

BASES DE DONNEES ORIENTEES OBJET

LE RELATIONNEL 'OBJET'

PRE-REQUIS

- **BASES DE DONNEES RELATIONNELLES**
- **PARADIGME OO**
- **LPOO**

- **SQL**
- **PL/SQL**

BIBLIOGRAPHIE

- 1. Introduction aux bases de données,
Chris J. Date, Vuillebert,
6me Edition, 1998.**
- 2. Bases de données objet et relationnel,
G. Gardarin,
Eyrolles, 1999**
- 3. Les objets, M. Bouzeghoub,
G. Gardarin, P. Valduriez,
Eyrolles, 1998**
- 4. Objet-relationnel sous Oracle8,
C. Soutou,
Eyrolles, 1999**
- 5. WWW.OMG.ORG**

SOMMAIRE GENERAL

- **I. Introduction : pourquoi les bases de données objet**
- **II. L'apport des SGBDOO**
- **III. Le relationnel étendu**
- **IV. Manipulation du relationnel étendu: Oracle (TP)**
- **V. Conclusion**

- **I. Introduction : pourquoi les bases de données objet**
- **II. L'apport des SGBDOO**
- **III. Le relationnel étendu**
- **IV. Manipulation du relationnel étendu: Oracle (TP)**
- **V. Conclusion**

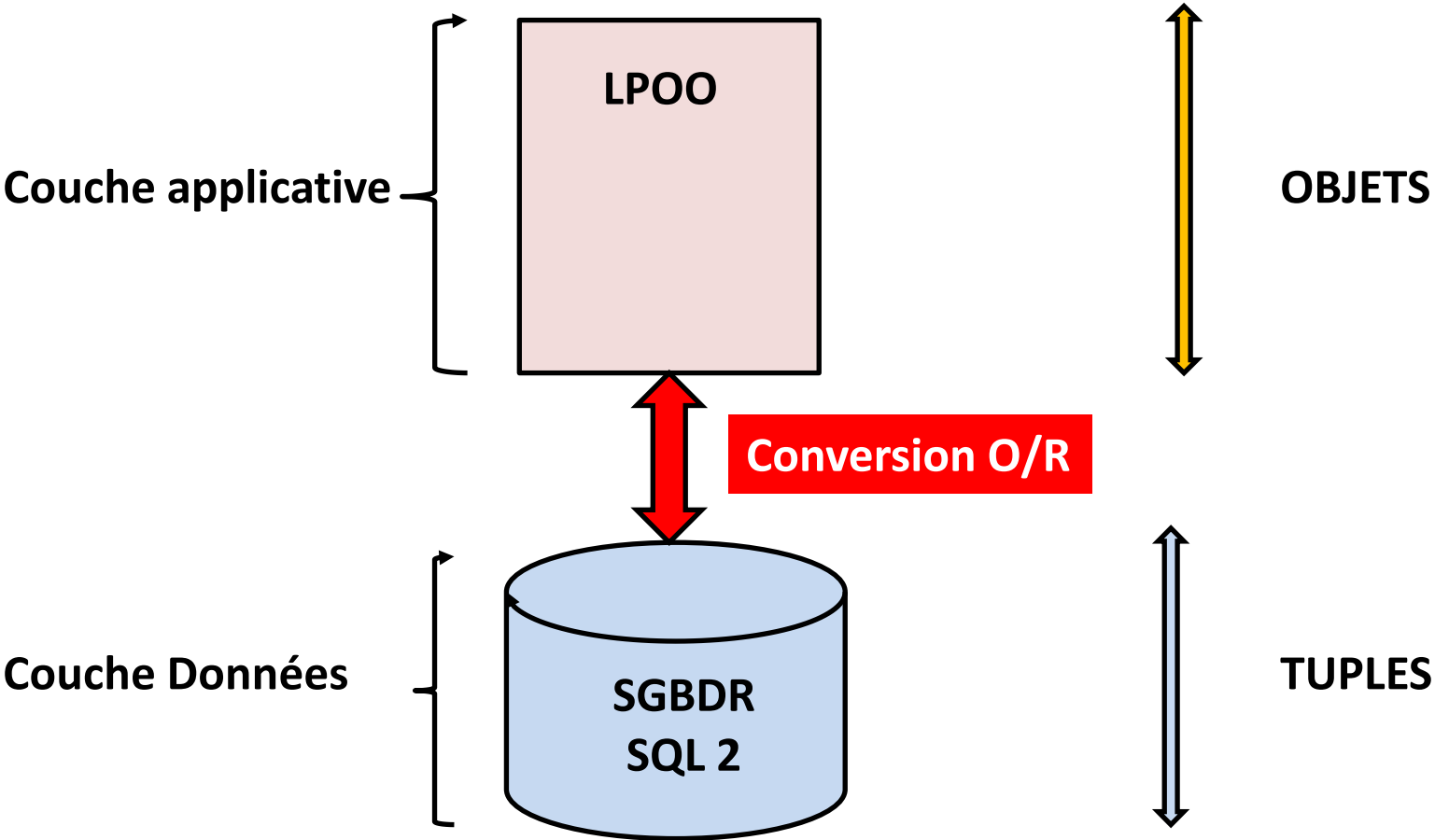
MOTIVATION

PRINCIPE DES BASES DE DONNEES:

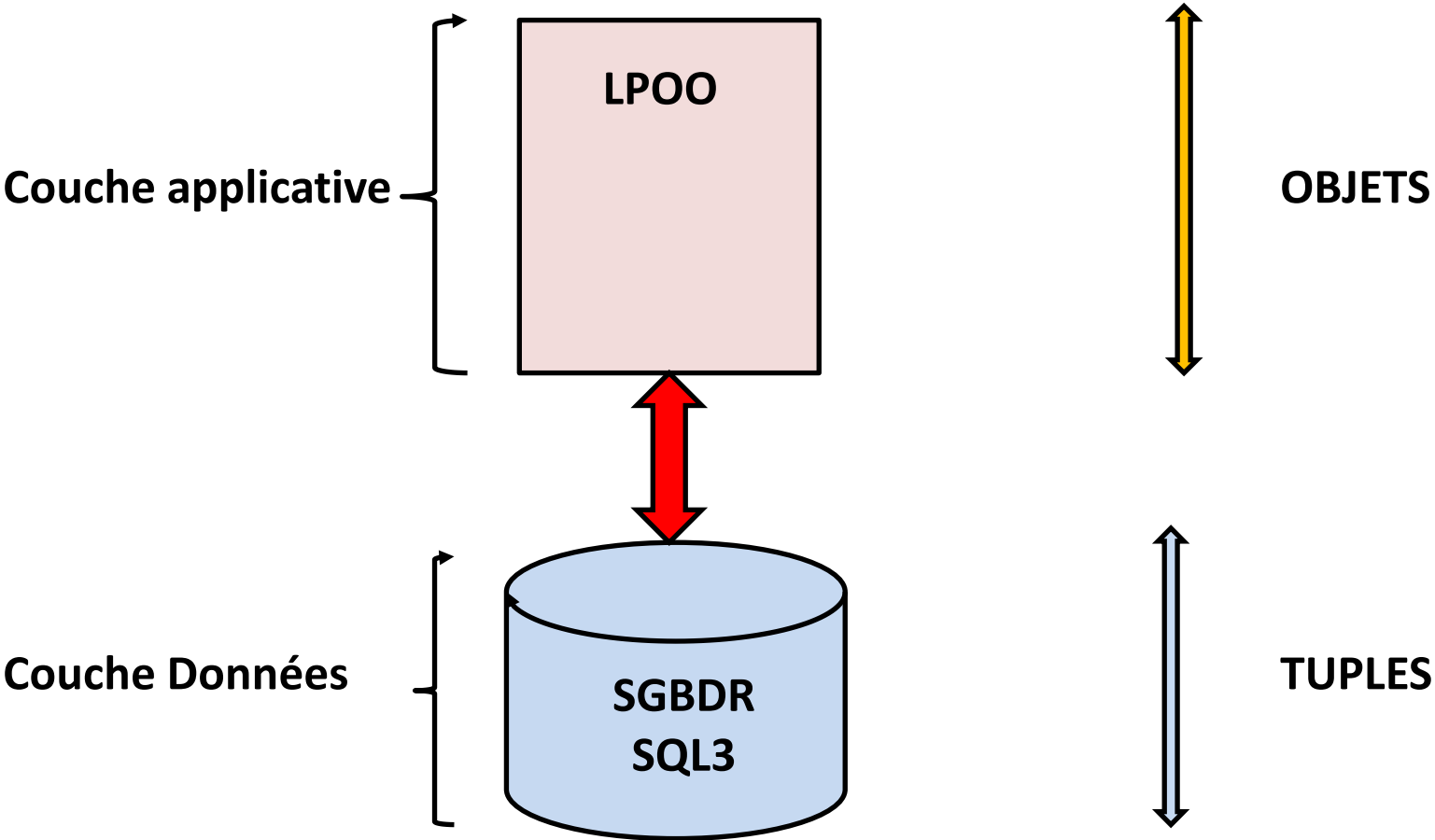


PRINCIPE DES BASES DE DONNEES

Introduction: Pourquoi les SGBDOO???



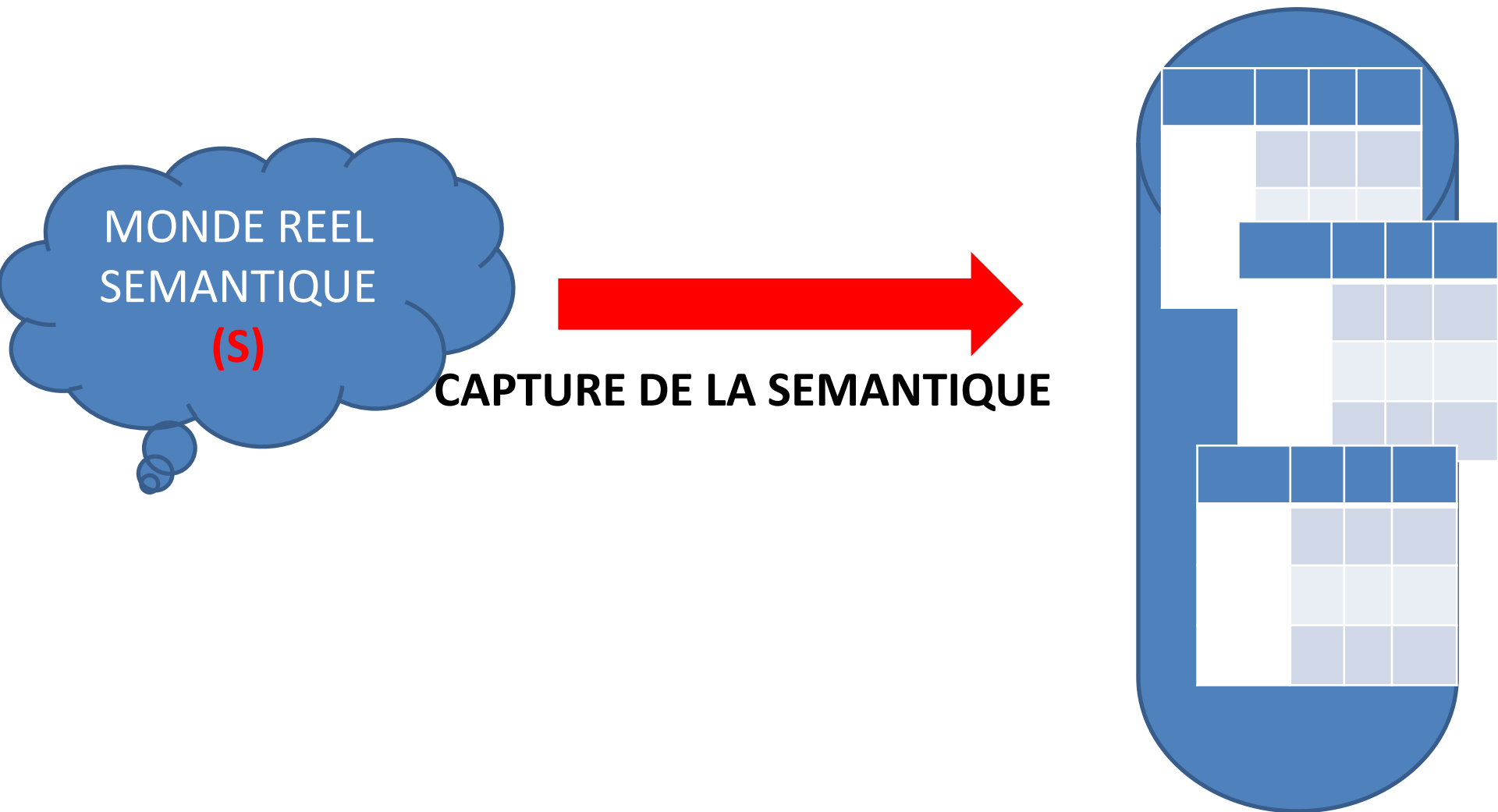
Introduction: Pourquoi les SGBDOO???



