2ème partie

Passage d'un modèle conceptuel de données à un modèle logique

PLAN

1 - Rappels

2 - Passage du modèle objet au modèle logique

3 - Transformation de l'héritage

1. RAPPELS

A partir d'un cahier des charges (le plus souvent un texte), on spécifie un **modèle conceptuel de données** (un diagramme de classes).

Concepts principaux d'un diagramme de classes :

- Objet
- Classe d'Objets
- Attribut
- Association
- Contraintes d'intégrité

1.6 Modèle logique : rappel (1)

UNE RELATION = UN **TABLEAU** A DEUX DIMENSIONS

UNE **COLONNE** = UN ATTRIBUT

EN-TÊTE DU TABLEAU = **SCHEMA** DE LA RELATION (DESCRIPTION DU TYPE)

UNE **LIGNE** = UN TUPLE

ENSEMBLE DES LIGNES = CONTENU DE LA RELATION (EXTENSION)

CLÉ PRIMAIRE = Un ou plusieurs attributs qui permettent de repérer de manière unique chaque valeur de tuple

CONTRAINTE RÉFÉRENTIELLE (clé étrangère) = Un ou plusieurs attributs qui permettent d'exprimer les liens entre les relations.

1.6 Modèle logique : rappel (2)

CONTRAINTE RÉFÉRENTIELLE (clé étrangère) = Un ou plusieurs attributs qui permettent d'exprimer les liens entre les relations.

Client(noCli, nomCli, adr_cli)

Commande(nuCom, datCom, adr_livr, noCli)

Une commande doit référencer un client existant : les valeurs prises par noCli de Commande doivent exister dans l'attribut noCli de Client

noCli	nomCli	adr_cli
75	Frédéric	Nice
66	Yamine	Toulouse
58	Dominique	Nancy
68	Pierre	Paris

Client

noCom	datCom	adr_livr	noCli
1	01/12/19	Paris	75
15	15/01/21	Toulouse	58
18	20/01/20	Nice	58
13	08/11/19	Paris	66
22	14/02/21	Lyon	97

?

Commande

2. Passage du modèle objet au modèle logique
Exemple : Description textuelle

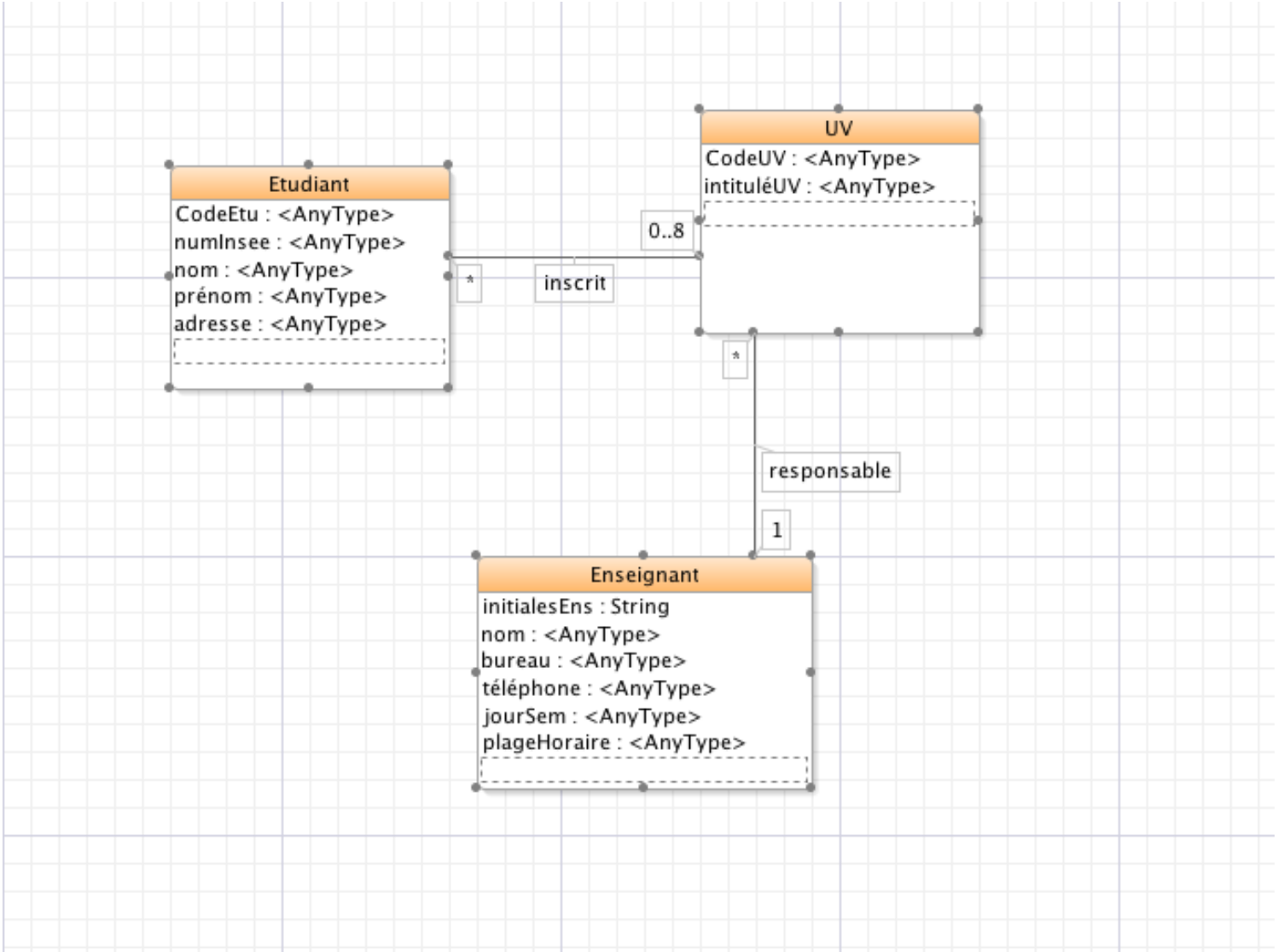
1.1 Inscription des étudiants

Lors de son inscription en début d'année scolaire, chaque étudiant remplit une fiche sur laquelle il indique certains renseignements comme son numéro d'identification nationale (`ninsee`), ses nom et prénom (`nom`, `prenom`), son adresse (`adresse`) et la liste des unités de valeurs (UV) qu'il s'engage à suivre (8 au plus sur les 15 possibles). Un code lui est automatiquement attribué (`codetu`).

Une UV est caractérisée par un code (`codeuv`) et un intitulé (`intuv`). Par exemple le code *UV3* identifie *Électronique numérique*. Chaque UV est placée sous la responsabilité d'un enseignant identifié par ses initiales (`initens`) et caractérisé par un nom (`nomens`), un numéro de bureau (`bureauens`) et un numéro de téléphone (`telens`). Cet enseignant se rend disponible un jour de la semaine (`jsem`) et durant une plage horaire précise (`hrens`) afin de donner tout renseignement concernant les UV qu'il dirige.

- Que faut-il modifier pour qu'un enseignant se rende disponible à différents moments (un seul créneau par UV qu'il dirige) ?
- Que faut-il modifier pour que différents créneaux soient disponibles par UV qu'il dirige ?

2. Passage du modèle objet au modèle logique
Exemple : Diagramme de classes



2. Passage du modèle objet au modèle logique
Exemple : Code SQL

1 · exo1SQL.sql · 2010-08-27 15:38 · Pierre Valarcher

```
CREATE TABLE SCHEMA1.ENSEIGNANTS
  (
    ENSEIGNANTS_ID NUMBER NOT NULL ,
    INITIALES_ENS  VARCHAR2(20) ,
    NOM            VARCHAR2(20) ,
    BUREAU         VARCHAR2(20) ,
    TÉLÉPHONE      VARCHAR2(20) ,
    JOUR_SEM       VARCHAR2(20) ,
    PLAGE_HORAIRE  VARCHAR2(20) ,
    CONSTRAINT ENSEIGNANTS_PK PRIMARY KEY ( ENSEIGNANTS_ID ) ENABLE
  );
CREATE TABLE SCHEMA1.INSCRITS
  (
    INSCRITS_ID  NUMBER NOT NULL ,
    ETUDIANTS_ID NUMBER ,
    UVS_ID       NUMBER ,
    CONSTRAINT INSCRITS_PK PRIMARY KEY ( INSCRITS_ID ) ENABLE
  );
CREATE TABLE SCHEMA1.UVS
  (
    UVS_ID     NUMBER NOT NULL ,
    CODE_UV    VARCHAR2(20) ,
    INTITULÉUV VARCHAR2(20) ,
    END_UV     NUMBER NOT NULL ,
    CONSTRAINT UVS_PK PRIMARY KEY ( UVS_ID ) ENABLE
  );
CREATE TABLE SCHEMA1.ETUDIANTS
  (
    ETUDIANTS_ID NUMBER NOT NULL ,
    CODE_ETU     VARCHAR2(20) ,
    NUM_INSEE    VARCHAR2(20) ,
    NOM          VARCHAR2(20) ,
    PRÉNOM       VARCHAR2(20) ,
    ADRESSE      VARCHAR2(20) ,
    CONSTRAINT ETUDIANTS_PK PRIMARY KEY ( ETUDIANTS_ID ) ENABLE
  );
ALTER TABLE SCHEMA1.INSCRITS ADD CONSTRAINT INSCRIT_ETUDIANT FOREIGN KEY
(
  ETUDIANTS_ID
)
REFERENCES SCHEMA1.ETUDIANTS
(
  ETUDIANTS_ID
)
ON
DELETE CASCADE ENABLE;
ALTER TABLE SCHEMA1.INSCRITS ADD CONSTRAINT INSCRIT_UV FOREIGN KEY ( UVS_ID ) REFERENCES
SCHEMA1.UVS ( UVS_ID ) ON
DELETE CASCADE ENABLE;
ALTER TABLE SCHEMA1.UVS ADD CONSTRAINT RESPONSABLE FOREIGN KEY ( END_UV ) REFERENCES
SCHEMA1.ENSEIGNANTS ( ENSEIGNANTS_ID ) ENABLE;
COMMENT ON TABLE SCHEMA1.ENSEIGNANTS
IS
  'UML to Offline Database Transform: Table ENSEIGNANTS created from UML class
"Enseignant"
Primary key constraint ENSEIGNANTS_PK created by default';
  COMMENT ON TABLE SCHEMA1.INSCRITS
IS
  'UML to Offline Database Transform: Table INSCRITS created from many-to-many
association "inscrit"
Primary key constraint INSCRITS_PK created by default
Foreign key constraint INSCRIT_ETUDIANT created for UML association "inscrit_Etudiant"
Foreign key constraint INSCRIT_UV created for UML association "inscrit_UV"';
  COMMENT ON TABLE SCHEMA1.UVS
```

2. Passage du modèle objet au modèle logique

Modèle objet	Modèle relationnel
Classe	Relation
Attribut	Attribut
Association 0..1, 1..1	Attribut
Association 0..n, n..m	Relation

2.1 Règle R1

- Chaque **classe** devient une **relation**. L'identifiant de la classe devient la clé primaire de la relation.

R1

`Client(noCli, nomCli, adr_cli)`

R1

`Commande(nuCom, datCom, adr_livr)`

Client

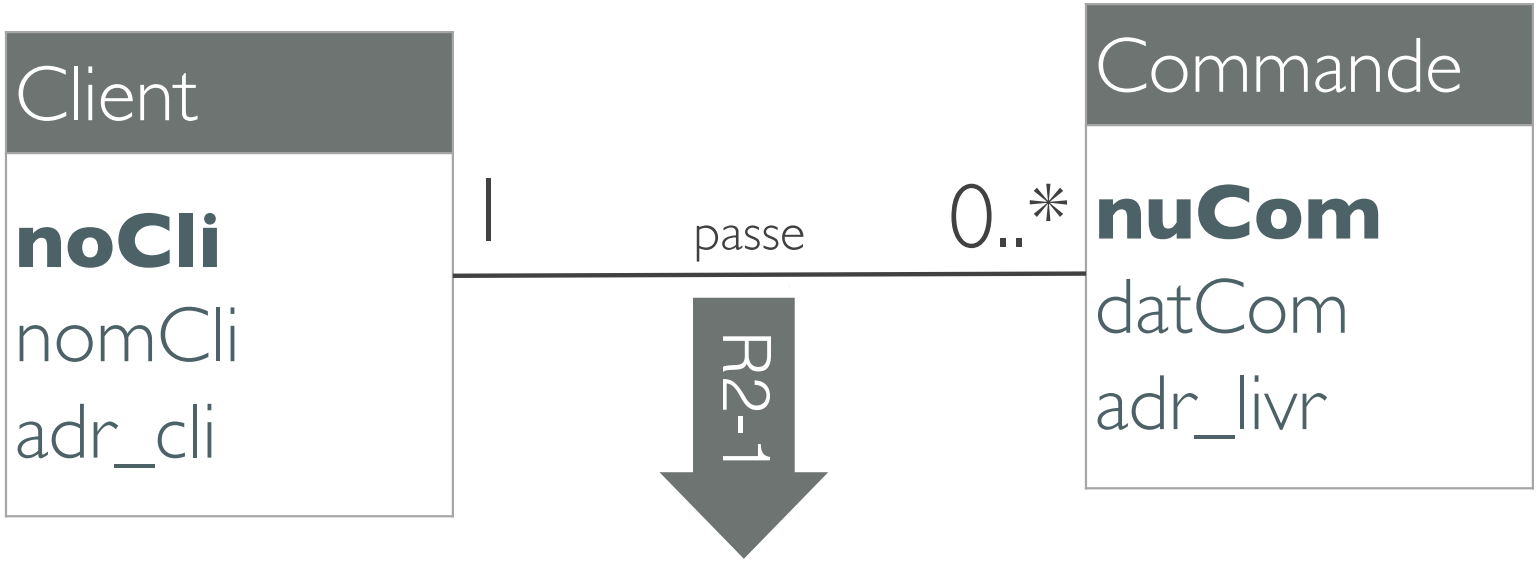
noCli	nomCli	adr_cli
75	Frédéric	Nice
66	Yamine	Toulouse
58	Dominique	Nancy
68	Pierre	Paris

