explicite

On spécifie alors pour le type de l'attribut le fait qu'il pointe vers un objet dont la structure est le TAD spécifié. dans une table d'objets.

Syntaxe : NOM_ATT REF TYPE_OBJET;

Exemple:

```
CREATE TYPE TVOITURE AS
OBJECT
(MODELE VARCHAR2(15),
IMMAT VARCHAR2(10)
);
/
CREATE TABLE OVOITURE OF
TVOITURE;
```

```
CREATE OR REPLACE TYPE
TPERSONNE AS OBJECT
(NOM VARCHAR2(15),
PRENOMS TLISTE_PRENOMS,
VOITURE REF TVOITURE
);
/
CREATE TABLE OPERSONNE
OF TPERSONNE;
```

2. Identification d'objet OID

```
CREATE TYPE TADRESSE
AS OBJECT
(RUE VARCHAR2(50),
VILLE VARCHAR2(25),
CODE_POSTAL NUMBER
);/
CREATE TABLE ADRESSE OF
TADRESSE
```

```
CREATE TYPE  TPROJETS AS OBJECT
(NUMP VARCHAR2(50),
MONTANT  NUMBER,
DUREE NUMBER);
/
CREATE TABLE PROJETS  OF TPROJETS;
```

```
CREATE TYPE TVOITURE AS
OBJECT
(NUMVOITURE VARCHAR2(8),
TYPE VARCHAR2(7),
MARQUE VARCHAR2(15));/
CREATE TABLE VOITURE of
TVOITURE
```

```
CREATE TYPE TPERSONNE AS OBJECT
(NUMP VARCHAR2 (5), SALAIRE NUMBER,
VOITURE REF TVOITURE,
ADRESSE TADRESSE
PROJET REF TPROJET)
```

```
CREATE TABLE PERSONNE OF TPERSONNE
```

2. Identification d'objet OID

2.3 Dictionnaire et insertion

- La table système USER_DEPENDENCIES stocke les dépendances entre types via les REF.
- L'insertion s'effectue en utilisant la fonction REF qui retourne l'OID d'un n-uplet spécifié :
- *Exemple*

```
INSERT INTO OPERSONNE VALUES
('ADLI', TLISTE_PRENOMS('AHMED','FADI'),
(SELECT REF(V) FROM OVOITURE V WHERE
V.NUMVOITURE='1025011816' ));
```

- Le problème est qu'il faut connaître une valeur pour accéder à l'objet. L'attribut de type REF est stocké comme un pointeur dans la table.

2. Identification d'objet OID

2.4 La fonction DEREF

- La fonction DEREF est l'inverse de REF et sert, à partir d'un OID, à accéder à l'objet associé.

- *Exemple :*

```
SELECT DEREF(P.VOITURE)
FROM OPERSONNE P
WHERE P.VOITURE.MODELE LIKE 'MEGANE%' ;
```

2. Identification d'objet OID

2.5 Références invalides

- Lorsque l'objet référencé est supprimé, la référence devient invalide. On parle alors de DANGLING REF.
- Le prédicat IS DANGLING permet de savoir si une REF est valide ou non.

- *Exemple :*

```
UPDATE OPERSONNE
SET VOITURE=NULL
WHERE VOITURE IS DANGLING ;
```

2. Identification d'objet OID

2.6 La fonction value

Cette fonction prend en paramètre l'alias d'une table ou une vue objet et renvoie l'INSTANCE du type de la table ou vue objet à partir d'une ligne de la table ou vue objet.

3. Méthodes

L'utilisateur peut définir autant de méthodes qu'ils le souhaite dans un TAD.

- En plus du CONSTRUCTEUR par défaut: constructeur ayant tous les champs du type comme paramètre, l'utilisateur peut définir ses propres constructeurs (à partir d'Oracle V10).
- La déclaration des méthodes se fait dans le TYPE SPECIFICATION et l'implémentation dans le TYPE BODY.
- Le mot clé MEMBER précède chaque méthode d'instance
- Le mot clé STATIC précède chaque méthode de classe.

3. Méthodes

Une méthode se caractérise par :

–Un nom

–Un ensemble de paramètres : les données en entrée(IN), les résultats (OUT), les données/résultats (IN/OUT), les appels par référence (CALL-BY-REFERENCE).

–Un ensemble de résultats retournés par la méthode.

–Un ensemble d'exceptions.

–Un ensemble d'indicateurs précisant la nature de la méthode (MAP/ORDER MEMBER, MEMBER, STATIC).

–PRAGMA RESTRICT_REFERENCE :WNDS, WNPS, RNDS, RNPS, TRUST)

–Un nom spécifique (un alias).