CRITIQUE DU MODELE RELATIONNEL

**PRENONS UN PEU DE REcul
VIS-À-VIS DU
RELATIONNEL!!!!!!**

PRINCIPE DES BASES DE DONNEES:



LIMITES DU MODELE RELATIONNEL : structures de données

PERSONNES	NUMP	NOM	PRENOM	DTNAIS	PERE	MERE	CONJOINT
	111	toto	titi		325	125	333
	325						
	125						
	896				111	333	
	333	tata					111

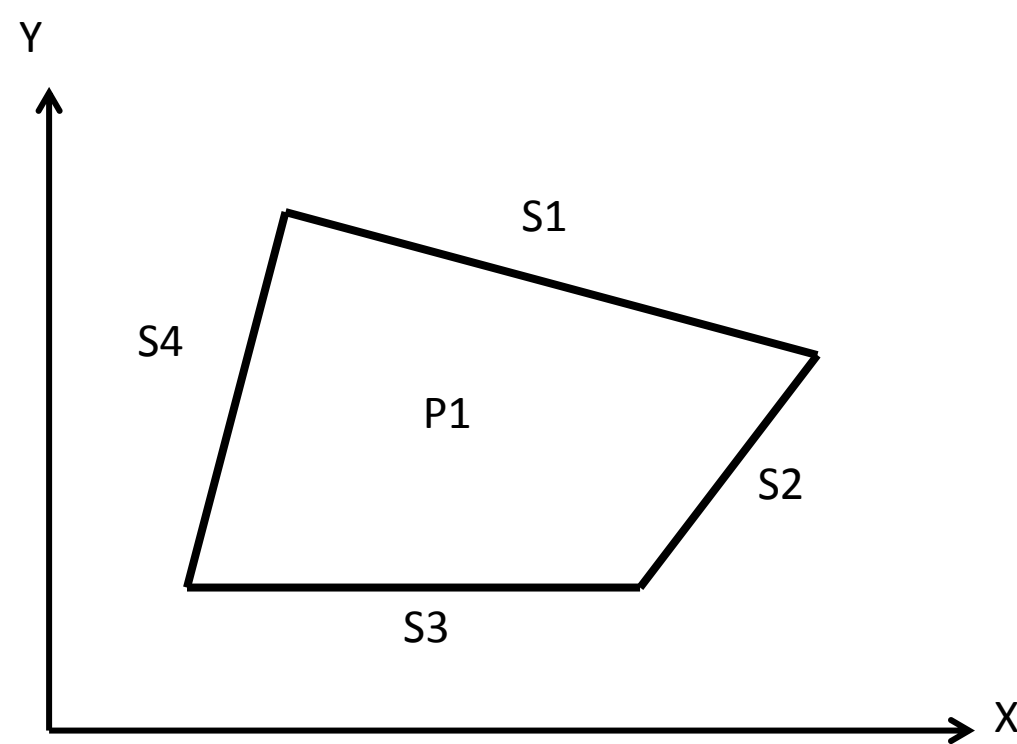
Structures de données élémentaires:
Table, ligne.

Type de données atomiques élémentaires:
NUMBER, VARCHAR, CHAR, DECIMAL, DATE, TIME ,ETC.



∀ La sémantique du monde réel modélisé, il faut pouvoir l'exprimer avec ces structures du modèle relationnel

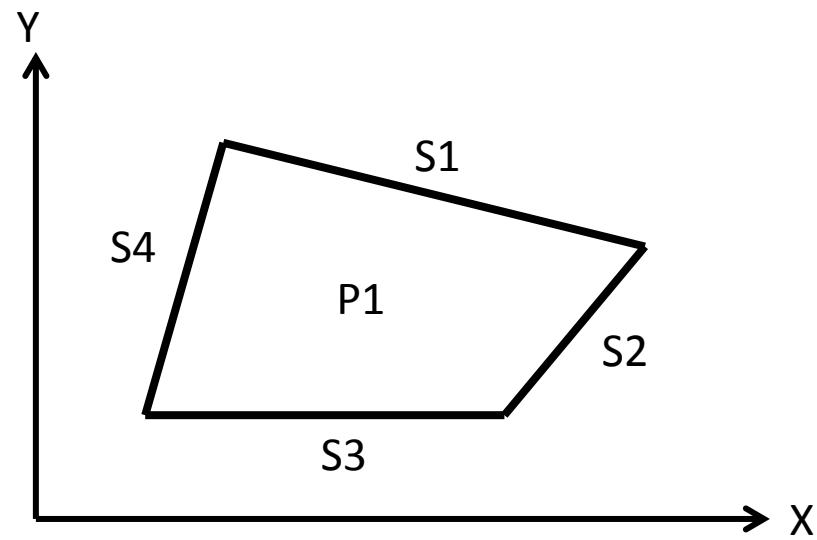
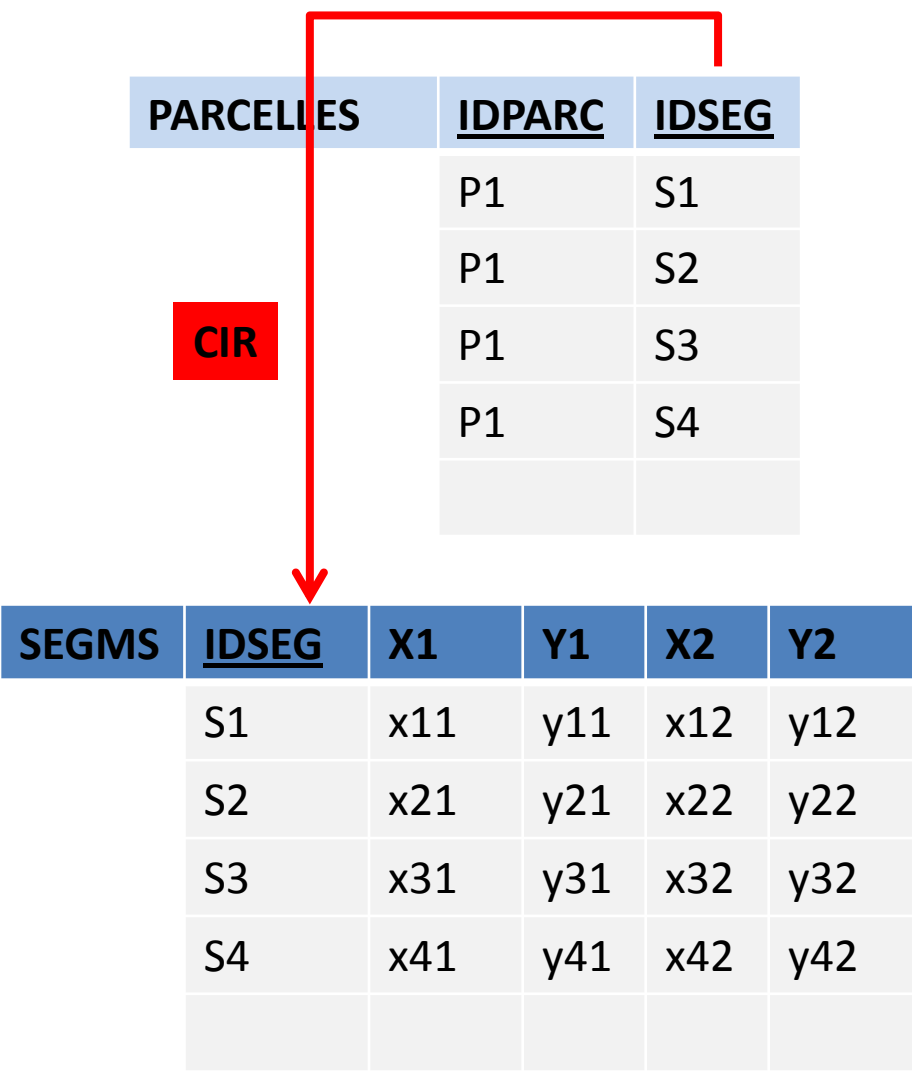
LIMITES DU MODELE RELATIONNEL : Capture de la sémantique



ENTITE SEMANTIQUE ELEMENTAIRE DANS LE MONDE MEDELISE: PARCELLE DE TERRAIN

COMMENT IMPLEMENTER CETTE ENTITE DANS LE MODELE RELATIONNEL???

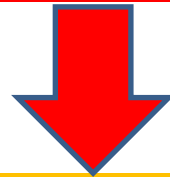
LIMITES DU MODELE RELATIONNEL : Capture de la sémantique



2 TABLES ET
UNE CONTRAINTE D'INTEGRITE REFERENTIELLE
ET
APRES????????????????!!!!!!!!!!!!!!

LIMITES DU MODELE RELATIONNEL : Capture de la sémantique

Le modèle relationnel SUBDIVISE une entité sémantique du monde réel en plusieurs morceaux et stocke chaque morceau dans des tables différentes



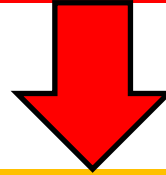
La contrainte d'intégrité référentielle pour garder le lien entre les morceaux de la même entité sémantique



L'opérateur de jointure pour "RECOLLER LES MORCEAUX"

**ET LA SEMANTIQUE DANS TOUT CA??
LE RELATIONNEL EST-IL CAPABLE DE ME RESTITUER TOUTES LA SEMANTIQUE AVEC
SES OUTILS FOURNIS???**

La surface de la parcelle P1???



```
SELECT S.IDSEG, S.X1, S.Y1, S.X2, S.Y2  
FROM SEGMS S, PARCELLES P  
WHERE S.IDSEG=P.IDSEG AND P.IDPAR=P1;
```

AVEC SQL LE RELATIONNEL N'EST MEME PAS CAPABLE DE RESTITUER LA SEMANTIQUE DU DEPART!!!!