Commande

noCom	datCom	adr_livr
1	01/12/19	Paris
15	15/01/21	Toulouse
18	20/01/20	Nice
13	08/11/19	Paris
22	14/02/21	Lyon

2.2 Règles R2 : associations binaires
R2-1 : 1 à plusieurs

- **Associations 1-à-plusieurs** : Il faut rajouter un attribut de type clé étrangère dans la relation "***fils***" de l'association. L'attribut porte le nom de la clé primaire de la relation "***père***" de l'association



`Client(noCli, nomCli, adr_cli)`

`Commande(nuCom, datCom, adr_livr, noCli)`

Client

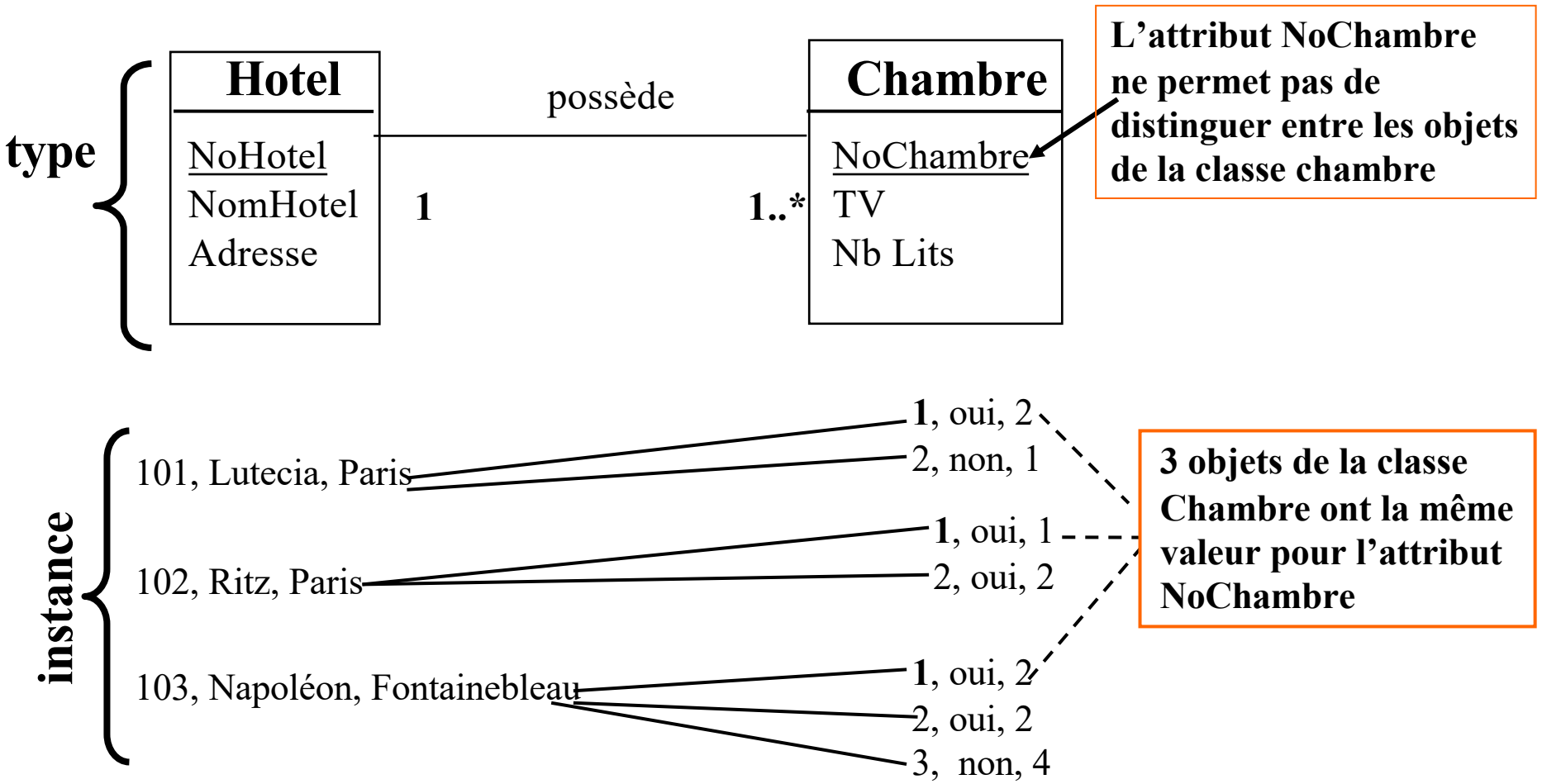
noCli	nomCli	adr_cli
75	Frédéric	Nice
66	Yamine	Toulouse
58	Dominique	Nancy
68	Pierre	Paris

Commande

noCom	datCom	adr_livr	noCli
1	01/12/19	Paris	75
15	15/01/21	Toulouse	58
18	20/01/20	Nice	58
13	08/11/19	Paris	66

2.2 Règle R2-1 : identifiant relatif (Rappel 1)

Exemple : représenter les hôtels et les chambres à louer

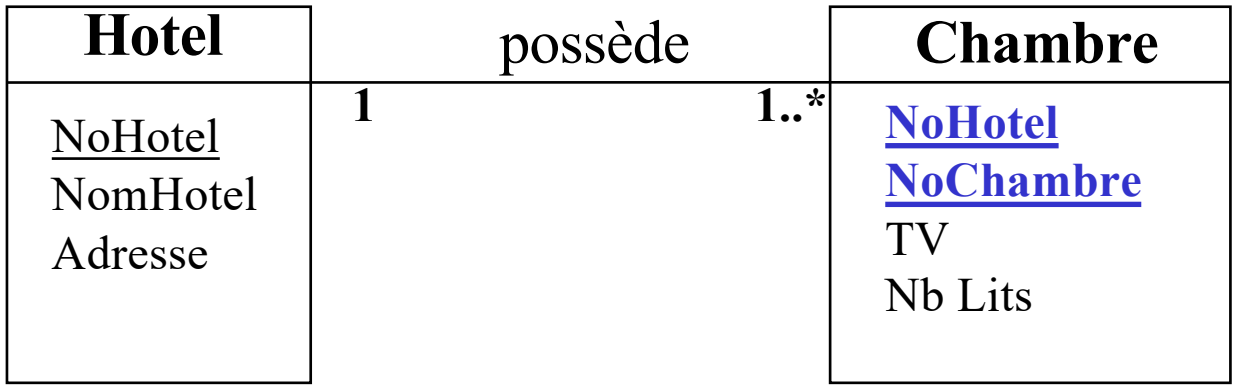


Une chambre dépend fortement de l'hôtel à laquelle elle appartient

2.2 Règle R2-1 : identifiant relatif (Rappel 2)

Solution

L'identifiant de la classe d'objet est constitué du numéro de séquence et de l'identifiant de la classe d'objet dont il dépend.

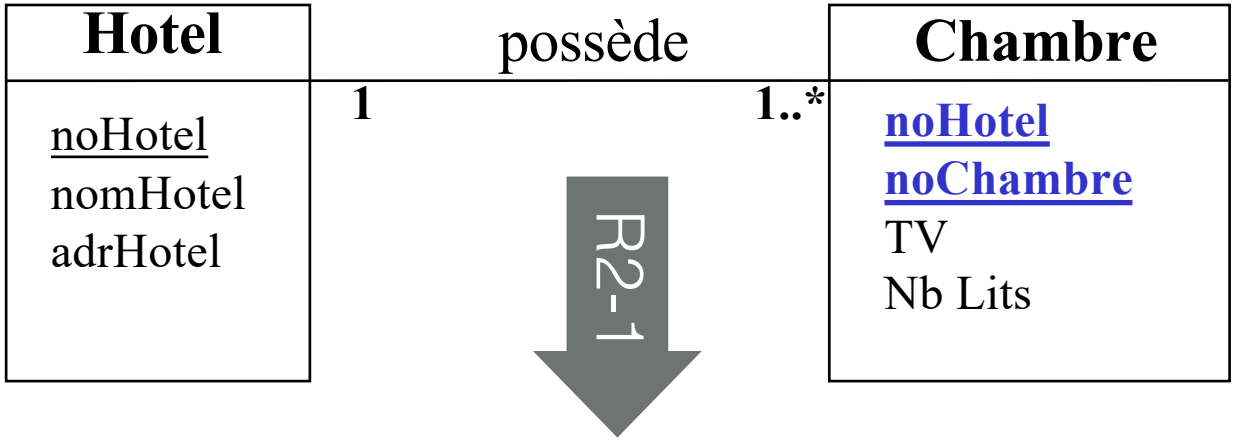


L'identifiant de la classe Chambre est composé des attributs NoHotel+ NoChambre.

On l'appelle identifiant relatif

2.2 Règle R2-1 : identifiant relatif

- **Associations 1-à-plusieurs avec identifiant relatif** : même règle + la clé de la relation *fils* est **l'identifiant relatif**.



Hotel(noHotel, nomHotel, adrHotel)

Chambre(noHotel, noChambre, TV, NbLits)

Hotel

noHotel	nomHotel	adrHotel
75	La Prom	Nice
66	Le Capitole	Toulouse
58	Le Palais	Nancy
68	Montmartre	Paris

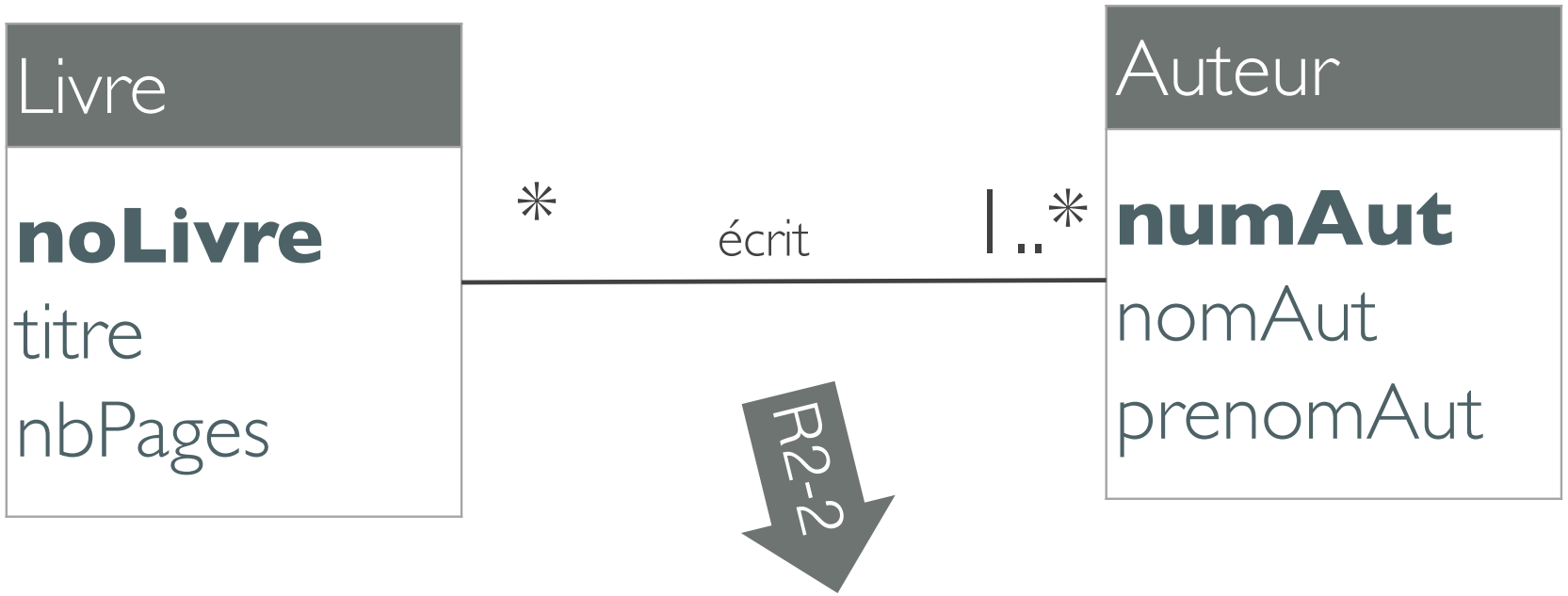
Chambre

noHotel	noChambre	TV	NbLits
75	12	oui	2
75	13	non	1
58	1	non	3
66	13	oui	1
68	7	oui	1

2.2 Règle R2-2 : Association binaire plusieurs-à-plusieurs

L'association devient une relation dont la **clé primaire** est composée par la **concaténation des clés** des classes connectées à l'association.

Chaque **attribut** devient **clé étrangère** si la classe connectée dont il provient devient une relation en vertu de la règle R1.

