# Des diagrammes objets vers le modèle Relationnel

# Résumé des concepts

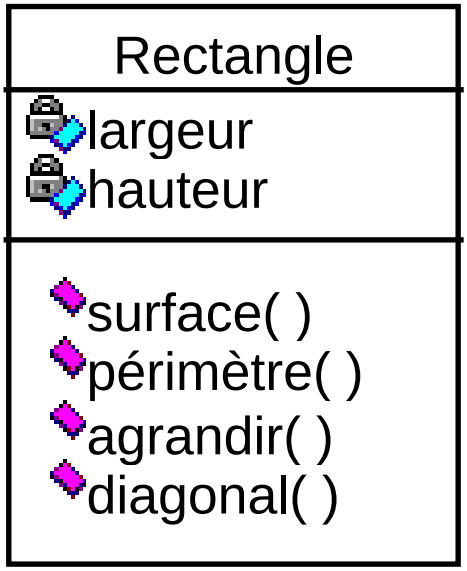
| Modèle Objet | Modèle Relationnel |
|---|---|
| Classe | Relation |
| Attribut | Attribut |
| Méthode | (vue, fonction, attribut, procédure) |
| **Association** | |
| 0..1,1..1 | Attribut |
| 0..n, n..m | Relation |
| Association attribuée | (voir la cardinalité de l'association) |
| **Agrégation** | |
| 0..1,1..1 | Attribut (+valeur null) |
| 0..n, n..m | Relation |
| **Généralisation** | |
| | Attribut (+valeur null) |
| | Relation |

Objet >> implémentation >> Relationnel

Relationnel >> interprétation >> Objet

# Classe -> Relation

Exemple:




identifier les objets ?

->ajouter une clé artificielle si une clé naturelle n'existe pas

Oid = Numéro d'objet

```
Create Table Rectangle (
        Id_rectangle integer primary key,
        largeur    number,
        hauteur    number)
```

# Que faire avec les méthodes ?

Pas de langage procédural (PL/SQL, Visual Basic, …)

**attributs mémorisés pour les méthodes calculées**

```
Create Table Rectangle (
        …
        surface number,
        perimetre number, …)
```

**OU vues**

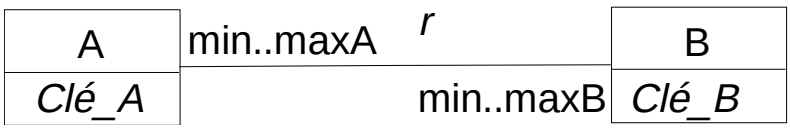
```
Create vue V_rectangle as
        Select
                Id_rectangle,
                argeur,
                hauteur
largeur*hauteur surface,
2*(largeur+hauteur) perimetre)
from Rectangle
```

**pour les autres méthodes, utiliser des mises à jour si possible**

```
Update Rectangle
                Set largeur=2*largeur, hauteur=2*hauteur
        Where num_rectangle= …
```

Si langage procédural alors il faut l'utiliser ! mais ne remplace pas les méthodes (héritage …)

# Association

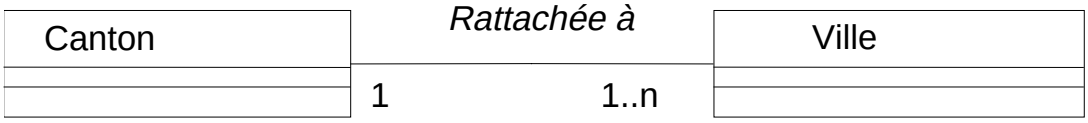



Objectif: mémoriser l'association entre A et B

Solution dépend de la cardinalité de r (maxA et maxB)

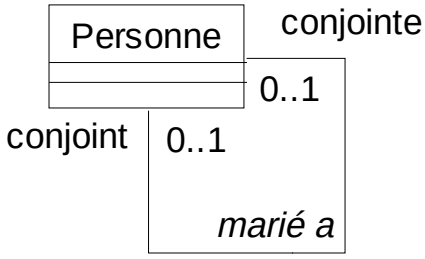
| MaxA / MaxB | 1 | >1 |
|---|---|---|
| 1 | - si la clé de A = la cléde B, ne rien faire<br>- sinon choisir une des autres solutions | Ajouter la clé de A dans la relation de B comme attribut |
| >1 | Ajouter la clé de A dans la relation de B comme attribut | Créer une relation r ayant comme attribut la clé de A et la clé de B |

