De  à SQL>

une synthèse

Christian Soutou

http://icare.iut-blagnac.fr/soutou

"You cannot design databases without a familarity with the techniques of the ER diagramming"

R.J. Muller,
Database Design for Smarties, Using UML for Data Modeling,
Morgan & Kaufman, 1999

Plan

- Historique / Généralités
- Face à face Merise/UML
- Profil UML pour les bases de données
- Exercice
- Bibliographie, Webographie

Historique

UML 2.0 2003

UML 1.3 1999

UML 1.2 1998

UML 1.1 1997

Rational, HP,
Microsoft,
Oracle,
Unisys, etc

UML 1.0

IBM et
ObjecTime

UML 0.9x

Unified Method 0.8

OOSE

Autres méthodes

Booch

OMT

Généralités

- UML (*Unified Modeling Language*)

- Objectifs
  - Représenter des systèmes par une notation unifiée basée sur les concepts objets
  - Langage de modélisation (analyse et conception)
  - Établir un couplage entre concepts et code exécutable (C++, Java…)

Les diagrammes

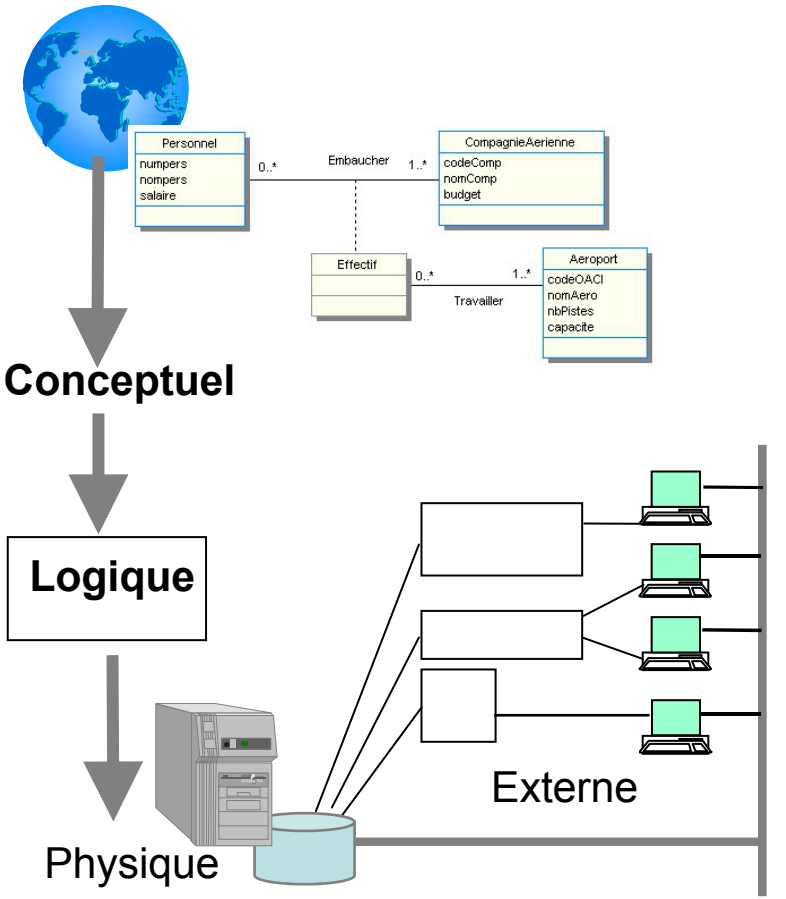
- Statiques (structurel)
  - **Diagrammes de classes**
  - **Diagrammes d'objets**
  - Diagrammes de cas d'utilisation (*use cases*)
  - Diagrammes de composants
  - Diagrammes de déploiement
- Dynamiques (comportemental)
  - Diagrammes de séquence
  - Diagrammes de collaboration
  - Diagrammes d'états-transitions
  - Diagrammes d'activités

Diagrammes statiques

- **Classes** : Description des classes et des relations (associations entre les classes)
- **Objets** : Description des objets et leurs liens, correspondent à des diagrammes de collaboration simplifiés, sans représentation des envois de messages
- Cas d'utilisation (*use cases*) : Description des fonctions du système du point de vue des utilisateurs
- Composants : Description des composants physiques du système
- Déploiement : Description des composants sur les dispositifs matériels

Niveaux de conception

- **Conceptuel : oui**
- **Logique : oui**
- Physique : non
- Externe : très peu



Pourquoi utiliser UML?

- Notoriété des entreprises qui appartiennent au consortium *UML* (DEC, HP, IBM, Microsoft, Oracle, Rational Software, Unisys...)
- La majorité des nouveaux projets utilisent cette UML via un outil
- Avantages indéniables des concepts objets (réutilisabilité, maintenance, prototypage)

Face à face Merise/UML

- Concepts de base
- Associations
- Agrégations
- Contraintes
- Héritage

Historique de Merise

- P. Moulin, J. Randon, S. Savoysky, S. Spaccapietra, H. Tardieu, M. Teboul, “Conceptual model as database design tool”, *Proceedings of the IFIP Working conference on Modelling in Database Management Systems ,* G.M. Nijssen Ed., North-Holland, 1976

- P.P. Chen, “The Entity-Relationship Model : Towards a Unified View of Data”, *ACM Transactions on Database Systems,* Vol 1, N°1, 1976

Face à face Merise/UML

- Les « + » d'UML
  - Les méthodes (dynamique des classes)
  - Les stéréotypes : mécanismes d'extensibilité par ajout de nouveaux types de classes

    `<<Table>> <<View>> <<Association>>...`
  - Lien avec les applications Java

- Le « - » d'UML : absence de méthode (ce n'est qu'une notation)

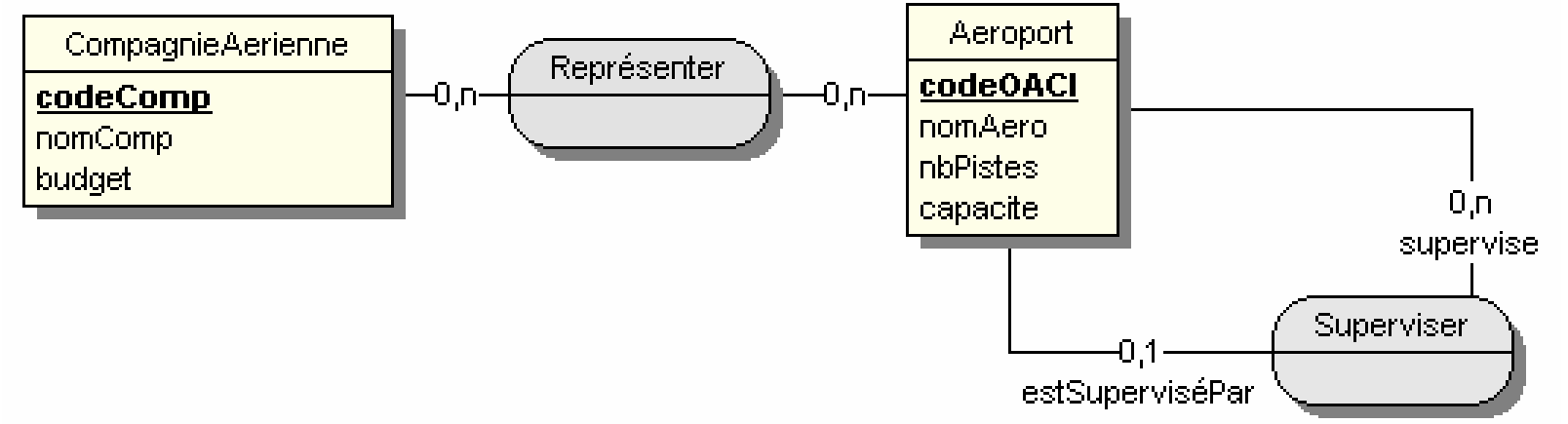
Face à face Merise/UML

- Équivalences des concepts de base
  - Entité ⇔ Classe
  - Occurrence ⇔ Objet
  - Association ⇔ Association / Classe-association
  - Contraintes ⇔ Contraintes / Notes
  - MCD ⇔ Diagramme des classes

Cardinalité est synonyme de multiplicité mais l'interprétation est différente

Associations Merise

- un identifiant (simple ou composé) par entité
- pas d'identifiant d'association
- pas de propriété pour les associations binaires *un-à-un* et *un-à-plusieurs*
- Rôles possibles