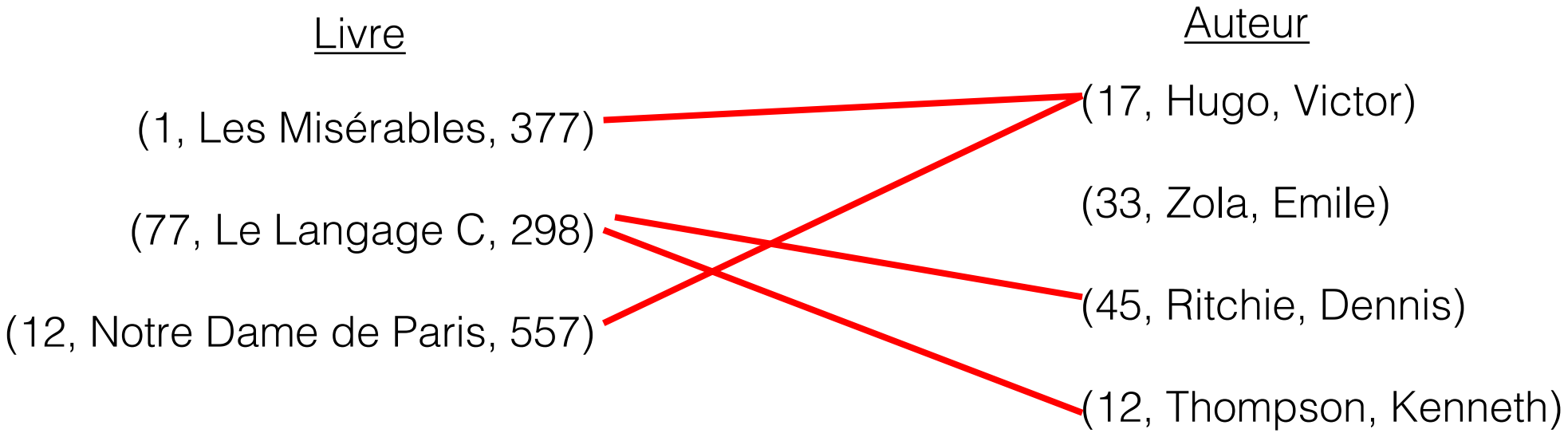


Livre(noLivre, titre, nbPages)

Ecrit(noLivre, numAut)

Auteur(numAut, nomAut, prenomAut)

Objets et liens du diagramme de classes



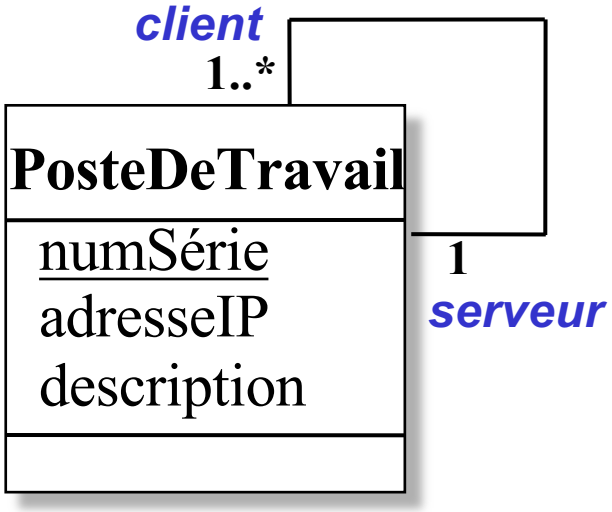
Tuples du schéma relationnel

Livre	Ecrit	Auteur
(1, Les Misérables, 377)	(1,17)	(17, Hugo, Victor)
(77, Le Langage C, 298)	(77,45)	(33, Zola, Emile)
(12, Notre Dame de Paris, 557)	(77,12)	(45, Ritchie, Dennis)
	(12,17)	(12, Thompson, Kenneth)

2.2 Règle R2-3 : Association réflexive

Rappel : Une association réflexive est une association binaire qui fait intervenir deux fois la même classe

On veut représenter les postes de travail du département informatique, certains sont des postes client (salle TP), d'autres sont des serveurs



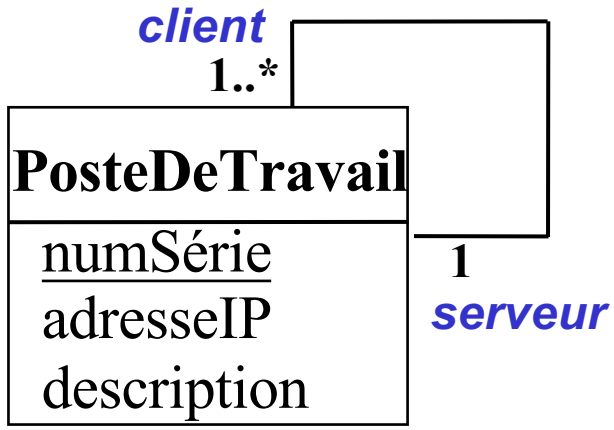
– Un poste serveur est lié à 1 ou plusieurs postes client

– Un poste client est relié à un seul serveur

Le nommage des rôles est essentiel à la clarté du diagramme.

2.2 Règle R2-3 : Association réflexive

Une association réflexive est traduite par une relation avec contraintes référentielles. La clé de la relation dépend de la multiplicité de l'association.



PosteDeTravail(numSérie, adresseIP, description)

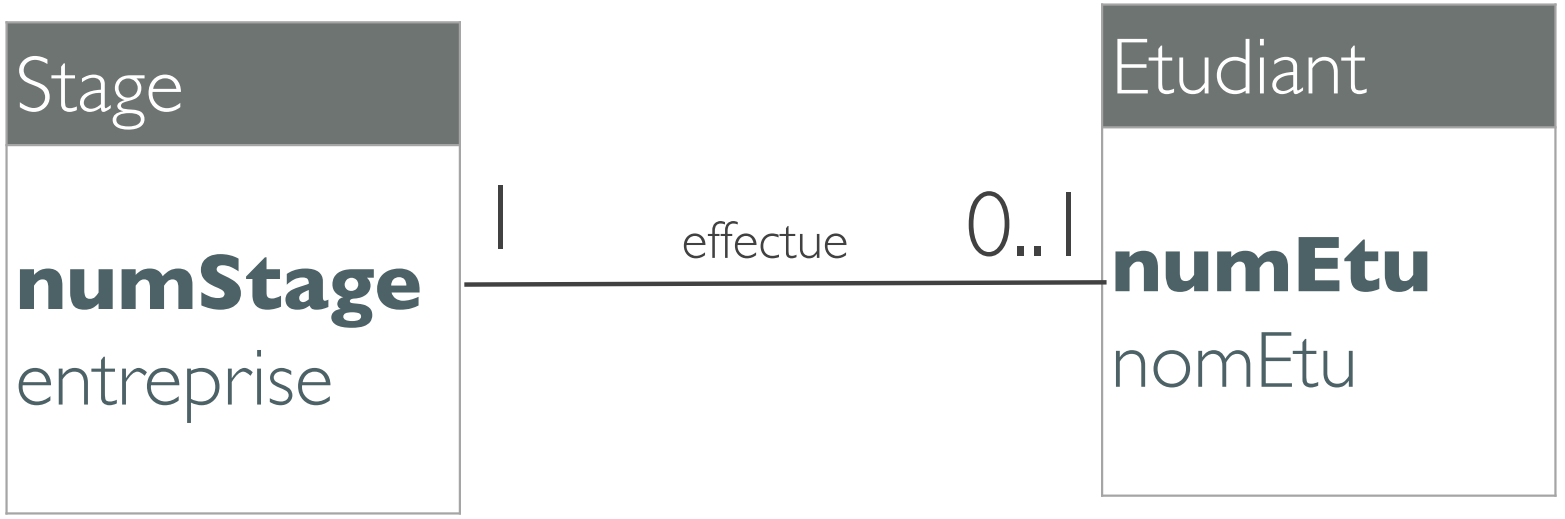
TypePoste(client, serveur)

Avec la contrainte que le client est différent du serveur

2.2 Règle R2-4 : Association avec multiplicités maximales à 1

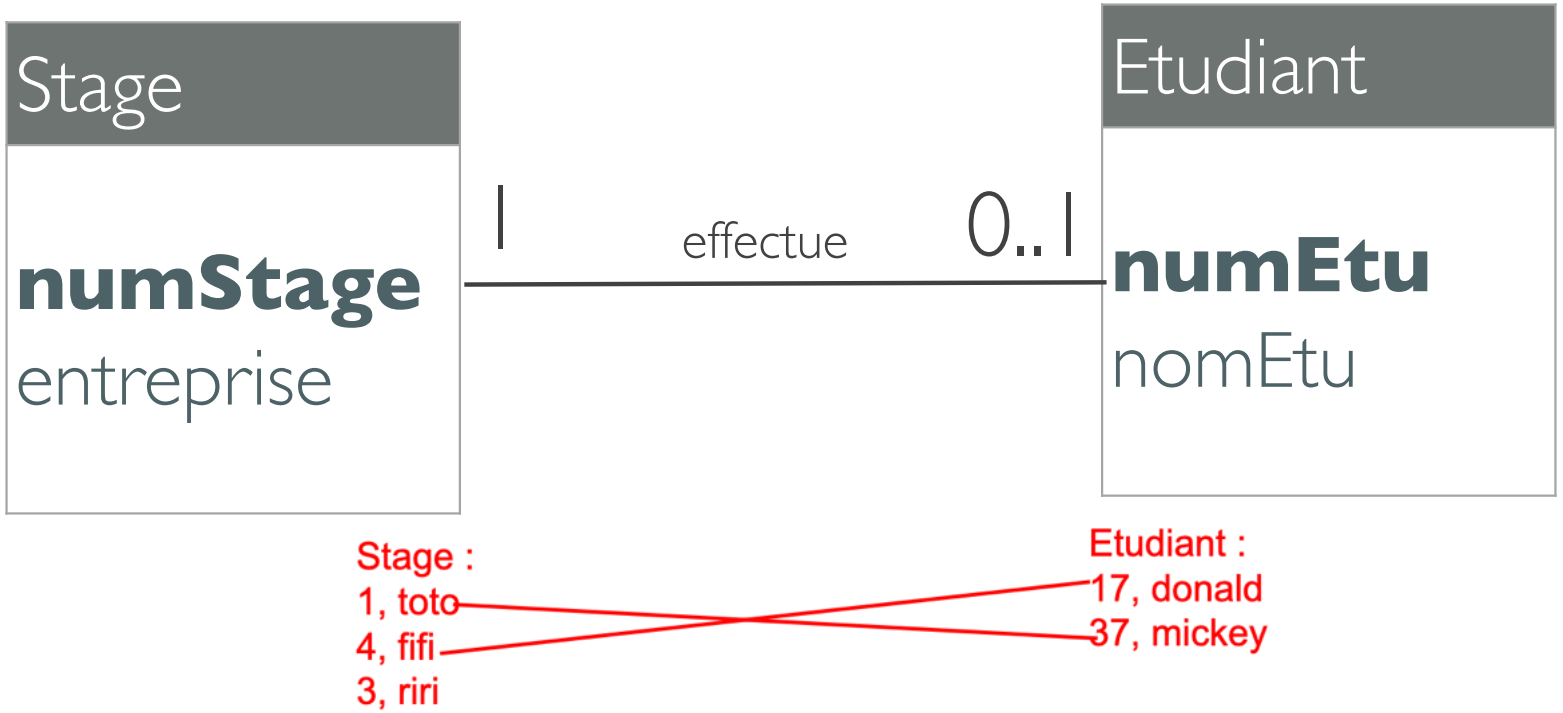
Association avec multiplicités maximales à 1 : il faut ajouter un attribut clé étrangère dans la relation dérivée de la classe ayant la multiplicité minimale égale à un. L'attribut porte le nom de la clé primaire de la relation dérivée de la classe connectée à l'association.