IL FAUT UN AUTRE LANGAGE DE PROGRAMMATION (PL)

SQL est un langage INCOMPLET DANS LE SENS DE LA CALCULABILITE

DONC, POUR FAIRE LES BD, IL FAUT MAITRAISER AU MOINS 2 LANGAGES:

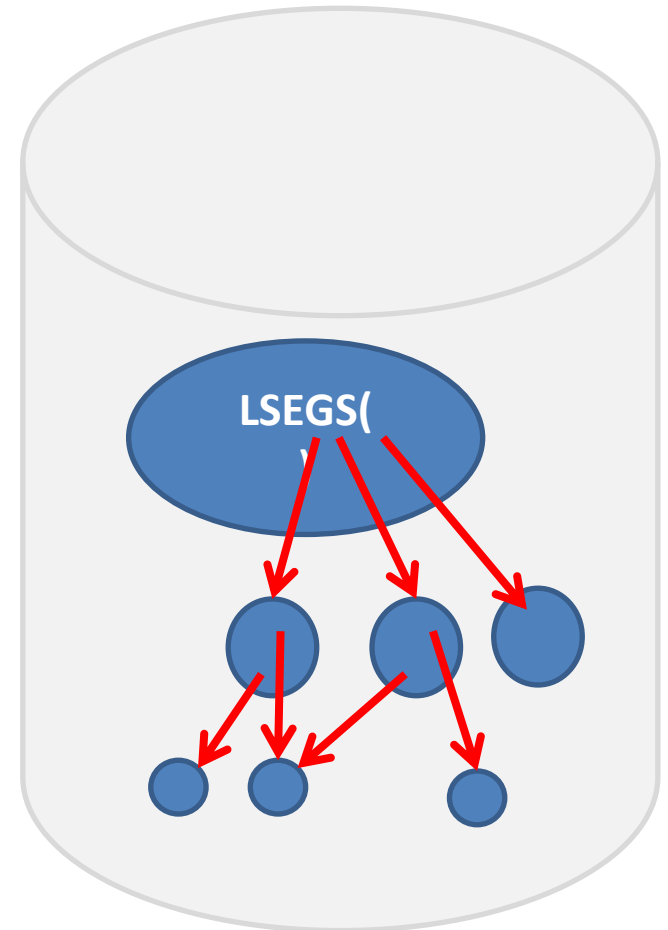
- **SQL POUR LES ACCES AUX DONNEES DANS LA BASE**
- **UN AUTRE LANGAGE DE PROGRAMMATION (PL) POUR PALIER AUX INSUFFISANCES DE SQL**

LIMITES DU MODELE RELATIONNEL : Capture de la sémantique

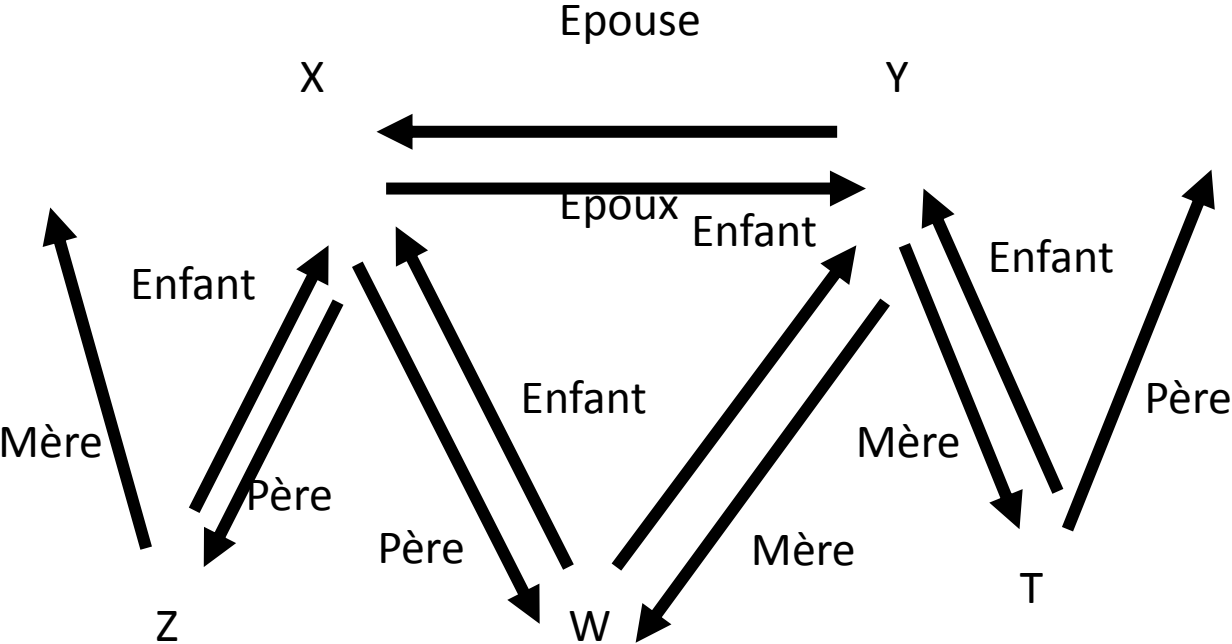
```
Classe CPARCELLES{  
    private int          IDPARC;  
    private Liste<CSEGMS> LSEGS;  
    .....  
}
```

```
Classe CSEGMS{  
    private int          IDSEG;  
    private CPOINTS      PTS1;  
    private CPOINTS      PTS2;  
    ..... }  
}
```

```
Classe CPOINT{  
    private int          IDPOINT;  
    private int          X;  
    private int          Y;  
    .....  
}
```



LIMITES DU MODELE RELATIONNEL : Capture de la sémantique



LIMITES DU MODELE RELATIONNEL : Capture de la sémantique

PERSONNES	NUMP	NOM	PRENOM	DTNAIS	PERE	MERE	CONJOINT
	111				325	125	333
	325						
	125						
	896				111	333	
	333						111

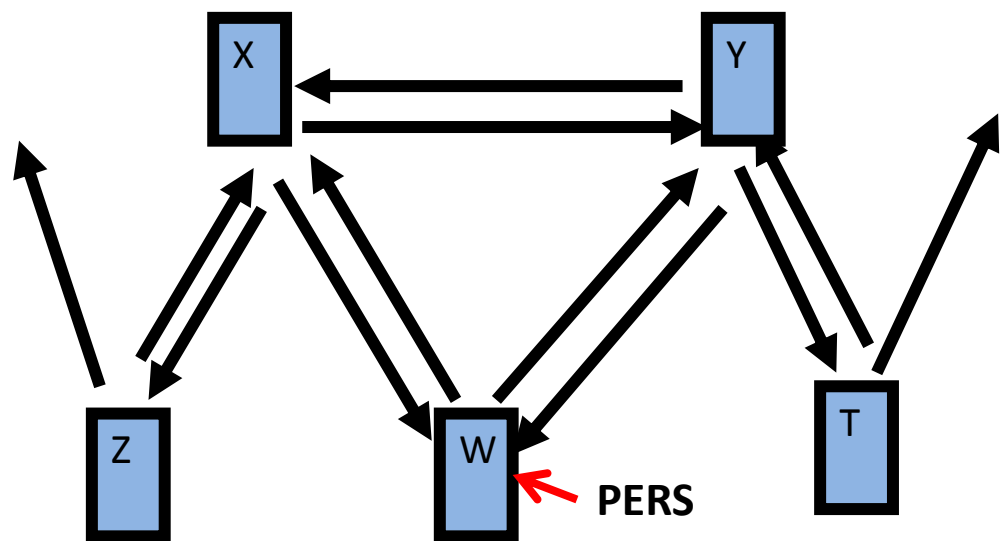
```
graph TD
    111 --> 325
    111 --> 125
    111 --> 333
    325 --> 125
    125 --> 896
    896 --> 333
    333 --> 111
```

Les frères et sœurs (à part entière) de la personne numéro 896??

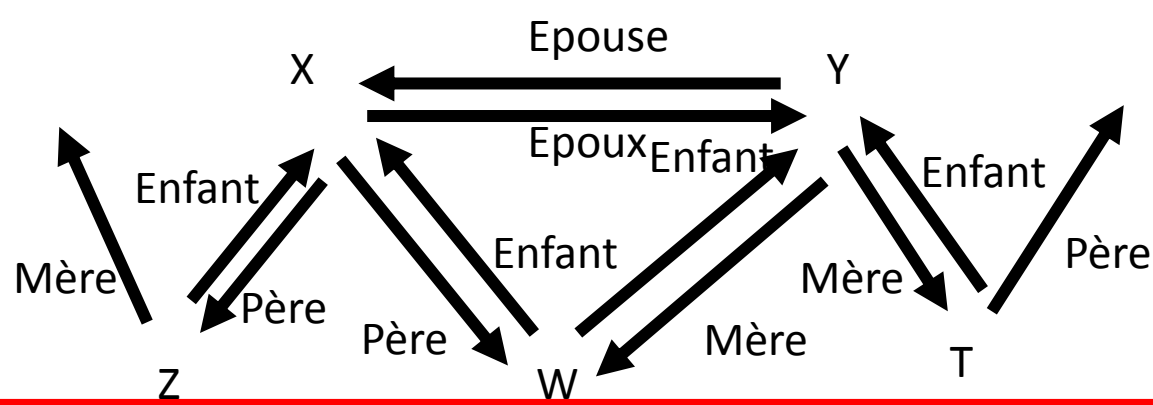
```
SELECT P1.NUMP
FROM PERSONNES P1, PERSONNES P2
WHERE P1.PERE=P2.PERE AND P1.MERE=P2.MERE AND P2.NUMP=896;
```

LIMITES DU MODELE RELATIONNEL : Capture de la sémantique

```
Class Personne{
private Integer      NUMP;
private String       NOM;
private String       PRENOM;
private Date         DTNAIS;
private Personne     PERE;
private Personne     MERE;
private Personne     CONJOINT;
private List<Personne> ENFANTS;
.....
.....
}
```



PERS.PERE.getEnfants() \cap PERS.MERE.getEnfants();



LIMITES DU MODELE RELATIONNEL : NOUVELLES APPLICATIONS



LIMITES DU MODELE RELATIONNEL : CONCLUSION

Insuffisance du modèle relationnel

Pauvreté des structures de données (tuple, table, integer, ...)

Faible capture de la sémantique (intégrité référentielle)

Incomplétude du SQL

Problème SQL/langage de programmation

Nouveaux besoins

Nouveaux types de données (image, dessin, vidéo, spatial..)

Nouvelles applications (bd multimédia, bd spatiale, etc.)

- I. Introduction : pourquoi les bases de données objet
- **II. L'apport des SGBDOO**
- III. Le relationnel étendu
- IV. Manipulation du relationnel étendu: Oracle (TP)
- V. Conclusion

**DEFINITION D'UN
SGBDOO (la norme)**

DEFINITION D'UN SGBDOO : NORME OMG

Aspect Bases de Données

- **Persistence**
- **Gestion de mémoires**
- **Langage de requêtes**
- **Multi-utilisateurs**
- **Fiabilité**
- **Sécurité**

Aspect Orienté Objet

- **Identité d 'objet**
- **Objet complexe**
- **Encapsulation**
- **Type ou Classe**
- **Héritage**
- **Surcharge et résolution tardive**
- **Complétude**
- **Extensibilité**

DEFINITION D'UN SGBDOO : NORME OMG

Les règles facultatives

- Héritage multiple
- Généricité
- Vérification de type
- Distribution
- Transaction de conception
- Versions

Les règles ouvertes

- Modèle de calcul
- Modèle de persistance
- Uniformité
- Nommage

DEFINITION D'UN SGBDOO : NORME OMG

Aspect Bases de Données

- **Persistence**
- **Gestion de mémoires**
- **Langage de requêtes**
- **Multi-utilisateurs**
- **Fiabilité**
- **Sécurité**

DEFINITION D'UN SGBDOO

Aspect Orienté Objet

- **Identité d 'objet**
- **Objet complexe**
- **Encapsulation**
- **Type ou Classe**
- **Héritage**
- **Surcharge et résolution tardive**
- **Complétude**
- **Extensibilité**

**IDENTITE D'OBJET
et
PERSISTANCE**

Problème???

La persistance=les objets survivent à l'application qui les a créés.

La persistance est à la charge du SGBD.

Comment reconnaître les objets d'une façon unique est sûr???

Comment rendre les objets persistants?