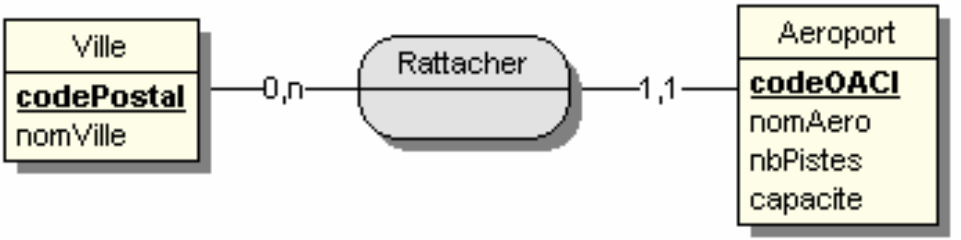


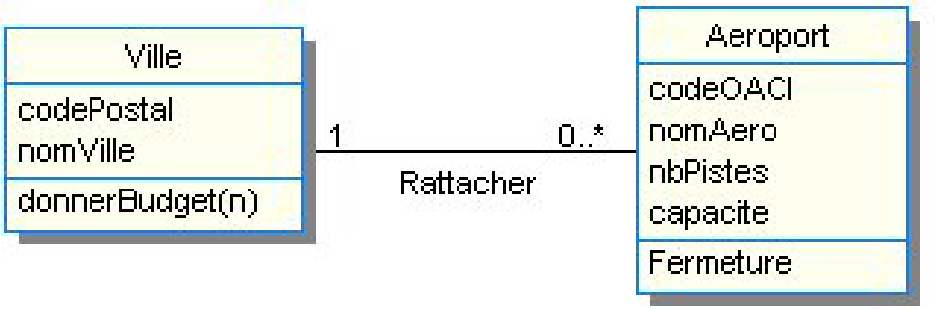
Cardinalité n'est pas multiplicité

Interprétation des cardinalités

- du modèle de Chen (*entity relationship*) / UML
- du modèle MCD de Merise

➢ Associations binaires : simple permutation

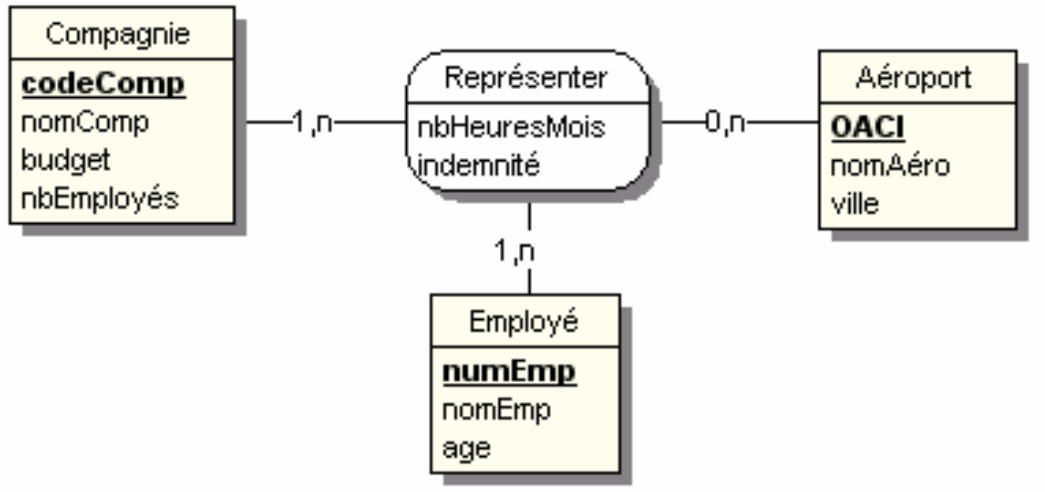




➢ Associations *n*-aires : raisonnement différent

Associations *n*-aires

- Dans Merise : lues d'une entité du sens entité concernée → entités connectées

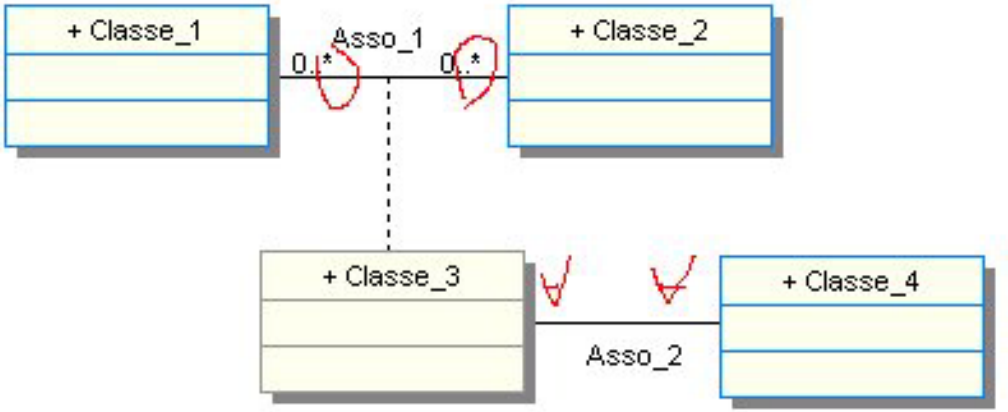


- Dans le modèle de P. Chen et pour **UML** : lues du sens entités connectées → entité concernée
- Rôles et attributs possibles dans tous les formalismes

Associations *n*-aires UML

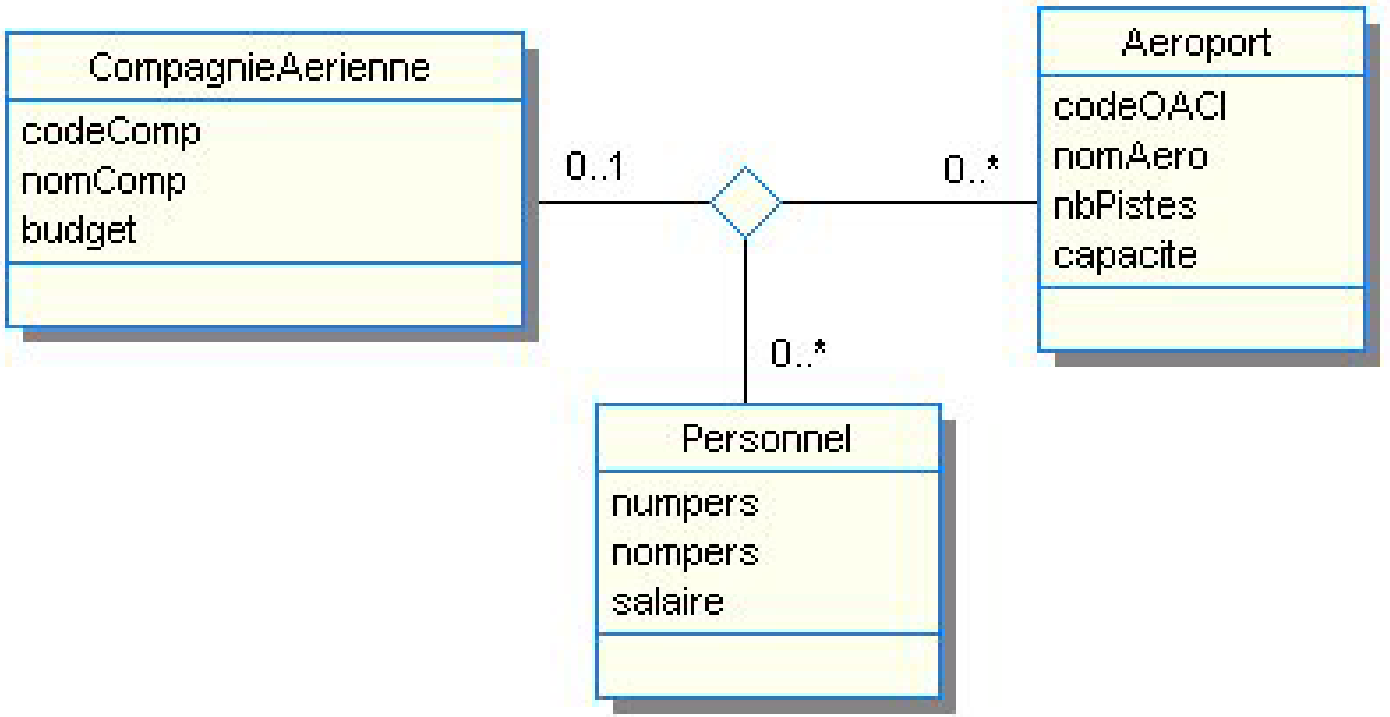
- Trois possibilités de représentation
  - Symbole losange
  - Classe stéréotypée

  - Classe-association (modélisation de contraintes d'unicité et d'inclusion)



