Base de Données Orienté Objet

Djellali H.

Master Réseau M1 RSI

Janvier 2021

HISTORIQUE RAPIDE DES SGBD

- **Première génération (années 60)**

  Séparation de la description des données et des programmes d'application, Traitement de fichiers reliés par des structures de graphe; SGBD IDMS
- **Deuxième génération (années 70)**

  Simplification du SGBD externe ; Modèle relationnel

  Langage non procédural, SGBD ORACLE, INFORMIX, …
- **Troisième génération (années 80)**
- **SGBD objet**

  - SGBD Relationnel objet : ORACLE 8

  - SGBD orienté objet : O2
- **Quatrième génération (années 90)**

  Bases de données et internet, Entrepôts de données (data warehouse)

  Fouille de données (data mining)

BD Objets

- **Approche Révolutionnaire (SGBD Orientés Objets)**
- • Bancilhon & al. 89 : "**Object-Oriented DBMS Manifesto", DODD Conf., dec 89**
- **Objet** = modélisation d'une entité observable du réel constituée de :

  • **Attributs** : propriétés construites à partir de types primitifs

  • **Relation(s)** : propriétés dont type = référence à un ou une collection d'objet(s)

  - De méthode(s) : fonctions pouvant être appliquées à l'objet

• **Produits : O2, …**

• **Langage standard : OQL (ODMG)**

BD Objets

- **Approche Evolutionnaire (SGBD Objet-Relational)**
- **ACM 89 : "Third Generation DBMS manifesto", 3 principes d'intégration:**
- • intégration des nouveaux concepts **d'objets et règles (encapsulation, héritage,** fonctions, déclencheurs, ...)
- maintient de l'approche ensembliste, SQL, mises à jour sur vues, ...
- intégration dans les architectures ouvertes et standardisées (client-serveur)
- **Produits : Postgres, IBM DB2 CS/UDB, CA-Ingres, Illustra, UniSQL/X, Oracle 8, Informix, ...**
- **Langage standard : SQL3 (ANSI X3 H2)**

PHILOSOPHIE OBJET
Faiblesses du modèle relationnel

- **Absence de pointeurs visibles :** pour lier des données qui se correspondent, on a besoin de faire des jointures (opérations coûteuses)
- **Non support des domaines composés** : on ne peut pas avoir pas exemple un attribut qui correspond à une adresse avec le numéro de la rue, le nom de la rue, la ville, …

  Impossible à cause de la première forme normale, qui impose l'atomicité des attributs
- **Pas d'opérations définies sur les données**
- On veut donc un SGBD capable de traiter des éléments de structure complexe ,
- des opérations sur les éléments
- des pointeurs reliant les éléments (pour de l'**héritage** par exemple)
- il nous faut des SGBD objet.

DEUX MANIÈRES D'UTILISER L'OBJET DANS LES SGBD

- On part des langages objet dans lesquels on intègre les notions des SGBD (persistance des données, aspect multi-utilisateurs, …). Ce sont
- **les SGBD orientés objet : O2** (basé sur C++)
- On part des SGBD relationnels dans lesquels on insère des notions objet. Ce sont **les SGBD**
- **relationnels objet : ORACLE 8 (SQL 3)**

SGBD Orientés Ob j e t s

- • Les 8 règles d'or d'un SGBD-OO
- • Le SGBD Orienté Objet « O2 »

Les 8 règles d'or d'un SGBD-OO

- **Les 8 règles d'or d'un SGBD-OO**

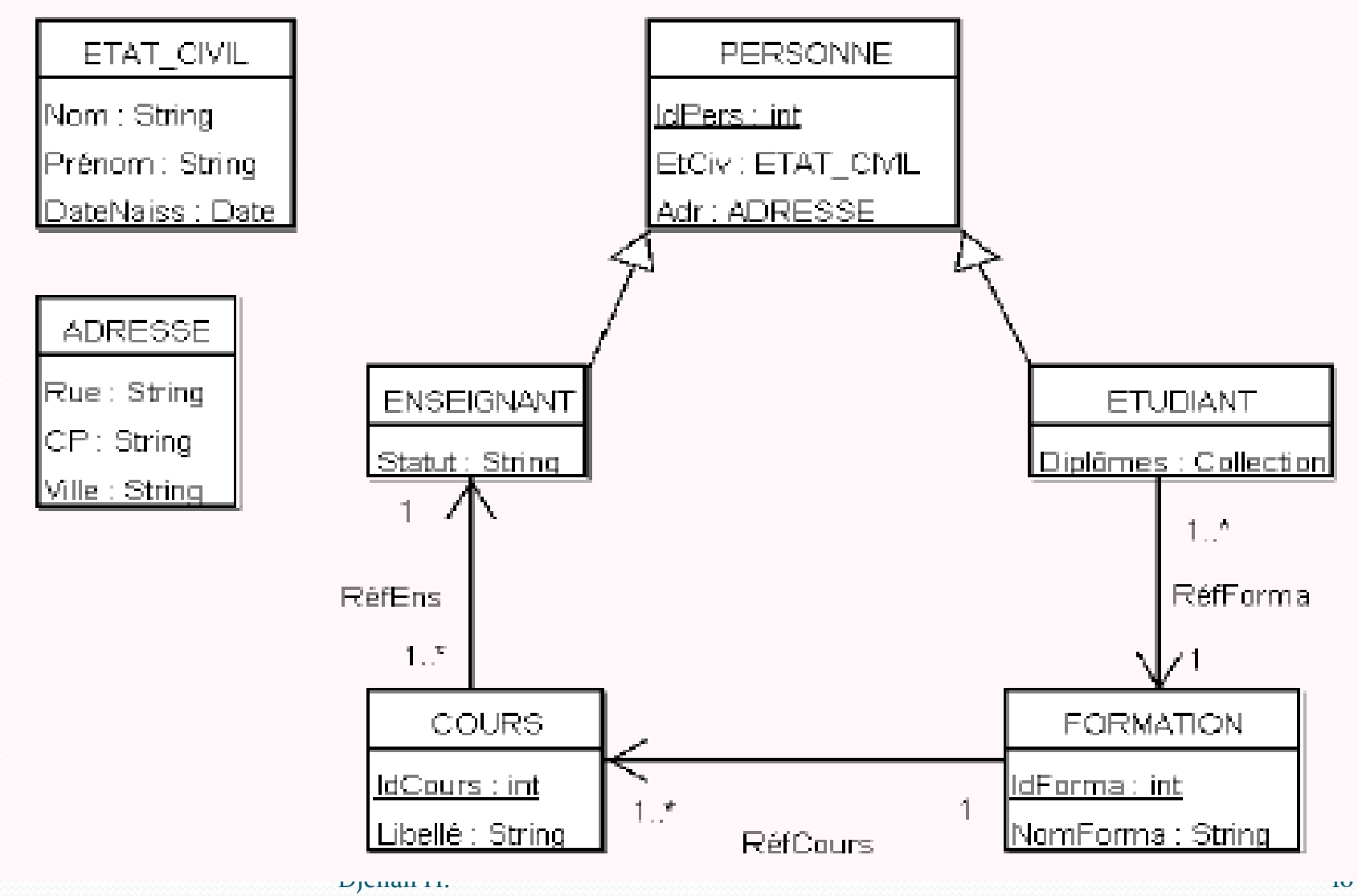
Fondements :