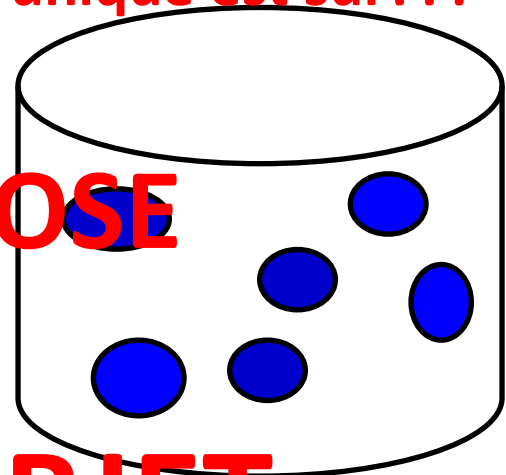
~~La valeur de l'objet?~~

IL FAUT AUTRE CHOSE

~~L'adresse physique de l'objet?~~

L'IDENTITE DE L'OBJET

~~Notion de clé~~



Disque Dur

- Les objets sont permanents dans la BD
- C'est le SGBD qui gère les objets dans la BD

- Chaque objet a une identité (**OID**) indépendante de:

- Sa valeur
- Son type ou classe
- Sa localisation physique



- L'identité d'un objet est:

- UNIQUE
- IMMuable
- PERMANENTE

OBJET=<OID, VALEUR>

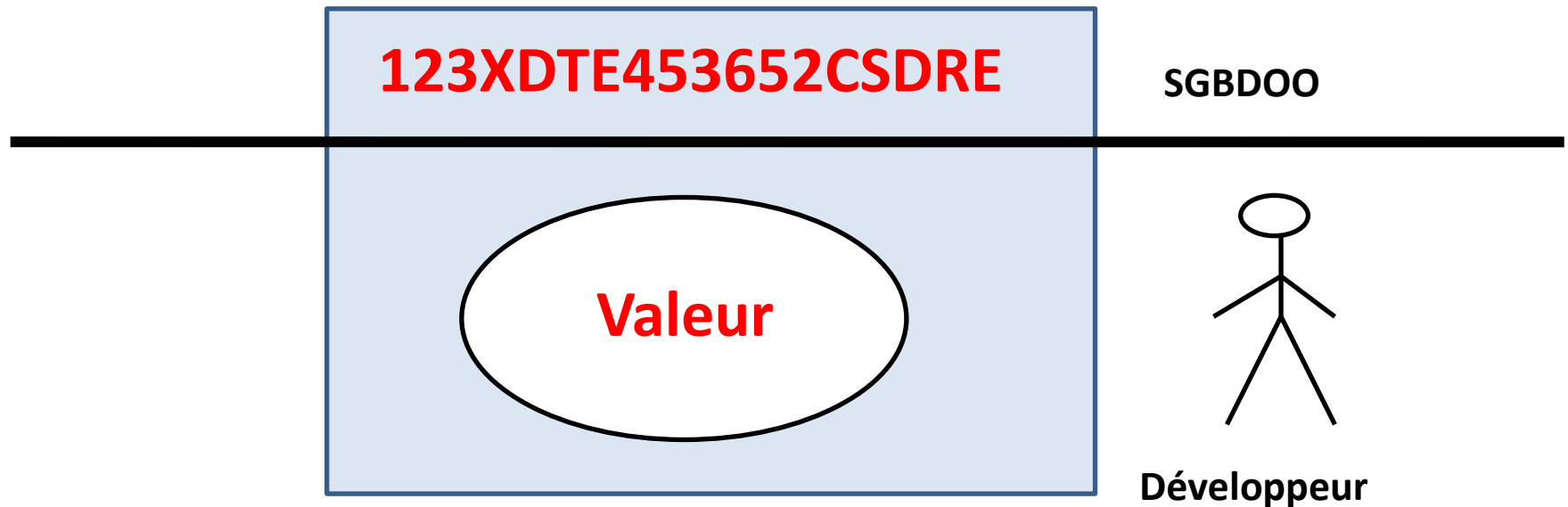
- L'identité d'un objet est attribuée par le SGBD et gérée uniquement par le SGBD

- L'identité d'objet n'est pas accessible par le développeur

La base de données=les valeurs des objets

Le développeur voit la base de données=les valeurs des objets

Le SGBDOO voit les objets=les OID des objets



Quelles sont les conséquences sur notre façon de faire les BD????

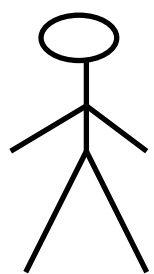
Conséquence 1: deux visions

Class A (a : type1,
b : **B**)

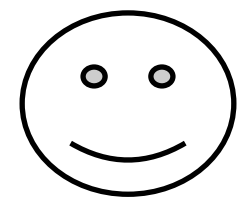
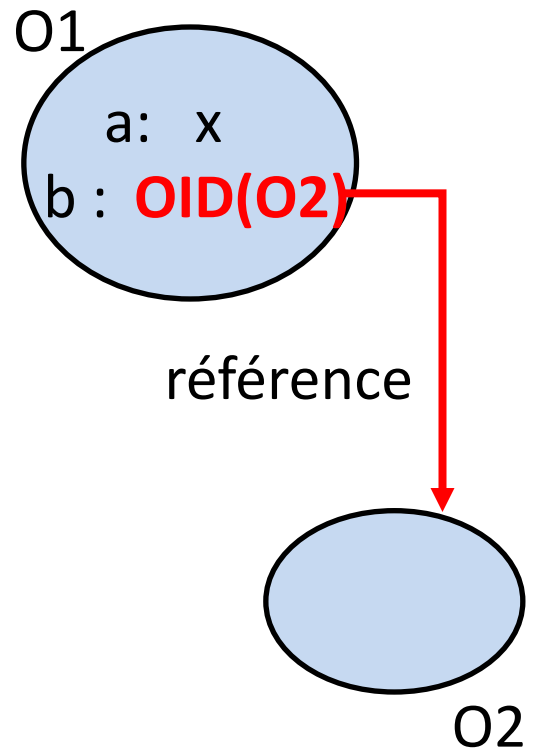
Class B (c : type2,
d : type3)

A O1 : new A,
B O2 : new B
O1.a:=x;

O1.b:=O2;



Développeur



SGBD OO

Conséquence: deux visions

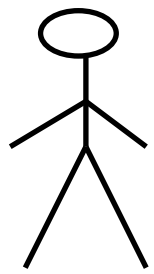
Class A (a : type1,
b : **B**)

Class B (c : type2,
d : type3)

PAS DE CONTRAINTNE D'INTEGRITE
REFERENTIELLE

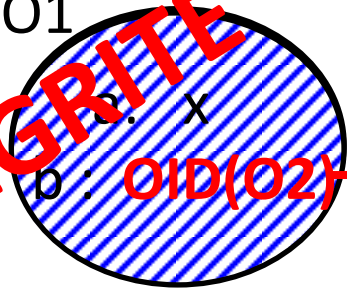
O1.a

O1.b

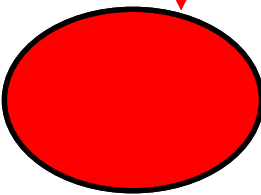


Développeur

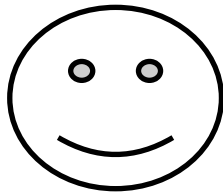
O1



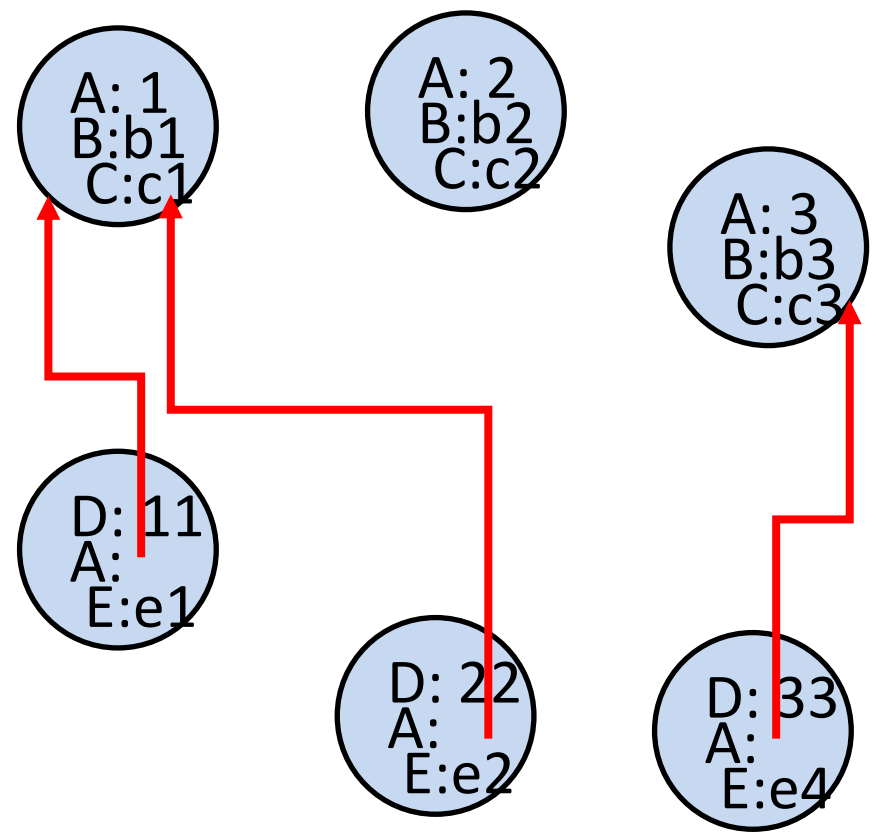
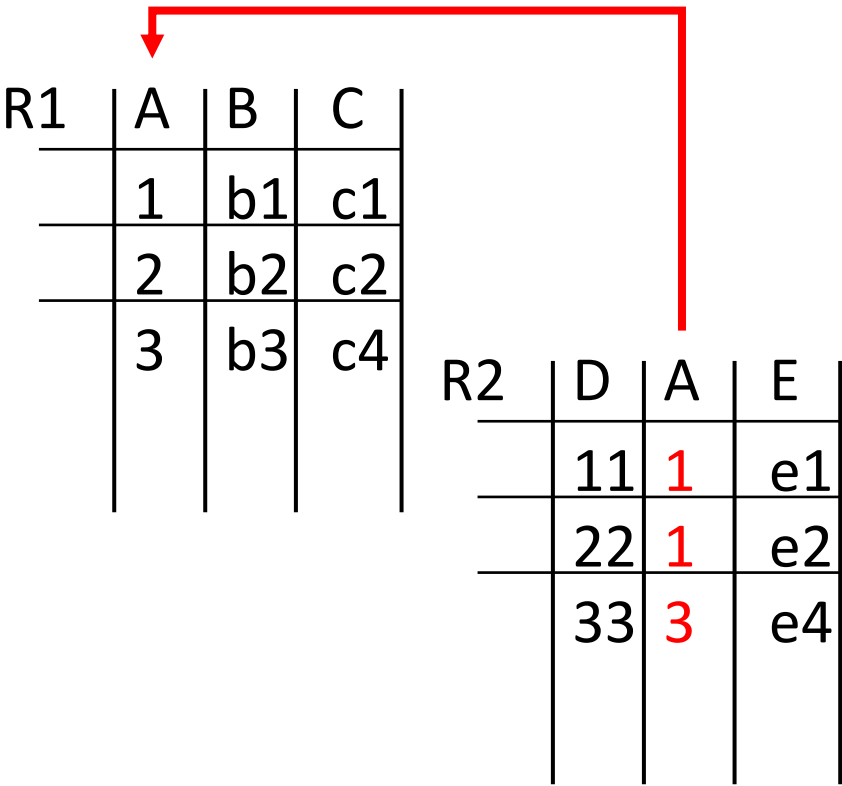
référence



O2



SGBD OO



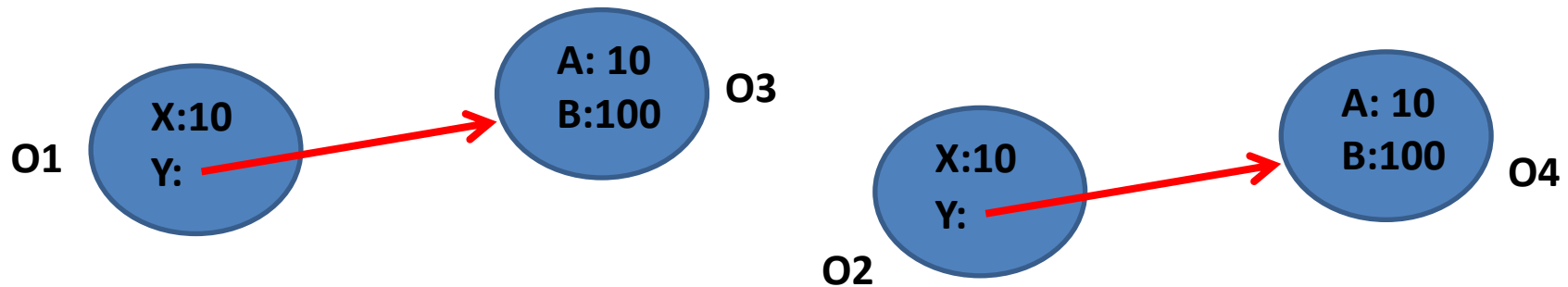
Modèle relationnel: système à valeur

Modèle objet: système à objet

IDENTITE ET EGALITE

La sémantique de la BD est dans les valeurs des objets

Les objets peuvent contenir des références vers d'autres objets



- Test d'identité ==
- Test d'égalité « de surface » =
- Test d'égalité « profonde » =* (deep equality)
(égalité arborescente)