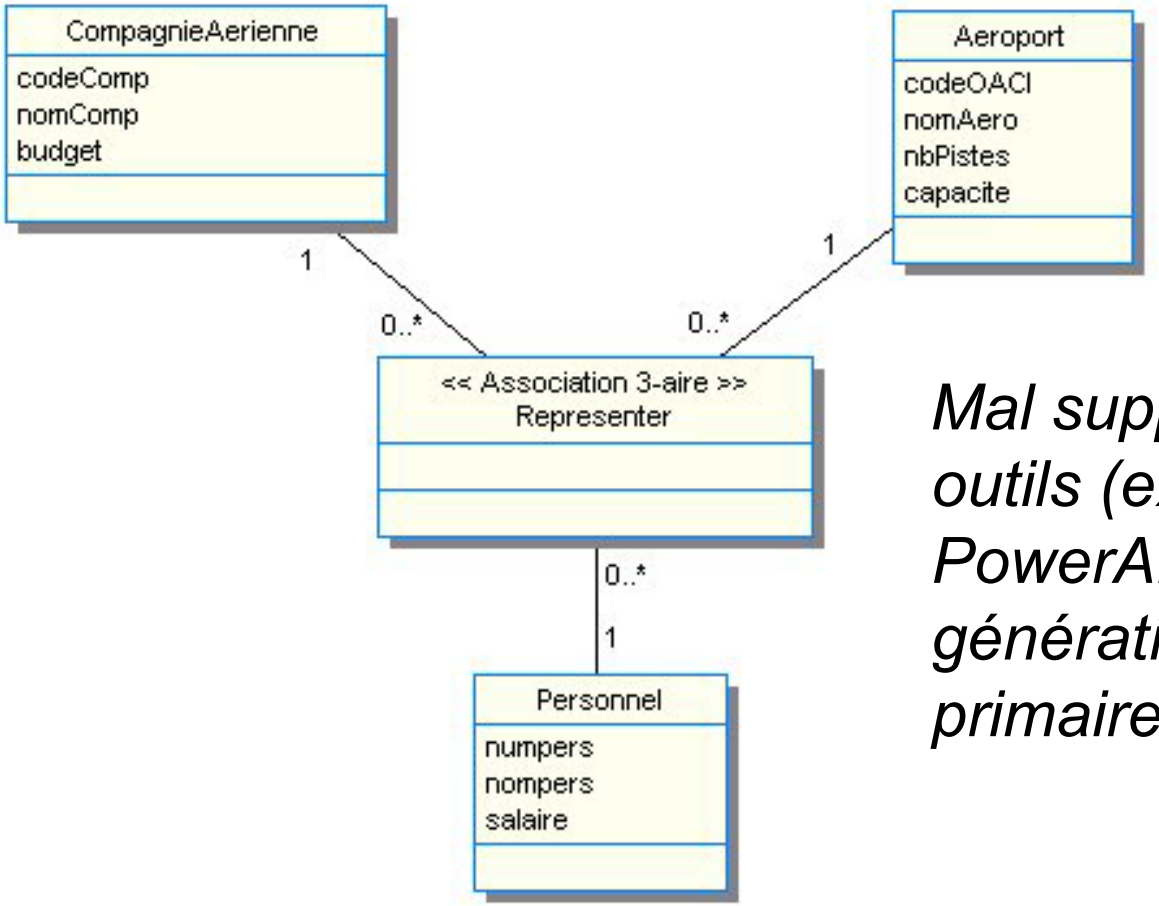
Symbole losange

- De plus en plus d'outils le prenne en compte



Classe stéréotypée

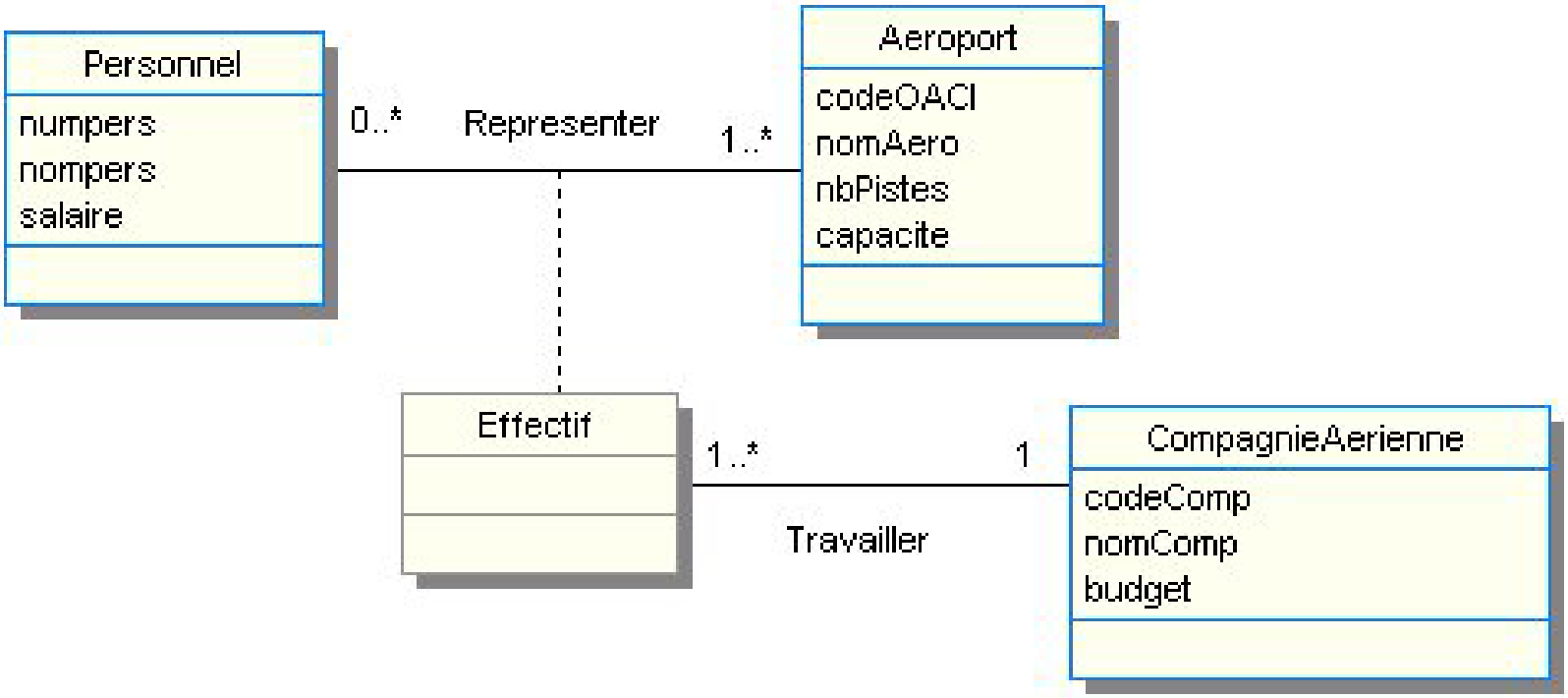
- Les liens doivent s'instancier simultanément



Mal supporté par les outils (ex. Rose / PowerAMC) dans la génération SQL (clé primaire)