- Bancilhon & al.89 : "**Object-Oriented DBMS Manifesto**",
- DODD Conf., dec 89

Les 8 règles d'or d'un SGBD-OO

Services langage	Services BD
R1 : Identité d'objet (ID) mise à jour, partage, graphes sémantique d'ID	**R5 : Persistance** modèle formel, algèbre, gestion mémoire virtuelle (garb.collec.)
R2 : Abstraction (types/classes), liaisons dynamiques	**R6 : Structuration complexe** (attributs et entités), constructeurs (tuples, set), schéma
R3 : Encapsulation (méthodes, messages)	**R7 : Sécurité** confidentialité, intégrité (déclencheurs, etc...), synchronisation ; reprise après panne, transactionnel
R4 : Réutilisation héritage (simple, multiple), polymorphisme (surcharge), versions	**R8 : Interfaces non procédurales** traitements ad hoc, optimiseur

Exemple BD Objet



R1 : Identité d'objet(ID)

- **Un objet existe s'il est identifiable (OID).** Chaque objet a une identité qui lui est propre et qui le distingue de tous les autres. Cette identité est permanente et immuable. On l'appelle l'oid de l'objet (de l'anglais "object identity)
- Cette identification **est un invariant dans la vie de l'objet**
- **L'existence d'un objet est indépendante de sa valeur.**

Identité d'objet(ID)

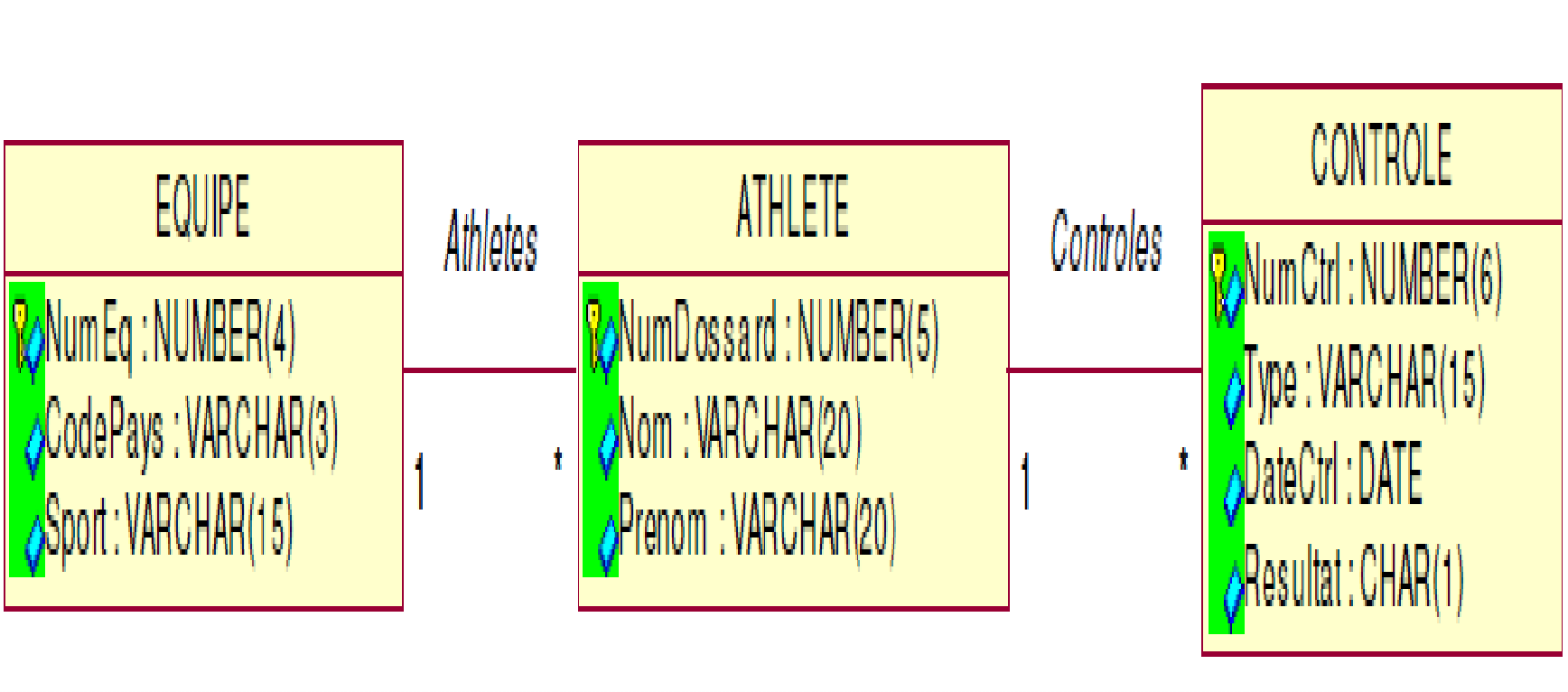
En conséquence il est possible de :

- **référer un objet de manière non ambiguë par son identifiant**
- **mettre à jour la valeur d'un objet sans remettre en cause son existence**
- **représenter des liens sémantiques entre objets (liens entre identifiants)**
- **distinguer l'identité de l'égalité (mêmes valeurs mais identités différentes)**

Liens entre les objets

- Il existe deux types de liens entre les objets :
- Les liens de **composition:** "tel objet est composé de tel(s) objet(s)". Ces liens sont décrits par des attributs particuliers, appelés attributs référence, qui ont pour domaine une classe d'objets;
- Les liens de **généralisation / spécialisation**: "tel objet de la base décrit sous un autre point de vue (plus général ou au contraire plus détaillé) la même entité du monde réel que tel autre objet de la base". Ce sont les liens IS-A des modèles sémantiques auxquels sont associés un mécanisme d'héritage des propriétés de la classe d'objets génériques par la classe d'objets spécialisés.

Diagramme de Classe



R2 : Abstraction

- Possibilité d'éliminer les caractéristiques non essentielles d'un objet de façon à
- Créer des regroupements d'objets appelés "**types**" **ou** "**classes**"
- un **objet est alors une instance d'un type ou d'une classe**

Encapsulation

- Le **type est défini avec tous les opérateurs spécifiques qui lui sont applicables**
- ces opérateurs sont souvent appelés "**méthodes**"
- L'appel d'une "méthode" se fait par envoi d'un "**message**"
- Seul de l'ensemble des opérateurs du type, la "**signature" est visible pour l'utilisateur, l'implantation des opérateurs est cachée.**

Réutilisabilité

- **Polymorphisme (l'encapsulation): une fonction polymorphe (ex: afficher) peut s'appliquer** à une diversité d'objets (image, texte…)

Héritage (abstraction - sous typage)

- Permet de factoriser entre objets:

  Soit des Structures de Données,

  Soit des opérateurs.

- Peut être simple, multiple, sélectif, partiel

Persistance

- Quels objets persistent dans la BD à la suite de l'application qui les a fait naître ?

Structuration complexe

- BDR : impossible de créer de nouveaux types à partir des types prédéfinis.

