**Transformation d’un diagramme de classes en un schéma relationnel**

**1.Conception, génération de code**

* Il est possible de traduire un diagramme de classe en modèle relationnel.
* Bien entendu, les méthodes des classes ne sont pas traduites.

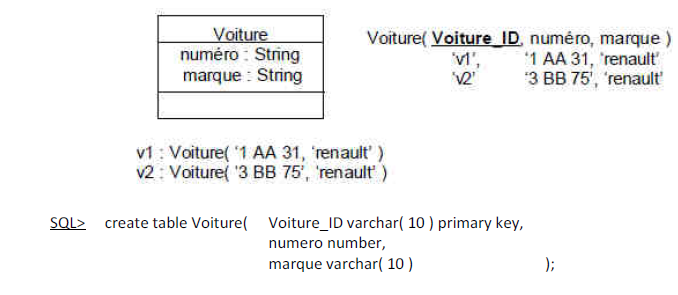
**2.Transformation d’une classe dans le modèle relationnel**

**Principe général de transformation**

* Les classes --> tables
* les associations | cardinalités --> clés étrangères

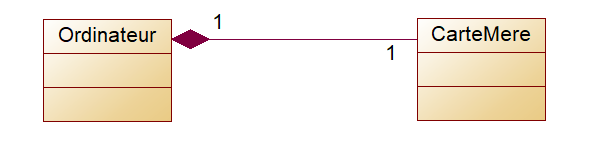
**Classe avec attributs**

* Chaque classe devient une relation.
* Les attributs de la classe deviennent des attributs de la relation.
* Si la classe possède un identifiant, il devient la clé primaire de la relation, sinon, il faut ajouter une clé primaire arbitraire.



**Association 1 vers 1**

* Pour représenter une association 1 vers 1 entre deux relations, la clé primaire de l'une des relations doit figurer comme clé étrangère dans l'autre relation.



* 2 classes = 2 tables ORDINATEURS (ID\_ORDINATEUR)

CARTE\_MERE (ID\_CARTE\_MERE)

* On déclare ORDINATEUR\_ID comme FK dans la table CARTE\_MERE qui va référencée la PK ID\_ORDINATEUR

**Association 1 vers plusieurs**

* Pour représenter une association 1 vers plusieurs, on procède comme pour une association 1 vers 1, excepté que c'est forcément la relation du côté plusieurs qui reçoit comme clé étrangère la clé primaire de la relation du côté 1.



* 2 classes = 2 tables CLIENTS (ID\_CLIENT)

COMMANDES (ID\_COMMANDE)

* Le côté \* est le côté porteur de la FK.
* On déclare CLIENT\_ID comme FK dans la table COMMANDE qui va référencée la PK ID\_CLIENT.

**Association plusieurs vers plusieurs**

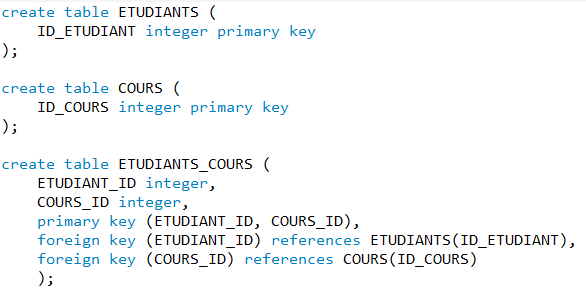
* Pour représenter une association du type plusieurs vers plusieurs, il faut introduire une nouvelle relation dont les attributs sont les clés primaires des relations en association et dont la clé primaire est la concaténation de ces deux attributs.
* La table d'association va permettre d'enregistrer les associations multiples entre les 2 tables.



* 2 classes = 2 tables ETUDIANTS (ID\_ETUDIANT)

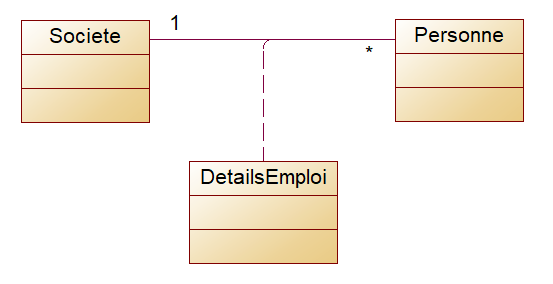
COURS (ID\_COURS)

* Pour gérer les relations (associations) multiples entre un étudiant et un cours, on ajoute la table d'association ETUDIANTS\_COURS:
* La PK de la table d'association la concaténation des clés ID\_ETUDIANT et ID\_COURS
* sa FK est les clés ETUDIANT\_ID et COURS\_ID référence des PK des 2 tables



**Classe-association plusieurs vers plusieurs**

* Le cas est proche de celui d'une association plusieurs vers plusieurs, les attributs de la classe-association étant ajoutés à la troisième relation qui représente, cette fois-ci, la classe-association elle-même.