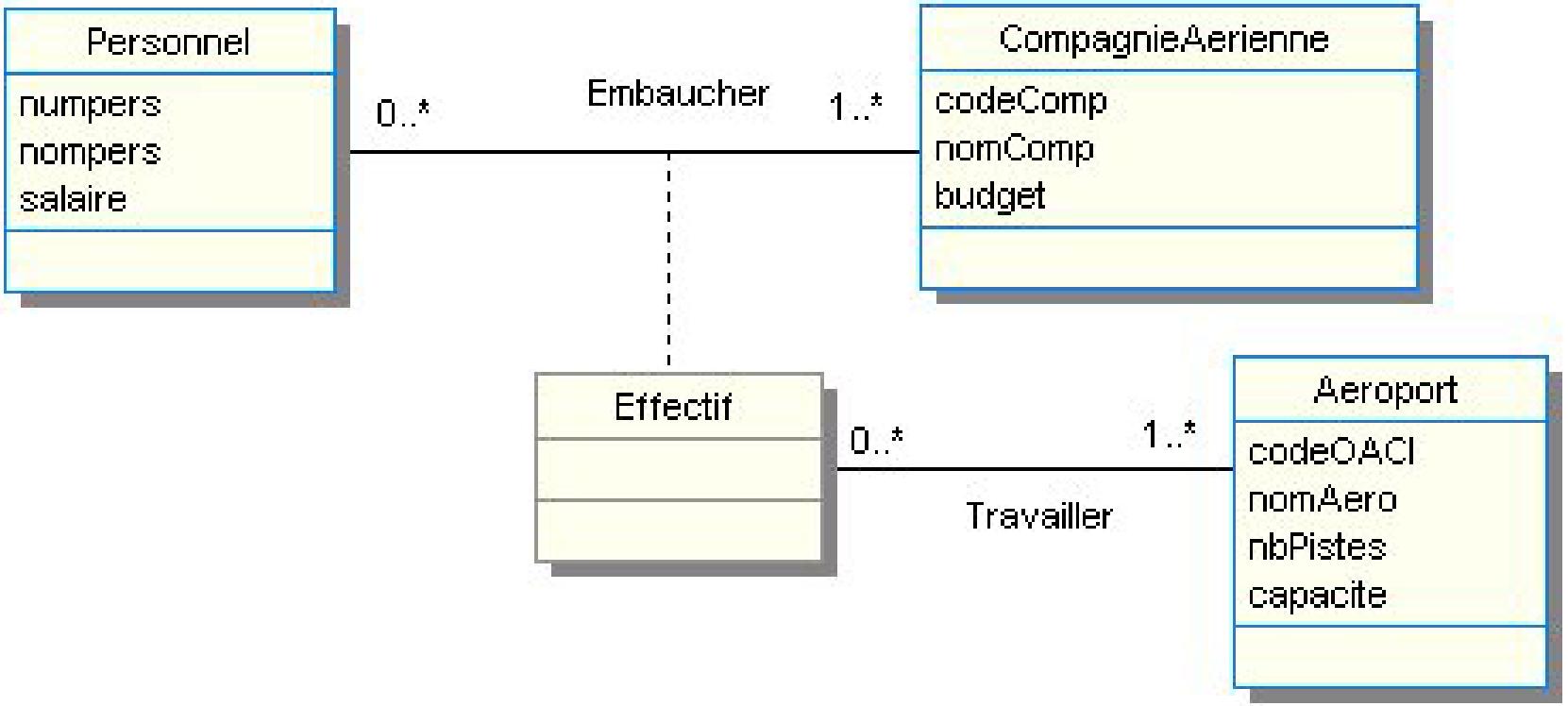
Contrainte d'unicité

- «*Un employé présent dans un aéroport ne travaille que pour le compte* ***d'une*** *seule compagnie aérienne* ».



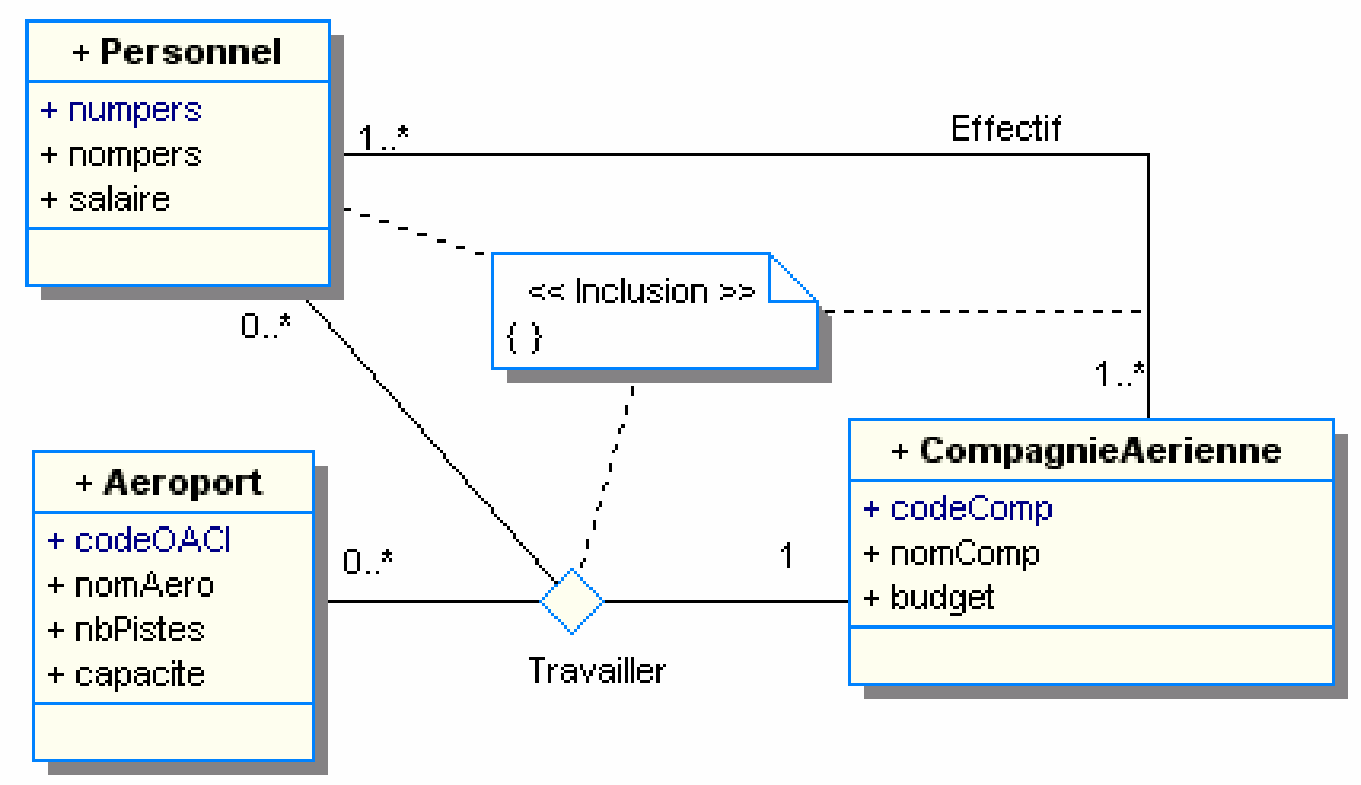
Contrainte d'inclusion

- «*Un employé ne peut représenter que* ***ses*** *employeurs dans tout aéroport où il travaille. »*

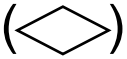


Cumul de contraintes

- *«Un employé ne peut représenter que **ses** employeurs dans tout aéroport où il travaille. »*
- *«Un employé présent dans un aéroport ne travaille que pour le compte **d'une** seule compagnie aérienne ».*

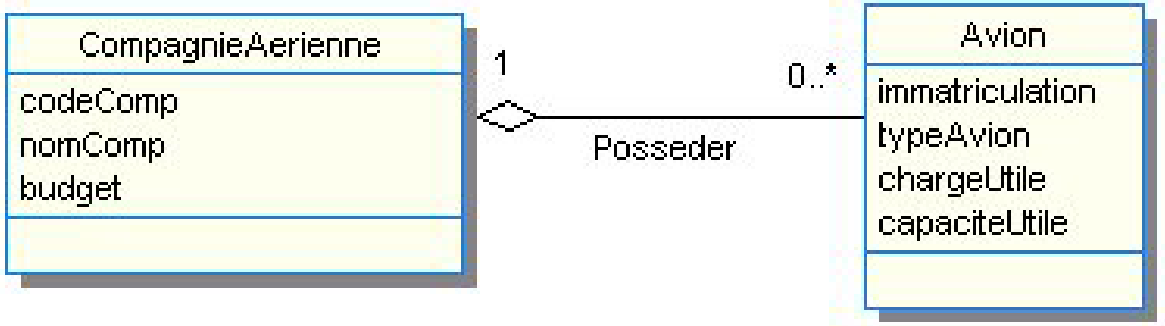


Agrégations

- L'agrégation () affine une association (réflexive, binaire ou *n*-aire) pour laquelle l'une des extrémités joue un rôle prédominant par rapport à l'autre
- L'agrégation concerne un seul rôle d'une association
- Agrégation forte : composition ()
- La notion d'agrégation a été un des aspects les plus discutés de la notation UML
- L'agrégation se traduira au niveau de SQL par déclencheurs (*triggers*) ou contrainte `CASCADE` (sur des clés étrangères)

Agrégations

- Il est préférable d'utiliser une agrégation dans les cas suivants
  - une classe fait partie d'une autre classe
  - une action sur une classe implique une action sur une autre classe
  - les objets d'une classe sont subordonnés aux objets d'une autre classe