IDENTITE ET EGALITE

Class A (a : entier, b : B)
Class B (c : entier, d : entier)

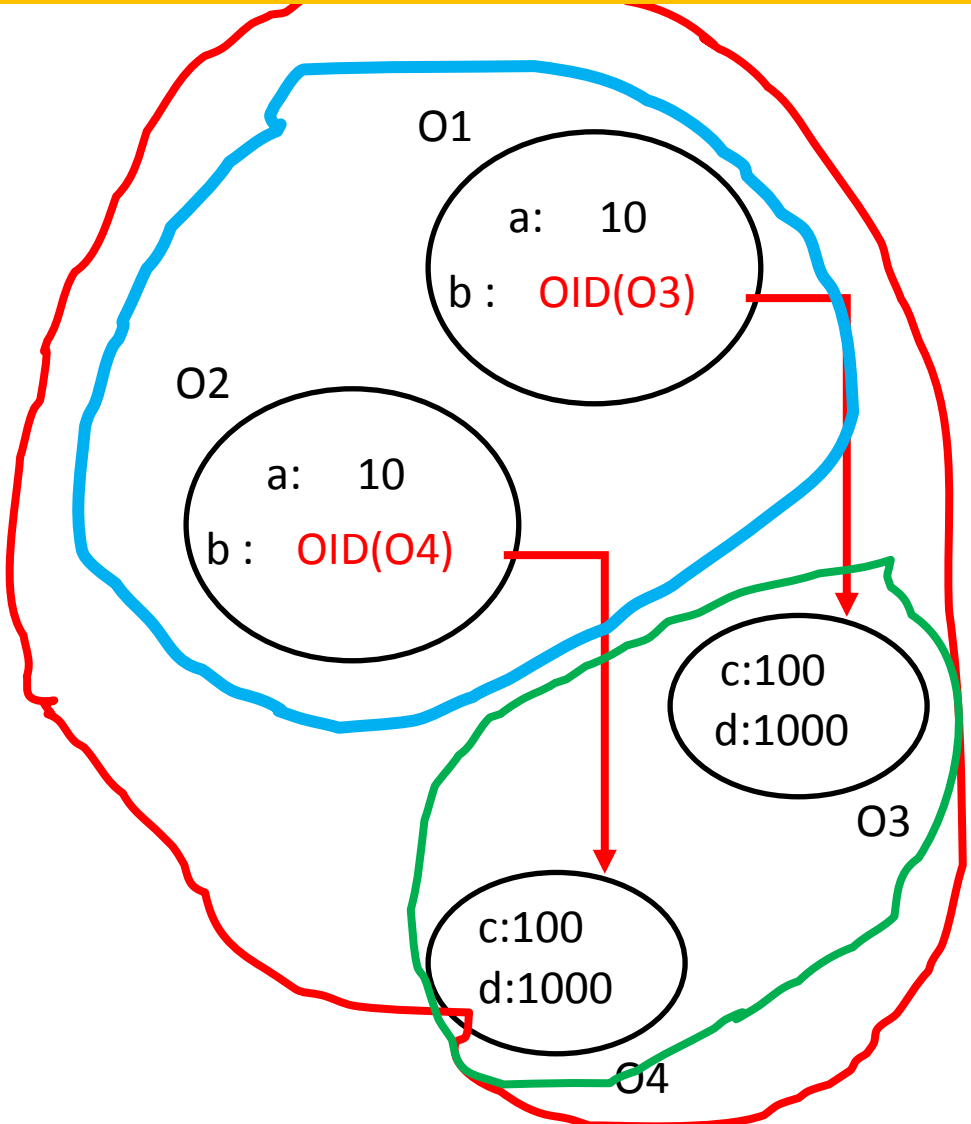
```
A O1 : new A;
A O2 : new A;
B O3 : new B;
B O4 : new B;

O2.a:=10;
O2.b:=O4;

O3.c:=100;
O3.d:=1000;

O4.c:=100;
O4.d:=1000;

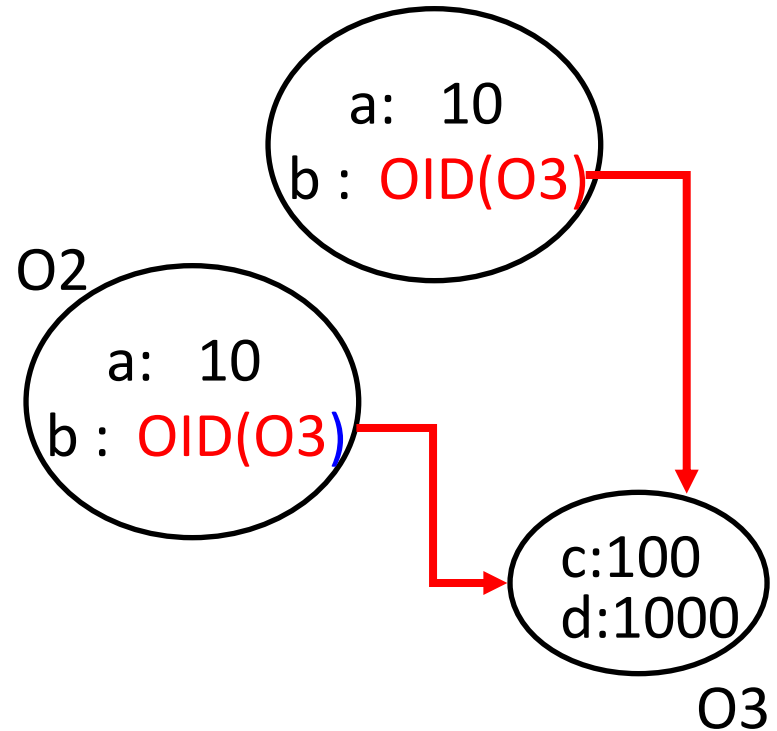
O1.a:=10;
O1.b:=O3;
```



IDENTITE ET EGALITE

Class A (a : entier, b : **B**)
 Class B (c : entier, d : entier)

A O1 : new A;	O2.a:=10;
A O2 : new A;	O2.b:= O3 ;
B O3 : new B;	
B O4 : new B;	O1=O2?
O3.c:=100;	
O3.d:=1000;	O1=*O2?
O1.a:=10;	
O1.b:= O3 ;	



IDENTITE ET EGALITE

Class A (a : entier, b : entier)

A O1 : new A;

A O2;

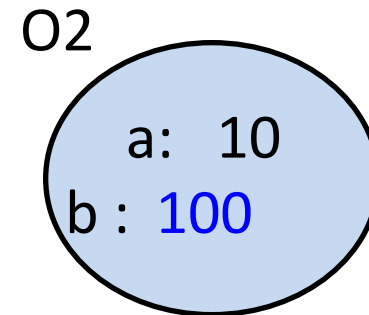
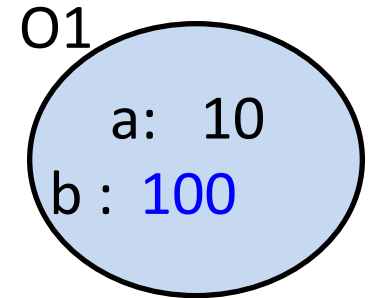
O1.a:=10;

O1.b:=100;

O2:=copy(O1)

O2==O1?

O2=O1?



Un objet est CLONE et NON COPIE

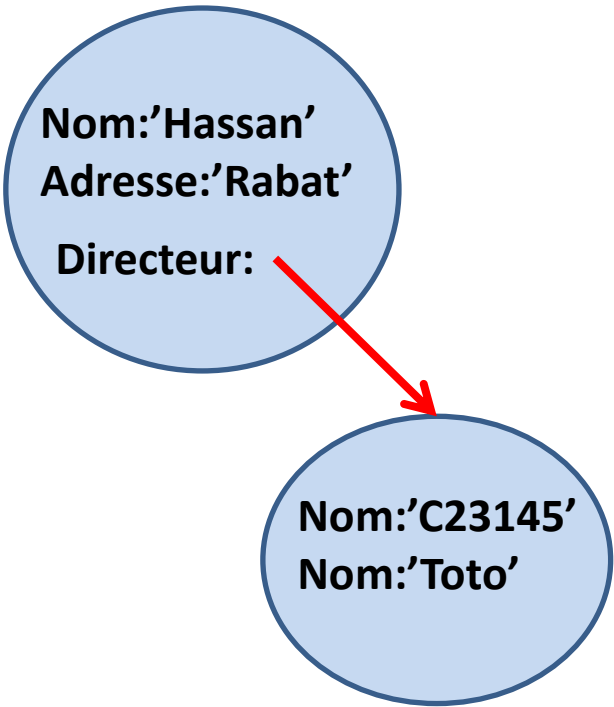
IDENTITE ET PARTAGE D'OBJET

```
class Personne{
private Integer    CIN;
private String     Nom;
.....
}
```

```
class Hotel {
private String     Nom;
private String     Adresse;
Private  Personne  Directeur;
.....
}
```

```
Personne pers=new Personne('C23145','toto');
Hotel Hot=new Hotel('Hassan','Rabat');
```

```
Hot.Directeur=pers;
```



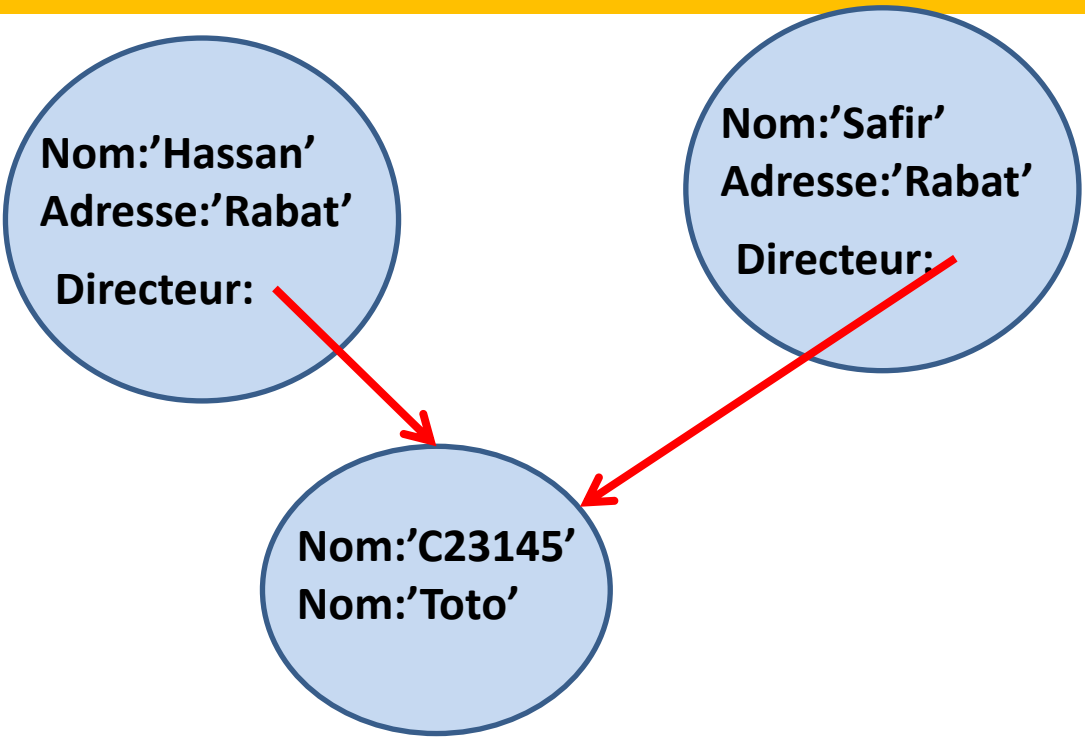
IDENTITE ET PARTAGE D'OBJET

```
class Personne{
private Integer    CIN;
private String     Nom;
.....
}
```

```
class Hotel {
private String     Nom;
private String     Adresse;
Private  Personne  Directeur;
.....
}
```

```
Personne pers=new Personne('C23145','toto');
Hotel HHass=new Hotel('Hassan','Rabat');
Hotel HSafir=new Hotel('Safir','Rabat');

HHass.Directeur=pers;
HSafir.Directeur=pers;
```



```
//ACCES PAR PLUSIEURS ENTREE
HHass.Directeur;
HSafir.Directeur;
```