- BDO : des constructeurs permettent l'extensibilité des types de base

L e SGBD Orienté Objet «O 2 »

- **Fonctionnalités**

• Un SGBD Orienté Objet : **O2 Technology (à l'origine startup INRIA)**

• Un environnement de Programmation Complet : débogueur, navigateur/éditeur dans le schéma et dans la base, une boite à outils : **O2Query, OQL, …**

- Des outils de génération d'interfaces : **O2LooK**

O2 : Types et Valeurs

- **Valeurs :**
- ***tuple (nom: "Dupont", age: 41, marie: true,***
- **enfants: [*tuple (prenom: "Pierre", age: 18),***
- ***tuple (prenom: "Jeanne", age: 14)],***
- **voiture: { *tuple (marque: "renault", modele: "R5", couleur: "rouge"),***
- ***tuple (marque: "volvo", modele: "240", couleur: "blanche") })***

• { [1, 2, 3] , [3, 2, 1] , [2, 3, 1] , [3, 1, 2] }

• les valeurs atomiques : ***25, 41 , "Pierre", true - la valeur nulle : nil***

Les Constructeurs

- **ensemble : set (1,2, 3) ; n-uplet : tuple (nom: "Pierre",age: 18) ; liste : list (12, 14, 12)**

Types :

- Le **type spécifie le type de la valeur des objets :**
- ***tuple (nom: string,* age: *integer,* marie: *boolean,***

enfants: ***list (tuple (prenom: string , age: string),***

voiture: ***set (tuple (marque: string , modele: string , couleur: string),***

- ***Set (list (integer))***

O2 UneClasse

•les **classes peuvent être utilisées comme des types atomiques**

- ***Classe Hôtel***
- *Type tuple (nom: string,*

adresse: *tuple (numéro: integer, rue: string, ville:* ***Ville),***
étoiles: integer)

- *method nombreDEtoiles: integer;*

classe Ville

- *type tuple (nom: string,*

pays: Pays, carte: Carte,

monuments: *set (Monuments),*

hôtels: *set (Hôtel),*

method nombreDeMonuments: integer;

Plan

- **Introduction**
  - Types d'objets et hiérarchies d'héritage
  - Références et associations
  - Collections
  - Méthodes
  - Évolutions de schéma
  - Vues objet
  - Dictionnaire des données
  - Privilèges

Objet et Base de Données

- **Bases de données relationnelles**
  - Paquetages
  - BLOB (*Binary Large Objects)*
  - Structures de données tabulaires uniquement
- **Bases de données objets**
  - Complexité et manque de base théorique du modèle de données
  - Performances limitées
  - Applications spécifiques (CAO, multimédia…)

Objet et Base de Données

- **Bases de données relationnelles-objets**

  – Extension du modèle de données relationnel à l'aide de types de données abstraits

  – Références directes à un OID (réduction du nombre de jointures)

  – Attributs : types structurés et collections

  ⇒ perte de la 1FN (NF2 : *Non First Normal Form)*

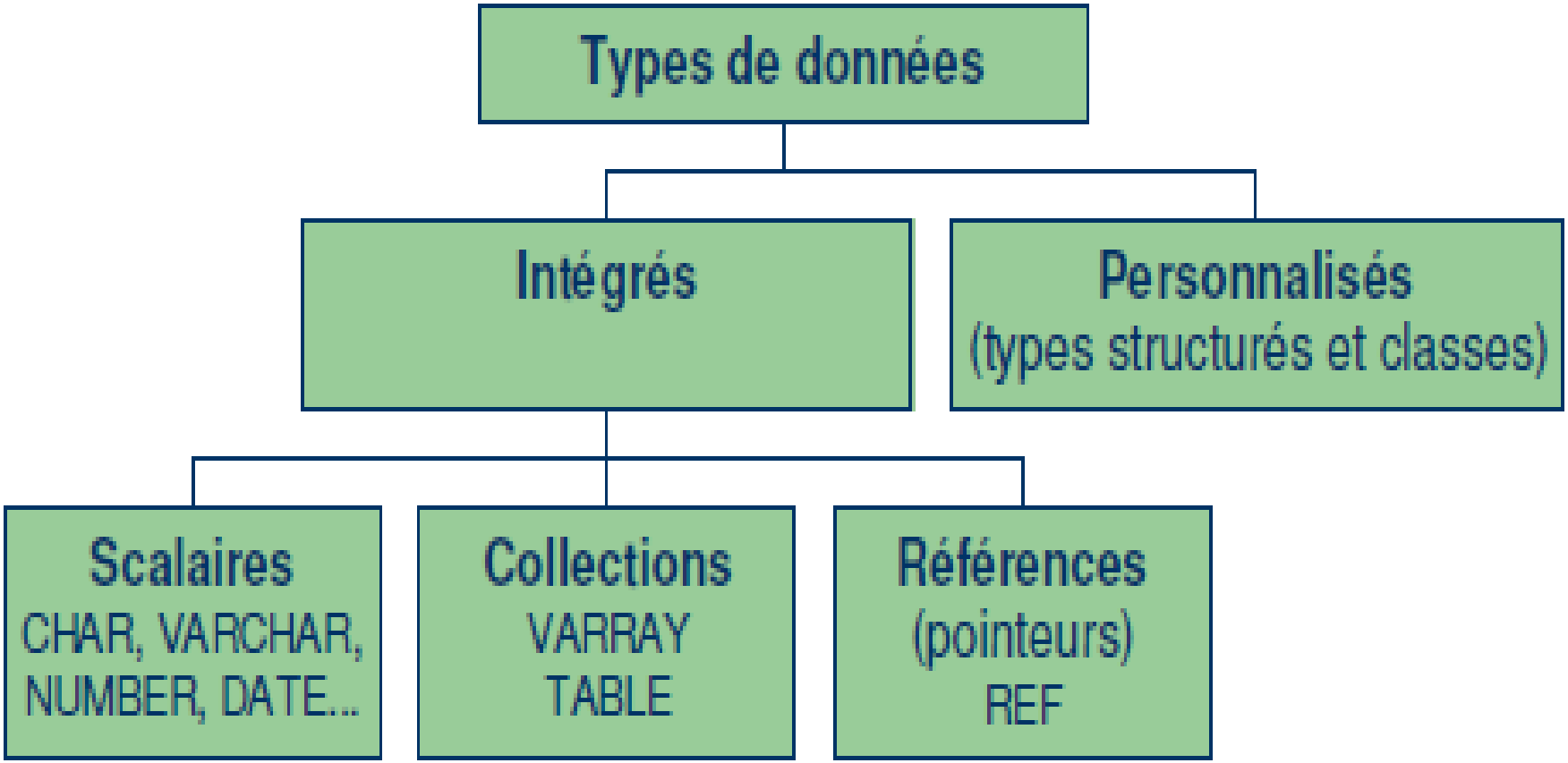
  – Méthodes définies au niveau des types

  – Héritage entre types

Avantages Objet-Relationnel OR

- Encapsulation des données des tables
- Préservation des acquis des systèmes relationnels
- (indépendance données/traitements, fiabilité, performance,
- compatibilité ascendante…)
- Extension du langage SQL (norme SQL3)
- Mise en oeuvre des concepts objets (classes, héritage,méthodes)