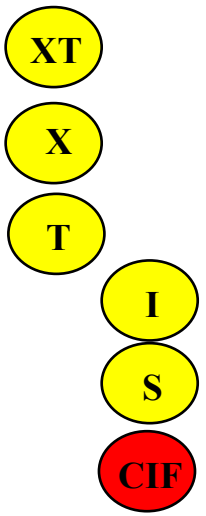


Contraintes

- Objectif : plus de sémantique à un schéma

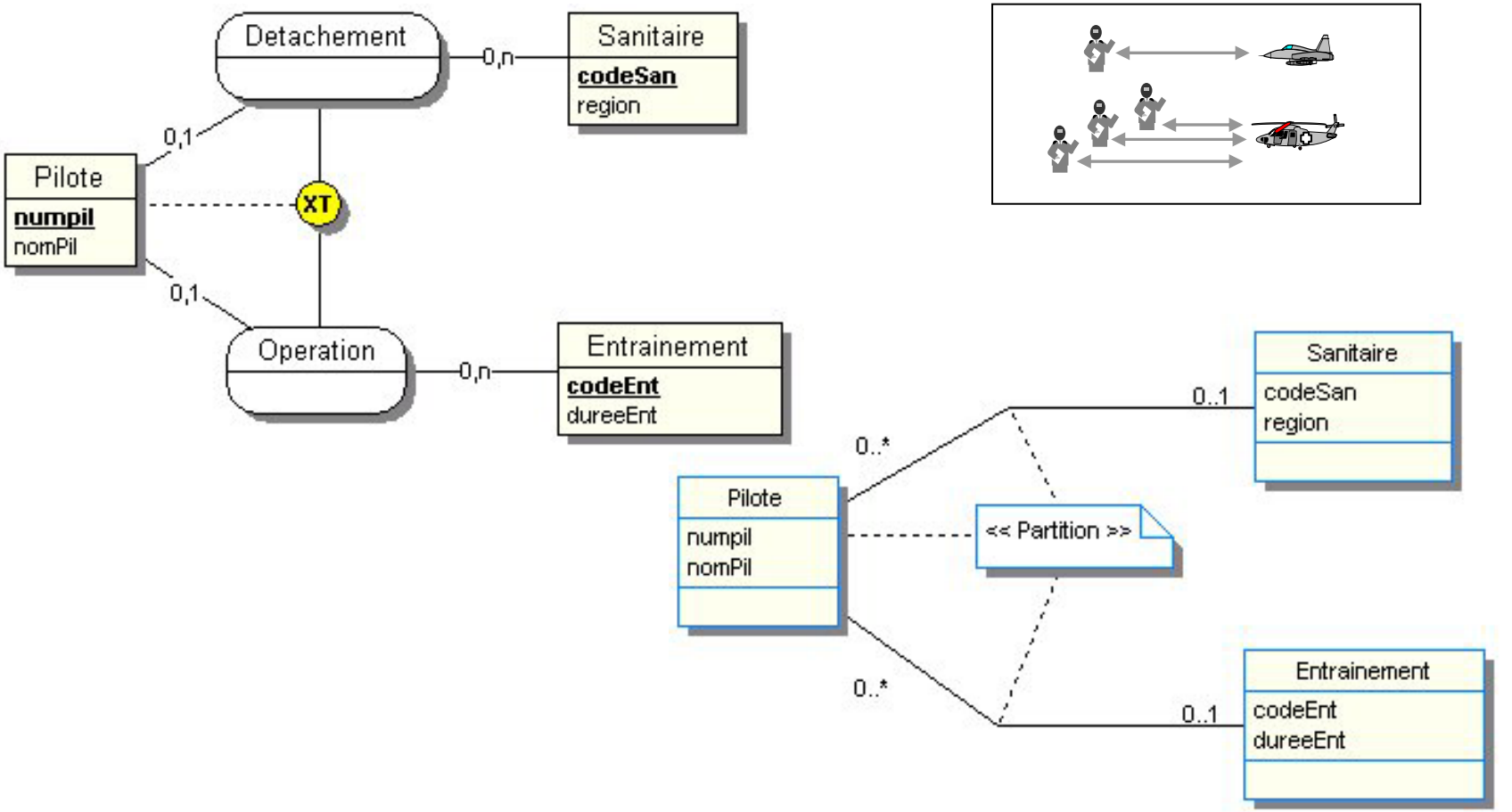
- Contraintes Merise/2
  - Partition
  - Exclusion
  - Totalité

  - Inclusion
  - Simultanéité
  - Unicité (CIF)



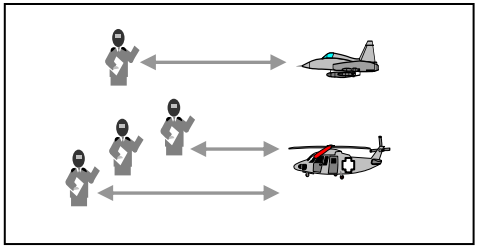
- Prises en compte par les outils mais pas de génération automatique de code associée