# Exemple max=1




```
Create Table Canton(
        NomCanton varchar(20) primary key,
        Surface number
        …)

Create Table Ville(
        NomVille varchar(20) primary key,
        Rattache_A varchar(20) references Canton(NomCanton),
        Population number
        …)
```

# Exemple max=1

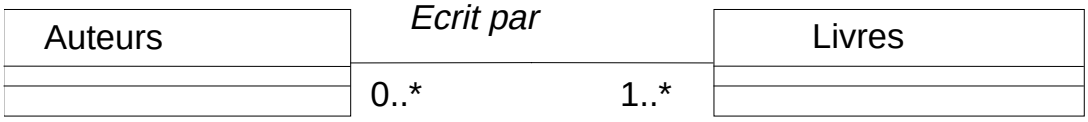



```
Create Table Personne(
    Id_personne number primary key,,
    Nom varchar(20),
    Prenom varchar(20),
    DateNaiss date,
    Conjoint number references Personne(Id_personne),
    …)
```

**problèmes ?**

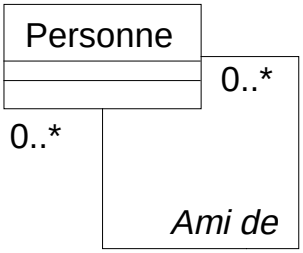
Maintenir la symétrie (redondance)

# association max>1




```
Create Table Auteur(
        Id_auteur number primary key,,
        Nom varchar(20),
        Prenom varchar(20),
        …)

Create Table Livre(
        ISBN number primary key,,
        Titre varchar(20),
        …)

Create Table Livre(
        Id_auteur number references Auteur,
        ISBN number references Livre,
```

# association max>1

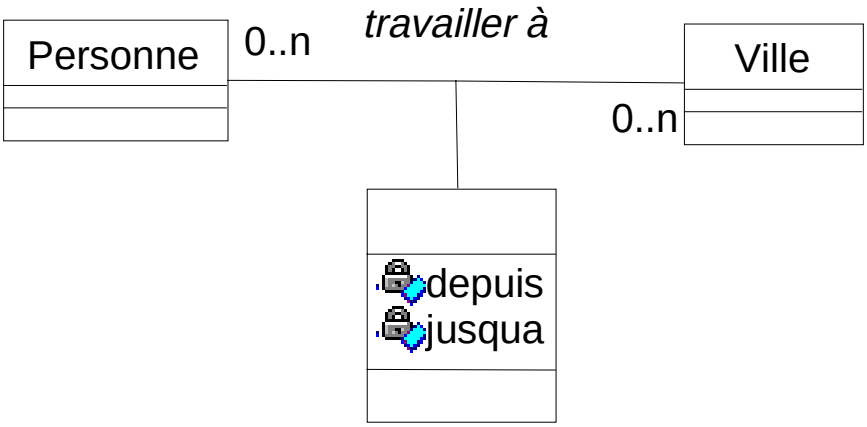



```
Create Table Personne(
        Id_personne number primary key,,
        Nom varchar(20),
        Prenom varchar(20),
        DateNaiss date,
        …)

Create Table Ami_de(
        Id_personne number,
        Id_ami references Personne(id_personne))
```

**problèmes ?**

Maintenir la symétrie (redondance) ?
Maintenir la transitivité (redondance) ?