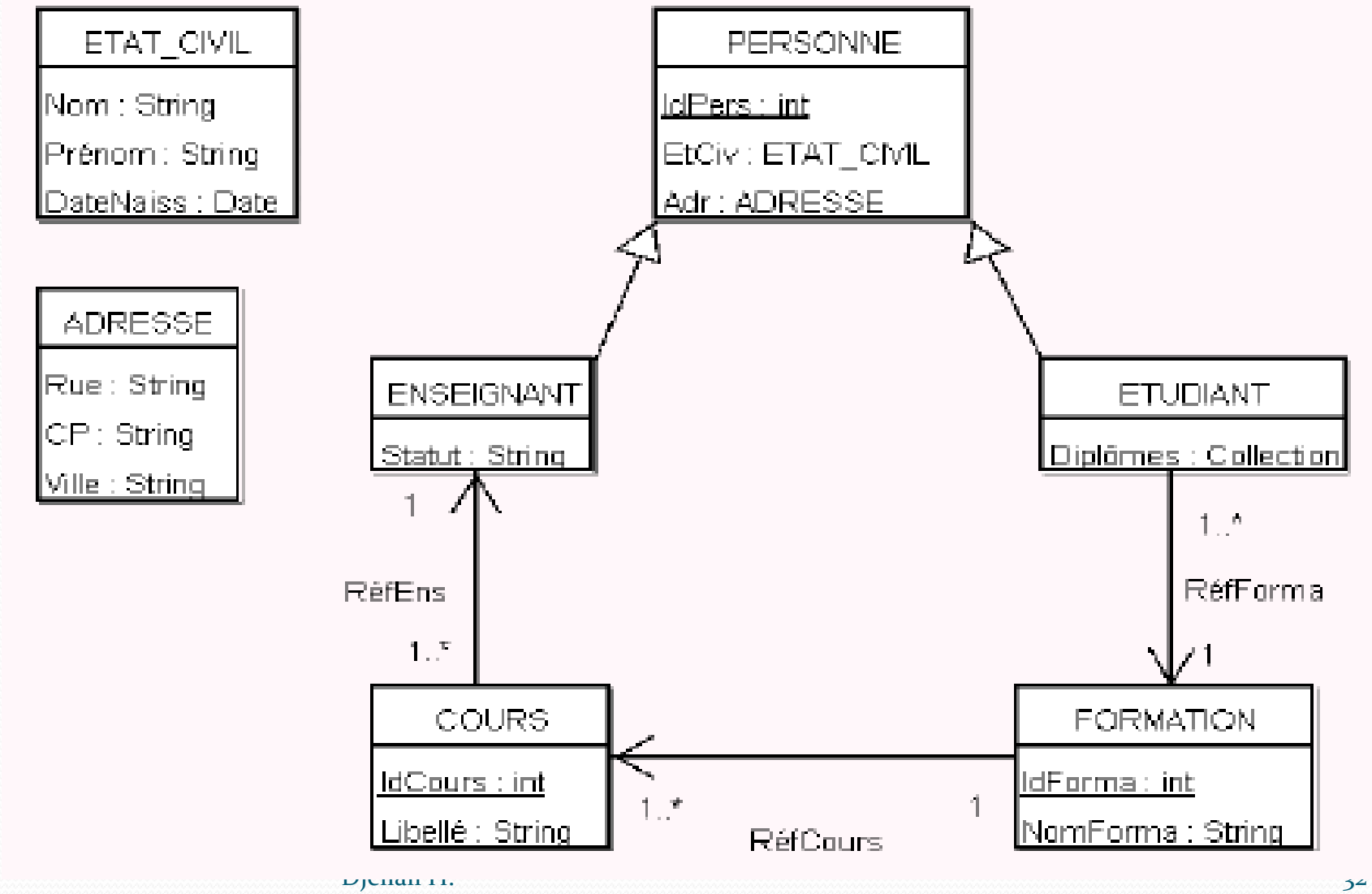
Type de Données



Inconvénient OR

- Modèle de données qui ne repose pas sur des
- principes simples / une théorie rigoureuse
- Pas de syntaxe commune de la part des éditeurs de SGBD (IBM, Oracle, PostgreSQL, SAP…)
- Migration relationnel ➔objet facile,
- mais retour en arrière complexe

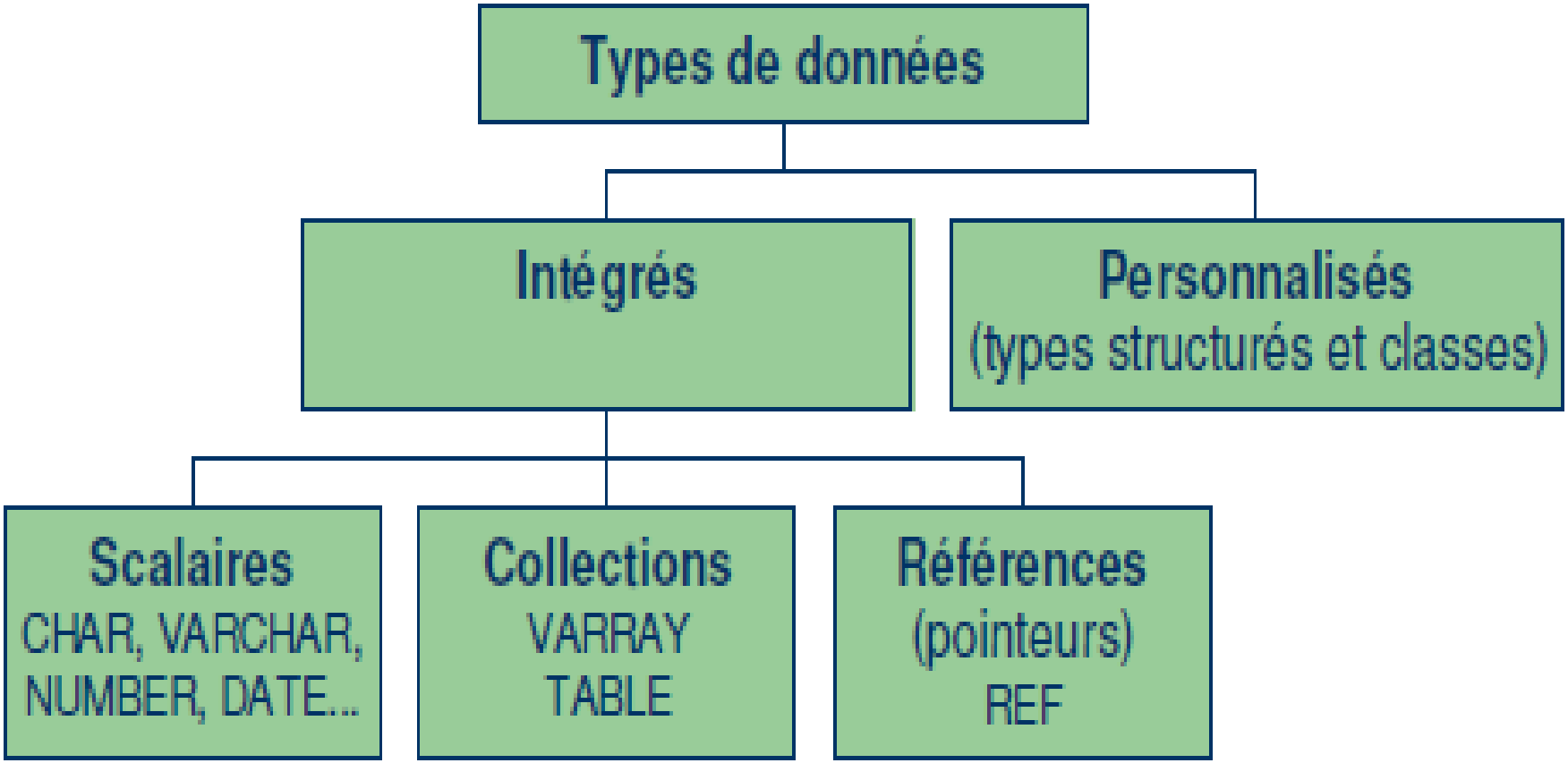
Exemple BD Objet



PLAN

- Introduction
  - **Types d'objets et hiérarchies d'héritage**
  - Références et associations
  - Collections
  - Méthodes
  - Évolutions de schéma
  - Vues objet
  - Dictionnaire des données
  - Privilèges