Contrainte de partition : CHECK



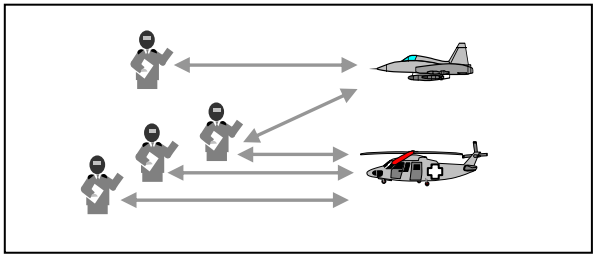
Exclusivité : CHECK

X <<Exclusivité>>



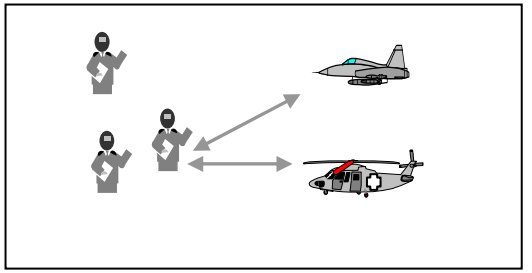
Totalité : CHECK

T <<Totalité>>



Simultanéité : CHECK

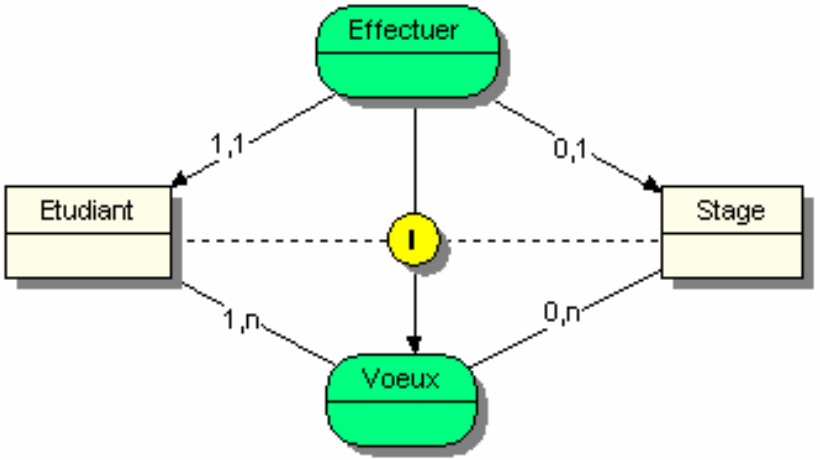
S <<Simultanéité>>

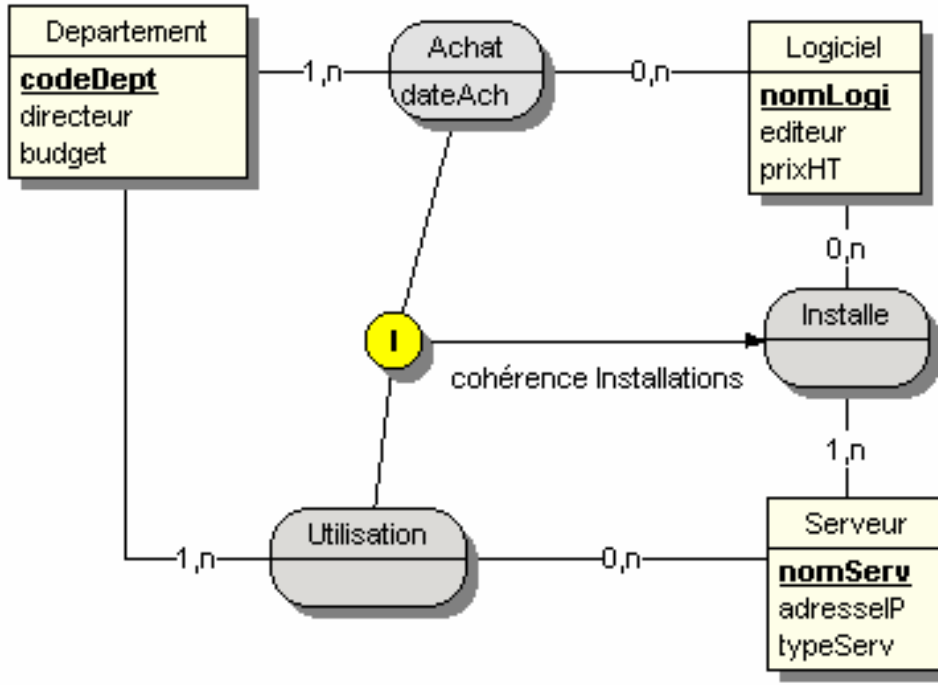


Code SQL pour la simultanéité

```
CREATE TABLE Pilote
(numpil NUMBER PRIMARY KEY, nompil VARCHAR(10),
 sani VARCHAR(10), entraine VARCHAR(10),
 CONSTRAINT fk_pilote_sanitaire
     FOREIGN KEY(sani)REFERENCES Sanitaire(codesan),
 CONSTRAINT fk_pilote_entrainement
    FOREIGN KEY(entraine) REFERENCES Entrainement(codent),
 CONSTRAINT ck_pilote_simultaneite CHECK
    ((sani IS NULL AND entraine IS NULL) OR
     (sani IS NOT NULL AND entraine IS NOT NULL))
);
```

Contrainte d'inclusion : déclencheur



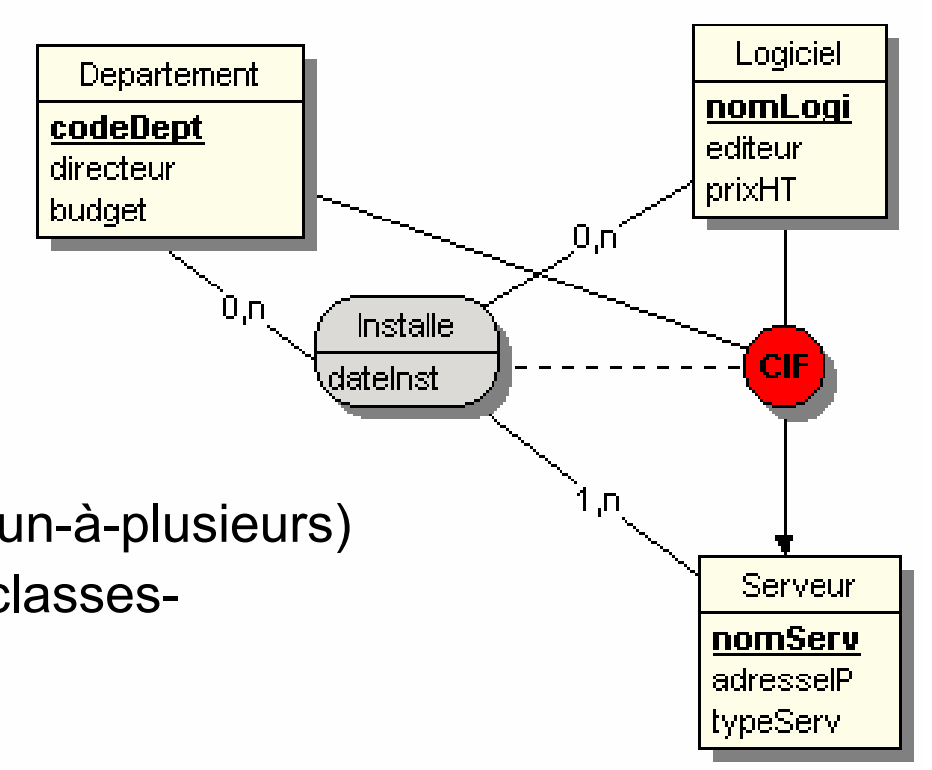


Merise/2 I

<<Sous-ensemble>>

Contrainte d'unicité

- Merise/2 (CIF)
  - Associations binaires (un-à-plusieurs)
  - Associations *n*-aires : inter-entités

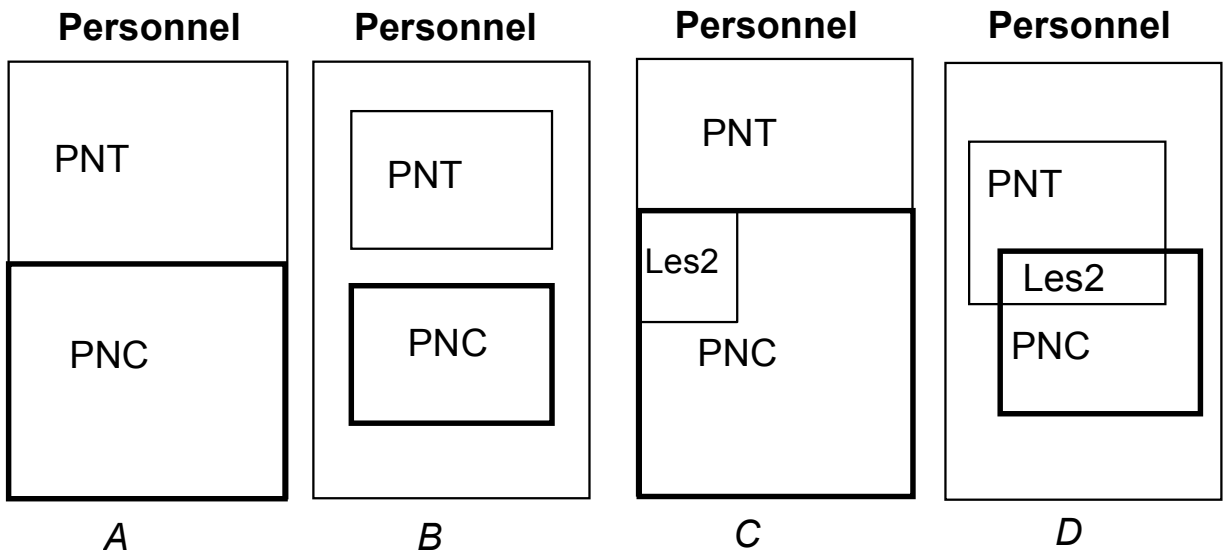


  - Associations binaires (un-à-plusieurs)
  - Associations *n*-aires : classes-associations

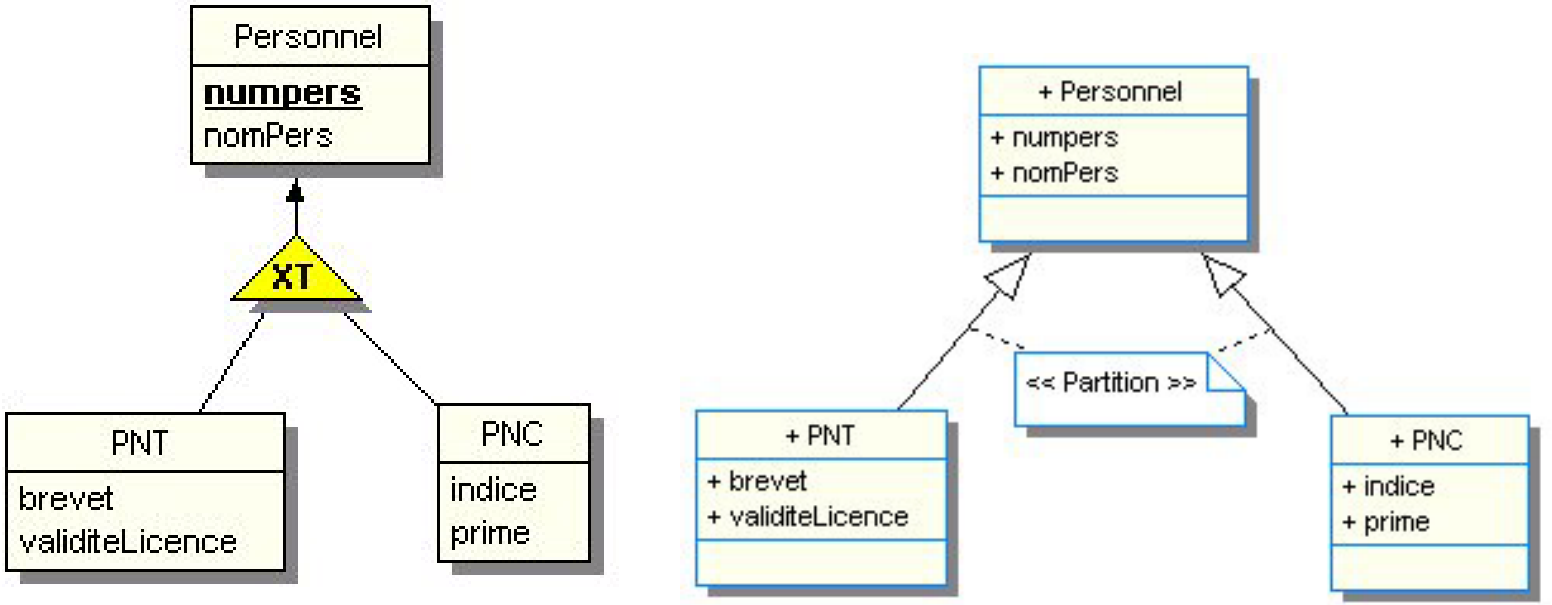
Héritage au niveau conceptuel

- Exemple : les navigants d'Air France
  - PNT : Personnel Navigant Technique
  - PNC : Personnel Navigant Commercial