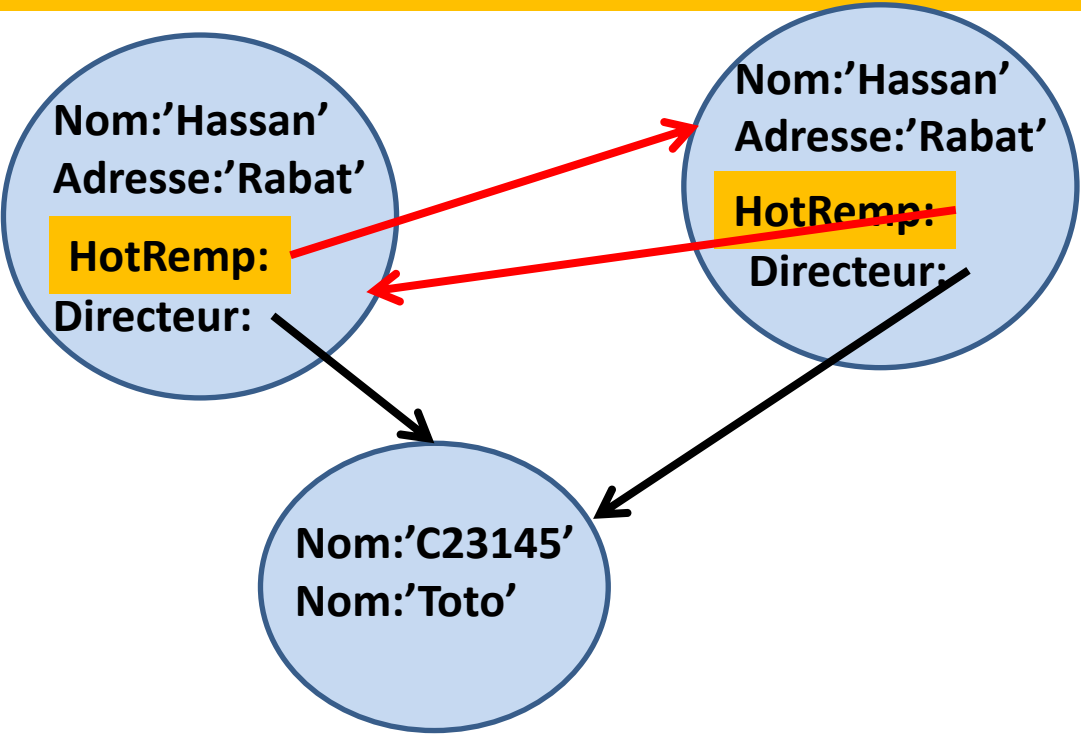
IDENTITE ET PARTAGE D'OBJET

```
class Personne{
private Integer    CIN;
private String     Nom;
.....
}
```

```
class Hotel {
private String     Nom;
private String     Adresse;
private Personne   Directeur;
private Hotel      HotRemp
.....
}
```

```
Personne pers=new Personne('C23145','toto');
Hotel HHass=new Hotel('Hassan','Rabat');
Hotel HSafir=new Hotel('Safir','Rabat');

HHass.Directeur=pers;
HSafir.Directeur=pers;
```



```
//ACCES PAR PLUSIEURS ENTREE
HHass.HotRemp=HSafir;
HSafir. HotRemp=HHass;
```

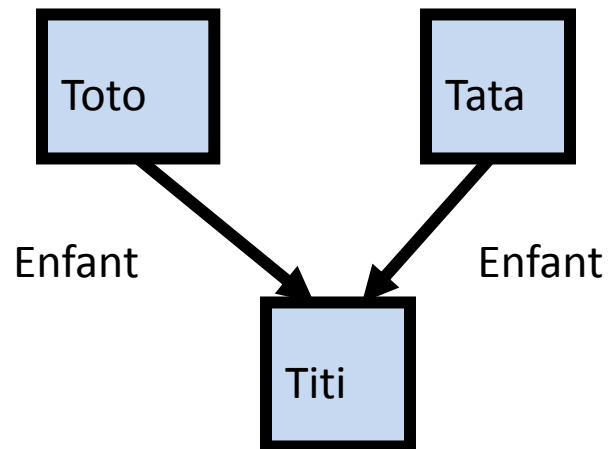
```
HHass.HotRemp.HotRemp ??
```

IDENTITE ET PARTAGE D'OBJET

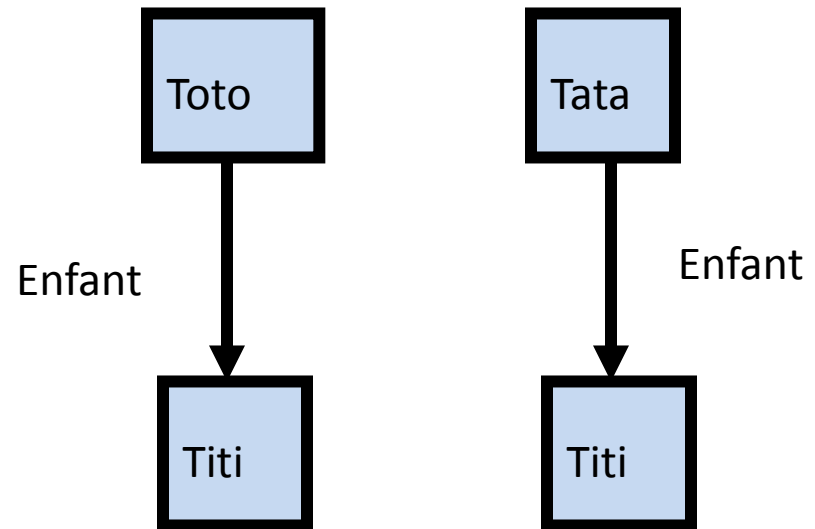
Possibilité de modéliser deux situations:

Toto a une fille nommée Titi

Tata a une fille nommée Titi



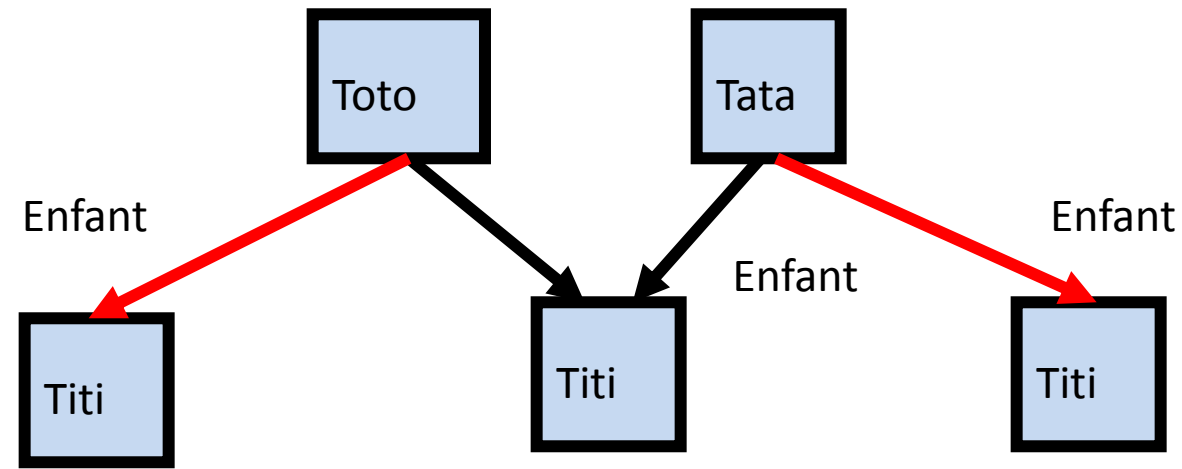
Situation 1: il s'agit de même enfant



Situation 2: il s'agit d'un enfant différent

IDENTITE ET PARTAGE D'OBJET

Possibilité de modéliser deux situations:
Toto a une fille nommée Titi d'un mariage antérieur
Tata a une fille nommée Titi d'un mariage antérieur
Toto et Tata ont une fille en commun qui s'appelle Titi



S'agit-il de même enfant: Toto→Enfant==tata→Enfant; ????????????????

IDENTITE ET PARTAGE D'OBJET: CONCLUSION

Représentation directe du monde réel

le SGBD voit les OBJETS (les OID) et le développeur voit les valeurs

OID et PRIMARY KEY sont deux notions différentes

On peut insérer plusieurs objets avec la même valeur. Pour le SGBD, il s'agit d'objets différents

Un objet n'est jamais copié ou dupliqué, il est référencé par son OID dans les autres objets

IDENTITE ET PARTAGE D'OBJET: CONCLUSION

Les Objets sont partagés à travers leurs OID

L'association entre les objets est implémentée à travers les OID

Il n'y a pas de contrainte d'intégrité référentielle et donc pas d'opérateur de jointure

Un nouveau mode de requête est possible: la navigation à travers le graphe des références.

DEFINITION D'UN SGBDOO

Aspect Orienté Objet

- Identité d 'objet
- **Objet complexe**
- Encapsulation
- **Type ou Classe**
- Héritage
- Surcharge et résolution tardive
- **Complétude**
- **Extensibilité**

- Les objets complexes sont construits à partir d 'objets atomiques et des constructeurs
- Les objets atomiques sont les entiers, les réels, les booléens, les chaînes de caractères.
- Les constructeurs sont le n-uplet (**tuple**), l 'ensemble (**set**), la liste (**list**) et le tableau (**array, vecteur**).
- L 'utilisation des constructeurs est indépendante du type des objets
- L 'utilisation des constructeurs peut être récursive

DEFINITION D'UN SGBDOO

Aspect Orienté Objet

- Identité d 'objet
- Objet complexe
- Encapsulation
- **Type ou Classe**
- Héritage
- Surcharge et résolution tardive
- **Complétude**
- **Extensibilité**

DEFINITION D'UN SGBDOO : NORME OMG

Les règles facultatives

- Héritage multiple
- Généricité
- Vérification de type
- Distribution
- Transaction de conception
- Versions

DEFINITION D'UN SGBDOO : NORME OMG

Les règles ouvertes

- **Modèle de calcul**
- **Modèle de persistance**
- **Uniformité**
- **Nommage**

DEFINITION D'UN SGBDOO : NORME OMG

Les règles ouvertes

- **Modèle de calcul**
- **Modèle de persistance**
- **Uniformité**
- **Nommage**

Rappel

- Dans les LPO: les objets sont Temporaires tant qu'ils ne sont pas écrits explicitement dans des fichiers
- Dans un SGBD
Les données persistent automatiquement

Qualités de la persistance

- Pour le même type on peut avoir des Instances persistantes ou pas
- Les opérations s'appliquent aux objets persistants ou temporaires
- Le statut persistant/temporaire peut être changé
- Un objet persistant ne peut référencer un objet temporaire

Techniques de persistance

- Nouvelles approches (SGBDOO)
 - Persistance à la demande
 - Persistance indépendante du type
- Différents modèle de persistance
 - Au niveau de l'objet ou de la Classe
 - Statique ou dynamique

Persistance de classe

- Une classe déclarée persistante alors toutes ses instances sont persistantes
- Les objets temporaires doivent être explicitement supprimés
- L'intégrité référentielle n'est pas Garantie