Type de Données



Types d'objets personnalisés

- CREATE OR REPLACE TYPE T_PERSONNE AS OBJECT (IdPers NUMBER(5),

EtCiV T_ETAT_CIVIL, Adr T_ADRESSE)

/

- **Affichage** de la structure d'un type
  - Ex. DESC T_PERSONNE

  Destruction d'un type
  - Ex. DROP TYPE T_PERSONNE;

Types d'objets personnalisés

- Exemples de création de types

```
CREATE OR REPLACE TYPE T_ETAT_CIVIL AS
   OBJECT (
Nom VARCHAR(50), Prenom VARCHAR(50),
DateNaiss DATE)
/
```

- CREATE OR REPLACE TYPE T_ADRESSE AS OBJECT (Rue VARCHAR(100), CP CHAR(5), Ville VARCHAR(50))

```
/
```

Type Recurssif

- Utilisation d'une référence
- – Ex. CREATE OR REPLACE TYPE T_PERSONNE AS OBJECT (

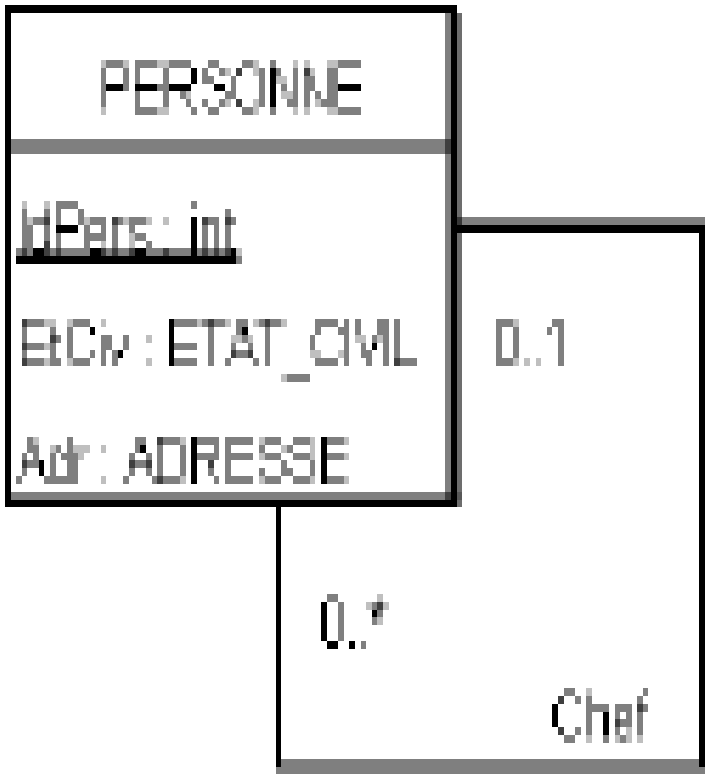
IdPers NUMBER(5),

EtCiV T_ETAT_CIVIL,

Adr T_ADRESSE,

chef REF T_PERSONNE)

- /



Persistance des objets

- **Objets colonnes** : types structurés décrivant des champs de table(s) relationnelle(s)
- **Objets lignes** : stockés en tant que n-uplets d'une table objet.
- **Objets non persistants** : utilisés en mémoire dans un programme PL/SQL

Objets Colonne

- **Table relationnelle**

– Ex: **CREATE TABLE PERSONNE (**
IdPers NUMBER(5) PRIMARY KEY,
EtCiv T_ETAT_CIVIL, Adr T_ADRESSE);

- **Instanciation**

– Ex: **INSERT INTO** PERSONNE VALUES (1000,
T_ETAT_CIVIL('Darmont', 'Jérôme', '15/01/1972'),
T_ADRESSE('5 av. P. M.France', '69676', 'Bron'));

- **Interrogation**

– Ex. SELECT IdPers, p.EtCiv.Nom, p.Adr.Ville
FROM PERSONNE p;

Objet Ligne

- **Table objet**

– Ex. CREATE TABLE PERSONNE OF T_PERSONNE (
CONSTRAINT pers_pk PRIMARY KEY (IdPers));
– Baser l'identifiant objet (**OID**) sur la clé primaire : ajout de la clause OBJECT IDENTIFIER IS PRIMARY KEY

- **Instanciation**
  – Ex. INSERT INTO PERSONNE VALUES (1000,
    T_ETAT_CIVIL('Darmont', 'Jérôme', '15/01/1972'),
    T_ADRESSE('5 av. P. M.France', '69676', 'Bron'));
- **Interrogation**
  – Ex. SELECT IdPers, p.EtCiv.Nom, p.Adr.Ville
    FROM PERSONNE p;

Objet Non Persistant

- **Déclaration PL/SQL**

– Ex. DECLARE
une_personne T_PERSONNE;

- **Instanciation**

– Ex. BEGIN
une_personne := NEW T_PERSONNE (1000,
T_ETAT_CIVIL('Darmont', 'Jérôme', '15/01/1972'),
T_ADRESSE('5 av. P. M.France', '69676', 'Bron'));

- **Utilisation**

– Ex. une_personne.IdPers := 1111;
une_personne.Adr.Ville := 'Lyon';
DBMS_OUTPUT.PUT_LINE(une_personne.EtCiv.Nom);

Mise à Jour d'objets Persistant

- **Modification**
  - Ex. **UPDATE** PERSONNE p
    SET p.Adr.Ville = 'Lyon'
    WHERE IdPers = 1000;
- **Suppression**
  - Ex. **DELETE** FROM PERSONNE p
    WHERE p.EtCiv.Nom = 'Darmont';

Chargement d'objets persistants

- **Dans une requête SQL**

  Ex. SELECT VALUE(p) FROM PERSONNE p;

- **Dans un bloc PL/SQL**

- Ex. DECLARE

```
une_personne T_PERSONNE;
BEGIN
  SELECT VALUE(p)  INTO une_personne
  FROM PERSONNE p
  WHERE IdPers = 1000;
  DBMS_OUTPUT.PUT_LINE(une_personne.EtCiv.Nom);
  END;  /
```

Héritage

- **Superclasse**
- – Ex. CREATE TYPE T_PERSONNE AS OBJECT (

IdPers NUMBER(5), T_EtCiV ETAT_CIVIL,
T_Adr ADRESSE)
NOT FINAL
/

- **Sous-classe**

– Ex. CREATE TYPE T_ENSEIGNANT UNDER T_PERSONNE (
Statut VARCHAR(20))
/

Substitution de types

- **Objets :** Un objet peut contenir une instance de tout

sous-type de son propre type.