# Association attribuée



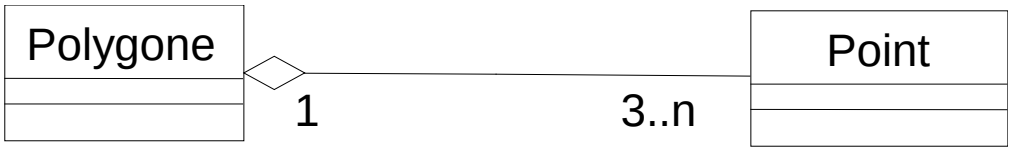

Solution: ajouter les attributs de l'association là où se trouve représentée l'association.

```
Create Table Personne(
        Id_personne number primary key,,
        Nom varchar(20),
        Prenom varchar(20),
        DateNaiss date,
        …)

Create Table Ville(
        NomVille varchar(20) primary key,
        Population number
        …)


Create Table Travail_a(
        Id_personne number references Personne,
        NomVille varchar(20) references Ville,
        Depuis date,
        Jusqua date)
```

# Agrégation et composition

Traiter comme une association

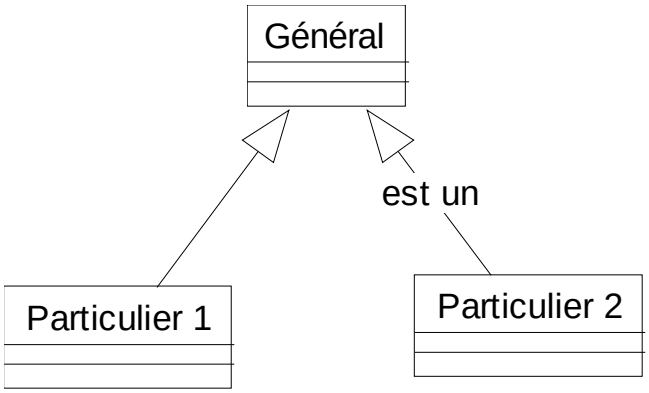
(la dépendance entre les classes est souvent une indication pour l'utilisation du delete cascade !)




```
Create Table Polygone(
        Id_polygone number primary key,
        Couleur varchar(20),
        …)

Create Table Point(
        Id_polygone number references Polygone,
        num_ordre number,
        x number,
        y number)
```

# Généralisation:TOUT dans UN

Solution 1: tout dans la même table et gestion des null

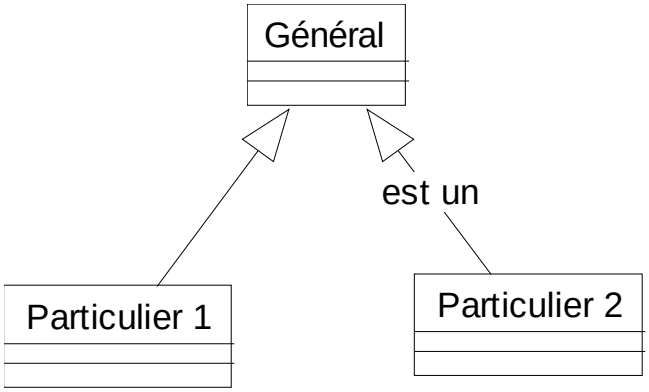



```
create table Général(
        id_général number primary key,
        attribut_de_général,
        …
        est_un_particulier1 Char(1), -- ‘N’=non, ‘O’=oui
        attribut_de_particulier1,
        …
        est_un_particulier2 Char(1), -- ‘N’=non, ‘O’=oui
        attribut_de_particulier2,
        …)
```

- Les attributs non utilisés sont laissés à null
- Créer une vue pour chaque classe pour retrouver exactement la bonne description

# Généralisation:CHACUN a sa place

Solution 2: chaque classe est une relation, on gère l'éclatement des informations