Exemple

Create persistent class Personne

Type tuple(nom : string,
 voiture : Véhicule)

Persistence de l'objet

- La sauvegarde est explicite par commande ou méthode
- L'intégrité référentielle n'est pas garantie

Exemple (O2)

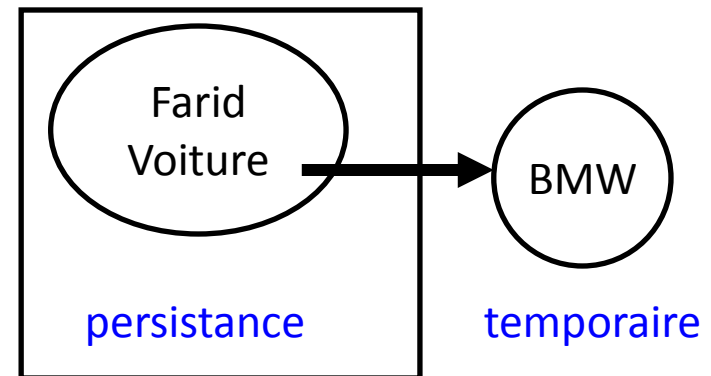
```
class Personne
Type tuple(          nom      : string,
                   voiture   : Véhicule)
```

```
Personne perso = new Personne;
Véhicule BMW = new Véhicule;
```

```
Perso->nom='Farid ;
```

```
Perso->voiture=BMW;
```

```
Put_object(perso);, //persistant
```



Persistence de l'objet: OMG

- L'objet est crée comme persistant ou temporaire
- Pas d'intégrité référentielle

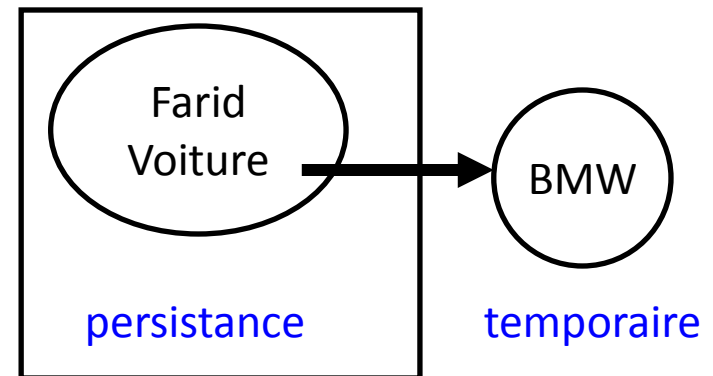
Exemple (OMG)

```
class Personne
Type tuple(          nom      : string,
                   voiture    : Véhicule)
```

```
Personne perso = new persistent Personne;
Véhicule BMW = new Véhicule;
```

```
Perso->nom='Farid ';
```

```
Perso->voiture=BMW;
```



Persistence par racine de persistance

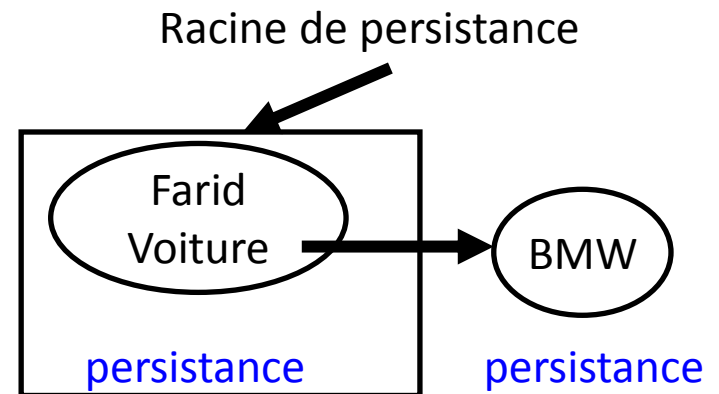
- Tout objet connecté directement ou par transitivité à une racine de persistance est persistant
- La racine de persistance est créée par nommage

Exemple (O2)

```
class Personne
Type tuple(          nom      : string,
                   voiture    : Véhicule)
Name les_persos=set(personne),
```

```
Personne perso = new Personne;
Véhicule BMW = new Véhicule;
Perso->nom='Farid';
Perso->voiture=BMW;
```

```
If ( newperso (perso->nom))
    les_persos+=set(perso)
Else ('erreur: existe déjà');
```



CONCLUSION

SGBDOO:

Capture aisée de la sémantique

Le développeur est très sollicité

SGBDR

Capture 'faiblement' la sémantique

Facilité de mise en œuvre

Le développeur n'est pas très sollicité

CONCLUSION

Vous connaissez les bases de données?

Vous connaissez la programmation objet?

Alors

Vous serez faire les bases de données objet!!!!

BD OO=programmation objet + persistance

LE MODELE RELATIONNEL ETENDU (LE RELATIONNEL OBJET/SQL3)

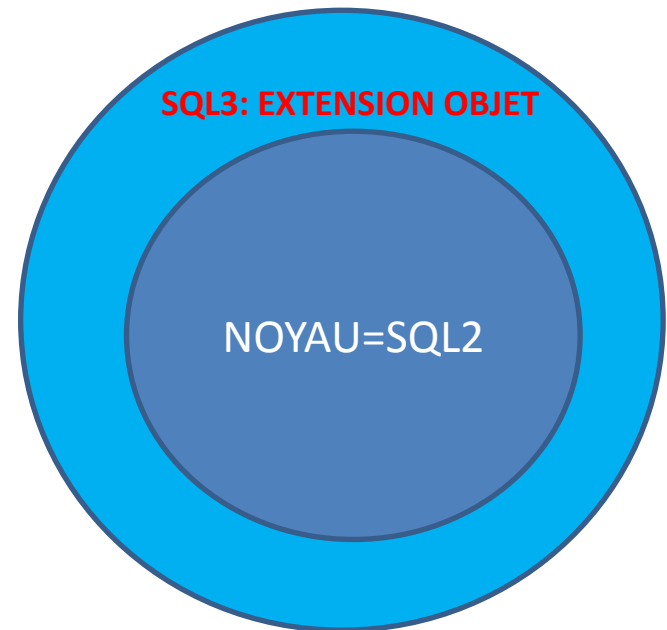
SQL3 = extension du SQL aux concepts objets

Le noyau est toujours relationnel

La table reste le support de prédilection pour la manipulation des données

Les SGBDs basés sur SQL3 sont dits:
OBJET-RELATIONNELS
ou
RELATIONNELS ETENDUS

Exemple: Oracle, Informix, Sybase, DB2...



Les objets d'un type ne deviennent persistants que lorsqu'ils sont insérés dans une table créée avec CREATE TABLE

Exemple d'objet complexe possible dans SQL3

Police	Nom	Adresse	Conducteurs		Accidents		
24	Toto	Casa	Conducteur	Age	Accident	Rapport	Photo
			Tata	45	134		
			Titl	17			
					219		
					037		

Exemple d'objet complexe d'Oracle

Class Assurance

Type (

Police

Integer,

Nom

String,

Adresse String,

Conducteurs

List(C_Conducteurs),

Accidents

List(C_Accidents))

.....

End;

Class C_conducteurs

Type (

Conducteur

String,

Age

Integer,

)

.....

End;

Exemple: implémentation de la table en BDOO pure

NON PREMIERE FORME NORMALE

Forme normale tolérant des domaines multivalués

MODELE RELATIONNEL IMBRIQUE

Un domaine peut lui même être valué par des tables de même schéma

L'APPORT DU MODELE OBJET

Objets complexes et type utilisateur

identité d'objet (REF)

Encapsulation des données

Héritage d'opérations et de structures

Collections (varray, nested table)

Analogie BDOO/R-OO

SGBDOO	Relationnel étendu (ROO)	Commentaire
OBJET	ADT	-pas de véritable objet -pas d'encapsulation
IDENTITE d'OBJET	(REFERENCE) POINTEUR	-les références sont gérées par le développeur explicitement
COLLECTION	VARRAY NESTED TABLE	-lourd à gérer -des tables référencées par la table mère

LE MODELE RELATIONNEL ETENDU VUE GENERALE DE L'APPORT DE L'OBJET

TYPE ABSTRAIT

SPECIFICATION DE TYPE

DECLARATION DES ATTRIBUTS

DECLARATION DES METHODES

CORPS DU TYPE

IMPLEMENTATION DES METHODES

Les types abstraits: exemple

```
• CREATE TYPE t_adresse AS OBJECT(  
•     Numero          NUMBER(4),  
•     Rue             VARCHAR2(20),  
•     Code_postal     NUMBER(5),  
•     Ville           VARCHAR2(20)  
• );
```

```
CREATE TABLE ADRS OF T_ADRESSE(  
CONSTRAINT PK primary key Numero,  
CONSTRAINT CNN CHECK (rue IS NOT NULL)  
);
```

Numero	Rue	Code_Postal	Ville

```
CREATE TABLE adresses OF t_adresse;
```

LA REFERENCE D'OBJET

Les types abstraits: exemple

- CREATE TYPE t_adresse AS OBJECT(
 - Numero NUMBER(4),
 - Rue VARCHAR2(20),
 - Code_postal NUMBER(5),
 - Ville VARCHAR2(20));

REF	Numero	Rue	Code_Postal
XXXXX			
YYYYYYY			
ZZZZZZ			

CREATE TABLE adresses OF t_adresse;

Les types abstraits: exemple

```
• CREATE OR REPLACE TYPE t_personne AS OBEJCT(  
•     PL          NUMBER(4),  
•     Nom         VARCHAR2(20),  
•     Adresse     t_adresse,  
•     .....  
•     Salaire     NUMBER(10,2)  
• );
```

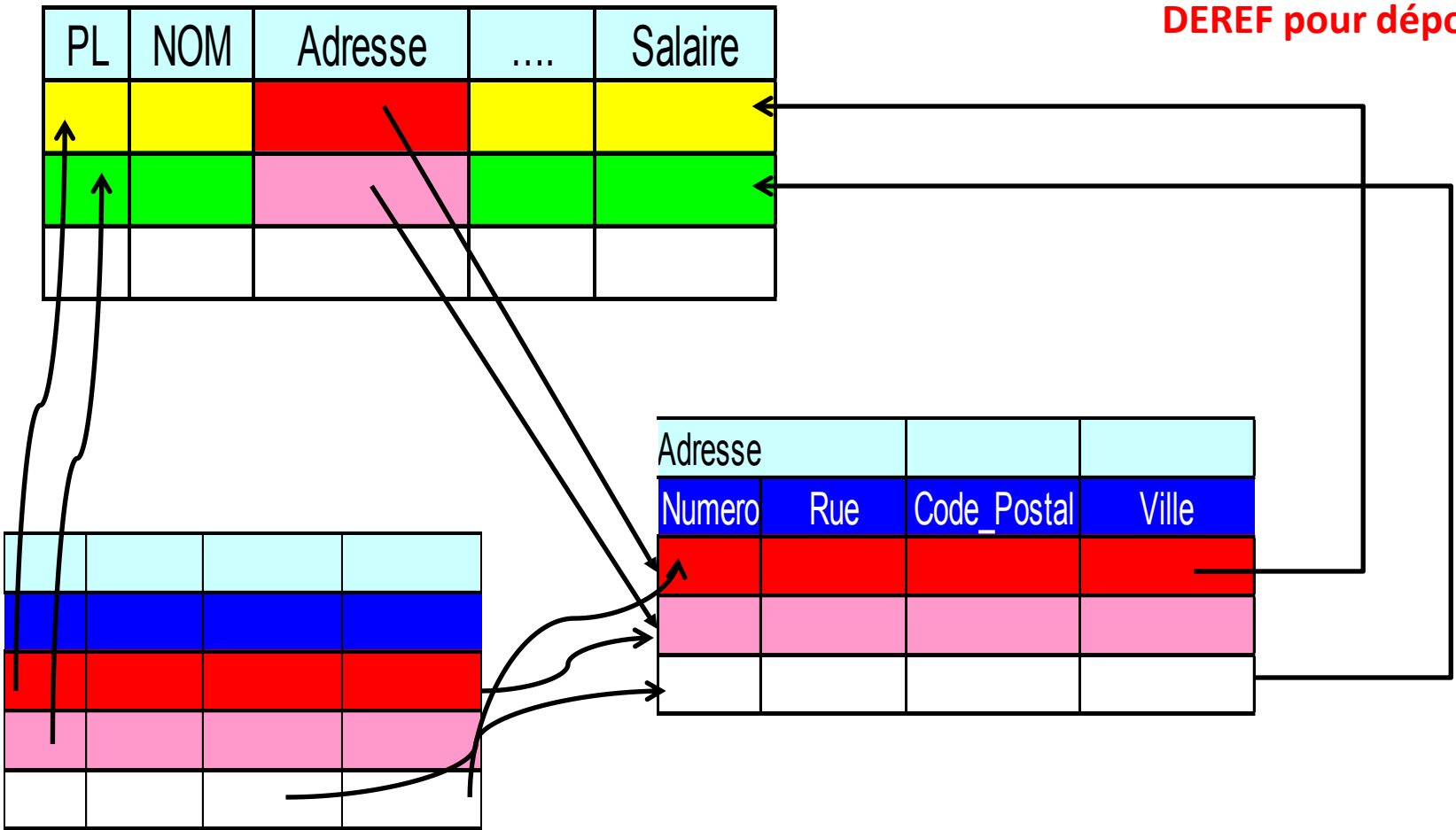
PL	NOM	Adresse				Salaire
		Numero	Rue	Code_Postal	Ville		

```
CREATE TABLE Employes OF t_personne;
```