- si les 2 multiplicités minimales sont 0, on a le choix
- si les 2 multiplicités minimales sont 1 (il faut peut-être fusionner les 2 classes en une...)




```
Stage(numStage, entreprise)

Etudiant(numEtu, nomEtu, numStage )
```

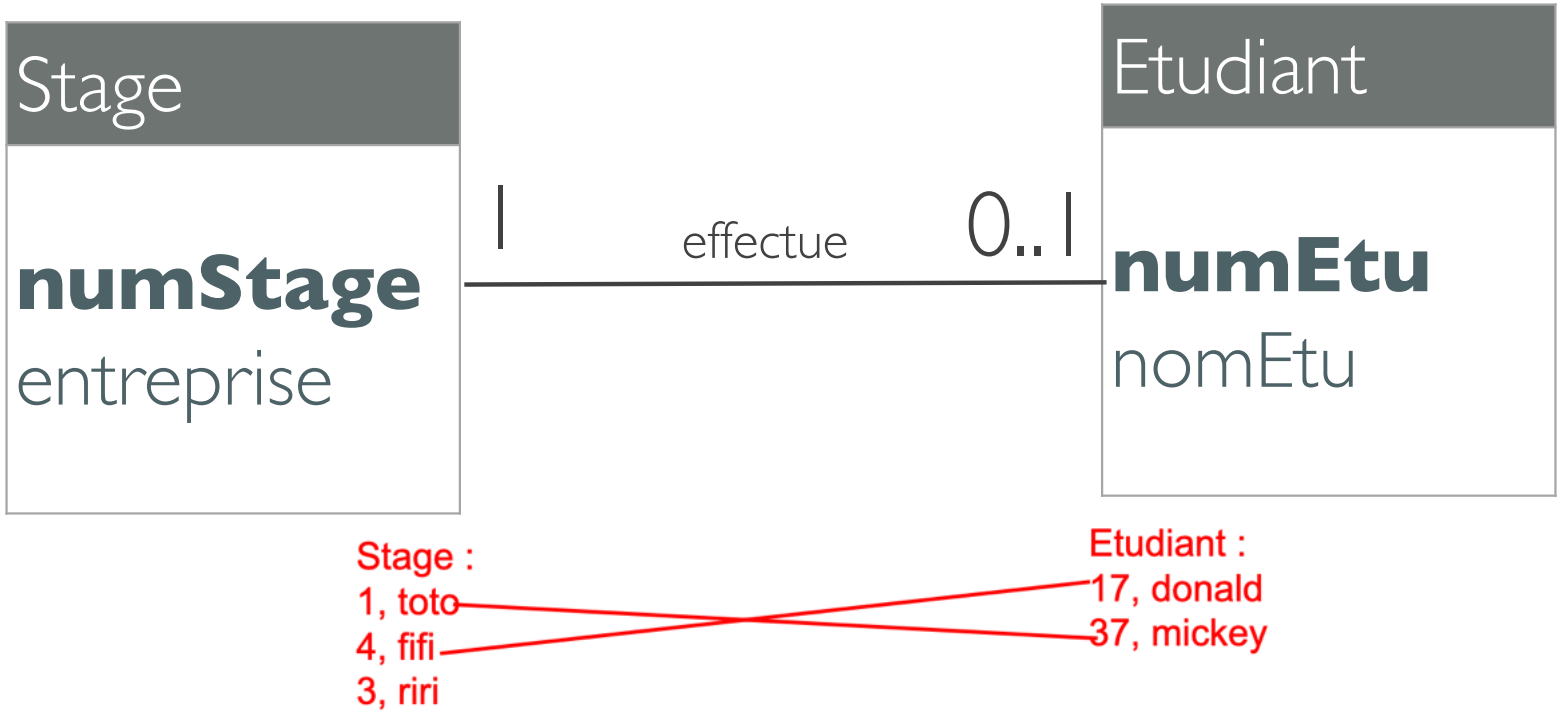


Stage(numStage, entreprise)

(1,toto)
(4,fifi)
(3,riri)

Etudiant(numEtu, nomEtu, numStage)

(17,donald,4)
(37,mickey,1)



Stage (<u>numStage</u>,entreprise,numEtu)

(1,toto,37)
(4,fifi,17)
(3,riri,NULL)

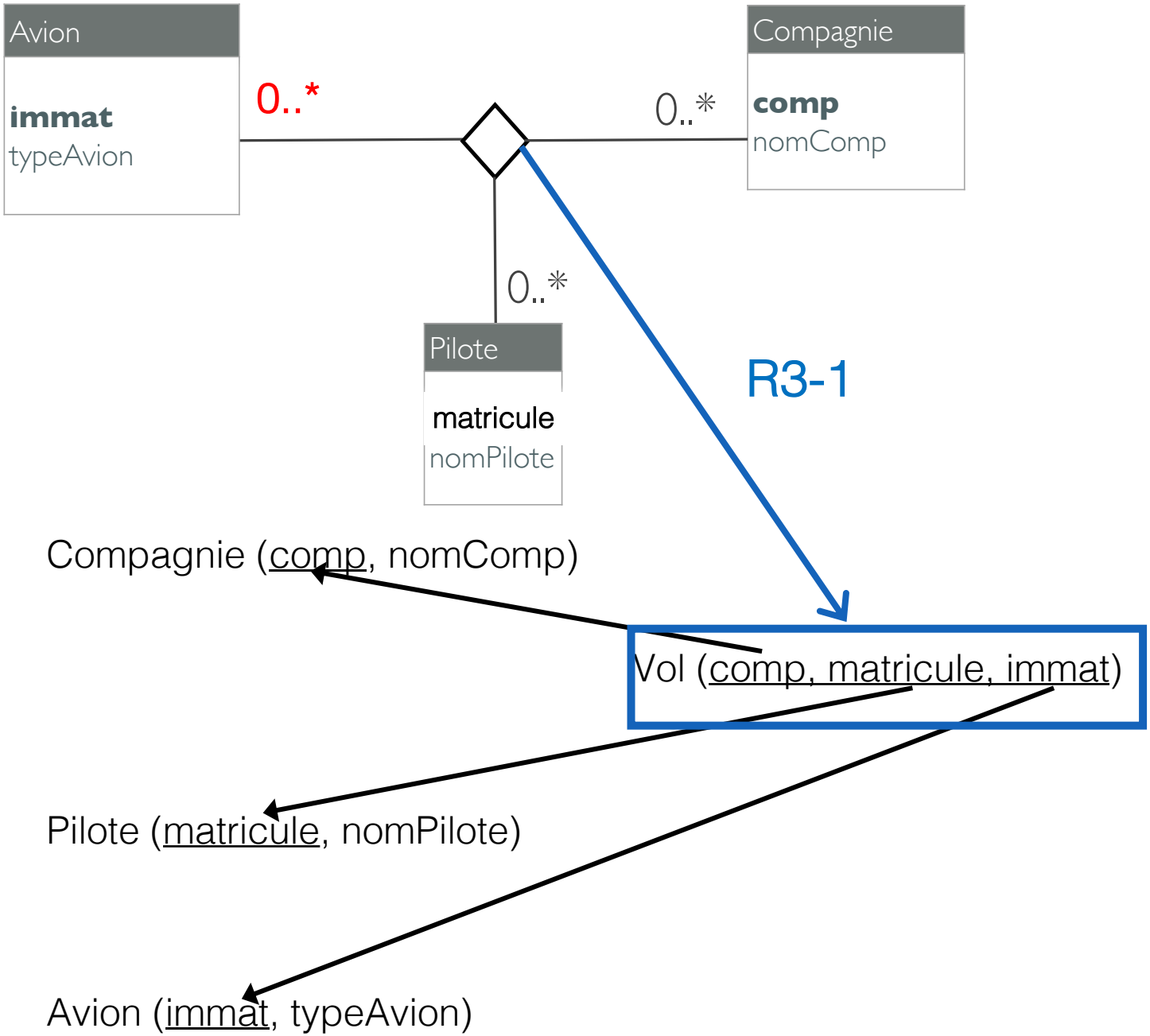
Etudiant (<u>numEtu</u>,nomEtu)

(17,donald)
(37,mickey)

2.3 Règles R3 : associations n-aire
R3-1 : Association n-aire avec multiplicités maximales *

L'association devient une relation dont la **clé primaire** est composée par la **concaténation des clés des classes** connectées à l'association.

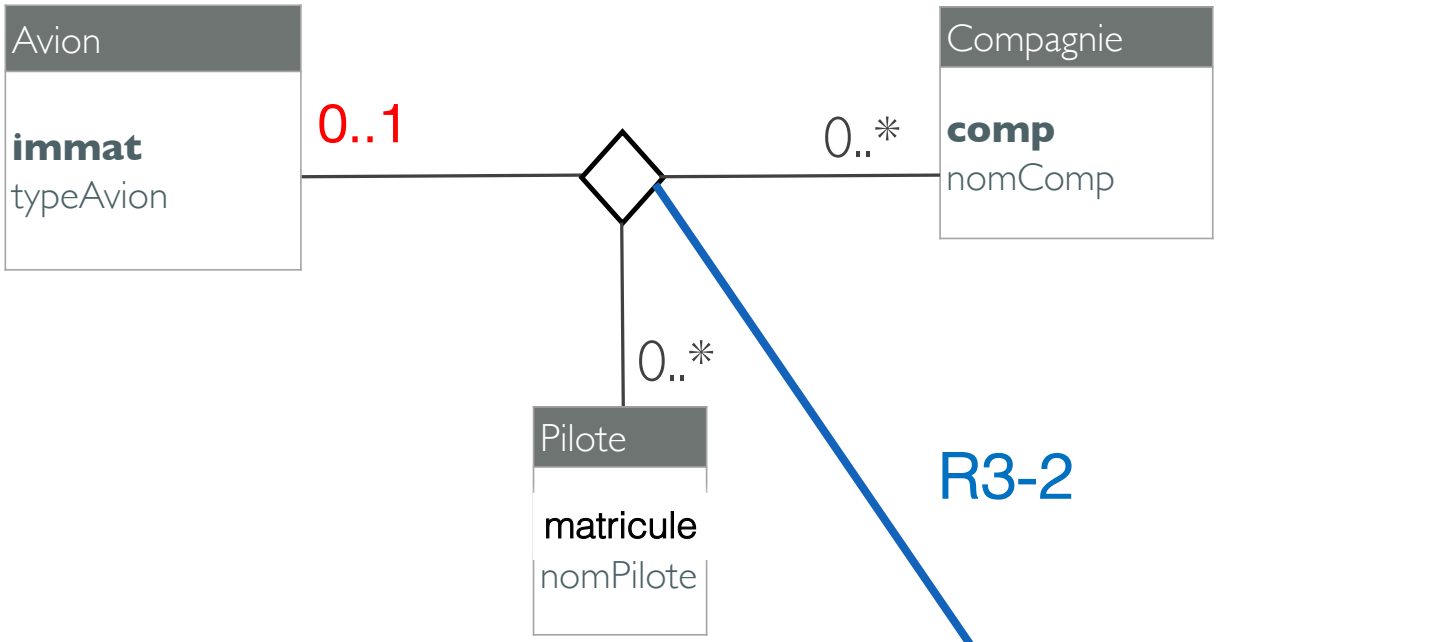
Chaque **attribut** devient **clé étrangère** si la classe connectée dont il provient devient une relation en vertu de la règle R1.


Avion
immat
typeAvion
0..*
0..*
Compagnie
comp
nomComp
0..*
Pilote
matricule
nomPilote
R3-1
Compagnie (comp, nomComp)
Vol (comp, matricule, immat)
Pilote (matricule, nomPilote)
Avion (immat, typeAvion)

2.3 Règle R3-2 : Association n-aire avec une multiplicité maximale à 1

L'association devient une relation dont la **clé primaire** est composée par la **concaténation des identifiants des classes** connectées à l'association avec une **multiplicité *.**

Chaque **attribut** devient **clé étrangère** si la classe connectée dont il provient devient une relation en vertu de la règle R1.



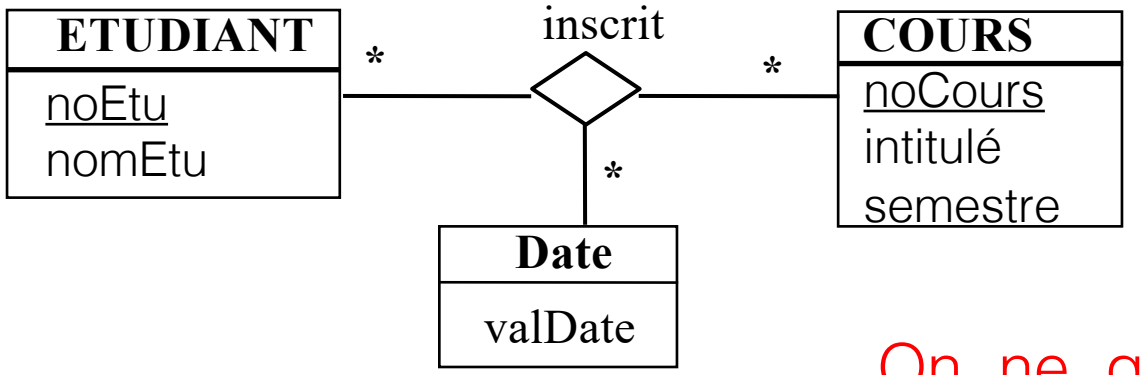
Compagnie (comp, nomComp)

Vol (comp, matricule, immat)

Pilote (matricule, nomPilote)

Avion (immat, typeAvion)

2.3 Règle R3-3 : classe temporelle



ETUDIANT (noEtu, nomEtu)