- **Références** : Une référence qui cible un type peut peut pointer vers une instance de tout sous-type de ce type.
- **Collections** : Une collection d'éléments d'un type peut contenir des instances de tout sous-type de ce type.

Substitution de types

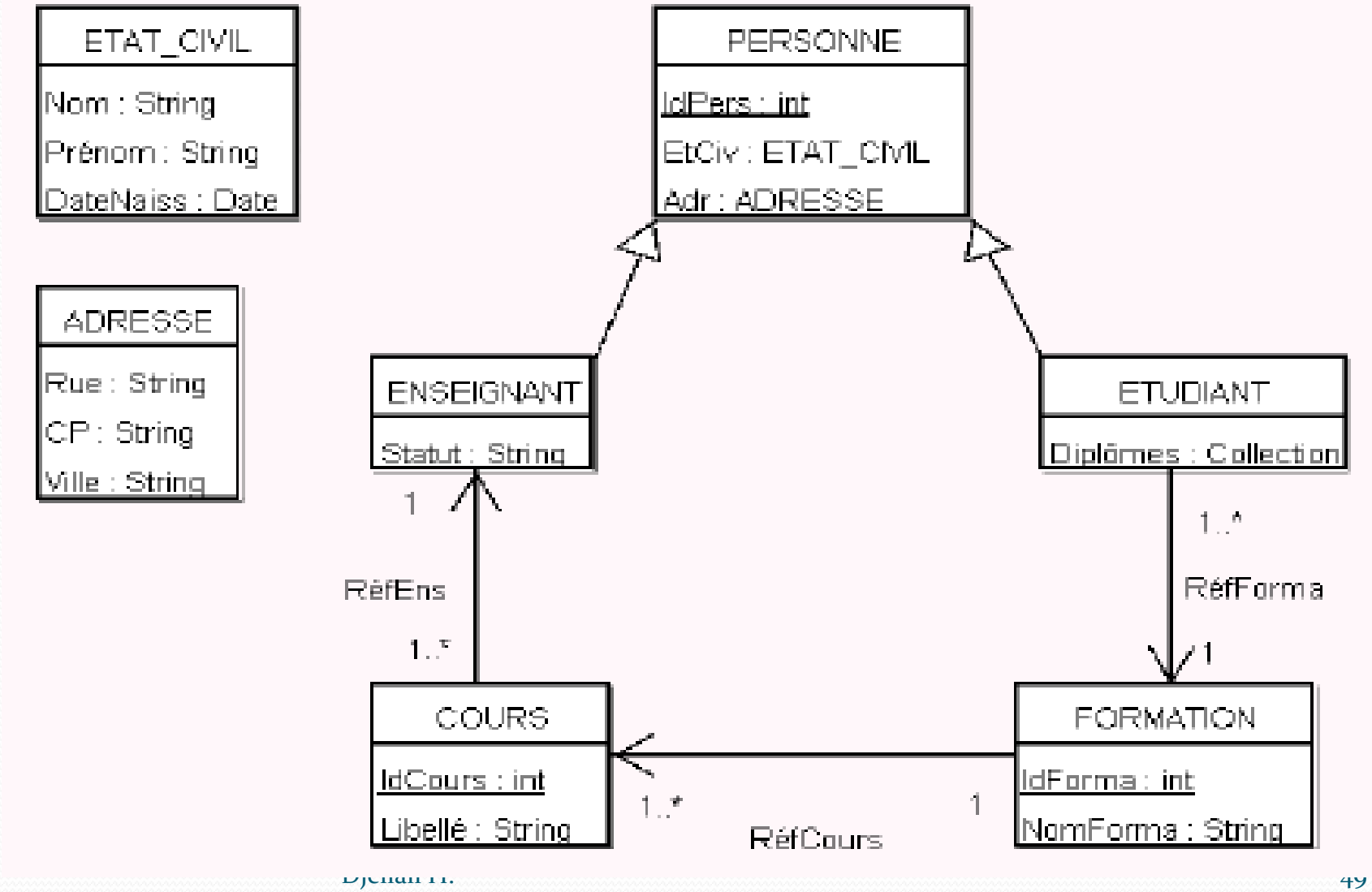
- **Exemple**
- **INSERT INTO PERSONNE VALUES** (T_PERSONNE(2000,
T_ETAT_CIVIL('Durand', 'Maurice', '02/02/1902'),
T_ADRESSE ('Place Bellecourt', '69002', 'Lyon')));

- **INSERT INTO PERSONNE VALUES** (T_ENSEIGNANT(3000,
T_ETAT_CIVIL('Grand', 'Aimé', '01/08/1962'),
T_ADRESSE ('Quai Claude Bernard', '69007', 'Lyon'),
'Professeur'));

Fonctions sur les objets Colonnes

- Adresses des personnes habitant à l'étranger
- SELECT VALUE(p) FROM PERSONNE p
  WHERE p.Adr IS OF(T_ADR_ETRANGER);
- Caractéristiques des personnes habitant à l'étranger

Exemple BD Objet



Mise en oeuvre de références

- **Définition**
- – Ex. CREATE TYPE T_COURS AS OBJECT(

IdCours NUMBER(2),

Libelle VARCHAR(50),

RefEns REF T_ENSEIGNANT)

/

– NB : RefEns pointe vers l'OID d'un enseignant.

- **Instanciation**

– Ex. CREATE TABLE COURS OF T_COURS(

CONSTRAINT pk_cours PRIMARY KEY(IdCours));

– Une référence peut aussi être une colonne d'une table relationnelle.

Mise en oeuvre de références

- **Insertion**
- – Ex. INSERT INTO COURS

VALUES(1, 'Economie', (SELECT REF(e) FROM ENSEIGNANT e WHERE e.IdPers = 9000));

– NB : L'OID de ENSEIGNANT ne doit pas être basé sur la clé primaire pour que la référence fonctionne.

- **Mise à jour**

– Ex. UPDATE COURS c

SET c.RefEns = (SELECT REF(e) FROM ENSEIGNANT e WHERE e.IdPers = 9002)

WHERE c.IdCours = 1;

Jointures implicites

- **Dans la clause WHERE**
- – Ex. SELECT c.Libelle FROM COURS c

WHERE c.RefEns.EtCiv.Nom = 'Darmont';

- **Dans la clause SELECT**

– Ex. SELECT c.RefEns.EtCiv.Nom

FROM COURS c WHERE c.Libelle = 'Économie';

- Déréférencement

– Ex. SELECT DEREF(RefEns) FROM COURS

WHERE IdCours = 1;

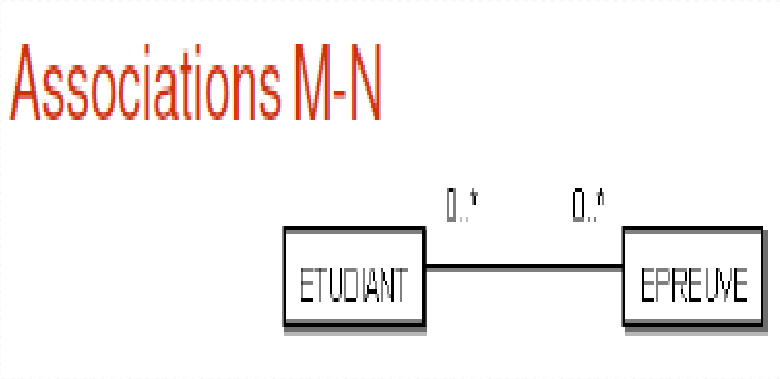
Associations

- Associations 1-N
  - Ex. COURS-ENSEIGNANT
- Associations M-N
  - Ex. CREATE TABLE NOTATION (