Les types abstraits: exemple

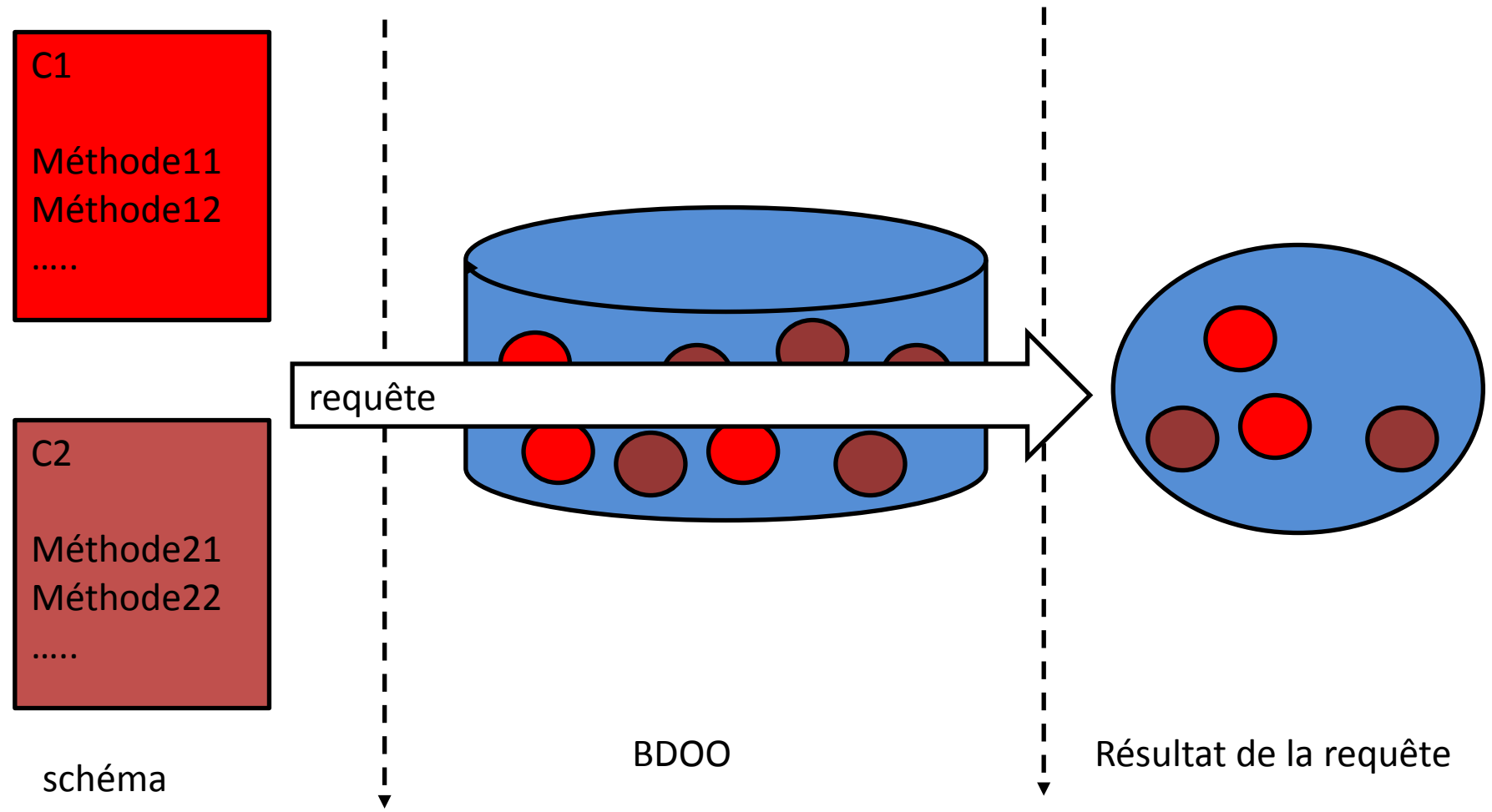
IDENTITE D'OBJET : REF véritable pointeur!!!!

REF pour pointer
DEREF pour dépointer

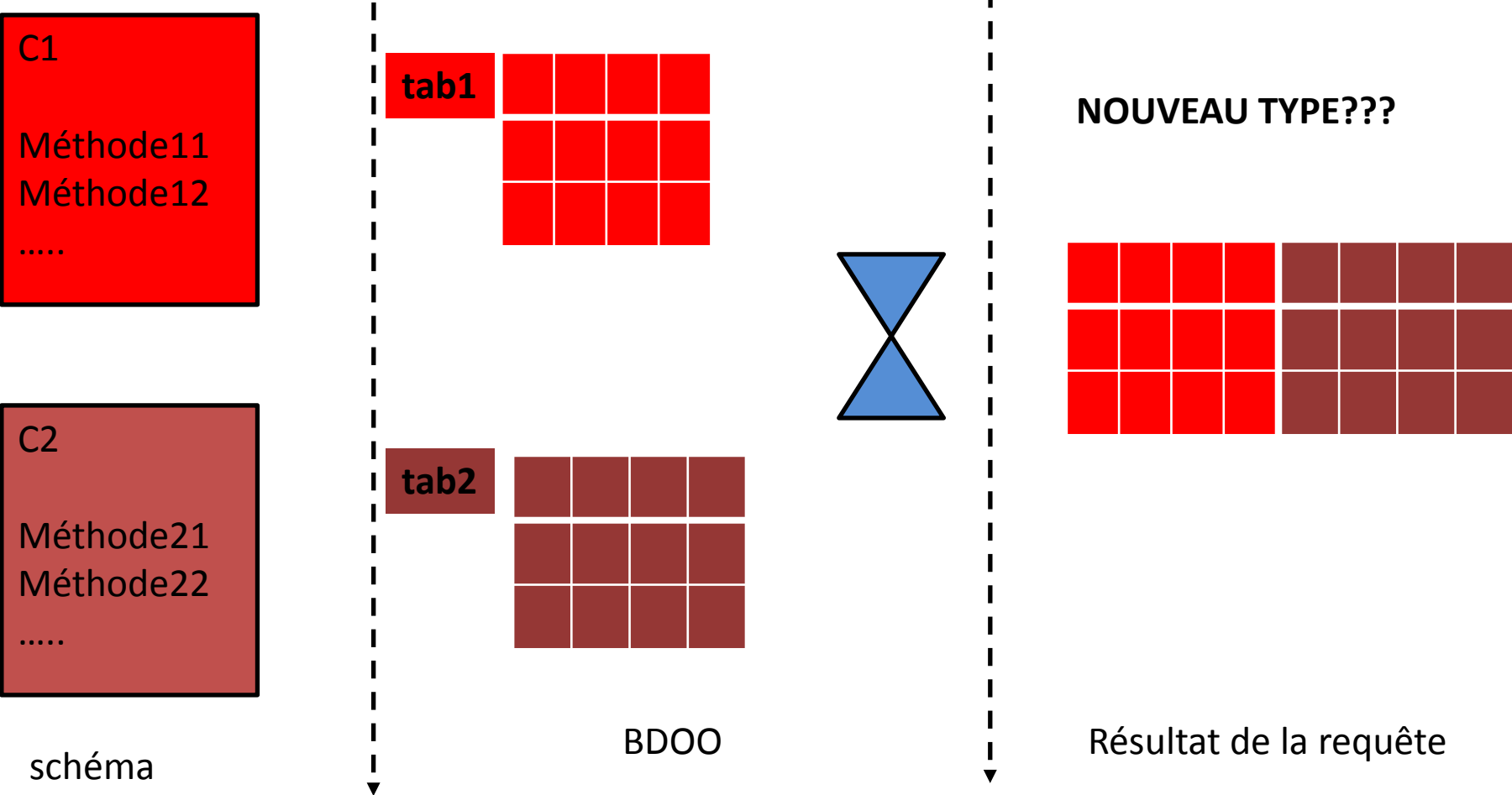


L'ENCAPSULATION

Encapsulation des données : SGBDOO



Encapsulation des données : SGBDR !!!!!



LES COLLECTIONS: TABLEAU

Tab_Employes

nom	prenoms				Adr_privees				Adr_publicues		
	1	2	...	max	1	2	...	max	1	...	max
	string	string	...	string	Tadr	Tadr	...	Tadr	REF Tadr	...	REF Tadr
Toto										...	

Tab_Adresses

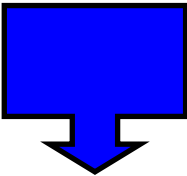
rue	ville
rue1	Ville1
rue2	Ville2

- Ensemble d'éléments ordonnés de même type. Chaque élément occupe une place unique
- **Utilisation: implémenter une relation n-aire avec une cardinalité fixée:**
- Exemple 1: une personne de sexe masculin peut avoir au maximum 4 conjointes
- Exemple 2: un module peut être composé au maximum de
- 5 cours.

LES COLLECTIONS: TABLE IMBRIQUEE

Numdep	nomdep	Projets			
11	NTIC		numprojet	nomprojet	budget
			123	Wifi	1 0001
			18	BDR	1.0002000,89
13	Physique		numprojet	nomprojet	budget

Select p.* from Departement d, **table(d.projets) p;**

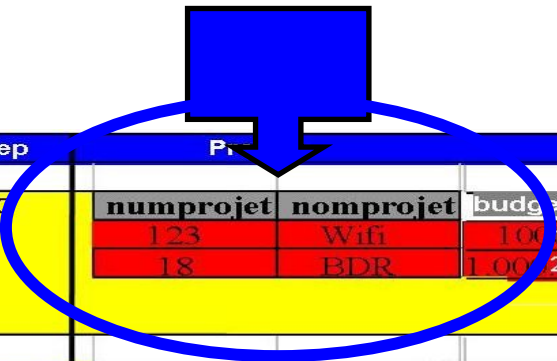


Numdep	nomdep	Projets		
11	NTIC	numprojet	nomprojet	budget
		123	Wifi	10000
		18	BDR	10000000.85
13	Physique	numprojet	nomprojet	budget

Projection sur la colonne projets en **fusionnant les tables imbriquées de tous les tuples**

```
THE (Select projets
      from Departements
      where numdep=11
    );
```

Doit retourner une SEULE table imbriquée au plus



Numdep	nomdep	Projets		
11	NTIC	numprojet	nomprojet	budget
		123	Wifi	10000
		18	BDR	1000000.85
13	Physique	numprojet	nomprojet	budget

Permet de récupérer la table imbriquée d'un tuple et de manipuler la table ainsi récupérée comme une table classique pour: Insert, update, select etc.

L'HERITAGE