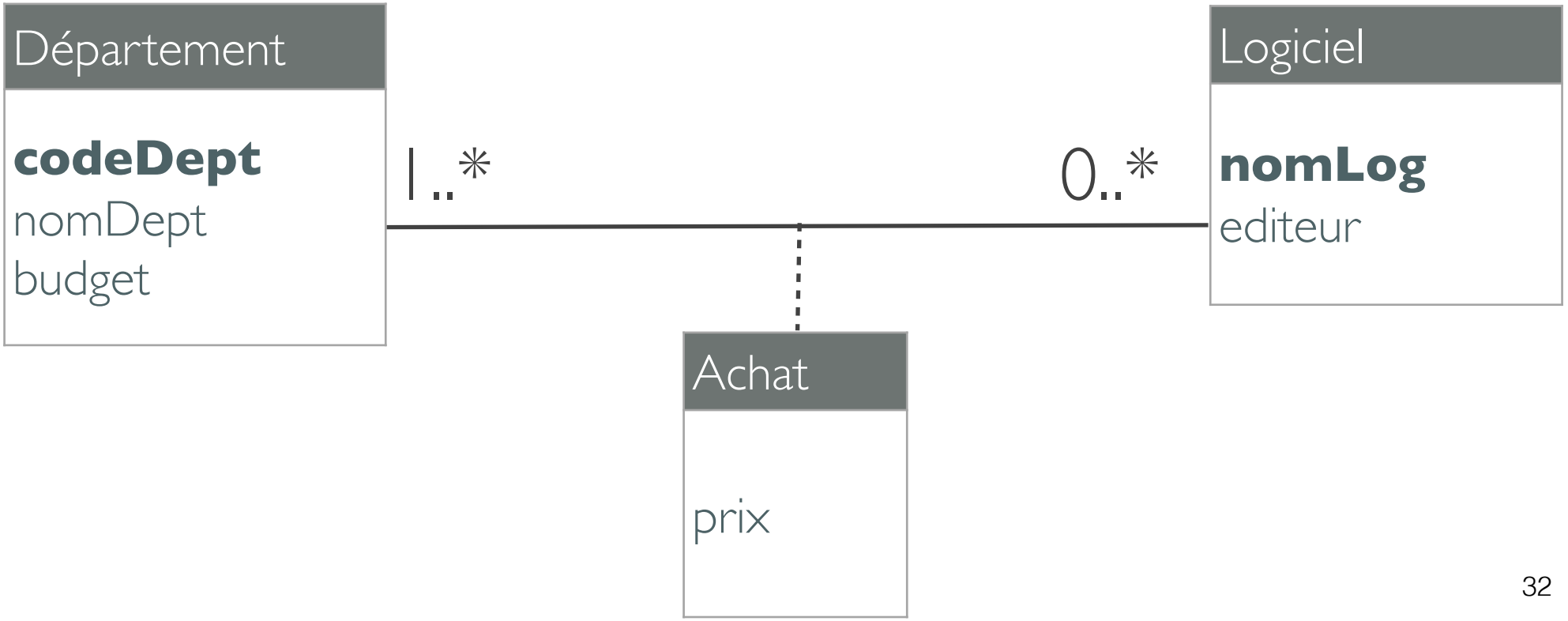
COURS (noCours,intitulé,semestre)

inscrit(noEtu,noCours,valDate)

On ne génère pas de relation Date car c'est une classe temporelle. La clé de *inscrit* est la concaténation de la clé de *ETUDIANT*, la clé de *COURS* et l'attribut *valDate.*

2.4 Règle R4 : classe-association

Rappel : Une association peut avoir des attributs qui dépendent de **toutes** les classes associées. Dans ce cas, l'association devient une classe-association.

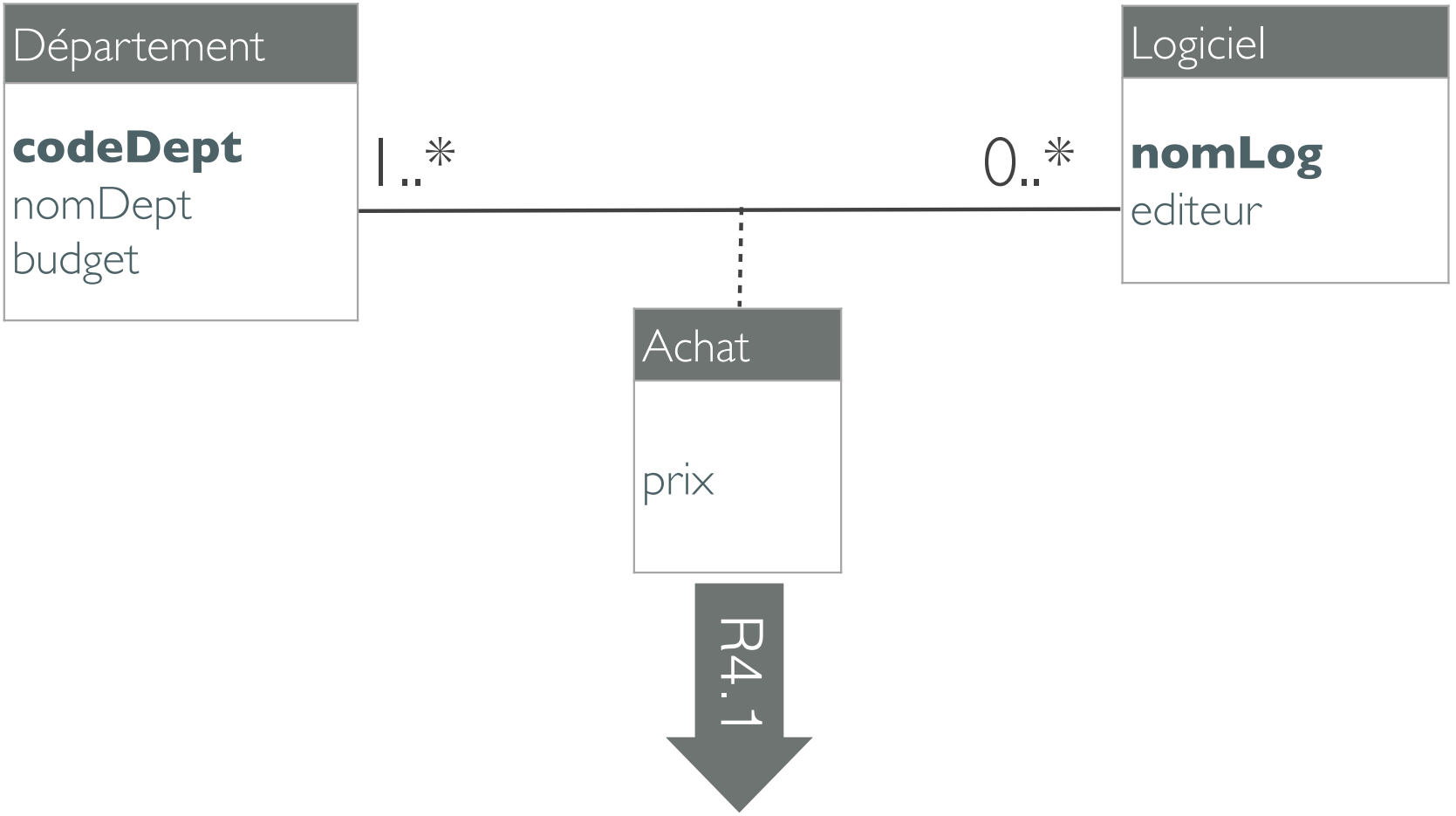


2.4 Règles R4 : classe-association
R4-1 : association binaire

La Classe-Association d'une association binaire devient une relation dont la **clé primaire** est composée par la **concaténation des clés des classes** connectées à la classe-association.

Chaque **attribut** de la clé devient **clé étrangère**

Les attributs de la classe-association deviennent des attributs de la relation



Département (codeDept, nomDept, budget)

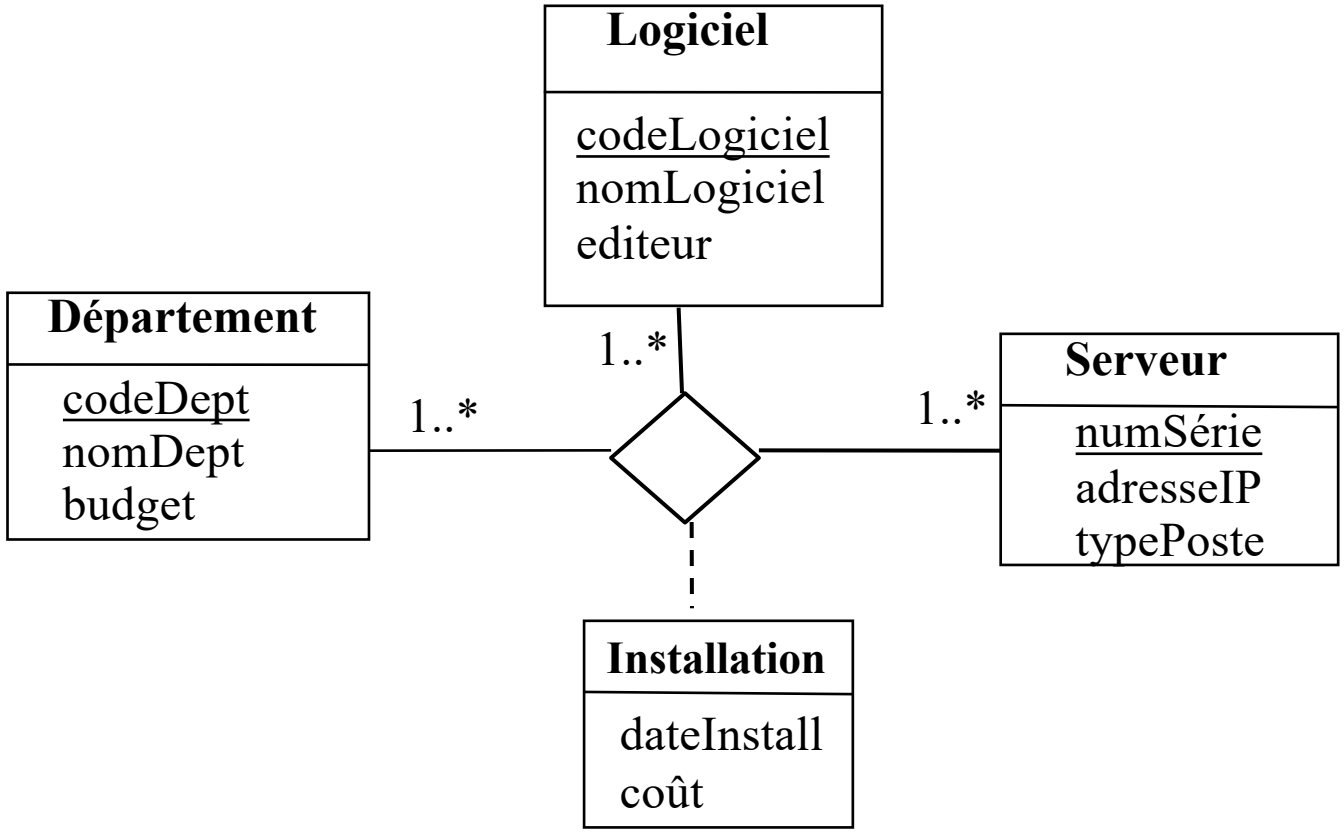
Achat (codeDept, nomLog, prix)

Logiciel (nomLog, editeur)

2.4 Règles R4 : classe-association
R4-2 : association n-aire

La Classe-Association d'une association n-aire ne donne pas une nouvelle relation.

Les attributs de la classe-association deviennent des attributs de la relation associée à l'association n-aire.



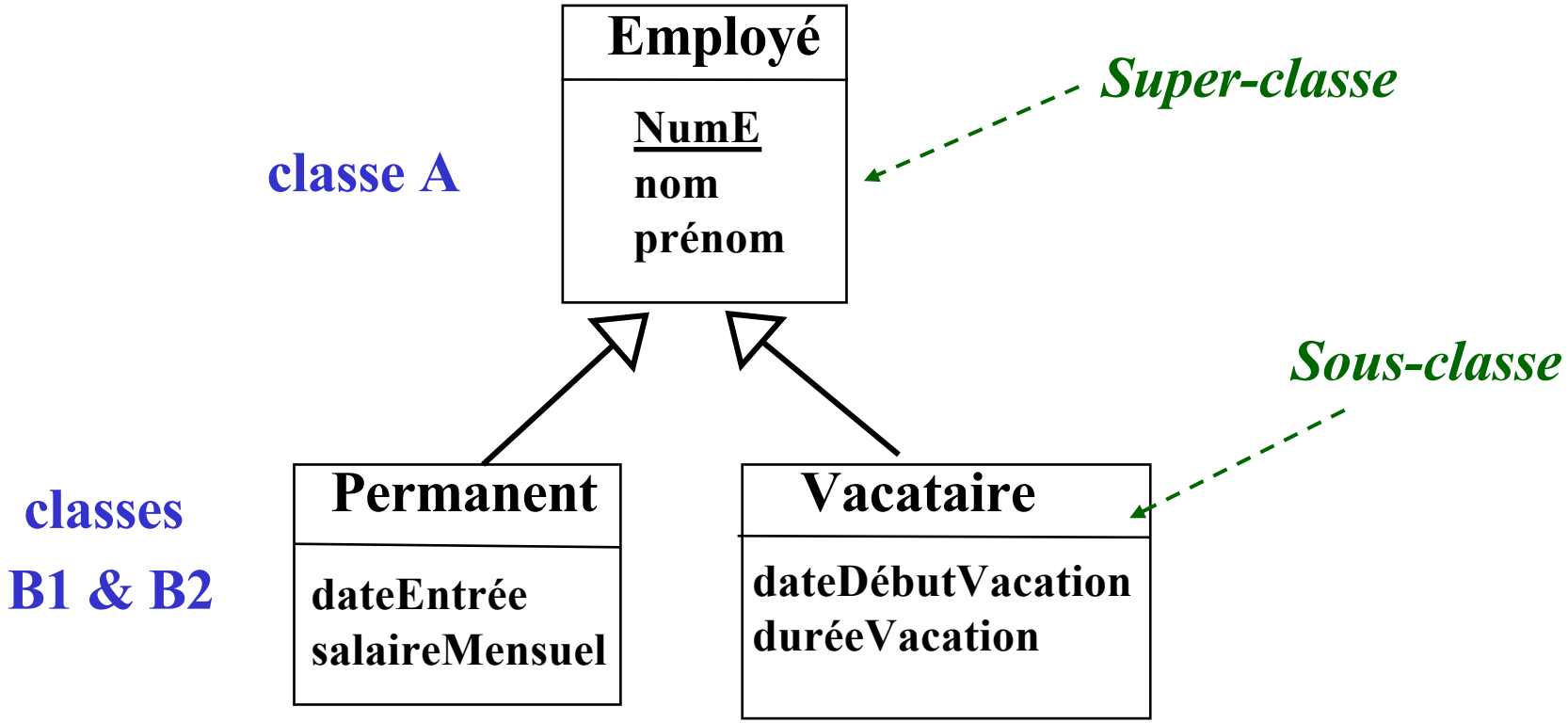
Département (<u>codeDept</u>, nomDept, budget) Serveur (<u>numSérie</u>, adresseIP, typePoste)