    RefEtu REF T_ETUDIANT,

    RefEpr REF T_EPREUVE,

    Note NUMBER(4,1));



Intégrité référentielle

- **Cohérence parent-enfant**
- – Ex. La suppression d'un enseignant auquel des cours sont rattachés doit être impossible.
- **Instanciation**

– Ex. CREATE TABLE COURS OF T_COURS(

CONSTRAINT pk_cours PRIMARY KEY(IdCours),

CONSTRAINT fk_cours_ref_ens1

RefEnsREFERENCES ENSEIGNANT);

Intégrité référentielle

- **Cohérence enfant-parent**
- – Ex. L'insertion d'une référence vers un enseignant inexistant doit être impossible.
- **Instanciation**

– Ex. CREATE TABLE COURS OF T_COURS(
CONSTRAINT pk_cours PRIMARY KEY(IdCours),
CONSTRAINT fk_cours_ref_ens1
RefEns REFERENCES ENSEIGNANT,
CONSTRAINT fk_cours_ref_ens2
CHECK (RefEns IS NOT NULL));

PLAN

- Introduction
  - Types d'objets et hiérarchies d'héritage
  - Références et associations
  - **Collections**
  - Méthodes
  - Évolutions de schéma
  - Vues objet
  - Dictionnaire des données
  - Privilèges

Tables imbriquées

- Définition : collection non-ordonnée et non limitée en taille
- Stockage : dans une table séparée
- Création d'un type TABLE

– Ex. CREATE TYPE TAB_COURS AS

TABLE OF T_COURS

- /

Tableaux prédimensionnés

- Définition : collection ordonnée et limitée en taille
  Stockage : dans un attribut BLOB de la table.
- Création d'un type VARRAY

– Ex. CREATE TYPE TAB_DIPLOMES AS
VARRAY(10) OF VARCHAR(20)

- /

Example NESTED TABLE:

- Créer une table d'objets Groupes qui contient le nom du groupe et ses components (un ensemble d'artistes ayant un nom, prénom et instrument).
  - CREATE TYPE **T_Artiste** AS OBJECT (Nom varchar(32), Prenom varchar(32), Instrument varchar(32));
  - CREATE TYPE **T_Components** AS TABLE OF T_Artiste;
  - CREATE TABLE **Groupes** (NomGroupe varchar(32), Components T_Components) **NESTED TABLE** Components STORE AS GroupeComponents;
  - Pour tester:
    - INSERT INTO Groupes VALUES ('Metallica', T_Components(T_Artiste('James','Hetfield','Vocals'), T_Artiste('Kirk','Hammet','Guitar'), T_Artiste('Jason','Newsted','Bass'), T_Artiste('Lars','Ulrich','Drums')));

Example VARRAY:

- Créer un table de courses qui contient un nom et une liste de participants définis comme un VARRAY de participants (définis par un rang et un nom)
- CREATE TYPE T_Participant AS OBJECT (Rang Number(4), Nom varchar(32));
- **CREATE TYPE** T_ ParticipantList **AS VARRAY(50) OF** T_Participant
- CREATE TABLE Course (
  Course varchar(32),
  ParticipantList T_ParticipantList
  );
- Pour tester:
  - INSERT INTO Course VALUES (
    'Athletissima 100m',
    T_ParticipantList(T_Participant(1,'Maurice Green'), T_Participant(2,'Linford Christie'), T_Participant(3,'Ato Bolton')
)
    );

Stockage : Table Relationnelle

- **Exemple**
- CREATE TABLE ENSEIGNANT (

IdPers NUMBER(5), EtCiV T_ETAT_CIVIL,

Adr T_ADRESSE, Statut VARCHAR(20),

ListeCours TAB_COURS,

CONSTRAINT pk_ens PRIMARY KEY(IdPers))

NESTED TABLE ListeCours

STORE AS TABLE_IMBRIQUEE_COURS;

Stockage : table objet

- Type
- – Ex. CREATE TYPE T_ENSEIGNANT AS OBJECT (
IdPers NUMBER(5), EtCiV T_ETAT_CIVIL,
Adr T_ADRESSE, Statut VARCHAR(20),
ListeCours TAB_COURS)
/
- Table
- – Ex. CREATE TABLE ENSEIGNANTS OF T_ENSEIGNANT
(CONSTRAINT pk_ens PRIMARY KEY(IdPers))
NESTED TABLE ListeCours
STORE AS TABLE_IMBRIQUEE_COURS;

Mises à jour

- **Ajout d'un élément parent**
- – Ex. INSERT INTO ENSEIGNANT VALUES (1000,

T_ETAT_CIVIL('Darmont', 'Jerome', '15/01/1972'),

T_ADRESSE ('5 av. P. M.France', '69676', 'Bron'),

'PR', TAB_COURS(T_COURS(1, 'BD'), T_COURS(2, 'Web')));

- **Ajout dans la collection**

– Ex. INSERT INTO TABLE

(SELECT ListeCours FROM ENSEIGNANT

WHERE IdPers = 1000)

VALUES (T_COURS(3, 'UNIX'));

Mises à jour

- **Modification**

– Ex. UPDATE TABLE
(SELECT ListeCours FROM ENSEIGNANT
WHERE IdPers = 1000) t SET t.Libelle = 'WEB'
WHERE t.Libelle = 'Web';

- **Modification d'éléments**

– Ex. UPDATE TABLE
(SELECT ListeCours FROM ENSEIGNANT
WHERE IdPers = 1000) t
SET VALUE(t) = T_COURS(10, 'Bases de données')
WHERE t.IdCours = 1;

Mises à jour

- Suppression
- – Ex. DELETE FROM TABLE

(SELECT ListeCours FROM ENSEIGNANT

WHERE IdPers = 1000) t

WHERE t.IdCours = 10;

- Note : La clause TABLE n'est pas utilisable avec les tableaux VARRAY, qui ne peuvent être directement modifiés qu'avec PL/SQL.