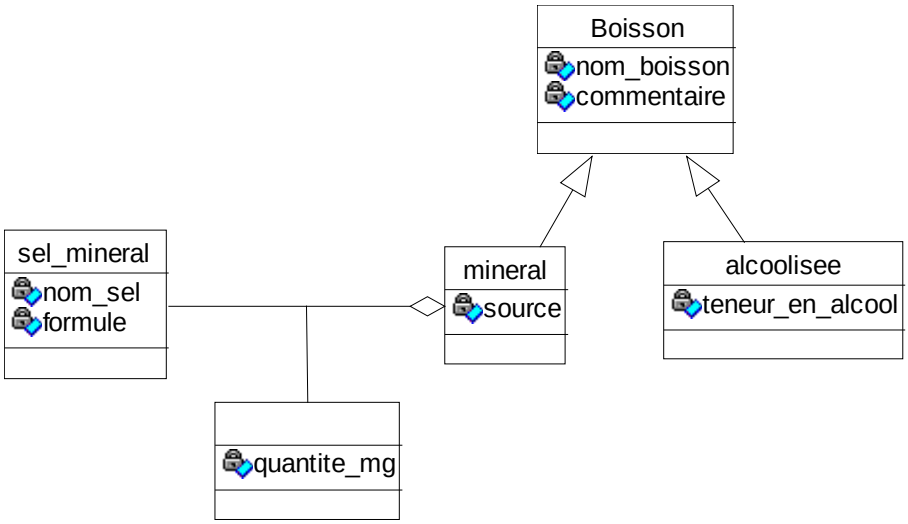



```
create table Général(
        id_général number primary key,
        attribut_de_général,
        …)
create table Particulier1(
        id_général number references Général,
        attribut_de_particulier2,
        …)
create table Particulier2(
        id_général number references Général,
        attribut_de_particulier2,
        …)
```

- Le type d'un objet est déterminé par sa présence dans une sous-relation (éventuellement ajouter un attribut dans Général)
- Créer une vue pour chaque classe pour retrouver exactement l'ensemble de la description

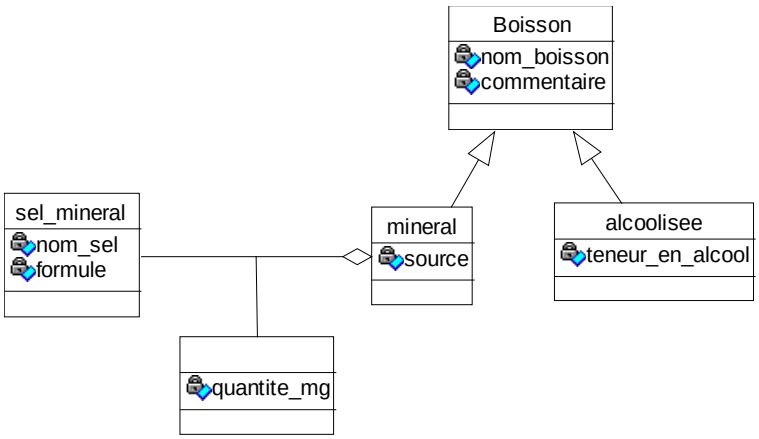
Plus difficile à gérer que la solution 1, mais plus précise

# Exemple: Solution 1




```
Create table Boisson(
        Nom_boisson varchar(20) primary key,
        Commentaire varchar(1000),
        Est_un_mineral char(1),
        Source varchar(20),
        Est_un_alcoolise char(1),
        Teneur_en_alcool number)

Create table Sel_mineral(
        Nom_sel varchar(20) primary key,
        Formule varchar(100))

Create table Composition(
        Nom_boisson varchar(20) references Boisson,
        Nom_sel varchar(20) references Sel_mineral,
        Quantite_mg number)

Exemple de vue:
Create view Mineral as
        Select Nom_boisson, Commentaire, Source
                From Boisson
                Where est_un_mineral ='O'
```

# Exemple: Solution 1




```
Create table Boisson(
      Nom_boisson varchar(20) primary key,
      Commentaire varchar(1000))
Create table Mineral(
      Nom_boisson varchar(20) primary key
                                    references Boisson,
      Source varchar(20))
Create table Alcoolise (
      Nom_boisson varchar(20) ) primary key
                                    references Boisson,
      Teneur_en_alcool number)
Create table Sel_mineral(
      Nom_sel varchar(20) primary key,
      Formule varchar(100))
Create table Composition(
      Nom_boisson varchar(20) references Mineral,
      Nom_sel varchar(20) references Sel_mineral,
      Quantite_mg number)
```

Exemple de vue:

```
Create view Mineral as
      Select m.Nom_boisson, b.Commentaire, m.Source
            From Boisson b, Mineral m
            Where m.nom_boisson=b.nom_boisson
```

# Exercice

Discuter les avantages et les inconvénients des solutions 1 et 2 pour modéliser la généralisation, pour les points

Performance,
Validation des contraintes,
Evolutivité,
…