Installation (<u>codeDept, codeLogiciel, numSérie</u>, dateInstall, coût)

Logiciel (<u>codeLogiciel</u>, nomLog, editeur)

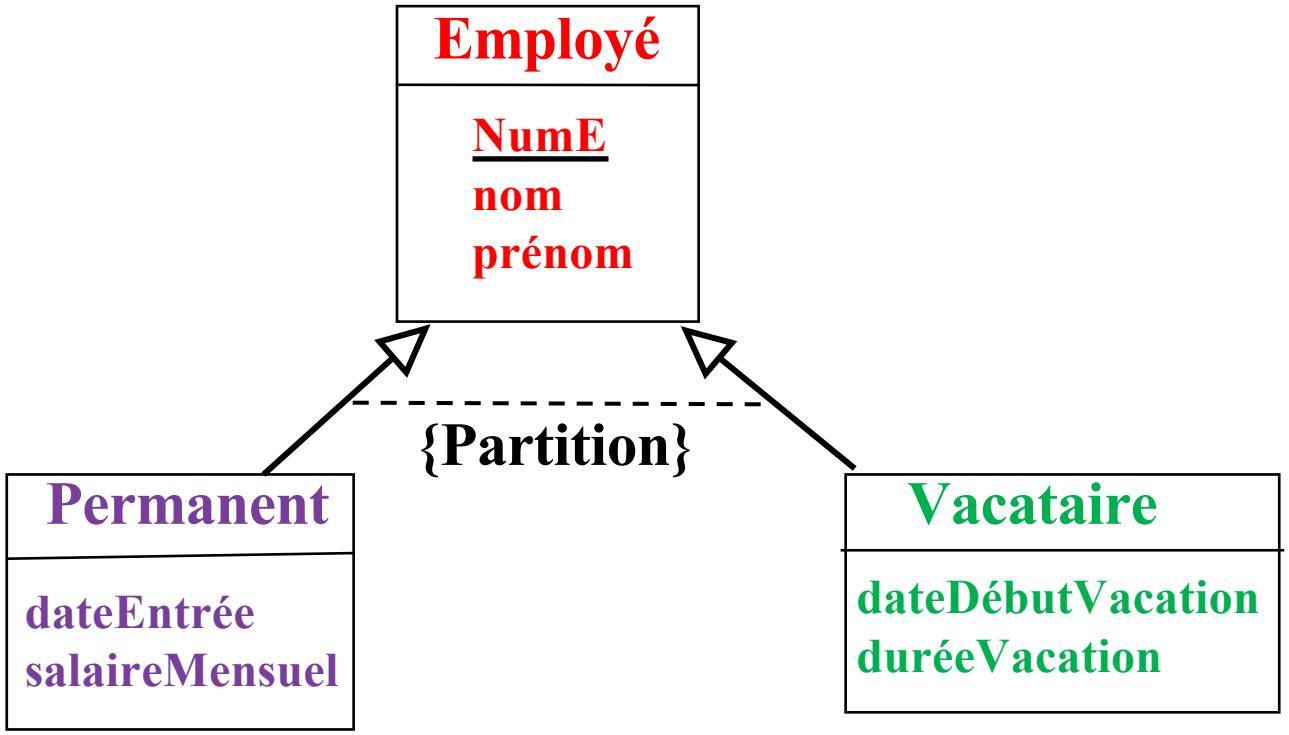
3. Transformation de l'héritage - Rappel

Rappel : l'héritage est une relation entre classes qui permet de représenter les attributs communs dans une classe (super-classe) et les attributs spécifiques dans une autre classe (sous-classes)



• ***Les classes B1 et B2 héritent de la classe A*** **signifie que tous les attributs de la classe A sont également ceux des classes B1 et B2**

3. Héritage - exemple



101, Dupond, Jean, 01/09/1997, 2000€

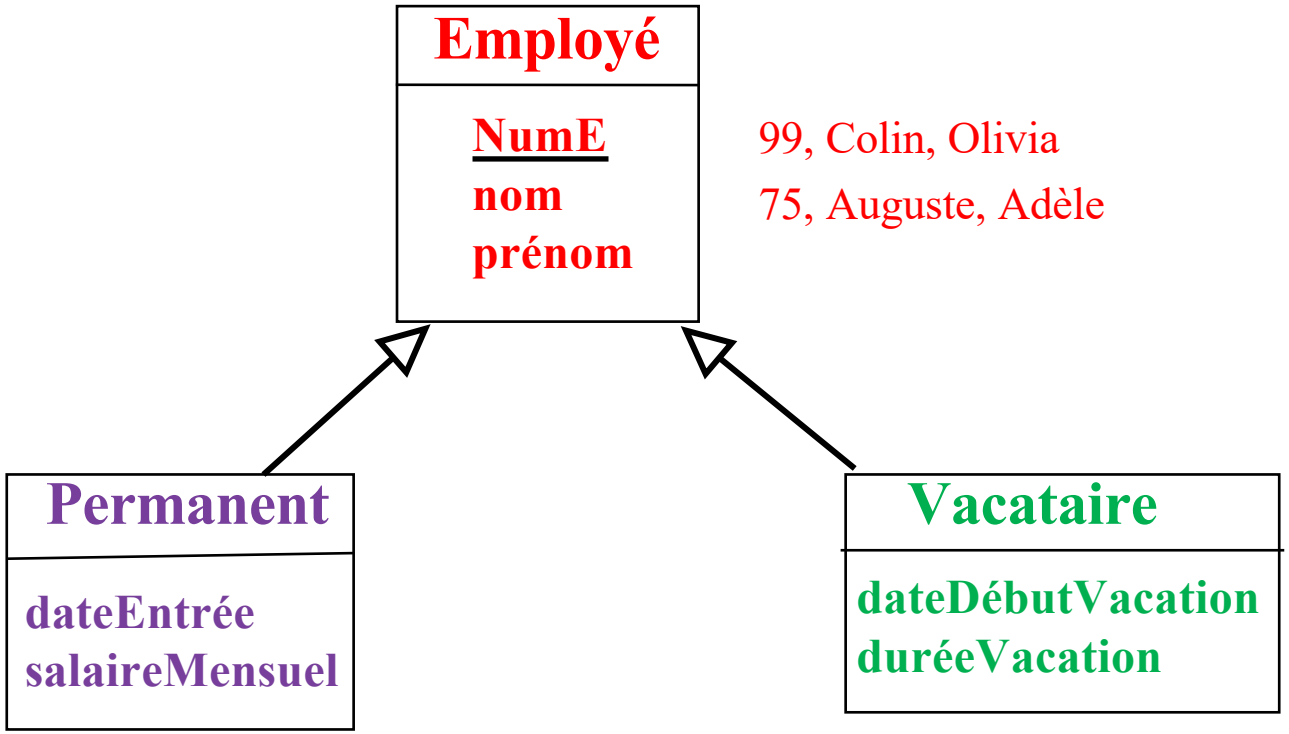
103, Ritz, Frédéric, 01/12/1975, 4000€

104, Napoléon, Bonaparte, 01/01/2000, 10000€

102, Durand, Annie, 19/10/2019, 3mois

105, Michel, Emeline, 17/01/2020, 7jours

3. Héritage - exemple



99, Colin, Olivia

75, Auguste, Adèle

101, Dupond, Jean, 01/09/1997, 2000€

103, Ritz, Frédéric, 01/12/1975, 4000€

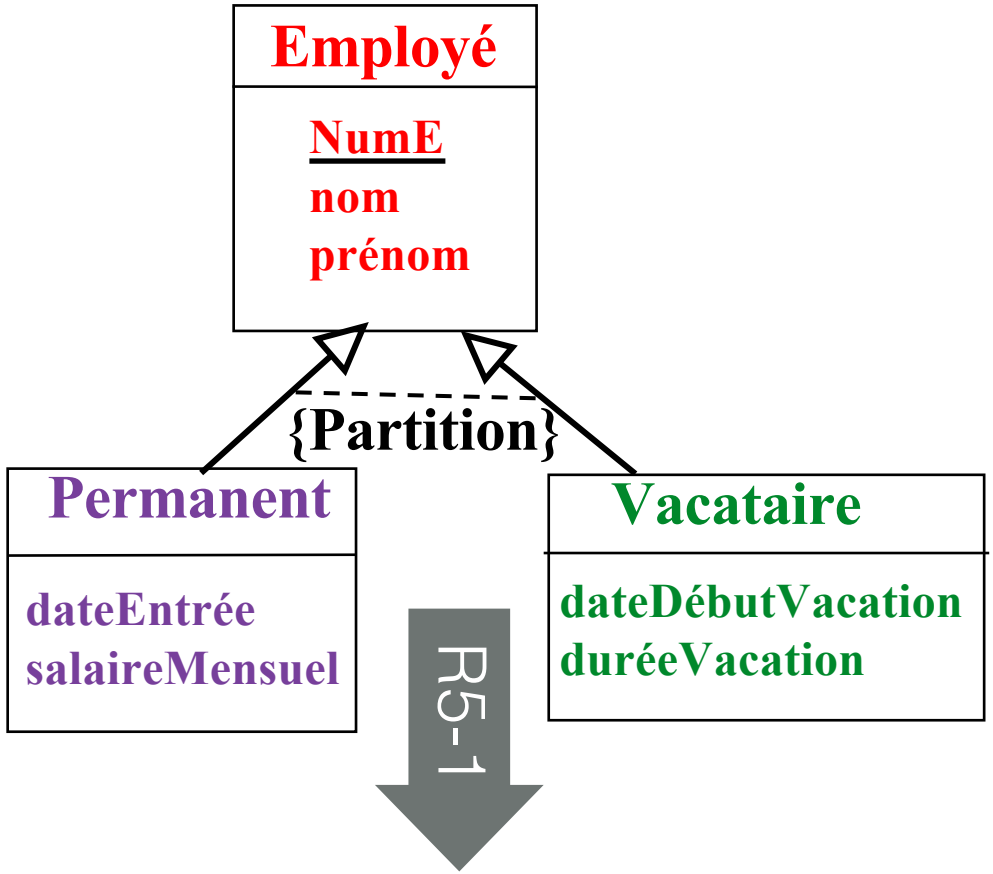
104, Napoléon, Bonaparte, 01/01/2000, 10000€

102, Durand, Annie, 19/10/2019, 3mois

105, Michel, Emeline, 17/01/2020, 7jours

3. Règle 5-1

Héritage par décomposition descendante : s'il existe une contrainte de partition, on garde les sous-classes. Les attributs de la super-classe migrent dans les sous-classes.

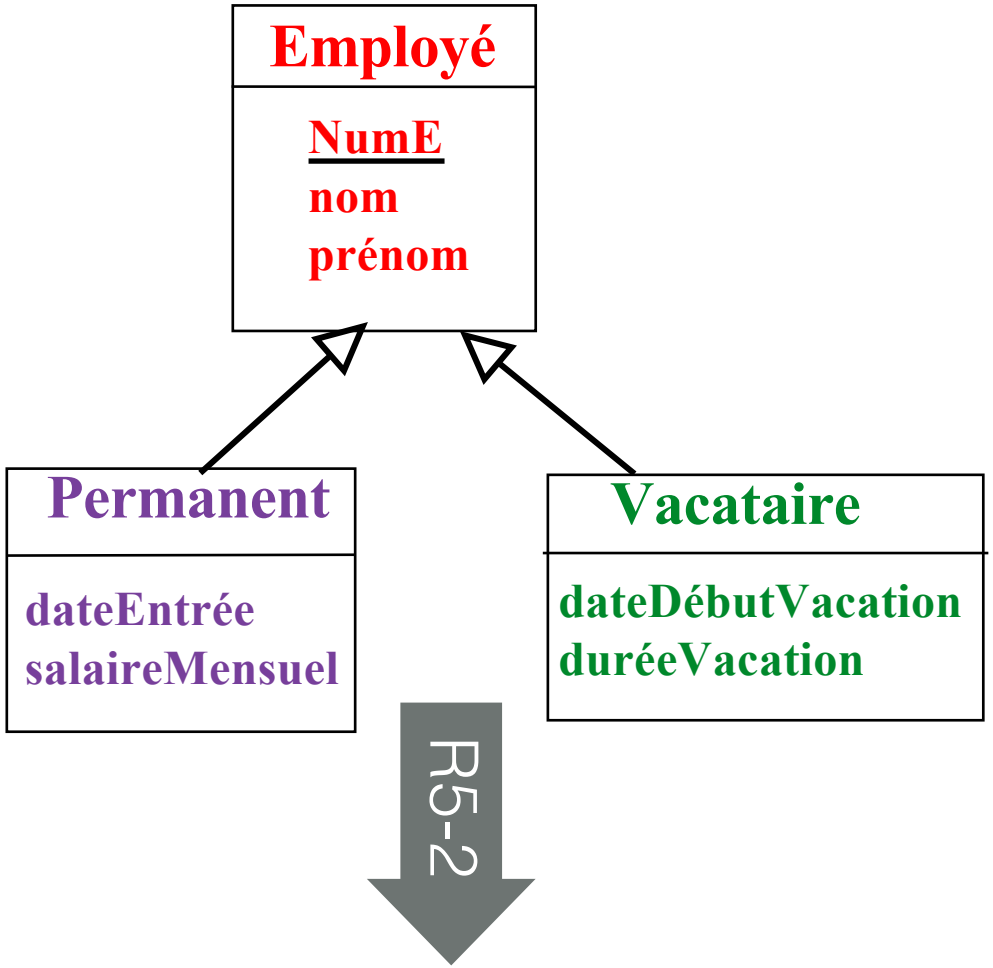


Permanent(NumE,nom,prénom,dateEntrée,salaireMensuel)