Caractéristiques des méthodes sous Oracle

- Encapsulation non automatique
- Surcharge possible
- Trois catégories de méthodes
  - Membre (MEMBER) : s'applique à un objet
  - Statique (STATIC) : s'applique à un type (encapsulation)
  - Constructeur (CONSTRUCTOR) : instanciation
- Appel : Requête, autre méthode, PL/SQL,
- programme externe

Déclaration

- Exemple
- CREATE TYPE T_COURS AS OBJECT(
- IdCours NUMBER(2),
- Libelle VARCHAR(50),
- RefEns REF T_ENSEIGNANT,
- MEMBER PROCEDURE ChangeLib (NouvLib VARCHAR),
- STATIC FUNCTION NbCours RETURN NUMBER,
- CONSTRUCTOR FUNCTION T_COURS (
- IdC NUMBER,
- Lib VARCHAR,
- IdE NUMBER) RETURN SELF AS RESULT)

Implémentation

- Exemple

```
CREATE TYPE BODY T_COURS AS
MEMBER PROCEDURE ChangeLib (NouvLib
   VARCHAR) IS
BEGIN
UPDATE les_cours
SET Libelle = NouvLib
WHERE IdCours = SELF.IdCours;
END;
```

Implémentation

- STATIC FUNCTION NbCours RETURN NUMBER IS n INTEGER;

BEGIN

SELECT COUNT(*)

INTO n

FROM les_cours;

RETURN n;

END;

Implémentation

- CONSTRUCTOR FUNCTION T_COURS (IdC NUMBER, Lib VARCHAR, IdE NUMBER) RETURN SELF AS RESULT IS

```
BEGIN
SELF.IdCours := IdC;
SELF.Libelle := Lib;
SELECT REF(e) INTO SELF.RefEns
FROM ENSEIGNANT e
WHERE e.IdPers = IdE;
RETURN;
END;
END;
/
```

Examples:

- Créer un type T_Rectangle charactérisée par deux points et ayant une méthode permettant d'avoir la surface du rectangle
  - CREATE TYPE T_Rectangle AS OBJECT (
    point1 T_Point,
    point2 T_Point,
    MEMBER FUNCTION surface RETURN NUMBER);
  - **CREATE OR REPLACE TYPE BODY** T_Rectangle AS
    MEMBER FUNCTION surface RETURN NUMBER AS
    BEGIN
    RETURN ABS((point1.x-point2.x)*(point1.y-point2.y));
    END surface;
    END;
  - Pour tester:
    - CREATE TABLE Rectangles OF T_Rectangle;
    - INSERT INTO Rectangles VALUES (T_Point(0,0),T_Point(100,100));
    - SELECT R.point1.x, R.point1.x, R.point2.x, R.point2.y, R.surface() FROM rectangles R

Appel

- Exemple

```
DECLARE
un_cours T_COURS;
nb_cours INTEGER;
nouveau_cours T_COURS;
BEGIN
SELECT VALUE(c) INTO un_cours FROM COURS c
WHERE c.IdCours=1;
un_cours.ChangeLib('Informatique');
nb_cours := T_COURS.NbCours();
nouveau_cours := NEW T_COURS(5, 'Statistiques', 1000);
INSERT INTO COURS VALUES(nouveau_cours);
END;
/
```

Surcharge

- Définition multiple d'une méthode
  - Au sein d'une même classe
  - Dans une sous classe
- Exemple 1

```
CREATE TYPE T_COURS AS OBJECT(
-- (…)
STATIC PROCEDURE Ajout(IdC Number,
Lib VARCHAR, IdE NUMBER),
STATIC PROCEDURE Ajout(Lib VARCHAR,
IdE NUMBER))
```

Plan

- Introduction
  - Types d'objets et hiérarchies d'héritage
  - Références et associations
  - Collections
  - Méthodes
  - **Évolutions de schéma**
  - Vues objet
  - Dictionnaire des données
  - Privilèges

Opérations possibles

- Ajout/suppression d'attributs
- Ajout/suppression de méthodes
- Augmentation (taille, précision) d'un attribut
- Modification des caractéristiques d'héritage (FINAL) et de persistance (INSTANTIABLE)

Marche à suivre

- Modifier la structure du type
- ALTER TYPE…
- Recompiler le corps du type
- CREATE OR REPLACE TYPE BODY…
- Rendre les tables conformes
- ALTER TABLE…
- Recompiler les programmes PL/SQL

Vue

Intérêt

Même utilité que les vues relationnelles

- Simplification de l'accès aux données
- Sauvegarde indirecte de requêtes complexes
- Présentation des données variable selon les utilisateurs (sécurité)
- Support de l'indépendance logique

- Migration facilitée vers la technologie objet
  - Utilisation de méthodes sur des données tabulaires
  - Interfaçage aisé avec des langages orientés objets

Type structure de vue

- Exemple
- CREATE TYPE T_ENS_COMPLET AS OBJECT (
- IdPers NUMBER(5),
- EtCiv T_ETAT_CIVIL,
- Adr T_ADRESSE,
- Statut VARCHAR(20))
- /

Création de vue objet

- Exemple
- CREATE VIEW VUE_ENS OF T_ENS_COMPLET

WITH OBJECT IDENTIFIER(IdPers) AS

SELECT p.IdPers, EtCiv, Adr, Statut

FROM PERSONNE p, ENSEIGNANT e

WHERE p.IdPers = e.IdPers;

Vues contenant une collection

- Exemple
- CREATE TYPE T_DIPLOME AS OBJECT(Libelle VARCHAR(20))
- CREATE TYPE TAB_DIPLOME AS TABLE OF T_DIPLOME
- CREATE TYPE T_ETU_DIP AS OBJECT
- (NumPers NUMBER(5), Diplomes TAB_DIPLOME)
- CREATE VIEW VUE_ETU OF T_ETU_DIP

```
WITH OBJECT IDENTIFIER(NumPers) AS
SELECT e.Numpers, CAST( MULTISET(
SELECT Libelle FROM DIPLOME d
WHERE d.NumPers = e.NumPers) AS TAB_DIPLOME
) AS Diplomes  FROM ETUDIANT e;
```

Types, tables objets, collections

- USER_TYPES (TYPE_NAME, SUPERTYPE_NAME, FINAL, INSTANTIABLE...)
- USER_TYPE_VERSIONS (TYPE_NAME, VERSION#...)
- USER_OBJECT_TABLES (TABLE_NAME, OBJECT_ID_TYPE, TABLE_TYPE, NESTED...)
- USER_COLL_TYPES (TYPE_NAME, COLL_TYPE,
- UPPER_BOUND, ELEM_TYPE_NAME...)

Attributs, méthodes

- USER_TYPE_ATTRS (TYPE_NAME, ATTR_NAME,
- ATTR_TYPE_NAME, LENGTH, PRECISION, INHERITED…)
- USER_TYPE_METHODS (TYPE_NAME, METHOD_NAME,
- METHOD_TYPE, FINAL, INSTANTIABLE, OVERRIDING,
- INHERITED…)
- USER_METHOD_PARAMS (TYPE_NAME, METHOD_NAME,
- PARAM_NAME, PARAM_MODE, PARAM_TYPE_NAME…)
- USER_METHOD_RESULTS (TYPE_NAME, METHOD_NAME,
- RESULT_TYPE_NAME…)

Droits d'invocation

- **Exécution des méthodes membres**
- – Avec les privilèges du créateur du type

CREATE TYPE nom_type AUTHID DEFINER

AS OBJECT(…)

/

– Avec les privilèges de l'utilisateur courant

CREATE TYPE nom_type AUTHID CURRENT_USER

AS OBJECT(…)

/

Merci de votre attention

Substitution de références(48)

- **Exemple**
- CREATE TYPE T_REF_PERS(Ref REF T_PERSONNE) /

- CREATE TABLE INDEX_PERS OF T_REF_PERS;
- INSERT INTO INDEX_PERS VALUES
(SELECT REF(p) FROM PERSONNE p
WHERE p.IdPers = 1000);

- INSERT INTO INDEX_PERS VALUES
(SELECT REF(e) FROM ENSEIGNANT e
WHERE e.IdPers = 9000);