–Une clause de langage (PL/SQL, C, Java,etc…)

–Une implémentation : le body du type.

3. Méthodes

N.B :

–**MAP** function : permet de définir une règle de comparaison s'appuyant une des colonnes du type.

–**ORDER** Function: permet de comparer deux objets.

–**STATIC** Function : Une méthode statique n'a pas l'objet courant comme paramètre implicite. Elle n'a pas besoin d'un objet pour être invoquée

–**WNDS** : interdit la modification de l'état de la base

–**WNPS** : interdit la modification de l'état du package(type)

–**RNDS** : interdit la lecture de l'état de la base

–**RNPS** : interdit la lecture de l'état du package(type)

–**TRUST**: désactive provisoirement les interdits positionnés

3. Méthodes

Déclaration:

Les méthodes sont déclarées lors de la création des TAD.

```
CREATE [OR REPLACE] TYPE NOM_TYPE AS
OBJECT
(
ATT1 TYPE1,
...
ATTN TYPEN,
MEMBER FUNCTION|PROCEDURE
SIGNATURE_METHODE_1,
STATIC FUNCTION|PROCEDURE
SIGNATURE_METHODE_2
);
/
```

3. Méthodes

Implémentation:

Le corps des méthodes est défini par :

```
CREATE OR REPLACE TYPE BODY NOM_TYPE
AS
MEMBER NOM_METHODE IS
BEGIN
...
END ;
...
END;
/
```

3. Méthodes

Exemple:

```
CREATE OR REPLACE TYPE TPERSONNE AS OBJECT
(NOM VARCHAR2(25),
DATE_NAISS DATE,
MEMBER FUNCTION AGE RETURN NUMERIC
);
/
CREATE OR REPLACE TYPE BODY TPERSONNE
AS
MEMBER FUNCTION AGE RETURN NUMERIC IS
AGE_VAL NUMBER;
BEGIN
SELECT ROUND((SYSDATE-SELF.DATE_NAISS)/365) INTO AGE_VAL
FROM DUAL;
RETURN AGE_VAL;
END ;
END;
/
```

3. Méthodes

- Il est possible de surcharger des noms de méthode. On déclare alors deux méthodes avec le même nom et des paramètres différents et Oracle choisit la méthode à exécuter.
- L'appel des méthodes s'effectue de la même manière que pour les fonctions et procédures stockées en indiquant le chemin avec la notation pointée comme pour les attributs :
- *Exemple*

```
SELECT P.NOM, P.AGE()
FROM OPERSONNE P;
```

- Les méthodes peuvent également être utilisées dans un programme PL/SQL, java ...

4. Héritage

- L'héritage n'était pas supporté avant la version 9 d'Oracle. Il permet de spécialiser un type via la clause *NOT FINAL* :

- CREATE TYPE NOM_TYPE AS OBJECT
(...)
NOT FINAL;
/
ou
- ALTER TYPE NOM_TYPE *NOT FINAL* ;
- Il est alors possible de créer des sous-types via la clause *UNDER* :

- CREATE TYPE SOUS_TYPE *UNDER* SUPER_TYPE
(ATTRIBUT_SPECIALISE_1 TYPE_ATT1,
...
ATTRIBUT_SPECIALISE_N TYPE_ATTN,
);/

4. Héritage

- Il y a héritage des attributs et des méthodes. La surcharge des méthodes est possible. Dans le cas de la généralisation, il est possible de déclarer un type comme NOT INSTANTIABLE.

- CREATE TYPE SUPER_TYPE AS OBJECT(...) NOT INSTANTIABLE NOT FINAL;
/
- La table créée au niveau du super-type stocke des sous-types et des super-types. Par exemple, si on considère le type Personne et le sous-type Etudiant, on peut insérer des étudiants dans Personne.

- Pour accéder à certains n-uplets d'un type spécifié, on utilise TREAT :

- select TREAT(value(p) as TEtudiant).num from OPersonne p;

5. Large Object-LOB

Grands objets :

- **Un grand objet** est un type de donnée capable de contenir une très grande quantité de données, par exemple un fichier de texte ou un graphisme.
- SQL3 définit trois types différents de grands objets :
  - ✓ Le grand objet binaire (BLOB, *Binary Large Object*), une chaîne binaire qui ne possède pas de jeu de caractères ni d'association de collationnement.
  - ✓ Le grand objet de caractères (CLOB, *Character Large Object*), le grand objet de caractères nationaux (NCLOB, *National Character Large Object*), tous deux des chaînes de caractères.