Vacataire(NumE,nom,prénom,dateDébutVacation,duréeVacation)

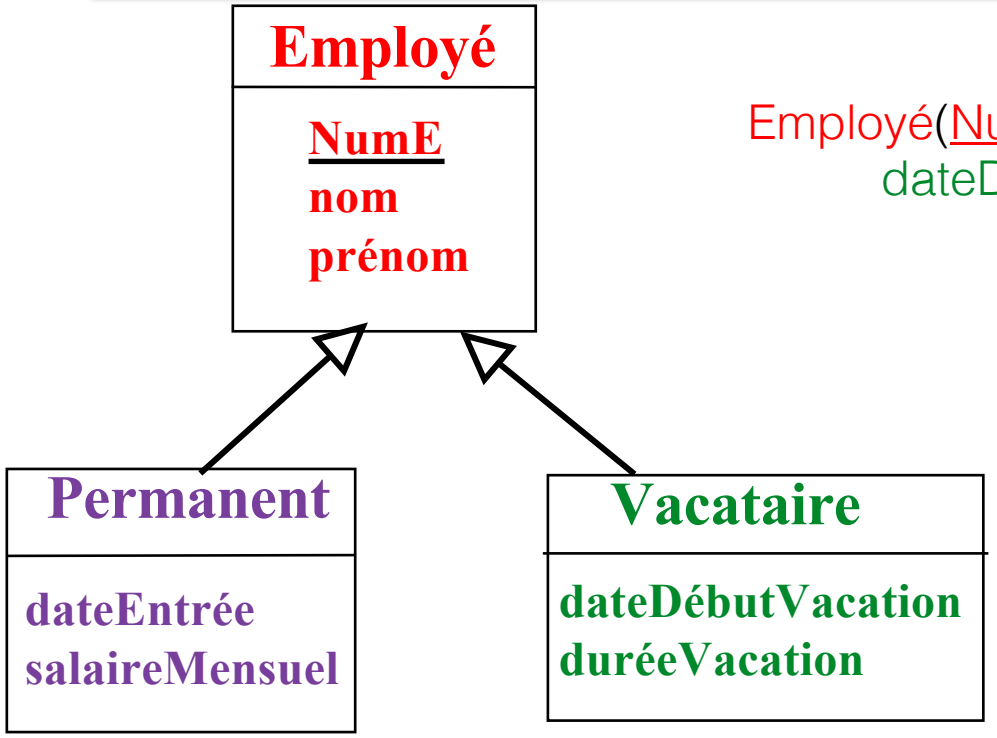
3. Règle 5-2

Héritage par décomposition ascendante : on garde la super-classe et on supprime les sous-classes. Les attributs des sous-classes migrent dans la super-classe.



Employé(NumE,nom,prénom,dateEntrée,salaireMensuel, dateDébutVacation,duréeVacation)

3. Règle 5-2 : exemple



Employé(NumE,nom,prénom,dateEntrée,salaireMensuel,
dateDébutVacation,duréeVacation)

Génère des valeurs nulles : solution adaptée si les sous-classes ont peu d'attributs propres.

101, Dupond, Jean, 01/09/1997, 2000€, NULL, NULL

103, Ritz, Frédéric, 01/12/1975, 4000€, NULL, NULL

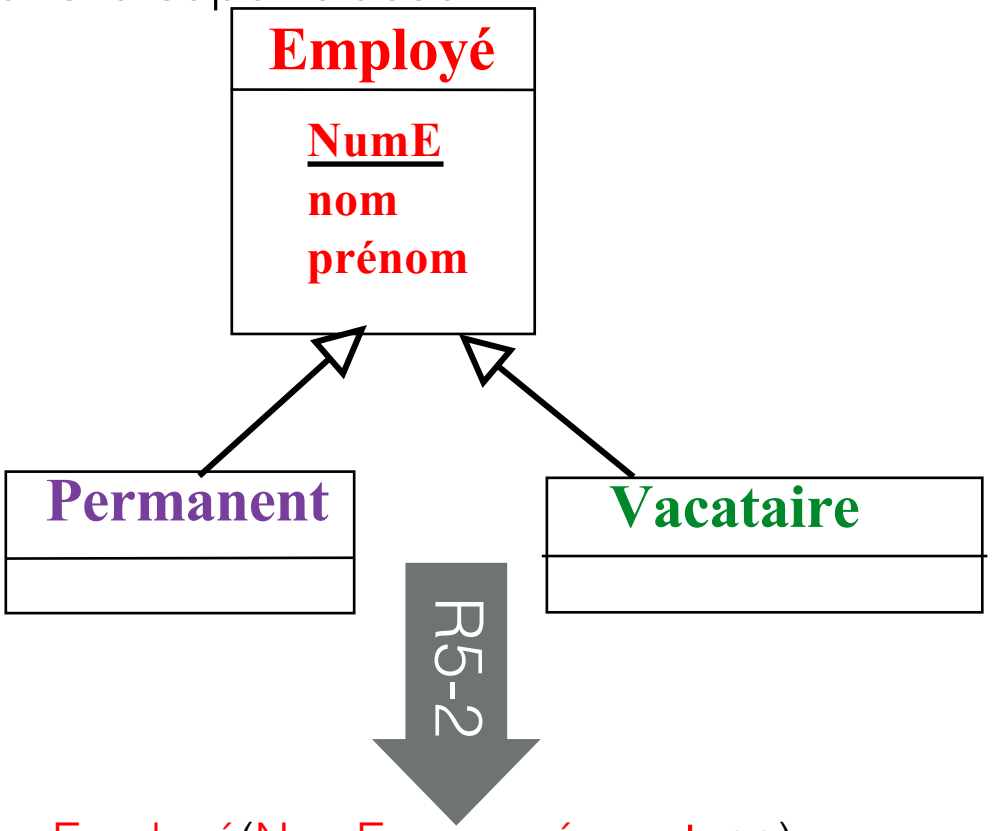
104, Napoléon, Bonaparte, 01/01/2000, 10000€, NULL, NULL

102, Durand, Annie, NULL, NULL, 19/10/2019, 3mois

105, Michel, Emeline, NULL, NULL, 17/01/2020, 7jours

3. Règle 5-2 (cas particulier)

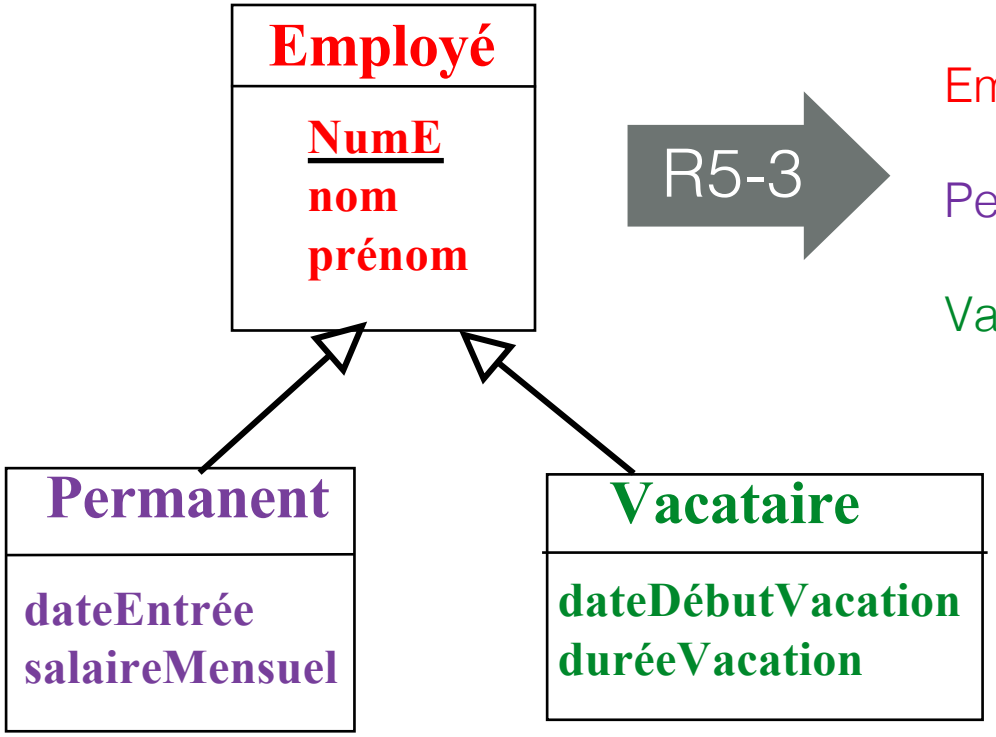
Héritage par décomposition ascendante : on garde la super-classe et on supprime les sous-classes. S'il n'y a pas d'attributs dans les sous-classes, on rajoute un attribut "type" dans la super-classe.



Employé(NumE,nom,prénom,**type**)
Avec type = Permanent ou Vacataire

3. Règle 5-3

Héritage par distinction : on garde la super-classe et les sous-classes. La clé primaire de la super-classe migre dans les sous-classes comme clé primaire et étrangère.



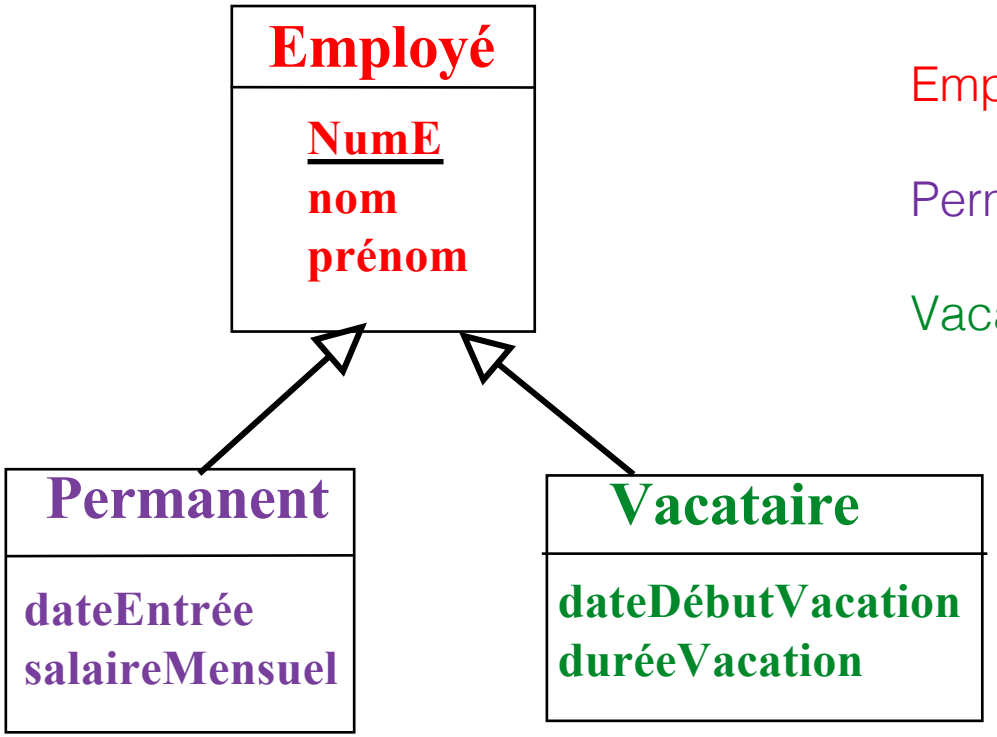
R5-3

Employé (NumE,nom,prénom)

Permanent(NumE,dateEntrée,salaireMensuel)

Vacataire(NumE,dateDébutVacation,duréeVacation)

3. Règle 5-3 : exemple 1 : cas général



Employé (NumE,nom,prénom)

Permanent(NumE,dateEntrée,salaireMensuel)

Vacataire(NumE,dateDébutVacation,duréeVacation)