- Contraintes possibles
  - de partition (*A*)
  - exclusion (*B*)
  - totalité (*C*)
  - aucune (*D*)



Héritage Merise/2 - UML



Héritage au niveau logique

- décomposition par distinction

PNC[numPers#, indice, prime**]**

Personnel[numPers, nomPers**]**

PNT[numPers#, brevet, validiteLicence**]**

- *Exclusion*
- *Partition*

- décomposition descendante (*push-down*)

PNC[numPers, nomPers, indice, prime**]**

PNT[numPers, nomPers, brevet, validiteLicence**]**

- *Totalité!!!*

- décomposition ascendante (*push-up*).

Personnel[numPers, nomPers,indice, prime,
brevet, validiteLicence**]**

- *Totalité (mais valeurs NULL)*

Profil UML pour les BD

- Définition
- Éléments
- Exemple sous Rose

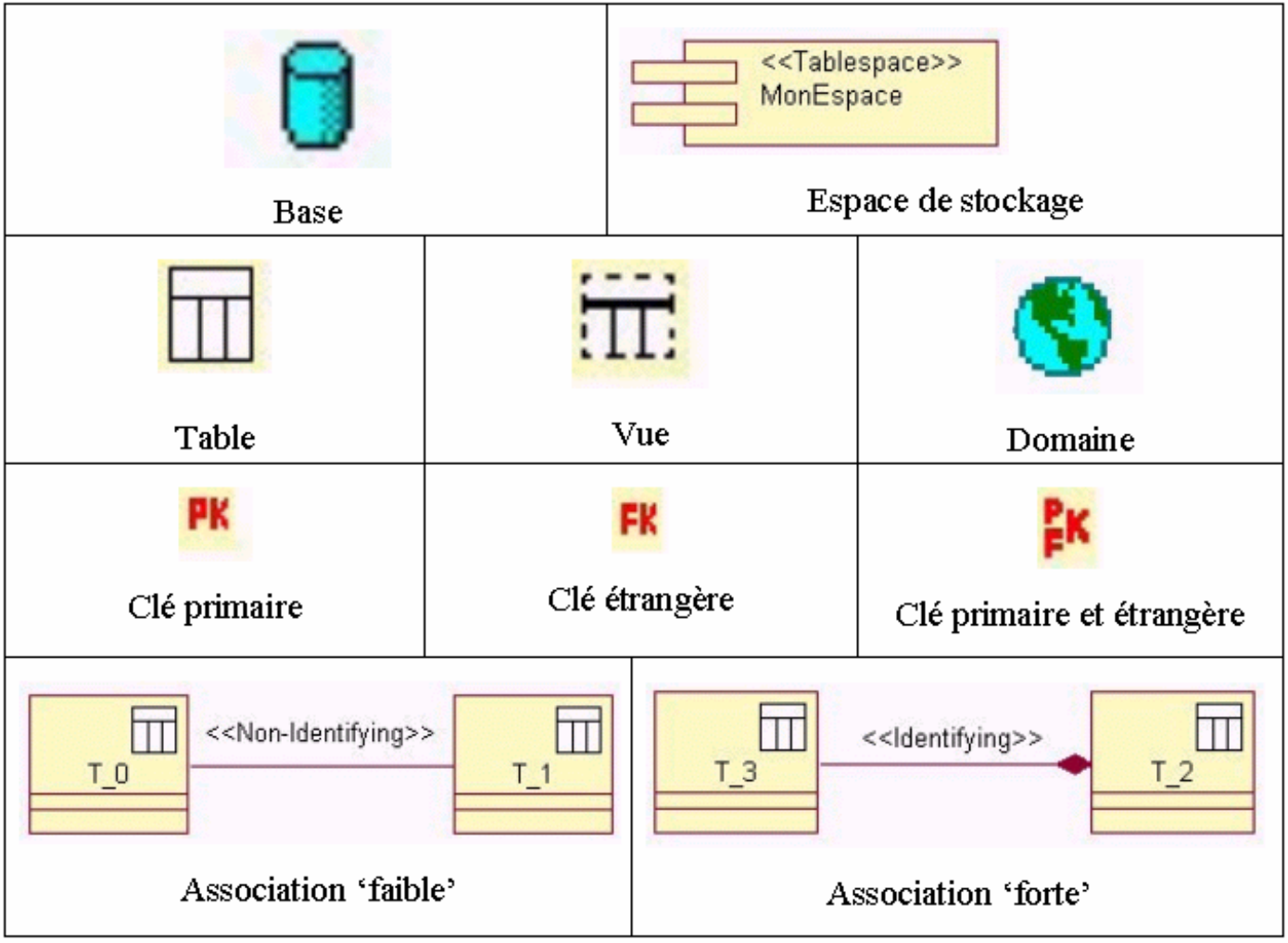
Profils UML

- Un profil est une proposition d'une communauté et regroupe un ensemble d'éléments UML
  - Composants
  - Stéréotypes
  - Icônes
  - …

    qui s'appliquent à un contexte particulier et qui conservent le méta-modèle d'UML intact

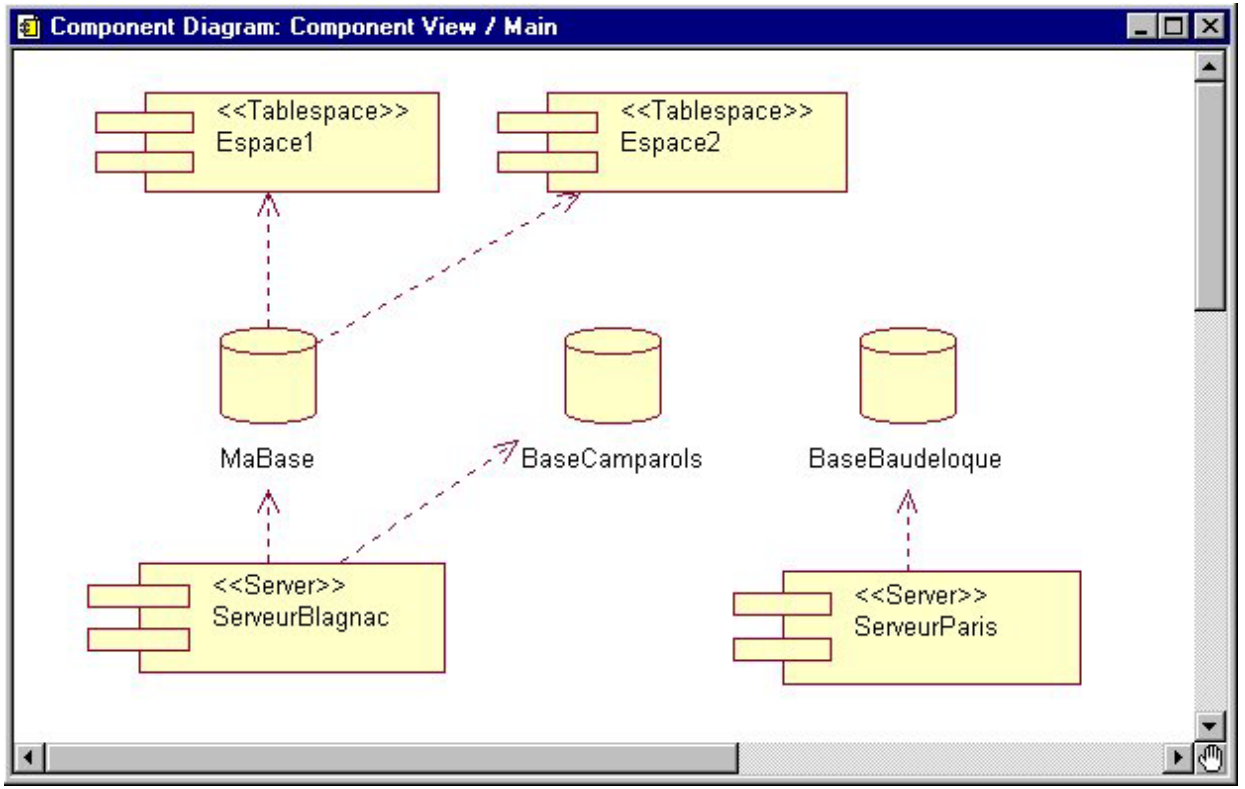
- ***UML profile for Data Modeling*** proposé par Rational Software