Exemple :

```
ALTER TABLE Personnel
ADD COLUMN résumé CLOB(50K);
ALTER TABLE Personnel
ADD COLUMN photo BLOB(12M);
```

6. Vues Objet

❑ L'objectif des vues objets est d'implémenter une structure orientée objet sur des tables relationnelles plates. Ce principe est très intéressant puisqu'il permet de faire cohabiter:

❑ Une ancienne base relationnelle avec une nouvelle base objet, ou bien de conserver les acquis du relationnel tout en montrant une base objet à l'utilisateur.

❑**Par exemple, on considère la vue objet *VOPersonne* permettant de recueillir les informations sur une personne :**

VOPersonne	Num Secu	Nom	Prenom	Adresses
	1245632563	ADIMI	Fadi	'10 RUE AHMED KATIB','ALGER','16140'

Cette vue objet s'appuiera sur une base relationnelle classique en *3NF* :

Personne	Num Secu	Nom	Prenom
	1245632563	ADIMI	Fadi

Habite	Num Secu	Rue	Ville	Code_postal
	1245632563	10 RUE AHMED KATIB	ALGER	16140

6. Vues Objet

Les étapes de création des vues objets

Pour créer une telle vue objet, on procédera selon les étapes suivantes :

1. *Création des tables relationnelles:*

```
CREATE TABLE PERSONNE
(NUM NUMBER CONSTRAINT CP_PERSONNE PRIMARY
KEY,
NOM VARCHAR2(25),
PRENOM VARCHAR2(25)) ;
CREATE TABLE HABITE (NUM ...);
```

6. Vues Objet

Les étapes de création des vues objets

Pour créer une telle vue objet, on procédera selon les étapes suivantes :

2. *Création des types de données abstraits:*

```
CREATE TYPE TADRESSE AS OBJECT
(RUE VARCHAR2(50),
VILLE VARCHAR2(25),
CODE_POSTAL NUMBER);
/
CREATE TYPE TLISTE_ADRESSES AS VARRAY(10) OF TADRESSE;
/
CREATE TYPE TPERSONNE AS OBJECT(
NUM NUMBER,
NOM VARCHAR2(25),
PRENOM VARCHAR2(25),
ADRESSES TLISTE_ADRESSES);
/
```

6. Vues Objet

Les étapes de création des vues objets

Pour créer une telle vue objet, on procédera selon les étapes suivantes :

3. Création de la vue objet:

```
CREATE VIEW VOPERSONNE OF TPERSONNE
WITH OBJECT OID(NUM)
AS
SELECT NUM, NOM, PRENOM, CAST(
MULTISET(
SELECT TADRESSE(RUE, VILLE, CODE_POSTAL)
FROM HABITE H
WHERE P.NUM = H.NUM)
AS TLISTE_ADRESSES)
FROM PERSONNE P ;
```

La clause *OBJECT OID* indique à Oracle sur quelle colonne s'appuyer pour construire les OID.

Conclusion

Comparaison de la modélisation des données des SGBDRO et SGBDOO.

Caractéristique	SGBDRO	SGBDOO
Identité d'objet (IDO)	Supporté via le type REF	Supporté
Encapsulation	Supportée via les TDU	Supportée mais rompue au niveau des requêtes
Héritage	Supporté (hiérarchies distinctes pour les TDU et les tables)	Supporté
Polymorphisme	Supporté (invocation d'une FDU fondée sur la fonction générique)	Supporté comme dans un langage modèle de programmation orientée objet
Objets complexes	Supportés via les TDU	Supportés
Associations	Support fort avec les contraintes d'intégrité référentielle définies par l'utilisateur	Supportées (par exemple à l'aide de bibliothèques de classes)

Conclusion

Comparaison des accès aux données des SGBDRO et SGBDOO.

Caractéristique	SGBDRO	SGBDOO
Création et accès à des données persistantes	Supportés mais non transparents	Supportés mais le degré de transparence diffère d'un produit à l'autre
Utilitaire de requête ad hoc	Support fort	Supporté via l'ODMG 3.0
Navigation	Supportée par le type REF	Support fort
Contraintes d'intégrité	Support fort	Aucun support
Serveur d'objets ou de pages	Serveur d'objets	Les deux
Évolution du schéma	Support réduit	Supportée mais le niveau de soutien dépend du produit

Conclusion

Comparaison du partage des données des SGBDRO et SGBDOO.

Caractéristique	SGBDRO	SGBDOO
Transactions ACID	Support fort	Supportées
Récupération	Support fort	Supportée mais le niveau de soutien dépend du produit
Modèles de transactions évolués	Aucun support	Supportés mais le niveau de soutien dépend du produit
Sécurité, intégrité et vues	Support fort	Support réduit