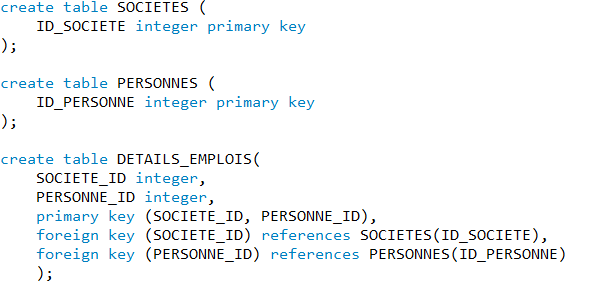


* 3 classes = 3 tables SOCIETES (ID\_SOCIETE)

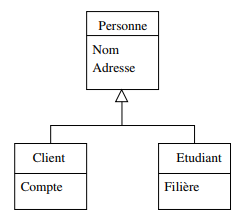
PERSONNE (ID\_PERSONNE)

DETAILS\_EMPLOIS (ID\_DETAILS\_EMPLOI)

* La table DETAILS\_EMPLOIS est la table d'association (jointure) .
* La FK de la table d'association la concaténation des clés ID\_SOCIETE et ID\_PERSONNE
* sa FK est les clés SOCIETE\_ID et PERSONNE\_ID référence des PK des 2 tables SOCIETES et PERSONNES.

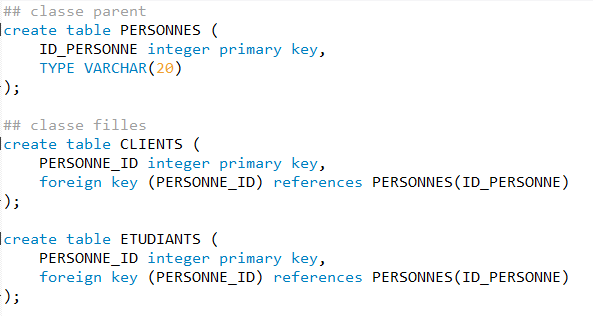


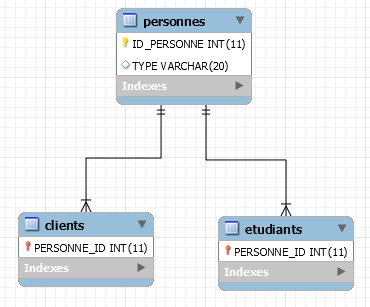
**Héritage**



Transformation de la relation d'héritage par référence (des filles vers la mère)

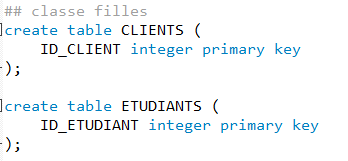
* Chaque classe, mère ou fille, est représentée par une relation.
* La clé primaire de la classe mère est utilisée pour identifier chacune de ses classes filles : cette clé étant pour chaque classe fille à la fois la clé primaire et une clé étrangère vers la classe mère.
* Ainsi, dans cette solution, un objet peut avoir ses attributs répartis dans plusieurs relations. Il faut donc opérer des jointures pour reconstituer un objet.
* L'attribut type de la relation correspondant à la classe parente doit indiquer quelles jointures faire.





Transformation de la relation d'héritage par les classes filles

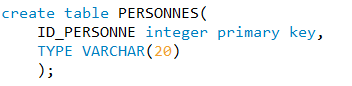
* Chaque classe fille est représentée par une relation, la classe mère n'est pas représentée (si elle est abstraite).
* Tous les attributs de la classe mère sont répétés au niveau de chaque classe fille.
* La clé primaire de la classe mère est utilisée pour identifier chacune de ses classes filles



Cette solution est adaptée dans le cas d'un héritage exclusif, c'est à dire si aucun objet d'une classe fille n'appartient aussi à une autre classe fille. Dans le cas contraire, le problème est que des redondances vont être introduites puisqu'un même tuple devra être répété pour chaque classe fille à laquelle il appartient.

Transformation de la relation d'héritage par la classe mère

* Seule la classe mère est représentée par une relation (ses classes filles ne sont pas représentées par des relations).
* Tous les attributs de chaque classe fille sont réintégrés au niveau de la classe mère.
* La clé primaire de la classe mère est utilisée pour identifier la relation.
* Un attribut supplémentaire de discrimination t (pour "type"), est ajouté à la classe mère, afin de distinguer les tuples. Cet attribut est de type énumération et a pour valeurs possibles les noms de la classe mère ou des différents classes filles.
* L'inconvénient de cette solution est qu'elle implique que les tuples contiennent de nombreuses valeurs nulles.



Si une classe fille a une clé primaire propre, cette clé sera réintégrée à la classe mère, au même titre qu'un autre attribut, mais elle n'officiera pas en tant que clé candidate car elle pourra contenir des valeurs nulles (elle sera néanmoins unique).