Éléments du profil UML/BD

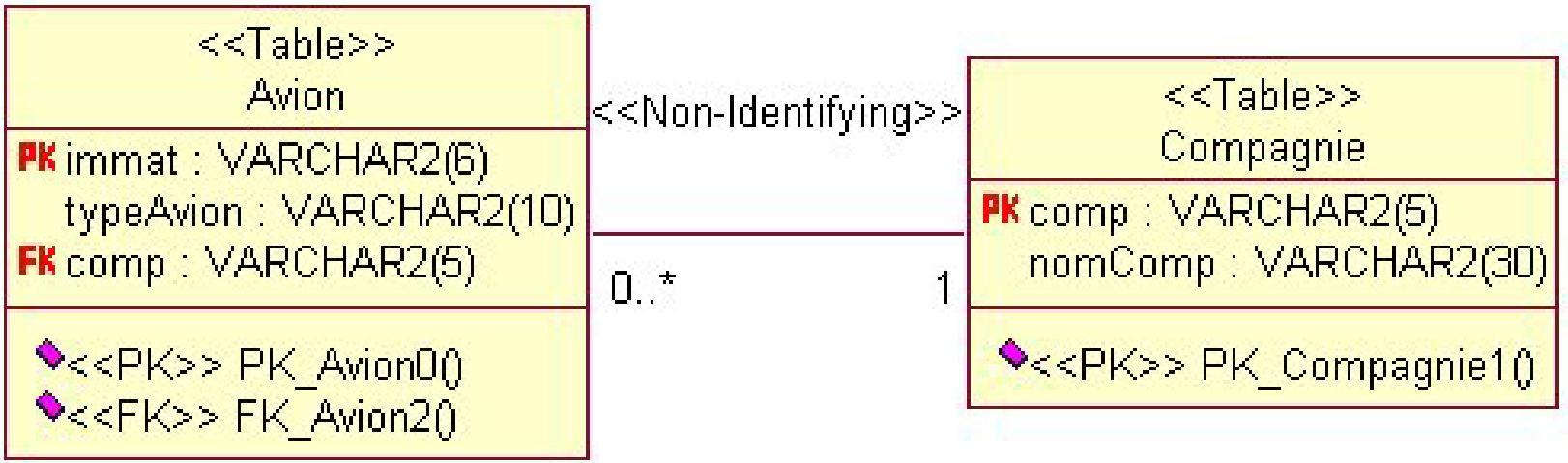


Profil UML/BD

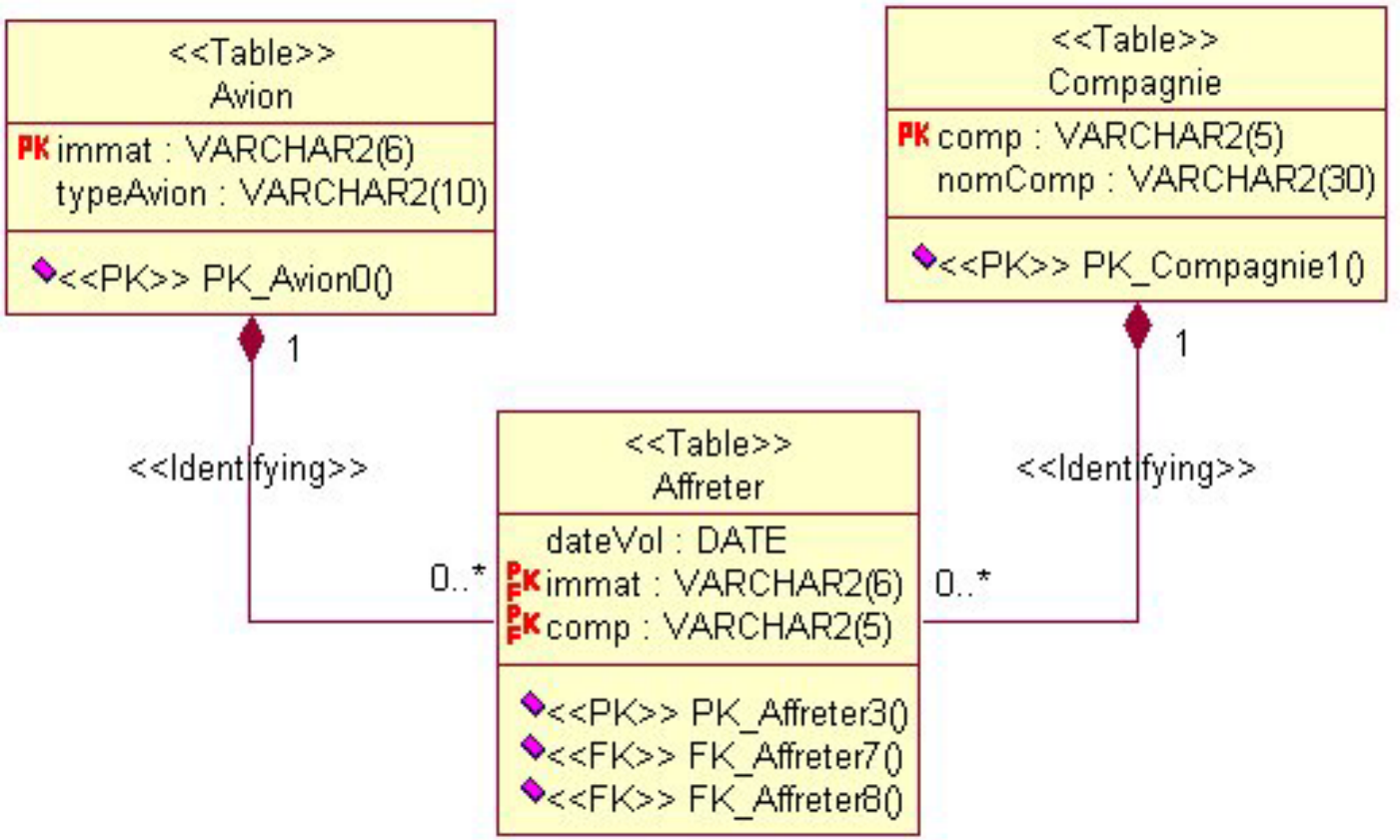
- Éléments d'une architecture



Association *un-à-plusieurs*



Association *plusieurs-à-plusieurs*



Exercice

- Un client d'une banque est caractérisé par (`num`, `nom`, `adresse`). Il faudra pouvoir stocker ses différents numéros de téléphone fixe, travail et portable (`numtel`).

- Un compte est rattaché à un client (`ncompte`, `solde`, `dateouv`). Un client peut disposer de plusieurs comptes. Un compte épargne est rémunéré à un taux (`txInt`).

- Un compte courant est caractérisé par un nombre d'opération de carte bleue (`nbopCB`).

- On veut stocker les opérations faites par un client (signataire ou pas) sur un compte courant (`dateop`, `montant`). Quand le montant est positif il s'agit d'un dépôt, quand il est négatif il s'agit d'un retrait.

Exercice

- Un compte courant peut avoir plusieurs signataires (qui sont des clients de l'agence). Pour un compte courant donné un signataire peut avoir différent droits (`droit`) :
  - 'D' signifie le droit de débiter sur le compte
  - 'R' signifie le droit de retirer un chéquier
  - 'X' signifie le droit de retirer et de clôturer le compte

- Il est important de savoir quel employé de la banque (`numEmp`, `nomEmp`) a affecté chaque droit.

Bibliographie / Webographie

- D. Nanci, B. Espinasse, B. Cohen, *Ingénierie des systèmes d'information : Merise*, Vuibert, 2000
- P.A. Muller, N. Gaertner, *Modélisation objet avec UML*, Eyrolles, 2003
- P. Roques, UML par la pratique, Eyrolles, 2003
- C. Soutou, *De UML à SQL*, Eyrolles, 2002
- Forum *Modélisation, Méthodes, ...*
  `http://www.developpez.net/forums/index.php`