Employé

99, Colin, Olivia

75, Auguste, Adèle

101, Dupond, Jean

103, Ritz, Frédéric

104, Napoléon, Bonaparte

102, Durand, Annie

105, Michel, Emeline

Permanent

101, 01/09/1997, 2000€

103, 01/12/1975, 4000€

104, 01/01/2000, 10000€

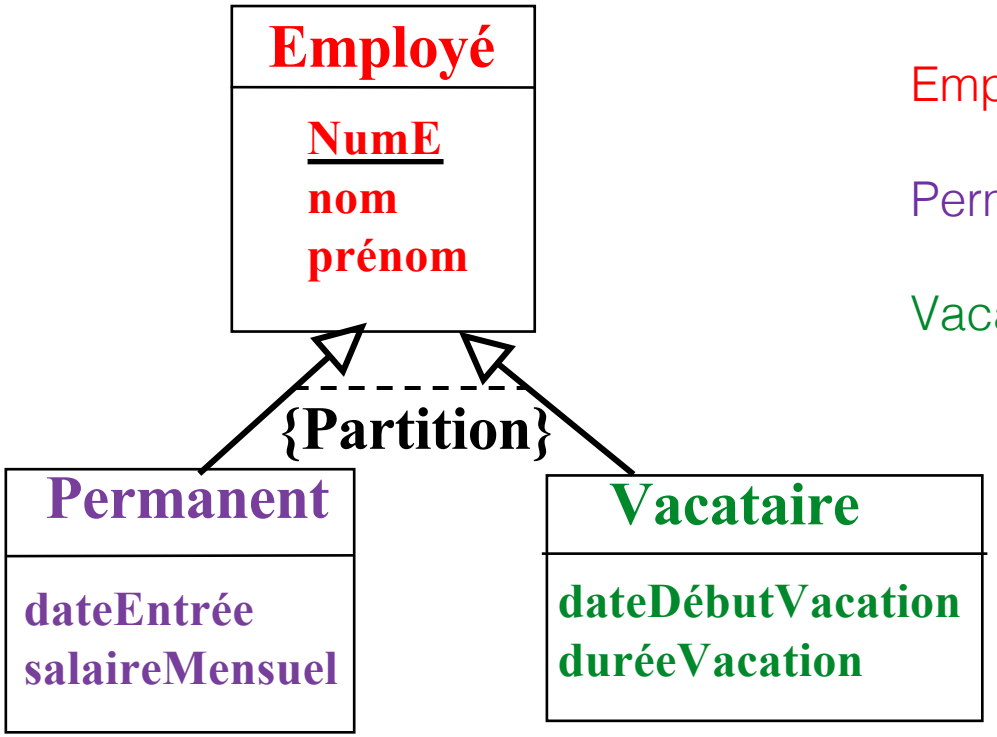
Vacataire

102, 19/10/2019, 3mois

105, 17/01/2020, 7jours

103, 01/12/2019, 2jours

3. Règle 5-3 : exemple 2 : partition



Employé (NumE,nom,prénom)

Permanent(NumE,dateEntrée,salaireMensuel)

Vacataire(NumE,dateDébutVacation,duréeVacation)

Employé

101, Dupond, Jean
103, Ritz, Frédéric
104, Napoléon, Bonaparte
102, Durand, Annie
105, Michel, Emeline

Permanent

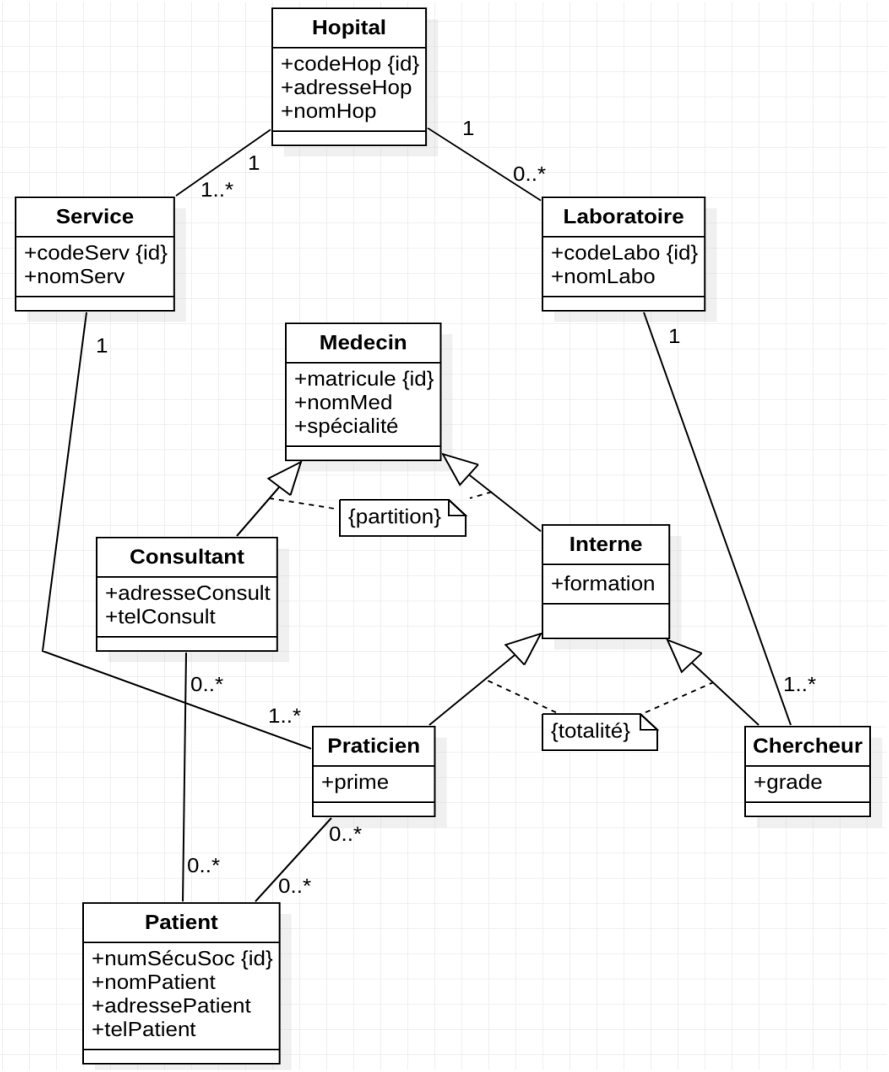
101, 01/09/1997, 2000€
103, 01/12/1975, 4000€
104, 01/01/2000, 10000€

Vacataire

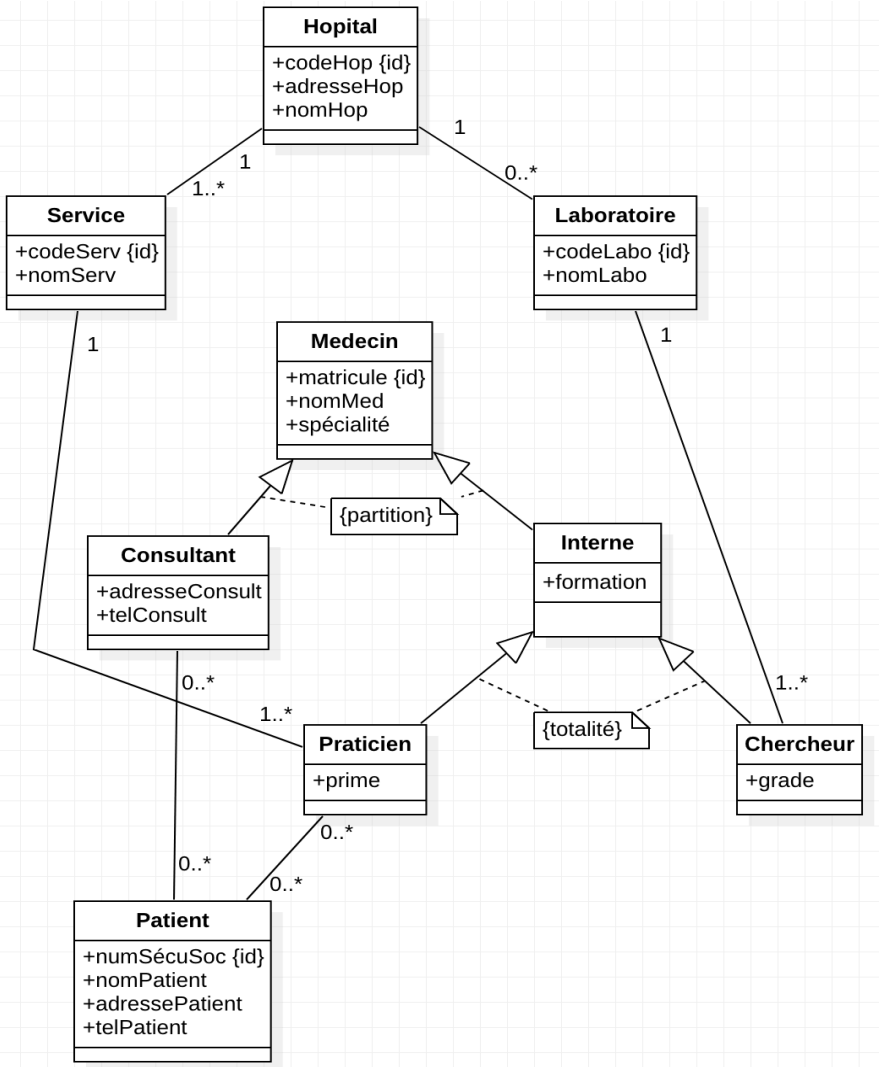
102, 19/10/2019, 3mois
105, 17/01/2020, 7jours

Solution adaptée si la super-classe a des associations propres.

Exemple 1



1ère étape : on vérifie que toutes les classes ont un identifiant.



2ème étape : Suppression de l'héritage

- héritage "Médecin"

Comme il y a une partition et que Medecin n'a pas d'association, on garde seulement Consultant et Interne

Consultant (matricule, nomMed, spécialité, adresseConsult, telConsult)

Interne (matricule, nomMed, spécialité, formation)

- héritage "Interne"

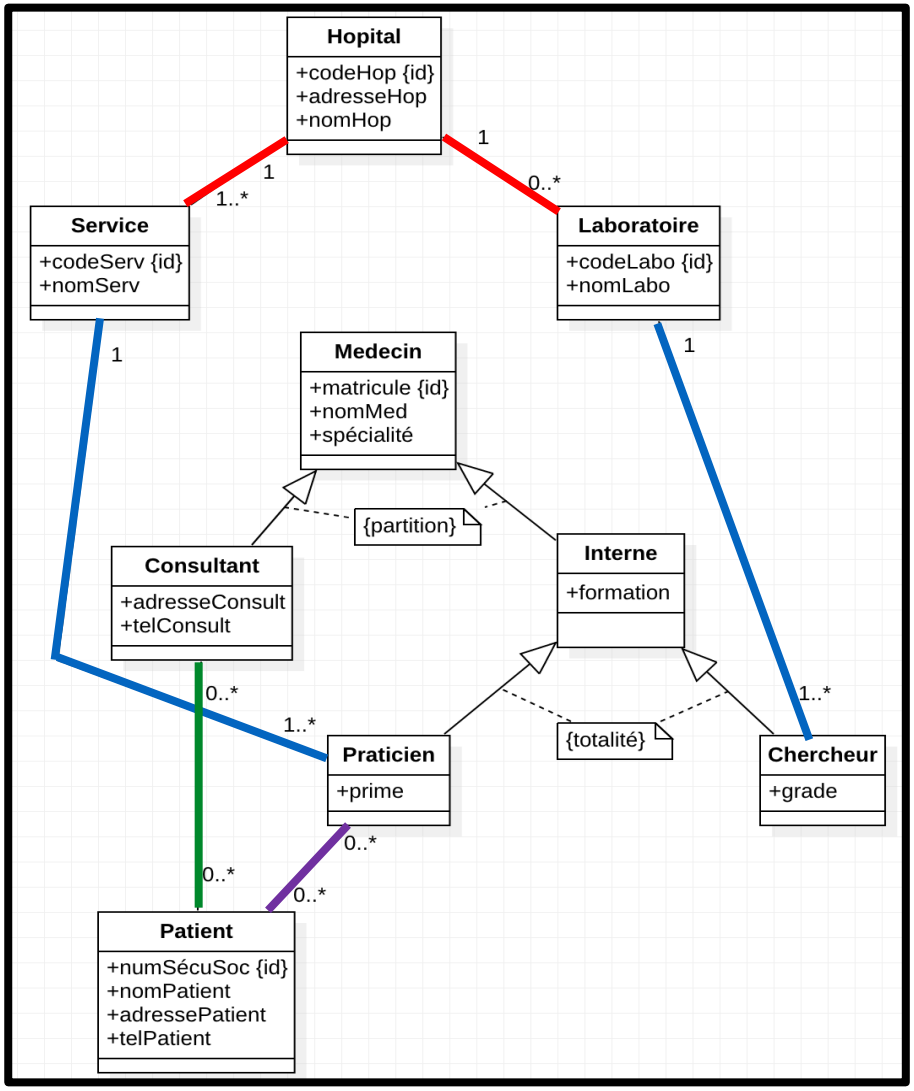
On garde les 3. On ne garde pas seulement Praticien et Chercheur car il y a totalité (ie, un interne peut être soit Praticien, soit Chercheur, soit les 2). On ne garde pas seulement Interne car il faut rajouter des contraintes pour la traduction des associations.

Interne (matricule, nomMed, spécialité, formation)

Praticien(matricule, prime)

Chercheur(matricule,grade)

Exemple 1



3ème étape : Suppression des associations

Hopital(codeHop, adresseHop, nomHop)

Service(codeServ, nomServ, codeHop)

Praticien(matricule, prime, codeServ)

Laboratoire(codeLabo,nomlabo,codeHop)

Chercheur(matricule,grade,codeLabo)

Interne (matricule, nomMed, spécialité, formation)

Patient(numSS,nomPatient,adressePatient,telPatient)

Pat-Prat(matricule, numSS)

Pat-Consult(matricule,numSS)

Consultant (matricule, nomMed, spécialité, adresseConsult, telConsult)